Automating Requirements Engineering in Game Development: An AI-Driven Approach to Player Frustration Analysis and Feedback Categorization

Jiacheng Xia
Computer Science, NYUAD
jx2467@nyu.edu

Advised by: Mohamad Kassab

ABSTRACT

Traditional requirements engineering practices are often not used in game development because of the subjective nature of player experience, and the pursuit of "fun" complicates structured feedback integration. There is currently no structured requirement engineering tool for player frustrations in the video game context. This capstone project presents an AI-driven tool that automates the analysis and categorization of player feedback using large language models and generates requirements, with a specific focus on identifying, interpreting, and predicting frustration. By using natural language processing and machine learning techniques, the tool transforms raw user feedback into actual design requirements, aiding developers in refining gameplay aesthetics, enhancing user experience, and addressing technical issues. The methodology combines interdisciplinary insights from human-computer interaction, psychology, and game studies. It develops a frustration taxonomy, and through mining and processing feedback data, predictive models are made. Ultimately, this work aims to bring greater structure and clarity to requirement engineering in digital games, making it more responsive, data-informed, and player-centric. Its effectiveness will be tested with validation from game industry peers or online communities to see if it aligns with industry experiences.

This report is submitted to NYUAD's capstone repository in fulfillment of NYUAD's Computer Science major graduation requirements.

جامعة نيويورك أبوظبي
NYU ABU DHABI

Capstone Seminar, Spring 2026, Abu Dhabi, UAE

KEYWORDS

human-computer interaction, NLP, requirement engineering, digital games

Reference Format:
Jiacheng Xia. 2026. Automating Requirements Engineering in Game Development: An AI-Driven Approach to Player Frustration Analysis and Feedback Categorization. In *NYUAD Capstone Seminar Reports, Spring 2026, Abu Dhabi, UAE.* 5 pages.

1 INTRODUCTION

Frustration, in general, is defined as the interruption or blocking of goal-directed behavior. Within this definition, there is a subset of frustration known as need thwarting. This type of frustration can lead to controlled motivation, that is, when people only react out of pressure, guilt, or external rewards; or lead to a lack of motivation. For the game context, this is undesirable in terms of design, as it is centered on the audience should like it.

Digital games are fundamentally different regarding requirement engineering compared to software that fulfills a certain function. For other types of software, there is a distinct difference between functional requirements and non-functional requirements. Game companies typically do not consciously apply common RE practices. Requirement engineering in typical software usually happens at the early stages of development, but for video games, the method of test and tune is usually applied, as the general idea is developed, and it is then tested to see if the targeted audience likes it or not. The feedback is often not officially documented, and the resulting product can vary enormously based on this feedback. Another characteristic is that game developers tend to minimize functional requirements, and that increases the importance of requirement gathering from user testing.

It is also difficult to apply traditional requirement engineering practices on games. It is common to generate tens of ideas initially but a few of them will be considered for detailed evaluation and implementation. Requirements can

also fluctuate a lot in later stages of testing and tuning. This further backs up the importance of the tool we are developing for this project, an AI-driven feedback analysis tool that uses large language models (LLMs) to automate the categorization of player feedback into structured themes, such as gameplay mechanics, user experience, and technical issues, while mapping frustrations to actionable development requirements. This tool can be used by game developers during user testing to brainstorm based on data fed to it and generate requirements on different aspects of the game based on the player feedback and their categories. It takes in what the players are frustrated by, and decides if something should be done about it, to make the game fun and engaging, which is the main focus of requirement engineering for games.

For challenges and the problems we expect to encounter, first, we would need to identify player frustrations and find a taxonomy to classify game-related frustrations based on multidisciplinary insights from game studies, human-computer interaction, and psychology. The framework should distinguish between detrimental frustrations hindering player experience and those contributing positively to engagement. It will also be challenging to apply this taxonomy into an AI tool to categorize and generate requirements, as the tool will need to map frustrations to actionable development requirements, interpret ambiguous language, prioritize recurring concerns based on frequency, assist designers in identifying critical areas for improvement, and address challenges in translating qualitative feedback into structured design insights. As a person who wants to be a game designer, this work is exciting as it will give me more insights into the game industry's playtesting realm, and this tool might come into use one day. It also proposes a new structured way of requirement engineering for games, which is what I believe to be currently lacking in the industry.

Research Questions

- What causes player frustrations?
- Can player frustration in games be categorized?
- Is there a correlation between certain types of player frustration and game aesthetics?
- Can we predict certain types of player frustration?
- How accurately can an LLM classify player frustration in free-form feedback?
- Which frustration categories best predict actionable design requirements?

2 RELATED WORK

2.1 RE in Game Design

A study involving interviews with 27 software professionals across seven organizations explored how requirements engineering (RE) is practiced in game development. The findings indicate that business practicalities and the pursuit of "fun" play a dominant role, often taking precedence over traditional RE concerns. Although game development teams use techniques similar to requirements engineering and management, they typically do not explicitly follow standard RE practices [2]. This aligns with my paper, which introduces the current context of how game companies are doing requirements engineering and provides a background for the tool I am developing to address the problems we have proposed.

2.2 Frustration Taxonomy

Psychologically, frustration happens when basic psychological needs are not satisfied. These needs are defined in a paper that introduces self-determination theory [4], based on the motivation theory, which has a lot to do with game design, as a commonly used definition of the player type is based on intrinsic and extrinsic motivation [3]. This fits in the paper as a theoretical psychological background of how frustrations occur and can help us identify and find a taxonomy to classify game-related frustrations. Another paper discusses how frustration can be used to design adaptive video games and research into its detection and measurement [1]. This paper can help with the categorization of frustration in video games and be seen as a form of frustration that contributes positively to game development, and thus be seen as positive feedback from the tool developed.

2.3 NLP for Game Feedback

Another paper has given an overview of techniques that can be used in game studies research, using these techniques to analyze game reviews [5]. In the three aspects this paper discusses, the most relevant ones are gameplay aesthetics from game reviews and sentiment analysis of game reviews. It provides pathways to sources that discuss understanding of the connections between game design elements and emotional patterns, and identifies aesthetic elements of games, which can be used for categorization of playtesting feedback. It also provides a method for sentiment analysis of games, which can help identify if the game review is positive or negative. This is highly related to the tool because the tool generates requirements based on player feedback. In short, this paper can serve as a pathway to many previous studies on this topic, and these studies will serve as the foundation of this research.

3 METHODOLOGY

To investigate this issue and fulfill the research goals, we plan to have separate stages in the development.

3.1 Phase One: Literature Review, Scope Definition, and Planning

In this phase, we will dive into related sources, gain insights into other related works, and identify the specific research questions and hypothesis.

3.1.1 Literature Review. We will research previous related works on the topic of requirement engineering in game development, literature that discusses how requirements in various game aesthetics that influence player experience, both positively and negatively, and explore empirical research examining how different types of software requirements, such as functional versus non-functional, or user interface versus gameplay features, correlate with end-user responses.

3.1.2 Define Research Questions and Hypotheses. After gathering enough background on this topic, we would define our specific research questions we would answer and provide topic topic-related hypothesis to answer and find evidence in the paper.

3.1.3 Select Target Repositories. Then we would look for repositories that have publicly accessible game testing and feedback data and requirement data, for example, Godot Engine.

3.1.4 Deliverables. Deliverables for this phase will be a proposal describing objectives, listing research questions, selected repositories, and planned methodology, and a literature review summary that highlights existing research gaps and establishes the foundation for our research questions.

3.2 Phase Two: Data Collection and Preprocessing

For this phase, we need to identify and extract requirement-related data from the game projects selected in the last phase.

3.2.1 Mine Requirements Artifacts. These data focus on the actual content of the requirements, and data sources may include commit messages, design documents, and issue tracker items. We would also need to mine player feedback data.

3.2.2 Mine Player Feedback. We would use APIs (e.g., GitHub Issue APIs) to find frustrated player feedback by filtering using keywords like frustrating, not fun, difficult, etc. Then we would extract metadata such as timestamps, sentiment, and connection to specific features. We will collect 500 anonymized Steam review comments containing these keywords and manually label 100 of them.

3.2.3 Data Cleaning and Annotation. Lastly, in this phase, we will do data cleaning and annotation. We would use Python with tool libraries like Pandas to clean and normalize the data retrieved in the previous two steps. We would then organize requirements by categories and finally create a mapping that links specific requirement artifacts with the subsequent feedback period.

3.2.4 Deliverables. At the end of this phase, we will have a dataset that consists of two tables: one for requirement artifacts and one for player feedback. We would also make scripts that do data extraction and preprocessing, and a data summary report describing the datasets, cleaning steps, and initial observations.

3.3 Phase Three: Exploratory Data Analysis and Feature Engineering

3.3.1 Exploratory Data Analysis (EDA). In this phase, first, we will do Exploratory Data Analysis. We would visualize the frequency and timings of various requirement categories across the project timeline, and plot trends in requirement changes with the changes in player sentiment. Then we would compare different requirement categories.

3.3.2 Textual Analysis and Categorization. We would use topic modelling to explore latent topics in the requirement documents, and we would validate such results with manual categorizations to refine this classification process.

3.3.3 Feature Engineering for Predictive Analysis. Then we would apply feature engineering for predictive analysis. We would create features from requirement artifacts, such as category tags, content complexity, and frequency of changes. Then we would create features from player feedback, such as frustration score and feedback volume. They will be combined into a consolidated dataset for correlation analysis and prediction.

3.3.4 Deliverables. At the end of this phase, we will produce a comprehensive EDA report with visuals and initial findings, a documented feature engineering script that translates raw data into a structured dataset, and an updated dataset with all derived features, along with a code repository and documentation.

3.4 Phase Four: Predictive Modeling and Correlation Analysis

3.4.1 Correlation Analysis. In this phase, we will determine if there is a significant correlation between certain types of requirements and measures of player frustration using statistical tests, and we will test out the hypothesis brought out in the first phase. Predictive Modeling We would also do predictive modeling, where we formulate a supervised learning problem, for example predict if a specific requirement artifact will bring measures of player high frustration. We will split the engineered datasets into training and test portions, and experiment with baseline classifiers or regressors. At last,

we will use cross-validation to tune the model. At the end of this phase, we will have a predictive model with complete, appropriately commented code, a model evaluation report with evaluation metrics, visualizations, and a discussion of the results. We would then have more insights into the relative effects on player frustration by various categories of requirements.

3.4.2 Model Evaluation. The final evaluations of the model will also be done in this phase. We will use appropriate evaluation metrics based on the type of task. For classification problems, we could apply metrics such as accuracy, precision, recall, and F1-score. We aim for greater than or equal to 0.8 recall on frustration detection and greater than or equal to 0.75 F1 on requirement-generation accuracy. For potential regression tasks, we could use Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE). Additionally, we would analyze model errors to gain insights into which categories of requirements tend to produce unpredictable outcomes.

3.4.3 Deliverables. At the end of this stage, we will produce a predictive model implementation with complete, commented code and a model evaluation report that includes evaluation metrics, visualizations, and a discussion of the results. We will also have more insights into the relative influence of various requirement categories on player frustration.

3.5 Phase Five: Recommendations, Validation, and Reporting

The last phase we will focus on recommendations, validations, and reporting.

3.5.1 Derive Actionable Recommendations. First, we would need to derive actionable recommendations, compile guidelines for game designers and requirement engineers. We would also explain how changes or additions to certain types of requirements could reduce negative player feedback.

3.5.2 Expert/User Validation. We would also seek validation from game development peers or online communities, and we will conduct small surveys or interviews to validate if the recommendations align with industry experiences.

3.5.3 Deliverables. At last, we will put together a final report and summarize the work with a presentation. Apart from that, at the end of that phase, we will have all the source code and datasets put together in a well-organized repository for game developers to use and build on.

4 EVALUATION

We will evaluate the model from two perspectives. One evaluation stage will be in phase 4, where we will base on the task to apply appropriate evaluation metrics. We will also evaluate errors to see if there are categories of requirements that tend to cause unpredictable outcomes. The other evaluation stage will be on phase 5, where we would seek validation from game industry peers or online communities, and conduct interviews or surveys to see if it aligns with industry experiences.

5 PROJECT TIMELINE

Phase	Start Date	End Date	Duration
Phase 1	May 19, 2025	June 9, 2025	3 weeks
Phase 2	June 10, 2025	July 28, 2025	7 weeks
Phase 3	July 29, 2025	September 7, 2025	6 weeks
Phase 4	September 8, 2025	October 19, 2025	7 weeks
Phase 5	October 20, 2025	November 23, 2025	5 weeks

6 BUDGET

Category	Item	Estimated Cost (USD)
Tools & Software	Cloud Services	$150
	API Access	$100
	Python Libraries / Packages	$50
	LLM Access	$300
Research Materials	Survey Incentives	$100
Miscellaneous	Presentation Tools / Report Printing	$50
	Contingency Fund	$100
	Total	**$850**

6.1 Justification

- Cloud Services: Estimated for hosting models and data processing on AWS or GCP for one month.
- API Access: Estimated for premium access to GitHub/issues/sentiment analysis APIs, which include several thousand requests per month.
- Python Libraries / Packages: Estimated for optional paid Python libraries or packages to enhance project functionality, assuming a few licenses for specialized tools
- LLM Access: Estimated for 5k calls to OpenAI or similar APIs at $0.06 per call for feedback analysis on project data.

- Survey Incentives: Estimated for gift cards or other rewards to incentivize 50 survey/interview participants at $2 per participant.
- Presentation Tools / Report Printing: Estimated for visual aids, slides, and printing materials for the final presentation of the project.
- Contingency Fund: Estimated as a buffer for unexpected expenses, approximately 10% of the total budget.

7 RISKS & MITIGATION

- Data sparsity: Player comments may not explicitly mention 'frustration', mitigated by expanding the set of keywords or using sentiment thresholds.
- API limits: LLM calls may exceed free-tier → plan for budget or local inference.
- Development data scarcity: developers may not release enough related to requirements → analyze resultant updates to deduce requirements

8 CONCLUSION

This project introduces a novel approach to requirements engineering in game development by integrating an AI-based feedback analysis with a frustration-centered design philosophy. We developed and evaluated a tool that is capable of mapping unstructured player feedback to meaningful requirements that can be used by game designers and developers through a phased methodology. The findings highlight how frustration patterns can help requirement prioritization, enabling better game aesthetics. By connecting player sentiment and developer action, this work provides both a practical tool and a theoretical framework for the future gaming industry. As the industry is focusing mainly on user-centered design, this approach proposes a more dynamic and empathetic model of game requirement engineering.

REFERENCES

[1] Kiel Gilleade and Alan Dix. 2004. Using frustration in the design of adaptive videogames. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*. 228–232. https://doi.org/10.1145/1067343.1067372

[2] Juha Kasurinen, Andrey Maglyas, and Kari Smolander. 2014. Is Requirements Engineering Useless in Game Development? In *Requirements Engineering: Foundation for Software Quality*, Camille Salinesi and Inge van de Weerd (Eds.). Lecture Notes in Computer Science, Vol. 8396. Springer International Publishing, Cham, 1–16. https://doi.org/10.1007/978-3-319-05843-6_1

[3] Andrzej Marczewski. 2015. *Even Ninja Monkeys Like to Play: Gamification, Game Thinking and Motivational Design* (1st ed.). CreateSpace Independent Publishing Platform. 65–80 pages.

[4] Richard M. Ryan and Edward L. Deci. 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist* 55, 1 (2000), 68–78. https://doi.org/10.1037/0003-066X.55.1.68

[5] José P. Zagal, Noriko Tomuro, and Andriy Shepitsen. 2012. Natural Language Processing in Game Studies Research: An Overview. *Simulation & Gaming* 43, 3 (2012), 356–373. https://doi.org/10.1177/1046878111422560 arXiv:https://doi.org/10.1177/1046878111422560