

**GameSense**

**Prepared By:**

**Osama Zumari / 21110167**

**Abdallah Obaid / 21110383**

**Noor Zuai’ter / 21110047**

**Supervised By:**

**Dr. Raneem Qaddoura**

Project Submitted in partial fulfillment for the degree of Bachelor of Science in Cyber Security

Spring-2025

**Table of Contents**

[1. Application Description 3](#_Toc195437223)

[1. Brief overview about the application 3](#_Toc195437224)

[2. Produce detailed diagrams or models with all components/elements labeled. (M1) 3](#_Toc195437225)

[3. Describe the different stages involved in the project implementation. (P2) 3](#_Toc195437226)

[2. Test Plan 3](#_Toc195437227)

[1. Explain the testing procedures and validation plans for the project/application. (M2) 3](#_Toc195437228)

[2. Analyze the implementation results and collected data to optimize performance. (D1) 4](#_Toc195437229)

[3. Reflection 4](#_Toc195437230)

[1. Evaluate the project/application performance based on the required objectives, specifications, and implementation outcomes. (D2) 4](#_Toc195437231)

4. Technical Aspects…………………………………………………………………………………

[5. Files to be submitted 4](#_Toc195437232)

[1. Implement the selected project using appropriate computing tools and methodologies (e.g., software development, simulation, and/or experimental). (P1) 4](#_Toc195437233)

[2. Discuss the process and results of your project through a professional presentation. (P4) 5](#_Toc195437234)

[References 5](#_Toc195437235)

# Application Description

## Brief overview about the application

The video game industry generates a huge amount of feedback, creates an overwhelming volume of information for games. Often, 5% of this feedback is considered when analyzing a single game’s reviews and players experiences, which leaves a huge number of valuable insights neglected. In addition to the lack of specificity in feedback makes a challenging experience for players to find specific information about video games in different aspects. Therefore, our system came to address and solve these issues by:

1. Help players to see most of feedback in a very summarized way ensuring no reviews are neglected.
2. Provide specific insights into different games aspects.
3. Save time and ease purchasing decisions due to rich information from other players experiences.

The application will function as a search engine that allow players to search for games and display the feedback of the searched game in a summarized way showing the positive and negative sides of different aspects of it and the number of agreements of players. In addition, an analyzing feature for business accounts that allows them to upload their own dataset to give them summarized results of their video games.

The system will provide summaries on the following classes: Gameplay, Visuals, Story, Price and Bugs.

## 2. Produce detailed diagrams or models with all components/elements labeled.

This section will outline the components of the system and how each component led to ensure proper functionality of the system:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Description | Input | Output |
| Class Assigner Model | A BERT large language model that trained in detecting whether specific classes are mentioned within the review or not. | Textual Reviews | A binary classification for multi classes, assigns 1 if a class is mentioned within a review, otherwise, assigns 0. |
| Sentiment Analyzer Model | A BERT large language model that trained in classifying the sentiment behind the review whether it is positive or negative. | Textual Reviews | Binary classification for multi classes, it passes through all mentioned classes and provides the sentiment for each mentioned class. Then an overall sentiment for the whole review. |
| Graphical User-Interface | A friendly user interface for users interactions and ease of using the system. | None | Choose one of the two features, Explorer, for searching a game and getting it’s summarization. And the Analyzer, for uploading textual reviews dataset for applying the summarization. |

Here is an example of how both large language models will perform:

Review: “The game is not good as I expected, but to be fair the graphics and story are the strength points for this game, but as for the gameplay? it is bad”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Gameplay | Graphics | Story | Price | Bugs | Overall Sentiment |
| Class Assigner | 1 | 1 | 1 | 0 | 0 | - |
| Sentiment Analyzer | Negative | Positive | Positive | No Vote | No Vote | Negative |

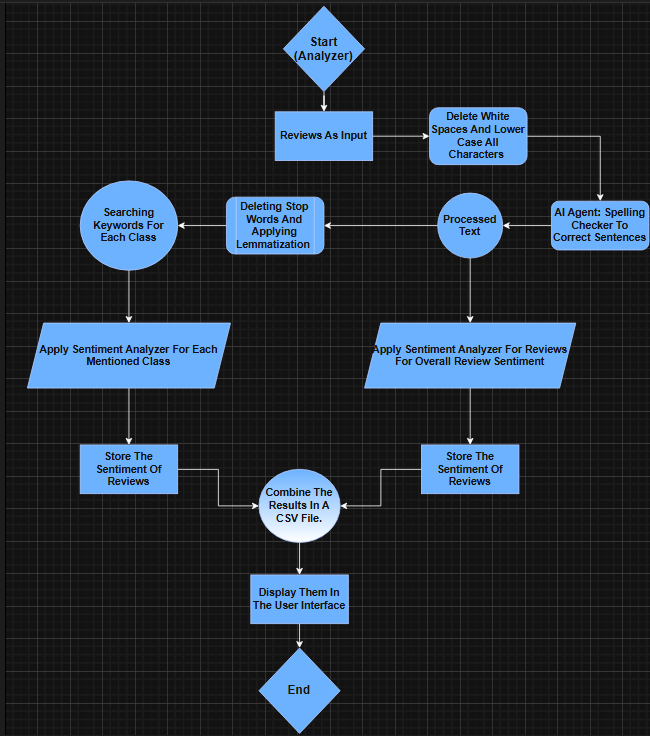
To apply sentiment analysis for each class accurately, we applied Context Window Trick which we call “Sentiment Scope Analysis”, it’s a technique that isolates the relevant parts of a sentence to focus only on the sentiment associated with a specific class. The process is done as follows:

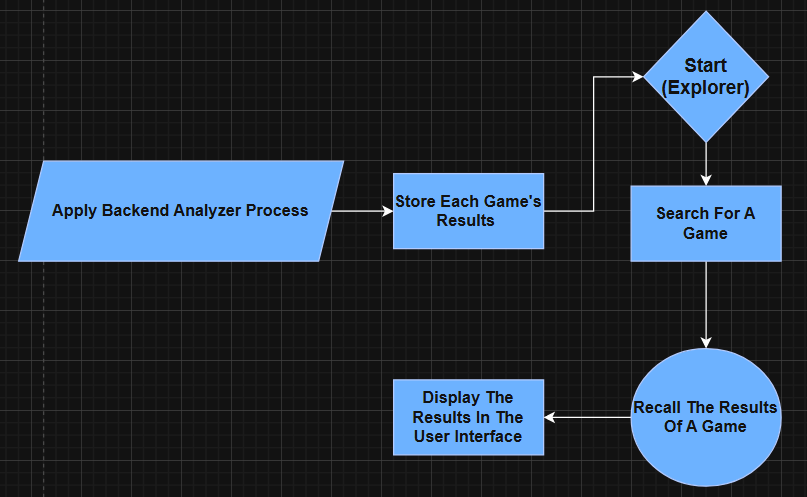
1. With the aid of the Class Assigner Model, we extracted a list of keywords for each class that the model relies on to determine if a class is mentioned in a review.
2. For each keyword written in a review, we applied the context window that takes 2 words before and after the keyword to ensure that only the relevant parts of the review are analyzed for the sentiment of that class. Then iterating this process for all classes that their keywords are mentioned within a review.

Furthermore, for a robust and accurate sentiment analyzation, we made a looping technique to take the sentiment with 2, 3 and 4 words before and after the keyword. Then, take the majority vote between them. This method ensures that the isolated parts are only for a specific class, taking into consideration the exaggeration words such as “I absolutely truly loved the graphics” where absolutely and truly are exaggerated words.

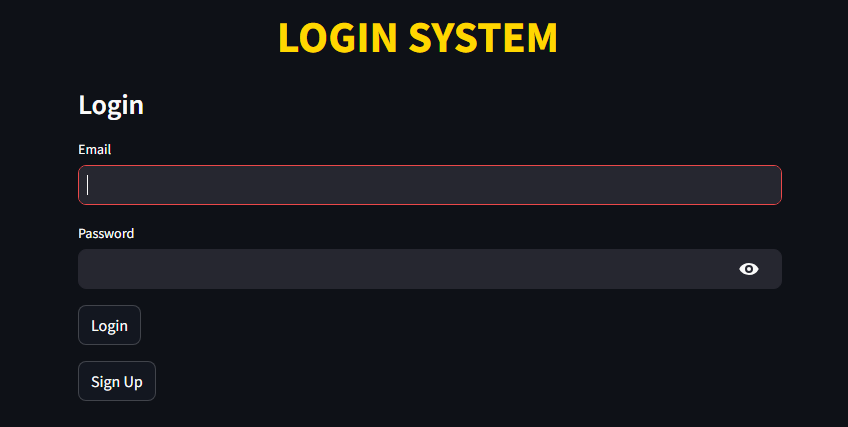
In the system workflow for the Analyzer feature, the Sentiment Analyzer Model will work behind the scenes and displaying the results in the graphical user interface. For the Explorer feature, the video games feedback summarization is already processed and stored in files that will be displayed inside the graphical user interface when a player searches for a game.

In the workflow of the system, the Class Assigner Model is not within the procedure since its aid was only to extract the keywords of each class. This helped the system to be efficient in terms of performance and processing time and resources since there is no need to call another large language model.

In the Analyzer feature, it starts by asking the user to upload a CSV file that contains textual reviews column only. The data will be passed to processing pipeline which is deleting white spaces and lower case all characters. Then, they are passed to the AI Agent that correct the spelling of the reviews with no misspellings in them. Then, the processed text goes to the Sentiment Analyzer for overall review to get a general idea on how many reviews were good or bad in that game. Then, it stores the results inside a csv file. Also, the processed texts will be passed to another processing for Context Window Analysis (Sentiment Scope Analysis) that will classify the sentiment for each class mentioned within a review. This is important to only capture the related useful parts of sentences for classes. Then, the whole keywords list for all classes will be searched through each review, if the keyword exists, it will apply the context window with 2, 3 and 4 windows to get the majority vote whether the class mentioned is positively or negatively reviewed. Then, it stores the results for both processes to display them in a structured format inside the graphical user interface.



In the Explorer feature, it starts by asking the user to search up for a game, when finding the wanted game, it shows the results of preprocessed data that shows the summary of the feedback for that game. These preprocessed data were applied by the Sentiment Analyzer model and stored inside a CSV file that will be recalled when a user searches up for a specific game, where it was applied on a sample of games feedback data to make the system rich with games for players to search for.

The third component is the Interface: 

it starts with asking the user to sign in or sign up, if the user has no account, sign up is the option to pick to display this page:

A screenshot of a login screen

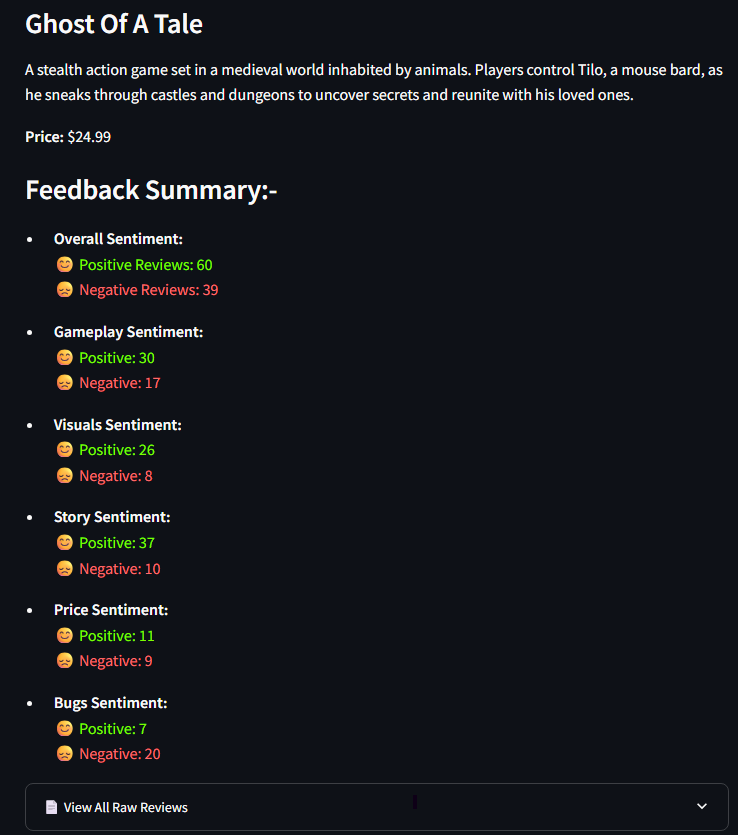
AI-generated content may be incorrect.

First, it asks the user the type of account, whether for organizations which have access to Explorer and Analyzer feature (which will be paid subscription later), or Gamer/Normal User which has access to Explorer feature only. Then typing the email and password.

A screen shot of a computer

AI-generated content may be incorrect.

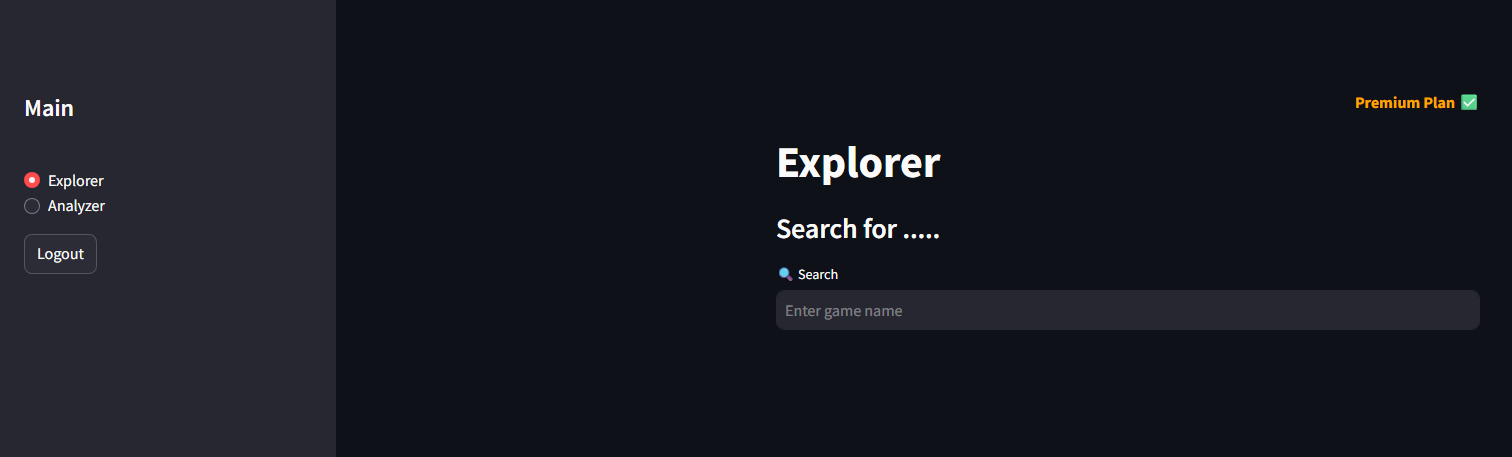
Once logged in as normal user, the Explorer page will appear only, asking the user to write a game name. For now, the system has 10 games only but is able to add as many as needed.

Then, when the user searches for a game, it will display the game description, Price and the processed feedback that the backend has resulted in. For example,

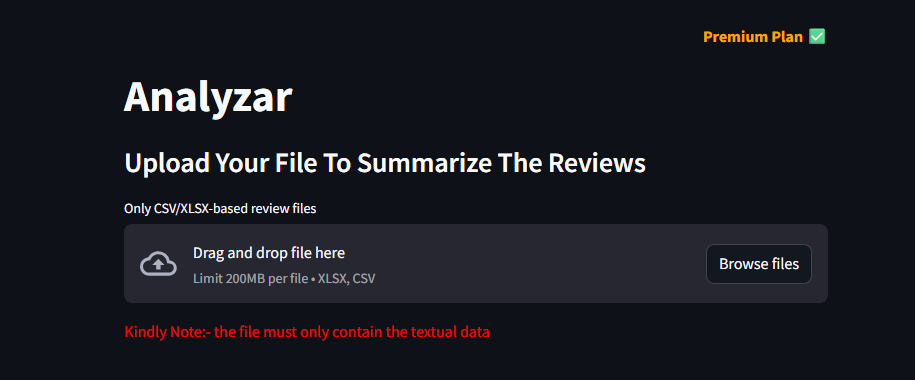
The game “Ghost of a tale”, it has in overall sentiment reviews of 60 positives and 39 negatives. Then, the game aspects are as follows: in gameplay, 30 positives and 17 negatives. And so on…

Down in the page, there is an expand button for viewing all the reviews before our processing to show the differences on how overwhelming it is to keep reading millions of reviews and yet face difficulties on finding specific reviews for specific aspects that you are looking for.

And for the organizational accounts:

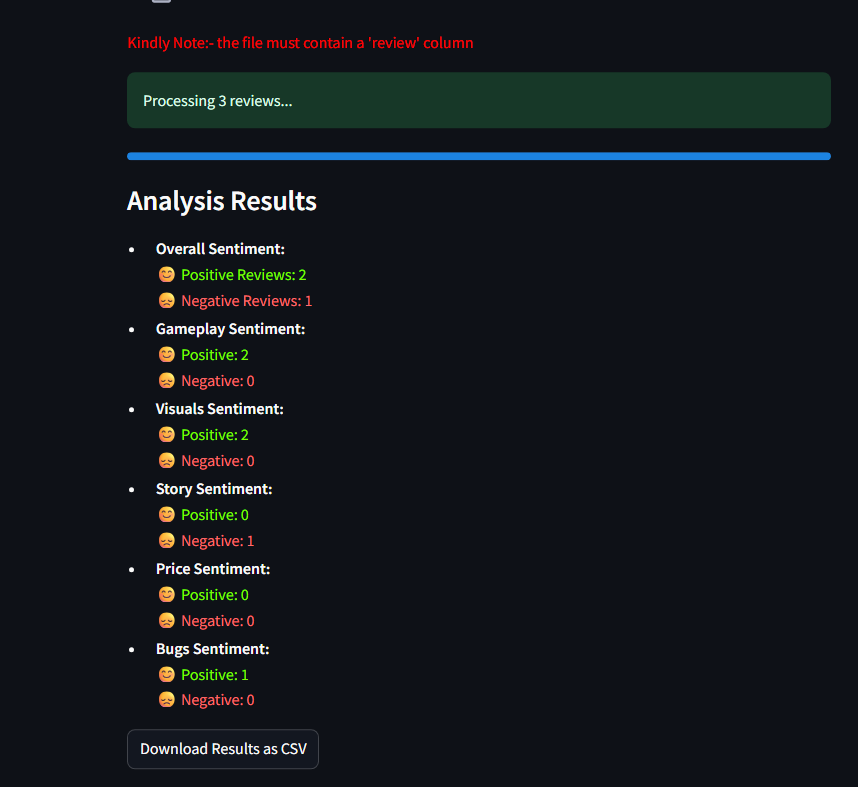


It shows a navigation bar on the left side of the page to select one of the features, automatically it starts with the explorer. It serves the same purpose as in normal users accounts.



The analyzer will accept only files of excel and csv types, which must contain only one column which is the textual reviews. This would help organizations to understand the reviews of their games.

When uploading the file, it will trigger the backend of processing the reviews to display the results as in the Explorer feature with a downloadable button for the summary:



## 3. Describe the different stages involved in the project implementation.

**Data collection and analysis:**

After the brainstorming sessions to identify the main problem of feedback and creating a proposal on how to build a system to solve these problems, we needed datasets that contain video games reviews from players and the sentiments behind these reviews. In a nutshell, two columns needed only, one is a textual review, and the other one is the sentiment behind that review. We collected datasets from different sources; they were mainly from Kaggle since it provided what is needed for the system.

Here are the most relied on datasets sources:

1. <https://www.kaggle.com/datasets/najzeko/steam-reviews-2021>
2. <https://www.kaggle.com/datasets/piyushagni5/sentiment-analysis-for-steam-reviews?select=train.csv>
3. https://www.kaggle.com/datasets/arashnic/game-review-dataset
4. <https://cseweb.ucsd.edu/~jmcauley/datasets.html#amazon_reviews>

Since these datasets combined are equal to millions of records, We scarped sample of them to get high, diverse data quality. Throughout our scraping journey, we found out that some of the data instances are miss labeled, which means that the sentiment does not match the review written. That’s why we did manual checking on the sentiment column to ensure that the labels are correct to get high quality of data, taking into consideration the slang language of players because in video game language, a sentence like “This game is sick” can be considered as positive sentiment despite the negativity of the word “sick”. These were the two phases of data, collecting and processing them.

**Suitable Model Selection, Processing, and Testing:**

Next, we searched for a large language model that is great for sentiment analysis tasks, we wanted to work with Bidirectional Encoding Representation from Transformers, BERT model. But to ensure faithful throughout the journey and experimenting different models for testing and comparison between them, we also developed RoBERTa and DeBERTa models as transformers modern models in the artificial intelligence field, and traditional methods such as TF-IDF representations. Next, the modeling phase where we took the models one by one and conducting different combinations experiments in hyperparameters. Starting with the modern models (BERT, RoBERTa, DeBERTa), we changed the following parameters: epochs, learning rate, dropout, weight decay, batch size, optimizer and norm clipping.

From these experiments, we have discovered the following:

1. These models share almost the same performance while BERT was the best among them with an accuracy equal to 89 for sentiment analysis task while the other models shared similar results but with a higher running time. Thus, we used BERT model to be our Sentiment Analyzer Model.
2. A small number of epochs were better than high numbers to prevent overfitting, when we experienced a higher number of epochs, the models performances were not good at testing phases compared to the lower number of epochs.
3. Batch size of 32 was the most optimal, we tried different numbers such as 8, 12, 16, 32, 64, 86 and 128. Our lucky number was 32 since the model performed better in training and testing without overfitting unlike higher or lower numbers that whether made the model perform less better or get overfitted.
4. Norm Clipping was an essential parameter to be placed even with a value of 50, all models kept getting their gradients vanished (equal to 0), the only method that solved the problem is putting a clip value to prevent this issue from happening. We tested the models after that, and they gave good results.

These are the essential parameters that changed the modern models behavior. In the other hand for the traditional method (TF-IDF) representations, it required more preprocessing instead of raw text such as in BERT, we created a customized stop words list to be deleted and saved a list that must not be deleted from the text and applied lemmatization on the text, yet it performed poorly even when using machine learning algorithms such as XGBoost. In a nutshell, we decided to take the BERT model.

**Clustering:**

The next phase was clustering, in our proposal, we said that we will apply clustering on the reviews to group by the similar reviews to see what they are focusing on in terms of games aspects. We developed and tried about 10 clustering algorithms such as: DBSCAN, K-Means, Optics, Hierarchical Clustering of Agglomerative and Divisive. with taking into consideration all of their hyperparameters and distances similarity measurements. But unfortunately, the performances were too poor to be used, even with logical approaches such as deleting stop words and applying lemmatization to reduce complexity and vocabulary, the performances were not applicable for the task. Thus, we replaced clustering with classification, it required more labelling and changing in the data root as follows:

**Classifying Reviews To Tackle Poor Clustering Performance:**

First, the dataset that we had was reviews and sentiments. We created a new dataset that has the reviews column and added more columns that will be the video games aspects which are: Gameplay, Graphics, Story, Price, Bugs, with binary values whether 0 (which means the review did not mention the corresponding class) or 1 (which means the review mentioned the corresponding class). Then, we manually labelled all the data for all classes. And to ensure consistency between team members in data labelling phase, we followed a set of rules of standards on when to label 1 for each class and when not. For example, in Bugs class, we set a rule that anything relates to freezing or lack of performance, we label that review 1 for Bugs class. A sample from each other to ensure that we all are labelling correctly and consistently. This way, we ensured data quality to be well optimized. From that point, a BERT model was created and trained on this recent generated data producing a model called “Class Assigner” which assigns 1 to the class if it is mentioned in the review.

**Testing:**

Till that part, two models were initiated (Class Assigner and Sentiment Analyzer) ready for advanced testing phase. As for the testing phase of Class Assigner, samples of reviews have been extracted from the datasets that are new to the model, writing manual reviews by us, scraping some reviews from players in Steam platform, and AI generated reviews. We spent enough time to extract what the keywords that the model focuses on are for each class through training and also while we were labelling, we saw common keywords such as “visuals”, “graphics”, “art” that kept repeating in the Visuals class. Due to the extraction of these keywords, we could dispense the Class Assigner Model since we got what keywords needed to consider whether the class is mentioned or not. Then, a similar testing approach for the Sentiment Analyzer Model, by extracting samples of reviews from the datasets that are new to the model, writing more manual reviews considering the slang language, also with scraping reviews from Steam and using AI tools for reviews generation.

Throughout the testing phase, we noted some issues with the models that needed to be solved, one of them were the negation sentiment with slang language, such as “The game is not sick as I expected”, even though the BERT model is pretrained and have base knowledge on understanding the negative and positive terms, the negation with slang language made confusion for the model, which by then, we needed to inject more slang data for the model to enhance and give it the ability to handle players language perspective. After that, we still saw some issues that required processing, for example, the word “Boring” is defined as positive sentiment, we checked the dataset on each “bore” and its derivates appear, we saw that it is mentioned most of the time when the sentiment is negative, yet the model still give it higher value towards the positive sentiment. Additionally, the words must be in the right spelling so the models can give better accuracy, especially the Class Assigner which requires correct word spelling; Therefore, we did a new technique for post processing these tweaks, which is using API from OpenAI to use one of their decoders models (gpt-4o-mini) to create an AI agent that specializes in checking the words spellings. First, we bought tokens from OpenAI to use their API key. Then, we created a system prompt with a few shot examples so the model can understand what it needs to do. Inside the system prompt, we assigned a task for the agent to whenever it sees the word “bore” and its derivates, change it to “bad” term. This way, it could assign the sentence that says “The gameplay is boring” to negative. We simplified the system’s prompt to spend less tokens for cost efficiency. After these post processing, the models were enhanced and performed better than before.

We approximate the true accuracy of the Class Assigner Model by around 90 percent, since it may not be able to handle out of vocabulary words unless with more data injection and updating on the model, And around 90 percent for the Sentiment Analyzer.

**Context Window For Explicit Sentiment Capturing:**

Lastly for the backend procedures, we needed to apply the context window trick to take the sentiment for the classes individually. The reason behind this trick is because the model will only give the overall sentiment analysis of the review, that’s why we split the sentence to make isolate the parts that only relate to the corresponding class and add positive or negative sentiment to that class for that game. to enhance the window of words, we have created a customized stop words list to delete to keep the important ones that won’t change the meaning of the sentence, and applied lemmatization on the text. This way, if for example we have this sentence: “I think the gameplay is actually great!” and the window is 2, the context without these processing would be “think the gameplay is actually” which indicates to no sentiment at all and still ambiguous, but after these processing, the text would be “think gameplay actual great” which is clear and can indicates the true sentiment behind the sentence. Despite deleting the stop words, some of them are indeed important for the model to understand the semantic behind the processed review. This example shows how important some stop words are for: “the gameplay is bad, but the graphics is good”, the word “but” is listed in the stop words list which will be deleted. However, it is important for the semantic shifting, that’s why we conducted an experiment to see how the model will perform with or without keeping the word “but”:

|  |  |
| --- | --- |
| Before Processing: the gameplay is bad, but the graphics is good | |
| Without But: gameplay bad graphics good | Negative for both classes (Gameplay and Visuals). |
| With But: gameplay bad but graphics good | Negative for Gameplay class, and Positive for Visuals class. |

And for more robust accuracy, we applied context windows of 2, 3 and 4 to take the majority vote of each sentiment. However, Sometimes it mistaken the sentiment because of the format of the sentence and the slang language behind the players, we approximate Sentiment Analyzer model in the context window field about 80 percent despite that in training and validation phase it gave about 90 percent. Furthermore, for faithfulness and testing more methods that may outcome context window performance, we used a semantic shift classification, which is a large language model encoder task to detect where the semantic shifted within a review. Using the keywords list, we guided the Encoder large language model that analyzes the embeddings of the text to detect where the topic changes within a review, the model should be able to understand when the review finishes talking about a specific class and started talking about another class. However, the performance was poor even with bigger encoders large language model parameters from Hugging Face. Therefore, we decided to keep the context window trick (Sentiment Scope Analysis) as the method used for classifying the sentiment for each mentioned class.

**Interface Construction:**

For designing the frontend, we used python Streamlit package to design a simple user interface that has the necessary functionality of the system. we have planned for designing three main pages only, user credentials that consist of logging in and signing up, signing up information must include email, password and the type of user, whether normal players or organizations and the Explorer feature page and the Analyzer feature page. The reason behind the user type in the credentials is later on we would make the Analyzer feature as a paid service that will only be available for organizational accounts. In the Explorer, when a user searches for a game, the backend will recall a csv file that contains a summary of the listed games and their aspects sentiments, in addition to a csv file that contains the reviews collected for the games. Then, the Analyzer page, as said, we will only give this feature for organizational accounts, additionally to the Explore feature. Inside the Analyzer, we only connected the attached file with the backend processes to trigger the pipeline of processing the textual reviews data and insert it into the Sentiment Analyzer model that will give the overall sentiment and sentiment scope analysis for classes. We also ensured that the system is fully working by testing every button and function inside of it that we will talk about in the next section.

# 2. Test Plan

## Explain the testing procedures and validation plans for the project/application.

We proceeded on testing all the components separately, then combining them together to make the end-to-end system. These are the approaches for testing each component and phase of building the system:

***Class Assigner:***

after training the model, we scraped more new unseen data from the used dataset (the parts that were not used in the training), steam reviews from different games, our own reviews and AI generation tools. This dataset was for each class we have (Gameplay, Visuals, Story, Price, Bugs). For these classes, we determined almost all keywords that a model relied on to assign 1 for the class. Then, we expected that these keywords for each class would be the model baseline to assign 1. So, we tested these keywords, and the results were great since the keywords are well defined, we ensured that by testing more samples using sentences that contain these keywords in different positions within a sentence. Therefore, we confirmed that the lists of keywords are valid, and the model completely relies on. The expected results matched the actual results. This also added flexibility to manually add more keywords for better sentiment class analyzing.

Here is the table that shows samples of each class:

|  |  |  |
| --- | --- | --- |
| Class | Keywords | Sample |
| Gameplay | gameplay, controls, movement, mechanics, combat, shooting, driving,  jumping, pacing, balance, physics, interaction, difficulty, ai, playstyle,  play style, technique, techniques, cpu, difficult, attack, attack, system | “To be honest, I feel that the gameplay is repetitive, no creativity at all.” |
| Visuals | graphics, art, drawing, drawings, style,  animation, visual, visuals, design, designs, texture, render,  appearance, resolution, resolutions, color, environment | “I was literally surprised about the visuals, it was so smooth and great for humans eyes.” |
| Story | story, storyline, plot, narrative, writing, dialogue, script,  cutscene, cutscenes, character, characters, backstory,  development, lore, quest, quests, theme, themes,  protagonist, antagonist, emotion, emotional, twist, ending,  arc, worldbuilding, immersive, immersion, journey, setting,  endings, end, narrator, arcs | “what makes this game good is the unique story for each character.” |
| Price | price, priced, pricing, cheap, expensive, cost, costs, value, worth,  deal, deals, sale, sales, discount, discounted, money, pay, paid,  purchase, purchased, buy, bought, refund, refunded, dlc, microtransaction,  microtransactions, transaction, transactions, season pass, loot box, loot boxes,  in-game purchase, in-game purchases, in game, add-on, add-ons, add on, add ons, affordable, overpriced, underpriced,  wallet, wallet-friendly, pricing model, price tag, not worth, worth it, ripoff,  overcharge, expensive, free, free-to-play, freemium, premium, addons | “Not worth it” |
| Bugs | bug, bugs, glitch, glitches, crash, crashes, crashed, crashing,  lag, laggy, freeze, freezes, freezing, stutter, stuttering,  frame drop, frame rate, fps drop, low fps, optimization, unoptimized,  unstable, desync, broken, gamebreaking, game-breaking, issue, issues,  problem, problems, technical, update broke, patched, patch broke,  unplayable, error, errors, connection issue, server lag, network lag,  buggy, glitchy, softlock, hardlock, lag spikes, performance, load times,  loading bug, input lag, jank, debug, runtime error, black screen,  infinite loading, flickering, corrupt save, save bug, fix client, client,  stuck, stick | “The game keeps freezing I enjoyed nothing.” |

***Spell Checker****:*

It is an AI agent extracting API key from OpenAI to use their gpt model that specializes in correcting the misspellings in the reviews, the reason behind this is that the video game feedback usually has slang language and unformal written language which sometimes it is hard for the AI models to detect this unfamiliarity. We tested this component by writing multiple misspelled reviews as hard as possible to ensure that the agent can handle the reviews. Also, we defined the system role and added a few shots prompting which is a technique that guides the model on how to behave using input examples and how the output must be. This component actually helped the Sentiment Analyzer Model by increasing the confidence rate for the model’s prediction about 2 to 10 percent especially in the larger sentences that contain more slang words and difficulties on detecting the sentiment.

Here is a great example on how bad someone can type yet the GPT model will do the correction job:

|  |  |
| --- | --- |
| Sample | Correction |
| Original: the gme ply iss too badad i cldnt belive i pid tht much mney onnit | the gameplay is too bad I couldn't believe I paid that much money on it |

***Sentiment Analyzer:***

Similar to what we did in the Class assigner model testing procedures, we applied it as well for the Sentiment Analyzer, by scraping unseen data from different sources to ensure that the model is performing well and can detect the sentiment properly. We also ensured that the data scarped is a mix of reviews of formal or slang language to ensure faithfulness of the model and not underfitted with various types of written language.

Here are samples of testing reviews and the model’s prediction:

|  |  |  |  |
| --- | --- | --- | --- |
| Sample | Actual Sentiment | Predicted Sentiment | Review Type |
| I've been playing this game on and off for about 6 months now and have about 15 hours of gameplay in. It's a pretty cool game and fun to play. Suspenseful too! | Positive | Positive | Straightforward review praising the game |
| This is as close to Redwall: The Game as we're getting for the moment folks. Jokes aside, the atmosphere draws you in immediately and effortlessly. What Thief should have been. | Positive | Positive | Using Idioms and public phrases like. |
| An Absolute Gem. | Positive | Positive | Slang method to praise a game. |
| this game is only about stealth; the graphics are great but i wasn't having fun at all | Negative | Negative | Straightforward of not enjoying the game despite a positivity in the graphics. |
| What is that hype all about? | Negative | Negative | Slang method by indicating that the game is not worth it. |

***Sentiment Scope Analysis (Context Window Trick):***

As we saw from the testing samples in the Sentiment Analyzer, there is a review that is negative but praised the graphics. That’s the reason for the whole system, to take all reviews into consideration and give credits to the aspects of the games. For this component, we tested it by scraping data that talks about one of the classes (Gameplay, Visuals, Story, Price, Bugs) in different phrases. Also, changing the parameter of the Window Size and the number of Votes included to ensure getting sentiments as accurate as possible since the component will only take the class scope, neglecting the unrelated parts of the reviews.

Here are samples of testing reviews and the model’s prediction:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Reviews | Gameplay | Visuals | Story | Price | Bugs |
| Unplayably buggy. Avoid. | No Vote | No Vote | No Vote | No Vote | Negative |
| The story line seems great and engaging, but the game (despite it saying version 8.33) is tainted with bugs and the UI is terribly handled with wrong indications as a consequence of re-mapping keys. Guards pathfinding is awful, graphics will break if you play for too long and I could keep going...  The game is great, but in this state, it's not worth 25 bucks. | No Vote | Negative | Positive | Negative | Negative |
| this game is only about stealth, the graphics are great but i wasn't having fun at all | No Vote | Positive | No Vote | No Vote | No Vote |

As mentioned, we tested all the cases that made the model perform poorly in some edge cases such as: “gameplay is bad, but the graphics is good”. After processing, the word “but” is deleted and gave both negative semantic. Then we kept the “but” term and did not delete it from the stop words list, and made the model performed accurately by classifying gameplay as negative and visuals as positive. This case opened our eyes on keeping more stop words that must be kept for the model to understand the semantic shifting within reviews such as “Not, No, But”.

For the user interface of the 3 pages, we tested all edge cases that could happen inside the system as follows starting with the credentials page:

|  |  |  |
| --- | --- | --- |
| Testing | Expected | Result |
| Typing wrong or non-existing email and password in log in page. | Error and no signing In. | A screenshot of a login screen  AI-generated content may be incorrect. |
| Typing existing email in sign up page. | Pop up a message indicating that the email is already taken. |  |
| Signing Up normally | Successfully create an account and take the user back to log in page. | A screenshot of a login system  AI-generated content may be incorrect. |
| Signing In normally as normal user | Enter explorer immediately | A black rectangular object with a grey stripe  AI-generated content may be incorrect. |
| Search for a game in the explorer | Provide the summary of the game searched. | A screenshot of a video game  AI-generated content may be incorrect. |
| Expand reviews work in the explorer | Display the raw reviews. | A screenshot of a video game  AI-generated content may be incorrect. |
| Searching for another game after searching for one | Smoothly change the summary of the next game. |  |
| Logging out from the account | Log out to Sign in page. | A black screen with yellow text  AI-generated content may be incorrect. |
| Checking all explorer testing approaches but inside an organizational account | Working properly | All were working properly because explorer is written and standard for both Normal users and organizational users. |
| Testing the upload button if it is working in the analyzer | Ask for file to attach |  |
| Test the analyzer results after uploading. | Display the summary of reviews. |  |
| Test the download results as csv button | Download the summary file. |  |

## Analyze the implementation results and collected data to optimize performance.

After testing the components individually, we started to connect a pipeline that passes through all stages to make an end-to-end, from backend to frontend connected together system. There are still some enhancements that could be applied to each component starting with the frontend where it needs to be developed using specialized front end such as react and fastAPI for more advanced and secure system.

Explorer: A Search Engine, the backend behind it just recalls the games and its reviews when the user searches for it. The frontend is friendly and simple for users, just a search bar, when looking for a game, the reviews will be displayed. There were no problems with it and cannot improve more on it expect non-functional aspects such as better design and more information that is related to games such as hours to play, developers credit and famous ratings platform. These additions could add to the system more useful information and complete the missing parts of the it, especially for normal users.

Analyzer: we did not want to make it much complicated for organizations, since they only will care about the results, we thought that no need to add more visuals inside that page.

In general, for the interface, there were some issues with the buttons where two clicks were needed to perform the action, the problem was that we did not use streamlit function called rerun, which is a function that refreshes the page after performing an action immediately.

In the other hand, the components in the backend, starting with the Class Assigner model, in the beginning, we thought of using this model to assign the mentioned classes, then, send it to the sentiment analyzer. However, that was too time-consuming since we are calling two large language models, which affected the performance of the system. That’s why we have extracted the list of keywords for each class, to minimize large language model calling resources and using predefined lists to recall only that also provide the same performance as the large language model. In addition, the AI agent was beneficial in other case than checking spellings, the word “bore” and its derivates is mistaken in the sentiment for the model where it gives it a high value towards the positivity which is wrong, we have injected more data for fine-tuning the model but still it gave the same issue, that’s why we used the AI agent to whenever it sees the word bore and its derivatives, change it to “bad”. Overall, the AI agent has increased the model’s performance by about 2 to 10 percent after experimenting. We saw that by writing positive sentences with misspellings, the model without corrections gave an accuracy of 72%, after applying the agent, the accuracy increased to 81%, and so many cases were like that. Thus, we ensure that the AI agent increases the model performance with the impossibility of degrading it. Lastly, for the sentiment scope analysis, behind the scenes we trained another large language model on a processed dataset that is similar to the shape that the reviews change when entering the context window trick (which is deleting the customized stop words and lemmatization), the data set was like below:

|  |
| --- |
| Not enjoy gameplay bad |
| Story bore repeat |
| Believe how weak visuals disappoint not amaze |

We did that for a more specific model since what it would enter inside of it will be similar to the data above. However, not only were the resources high since calling another LLM is expensive, but also the performance of it was bad. That’s why we stick to our Sentiment Analyzer model that performed well even when changing the data shape. We focused more on improving the Sentiment Analyzer model, we have trained it multiple times whenever we found new weak points. For example, the model at first was still performing poorly for slang language, we injected more slang language data. Then, we found out that the negation phrases is a weak point as well, where the sentence “The game is not bad” classified as weak positive which almost falls under the negative sentiment. These injections have increased the model’s accuracy and performance became better than before. In addition to the multiple experiments we conducted for hyperparameters tuning already covered the rest of issues such as overfitting or vanishing gradients.

Also, we ensured that the logic approach (algorithm) for processing the data is correct, which starts with correcting misspellings, then deleting white spaces, and for the sentiment scope analysis (context window), deleting customized stop words and replacing some words with another and applying lemmatization on the text.

Overall, training a large language model is expensive and GPU is needed for that task especially when using a large dataset.

# 3. Reflection

## Evaluate the project/application performance based on the required objectives, specifications, and implementation outcomes.

We had a visual image in the beginning of building the system for each component and the logical workflow that we planned to walk through. At first, we were doing the clustering instead of classification, but the results were poor which lead to shifting the idea to a similar approach which is classification. Other than that, the image still remained the same. The first major model was the class assigner, it was expected to label 1 for each mentioned class, and 0 for not mentioned class, it was hard at the beginning because we could not know how the BERT model would perform for multi-classification, and on what basis it would understand the mentioned or not. However, the advantage is interpretability of the model, we could define a list of keywords which made the model explainable.

In a nutshell, we expected the output to be as it is right now in terms of functionality, we faced some challenges like what if the word is not written correctly and its part of the keywords list. Thus, we concluded that we needed a spell checking for that matter, we used Python library that is called (pyspellchecker), however, the performance was too poor to be used, and not directed on the video games field and was too slow to run which took too much time on a couple of sentences only where the upcoming datasets may contains thousands for a game and could reach to millions in total. Therefore, we decided to get the OpenAI API key to use their gpt model. Additionally, since we are using the list of keywords that the model guided us through, we are not using the model in the system. So, there is no accuracy of the model rather than the accuracy of the keywords, every method has it’s error and success rate, we approximate the accuracy of our approach about 85% to 90%, and the ability to increase is high since the approach is flexible but requires manual adjusting. The challenge in the future may require the model to enter the workflow especially if out of vocabulary words increased with the time or more idioms or slang that talk about a class but not written in the keywords list. However, even though this might be a concern, the advantage of the model is its flexibility, we could adjust the keywords as needed to match the desired outcome. This method saves time on the running processes since there is no need to call the large language model every time. Also, To align with policies and regulations about explainable AI, we ensured that our approach of extracting keywords for each class provides a transparent and interpretable of the model’s behavior since these keywords is what the model uses to assign classes, making its behavior justifiable.

Next, the sentiment analyzer model was straightforward approach, using a pretrained model which is BERT, and fine-tuning it for the specific task using our collected data. But it was not giving the required since it needed more data injections for different cases such as negation and slang which by then increased the accuracy of the model and its performance. Also, for the sentiment scope analysis (applying the context window) it gave a great performance at the end after the injections. The great thing about the model is it can be tuned and injected into more data in it which can be continuously learning on new datasets. The accuracy of sentiment analysis models depends on the positive and negative terms exists within sentences according to the context as well, but in the gaming field, it is not always enough since sometimes slang language such as “the game is sick” indicates to a positive sentiment. Which why we approximate that the accuracy maybe between 80% to 90% since the unpredictability of how players may write their reviews specifically in the sentiment scope analysis (context window) after the processing which reduce the number of words within a review that may end up with sentences like “gameplay good but bad”, this is a very rare case but the sentiment scope can be difficult on these cases. But with the aid of the AI agent on checking the spellings, it would eliminate this concern.

As for the whole system with the user interface, we have tested the system that users will interact with completely, taking all edge cases possible, the Explorer feature is performing well in terms of speed, there are no latencies in it. For bigger datasets and games, these reviews will be stored in a database and stronger computer units will be used for retrieving the data to display it in the system, which would make it even faster. Same for the analyzer, since it uses a large language model when uploading a dataset, it would be faster when using GPUs instead of local CPUs depending on the size of the dataset. Despite the need for GPUs, the Analyzer works perfectly fine with CPUs for small datasets. When we trained the Sentiment Analyzer model with a large dataset about 100k of records using CPU, it took about 18 hours to complete, but when GPU was used, it took about an hour only. In general, the functionality of the system is proper and as expected after solving the challenges that were considered as opportunities for improving the system. We could work more into system design by using other specialized frameworks such as REACT and FastAPI. Also, adding more information inside the Explorer to make the system completely useful for all players and must be downloaded for any player that has passion in games.

Overall, all models are highly flexible, and the keywords list can be adjusted to benefit the work case. Also, the AI agent can do even more tasks to outcome any challenges or edge cases that could not be solved from the root. Even the Sentiment Scope Analysis (Context window), it can be adjusted furthermore whether by determining the number of windows, applying majority vote as we did. Lastly, the Sentiment Analyzer model also can be fine-tuned even more to match the tasks needed to be done, which by that, it makes all the system’s components highly flexible, and the main idea of it can be applied in various sectors not only in video games industry.

In conclusion, the pipeline of the project started with 3 large language models, but ended up using 1 only, with all the benefits from the other two unused models to reduce time complexity without trading the performance. In addition, adding more techniques in the middle to ensure higher accuracy and better performance such as the context window trick, AI agent and preprocessing the text. Finishing all these components and connect them with a single pipeline and display it in a friendly user-interface that serves as the page that users will interact with. Ensuring that we tested every aspect of every single component, with taking into consideration any potential threat and edge cases.

# 4. Technical Aspects

In this section, we will talk about the technical aspects of the helping components that we used for the functionality of the system, by showing the benefits of them and their logic***:***

First, talking about how the text processing is applied, we will discuss how the logical approach as well has been applied to the text in terms which function to apply first:

First Processing: AI agent of checking spelling, at the beginning, all the text will be passed to the AI agent to correct the spelling of the reviews, which is important to be the first step since some reviews may have connected words without white spaces or misspelled such as: “Goodvisuals but the gmplay is bad”. In this particular example, the keywords list of each class would not detect the words of visuals and gameplay. This is the system prompt used for the agent to understand the task properly.

A screenshot of a computer program

AI-generated content may be incorrect.

Starting with defining the main role of it and providing a few shot examples that lead to increases in the performance of the agent. Then, important rules list for ensuring that the models handle the “boring” edge case and the connected words.

Second Processing: which is deleting the extra white spaces and lowering all characters



This function is applied among all reviews to match the same attributes and characteristics.

It was applied to all reviews that will be processed and analyzed. After the AI agent finishes the corrections, it will be passed to this function to ensure that all words have the same format in terms of lower characters and ensure that no white spaces are still in the reviews.

Third Processing: which is deleting stop words, numbers, special characters and applying lemmatization to VERB part of speech tags. In addition, replacing methods for words that has connected negations such as “wouldn’t”, “haven’t” and so on.

A screen shot of a computer screen

AI-generated content may be incorrect.The idea behind it is to keep “not” negation word to keep the sentiment in the reviews.

In addition, defining the needed keywords from the stop words list such as:

For the same purpose which is keeping the main and important sentiment shifting keywords.

This phase is only applied for the reviews inside the Sentiment Scope Analysis (Context Window) to reduce any unrelated and exaggeration words that indicate sentiments. Keeping in mind to keep the words that would change the sentiment and meaning of the reviews.

Fourth Processing: it is the last processing text that will be for the context window only to apply the Sentiment Scope Analysis model properly.

A computer screen shot of text

AI-generated content may be incorrect.

By defining the keywords of each class, it will apply the context window function by looking at each word in the given review, then detect if any word matches any word in the keywords list of that class. After that, when finding it, it would returns none if no keywords mentioned. And if found, the number of windows indicates to the number of how many words take before and after that keyword. Then return the isolated and relevant parts of the class.

These are the steps of applying processing to the textual reviews that are inputs for the models, starting with spellings corrections, then removing white spaces and formulating all text into lower cases. After that for the sentiment scope analysis only, the text will be passed to deleting stop words, numbers, replacing and keeping important words for to apply the context window trick.

In addition to the main components such as Class Assigner that extracted the keywords lists, and the Sentiment Analyzer model, we could complete the system with full functionality that was well-tested and ensured a proper outcome at the end. Also, the ability to design and develop components that are flexible and can be adjusted accordingly as needed.

# 5. Sources

Code Source:

Presentation Source: