Dalion, Adrian Lloyd L. 2022-03103
Salcedo, Chris Samuel O. 2022-05055
Suyman, Ann Junah L. 2022-09089

Suking Tindahan: A Mood-based Filipino Film Recommendation System

Introduction and Background

With the rapid growth of online streaming services, viewers are often overwhelmed by the sheer volume of content available. This abundance of choice can lead to *decision fatigue*, where users struggle to decide what to watch. In fact, a recent survey found that 41% of viewers have difficulty choosing a program from ever-growing streaming libraries. Such indecision frequently makes users rewatch familiar shows or feel unsatisfied with random picks.

The problem is compounded for Filipino audiences seeking local films. There is currently no single platform dedicated to personalized Filipino movie recommendations, making it hard to discover films that suit one's mood. While major platforms like Netflix host some Filipino titles, they do not account for a user's current mood in their recommendation algorithms, limiting the personalization of suggestions. Moreover, decades of classic Filipino movies have become *"lost in time,"* with only a few being picked up by various streaming providers. This fragmentation means Filipino cinephiles must manually search across multiple services and curated lists to find content that resonates with their emotional state.

Suking Tindahan is proposed to address these gaps. This system will serve as a centralized web platform to help users discover Filipino films tailored to how they feel at the moment. By analyzing the user's self-described mood (through natural language input) and leveraging a comprehensive movie database, the system can suggest films that not only align with the user's emotions but also highlight Filipino cinema's rich offerings. In essence, the platform combines sentiment analysis and recommendation systems techniques to bridge the disconnect between a viewer's current feelings and the movies available. The development of this system is justified by the need to enhance content discovery for Philippine cinema and to improve user satisfaction in the browsing experience. Movies are an essential part of people's lives, offering entertainment and a *"necessary disconnect from the outside world"* that can reduce stress and anxiety. Ensuring that viewers can easily find a film that matches their mood or lifts their spirits will make that therapeutic aspect of cinema more accessible. By focusing on Pinoy (Filipino) content, the system also supports local culture, addressing the lack of exposure and personalized curation that Filipino films experience in the shadow of Hollywood and other international content. In summary, the proposed system aims to provide a much-needed solution for Filipino movie enthusiasts who currently struggle to find the proper Filipino film at the right time and in the right mood.

Objectives / Purpose

The main objectives of the Pinoy Mood-Based Movie Recommendation System are as follows:

- **Deliver mood-matched Filipino movie suggestions:** Based on the user's current mood (determined via text input), the system will suggest Filipino movies that reflect or complement that emotional state. Personalization may also incorporate the user's stated preferences, such as favorite actors or genres, to further refine recommendations.
- **Help users discover hidden gems and classics:** The system will expose users to *"hidden gems"* of Philippine cinema – including indie films or critically acclaimed classics – that they might not find on mainstream streaming recommendations. This broadens the user's viewing experience and helps preserve awareness of classic Pinoy films.
- **Provide rich movie information:** For each recommended film, the system will display detailed information – including plot synopsis, genre, cast, release year, and movie posters – to inform the user about the movie's context. It will also show trailers or teasers (via YouTube integration) so users can preview the film, and it will list direct streaming links or availability information (e.g., whether the movie is on Netflix, iWantTFC, etc.).
- **Facilitate access to viewing platforms:** The system's recommendations will be accompanied by information on where the user can watch the film immediately. By integrating data on legal streaming platforms, the system makes it easy for users to find which service (Netflix, Hulu, iWantTFC, etc.) currently offers the movie or if it's available for rent/purchase, thus streamlining the move from recommendation to viewing.

These objectives collectively ensure that the proposed system is user-centered – focusing on personalization (through mood and preferences), comprehensiveness of information, and convenience in accessing content. Ultimately, the purpose is to enrich the user's movie selection process, turning what is often a tedious decision task into a delightful discovery of Filipino cinema tailored to the user's feelings and interests.

Significance of the System

The **Suking Tindahan** offers several key benefits to its stakeholders and target users:

- **Empowering Filipino Viewers:** For Filipino movie lovers, especially **cinephiles seeking tailored recommendations**, this system fills a crucial void. It promotes access to local films by bringing them together in one personalized platform. Users will no longer need to hop between services or scroll endlessly; the system makes Filipino films more accessible with minimal effort (Pinoy Mood-based Movie Recommendation System.pdf). By providing recommendations that align with a user's mood, the system also enhances the **user experience**, making content consumption more satisfying and enjoyable (Pinoy Mood-based Movie Recommendation System.pdf). Instead of feeling overwhelmed by choices, users get a curated, relevant list of films that they are more likely to enjoy at that moment, thereby mitigating decision fatigue.

- **Cultural Preservation and Appreciation:** The platform has broader cultural significance in that it **promotes Philippine cinema** and encourages users (significantly younger generations) to appreciate both classic and contemporary Pinoy films (Pinoy

Mood-based Movie Recommendation System.pdf). Many older Filipino movies are rarely seen on mainstream platforms, leading to diminished awareness over time. By recommending classic films when appropriate (for example, suggesting a renowned drama from the 1970s to a user in a contemplative or serious mood), the system helps keep these classics alive and accessible. This benefits not only casual viewers but also **students and researchers of Philippine cinema**, who gain a tool for discovering films relevant to their study (e.g., finding movies by mood or theme for analysis). In the long term, such a system can contribute to a renewed interest in locally produced films and support the Philippine film industry through increased viewership and appreciation.

- **Stakeholder Benefits and Industry Impact:** Key stakeholders include the end-users (Filipino audiences), content creators/distributors, and streaming service providers. End-users benefit from convenience and personalization, as described above. Filipino filmmakers and the local film industry gain a promotional avenue – the system might surface independent films or festival films to users whose mood aligns with those works, giving those films a larger audience. Streaming services that carry Filipino content (such as Netflix's Filipino section, iWantTFC, VivaMax, etc.) also stand to benefit: by guiding users to their platforms to watch the recommended movies, the system can drive traffic to legal streaming channels (which aligns with combating piracy and supporting legal content consumption). In essence, the recommendation system acts as a **curated discovery layer** on top of existing streaming infrastructures. Easing the discovery process could increase overall user engagement time with Filipino content, which is a positive outcome for all stakeholders – users get enjoyable content, and providers get more viewers for Filipino movies.

In summary, the significance of this system lies in its dual impact on user experience and cultural promotion. It not only saves users time and effort by providing personalized, mood-matched suggestions, but it also serves as a platform that highlights Filipino stories and talents. By making local films more accessible and relevant to viewers' lives (through the lens of mood and emotion), the system encourages a deeper connection between the audience and Philippine cinema. This alignment of technology with cultural content can enrich entertainment options for Filipinos and strengthen the presence of local cinema in the digital age.

Materials and Methods

Overview: The Suking Tindahan Recommendation System will be implemented as a web-based information system that integrates several software components (APIs) and utilizes standard hardware and data resources. The development approach involves combining **Natural Language Processing (NLP)** for mood detection with a **movie recommendation engine** that filters and fetches content from external databases. The following sections describe the software and hardware components and the data to be used.

Software Components (APIs and Frameworks)

- **The Movie Database (TMDb) API:** TMDb is a widely used online movie database that provides a rich repository of movie information via web APIs. *"The Movie Database (TMDB) is a community-built movie and TV database"* with a substantial international scope, making it suitable for retrieving information on Filipino films as well as global titles. We will use the TMDb API to fetch **Filipino movie details** such as titles, synopses, genres, release dates, cast lists, posters, and trailers. TMDb's dataset includes both contemporary and classic films, which is crucial for our system to recommend older Filipino classics alongside modern movies. The API will allow filtering or querying by relevant criteria – for instance; we can search for movies by country of origin or original language (to focus on Filipino productions) and by genre or keywords that might correlate with certain moods (e.g., "romance" for a love-struck mood, "comedy" for a happy mood). TMDb also provides high-quality images (posters/backdrops) and often links to trailers. Its reliability and breadth (used in over 180 countries and handling billions of requests) ensure that our system will have a stable backbone for movie data.

- **Google Cloud Natural Language API:** To detect the user's mood from the input text, we will utilize Google's Cloud Natural Language API. This is a cloud-based NLP service that *provides natural language understanding technologies, such as sentiment analysis, entity recognition, entity sentiment analysis, and other text annotations*. In our context, the user will enter a brief text describing their feelings or state of mind (for example: *"I feel a bit lonely but hopeful today"*). The Natural Language API's **Sentiment Analysis** feature will analyze this text and return a sentiment score and magnitude, indicating how positive, negative, or neutral the text is and the strength of that emotion. We will interpret these results to classify the user's mood (e.g., positive/happy, negative/sad, neutral, or even more fine-grained categories like *joyful, melancholic, tense*, etc., depending on the analysis). Google Cloud NL API supports multiple languages; if users input text in Filipino/Tagalog, the API can attempt to detect the language, or we can preprocess the input (for the prototype, we might encourage input in English or Tagalog phrases that the API can handle). The choice of Google's API ensures we have a robust, pre-trained machine-learning model handling mood detection, avoiding the need to build our own sentiment analyzer from scratch. The output from this component will inform the recommendation logic – for example, a highly negative sentiment might trigger recommendations of uplifting or heart-warming films to improve the user's mood, whereas a positive sentiment might match with lighthearted or celebratory films.

- **JustWatch API:** Once candidate movie recommendations are selected, the next step is to inform the user where those movies can be watched. **JustWatch** is an online streaming guide service that aggregates data on the availability of movies across different streaming platforms. According to its description, *"JustWatch is a search engine that allows you to find out where your favorite movies and TV series are available across streaming services such as Netflix and Amazon Video."* We will use the JustWatch API (or the TMDb API's built-in *Watch Providers* data, which is powered by JustWatch) to find **legal streaming platforms for each recommended movie**. For instance, if the

system recommends a particular Filipino film, JustWatch can tell us if that film is currently on Netflix Philippines, iWantTFC, Amazon Prime, YouTube (free or paid), etc., including whether it's available by subscription, rent, or purchase. In implementation, we will likely query JustWatch's API by providing a movie title or TMDb ID and specifying the region as the *Philippines* to get country-specific streaming options. The system will then display to the user something like "*Available on Netflix, or Rent on Google Play Movies*," etc., next to each recommendation. This integration of JustWatch is crucial to achieving our objective of making it easy for users to go from seeing a recommendation to actually watching the film. Without it, users might get a recommendation but then have to manually search each streaming service to find the movie, reintroducing the inconvenience our system aims to eliminate.

- **YouTube Data API:** To enrich the user's decision-making, the system will incorporate the YouTube Data API to fetch **trailers or teaser videos** for the recommended movies. The YouTube Data API enables developers to search for YouTube content and retrieve videos programmatically; it *"acts as a bridge, allowing developers to tap into the vast video content … by providing programmatic access to search, channel data, and video information"*. Using this API, our system can automatically search YouTube for the official trailer or a teaser clip of a given movie (most likely using the movie title and possibly year as query terms). In many cases, TMDb already provides a YouTube trailer link as part of its data; when available, we will use that directly. If not, the YouTube API will serve as a backup to ensure we can show a relevant video. The trailer will be embedded or linked in the recommendation result so that users can watch a short preview. This feature is significant because a trailer gives users a quick glimpse of the film's tone and quality, which can help them decide if it indeed matches their mood or interest. For example, if the system suggests a comedy film for a user who's feeling down, watching the trailer via our platform could confirm that the film seems funny and uplifting, increasing the likelihood that the user will proceed to watch the whole movie. The YouTube API integration will be limited to read-only actions (search and retrieve video metadata/embed links); no user uploads or sensitive operations are needed.

In addition to these APIs, standard web development frameworks will be used to build the system's **front end** (user interface) and **back end** (server logic). The exact tech stack will be decided during development (as noted in the schedule, Week 1 includes tech stack selection). For instance, the project could use JavaScript (Node.js with NextJS and a frontend library like ReactTS) to implement the logic that ties these components together. The choice will consider the ease of integrating HTTP requests to the APIs, parsing JSON responses, and presenting results to the user in a responsive interface. Regardless of the framework, the architecture will follow a simple flow: **user enters mood -> backend calls NL API -> based on mood result, backend calls TMDb for movie discovery -> for each movie, call JustWatch (if needed) for availability and YouTube for trailer -> aggregate results and send to frontend for display.**

Hardware Components

The hardware requirements for this project are relatively minimal, as the core functions rely on cloud-based services. During development, a standard **personal computer or laptop** will be used to build and test the application. This machine should have a stable internet connection (to access the APIs) and moderate processing power and memory (typical of modern PCs) to run the development environment and local server. No specialized hardware (e.g., GPUs, sensors) is necessary because we are not training complex models locally – sentiment analysis and data storage are handled by external services or cloud infrastructure.

For deployment, the system can be hosted on a **web server or cloud platform**. This could be a cloud service like Vercel. The server will handle API requests and serve the web frontend.

On the client side, end-users will access the system through their devices, which could be **desktop computers, laptops, tablets, or smartphones** with a web browser. The system will be designed to be mobile-responsive so that even on a smartphone, the user can input their mood and view recommendations comfortably. In summary, the hardware environment includes a development machine for building the system, a hosting server for running the live system, and user devices for accessing it. All of these rely on internet connectivity to communicate with the remote APIs (TMDb, Google NLP, etc.) and to deliver content to the user. We will also maintain a small database on the server (this can be a lightweight SQL or NoSQL database) to store any necessary persistent data – for example, user profiles or favorites (if we allow accounts), logs of recommendations (for analysis), or cached movie data to reduce repeated API calls. This database will be hosted on the same server or as a cloud database service; hardware-wise, it does not require separate physical infrastructure beyond the chosen hosting solution.

Data and Data Sources

The project will handle several types of data sourced from both user input and external databases:

- **User Input Data:** The primary input from the user is a textual description of their mood or feelings. This is unstructured text (natural language) which will be sent to the Google Cloud Natural Language API for analysis. The privacy of this data is essential – the system will transmit the text securely to the API and will not store the user's raw input beyond the immediate analysis (unless the user opts to save a "session" or history for their own use). The output from the sentiment analysis is structured data (a sentiment score and possibly a magnitude, along with a language code and any extracted sentiment-associated entities). The system will interpret this output into a **mood category** internally (for example, a score < –0.25 might be categorized as "negative mood," a score > 0.25 as "positive mood," etc., with magnitude guiding the intensity).

- **Movie Metadata:** This includes all information about movies that will be recommended – titles, overviews (summaries), genres, release year, runtime, actors, etc. This data will be fetched on the fly from the **TMDb API**. TMDb returns data in JSON format, which our backend will parse. For instance, we will use endpoints like "Discover" or "Search" to find

movies matching specific criteria (such as Filipino movies of a particular genre or keyword). We may also use specific **movie IDs** if needed to fetch detailed info or to retrieve the trailer link and poster image path. The *type* of this data is primarily textual (names, descriptions), numeric (ratings, year), and links to media (URLs or paths for images and videos). We plan to filter TMDb's data primarily to **Filipino content**. TMDb does not have a direct "mood" tag, so the strategy will be to map moods to genres or keywords. For example, if the NL API determines the user is in a "happy/positive" mood, the system might query TMDb for Filipino **comedy** or **romance** films; for a "sad" mood, perhaps uplifting dramas or **feel-good** family movies; for "angry or stressed" moods, maybe action or inspirational stories, etc. These mappings will be refined through testing and could also be informed by any existing literature on mood-based recommendations. The data volume for movie metadata per request is not significant (a few dozen movies each with a couple of KB of info), which is easily handled by the backend.

- **Streaming Availability Data:** For each movie that is recommended, the system will gather data on where it's available to watch. Through the **JustWatch API**, we expect to receive a list of providers (like Netflix, Amazon, etc.) with details like type (subscription, rent, buy) and possibly a direct deep link. This data is essentially a set of labels (provider names) and can include URLs. We will filter it to show only the providers relevant to the Philippines region. The data ensures that each movie recommendation is accompanied by at least one option for legally accessing the film. If a movie is not available on any streaming service (which can happen especially for ancient films or obscure titles), the system will indicate that (or suggest alternatives). This aspect uses data that updates over time as streaming catalogs change, which is why querying an API like JustWatch in real time is preferable to a static dataset.

- **Trailer Video Data:** The trailer content from YouTube is multimedia data. We will typically only store or handle the **video ID or embed link** rather than the video file itself (we do not host videos; we just link/embed from YouTube). The YouTube Data API will return metadata, including the video ID, title, and description of the search results. We will extract the relevant video ID and use YouTube's embed functionality to display it. Thus, the data we handle is a short embed URL or iframe snippet, and the video streams directly from YouTube to the user's browser. We might store the YouTube link in memory or database for each recommended movie to avoid searching again if the user refreshes or for caching purposes, but it's lightweight.

- **User Profiles/Feedback (optional):** While not explicitly required in the proposal, it's worth noting that if we extend the system, we might incorporate user accounts where a user can save favorite movies or provide feedback (like liking a recommendation or marking one as not relevant). In such a case, additional data like user IDs, lists of favorite actors or liked movies, and past interaction history would be stored. This data would be stored in our system's database. For the scope of this proposal, we assume personalization is primarily via mood and optional favorite actor input each session (rather than a long-term learning profile), so we will not delve deep into this. However, it

remains a possible extension where collaborative filtering data (user ratings) could also come into play.

All data sources used are **established and reputable** services. TMDb is an open community-driven database with an API key access, Google Cloud NL is a trusted enterprise API for NLP, JustWatch is a commercial service with public API endpoints for search, and YouTube is a Google service with a well-documented API. We will comply with each API's usage policies (e.g., using API keys, respecting rate limits, and attributing as required by TMDb's terms). Data security considerations include using HTTPS for all API calls and handling API keys securely on the server side. Finally, the integration of these diverse data types (text, images, video links) will be handled in the web interface by presenting a **unified view** for each recommended movie: e.g., an entry containing the movie title, an image (poster), text description, maybe icons or badges for mood/genre, a trailer video embed, and icons/logos of streaming services where it's available.

Schedule of Activities

The project will be executed over a period of six weeks, with each week focused on specific milestones and deliverables. The following table outlines the schedule of activities:

Week	Activities / Milestones
Week 1	**Project Initiation:** Define the project scope and requirements in detail. Formulate the objectives and success criteria for the system. Select the technology stack (programming language, frameworks, databases) and gather API keys/credentials for TMDb, Google NL, JustWatch, and YouTube. Develop a detailed project plan.
Week 2	**Design Phase:** Create the user interface (UI) and user experience (UX) design mock-ups. Develop wireframes for the main screens (mood input page, recommendation results page, etc.). Establish the visual branding (color scheme, logos/icons, overall aesthetic) to ensure the application is user-friendly and appealing. Review the designs with stakeholders or mentors and refine them based on feedback.

Week 3 **Backend Development & API Integration:** Begin implementation of the backend logic. Set up the server environment and integrate the external APIs. This includes writing modules to call the Google NL API for sentiment analysis, parsing its response, and calling the TMDb API to search for movies based on mood results. Develop the core *recommendation algorithm* that maps mood output to movie query parameters (genres/keywords). If user preference (e.g., favorite actor input) is included, incorporate that into the search/filter logic (for example, ensure some recommendations feature the specified actor). By the end of the week, the backend should be able to take a mood input and return a preliminary list of movie suggestions (without yet focusing on the front-end display). Unit tests will be written for API call functions to validate that data is being retrieved as expected.

Week 4 **Frontend Development:** Develop the client side of the application. Implement the web pages according to the Week 2 designs using HTML/CSS/JavaScript (or a frontend framework if chosen). Connect the frontend to the backend via HTTP requests (for example, the user submits the mood form, which triggers an API call to our backend endpoint). Ensure the recommended movie data received from the backend is displayed clearly: e.g., create a results layout that shows each movie's poster, title, summary, etc. Implement the embedding of YouTube trailers in the results. Make the interface responsive to different screen sizes. Basic interactivity is added (like the ability to click a streaming provider link to open the movie on that service).

Week 5 **Integration, Testing, and Optimization:** This week focuses on end-to-end integration of all components and rigorous testing. Integrate the JustWatch API data into the backend flow so that each movie result also contains information on where it can be streamed. Conduct comprehensive testing: *Functional testing* (does the system correctly identify moods and return appropriate recommendations?), *Usability testing* (is the UI clear and easy to use?), and *Compatibility testing* (does it work across common browsers and devices?). Gather a small group of test users to try the system and provide feedback. Based on testing, debug any issues (e.g., API errors, incorrect mapping of moods to movies, layout problems) and optimize performance (caching frequent requests, minimizing load times for images/trailers). This phase may also involve refining the recommendation logic – for example, if testing shows that some mood inputs yield unsatisfactory suggestions, we will tweak the mapping or include additional rules (perhaps a very specific mood might default to a well-rated popular film in a fitting genre as a safe suggestion). Security checks are also done (ensuring API keys are not exposed on the client side, etc.).

Week 6	**Deployment and Project Closure:** Deploy the application to a live server or cloud service so that it can be accessed by the target users. Verify that the deployed version is functional (final round of testing in the production environment). Do final optimizations, such as scaling server resources if needed or setting up a custom domain. Prepare documentation for the project, including a user guide and technical documentation for maintenance. Finally, prepare for the **project presentation** – create slides and demonstrations to showcase the system's features, discuss the objectives achieved, and highlight the significance of the work. The project presentation (and/or defense) will conclude the schedule, where feedback from evaluators will be taken for future improvements.

This timeline ensures structured progress from planning to execution to deployment within a 6-week window. Each week's activities build on the previous one (for example, the design in Week 2 guides the frontend in Week 4, and the backend developed in Week 3 is integrated and fine-tuned in Week 5). The Gantt chart representation of this schedule would show these tasks sequentially with some overlap between the development and testing phases. Milestones such as *"Initial prototype ready"* (end of Week 4) and *"Testing completed"* (end of Week 5) help track progress. By following this schedule, the project team will be able to deliver a functional **Pinoy Mood-Based Movie Recommendation System** within the planned time frame while allowing time for iteration and quality assurance.

References

Aguila, R., & Beltran, W. (2022, December 10). *Discover Filipino films with FDCP's streaming platform "JuanFlix"*. The Benildean. (Discover Filipino films with FDCP's streaming platform "JuanFlix")

Bhushan, S., Singh, A. K., Kumar, S., Sharmila, & Bhuyan, B. P. (2024). *Mood-Based Movie Recommendation System Using Sentiment Analysis*. In **Proceedings of the 4th International Conference on Machine Learning, Advances in Computing, Renewable Energy and Communication (MARC 2023)** (Lecture Notes in Electrical Engineering, vol. 1232, pp. 235–246). Springer, Singapore. (Mood-Based Movie Recommendation System Using Sentiment Analysis | SpringerLink) (Mood-Based Movie Recommendation System Using Sentiment Analysis | SpringerLink)

Furey, M. (2023, September 28). *How Reelgood is Putting an End to Decision Fatigue*. Reelgood for Business. (How Reelgood is Putting an End to Decision Fatigue - Reelgood for Business)

Google Cloud. (n.d.). *Cloud Natural Language API – Documentation*. Retrieved March 27, 2025, from cloud.google.com. (Cloud Natural Language API | Google Cloud)

Joseph, B. (2025, January 22). *Video Streaming Is Taking a Hit Because Users Have Too Much Content to Choose From*. HackerNoon. (Video Streaming Is Taking a Hit Because Users Have Too Much Content to Choose From | HackerNoon)

JustWatch. (2023). *JustWatch – Streaming Search Engine for Movies and TV Shows* (Web Page). Retrieved March 27, 2025, from Wikipedia: https://en.wikipedia.org/wiki/JustWatch (JustWatch - Wikipedia)

The Movie Database (TMDb). (n.d.). *About TMDb*. Retrieved March 27, 2025, from https://www.themoviedb.org/about (About TMDB — The Movie Database (TMDB))

Tubb, R. (n.d.). *JustWatch – Streaming Search Engine for Movies and TV*. Tubblog. (JustWatch - Streaming Search Engine for Movies and TV)

YouTube. (n.d.). *YouTube Data API – Overview*. Google for Developers. Retrieved March 27, 2025. (YouTube Data API Overview | How to Implement it in Your Applications)