LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT

MUSIC RECOMMENDER SYSTEM WITH SENTIMENT ANALYSIS

UE20CS461A – Capstone Project Phase – 2

Submitted by:

Reshmi Pradeep	**PES2UG20CS270**
Alekhya Sundari R	**PES2UG20CS900**
Nanduri	**PES2UG20CS905**
D. Mrudula	**PES2UG20CS910**
Harshitha Golla	

Under the guidance of

Dr. L. Kamatchi Priya
Professor
PES University

August - December 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
Electronic City, Bengaluru – 560 100, Karnataka, India

TABLE OF CONTENTS

1. Introduction

1.1. Overview

The low-level design document provides a detailed plan for a music recommendation system. It considers design constraints, assumptions, and dependencies, encompassing factors like data quality and user behavior. The document includes class and use case diagrams, an algorithmic approach for web scraping and sentiment analysis, and outlines the modular architecture of the system. The proposed methodology involves data preprocessing, machine learning techniques, and a hybrid recommendation system. The system comprises multiple modules, and a user interface is deployed for end-users, offering personalized music recommendations based on sentiment analysis of their Twitter activity.

1.2. Purpose

This Low-Level Design Document (LLD) aims to fully describe the technical aspects of the sentiment analysis-based music recommendation system.

The goal of the document is to give a thorough breakdown of the system's architecture, component specifications, communication protocols, data handling techniques, security and privacy safeguards, scalability and performance considerations, error-handling tactics, deployment and maintenance procedures, dependencies, and documentation standards. This document, which offers precise and lucid insights into the system's architecture, is an essential tool for assuring the project's technical success.It also demonstrates how various deep learning and machine learning architecture are put up using various frameworks.

1.3. Scope

A Low-Level Design (LLD) document's scope includes a thorough and in-depth analysis of the architecture and parts of a software system. It describes the precise design of each module or sub-system, delving into the technical details. Data structures, algorithms, data flow diagrams, control flow diagrams, and the exact interfaces and interactions between these components are only a few of the numerous intricacies covered in this. The intricate details of error handling techniques, performance enhancements, memory management, and other platform-specific issues are covered in the LLD paper. To ensure uniformity throughout the development process, it may also include recommendations for coding practices and standards. The LLD document acts as a crucial bridge between the high-level design and the actual implementation, offering developers a granular view of how to construct and integrate each part of the system, ensuring a coherent and efficient final product.

2. Design Constraints, Assumptions, and Dependencies

2.1. Assumptions

- The system assumes that users have public Twitter accounts and have tweeted quite a number of tweets to get a better understanding of their personality.
- The web scraper provides reliable and consistent access to web data.
- The users don't use much sarcasm in their tweets or threads.of the scraped data.
- The emotional classifier is accurate and can correctly identify the sentiment of the scraped data.
- The music dataset used for content-based filtering is comprehensive and up-to-date.
- Users will provide honest and accurate feedback for collaborative filtering.

2.2. Design Constraints

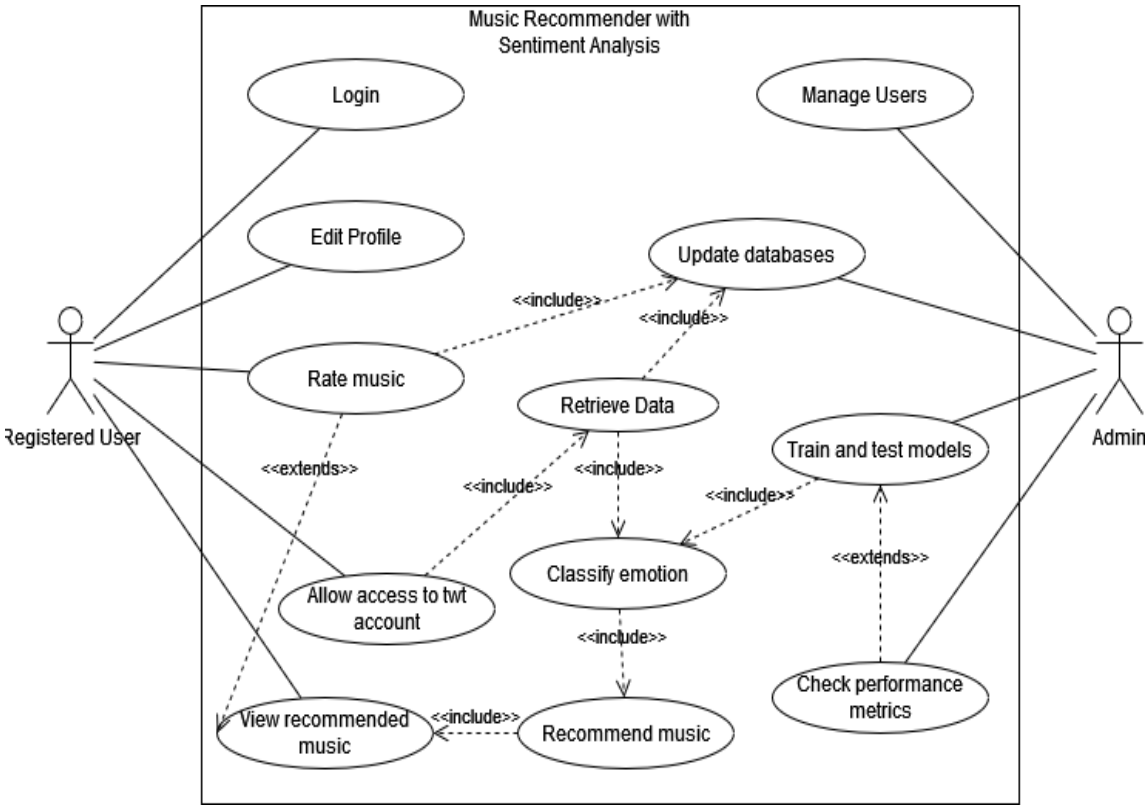
- The system is limited to the availability and quality of the data obtained from the web scraper and the music dataset.
- The performance of the emotional classifier and recommendation system may be affected by the size and complexity of the datasets.
- The user interface and functionality may be limited by the capabilities of the web application framework used for development.

2.3. Dependencies

- The system relies on access to the web data obtained by the web scraper and the music dataset.
- The sentiment analysis algorithm and recommendation system depend on the quality and completeness of the datasets.
- The music that is recommended is dependent on the sentiment classified by the emotion classifier.

3. Design Description

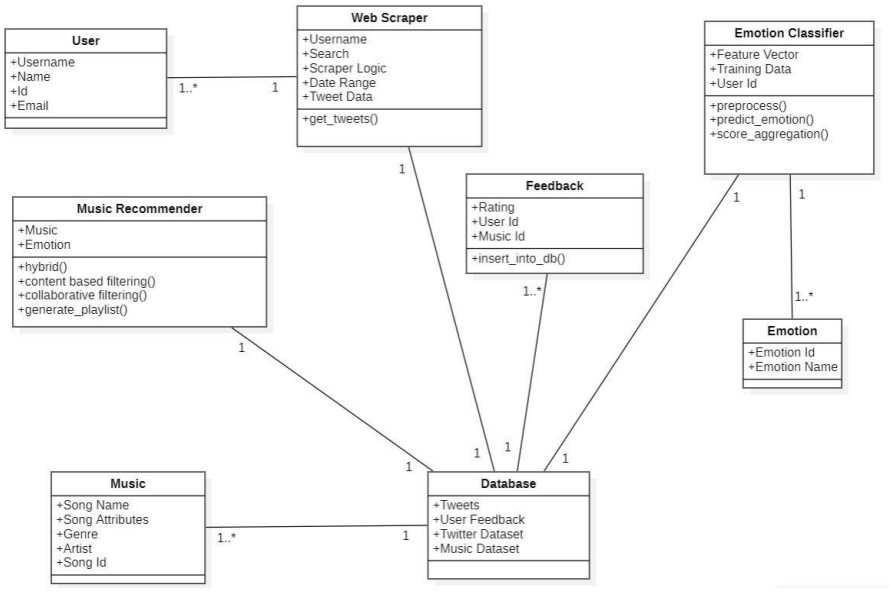
3.1. Use Case Diagram



Use Case Item	Description
Registered User	The registered user is the person who enters their twitter/threads username to receive music recommendations.
Admin	The admin performs tasks such as managing users, updating the database, and overseeing the system's operations.
Login	This use case allows registered users to log into the system by providing their credentials
Edit Profile	This use case allows registered users to edit their profiles to update personal information,
Rate music	This use case enables registered users to provide feedback on music they have been recommended to improve the music recommendations.
Allow access to twitter account	This use case allows the web scraper to access the user's account so that tweets/threads can be extracted.

View recommended music	This use case enables registered users to view music recommendations generated by the system based on the emotion classification of their tweets.
Manage Users	This use case allows admins to manage user accounts, including tasks like creating, updating, or deleting user profiles.
Update Database	This use case represents the process of updating the system's database with user feedback.
Retrieve Data	This use case enables the user's tweets/threads to be extracted through web scraping.
Classify emotion	This use case enables the system to analyze the emotion of a user's tweets/threads.
Recommend music	This use case enables the system to generate recommendations based on the emotion that is classified.
Train and test model	This use case represents the training and testing of machine learning models or algorithms used for music recommendations and sentiment analysis.
Check performance metrics	This use case enables the system to evaluate the performance of its recommendation and sentiment analysis algorithms. This involves tracking metrics like accuracy, precision, recall, and F1-score to ensure the system's effectiveness.

3.2. Class Diagram



3.2.1. User

3.2.1.1. Description

The User class is a class used to define the default values and assign the required values to each user that is created in the system.

3.2.1.2. Data members

Data Type	Data Name	Access Modifiers	Initial Value	Description
String	Username	public	“”	stores username
String	name	public	“”	stores name
integer	Id	public	0	stores unique id
String	Email	public	“”	stores email id

3.2.2. Web Scraper

3.2.2.1. Description

The web scraper class is used to define the web scraping module of the system.

3.2.2.2. Data members

Data Type	Data Name	Access Modifiers	Initial Value	Description
String	Username	public	“”	stores username
String	Search	public	“”	stores search data
date	Date Range	public	01-01-2000	stores start date from which tweets are to be retrieved
String	Tweet data	public	“”	stores user’s data

3.2.2.3. get_tweets()

- This method is used to retrieve a user's tweets from their twitter account
- It takes the username and date range as input
- It gives a .csv file of the user's tweets as the output
- Parameters are username and date range
- Exceptions occur if the user's account is private or doesn't exist

3.2.2.4. get_threads()

- This method is used to retrieve a user's threads from their threads account
- It takes the username as input
- It gives a .csv file of the user's threads as the output
- Parameters are username
- Exceptions occur if the user's account is private or doesn't exist

3.2.3. Emotion Classifier

3.2.3.1. Description

The emotion classifier class is used to define the emotion classifier module of the system. This used to classify the user's data into different emotions and finally find the overall emotion.

3.2.3.2. Data members

Data Type	Data Name	Access Modifiers	Initial Value	Description
list	Feature_vector	public	[]	stores the combined feature vector of USE, GLoVe and Tf-idf
String	Training_data	public	""	stores training data
integer	userid	public	0	stores unique userid

3.2.3.3. preprocess()

- This method is used to preprocess user's tweets stored in .csv file
- It takes the .csv file with tweets or threads as input
- It gives a .csv file of the cleaned user's tweets as the output

- Parameter is name of the .csv file
- Exceptions occur if the number of tweets retrieved is zero

3.2.3.4. predict_emotion()

- This method is used to predict an emotion for each value in the .csv file
- It takes the input as cleaned csv file
- It gives a .csv file of the user's tweets, each classified with an emotion as the output
- Parameter is name of the input csv file
- Exceptions occur if the number of tweets in csv file is zero

3.2.3.5. score_aggregation()

- This method is used to aggregate all the classified emotions to find the overall emotion of the user
- It takes the .csv file of the user's tweets, each classified with an emotion as input
- It gives a string showing the emotion as the output
- Parameter name of the csv file
- Exceptions occur if any tweet in the file is not classified with an emotion

3.2.4. Emotion

3.2.4.1. Description

The emotion class is used to define the emotions used for classification in the model.

3.2.4.2. Data members

Data Type	Data Name	Access Modifiers	Initial Value	Description
integer	emotion_id	public	0	stores unique id of each emotion
String	emotion_name	public	""	stores name of each emotion

3.2.5. Music Recommender

3.2.5.1. Description

The music recommender class is used to define the music recommender module of the system. This is used to recommend music to the user based on the classified emotion and user feedback.

3.2.5.2. Data members

Data Type	Data Name	Access Modifiers	Initial Value	Description
list	music	public	[]	stores the music that can be recommended based on the emotion
integer	emotion	public	“”	stores id of the overall emotion classified

3.2.5.3. content_based_filtering()

- This method is used to make music recommendations by analyzing the attributes of songs and then suggesting items that are similar in terms of those attributes to songs the user has shown interest in.
- It takes input as the classified emotion of the tweet or thread.
- It gives a list of songs that is recommended to the user.
- The parameter is emotion.
- Exception occurs when the attribute data is incomplete or incorrect or if the classified emotion is incorrect.

3.2.5.4. collaborative_filtering()

- This method is used to make music recommendations by taking user feedback into consideration as well.
- It takes input as the classified emotion of the tweet or thread.
- It gives a list of songs that is recommended to the user.
- The parameter is emotion.
- Exception occurs when there is no user with the target emotion in the database or if the classified emotion is incorrect.

3.2.5.5. hybrid()

- This method is used to combine both of the recommender systems to provide a final list of music recommendations to the user.
- It takes input as the classified emotion of the tweet or thread.
- It gives a list of songs that is recommended to the user.
- The parameter is emotion.
- Exception occurs if the classified emotion is incorrect

3.2.5.6. generate_playlist()

- This method is used to generate a playlist for the user which includes all the music recommendations that were given.

3.2.6. Music

3.2.6.1. Description

The music class is used to define all the songs that can be recommended to the user.

3.2.6.2. Data members

Data Type	Data Name	Access Modifiers	Initial Value	Description
String	song name	public	[]	stores the name of the song
List	song attributes	public	“”	stores the different attributes of songs like tempo, etc.
String	genre	public	“”	stores the genre of the song
String	artist	public	“”	stores the artist of the song
integer	songid	public	“”	stores the id of a particular song

3.2.7. Database

3.2.7.1. Description

The database class is used to store the records of the user feedback and other data.

3.2.7.2. Data members

Data Type	Data Name	Access Modifiers	Initial Value	Description
String	tweets/threads	public	[]	stores the tweets/threads of the user.
integer	user feedback	public	“”	stores the feedback that the user provides
-	twitter dataset	public	-	stores the twitter dataset that is used to train the sentiment analysis model.
-	music dataset	public	-	stores all the songs which are used to give recommendations to the user

3.2.8. Feedback

3.2.8.1. Description

The feedback class is used to define the feedback part of the system where users can give ratings for songs to improve music recommendations.

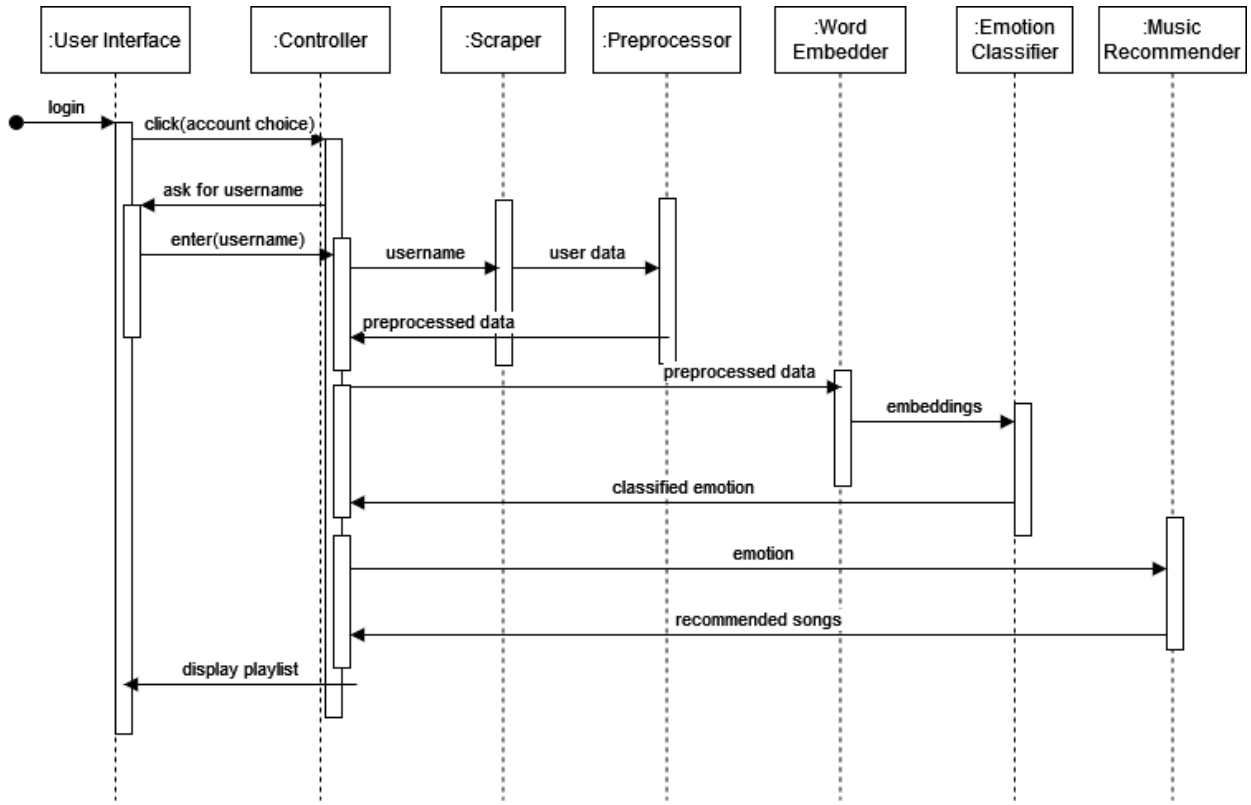
3.2.8.2. Data members

Data Type	Data Name	Access Modifiers	Initial Value	Description
integer	rating	public	0	stores the rating of the user for the recommended music.
integer	userid	public	0	stores the userid of the user.
integer	music id	public	0	stores the id of a song

3.2.8.3. insert_into_db()

- This method is used to insert a record into the database.
- The input is taken as a particular record to be inserted into the database
- The record successfully gets added to the database.
- The parameter is the record to be added.
- Exception occurs if the record is not properly inserted into the database due to a data type error, etc.

3.3. Sequence Diagram



4. Proposed Methodology / Approach

4.1. Algorithm and Pseudocode

4.1.1. Web Scraper

This component scrapes tweets from a Twitter account or threads from the threads account whose username is provided by the user.

4.1.1.1. Twitter Scraping

- Initialize a Twitter session and sign in.

- Specify the target username for tweet retrieval.
- Fetch user information and collect tweets from the target account.

4.1.1.2. Threads Scraping

- Specify the target username for threads retrieval.
- Fetch user information and collect threads from the target account.

4.1.2. Data Preprocessing

Applies preprocessing steps such as tokenization, stopword removal, and lemmatization to clean the text data:

- Convert to lowercase.
- Remove URLs, hashtags, mentions, special characters, and numbers.
- Tokenize and expand contractions.
- Remove stop words, punctuations, and lemmatize tokens.
- Store pre-processed tweets.

4.1.3. Word Embeddings

This combines multiple embedding techniques (GloVe, TF-IDF, USE) to represent tweets, allowing for a richer understanding of text data. The resulting embeddings are used in downstream tasks like sentiment analysis and music recommendation.

4.1.3.1. GloVe Embeddings

- Load pre-trained GloVe word embeddings into a dictionary.
- For a list of tweets, preprocess each tweet by converting to lowercase and splitting into words.
- For each word in the preprocessed tweets, obtain its GloVe word embedding from the dictionary.
- Calculate the average word embeddings for each tweet, resulting in a matrix of tweet embeddings.

4.1.3.2. TF-IDF Vectorization

- Initialize a TF-IDF vectorizer.
- Fit and transform the preprocessed tweets to compute TF-IDF vectors.
- Retrieve the feature names (words) corresponding to each column in the TF-IDF matrix.
- Convert TF-IDF vectors to a dense array for easier handling.

4.1.3.3. Universal Sentence Encoder

- Load the Universal Sentence Encoder model.
- Encode the tweets using the Universal Sentence Encoder (USE) to obtain tweet embeddings.

4.1.3.4. Combining Embeddings

- Stack the embeddings obtained from GloVe, TF-IDF, and USE along a new dimension.
- Pad the arrays with zeros to match the maximum number of tweets.
- Save the resulting stacked embeddings to a file for later use.

4.1.4. DCNN Model :

The DCNN (Dilated Convolutional Neural Network) model is used for classifying the emotions expressed in textual data, enabling sentiment analysis, and understanding the emotional content of text.

4.1.4.1. Load and Preprocess Data

- Load the dataset containing text and emotion labels.
- Tokenize the text data and split it into training and testing sets.

4.1.4.2. Build CNN Model for Text Classification

- Define input layer with a specified sequence length.
- Create an embedding layer for learning word representations.
- Use dilated convolutional layers with batch normalization and dropout.
- Add dense layers for classification, with regularization.
- Compile the model using AdamW optimizer and binary cross-entropy loss.

4.1.4.3. Train and Evaluate Model

- Train the model on the training data.
- Implement early stopping to prevent overfitting.
- Evaluate the model on the test data and print the test loss and accuracy.

4.1.5. Hybrid Recommender system

4.1.5.1. Content based recommender system

Make recommendations by analyzing the characteristics or attributes of songs and then suggesting items that are similar in terms of those attributes to songs the user has shown interest in.

- Dataset: The Moodify Dataset comprehensive collection of music data that incorporates emotional categorization into song classification. This dataset consists of approximately 278,000 songs sourced from Spotify, and each song is labeled with specific emotions. The primary objective of the Moodify project is to classify songs not only based on their lyrical and musical features but also by incorporating emotions as a key factor in the categorization process.
- Extreme Gradient Boosting(XG Booster): used to classify music based on emotions. It is a popular and efficient gradient boosting machine learning algorithm known for its high predictive accuracy and computational efficiency.

- As the dataset contains the url of the songs used spotify api to convert to the title of the song.

4.1.5.2. Collaborative recommender system

It is used to predict a user's interests or preferences by collecting and analyzing information about the behaviors and preferences of a group of users.

- Created a dataset with user name,emotion,title of the song,rating.
- The input to the system is emotion.
- For user user similarity it calculates the similarity between the target emotion users and suggests the songs which are rated high.
- For item item similarity it calculates the similarity between the songs of the target emotion required and recommends the songs which are similar.
- The final recommendation is the combination of both the recommender systems.

4.2. Implementation and Results

4.2.1. Comparison of the Machine learning algorithms used in recommender systems

Models		Accuracy
Support Vector Machine	Linear Kernel	84.48
	RBF kernel	89.01
Decision Tree Classifier		90.57
Random Forest		94.04
Extreme Gradient Boosting (XG Boost)		96.63

Shows that the SVM with a linear kernel achieves an accuracy of 84.48%, while the SVM with an RBF kernel outperforms it, exhibiting greater accuracy. This superiority can be attributed to the latter's capacity to capture intricate patterns within the data, making it suitable for complex and high-dimensional datasets. This also makes SVM models computationally intensive. Decision trees are known for their computational efficiency but carry the risk of overfitting. Ensemble methods like Random Forest manage to strike a balance by combining multiple decision trees, offering flexibility and competitive accuracy. The extreme gradient boosting method, XGBoost, consistently outperforms Random Forest, predominantly due to its distributed computing capabilities and advanced feature engineering.

Models	Evaluation Metrics			
	Precision	*Recall*	*F1 score*	*Support*

SVM	0.84	0.84	0.84	55588
Decision Tree	0.91	0.91	0.91	55588
Random Forest	0.94	0.94	0.94	55588
XG Boost	0.97	0.97	0.97	55588

It is observed that XGBoost and Random Forest have the highest precision, recall, and F1 scores indicating that they perform well in terms of classification accuracy on the given dataset.

4.2.2. Comparison of models used for emotion classification

Four models were initially used to classify emotions which were LSTM, Bi-LSTM, CNN and DCNN. Three parallel layers of DCNN which have dilation rates 1,2 and 4 are stacked with a global average pooling layer to fine-tune the model, which showed the highest accuracy.

Models	Accuracy
LSTM	38%
Bi - LSTM	10%
Convolutional Neural Network (CNN)	16%
Dilated Convolutional Neural Network (DCNN)	53%

Appendix A: Definitions, Acronyms and Abbreviations

- DCNN: Dilated Convolution Neural Networks - deep learning models that use a convolution operation with increased gaps between the input feature maps to increase the receptive field without increasing the number of parameters
- GloVe: Global Vectors for Word Representation - an unsupervised learning algorithm for obtaining vector representations for words.

- TF-IDF: Term Frequency-Inverse Document Frequency - a measure that can quantify the importance or relevance of string representations in a document amongst a collection of documents
- USE: Universal Sentence Encoder - encodes text into high dimensional vectors that can be used for text classification, semantic similarity, clustering, and other natural language tasks
- XGBoost: eXtreme Gradient Boosting - XGBoost is a decision-tree-based ensemble machine learning algorithm that uses a gradient boosting framework.

Appendix B: References

- Irvin Dongo, Yudith Cardinale, Ana Aguilera "A qualitative and quantitative comparison between Web scraping and API methods for Twitter credibility analysis" in International Journal of Web Information Systems © Emerald Publishing Limited 1744-0084 DOI 10.1108/IJWIS-03-2021-0037
- Project Management Institute. (2017). A guide to the Project Management Body of Knowledge (PMBOK guide) (6th ed.). Project Management Institute.
- P. K. K. N. and M. L. Gavrilova, "Latent Personality Traits Assessment From Social Network Activity Using Contextual Language Embedding," in IEEE Transactions on Computational Social Systems, vol. 9, no. 2, pp. 638-649, April 2022, doi: 10.1109/TCSS.2021.3108810.
- H. Ahmad, M. U. Asghar, M. Z. Asghar, A. Khan and A. H. Mosavi, "A Hybrid Deep Learning Technique for Personality Trait Classification From Text," in IEEE Access, vol. 9, pp. 146214-146232, 2021, doi: 10.1109/ACCESS.2021.3121791.
- Praphula Kumar Jain, Waris Quamer, Vijayalakshmi Saravanan, and Rajendra Pamula, “Employing BERT-DCNN with sentic knowledge base for social media sentiment analysis” in Journal of Ambient Intelligence and Humanized Computing, January 2022
- Saba Yousefian Jazi 1 & Marjan Kaedi1 & Afsaneh Fatemi1,”An emotion-aware music recommender system: bridging the user’s interaction and music recommendation” in Multimedia Tools and Applications(springer),15 January 2021

Appendix C: Record of Change History

#	Date	Document Version No.	Change Description	Reason for Change
1.				
2.				
3.				

Appendix D: Traceability Matrix

[Demonstrate the forward and backward traceability of the system to the functional and non-functional requirements documented in the Requirements Document.]

Project Requirement Specification Reference Section No. and Name.	DESIGN / HLD Reference Section No. and Name.	LLD Reference Section No. Name
Reference Section No. 4, Functional Requirements	Reference Section 10 - Design Details	
Reference Section No. 6 - Non Functional Requirements	Reference Section 10 - Design Details, Reference Section 3 - Design Considerations, Reference Section 4 - High Level System Design	