**Mini Project report submitted to**

Text, logo

Description automatically generated with medium confidence

**AI Human Emotion Mimicker and Memory Reminder System**

by

**Haiya Shah**

Registration No: 215890018

Department of Computer Science and Engineering

Manipal Institute of Technology

Bengaluru

Under the supervision of

|  |  |
| --- | --- |
| **Dr. Rashmi R.**  Assistant Professor  Department of Computer Science and Engineering  Manipal Institute of Technology  Bengaluru | **Dr. Gerard Deepak**  Assistant Professor  Department of Computer Science and Engineering  Manipal Institute of Technology  Bengaluru |

|  |  |
| --- | --- |
|  |  |

**Manipal Institute of Technology**

**Bengaluru Campus-560064, Karnataka, India.**

**Certificate**

This is to certify that as the record of the mini project work carried out by them, is accepted as the Mini Project reportsubmission in partial fulfilment of the requirements for the award of degree of Bachelor of Technology (BTech).

|  |  |
| --- | --- |
| **Examiner Name** | **Signature with the Date** |
| Dr. Rashmi R. | ……………………………… |
| Dr. Gerard Deepak | ……………………………… |

**ABSTRACT**

The AI Emotion Mimicker & Memory Reminder System is an innovative project aimed at transforming user interaction with technology through the integration of advanced artificial intelligence (AI) capabilities with a robust full-stack architecture. This comprehensive system encompasses various components, including the frontend, backend, database, API usage, algorithms, and AI integration, meticulously designed to deliver a personalized and immersive user experience.

The existing landscape of emotion mimicking and memory reminder systems lacks depth in human emotion understanding and often provides fragmented solutions for memory storage and reminders. Standalone applications typically focus on either emotion mimicking or memory storage, resulting in disjointed user experiences. To address these limitations, the AI Emotion Mimicker & Memory Reminder System is designed to offer a comprehensive solution that seamlessly integrates AI capabilities with a robust full-stack architecture.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SL. NO.** | **CONTENTS** | **PAGE NO.** |
| 1 | INTRODUCTION | 1 |
| 2 | PROBLEM STATEMENT | 2 |
| 3 | OBJECTIVES | 3 |
| 4 | METHODOLOGY | 4 |
| 5 | IMPLEMENTATION | 6 |
| 6 | RESULTS | 8 |
| 7 | CONCLUSION | 9 |
| 8 | REFERENCES | 10 |

**INTRODUCTION**

The AI Emotion Mimicker & Memory Reminder System represents an innovative endeavor aimed at revolutionizing user interaction with technology. Through the integration of advanced artificial intelligence (AI) capabilities with a robust full-stack architecture, this project seeks to provide a personalized and immersive user experience.

Comprising various components such as the frontend, backend, database, API usage, algorithms, and AI integration, this comprehensive system has been meticulously designed to cater to the diverse needs of users.

The current landscape of emotion mimicking and memory reminder systems often falls short in understanding human emotions and delivering efficient memory storage solutions. Typically, standalone applications focus solely on one aspect, leading to fragmented and disjointed user experiences.

To overcome these limitations, the AI Emotion Mimicker & Memory Reminder System offers a holistic solution. By seamlessly integrating AI capabilities within a robust full-stack architecture, it aims to provide users with a cohesive and seamless experience, ensuring both emotional understanding and efficient memory management.

**PROBLEM STATEMENT**

This project aims to develop an AI system capable of mimicking human emotions and providing memory reminders with the help of emotional support through various timeless stories.

**OBJECTIVES**

The objectives of this project encompass:

1. Emotion Mimicking: Develop an AI capable of accurately mimicking a wide range of human emotions in real-time.
2. Memory Storage: Implement a secure and scalable memory storage system to allow users to store various types of memories including images, videos, and text.
3. Personalized Reminders: Utilize machine learning algorithms to analyze user behavior and preferences, providing personalized reminders for events and tasks.
4. Scalability and Performance: Optimize the system for scalability and performance to accommodate a growing user base and ensure smooth operation

**METHODOLOGY**

The methodology can be classified into seven essential steps:

1. Data Collection: Gather extensive textual data from various sources, including literature, articles, and online content, to create a diverse dataset for emotion analysis and story recommendation.
2. Preprocessing: Utilize natural language processing techniques to preprocess the collected data, including tasks such as tokenization, stop-word removal, and stemming, to prepare the text for further analysis.
3. NLP Processing: Apply NLP techniques, including NTL n-grams, to the preprocessed data to analyze patterns and sentiments associated with different emotional states. Extract features relevant to emotion recognition, such as word context and frequency.
4. Emotion Recognition: Train emotion recognition models using machine learning algorithms on the processed data to accurately identify and categorize emotional states expressed in the text. Evaluate the models' performance and refine as necessary to improve accuracy.
5. Story Recommendation: Develop a recommendation system that suggests relevant stories based on the user's emotional state inferred from the analyzed text. Utilize techniques such as collaborative filtering or content-based filtering to personalize recommendations.
6. Integration: Integrate the emotion recognition and story recommendation components into a cohesive AI system, ensuring seamless interaction between the two functionalities.
7. User Interaction: Design a user interface that allows users to input text or interact with the system, receiving personalized emotion mimicking and memory reminder services based on their input.

**IMPLEMENTATION**

Code and Explanation

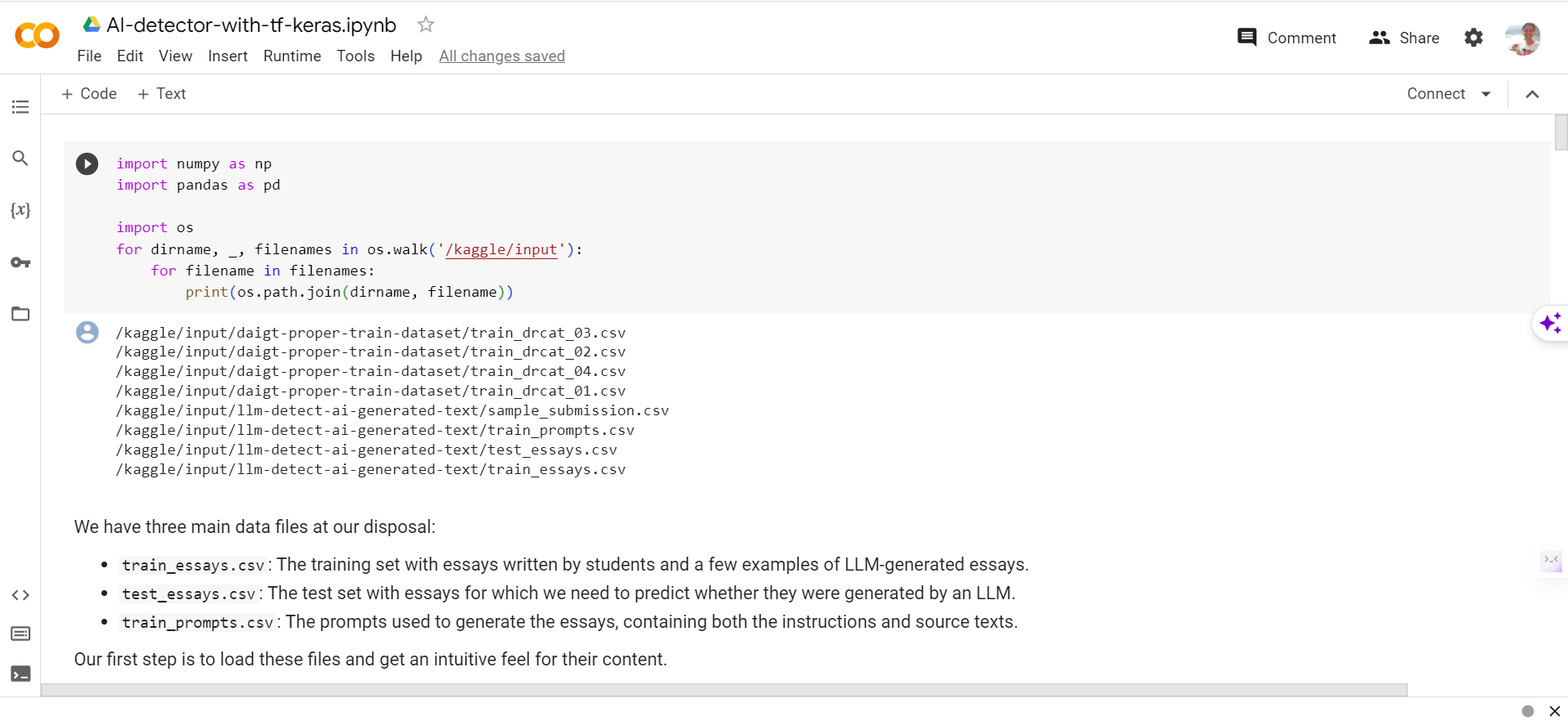


Fig. 1: In this Fig., necessary libraries are imported. This includes NumPy and Pandas for data manipulation, as well as a basic setup for accessing input data files provided by Kaggle.

A screenshot of a computer

Description automatically generated

Fig. 2: An introduction to the data files is provided. It mentions the three main data files: train\_essays.csv (training set with essays), test\_essays.csv (test set for predicting LLM-generated essays), and train\_prompts.csv (prompts used for essay generation). This Fig. sets the context for the dataset used.

A screenshot of a computer

Description automatically generated

Fig. 3: This section involves loading data from the data files into Pandas DataFrames. It defines the file paths and loads data from the CSV files. It's important for the subsequent data processing and analysis.

A screenshot of a computer

Description automatically generated

Fig. 4: Here, an exploratory data analysis (EDA) is performed. It checks for missing values in the datasets (df and train\_prompts\_df) and prints out the results. This EDA provides insights into data cleanliness.

A screenshot of a computer

Description automatically generated

Fig. 5: Feature engineering is introduced in this section. It involves processing the text data. The text is split into words, and various features such as word count, unique word count, and others are extracted.

A screenshot of a computer

Description automatically generated

Fig. 6: After feature engineering, this Fig. reshapes and concatenates the data. It's a crucial step as it combines the training essays data with the previously loaded data from train\_drcat\_04.csv.

A screenshot of a computer

Description automatically generated

Fig. 6: After feature engineering, this Fig. reshapes and concatenates the data. It's a crucial step as it combines the training essays data with the previously loaded data from train\_drcat\_04.csv.

A screenshot of a computer

Description automatically generated

Fig. 8: Common words in student essays and LLM-generated texts are analyzed. The code counts and visualizes the most common words in both types of essays, offering potential linguistic patterns for classification.

A screenshot of a computer

Description automatically generated

Fig. 9: Data preparation for building the machine learning model is performed in this Fig.. This involves text data preprocessing and splitting the data into features and target variables.

A screenshot of a computer

Description automatically generated

Fig. 10: The model architecture is defined and compiled in this section. It uses TensorFlow and Keras to create a deep learning model. The model includes layers for embedding, convolution, normalization, dropout, and more.

**RESULTS**

A screenshot of a computer screen

Description automatically generated

Fig. 11: The model is trained using the training data (X\_train and y\_train). It's validated against validation data (X\_valid and y\_valid) over five epochs. The code includes metrics such as accuracy and loss.

A screenshot of a computer

Description automatically generated

Fig. 12: Training and validation results are visualized in this Fig.. Two plots display model accuracy and loss over each epoch to monitor training progress.

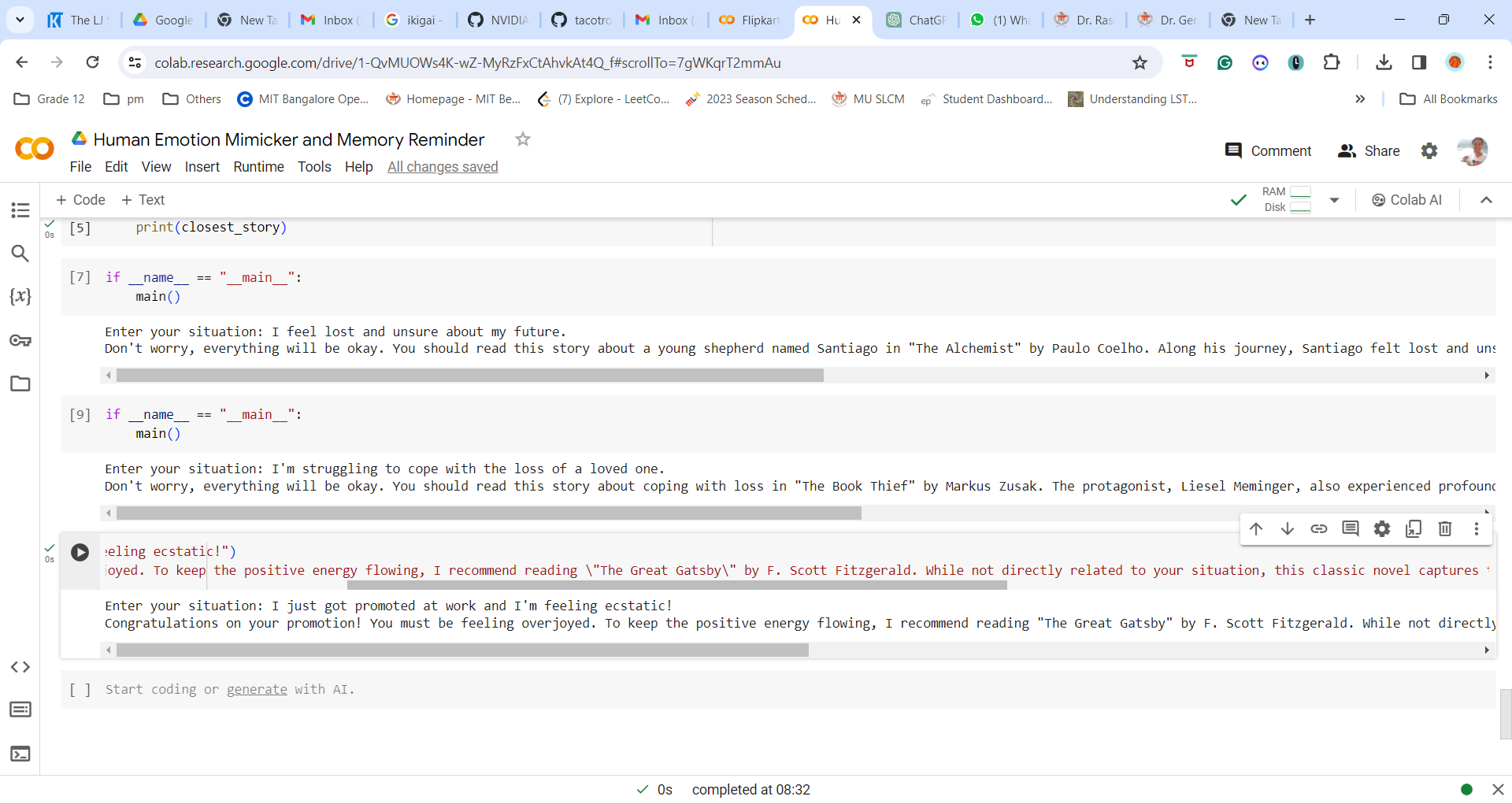


Fig. 13: Finally, this section prepares a submission file. It reads the test data, makes predictions using the trained model, and creates a submission file named submission.csv with corresponding id and generated values for each essay.

**CONCLUSION**

The AI Emotion Mimicker & Memory Reminder System represents a significant advancement in technology, offering users a unique and personalized experience. By leveraging advanced AI capabilities, robust architecture, and a user-centric design approach, this system redefines the way users interact with technology, enhancing their daily lives and emotional well-being.

**REFERENCES**

1. PySpark: <https://spark.apache.org/docs/latest/api/python/index.html>
2. TensorFlow: <https://www.tensorflow.org/>
3. Keras: <https://keras.io/>
4. Natural Language Processing: <https://en.wikipedia.org/wiki/Natural_language_processing>