Synopsis

on

Sentiment Analysis for Movie Review

to be developed to fulfill the requirements for

Major Project

Submitted to

Department of Computer Applications
CGC University, Mohali, Punjab

under the supervision of

Mr. Suraj Singh
Designation

Submitted by

Saurabh Singh	Pratyush Rawat	Abhishek	Badrinath
2424015	2424002	2423950	2423965
B	B	A	A

MCA 3rd Sem
(Batch 2024-26)

TABLE OF CONTENTS

Note: The synopsis must be at least 5 pages in length with atleast 5 reference in given format as per the given project guidelines. This instruction is provided for your convenience to ensure your submission meets the required standards. (Delete this line before taking printout)

1. Abstract

In the modern digital landscape, the exponential growth of user-generated content, particularly in the form of online movie reviews, makes manual assessment of public opinion impractical. This inability to quickly and accurately gauge audience sentiment can negatively impact filmmakers, marketers, and prospective viewers. This project, therefore, addresses the critical need for an automated, efficient, and highly accurate system to process and classify the sentiment expressed in textual movie reviews.

- **Need of the Project:** The project aims to overcome the manual effort and inherent subjectivity involved in analyzing a large volume of reviews, offering a fast, objective, and scalable solution for Opinion Mining.
- **Targeted Audience:** The system is beneficial to **Filmmakers and Production Houses** for measuring a movie's performance and identifying its strengths/weaknesses; **Critics** for augmenting their analysis; and **Moviegoers** who require a quick, aggregate understanding of a film's reception.
- **Methodology:** The methodology is a fusion of modern Deep Learning and Full-Stack Web Development. It involves training an advanced Natural Language Processing (NLP) model (e.g., LSTM or BERT) using the **IMDB Movie Review Dataset**. The model development, training, and deployment logic will be handled using **Python** and its extensive machine learning libraries (TensorFlow/Keras). The final, trained model

will be exposed via a RESTful API (e.g., using Flask or Django) and consumed by a modern, intuitive user interface developed using **React.js**.

2. Introduction

Sentiment Analysis is a cornerstone of Natural Language Processing (NLP), focusing on extracting subjective information and emotional tone from text. This chapter establishes the project's background, aiming to move beyond simple numeric ratings to derive a deeper, qualitative understanding of audience feedback on cinema.

The **operational environment** will be a cohesive full-stack architecture. The data and machine learning components reside in a **Python** environment, while the user interaction happens entirely on a responsive, single-page application built with **React.js**. This structure ensures the system is not only accurate but also practical and accessible.

2.1 Project Aim and Objectives

The primary **Aim** is to construct an end-to-end web application capable of accurately classifying movie reviews as either **Positive** or **Negative** sentiment, leveraging Deep Learning techniques for state-of-the-art performance.

The project **Objectives** include:

- To acquire, clean, and pre-process the **IMDB Movie Review Dataset** for binary classification.
- To implement and comparatively evaluate multiple Machine Learning and Deep Learning models (e.g., Bi-LSTM, BERT) using **Python** to select the best performer.
- To develop a RESTful API using a **Python** web framework (e.g., Flask) to serve the trained sentiment model.
- To design and implement a user-friendly frontend application using **React.js** for real-time submission of movie reviews and instantaneous display of the predicted sentiment.
- To achieve a final model classification accuracy exceeding 85% on the test set.

2.2 Technology to be Used

The project utilizes a robust technology stack optimized for both ML computation and modern web delivery.

Component	**Technology/Libraries**	**Purpose**
Backend/ML	**Python**	Core language for data processing, model training, and API creation.

Component	Technology/Libraries	Purpose
Deep Learning	**TensorFlow/Keras**	Framework for building and training Neural Networks (e.g., LSTM, CNNs).
Data Processing	Pandas, NumPy, NLTK	Libraries for data cleaning, text preprocessing, and feature engineering.
Frontend/GUI	**React.js**	Library for building the dynamic, single-page web application (User Interface).
API/Deployment	Flask/Django (Python)	Lightweight framework for creating the API that connects the React frontend to the Python ML model.
Database/Data Source	**IMDB Movie Review Dataset**	The core dataset for training and testing the model.

2.3 Hardware and Software Requirement

The following are the essential requirements for the project development and deployment:

Component	Minimum Requirement	Preferred Requirement
Processor	Dual-Core Processor (Intel i3/AMD equivalent)	Quad-Core Processor (Intel i5/AMD equivalent or better)
Operating System	Windows 10/11, Linux (Ubuntu), or macOS	Windows 10/11 (for ease of setup)
Memory (RAM)	8 GB RAM	16 GB RAM (Crucial for deep learning models)

Component	Minimum Requirement	Preferred Requirement
Hard Disk Space	100 GB Free Space (SSD Recommended)	256 GB Free SSD Space
Software Version	**Python 3.8+**, **Node.js 14+**, Web Browser	**Python 3.10+**, **Node.js 18+**, IDE (VS Code)
Other	Stable Internet Connection	GPU (Optional, for faster model training)

3. System Analysis

3.1 SRS (Software Requirement Specification) [36363636]

The proposed system, named **MovieSentimentAI**, is a web application with the following key functionalities:

Functional Requirements:

1. **Review Input Interface:** A simple text area on the **React.js** frontend must allow users to paste or type a movie review.
2. **Sentiment Classification:** The system must accept the review text, pass it to the **Python** API, and return the predicted sentiment (**Positive** or **Negative**).
3. **Confidence Score Display:** The system must display the model's prediction along with a confidence percentage (e.g., "Positive - 92% Confidence").
4. **Bulk Analysis:** The system should allow the upload of a .txt or .csv file containing multiple reviews, process them, and generate an overall sentiment summary (e.g., 70% Positive, 30% Negative).

Non-Functional Requirements:

1. **Performance:** The system must provide real-time classification, with a latency of less than 2 seconds for a single review.
2. **Accuracy:** The core sentiment model must maintain a classification accuracy of at least **85%** on the validation set.
3. **Scalability:** The architecture (React Frontend, Python Backend API) should allow for future scaling to handle multiple concurrent users.

4. **Usability:** The **React.js** user interface must be intuitive, responsive, and clearly display all results.

3.2 Existing v/s Proposed System

Feature	Existing System (Manual/Keyword-Based)	Proposed System (Deep Learning + Full-Stack)
Data Handling	Only handles a small volume of text.	Highly scalable, can process thousands of reviews.
Core Technology	Dictionary lookup, lexicon-based, or simple ML (e.g., Naive Bayes).	Advanced Deep Learning models (e.g., Bi-LSTM) for semantic understanding.
Accuracy	Prone to errors, struggles with sarcasm, idioms, and negation.	Higher accuracy by learning contextual relationships in the text.
User Interface	Often command-line based or basic scripts.	Modern, dynamic, and user-friendly interface using **React.js**.
Deployment	Difficult to deploy for public use.	Seamless web deployment via **Python** API/Web Server.

3.3 Software Tools to Be Used

- **Backend & ML:** Python (3.10+), TensorFlow 2.x, Keras, Scikit-learn, NLTK, Flask/Django.

- **Frontend:** React.js, HTML5, CSS3, JavaScript (ES6+).
- **Version Control:** Git & GitHub.
- **Development Environment:** VS Code, Jupyter Notebook/Google Colab.

4. Data Flow Diagram

Process	Description	Data Flow (Input → Output)
P1. Review Input	Captures the user-provided text via the React.js UI and sends it to the API.	**Review Text** $\rightarrow$ **Raw Review**
P2. Text Preprocessing	Cleans, normalizes, and converts the raw text into a numerical format (vector).	**Raw Review** $\rightarrow$ **Processed Text Vector (Token IDs)**
P3. Sentiment Classification	The Python/Keras model receives the vector and computes the sentiment probability.	**Processed Text Vector** $\rightarrow$ **Prediction Score/Label**
P4. Result Reporting	The Python API structures the result and sends it back to the React.js frontend for display.	**Prediction Score/Label** $\rightarrow$ **Sentiment Result**

Data Stores:

- **D1: Trained ML Model:** Stores the finalized, optimized Bi-LSTM model (a persistent file like an H5 file).
- **D2: Vocabulary/Word Index:** Stores the mapping of words to their numerical indices used by the model.

5. Timeline (Suggested)

Activity	Period 1 (19 Jul – 28 Aug 2025)	Period 2 (29 Aug – 22 Oct 2025)	Period 3 (23 Oct – Nov 2025)
Synopsis Submission	X		
Data Acquisition & Preprocessing (IMDB)	X		
ML Model Implementation & Training (Python/Keras)		X	
API Development (Flask) & Integration		X	X
Frontend Development (React.js)			X
System Testing & Performance Evaluation			X
Report Writing & Documentation			X
Final Submission & Presentation			X

6. References

(Using IEEE CITATION Style)

[1] R. T. S. V. K., "Sentiment Analysis in Action: A Case Study with Movie Reviews using NLP Techniques," *Medium/DataIlm*, 2024. [2] K. Aggarwal and V. Gupta, "SENTIMENT ANALYSIS FOR MOVIE REVIEWS," *Technical Review*, 2019. [3] J. A. V. S. S. S. B. A. L., "Sentiment Analysis Of IMDB Movie Reviews," *DiVA portal*, 2024. [4] A. J. L. R. C. V. S. V. D. N., "Sentiment Analysis from Movie Reviews Using LSTMs," *International Journal of Information Systems and Engineering*, vol. 24, no. 1, pp. 19-27, 2024. [5] A. B. K. S. E., "Sentiment Analysis for Movie Reviews," *CSE255: Data Mining and Predictive Analytics*, 2015.