**SeriniBot**

**Your Caring Mental Health Companion**

A PROJECT REPORT

BY

ABHIK GHOSH (E23CSEU0504)A blue and red text on a black background

Description automatically generated

SUBMITTED TO

SCHOOL OF COMPUTER SCIENCE ENGINEERING AND TECHNOLOGY, BENNETT UNIVERSITY

GREATER NOIDA, 201310, UTTAR PRADESH, INDIA

April 2025

# DECLARATION

I/We hereby declare that the work which is being presented in the report entitled “SeriniBot”, is an authentic record of my/our own work carried out during the period from JAN, 2023 to April, 2023 at School of Computer Science and Engineering and Technology, Bennett University Greater Noida.

The matters and the results presented in this report has not been submitted by me/us for the award of any other degree elsewhere.

Signature of Candidate

Abhik Ghosh

(Enroll. No. E23CSEU0504)

# ACKNOWLEDGEMENT

I/We would like to take this opportunity to express my/our deepest gratitude to my/our mentor, **Dr. Pooja Choudhary** for guiding, supporting, and helping me/us in every possible way. I/we was/were extremely fortunate to have him as my/our mentor as he provided insightful solutions to problems faced by me/us thus contributing immensely towards the completion of this capstone project. I/We would also like to express my/our deepest gratitude to VC, DEAN, HOD, faculty members and friends who helped me/us in successful completion of this capstone project. Any other name you can mentioned here. (Acknowledgement is your personal view, so you can write it in your way by maintaining integrity of technical report).

Signature of Candidate

Abhik Ghosh

(Enroll. No. E23CSEU0504)

**TABLE OF CONTENTS**

[ABSTRACT vi](#_Toc195903782)

[1. INTRODUCTION 1](#_Toc195903783)

[2. Project Objectives 2](#_Toc195903784)

[3. PREVIOUS WORKS 3](#_Toc195903785)

[4. System Architectur 3](#_Toc195903786)

[4.1. Overall Design 3](#_Toc195903787)

[4.2. Frontend-Backend Communication 4](#_Toc195903788)

[5. Data Collection and Preprocessing 4](#_Toc195903789)

[5.1. Data Collection 4](#_Toc195903790)

[1.1. Preprocessing Steps 4](#_Toc195903791)

[6. Model Training Pipeline 5](#_Toc195903792)

[6.1. Dataset and Data Module 5](#_Toc195903793)

[6.2. Model Architecture 5](#_Toc195903794)

[6.3. Training Setup and Execution 6](#_Toc195903795)

[7. API Development and Integration 7](#_Toc195903796)

[7.1. API Structure 7](#_Toc195903797)

[7.2. Endpoints 7](#_Toc195903798)

[8. TRAINING and Results 8](#_Toc195903799)

[8.1. Accuracy Metrics 8](#_Toc195903800)

[8.2. Obtained Results 8](#_Toc195903801)

[8.3. Interpretation and Next Steps 9](#_Toc195903802)

[9. Challenges and Limitations 10](#_Toc195903803)

[9.1. Data Quality and Diversity 10](#_Toc195903804)

[9.2. Model Interpretability 10](#_Toc195903805)

[9.3. Ethical and Privacy Concerns 10](#_Toc195903806)

[9.4. Resource Management 10](#_Toc195903807)

[10. Future Work 11](#_Toc195903808)

[10.1. Multimodal Integration 11](#_Toc195903809)

[10.2. Continuous Learning and Feedback 11](#_Toc195903810)

[10.3. Enhanced Interpretability 11](#_Toc195903811)

[10.4. Telehealth Integration 11](#_Toc195903812)

[10.5. Security and Scalability 11](#_Toc195903813)

[11. Conclusion 12](#_Toc195903814)

[12. References 12](#_Toc195903815)

ABSTRACT

SeriniBot – Your Caring Mental Health Companion – is a comprehensive AI system designed to process user text and detect depression indicators using advanced deep learning and NLP techniques. The project implements a robust model training pipeline based on PyTorch Lightning with checkpointing, early stopping, and mixed precision for optimized performance. A FastAPI-based RESTful API then serves the trained model, enabling real-time, empathetic responses to user inputs within a conversational chatbot interface.

1. INTRODUCTION

Depression is a widespread mental health issue that impacts millions of individuals globally. Many people struggle with accessing traditional therapy due to high costs, social stigma, and other barriers. In response, technological innovations, especially in artificial intelligence, have paved the way for supplementary mental health solutions that offer preliminary support.

SeriniBot is conceptualized as a caring companion that engages users in conversational interactions to help detect early signs of depression. By analysing linguistic cues and subtle emotional signals from user input, the chatbot provides guidance and directs users toward professional help when needed. This project brings together advanced NLP techniques and deep learning models to deliver immediate and personalized mental health support.

The integration of SeriniBot in digital health platforms offers a new paradigm for accessible care. The system is designed to be user-friendly and scalable, ensuring that it can serve a broad audience while maintaining high standards of accuracy and confidentiality. This report details the technical journey from data preprocessing through model training to API deployment.

1. Project Objectives

The primary objective of SeriniBot is to develop an intelligent system that processes text-based inputs from users and accurately identifies markers of depression. Through this, the chatbot can provide real-time, empathetic responses that offer preliminary mental health support. It does so by leveraging a deep learning model specifically tailored for sentiment and emotional analysis.

In addition to depression detection, the project aims to build a highly efficient and reproducible model training pipeline. This includes setting up robust data preprocessing, implementing a custom dataset and data module, and configuring training with advanced techniques such as mixed precision and early stopping. Moreover, the project seeks to deploy the model through a FastAPI RESTful service that ensures low-latency predictions and secure handling of sensitive user inputs.

Another key objective is to ensure ethical usage and user privacy. SeriniBot is intentionally designed as a supplementary tool, not a replacement for professional therapy, and all user data is anonymized. The architecture is modular and designed for future improvements, such as integrating multimodal inputs (e.g., voice data) and continuous learning mechanisms based on user feedback.

1. PREVIOUS WORKS

Numerous studies underscore the potential of AI in providing mental health support. Research by Fitzpatrick et al. (2017) demonstrated that AI-driven conversational agents can effectively deliver CBT-based interventions, reducing depressive symptoms in users. Similarly, insights from Inkster et al. (2018) illustrate that digital mental health interventions can bridge the accessibility gap, providing immediate support for those in need.

Recent advancements in deep learning—especially transformer-based architectures like DistilRoBERTa—have greatly improved the ability to capture context and sentiment in user-generated content. These models are adept at identifying subtle linguistic cues that may indicate depression. However, ethical and interpretability challenges persist, as deep learning models can sometimes act as “black boxes,” making their decision processes less transparent.

Comparative analyses of NLP frameworks have shown that careful preprocessing and model selection are crucial for success in this domain. Furthermore, research emphasizes the need for continuous improvement and ethical considerations when deploying AI for mental health, ensuring that such systems remain supportive, non-judgmental, and confidential. These findings inform the design decisions made for SeriniBot.

1. System Architectur
   1. Overall Design

SeriniBot’s architecture is divided into two major components: the model training pipeline and the inference (API serving) pipeline. The training pipeline is responsible for preprocessing data, training the deep learning model, and optimizing it using state-of-the-art techniques. By leveraging PyTorch Lightning, the training process is made modular and efficient, incorporating callbacks for early stopping, checkpointing, and mixed precision training.

The inference pipeline is implemented via FastAPI, which offers a lightweight, asynchronous web server for deploying the model. This API layer receives user text input, processes it by tokenizing and normalizing the data, and then passes it through the trained model to generate a prediction. The result is then returned to the frontend in real time, ensuring that users experience a smooth, responsive interaction.

* 1. Frontend-Backend Communication

On the frontend, users interact with a chatbot interface designed to simulate natural conversation. This interface, built with modern web technologies, sends asynchronous HTTP requests to the backend API. The API processes the request, runs the text through the deep learning model, and responds almost instantaneously.

This decoupled architecture not only enhances scalability but also allows the system to handle high volumes of traffic efficiently. In production, additional measures such as load balancing and container orchestration (e.g., using Kubernetes) can be implemented to further improve response times and ensure robustness during peak usage. By adhering to industry best practices, the communication between frontend and backend in SeriniBot is both secure and efficient.

1. Data Collection and Preprocessing
   1. Data Collection

Data is gathered from various online sources that document authentic user experiences. This includes forum posts and social media content that provide rich, diverse examples of language that may indicate depression. The dataset is then cleaned and curated to ensure that the inputs are representative of real-world scenarios. It is crucial that the data is both balanced and comprehensive to train a model that can generalize well.

* 1. Preprocessing Steps

Preprocessing is essential for transforming raw text into a format suitable for deep learning. The process involves several stages:

* **Cleaning and Normalization:** Removing extraneous punctuation and special characters, and normalizing text (e.g., converting to lowercase) to ensure consistency.
* **Tokenization:** Using the RobertaTokenizer, text is split into tokens while ensuring that special tokens are added. The text is padded or truncated to a fixed length (here, 128 tokens) to maintain a uniform input size.
* **Vectorization:** The tokenized data is converted into numerical tensors, which the model can process. This includes generating input IDs and attention masks, which indicate the importance of each token in understanding the context.

Finally, the dataset is split into training and validation sets using stratified sampling, ensuring that both sets maintain the same label distribution. This careful and thorough preprocessing significantly contributes to the model’s performance and reliability.

1. Model Training Pipeline
   1. Dataset and Data Module

A custom PyTorch dataset, named DepressionDataset, is designed specifically for processing and preparing textual data for deep learning. This dataset class employs the Roberta tokenizer to encode the raw text into token IDs, attention masks, and labels. The encoding process adds special tokens, applies padding and truncation, and converts the inputs into a format suitable for the model.

In addition, a PyTorch Lightning DataModule (named DepressionDataModule) is implemented to automate data preparation. The DataModule splits the dataset into training and validation subsets, constructs DataLoaders for each, and supports multi-worker data loading to optimize throughput during training. This modular approach not only simplifies the training loop but also enhances the reproducibility and scalability of the model training process.

* 1. Model Architecture

The core of SeriniBot’s training pipeline is the DepressionClassifier, a PyTorch Lightning module that builds on the pre-trained distilroberta-base model for sequence classification. This module is carefully designed with the following key features:

* **Forward Pass:** It processes tokenized input by feeding it through the transformer model and computes a loss if labels are provided.
* **Training and Validation Steps:** Both steps compute the model’s loss and update accuracy metrics using TorchMetrics. Predictions are generated by taking the argmax of output logits.
* **Epoch-End Hooks:** After each epoch, the training and validation accuracies are computed and printed, providing immediate feedback on model performance.
* **Optimizer and Scheduler:** AdamW is used as the optimizer, with a linear learning rate scheduler that includes a warmup period. The scheduler is dynamically configured based on the number of training steps per epoch and the total number of epochs.

By using PyTorch Lightning, the model code is simplified, modularized, and easily extendable. This structure enables rapid experimentation and efficient resource utilization, particularly when employing mixed precision training to reduce memory load and speed up training.

* 1. Training Setup and Execution

The training script begins with setting random seeds for reproducibility and clearing the GPU cache. The dataset (loaded from a CSV file such as reddit\_depression\_dataset.csv) is cleaned, and any rows with missing values are dropped to ensure high-quality data.

Once the data is preprocessed, the tokenizer is initialized, and the DataModule is set up to split the data and prepare DataLoaders. Training parameters, including batch size and maximum token length, are defined early on to ensure consistency during training.

A PyTorch Lightning Trainer is then configured with specific settings: GPU acceleration, mixed precision (“bf16-mixed”), and callbacks for early stopping and checkpointing. The training process is profiled using SimpleProfiler to capture performance metrics. Additionally, code is provided to resume training from a checkpoint, underscoring the system’s modularity and fault tolerance.

After training, the best model is saved as a state dictionary, making it straightforward to load the model for inference. This separation of training and serving phases is critical for maintaining both development efficiency and production readiness.

1. API Development and Integration

The backend API for SeriniBot is implemented using FastAPI, a modern, high-performance web framework that simplifies RESTful API creation. The API serves as the gateway for the trained model, enabling real-time inference when users interact with the chatbot.

* 1. API Structure

At the core of the API is an asynchronous lifespan event handler that manages resource initialization and cleanup. During startup, the handler loads the model (from a checkpoint if available) and moves it to the appropriate device (GPU or CPU). This ensures that the model is ready for inference when the API begins processing requests. On shutdown, resources are released gracefully.

Dependency injection is used to ensure that the model is available in API endpoints. A helper function checks the model’s status and returns a service unavailable error (503) if the model is not loaded, which provides robustness and clear error messaging.

* 1. Endpoints

The API exposes several endpoints:

* **/predict:**  
  This POST endpoint accepts a JSON payload containing user text. The input is tokenized using the same preprocessing steps as during training, and a forward pass is executed under torch.no\_grad() for efficient inference. The output logits are processed, and a prediction (“Depressed” or “Not Depressed”) is returned.
* **/health:**  
  A simple GET endpoint that performs a health check, verifying that the model is loaded and operational. This endpoint is critical for monitoring and maintaining the service.
* **/examples/request and /examples/response:**  
  These endpoints provide sample request and response models, enhancing the auto-generated Swagger documentation. This aids external developers in understanding the API’s expected behavior and data formats.

Security measures include the use of CORS middleware, initially configured with permissive settings for development. In production, these settings should be tightened to restrict access to trusted domains and ensure secure data handling.

1. TRAINING and Results

The DepressionClassifier was trained using PyTorch Lightning’s Trainer, configured to run for a single epoch (max\_epochs=1), employ mixed‑precision (bf16) for efficient GPU utilization, and integrate callbacks for early stopping and checkpointing. Setting max\_epochs=1 ensures that training halts immediately after one full pass through the dataset, as specified in the Trainer configuration [Lightning AI](https://lightning.ai/docs/pytorch/stable/common/trainer.html?utm_source=chatgpt.com). The EarlyStopping callback monitored the validation loss at the end of each validation epoch, stopping early if no improvement was observed over a predefined patience window [Lightning AI](https://lightning.ai/docs/pytorch/stable/common/early_stopping.html?utm_source=chatgpt.com). Mixed‑precision (“bf16‑mixed”) was used to accelerate matrix operations and reduce memory consumption, following best practices for modern GPU training [PyTorch Lightning Documentation](https://pytorch-lightning.readthedocs.io/en/1.5.10/api/pytorch_lightning.trainer.trainer.html?utm_source=chatgpt.com).

A simple profiler was enabled to record timing statistics for critical training events—such as batch fetching, forward passes, backward passes, and optimizer steps—and printed a concise report after training completed [PyTorch Lightning Documentation](https://pytorch-lightning.readthedocs.io/en/1.5.10/advanced/profiler.html?utm_source=chatgpt.com). Profiling revealed the relative cost of each component of the training loop (e.g., data loading vs. model computation), guiding potential optimizations for future iterations [PyTorch Lightning Documentation](https://pytorch-lightning.readthedocs.io/en/1.6.5/advanced/profiler.html?utm_source=chatgpt.com).

* 1. Accuracy Metrics

Two primary metrics were tracked throughout training: **training accuracy**, which measures model performance on the data used for weight updates, and **validation accuracy**, which assesses generalization on a held‑out dataset [Stack Overflow](https://stackoverflow.com/questions/51344839/what-is-the-difference-between-the-terms-accuracy-and-validation-accuracy?utm_source=chatgpt.com). A validation set was carved out via stratified splitting to ensure representative class distributions, providing a reliable gauge of how the model might perform on unseen data [Wikipedia](https://en.wikipedia.org/wiki/Training%2C_validation%2C_and_test_data_sets?utm_source=chatgpt.com). Monitoring both metrics guards against overfitting—where training accuracy climbs while validation accuracy stagnates or declines—and underfitting, where both accuracies remain low [Data Science Stack Exchange](https://datascience.stackexchange.com/questions/96409/cnn-training-accuracy-vs-validation-accuracy?utm_source=chatgpt.com).

* 1. Obtained Results

At the conclusion of the single training epoch, the model achieved a **validation accuracy of 0.9487** and a **training accuracy of 0.9385**, indicating that it generalizes slightly better to the unseen validation set than it does to the training set. This phenomenon—validation accuracy exceeding training accuracy—can occur when regularization, data augmentation, or dropout introduces beneficial noise during training, thereby improving generalization on the clean validation data [Data Science Stack Exchange](https://datascience.stackexchange.com/questions/96409/cnn-training-accuracy-vs-validation-accuracy?utm_source=chatgpt.com). Such a result suggests that SeriniBot’s model has learned robust features without simply memorizing the training examples.

Although a full detailed classification report (precision, recall, F1‑score) was not generated at this stage, preliminary accuracy metrics suggest performance on par with or exceeding typical transformer‑based text classifiers [Scikit-learn](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html?utm_source=chatgpt.com). Future work will include generating comprehensive per‑class metrics to further elucidate strengths and weaknesses in class predictions.

* 1. Interpretation and Next Steps

The high validation accuracy (≈ 95%) demonstrates strong discriminative power in detecting depressive language patterns. Given the limited difference between training and validation accuracies, the model shows minimal signs of overfitting, indicating that the current architecture and regularization strategies are effective. However, continuous monitoring and additional epochs with early stopping would further confirm stability, and profiling insights highlight areas—such as data loading or backward computation—that could benefit from optimization [PyTorch Lightning Documentation](https://pytorch-lightning.readthedocs.io/en/1.5.10/advanced/profiler.html?utm_source=chatgpt.com).

Going forward, integrating more granular evaluation methods (e.g., ROC‑AUC, confusion matrices) and expanding the training dataset with diverse conversational examples will help refine SeriniBot’s sensitivity and specificity. Such enhancements will ensure the chatbot remains a reliable “Caring Mental Health Companion” capable of supporting users with nuanced, context‑aware feedback.

1. Challenges and Limitations
   1. Data Quality and Diversity

One of the primary challenges is ensuring that the dataset accurately captures the nuances of depression-related language. Social media and forum data can be noisy, and language is highly variable. While preprocessing and stratification help mitigate these issues, obtaining a well-curated dataset remains an ongoing effort.

* 1. Model Interpretability

Deep learning models, particularly transformers, operate as “black boxes.” This lack of transparency can be a barrier when explaining the decision-making process to clinicians or users. Although the model achieves high accuracy, its interpretability is limited, and additional techniques like attention visualization may be required to enhance trust and transparency.

* 1. Ethical and Privacy Concerns

Handling sensitive mental health data necessitates strict privacy measures and ethical guidelines. SeriniBot anonymizes user inputs and provides disclaimers that it is a supplementary support tool, not a replacement for professional care. Continuous evaluation and adherence to data privacy regulations are required to maintain user trust and avoid potential misuse of sensitive information.

* 1. Resource Management

Deploying deep learning models in real-time, especially under mixed precision and GPU constraints, requires careful resource management. The system is optimized for performance through efficient DataLoader usage and proper device handling, but scaling the solution for high-traffic environments may require additional optimizations and potentially the use of container orchestration solutions like Kubernetes.

1. Future Work
   1. Multimodal Integration

Future iterations of SeriniBot may incorporate additional data modalities such as voice input and facial recognition to provide a more holistic analysis of user emotional states. This multimodal approach could enhance the model’s ability to detect subtle signs of distress and offer more personalized responses.

* 1. Continuous Learning and Feedback

Integrating a mechanism to collect user feedback on predictions could allow for periodic retraining of the model. By incorporating a continuous learning loop, SeriniBot can evolve over time and maintain high accuracy as language patterns and user behavior change.

* 1. Enhanced Interpretability

Developing tools to visualize attention maps or to generate explanations for model predictions is a promising area of future work. Enhanced interpretability will not only improve clinician trust but also empower users to understand the basis of the chatbot’s recommendations.

* 1. Telehealth Integration

An important future direction is integrating SeriniBot with telehealth platforms. This would facilitate seamless referrals to licensed mental health professionals when the chatbot detects severe distress. Integrating scheduling systems and secure messaging can bridge the gap between preliminary support and professional care.

* 1. Security and Scalability

As the system transitions from development to production, enhancing security measures is crucial. This includes restricting CORS settings, implementing rate limiting, and possibly leveraging cloud-native solutions to ensure that the system scales to meet high demand while remaining secure.

1. Conclusion

SeriniBot – Your Caring Mental Health Companion – represents a meaningful advancement in AI applications for mental health support. By combining a sophisticated deep learning model with a user-friendly API powered by FastAPI, the system provides immediate and empathetic feedback for users potentially experiencing depression. The project’s well-structured training pipeline, leveraging PyTorch Lightning, facilitates efficient model development through techniques like mixed precision and checkpointing.

While challenges remain in areas such as data quality, model interpretability, and ethical deployment, SeriniBot lays a solid foundation for further developments. Future enhancements such as multimodal data integration, continuous learning loops, and telehealth connectivity have the potential to further enrich the system’s capabilities. Overall, SeriniBot exemplifies how AI can be thoughtfully employed to create compassionate mental health interventions, offering support where it is most needed.

1. References

* Fitzpatrick, K.K., et al. (2017). *Evaluating a fully automated conversational agent delivering cognitive behavioral therapy.*
* Inkster, B., et al. (2018). *Digital mental health interventions: opportunities and challenges.*
* Huang, et al. (2020). z*Deep learning for sentiment analysis in mental health applications.*
* Fusar-Poli, P., et al. (2018). *The Science of Prognosis in Psychiatry: A Review.* JAMA Psychiatry.
* FastAPI Documentation. Available at: FastAPI
* PyTorch Lightning Documentation. Available at: [Lightning](https://lightning.ai/)
* Additional tutorials and case studies on deploying NLP models via FastAPI and Hugging Face Transformers [medium.com](https://medium.com/%40aminajavaid30/text-classification-api-with-fastapi-docker-and-hugging-face-transformers-088bc8d8e8c5).