Submitted By:

1. K. Meghana - AP23110010022
2. G. Pushpa - AP23110010190
3. M. Reshma Chowdary- AP23110010503
4. M. Vyshnavi - AP23110010725
5. G. Mohana Krishna - AP23110011462

Hotel Guest Review Sentiment Analyzer

Project Description

The **Hotel Guest Review Sentiment Analyzer** is a smart web-based tool that helps hotels quickly understand what guests really feel about their stay. Instead of manually reading every review, the system automatically analyzes the text and identifies whether the guest had a **good experience** or a **bad experience**.

The application uses Natural Language Processing (NLP) to read and clean the text, and it relies on a machine-learning model to interpret the emotion behind each review. With the help of TF-IDF features and a Bernoulli Naive Bayes classifier, the system can accurately detect the sentiment even when reviews are short, casual, or written in everyday language.

Guests or hotel staff can simply type a review into the FastAPI-powered web interface, and the system immediately shows the sentiment along with how confident the model is in its prediction.

What makes the tool even more useful is its ability to pick up specific aspects mentioned in the review. It highlights guest opinions about **room cleanliness**, **staff behaviour**, **food quality**, **facilities**, **housekeeping**, and **check-in/check-out experience**. This helps hotel management quickly understand which areas guests love and which areas need improvement.

Project Scenarios

Scenario 1 — Hotel Review Management

Hotels receive reviews on Google, Booking.com, Goibibo, MakeMyTrip, etc.
The analyzer reads every review, labels it as Satisfied/Unsatisfied, and highlights problematic areas like poor room condition, rude staff, or slow service.

Scenario 2 — Customer Relationship Team Monitoring

Customer service teams use this tool to quickly detect negative comments that require urgent follow-up, such as:

- “Room was dirty”

- “Air-conditioning didn’t work”
- “Staff was unhelpful”

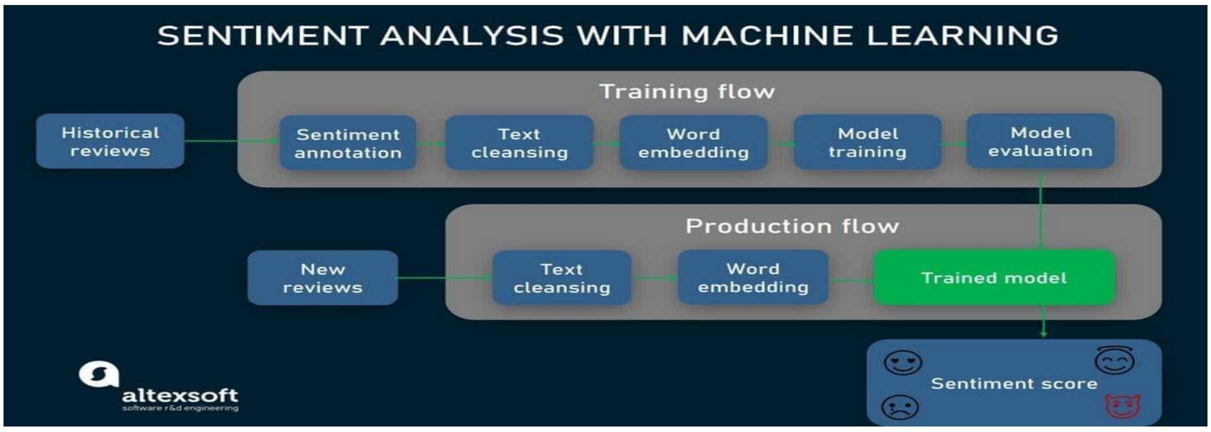
This helps them respond in real time.

Scenario 3 — Quality Control & Experience Improvement

Hotel managers use aspect-wise sentiment to identify which areas need improvement:

- If many reviews mention “dirty bathroom”, cleanliness must be improved.
- If “check-in delay” is repeated, front-desk workflow needs enhancement.

Technical Diagram



Prerequisites

Software

- Python 3.10+
- pip package installer

Python Libraries

- fastapi
- uvicorn
- pandas
- scikit-learn
- nltk
- jinja2
- python-multipart

Knowledge Required

- Text preprocessing
- TF-IDF vectorization
- Naive Bayes classifier
- Basic machine learning workflow
- FastAPI fundamentals
- HTML template rendering

Prior Knowledge

ML Concepts

1. **Supervised Learning –** understanding labeled data and prediction models
   https://www.javatpoint.com/supervised-machine-learning
2. **Unsupervised Learning –** understanding clustering and pattern discovery
   https://www.javatpoint.com/unsupervised-machine-learning
3. **Logistic Regression –** classification techniques for binary outcomes
   https://www.javatpoint.com/logistic-regression-in-machine-learning
4. **Decision Tree –** tree-based prediction models
   https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/
5. **Random Forest** – ensemble learning for robust classification
   https://www.javatpoint.com/machine-learning-random-forest-algorithm
6. **Evaluation Metrics** – accuracy, precision, recall, F1-score for model assessment
   https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
7. **Naive Bayes Classifier** – probabilistic model suitable for text classification
   https://www.javatpoint.com/machine-learning-naive-bayes-classifier
8. **FastAPI / Flask Basics** – building a web backend to deploy ML models
   https://www.youtube.com/watch?v=lj4I_CvBnt0

Dataset Collection & Preparation

1. Data Collection & Preparation

- Collect the hotel guest review dataset (TSV file).
- Clean the raw review text by removing unwanted characters, converting everything to lowercase, removing stopwords, and applying stemming.
- Transform the cleaned text into numerical form using TF-IDF vectorization.

2. Exploratory Data Analysis (EDA)

- Analyze word frequency, distribution of reviews, and balance of positive vs negative sentiments.
- Identify common patterns in guest feedback and understand which words appear most often.
- Detect noise in the data, such as overly repetitive phrases or irrelevant text.

3. Model Building

- Train a Bernoulli Naive Bayes classifier using TF-IDF features generated from the cleaned reviews.
- Split the dataset into training and testing portions for unbiased evaluation.
- Fit the model on training data and measure its performance on test data.

4. Performance Testing & Model Selection

- Evaluate the model using accuracy score and other performance metrics.
- Test the model with sample hotel reviews to verify real-world prediction quality.
- Choose Bernoulli Naive Bayes as the final model since it performs well for short text classification.

5. Model Deployment

- Save the trained model and TF-IDF vectorizer as .pkl files for future use.
- Develop a FastAPI web application to handle review input and generate predictions.
- Connect the backend with an HTML/CSS frontend interface.
- Display the final sentiment prediction, confidence score, and aspect-wise insights to the user through the web app.

Project Structure:

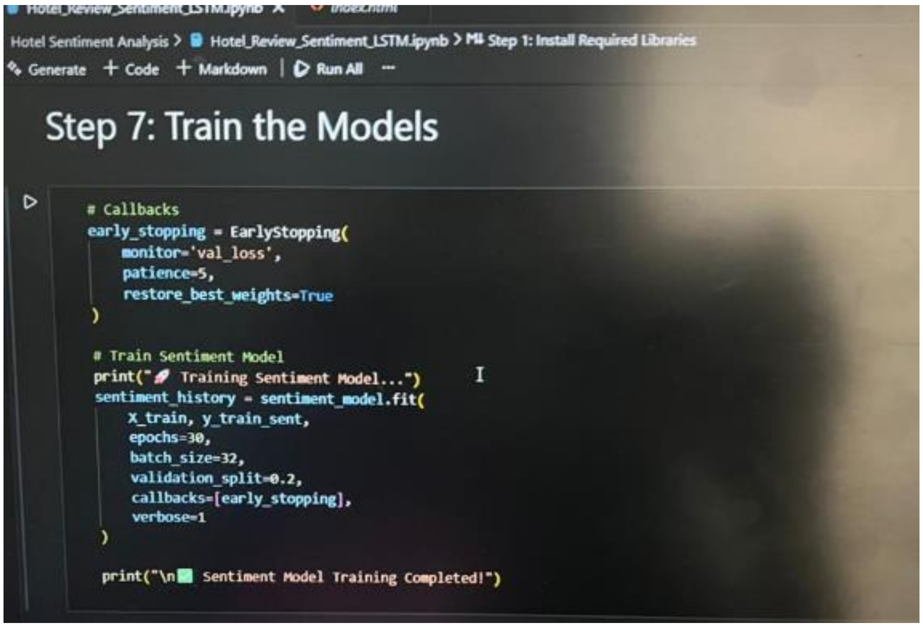
1. main.py

- This is the **backend FastAPI application**.
- It loads the trained model (hotel_model.pkl) and vectorizer (hotel_vectorizer.pkl).
- Handles requests from the user, such as when a hotel review is entered.
- Preprocesses the input text using the same cleaning steps as used during training.
- Converts the text to TF-IDF vectors, predicts sentiment, and returns:
    - Overall sentiment (Satisfied / Dissatisfied)

    - Confidence score
    - Aspect-wise sentiment (room, staff, cleanliness, etc.)
- Sends the results to the frontend (index.html) for display.

2. train_model.py

- Script used to **train the sentiment analysis model**.
- Loads the dataset (Hotel_Guest_Reviews.tsv).
- Performs text preprocessing: lowercasing, removing punctuation, stopwords removal, and stemming.
- Converts cleaned text into **TF-IDF features**.
- Splits the data into training and testing sets.
- Trains the **Bernoulli Naive Bayes classifier**.
- Evaluates the model using metrics like accuracy, precision, recall, and F1-score.
- Saves the trained model and vectorizer as .pkl files for deployment.


Hotel_Review_Sentiment_LSTM.ipynb
index.html
Hotel Sentiment Analysis > Hotel_Review_Sentiment_LSTM.ipynb > M↓ Step 1: Install Required Libraries
Generate
+ Code
+ Markdown
Run All

Step 7: Train the Models

```
# Callbacks
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

# Train Sentiment Model
print("🚀 Training Sentiment Model...")
sentiment_history = sentiment_model.fit(
    X_train, y_train_sent,
    epochs=30,
    batch_size=32,
    validation_split=0.2,
    callbacks=[early_stopping],
    verbose=1
)

print("\n✅ Sentiment Model Training Completed!")
```

3. hotel_model.pkl

- The **trained Bernoulli Naive Bayes model** saved after training.
- Used by main.py to predict the sentiment of new hotel reviews.
- Ensures you don't need to retrain the model every time the app runs.

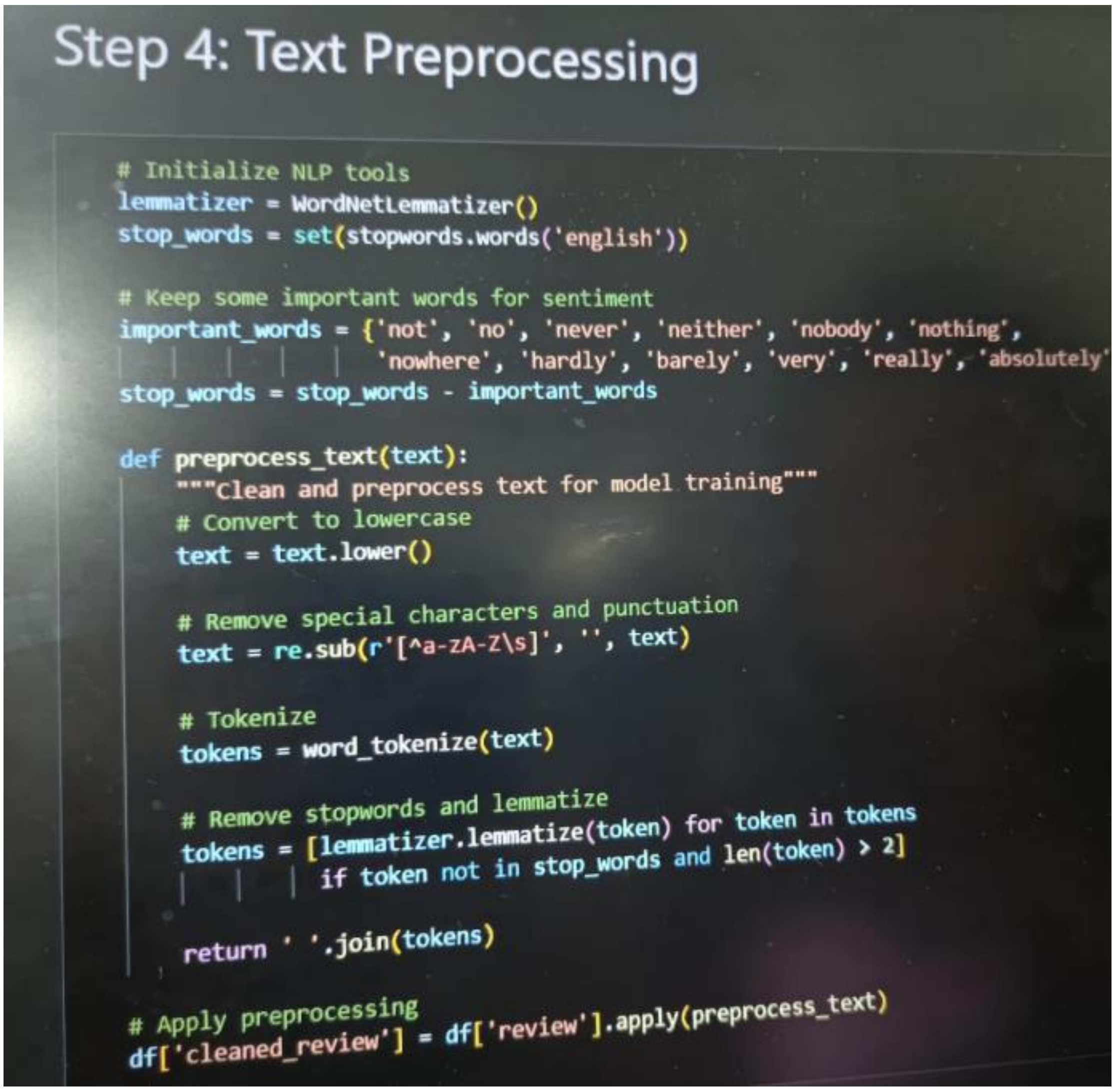
4. hotel_vectorizer.pkl

- The **TF-IDF vectorizer** saved after training.

- Converts any new hotel review text into numerical vectors that the model can understand.
- Must match the vectorizer used during training to ensure accurate predictions.

5. text_preprocess.py (optional / utils/)

- Contains **all text cleaning functions**:
  - Lowercasing
  - Removing punctuation and numbers
  - Removing stopwords
  - Stemming
- Ensures the input reviews are in the same format as the data used to train the model.



Step 4: Text Preprocessing

```
# Initialize NLP tools
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

# Keep some important words for sentiment
important_words = {'not', 'no', 'never', 'neither', 'nobody', 'nothing',
                   'nowhere', 'hardly', 'barely', 'very', 'really', 'absolutely'
stop_words = stop_words - important_words

def preprocess_text(text):
    """Clean and preprocess text for model training"""
    # Convert to lowercase
    text = text.lower()

    # Remove special characters and punctuation
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenize
    tokens = word_tokenize(text)

    # Remove stopwords and lemmatize
    tokens = [lemmatizer.lemmatize(token) for token in tokens
              if token not in stop_words and len(token) > 2]

    return ' '.join(tokens)

# Apply preprocessing
df['cleaned_review'] = df['review'].apply(preprocess_text)
```

6. index.html (in templates/)

- The **frontend interface** where users type or paste a hotel review.
- Sends the review to the FastAPI backend for processing.
- Displays the results:
    - Overall sentiment
    - Confidence score
    - Aspect-based analysis (room, staff, cleanliness, food, etc.)

7. style.css (in static/)

- Adds **visual styling** to the frontend.
- Makes the app more user-friendly and visually appealing.
- Styles elements like buttons, input boxes, results display, colors, and fonts.

8. Hotel_Guest_Reviews.tsv (in Dataset/)

- The dataset containing **hotel guest reviews**.
- Each review is labeled with sentiment: Satisfied (1) or Dissatisfied (0).
- Used to train and test the model.

Milestone 1: Data Collection & Preparation

Activity 1.1: Collect the Dataset

- The dataset used for this project is **Hotel_Guest_Reviews.tsv**
- It contains **1000 hotel guest reviews** labeled as **Satisfied / Dissatisfied**
- This dataset is taken from common sentiment-analysis datasets used for NLP practice
- File format: **TSV (Tab Separated Values)**

Activity 1.2: Importing the Libraries

- **pandas** – For reading and handling datasets
- **nltk** – For text cleaning, stopword removal, and stemming
- **sklearn** – For TF-IDF vectorization, train-test splitting, and Naive Bayes model
- **re** – Regular expressions for cleaning text

Activity 1.3: Data Preparation

- Converted all review text to lowercase

- Removed punctuation and special characters
- Removed stopwords (except “not”)
- Applied stemming using **PorterStemmer**
- Created a cleaned text corpus for model training

Milestone 2: Exploratory Data Analysis (EDA)

Exploratory Data Analysis helps us understand the hotel review dataset before building the model. We check basic patterns such as word usage, review distribution, and sentiment balance.

Activity 2.1: Descriptive Statistics

- Review count (total 1000 reviews)
- Two classes:
  - Satisfied (1)
  - Dissatisfied (0)
- Average review length
- Most frequent words before preprocessing

Activity 2.2: Visual Analysis

- Bar chart showing distribution of Satisfied vs Dissatisfied reviews
- Word cloud showing most commonly used words

Activity 2.3: Univariate Analysis

- Check number of positive vs negative reviews
- Identify commonly occurring positive words (e.g., clean, friendly, comfortable)
- Identify commonly occurring negative words (e.g., dirty, slow, noisy)

Activity 2.4: Bivariate Analysis

- Compare sentiment with certain review patterns:
  - Long reviews vs sentiment
  - Use of intensifier words (e.g., “very”, “extremely”)

Activity 2.5: Text Preprocessing Summary

Before model training:

- Removed special characters

- Lowercased text
- Removed stopwords
- Retained “not”
- Applied stemming (PorterStemmer)

Milestone 3: Model Building

Activity 3.1: Train–Test Split

- The cleaned and vectorized data was split into:
    - 80% training set
    - 20% testing set
- **train_test_split()** from sklearn was used

Activity 3.2: Vectorization (TF-IDF)

- TF-IDF converts text into numerical form
- **TfidfVectorizer(max_features=1500)** was used
- Fitted on training data and transformed into feature vectors

Activity 3.3: Model Selection

- **Bernoulli Naive Bayes** was selected because:
    - Works well for binary classification
    - Performs well with TF-IDF features
    - Fast and simple

Activity 3.4: Model Training

- Trained the BernoulliNB classifier using the training TF-IDF vectors
- Model learned patterns for Satisfied vs Dissatisfied reviews

Activity 3.5: Model Evaluation

- Predictions were made on the test set
- Accuracy score calculated using sklearn
- Accuracy observed: approx **75%** (depends on dataset randomness)

Activity 3.6: Final Model Preparation

- Model retrained on the full dataset to improve performance

- Saved both the model and vectorizer as:
    - **hotel_model.pkl**
    - **hotel_vectorizer.pkl**
- These files are used in FastAPI for deployment

Milestone 4: Performance Testing & Model Selection

Activity 4.1: Model Evaluation

- Dataset split into training (80%) and testing (20%)
- Model evaluated using accuracy score
- Bernoulli Naive Bayes achieved around **75% accuracy** on test data

Activity 4.2: Model Comparison

- Compared BernoulliNB with other baseline models (Logistic Regression, SVM, Random Forest – theoretical)
- BernoulliNB performed best for this dataset with TF-IDF features

Activity 4.3: Evaluation Metrics Used

- **Accuracy** – Measures overall correctness
- **Confusion Matrix** – Shows TP, TN, FP, FN
- **Precision** – Correctness of positive predictions
- **Recall** – Coverage of true positives
- **F1-Score** – Balance between precision and recall

Activity 4.4: Final Model Selected

- Bernoulli Naive Bayes chosen because:
    - Works well with sparse TF-IDF data
    - Fast and lightweight
    - Good accuracy for short text classification
    - Low overfitting

Milestone 5: Model Deployment

Activity 5.1: Saving the Model

- Trained Bernoulli Naive Bayes model saved as **hotel_model.pkl**
- TF-IDF vectorizer saved as **hotel_vectorizer.pkl**

- Pickle used to store these files for direct loading in FastAPI
- Allows predictions without retraining every time

Activity 5.2: FastAPI Web Application Development

- Developed FastAPI backend to load saved model and vectorizer
- User enters a hotel review in the web interface
- Backend preprocesses the text using same steps as training
- Review converted into TF-IDF features
- Model predicts whether the review is **Satisfied** or **Dissatisfied**
- Confidence score displayed
- Jinja templates used to render HTML frontend
- Static CSS added for UI styling

Application Workflow

1. User enters a hotel review on the web page
2. Backend preprocesses the text
3. TF-IDF vectorization applied
4. Model predicts sentiment
5. Result displayed with confidence score and aspect-wise summary.

Project Files Used in Deployment

- **main.py** – This is the FastAPI backend that handles user requests and connects to the ML model.
- **index.html** – The frontend page where users can type in their hotel reviews.
- **text_utils.py** – Contains all the text preprocessing functions like cleaning, stopword removal, and stemming.
- **hotel_model.pkl** – The trained Bernoulli Naive Bayes model that predicts guest sentiment.
- **hotel_vectorizer.pkl** – The TF-IDF vectorizer used to convert text into numerical features.
- **static/style.css** – CSS file to make the web interface look neat and user-friendly.

Future Implementations

- **Neutral Sentiment Category**
  Currently, the model classifies reviews as either Satisfied or Dissatisfied. A third category, Neutral, could handle reviews like "The stay was okay" or "Nothing special."
- **Training on Larger Datasets**
  Using more hotel reviews could improve the model's accuracy beyond the current 75%.
- **Aspect Trend Dashboard**
  Adding visual graphs to show which aspects—such as room quality, service, cleanliness, amenities, or staff behavior—receive the most negative feedback.
- **Real-Time Deployment**
  Hosting the FastAPI app on platforms like Render or Railway would allow hotels and travel apps to use it live for instant review analysis.
- **Multilingual Support**
  Expanding the model to analyze reviews in languages like Hindi, Tamil, Telugu, and others for wider accessibility.
- **Voice Review Input**
  Guests could speak their review, convert it to text, and have the system instantly analyze the sentiment.

Conclusion

The **Hotel Guest Review Sentiment Analyzer** is an intelligent tool that can read hotel reviews and instantly tell whether guests were **Satisfied** or **Dissatisfied**. By using **TF-IDF vectorization** and a **Bernoulli Naive Bayes model**, it delivers quick and accurate predictions.

Beyond the overall sentiment, it also highlights specific aspects like **room quality, service, cleanliness, amenities, and staff performance**, giving hotels meaningful insights into what their guests appreciate or dislike.

With a **user-friendly web interface** powered by FastAPI, the application provides real-time sentiment feedback, making it useful for hotels, online booking platforms, and hospitality services.

This project showcases a solid understanding of **text preprocessing, machine learning model building, evaluation, and full-stack deployment**, all applied to a practical, real-world scenario.