IntelliReview –

AI-Based Review Summarization

and Rating Prediction

Using IBM Granite

**Submitted By** – Yash Anand

**Project Description:**

AI-Based Review Summarization and Rating Prediction is a comprehensive full-stack application developed to convert vast amounts of unstructured customer reviews into structured, insightful summaries. By utilizing advanced natural language understanding, the system identifies key sentiments and topics, generates concise and human-readable summaries, and predicts the likely rating (on a scale of 1 to 5 stars) associated with the review content. This enables businesses to monitor customer feedback at scale, streamline sentiment analysis, and make informed decisions based on real user experiences.

At the core of this project is a hybrid architecture that combines two powerful AI models: **IBM Granite 7B**, integrated via Hugging Face, and **Ollama’s Mistral model**, running locally for real-time fallback. The use of IBM Granite ensures high-quality summarization and enterprise-grade NLP capabilities, while the inclusion of Ollama provides flexibility and responsiveness in scenarios where the Granite model may be delayed or resource-intensive. This smart fallback mechanism ensures uninterrupted performance without compromising output quality. The hybrid setup thus allows the system to function efficiently in both research environments and production-grade applications.

Users can interact with the system in two primary modes: single-review input or bulk-upload via CSV. Each review is processed through the hybrid summarization engine, after which **IBM Watson Natural Language Understanding (NLU)** is used to analyze sentiment—classifying it as positive, neutral, or negative. The final output includes the original review, a summary, the predicted star rating, the sentiment classification, and a timestamp if available.

The frontend, developed using HTML, Tailwind CSS, and JavaScript, provides a sleek, intuitive interface. It not only displays individual summaries and ratings but also visualizes sentiment trends over time using interactive bar charts powered by Chart.js. For bulk reviews, the system groups and tracks sentiments by date, enabling deeper temporal analysis.

In addition, the system supports exporting summarized results as downloadable .txt files for reporting or archival purposes. This combination of automation, visual analytics, and export capability positions the application as a practical tool for organizations looking to enhance customer insight, quality control, and brand monitoring across industries such as hospitality, retail, and automotive services.

**Scenarios:**

**Scenario 1: Bulk Sentiment Monitoring for Hospitality Chain**

A regional hotel manager responsible for several locations is tasked with evaluating guest satisfaction across branches. They export and upload a CSV file filled with customer reviews gathered from different cities and timeframes. The system quickly summarizes each review, identifies the sentiment, predicts star ratings, and plots an interactive sentiment trend by date. Using the visual data, the manager identifies service gaps—such as declining satisfaction in one location during holidays. The summaries and sentiment heatmap empower the team to adjust strategies, improve staff training, and tailor guest experiences, ultimately enhancing the chain’s overall reputation and service standards.

**Scenario 2: Real-Time Feedback Analysis for an E-commerce Brand**

A business owner managing an online clothing store notices some customer frustration around a recent product. To investigate quickly, they paste a specific customer review into the application. Instantly, the system returns a concise summary, star rating prediction, and sentiment classification. The user identifies a trend: low ratings tied to sizing issues. With this insight, they revise product descriptions, update size guides, and respond to customers directly. The ability to analyze feedback in real-time enhances customer relations and product performance, allowing the brand to maintain credibility while actively improving based on consumer voice.

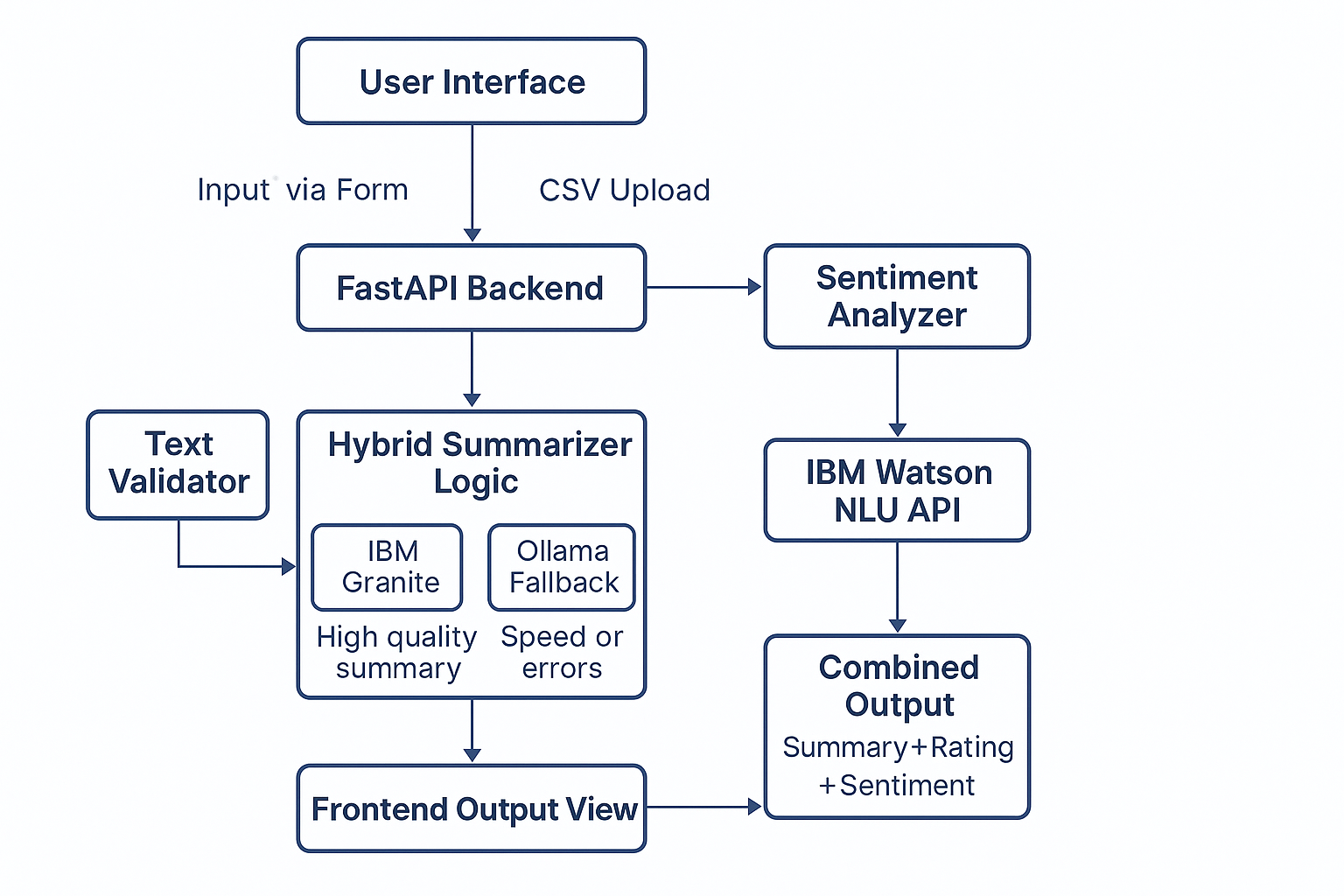
**Scenario 3: Automotive Service Review Analysis**

An automotive company gathers quarterly service feedback across its franchise outlets. To uncover service consistency and regional issues, they upload the CSV file containing all collected reviews into the application. The system summarizes each entry, classifies sentiment, predicts customer ratings, and displays an analytical chart for managerial reporting. This data is visualized to show spikes in dissatisfaction, particularly during specific timeframes or from certain service centers. Armed with these insights, leadership develops branch-wise strategies, reinforces quality control, and uses the generated summary report during internal stakeholder briefings to guide improvement policies.

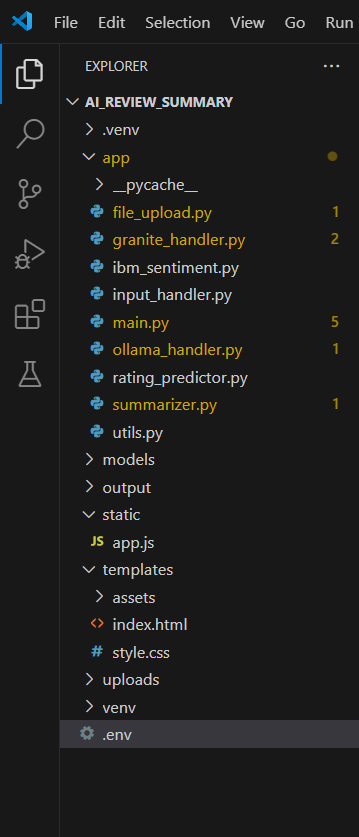
**Pre-requisites:**

* FastAPI Framework : [FastAPI Documentation](https://fastapi.tiangolo.com/)
* Hugging Face Transformers Library : [Huuging Face Library](https://huggingface.co/docs/transformers/en/index)
* Ollama Local LLM Setup : [Ollama Model Download](https://ollama.com)
* IBM Cloud Natural Language Understanding Service : [IBM Natural Language Documentation](https://cloud.ibm.com/apidocs/natural-language-understanding)
* Chart.js : [ChartJS Library](https://www.chartjs.org/)

**Technical Architecture:**



**Project Directory Structure:**



**app/ Folder:**

* **file\_upload.py** – Handles saving and parsing uploaded CSV files containing user reviews.
* **granite\_handler.py** – Loads and runs IBM Granite 7B model via Hugging Face for summarization.
* **ibm\_sentiment.py** – Connects to IBM Watson NLU API to perform sentiment analysis on reviews.
* **input\_handler.py** – Validates single-review text and processes CSV inputs for clean review extraction.
* **main.py** – Serves as the FastAPI application entry point and defines all core API routes.
* **ollama\_handler.py** – Interfaces with the Ollama Mistral model for summarization fallback.
* **rating\_predictor.py** – Provides a heuristic method to predict review ratings based on keywords (optional/legacy).
* **summarizer.py** – Orchestrates hybrid summarization using IBM Granite and Ollama logic.
* **utils.py** – Contains text cleaning and validation utilities used across modules.

**Other Key Folders/Files:**

* **app.js** – Manages frontend behavior, form interactions, API calls, and chart rendering.
* **index.html** – The main UI structure that hosts input forms, summary cards, and sentiment charts.
* **style.css** – Applies custom styling and transitions for a modern, responsive interface.
* **.env** – Stores sensitive credentials like Watson NLU API key and service URL.

**Project Overview:**

**Activity 1: Environment Setup and Model Architecture Design**

* **Activity 1.1:** Installed and configured IBM Watson Natural Language Understanding (NLU) service and generated API keys from IBM Cloud.
* **Activity 1.2:** Downloaded and integrated IBM Granite 7B model using Hugging Face snapshot\_download() with transformers and torch libraries for local summarization.
* **Activity 1.3:** Installed and launched Ollama runtime for local LLM fallback using the Mistral model on localhost:11434.
* **Activity 1.4:** Defined modular architecture for review flow, including separate FastAPI endpoints for single and bulk summarization, model fallback logic, and frontend integration.

**Activity 2: Backend Functionalities Implementation**

* **Activity 2.1:** Developed core FastAPI endpoints /summarize and /upload/summarize\_all for single and bulk review processing.
* **Activity 2.2:** Integrated IBM Granite 7B via Hugging Face in summarizer.py for primary summarization, with Ollama Mistral as fallback using exception handling.
* **Activity 2.3:** Implemented ibm\_sentiment.py to perform sentiment analysis using IBM Watson NLU and extract sentiment labels.
* **Activity 2.4:** Handled CSV parsing in file\_upload.py and review validation in input\_handler.py to ensure clean inputs.
* **Activity 2.5:** Structured backend output in JSON format for smooth frontend integration.

**Activity 3: Frontend Development and Integration**

* **Activity 3.1:** Designed the HTML layout using Tailwind CSS with toggle modes for text input and CSV upload.
* **Activity 3.2:** Developed review output cards that display original review, summary, sentiment, rating, and date.
* **Activity 3.3:** Integrated Chart.js to visualize sentiment trends over time, grouped by review date.
* **Activity 3.4:** Implemented download feature using JavaScript Blob API to allow export of summarized review reports as TXT files.

**Activity 4: Testing and Validation**

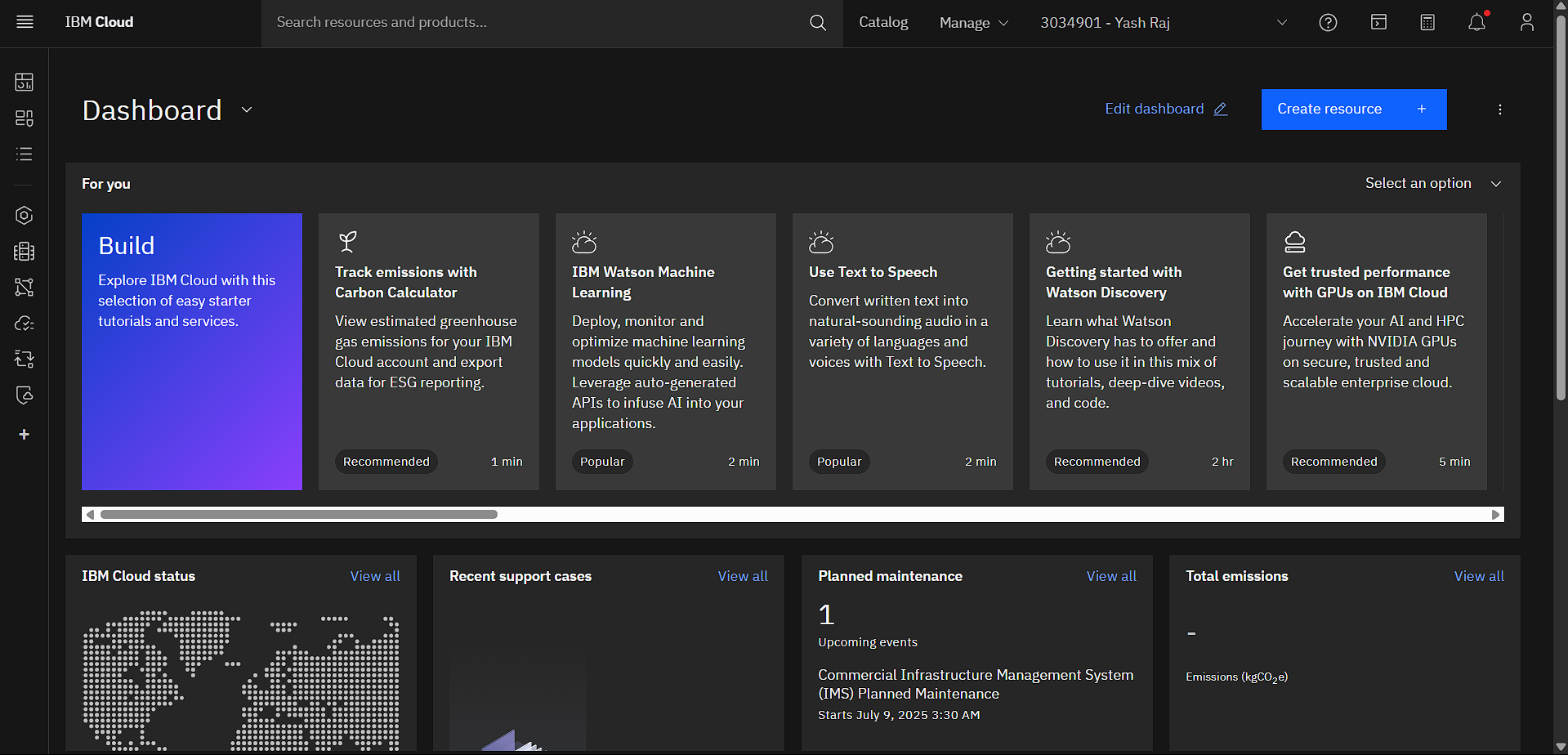
* **Activity 4.1:** Verified IBM Granite model’s local functionality via Hugging Face integration and prompt-based testing.
* **Activity 4.2:** Configured summarizer.py to collaboratively use IBM Granite and Ollama Mistral for generating richer, more accurate summaries. Verified functionality by testing both models and validating combined output quality.
* **Activity 4.3:** Conducted end-to-end UI testing for CSV uploads, review rendering, and trend visualization accuracy.
* **Activity 4.4:** Validated sentiment output accuracy by comparing Watson NLU labels with review tone manually.
* **Activity 4.5:** Monitored FastAPI logs and console output to confirm smooth backend–frontend communication and error tracking.

**Milestones:**

**Milestone 1: Environment Setup and Core Model Integration**

Configured IBM Watson NLU for sentiment analysis and set up API credentials. Integrated IBM Granite 7B locally via Hugging Face for review summarization. Installed and launched Ollama with Mistral as a fallback model and structured the backend architecture using FastAPI.

**Activity 1.1:** Installed and configured IBM Watson Natural Language Understanding (NLU) service and generated API keys from IBM Cloud.

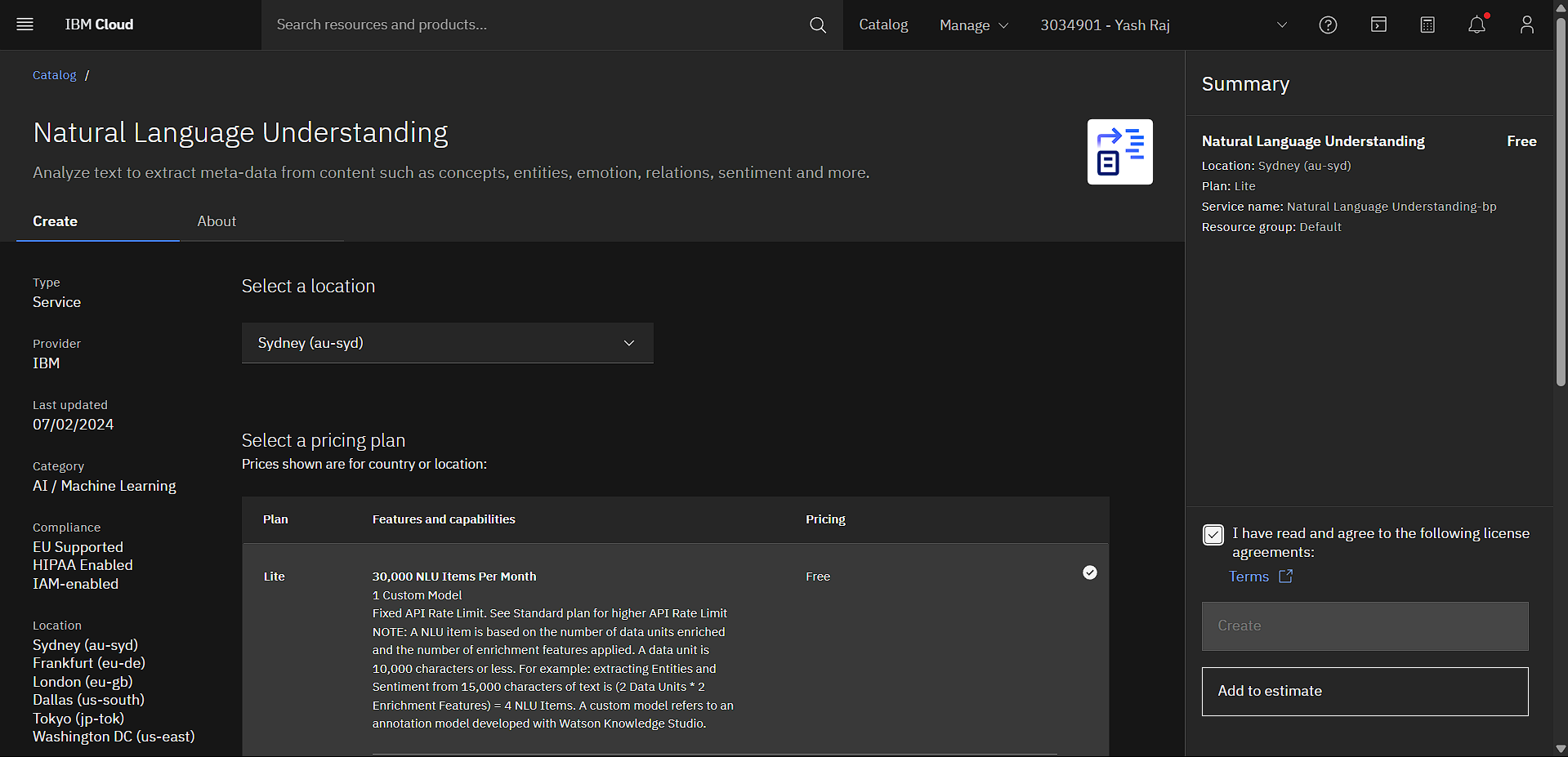


**Create an IBM Cloud Account**

* Visit <https://cloud.ibm.com> and sign up or log in.
* Use your email to create a free IBM Cloud Lite account (which includes Watson NLU with limited usage).

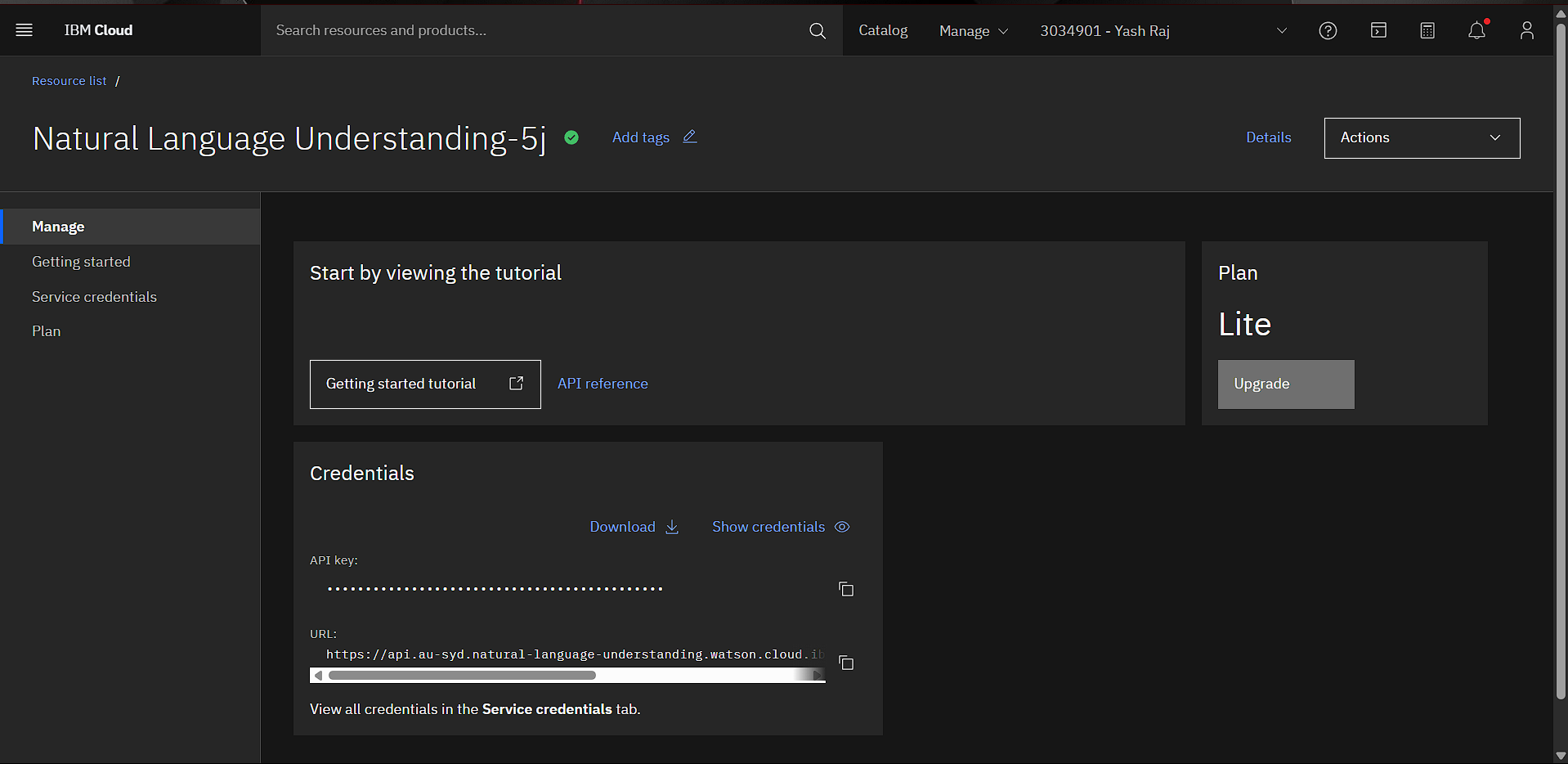
**Access the IBM Cloud Dashboard**

* After logging in, you’ll be redirected to the **Dashboard** (as shown in your screenshot).
* This is the central control panel for all your IBM services and resource provisioning.



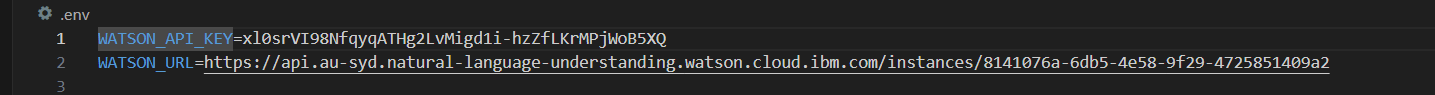
**Create a Watson NLU Resource**

* Click on **“Create resource”** in the top-right corner.
* Go to **AI → Natural Language Understanding** from the service catalog.
* Choose the **Lite plan** (free tier) and name your instance (e.g., Watson-NLU-ReviewApp).
* Click **“Create”** to provision the service.



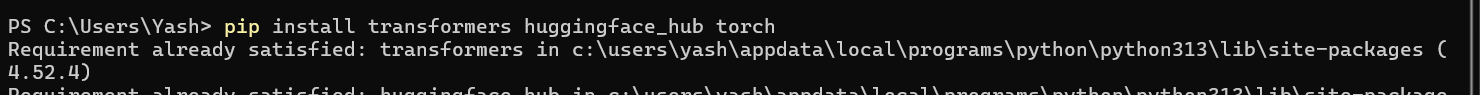
**Generate API Credentials**

* Go to the **“Service Credentials”** tab.
* Click **“New Credential”** to generate your API key and URL.
* Copy the apikey and url values — you will use these in your .env file:



This .env file stores the IBM Watson NLU API key and endpoint URL required for secure authentication. These environment variables are accessed in the backend to connect with the NLU service during sentiment analysis.

**Activity 1.2:** Downloaded and integrated IBM Granite 7B model using Hugging Face snapshot\_download() with transformers and torch libraries for local summarization.

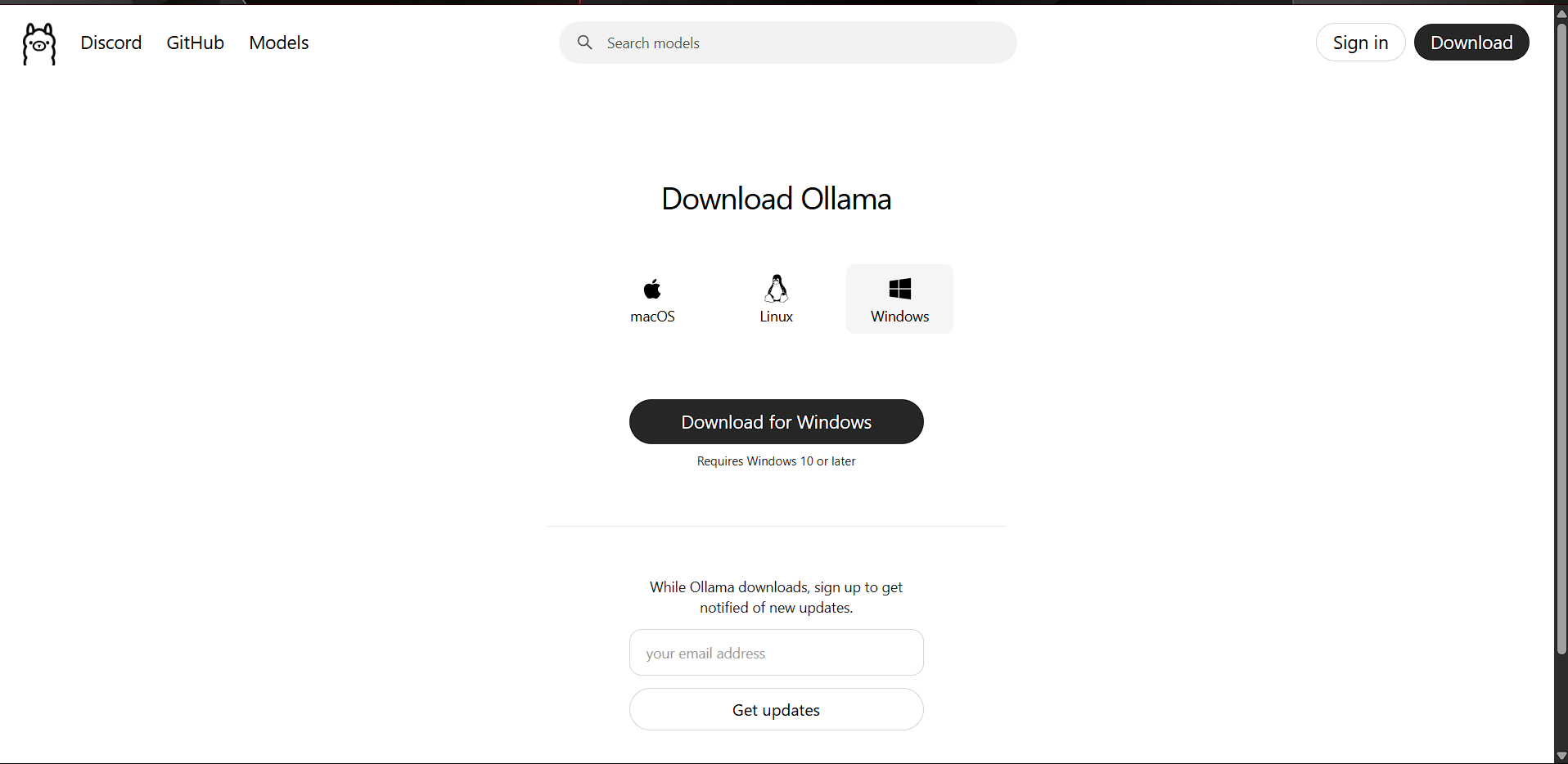


To integrate IBM Granite 7B locally, the model was accessed via Hugging Face using the transformers and huggingface\_hub libraries. This allowed the system to run high-quality summarization prompts without relying on external APIs.

**Steps followed:**

* Installed dependencies using pip install transformers huggingface\_hub torch.
* Defined model ID as ibm-granite/granite-7b-base and set a local directory for caching.
* Used snapshot\_download() to fetch the complete model and configuration files.
* Loaded the model using AutoModelForCausalLM.from\_pretrained() and initialized the tokenizer.
* Passed review prompts through the model and generated outputs using generate().

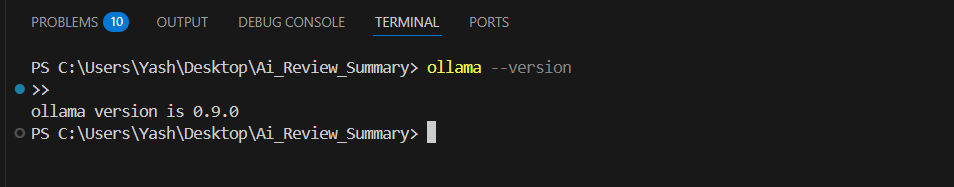
**Activity 1.3:** Installed and launched Ollama runtime for local LLM fallback using the Mistral model on localhost:11434.



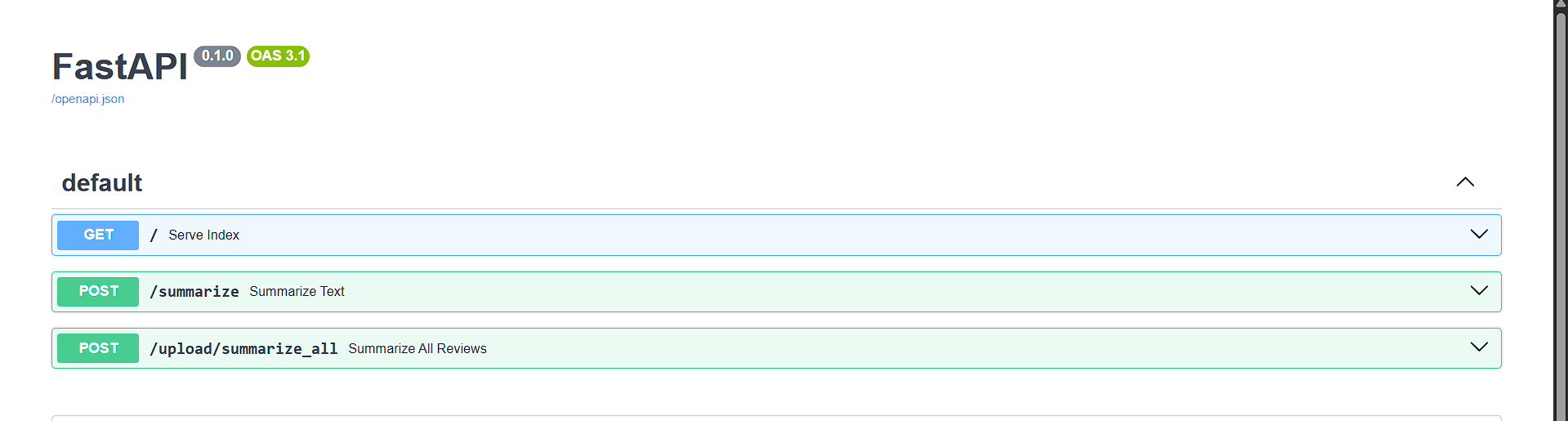
Ollama was configured to run the Mistral language model locally, enabling responsive and lightweight review summarization. This integration complemented IBM Granite by ensuring flexible model execution within the same backend.

**Steps followed:**

* + Installed Ollama from [ollama.com](https://ollama.com) and completed system setup.
  + Pulled the Mistral model using the terminal command ollama pull mistral.
  + Launched the model using ollama run mistral to enable local interaction.
  + Connected to Ollama’s REST API at http://localhost:11434 for sending summarization prompts.
  + Integrated this API into summarizer.py to work alongside IBM Granite.



**Activity 1.4:** Defined modular architecture for review flow, including separate FastAPI endpoints for single and bulk summarization, model fallback logic, and frontend integration.

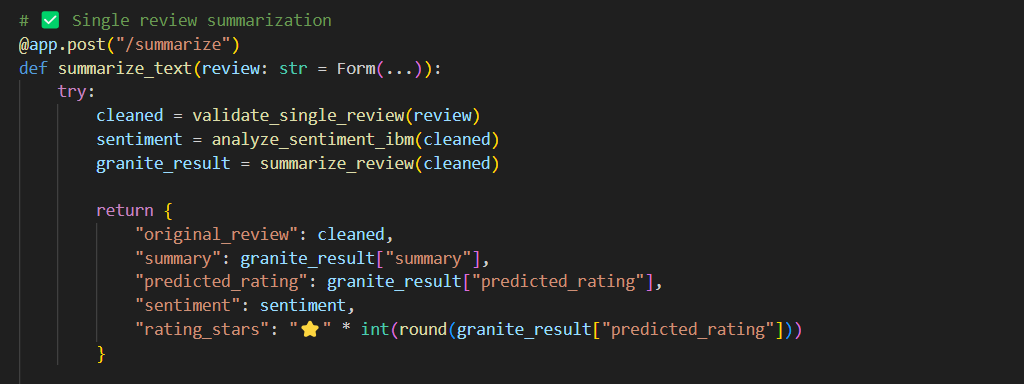


* This interface confirms the successful setup of the FastAPI project structure with dedicated endpoints. It includes /summarize for single-review input and /upload/summarize\_all for bulk CSV summarization, supporting the modular backend design defined in the architecture.

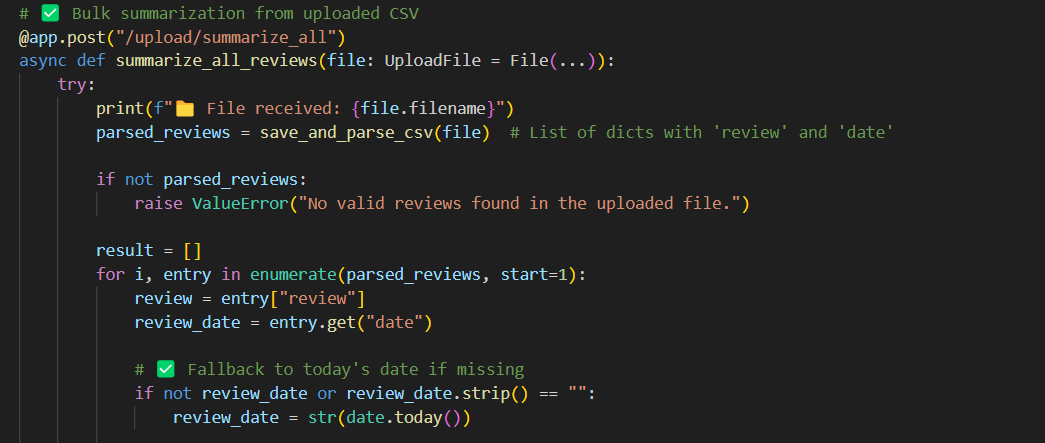
**Milestone 2: Backend Functionalities Implementation**

In this milestone, core API routes were developed to handle both single and bulk review processing using FastAPI. The backend integrated hybrid summarization logic, IBM Watson NLU for sentiment analysis, and structured JSON formatting to support seamless frontend rendering.

**Activity 2.1:** Developed core FastAPI endpoints: /summarize for single review and /upload/summarize\_all for CSV input.

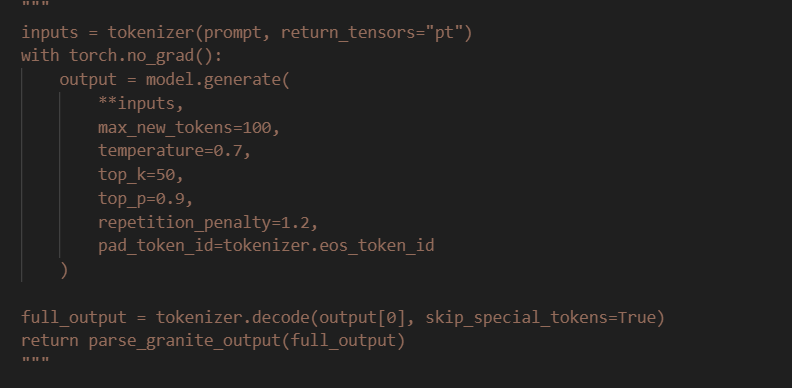


* This function defines the /summarize endpoint in FastAPI for processing single review input. It validates the text, performs sentiment analysis using IBM Watson NLU, generates a summary via the hybrid model, and returns a structured JSON response.

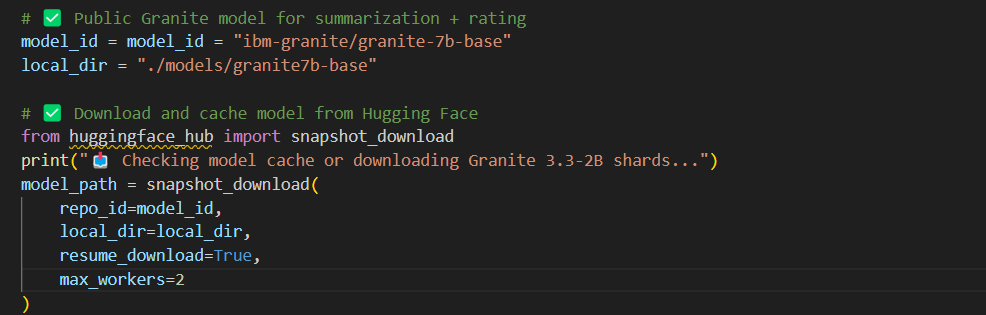


* This function handles the /upload/summarize\_all endpoint for bulk CSV review processing. It parses uploaded files, extracts reviews with timestamps, and assigns fallback dates when missing — enabling structured summarization at scale.

**Activity 2.2:** Built summarizer.py to coordinate Granite and Ollama fallback using exception handling.



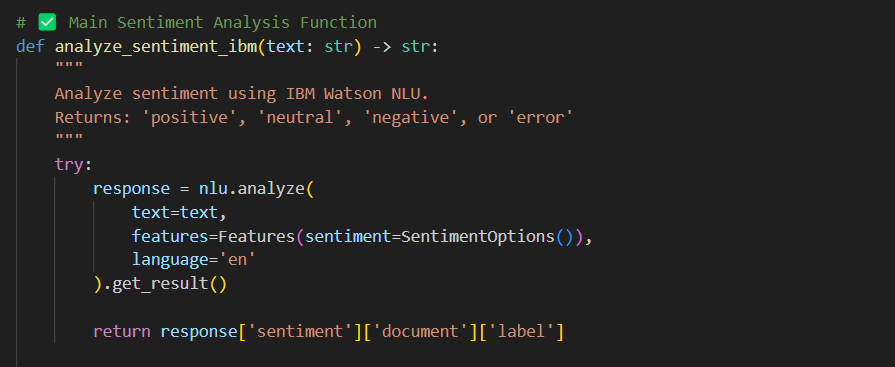
* This section shows how the IBM Granite model generates review summaries using tokenizer() and model.generate() with custom decoding parameters to control the response structure.



* Demonstrates the use of snapshot\_download() from Hugging Face to locally fetch and cache the IBM Granite 7B model, ensuring efficient, offline-ready summarization.

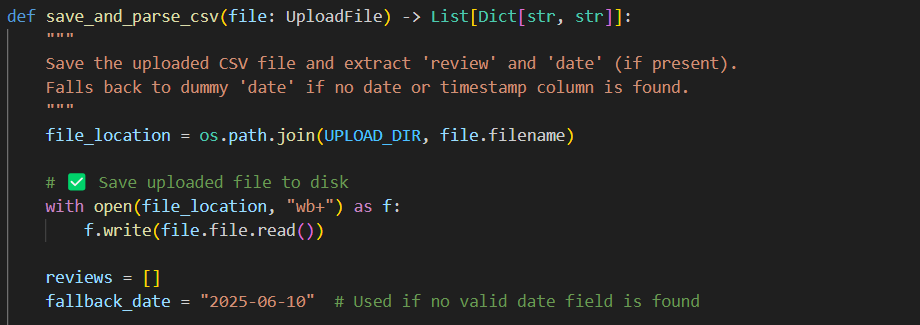


* Presents the complete summarization pipeline — from crafting the prompt and running the model to decoding and cleaning the output — validating Granite’s end-to-end integration.
* **Activity 2.3:** Created ibm\_sentiment.py to process sentiment analysis via Watson NLU and return positive/neutral/negative labels.

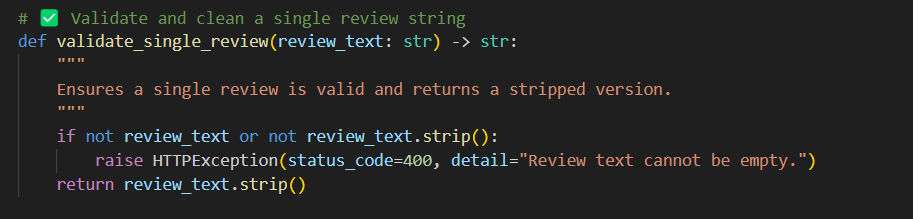


* This function connects to IBM Watson NLU to analyze the sentiment of a given review text. It returns a document-level label such as 'positive', 'neutral', or 'negative', supporting automated tone classification in the backend.

**Activity 2.4:** Parsed CSV uploads using file\_upload.py and added review validation logic in input\_handler.py.

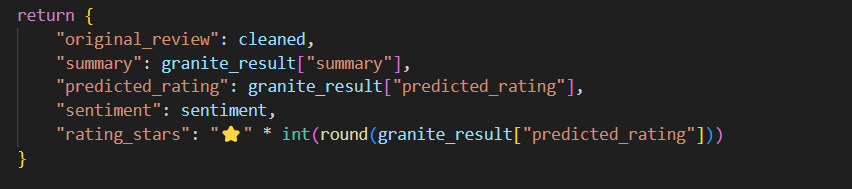


* This function saves the uploaded CSV file and extracts review and date fields, applying a fallback date when none is provided. It ensures reviews are parsed into a consistent structure for bulk processing.

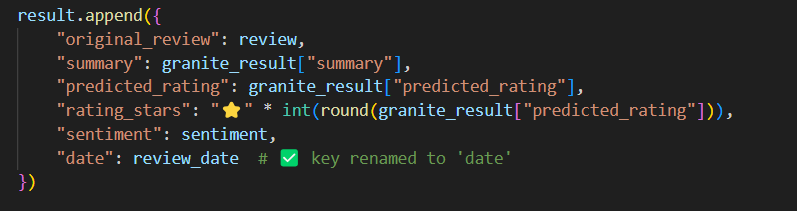


* This function checks whether the input review text is non-empty and returns a cleaned version. It enforces validation for single-review submissions to maintain input quality and prevent errors.

**Activity 2.5:** Designed output in structured JSON format for easy frontend consumption.



* This return statement structures the output for a single review into JSON, including the summary, predicted rating, sentiment, and star icons. It ensures uniform response formatting for frontend consumption.

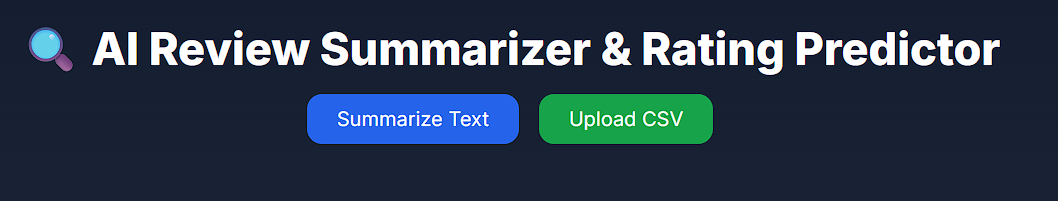


* Each parsed review from the CSV is appended as a structured JSON object with fields like summary, sentiment, rating, and review date. This consistent format supports rendering multiple reviews dynamically on the frontend.

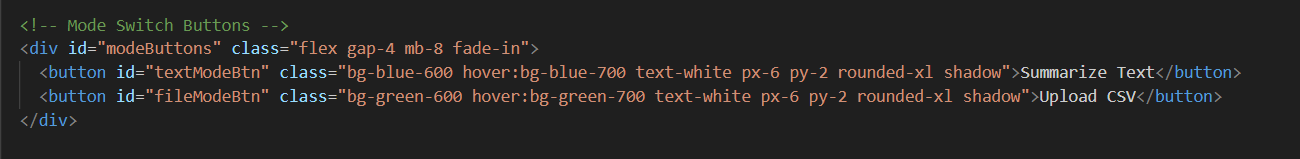
**Milestone 3: Frontend Development and Integration**

This milestone focused on designing a responsive and interactive user interface using HTML, Tailwind CSS, and JavaScript. Key components like mode toggles, review cards, and sentiment charts were implemented for smooth user interaction. The frontend seamlessly displays summaries, ratings, and sentiment trends from backend responses.

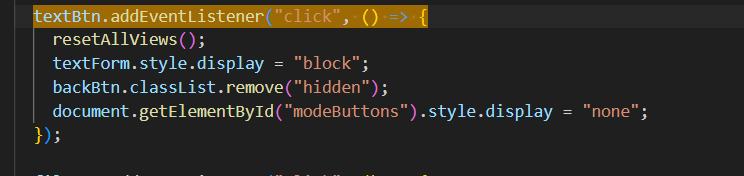
**Activity 3.1:** Designed the HTML layout using Tailwind CSS with toggle modes for text input and CSV upload.



* The interface displays a user-friendly header with toggle buttons for "Summarize Text" and "Upload CSV". These options guide users to choose between manual input and file-based review processing.

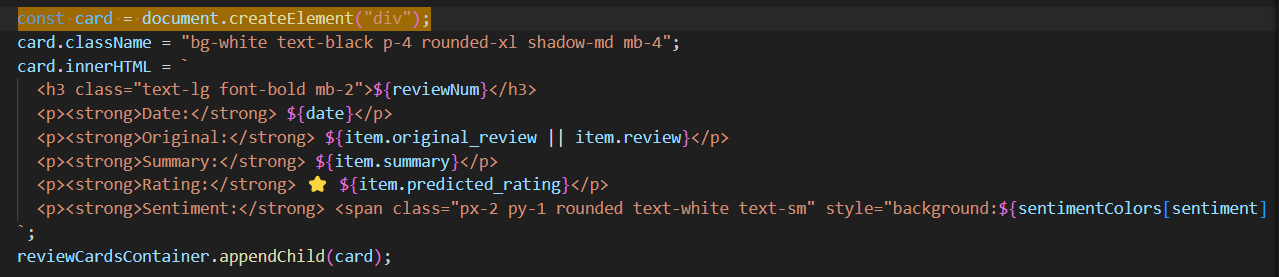


* This HTML snippet defines the mode switch buttons styled using Tailwind CSS. Each button is linked to specific frontend behavior to activate either the text input form or the file upload form.



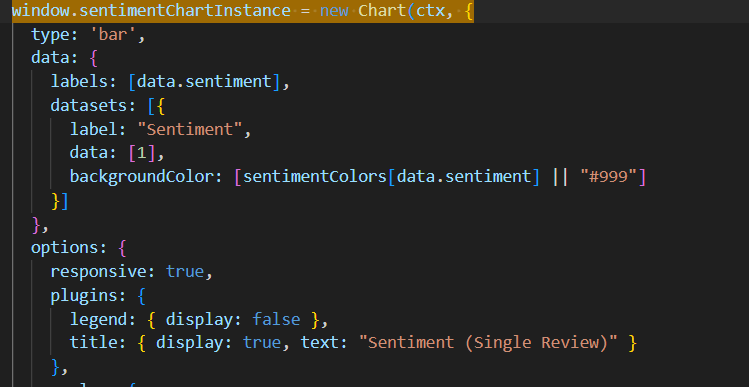
* This JavaScript code handles the toggle event for the text input mode. It dynamically displays the correct form while hiding the mode selector to create a smooth single-page interaction.

**Activity 3.2:** Developed review output cards that display original review, summary, sentiment, rating, and date.



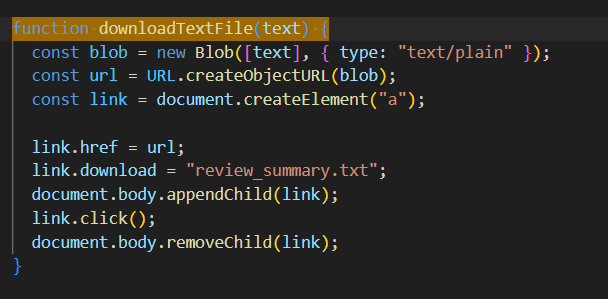
* This JavaScript code dynamically creates review cards displaying key fields like original text, summary, sentiment, and predicted rating. Each card is styled using Tailwind CSS and added to the interface to visually represent analyzed reviews in real time.

**Activity 3.3:** Integrated Chart.js to visualize sentiment trends over time, grouped by review date.



* This Chart.js configuration generates a bar chart to visualize the sentiment of a single review. It dynamically assigns colors based on sentiment type and includes a custom title for clarity, enhancing real-time feedback representation.

**Activity 3.4:** Implemented download feature using JavaScript Blob API to allow export of summarized review reports as TXT files.

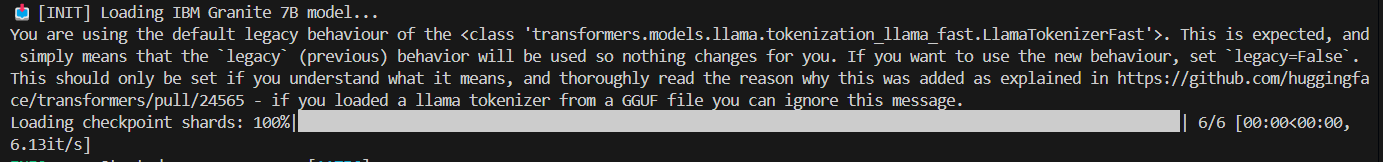


This configuration initializes a Chart.js bar graph to display the sentiment of a single review using a color-coded bar. It dynamically sets the label and color based on sentiment type (positive, neutral, or negative). The chart is responsive and clearly titled for immediate visual interpretation.

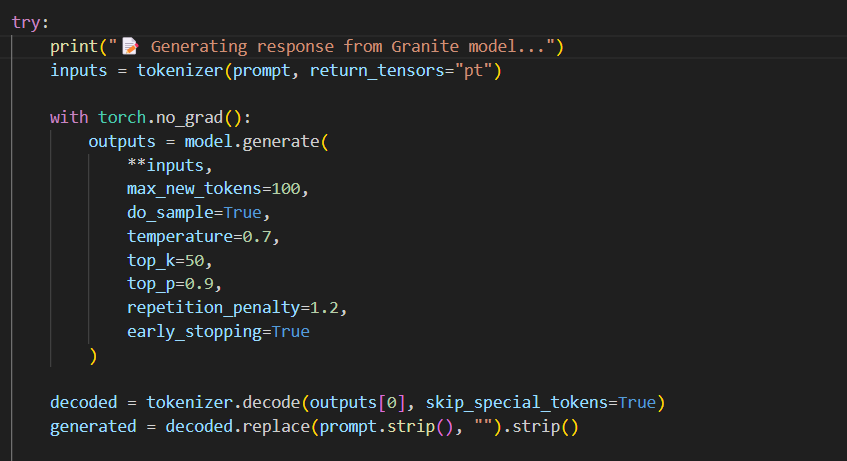
**Milestone 4: Testing and Validation**

This milestone involved thorough testing of model outputs, API performance, and UI responsiveness. Both IBM Granite and Ollama were validated independently to ensure accurate summarization. Sentiment labels were manually cross-checked, and end-to-end flows were tested using real reviews and CSV uploads.

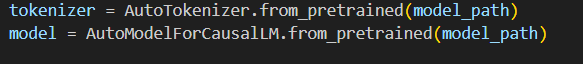
**Activity 4.1:** Verified IBM Granite model’s local functionality via Hugging Face integration and prompt-based testing.



* This output shows the IBM Granite 7B model loading locally with all shards fetched successfully via Hugging Face. It confirms the model was correctly cached and initialized for runtime summarization.

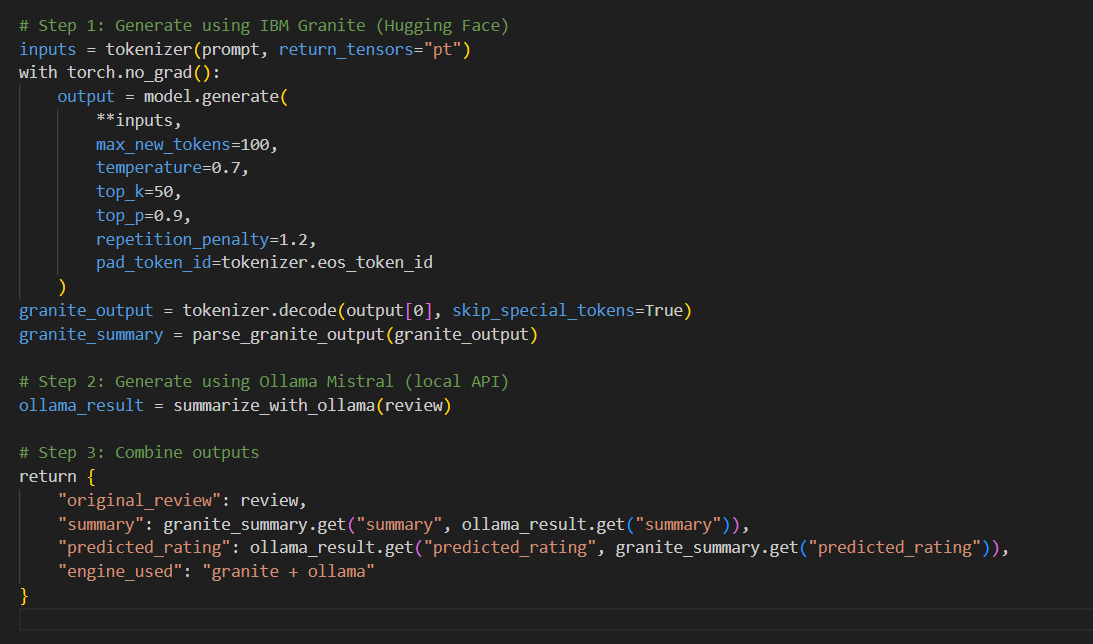


* This block executes a review summarization prompt through the Granite model using generate(). It demonstrates real-time inference with temperature and sampling controls, validating prompt-based output generation.



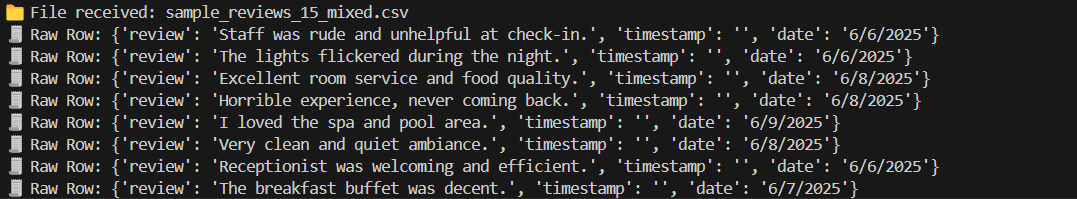
* This snippet confirms successful loading of the IBM Granite tokenizer and model from the local path. It's a crucial setup step that enables end-to-end summarization testing on local infrastructure.

**Activity 4.2:** Configured summarizer.py to strategically combine IBM Granite and Ollama Mistral models for improved summary quality. The system utilizes both models collaboratively to ensure diverse and accurate outputs. Functionality was verified by testing each model independently and observing merged results.



* This function demonstrates the collaborative use of IBM Granite and Ollama Mistral for generating review summaries. It runs both models independently and merges their outputs to produce a more accurate and diverse result, reflecting a hybrid AI approach.

**Activity 4.3:** Conducted end-to-end UI testing for CSV uploads, review rendering, and trend visualization accuracy.

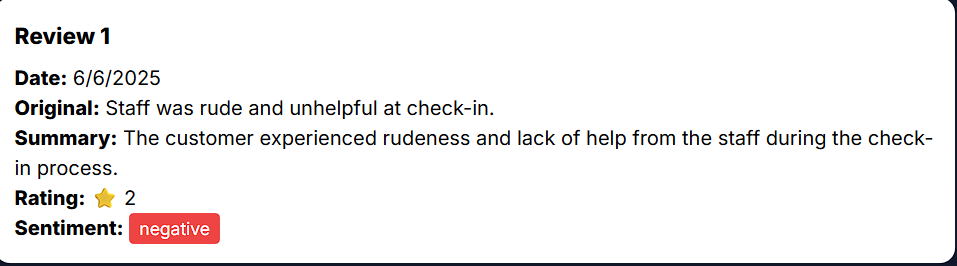


* This output confirms the successful receipt and parsing of a bulk CSV file containing multiple reviews with associated dates. Each row is extracted, validated, and logged in the terminal for real-time visibility. It demonstrates effective backend handling during end-to-end UI testing of file uploads.

**Activity 4.4:** Validated sentiment output accuracy by comparing Watson NLU labels with review tone manually.

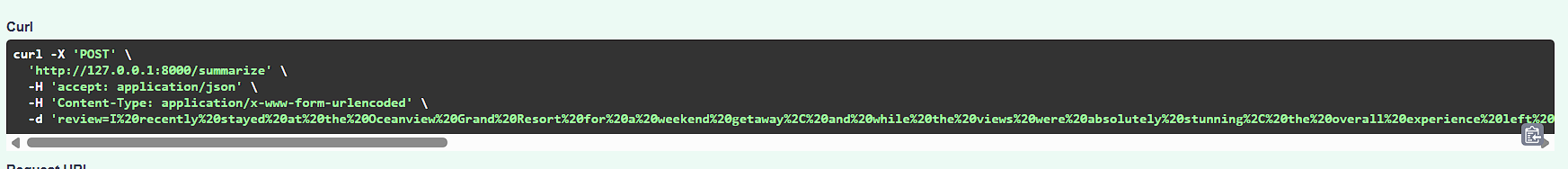


* This console log shows the model-generated summary for a review describing poor staff behavior at check-in. The summary accurately reflects the negative sentiment intended by the original review content.

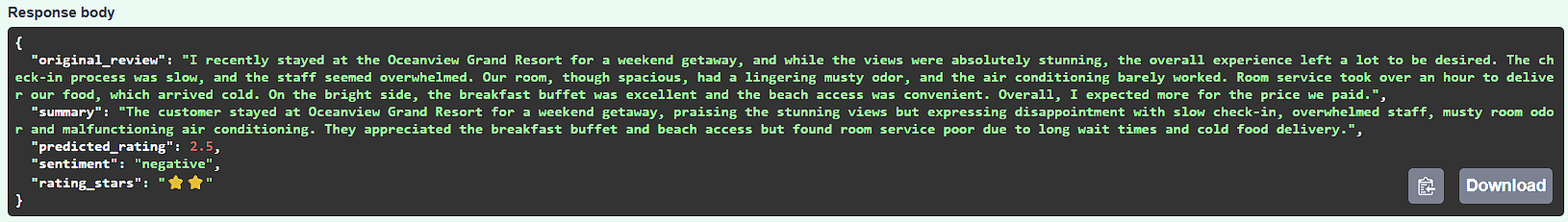


* The frontend output card displays the original review, the generated summary, and a sentiment label of “negative.” This validates that the Watson NLU sentiment output aligns correctly with the actual tone of the review.

**Activity 4.5:** Monitored FastAPI logs and console output to confirm smooth backend–frontend communication and error tracking.



* This shows a structured cURL command used to test the /summarize endpoint with review input. It verifies backend accessibility and simulates real client-side API requests for debugging and testing.

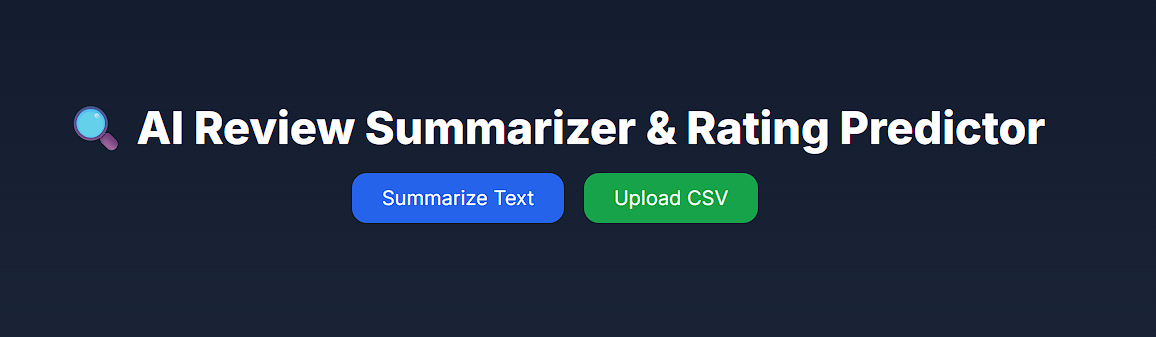


* The response body confirms successful summarization, displaying original review, generated summary, predicted rating, and sentiment. This proves the system returns clean, structured data as expected for frontend use.



* This error output demonstrates FastAPI’s built-in validation system, which triggers when input formatting is incorrect. It helps identify malformed requests and maintain robust API behavior.

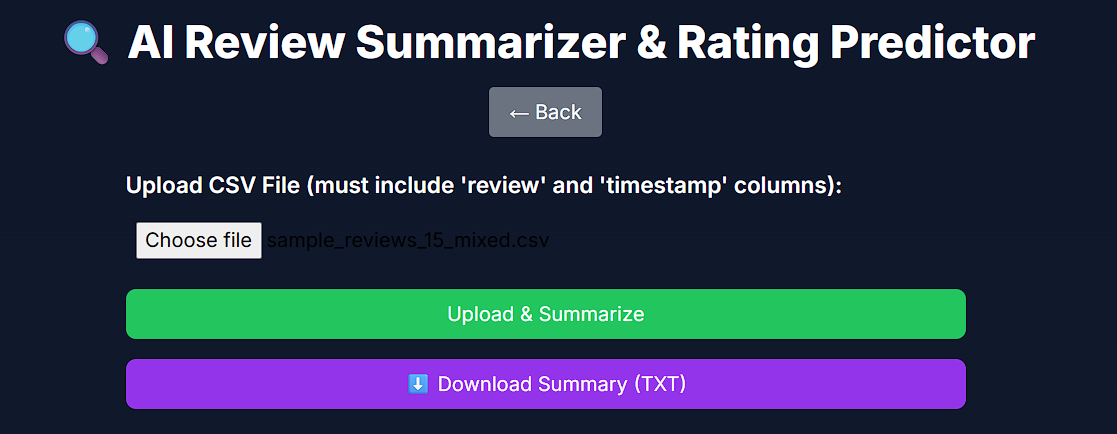
**Project Output –**



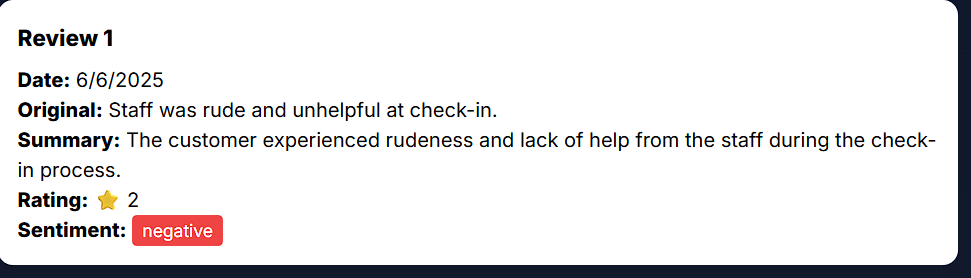
* This is the landing interface of the AI Review Summarizer & Rating Predictor web app. It provides users with two clear input options: one for entering a single review and the other for uploading a CSV file. The layout is clean and user-friendly, guiding the interaction flow efficiently.



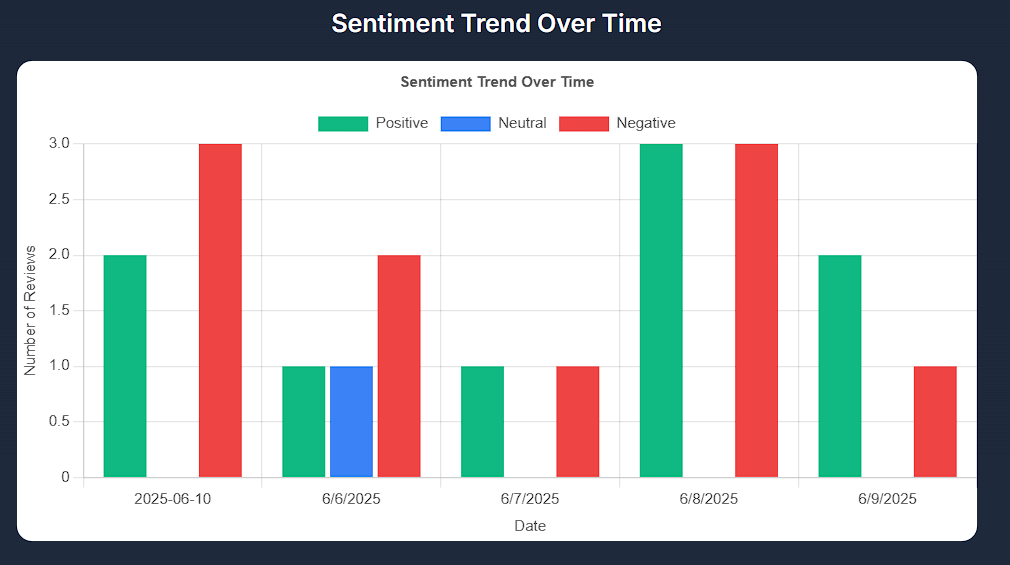
* This output panel displays the results of analyzing a single customer review. The system extracts the original text, generates a concise summary, predicts a star rating (2.5/5), and classifies the sentiment as negative. The summary highlights specific pain points such as slow service and room issues, balanced with minor positives. It demonstrates the effectiveness of the hybrid AI in capturing detailed review insights.



* This interface enables users to upload a CSV file containing multiple reviews and their timestamps. Upon submission, the system processes all entries and displays summarized results in bulk. A download button allows users to export the generated summaries as a .txt report.



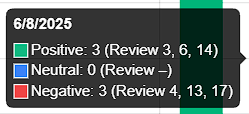
**(Review Card Output):**  
This output card displays a processed review from the CSV upload, showing the original text, a generated summary, a star rating of 2, and a red-tagged negative sentiment. It offers a clear, visual representation of the analysis.



**(Sentiment Trend Chart):**  
The sentiment trend chart visualizes review sentiment distribution over time. It uses colored bars to represent positive (green), neutral (blue), and negative (red) feedback per date. This helps identify shifts in customer satisfaction across days.

**Real-World Insights from Sentiment Trend Chart:**

* Identifies dates with spikes in negative feedback
* Helps evaluate impact of service changes over time
* Supports quick decision-making for issue resolution
* Aids in forecasting customer satisfaction trends
* Useful for visual reporting to stakeholders



**(Tooltip Insight on Hover):**  
This hover tooltip on the chart shows exact sentiment counts and associated review numbers for a specific date. It enhances interactivity by giving deeper insights into which reviews contributed to sentiment peaks.

**Conclusion:**

The **AI-Based Review Summarization and Rating Prediction System** stands as a well-structured, intelligent solution designed to process, evaluate, and present customer reviews at scale. Through a thoughtful integration of cutting-edge technologies like IBM Granite (via Hugging Face), IBM Watson NLU, and Ollama's Mistral model, the project successfully delivers accurate summaries, sentiment classification, and rating predictions. The system is tailored for both individual and bulk review processing, offering businesses a fast and visual way to understand customer perception.

With its full-stack architecture — combining a FastAPI backend and a Tailwind CSS-powered frontend — the platform ensures seamless data flow, modular integration, and a responsive user experience. Whether a user inputs a single review or uploads a CSV file with hundreds of entries, the platform processes each efficiently, summarizes the content using hybrid models, and renders the results via review cards and sentiment trend charts.

**What Was Achieved**

* Developed a dual-model summarization strategy combining **IBM Granite** and **Ollama Mistral**, blending the formality and precision of enterprise AI with the responsiveness of a lightweight local model.
* Successfully implemented **IBM Watson NLU** for real-time sentiment classification and ensured that outputs matched the natural tone of customer reviews.
* Designed interactive UI components that allow for easy toggling between input modes, result visualization, and .txt export functionality.
* Enabled end-to-end handling of both single and batch review workflows with structured JSON responses and frontend display.

**Challenges Faced**

* **Model Weight Handling**: The IBM Granite model required substantial local resources and time to load, especially during first-time setup via Hugging Face.
* **Response Variability**: Differences in formatting between Granite and Ollama outputs required robust parsing logic and fallback handling.
* **Sentiment Consistency**: Validating Watson NLU’s predictions against subjective review tones occasionally revealed the need for additional text preprocessing.
* **UI-Backend Sync**: Ensuring smooth rendering of dynamic content such as charts and multiple review cards required careful DOM handling and state management.

**Learning Outcomes**

* Gained hands-on experience in integrating **Hugging Face Transformers** and **LLMs** into production-ready systems.
* Learned to build **hybrid AI pipelines** by blending two models for maximum performance and reliability.
* Understood the inner workings of **IBM Watson NLU**, including authentication, API handling, and sentiment interpretation.
* Mastered **FastAPI routing and response structuring**, alongside frontend implementation using **Tailwind CSS** and **Chart.js** for visual analytics.
* Learned to anticipate user-side issues (e.g., empty fields, malformed inputs) and address them through effective validation.

**Future Scope**

The current version lays a solid foundation but can be further extended through several practical enhancements:

* **Domain Classification**: Automatically tagging reviews as belonging to hospitality, retail, tech, etc., using topic modeling or zero-shot classification.
* **Advanced Export Options**: Supporting downloadable reports in **PDF** and **Excel** formats to aid business reporting.
* **User Accounts**: Enabling login functionality with user-specific dashboards for history tracking and segmented insights.
* **Multilingual Support**: Adding the ability to process reviews written in other languages by integrating translation APIs or multilingual models.
* **Admin Panel**: A back-office interface to manage datasets, track usage logs, and monitor overall system health.

This project not only meets the technical expectations set forth in the original requirements but also showcases the practical application of AI in business feedback analysis. It serves as a scalable blueprint for future AI-integrated SaaS tools in customer experience intelligence.