Infosys Springboard Virtual Internship 6.0 Completion Report

# Team Details

Batch Number: 2

Start date: 12th Aug 25

Name: Lakshita Setia

Internship Duration: 8 Weeks

# 1. Project Title

**TextMorph – Advanced Text Summarization and Paraphrasing**

# 2. Project Objective

The objective of this internship project was to develop an AI-driven application that can automatically **summarize** and **paraphrase** long textual content efficiently. The project aimed to combine modern Natural Language Processing (NLP) models with an interactive Streamlit interface to help users condense and rephrase large volumes of text without losing context or meaning.  
The system was designed to help individuals, students, and organizations save time while improving clarity, especially when processing articles, reports, and academic material.

# 3. Project description in detail

The TextMorph application integrates advanced NLP models and a user-friendly web interface to provide two major functionalities — *Text Summarization* and *Paraphrasing*. The goal was to automate lengthy reading and writing tasks by leveraging deep learning and transformer-based text generation.

**1. System Overview**

The project followed a modular approach consisting of:

* **Summarization Module** – condenses long text into concise summaries using a transformer-based summarization pipeline.
* **Paraphrasing Module** – rewrites input text in a new form while preserving original meaning (under implementation).
* **Streamlit Front-End** – provides an easy interface for users to input, view, and download results.
* **Environment Configuration** – environment variables stored securely in .env for API **usage.**

**2. Functional Workflow**

1. **Input Processing:**  
   Users can paste text document.
2. **Summarization:**  
   Implemented using the Hugging Face transformer pipeline with models such as facebook/bart-large-cnn and t5-base.  
   The text is tokenized, truncated to the maximum token length, and passed through the summarizer model.
3. **Paraphrasing (Future Module):**  
   The paraphrasing function will utilize a pre-trained model like tuner007/pegasus\_paraphrase to produce alternate rewordings of the same text.
4. **Output Display:**  
   The generated summary or paraphrase is displayed on the Streamlit dashboard. Users can copy or download the results as a text file.
5. **Environment Setup:**  
   The .env file contains environment variables for Hugging Face tokens and API keys, ensuring secure configuration without exposing credentials.

**3. Technology Stack**

* **Programming Language**: Python
* **Framework**: Streamlit
* **Libraries Used**: Transformers, Hugging Face, dotenv, PyPDF2, python-docx
* **Environment Management**: venv
* **Version Control**: Git & GitHub
* **Model Base:** BART / T5 / Pegasus (for text summarization and paraphrasing)

# 4. Timeline Overview

|  |  |  |
| --- | --- | --- |
| Week | Activities Planned | Activities Completed |
| Week 1 | Project setup, environment creation, repo initialization | Setup completed, environment activated |
| Week 2 | Implement summarization pipeline | BART summarizer implemented and tested |
| Week 3 | Streamlit integration | Streamlit UI added, tested input-output workflow |
| Week 4 | Paraphrasing model integration | Paraphrasing pipeline research, partial integration |
| Week 5 | Add Git Configuration | Secure config added, GitHub repo cleaned and updated |
| Week 6 | UI enhancements and output export | Added download and copy features |
| Week 7 | Testing and debugging | Functional testing for summarization completed |
| Week 8 | Documentation and report preparation | Final demo, report, and GitHub upload completed |

# 5a. Key Milestones

|  |  |  |
| --- | --- | --- |
| Milestone | Description | Date Achieved |
| Project Kickoff | Internship and project orientation | 01 Sept 2025 |
| Prototype/First Draft | Environment setup and repo initialization | 15 Sept 2025 |
| Mid-Term Review | First working summarization model, Streamlit Interface completed | 03 Oct 2025 |
| Final Submission | Paraphrasing code integrated | 24 Oct 2025 |
| Presentation | Final project presentation and report submission | 27 Oct 2025 |

# 5b. Project execution details

The project was executed through distinct development phases:

**Phase 1 – Environment Setup**

* Created a virtual environment (.venv) and installed dependencies using requirements.txt.
* Implemented .env for Hugging Face API token storage.
* Configured .gitignore to avoid pushing environment files to GitHub.

**Phase 2 – Summarization Pipeline**

* Implemented using the Hugging Face Transformers library.
* Used pre-trained summarization models (facebook/bart-large-cnn and t5-small) for extractive and abstractive summaries.
* Added token length control and error handling for long text.

**Phase 3 – Paraphrasing Pipeline**

* Integrated the Pegasus Paraphrase model (under development).
* Designed functions to generate multiple paraphrase variations for user selection.

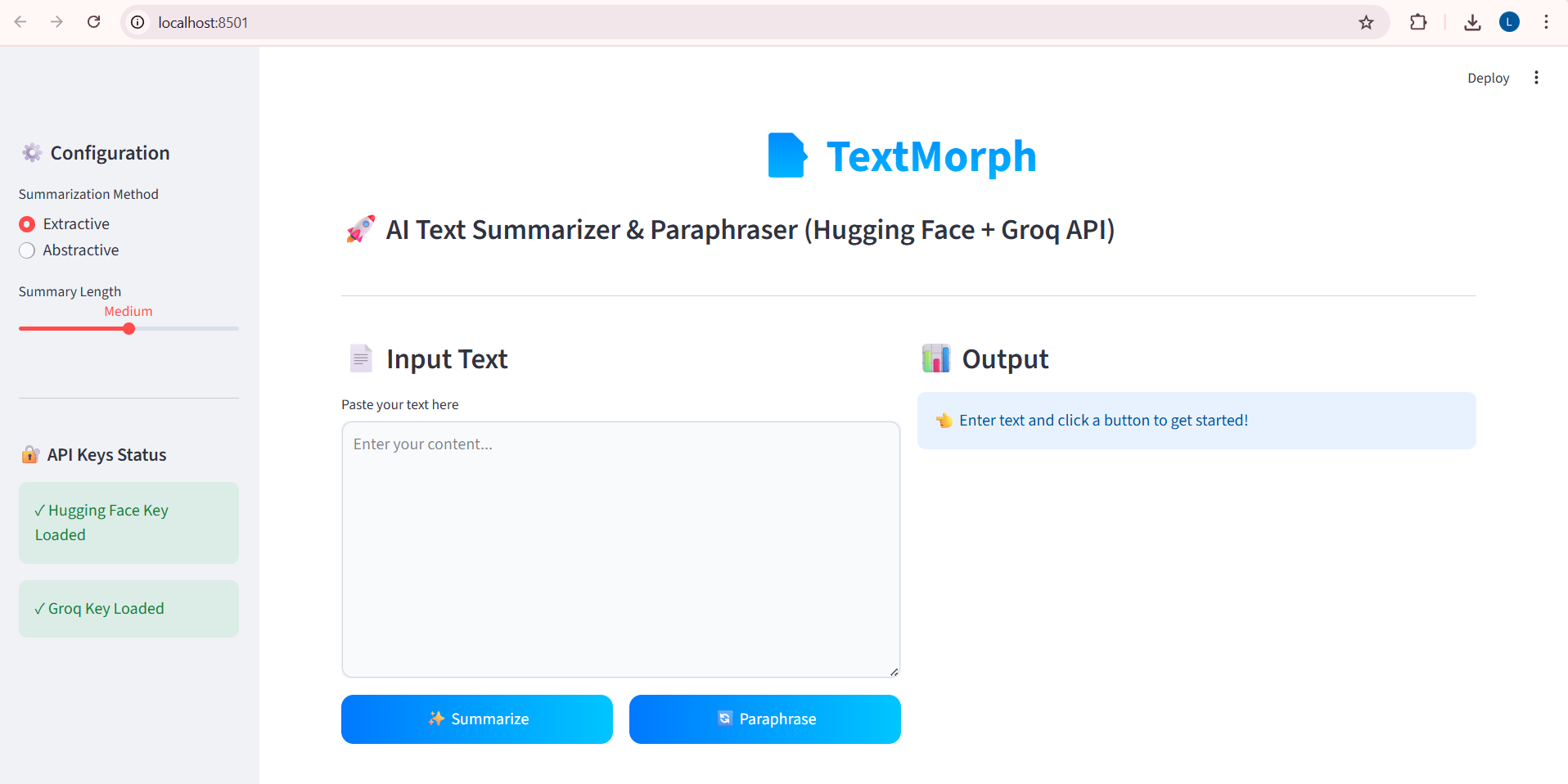
**Phase 4 – Streamlit App Development**

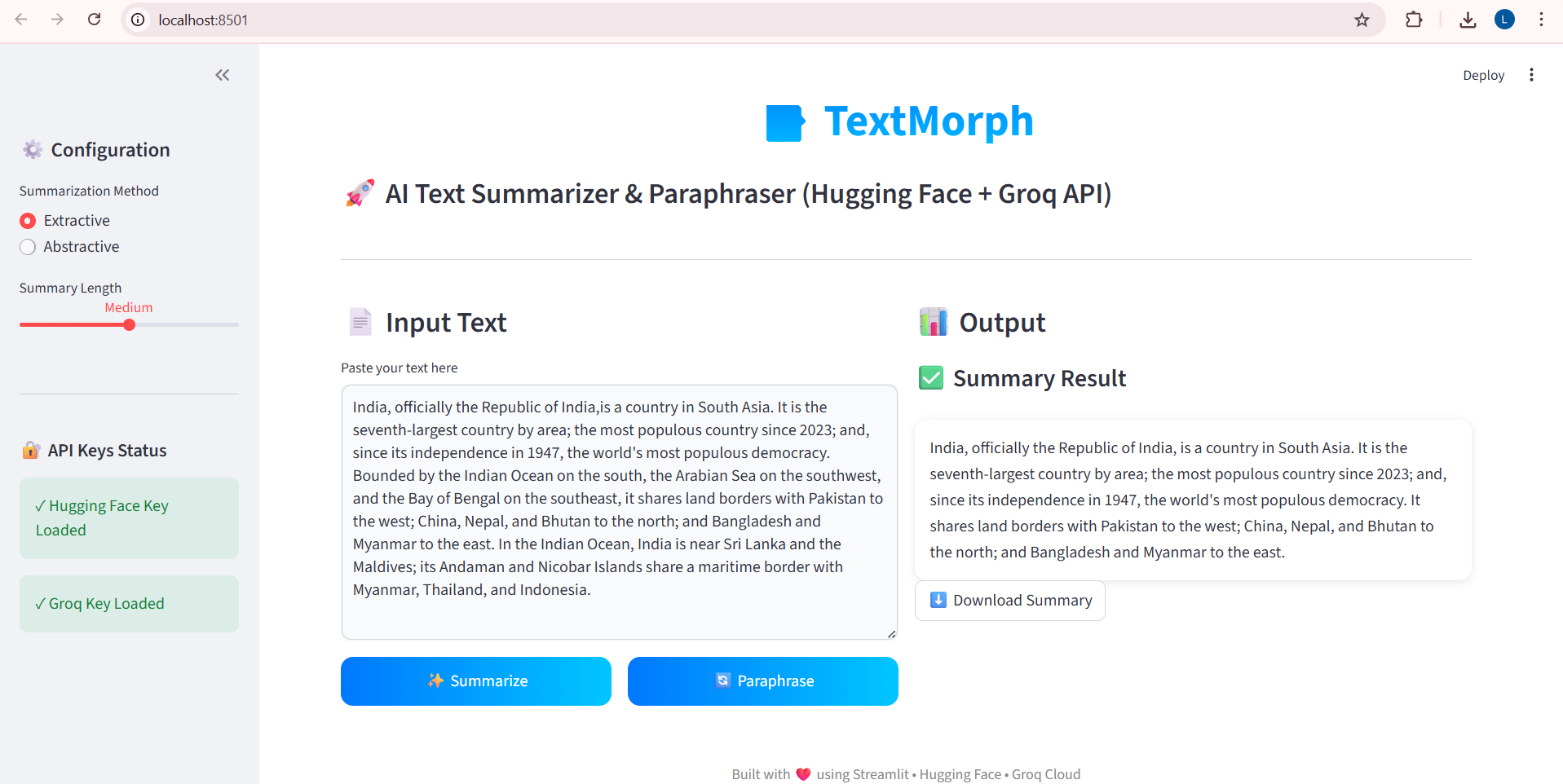
* Built an interactive user interface with file upload, text input, and download features.
* Added summarization and paraphrasing buttons with loading indicators.
* Integrated both functionalities into a clean two-tab layout for user experience.

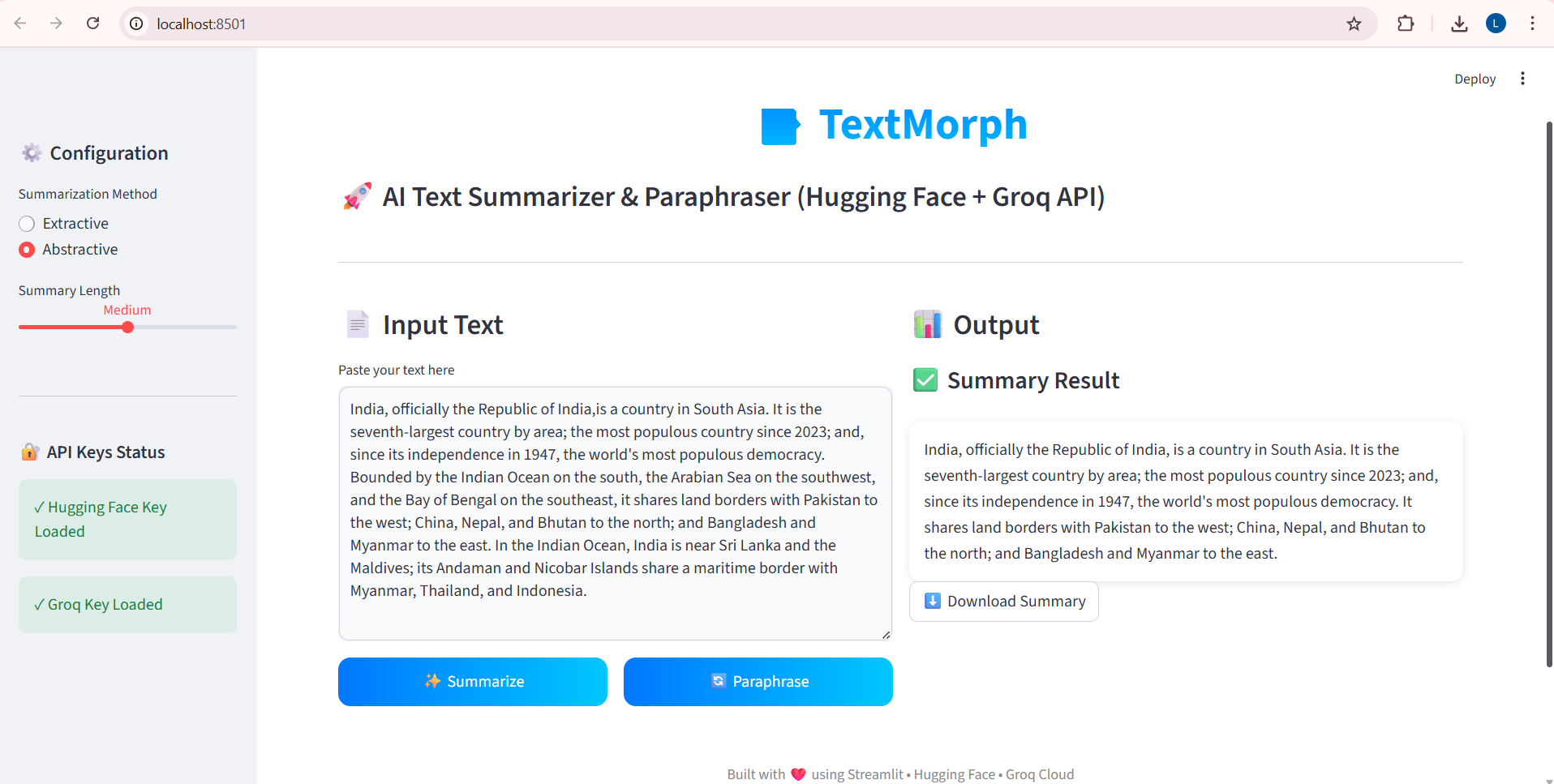
**Phase 5 – Testing and Documentation**

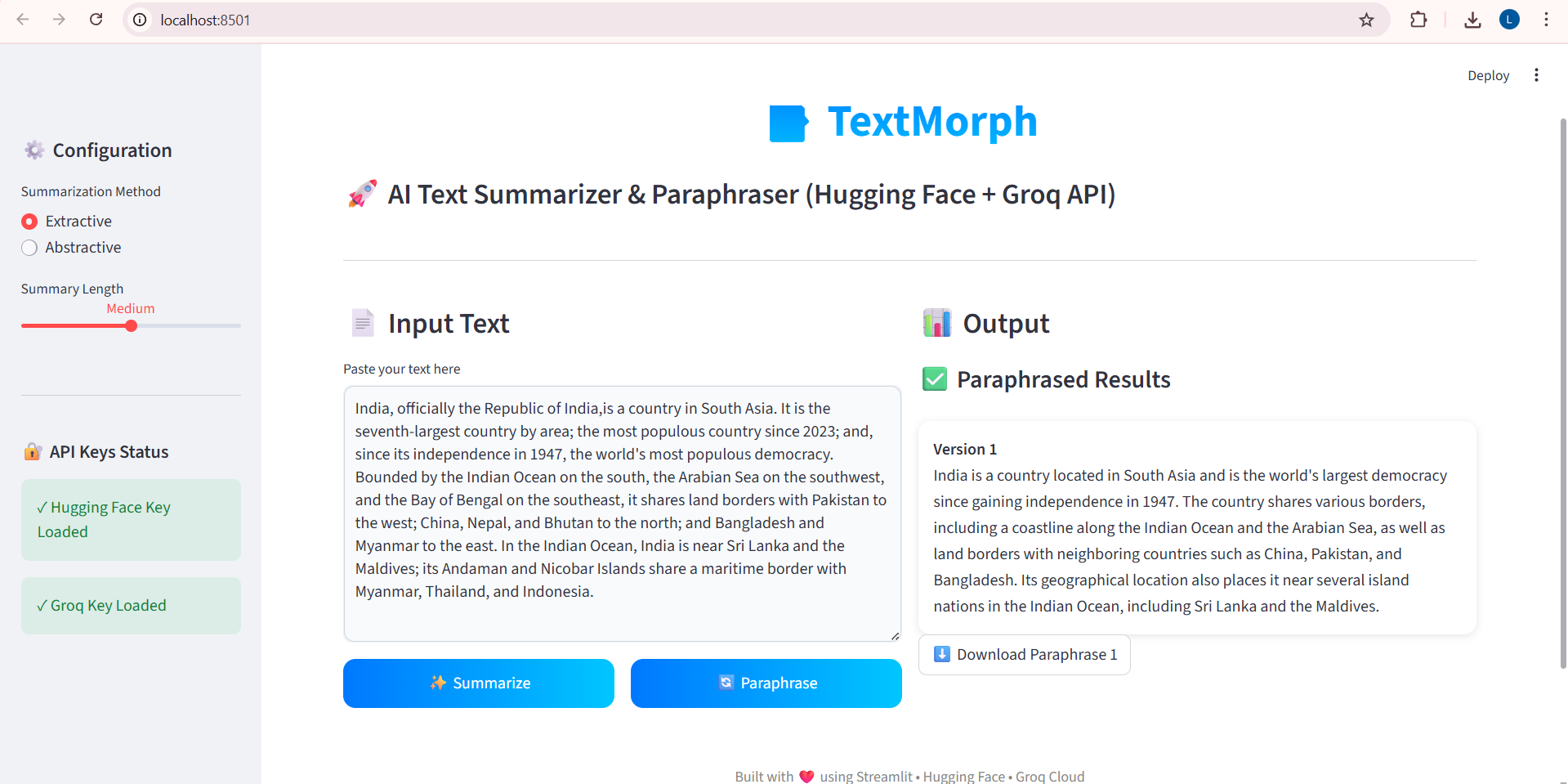
* Tested summarization accuracy with news articles, essays, and research abstracts.
* Verified GitHub commits and cleaned unnecessary files using .gitignore.
* Documented all modules and added README.md and .env.example.

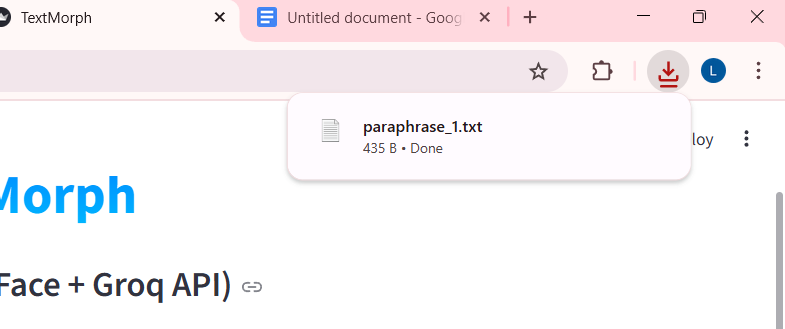
# Snapshots / Screenshots

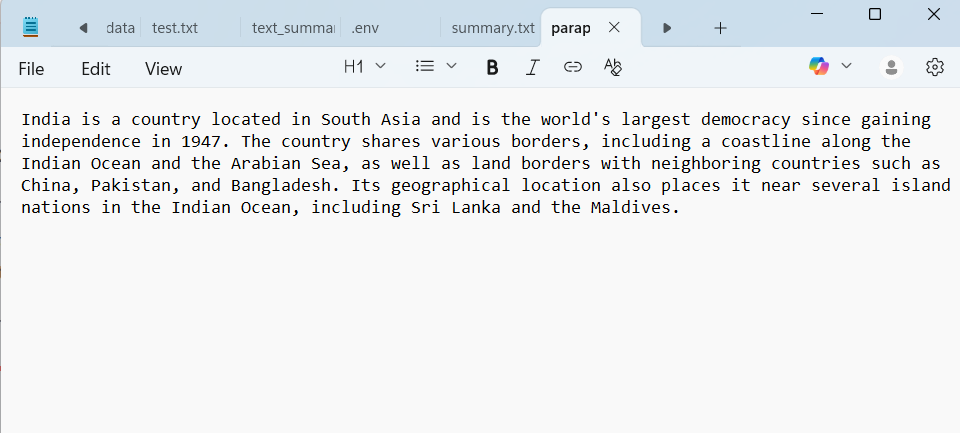












# Challenges Faced

**1. Model Integration and API Limitations**

One of the primary challenges was integrating Hugging Face transformer models with the Streamlit interface. While the pre-trained models provided high-quality summarization and paraphrasing, handling large text inputs and API response delays was difficult. API rate limits and occasional latency issues required efficient caching and optimization to maintain smooth performance. Additionally, balancing between extractive and abstractive summarization models involved experimenting with different model architectures to achieve meaningful yet concise summaries.

**2. Environment Configuration and Dependency Management**

Setting up the environment correctly proved challenging in the initial stages. Managing multiple dependencies such as transformers, torch, and streamlit, each with their version compatibility issues, often led to import errors and dependency conflicts. Creating a virtual environment and maintaining a structured requirements.txt helped overcome these issues, but it required patience and repeated troubleshooting to ensure everything ran smoothly across systems.

**3. Handling Large Text Inputs**

Another major challenge was managing long or unstructured text inputs that exceeded the model’s maximum token limit. The models would either truncate content or fail to process the input efficiently, leading to incomplete summaries. To solve this, I had to research text chunking methods — breaking the text into smaller overlapping segments while preserving context — and ensuring the final summarized output remained coherent and logically connected.

**4. Designing an Interactive and Responsive Frontend**

Creating a visually appealing yet functional frontend using Streamlit came with its own learning curve. Streamlit simplifies UI development but provides limited customization options by default. Implementing advanced UI features, custom CSS styling, and layout alignment required creative experimentation. Ensuring that buttons, text areas, and output sections were intuitive and responsive on different screen sizes was also a continuous process of refinement.

**5. Secure Handling of API Keys and Sensitive Data**

Managing API keys securely while deploying the project was another challenge. Initially, storing keys directly in the code seemed convenient, but it posed a major security risk. I learned how to use .env files and the python-dotenv package to load sensitive credentials safely at runtime. Configuring .gitignore correctly to prevent accidental uploads of secret files to GitHub was essential, but initially confusing, especially when integrating .env.example for safe sharing.

**6. Version Control and Synchronization Issues**

During development, maintaining synchronization between local files and the GitHub repository led to frequent push and pull errors. Conflicts occurred when local commits were ahead of the remote repository, or when some files were mistakenly ignored. Learning Git commands such as pull --rebase and managing tracked vs. untracked files improved my understanding of version control systems. Although initially frustrating, these issues strengthened my ability to debug Git-related problems systematically.

**7. Balancing Performance and Output Quality**

Finding the right balance between processing speed and summary accuracy was challenging. Some models generated faster results but produced grammatically inconsistent summaries, while others offered better coherence at the cost of longer processing times. Fine-tuning parameters like maximum length, beam search, and temperature settings required experimentation to achieve an optimal trade-off between speed and quality.

**8. Documentation and Code Structuring**

Documenting the entire pipeline — from configuration files to summarization and paraphrasing modules — was also a learning experience. Maintaining modular code while keeping it readable and well-commented was difficult during early iterations. Creating separate files for extractive, abstractive, and paraphrasing logic using a consistent architecture helped in maintaining clarity and reusability, but it took time to plan and organize efficiently.

# 8. Learnings & Skills Acquired

**Technical Learnings**

* **Streamlit Development:**

I learned how to build end-to-end interactive web applications using Streamlit. This included creating user interfaces with text input boxes, buttons, and sidebars, and connecting them to backend Python logic for summarization and paraphrasing. I also understood how to maintain user sessions using session state and optimize performance with caching. Additionally, I explored layout design, dynamic updates, and custom styling to improve usability and user experience.

* **Hugging Face Models:**

I gained a strong understanding of transformer-based language models and how they are applied to summarization and paraphrasing. I learned how pre-trained models like BART and T5 convert long texts into concise summaries while preserving meaning. I explored model parameters like beam search and length control to fine-tune output quality. I also understood the advantages of using Hugging Face’s inference API for simplified deployment and scalability.

* **Environment Management:**

I learned how to securely handle API keys and environment variables using .env files. This helped me prevent sensitive information from being exposed in the project repository. I understood the role of dotenv in loading environment variables at runtime and the importance of using .gitignore to exclude confidential files. This practice ensured better security and portability of the project.

* **Version Control:**

I developed hands-on experience with version control using Git and GitHub. This included initializing repositories, committing changes, and synchronizing local and remote branches. I learned how to manage project updates systematically and resolve conflicts during collaboration. I also understood the importance of .gitignore files and maintaining clean, well-documented commit histories for team transparency and reproducibility.

* **Text Processing:**

I gained knowledge of how textual data is preprocessed before being passed to language models. This involved cleaning text, removing noise, and understanding tokenization — how text is broken into smaller units for model processing. I learned about token limits in transformer models and how to split long documents into smaller overlapping chunks to maintain context. This improved both accuracy and efficiency during summarization.

* **Error Handling and Debugging:**

Through this project, I learned how to identify and handle potential runtime errors systematically. This included managing API errors, invalid inputs, and connection issues. I also practiced providing user-friendly feedback through clear error messages in the interface. These skills improved the reliability and stability of the application during execution.

* **Dependency and Environment Management:**

I learned the importance of using a virtual environment to isolate project dependencies. Managing packages through a requirements.txt file helped ensure consistent setup across systems. This experience also improved my understanding of reproducibility and environment configuration in Python projects.

* **Integration and System Design:**

Finally, I understood how multiple components—frontend (Streamlit), backend (model pipelines), and configuration management—work together in a single application. I learned how modular code design enhances scalability, debugging, and maintenance. This helped me gain a complete understanding of how AI-based applications are structured from data input to model inference and output display.

**Soft Skills**

* Improved time management and systematic debugging skills.
* Enhanced ability to document and explain code functionality.
* Improved communication and collaboration in project versioning and reporting.

# 9. Testimonials from team

The project gave real-world exposure to developing NLP applications using open-source tools.  
It helped in understanding model pipelines, Streamlit deployment, and working on GitHub.  
The process enhanced both technical and analytical thinking, preparing for more complex AI projects in the future.

# 10. Conclusion

This internship was a highly valuable learning experience that blended the theoretical understanding of AI with practical implementation.  
Developing **TextMorph** provided hands-on exposure to summarization models, Streamlit, and NLP pipelines.  
The project strengthened Python and model integration skills while emphasizing real-world application design, efficient environment management, and clean code practices.  
It aligns with future career goals in AI and Machine Learning, improving readiness for advanced NLP and LLM-based systems.

# 11. Acknowledgements

I would like to express my sincere gratitude to **Infosys Springboard** for providing this internship opportunity and the platform to apply AI knowledge practically.  
Special thanks to my mentors for their constant guidance and feedback throughout the project development.  
Finally, I thank my peers and community members who supported and motivated me during every phase of this project.