# Project 2 Report Template: Scaling and Deploying AI Models

**1. Milestone 1: Designing for Non-Expert Users**

**• Briefly describe the model and its intended application.**

The project employs a DistilBERT-based model, a smaller and faster variant of the popular BERT (Bidirectional Encoder Representations from Transformers) language model. Specifically, the project utilizes a DistilBERT model (distilbert-base-uncased-finetuned-sst-2-english) that was initially trained for the task of sentiment analysis and fine-tuned for classifying SAT-2 English test essays. The model's role is to classify short pieces of academic English (for example, literature excerpts, essays, or grammatical mistakes) according to style, tone, or topic according to SAT-2 criteria.

A real-life application of the model might be useful for Students learn to analyze English texts, teachers can readily assess sample paragraphs, services for test preparation computerize writing style feedback or tone of the passage.

Rather than needing to be read manually by a specialist and labeled with its category, the model can instantly predict the most likely classification or emotional tone of any text (positive, negative, neutral), potentially saving a great deal of time and labor in mass SAT practice contexts.

**• What challenges might a non-expert face using your model?**

The following are the challenges a non-expert could face while using the model

1. The model leverages advanced libraries of Python such as transformers, torch, and streamlit, which are normally demanding of careful environment setup, GPU drivers, as well as ample disk storage.  
   End-users who are non-experts might be unaware of installing such dependencies, particularly for HPC environments with stringent quota constraints (e.g., ICE HPC). In addition, library variations can introduce errors (e.g., conflict of versions of Transformers and PyTorch).  
   Without experience, users may find it difficult to debug such compatibility problems.
2. Command-line interfaces (e.g., SSHing into ICE, activating Conda environments, running scripts such as bash run\_model.sh) are intimidating for non-technical users. Even minor mistakes (missing spaces, typographical errors, incorrect paths) cause failures, which are confusing without error tracing.
3. Without a guide, users can enter extremely long text blocks, fragmented sentences, or ill-formatted text, which will yield poor or even failed model predictions.
4. HPC users typically encounter disk quota, memory constraints, as well as job time constraints. They might not know how to monitor their utilization, so their job will fail.
5. Initial model loading, especially for Hugging Face models, takes tens of seconds. Users believe that the application is "broken" when, in fact, it is loading.

• **How did you design the interface or workflow to make it more user-friendly?**

- Built an interactive Streamlit web app for easy input and output.  
- Added help sections and sample examples to guide users.  
- Automated error handling for blank or invalid inputs.  
- Benchmarked inference times and displayed prediction results in a clean format.

**• Include screenshots or examples if applicable.**

A computer screen with a computer screen

AI-generated content may be incorrect.

A computer screen shot of a blue rectangle

AI-generated content may be incorrect.

## 2. Milestone 2: Connecting Local Execution to Remote Server

• Describe the setup of your local-to-remote pipeline.

• How did you handle input/output data transfer?

• Provide SLURM job script or details of other script configuration.

• Include benchmarking results comparing local and remote execution (e.g., time, memory).

• Insert any charts, tables, or graphs here.

## 3. Milestone 3: Publishing to Cybershuttle

• Describe how you packaged your notebook or application.

• What modifications were required to publish it in the Cybershuttle sandbox?

• Include a link to your deployed app or notebook.

• What user feedback (if any) did you consider or test against?

## 4. Performance Analysis and Optimization

• Describe any optimizations you implemented (e.g., batch size, precision, DeepSpeed).

• How did these affect model performance or usability?

• Include comparative results in tables or charts.

## 5. Reflection and Future Work

• What did you learn from this project?

• What were the biggest challenges, and how did you overcome them?

• What would you do differently in a future version of this project?

## 6. Appendix (Optional)

• Include any additional logs, screenshots, or configurations you want to reference.