Engineering Thesis Project Plan: Automated Manipulation Detection System

Phase 1: Model Training & Baseline Establishment

Goal: Create a specialized "Manipulation detection Expert" model by fine-tuning Bielik-4.5B on the manipulation part of MIPD dataset.

- **Key Tasks:**
  - **Data Prep:** Format MIPD dataset into Alpaca/ChatML format.
  - **Fine-Tuning:** Use Unsloth SFTTrainer (QLORA, 4-bit) to train the model.
  - **Evaluation:** Run inference on the MIPD Test Set (using the long-context strategy if needed).
  - **Benchmarking:** Calculate precision, recall, and F1-score for each manipulation tag.

Phase 2: User Interface & Inference Integration

Goal: Deploy the trained model locally using gguf model and adapter via Ollama and provide a user-friendly web interface.

- **Key Tasks:**
  - **Model Conversion:** Export the Unsloth adapter merged with the base model to GGUF format (q4_k_m).
  - **Inference Engine:** Setup Ollama locally with a custom Modelfile pointing to the GGUF.
  - **Frontend Development:** Build a React application with a chat/analysis interface.
  - **Integration:** Connect React to the Ollama API to stream responses.

Phase 3: Synthetic Data Generation (Chain-of-Thought)

Goal: Use a larger "Teacher Model" Qwen 2.5 7B to generate reasoning for the MIPD dataset to improve the "Student Model's" explainability and potentially quality.

- **Key Tasks:**
  - **Teacher Generation:** Run a script where the Teacher model looks at the input text + ground truth labels and writes an "Authoritative Reasoning."
  - **Hard-Constraint Filtering:** Ensure the generated reasoning contains verbatim names of the techniques and a mandatory anchor phrase for "clean" texts.
  - **Phase 3 Retraining:** Fine-tune the Bielik model again on this new "Text -> Reasoning + Labels" dataset.

Phase 4: MLOps Loop & Continuous Improvement

Goal: Create a system for continuous improvement and automated benchmarking.

- **Key Tasks:**
  - **Data Management UI:** A simple administration page to verify model errors and add them to a "New Training Data" bucket.
  - **Backend Orchestrator:** A Python script (FastAPI?) that triggers training jobs on the expanded dataset.
  - **Auto-Benchmark:** A script that automatically runs the test set on the newly trained adapter.
  - **Hot-Swap Logic:** Automated logic to update the local inference engine if a new model version achieves a higher F1-score