Of course. Here is a step-by-step guide to implementing the technical task from scratch, based on the provided instructions.

Given the deadline is 10 PM IST today, June 18, 2025, it's crucial to manage your time effectively.

Step 1: Project Setup & Environment

Create Project Directory: Make a root folder for your project. Inside, create the following structure:

data/

logs/

notebooks/ (for the training notebook)

src/ (for the agent script)

evaluation/

Set Up Virtual Environment: To avoid package conflicts, use a virtual environment.

Bash

python -m venv venv

source venv/bin/activate # On Windows, use `venv\Scripts\activate`

Install Core Libraries: Install essential Python libraries. You'll likely need torch, transformers, datasets, peft (for LoRA), bitsandbytes (for QLoRA), and accelerate.

Step 2: Data Preparation (Task 1.A)

Gather Q&A Pairs: Collect at least 150 question-and-answer pairs related to command-line tools like Git, Bash, tar, grep, and venv.

Sources: You can use public datasets from platforms like Hugging Face Datasets, or scrape Q&A sites like Stack Overflow. Ensure the data is publicly available and properly licensed.

Content: Questions should be natural language queries (e.g., "How do I create a new git branch?"). Answers should be a clear, step-by-step plan that includes the necessary shell commands.

Format and Validate:

Structure the data in a single JSON file (e.g., data/command\_qa.json). A good format is a list of objects, where each object has a "question" and "answer" key.

Validate the data by manually reviewing each pair to ensure the commands in the answers are correct and effectively solve the question.

Step 3: Model Fine-Tuning (Task 1.B)

Choose a Model: Select an open-weights model with 2 billion or fewer parameters, such as TinyLlama-1.1B or google/phi-2.

Set Up Colab: Open a new Google Colab notebook and ensure the runtime is set to use a T4 GPU.

Write the Training Script/Notebook:

Load Data: Load your JSON data file using the datasets library.

Load Model & Tokenizer: Load the pre-trained model and its corresponding tokenizer. For QLoRA, use bitsandbytes to load the model in 4-bit precision.

Configure LoRA/QLoRA: Use the peft library to create a LoraConfig. Specify the target modules (e.g., query, key, value layers), r (rank), and lora\_alpha.

Fine-Tune: Use the transformers.Trainer or a custom training loop. Fine-tune the model for one epoch on your prepared dataset.

Save Adapter: After training, save only the trained LoRA adapter files, not the full model. These will be significantly smaller. Download them and place them in your project directory.

Step 4: Build the CLI Agent (Task 1.C)

Create agent.py: In your src/ folder, create a Python script named agent.py.

Implement Agent Logic:

Argument Parsing: Use argparse or sys.argv to accept a natural language instruction from the command line. Example usage: python src/agent.py "Create a new Git branch and switch to it".

Load Model and Adapter: Write code to load the base model (e.g., Phi-2) and then apply your trained LoRA adapter to it.

Generate Plan: Pass the user's instruction to the fine-tuned model to generate a step-by-step plan.

Dry-Run Execution: Check if the first line of the generated plan is a shell command. If it is, do not execute it directly. Instead, print it to the console prefixed with echo.

Logging: For each step generated by the model, log the step to logs/trace.jsonl. Each line in this file should be a JSON object representing one step.

Step 5: Evaluation (Task 1.D)

Design Edge Cases: Create two unique and challenging test prompts that go beyond simple commands. Think about multi-step processes or ambiguous requests.

Static Evaluation (eval\_static.md):

Generate responses for the five test prompts from the instructions (§7) and your two edge cases using both the base model and your fine-tuned model.

Present these side-by-side in a new file evaluation/eval\_static.md.

For each pair of responses, calculate a BLEU or ROUGE-L score to measure textual similarity against a reference answer (you may need to write one yourself).

Dynamic Evaluation (eval\_dynamic.md):

Run your agent.py script with the same seven prompts.

Record the full output of the agent for each prompt in evaluation/eval\_dynamic.md.

Create a table where you score the quality of each generated plan on a scale of 0 (bad), 1 (partially correct), or 2 (perfect).

Step 6: Reporting & Documentation (Tasks 1.E & 3)

Create report.md: Write a one-page summary covering:

Data Sources: Where you got your Q&A pairs.

Hyper-parameters: The settings used for LoRA/QLoRA (e.g., rank, alpha, learning rate).

Cost/Time: How long the fine-tuning took on Colab.

Evaluation Results: A summary of your findings from the static and dynamic evaluations.

Improvement Ideas: Two concrete ideas for how to make your agent better.

Create README.md: Provide clear, end-to-end instructions on how to set up the environment, run the training, and use your agent.py script so that your work is reproducible.

Record a Demo Video: Create a short (under 5 minutes) video demonstrating your project's functionality.

Step 7: Final Submission (Task 4)

Organize Deliverables: Assemble all required files into a single folder:

README.md

data/ folder with the JSON file

Training notebook/script and the LoRA adapter files (ensure the total is under 500 MB)

src/agent.py script

evaluation/eval\_static.md and evaluation/eval\_dynamic.md

report.md

The demo video (e.g., demo.mp4)

Submit via Email:

Do NOT use public links or repositories.

Email the deliverables as a compressed archive (or a private, access-restricted link like Google Drive) to hr@fenrir-security.com.

Use the exact subject line: "AI/ML Internship Technical Task Submission - [Your Name]".

Ensure the email is sent before the hard deadline of 10 PM IST, June 18, 2025.

Remember the strict confidentiality policy: do not share any part of this task publicly.