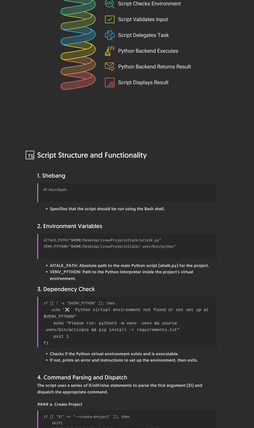
aitalk Bash Script Documentation

This document provides a comprehensive overview of the aitalk Bash script, which serves as a command-line interface (CLI) wrapper for the aitalk Python project. The script facilitates user interaction with various features of the project, including project creation, command explanation, chat functionality, git summary, and file summarization. It ensures that the correct Python environment is utilized and validates user input before delegating tasks to the Python backend.



Script Structure and Functionality

1. Shebang

```
#!/bin/bash
```

- Specifies that the script should be run using the Bash shell.

2. Environment Variables

```
AITALK_PATH="$HOME/Desktop/linuPro/pictalk/aitalk.py"
VENV_PYTHON="$HOME/Desktop/linuPro/pictalk/.venv/bin/python"
```

- `AITALK_PATH`: Absolute path to the main Python script (aitalk.py) for the project.
- `VENV_PYTHON`: Path to the Python interpreter inside the project's virtual environment.

3. Dependency Check

```
if [[ ! -x "$VENV_PYTHON" ]]; then
    echo ❌ "Python virtual environment not found or not set up at $VENV_PYTHON"
    echo "Please run: python3 -m venv .venv && source .venv/bin/activate && pip install -r requirements.txt"
    exit 1
fi
```

- Checks if the Python virtual environment exists and is executable.
- If not, prints an error and instructions to set up the environment, then exits.

4. Command Parsing and Dispatch

The script uses a series of if/elif/else statements to parse the first argument ($1) and dispatch the appropriate command.

a. Create Project

```
if [[ "$1" == "--create-project" ]]; then
    shift
    "$VENV_PYTHON" "$AITALK_PATH" --create-project "$@"
```

- If the first argument is --create-project, it shifts the arguments and calls the Python script with the remaining arguments.

b. Explain Last X Commands

```
elif [[ "$1" == --explain-[0-9]*4 ]]; then
    "$VENV_PYTHON" "$AITALK_PATH" "$1"
```

- If the first argument matches --explain-X (where X is a number), it passes the argument to the Python script.

c. Chat Mode

```
elif [[ "$1" == "--chat" ]]; then
    "$VENV_PYTHON" "$AITALK_PATH" --chat
```

- If the first argument is --chat, it launches the chat mode.

d. Git Summary

```
elif [[ "$1" == "--git-summary" ]]; then
    "$VENV_PYTHON" "$AITALK_PATH" --git-summary
```

- If the first argument is --git-summary, it runs the git summary feature.

e. Summarize File

```
elif [[ "$1" == "--summarize" ]]; then
    PROMPT="$2"
    FILE="$3"
    if [[ -z "$PROMPT" || -z "$FILE" ]]; then
        echo ❌ Usage: aitalk --summarize \"<prompt>\" <file>"
        exit 1
    fi
    if [[ ! -f "$FILE" ]]; then
        echo ❌ File not found: $FILE"
        exit 1
    fi
    "$VENV_PYTHON" "$AITALK_PATH" --summarize "$PROMPT" "$FILE"
```

- If the first argument is --summarize, it expects a prompt and a file path.
- Validates that both are provided and that the file exists.
- If valid, calls the Python script with the prompt and file.

f. Help

```
elif [[ "$1" == "--help" ]]; then
    echo "Usage:"
    echo "  aitalk --create-project \"make a react app\""
    echo "  aitalk --explain-10"
    echo "  aitalk --git-summary"
    echo "  aitalk --summarize \"summarize this file\" file.txt"
    echo "  aitalk --chat"
    echo "  aitalk --help"
    exit 0
```

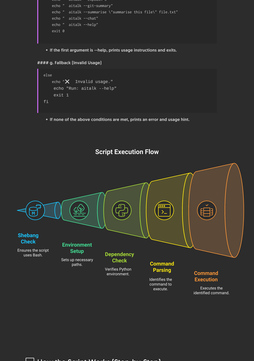
- If the first argument is --help, prints usage instructions and exits.

g. Fallback (Invalid Usage)

```
else
    echo ❌ Invalid usage."
    echo "Run: aitalk --help"
    exit 1
fi
```

- If none of the above conditions are met, prints an error and usage hint.

Script Execution Flow



How the Script Works (Step-by-Step)

1. Checks for the Python virtual environment.
2. Parses the first command-line argument to determine which feature to run.
3. Validates input (e.g., checks for required arguments and file existence).
4. If valid, delegates control with the correct arguments using the virtual environment's Python interpreter.
5. Handles errors and provides helpful messages for missing dependencies, invalid usage, or missing files.

Bash Script Execution Cycle

Supported Commands

- `--create-project "<description>"`: Create a new project with the given description.
- `--explain-X`: Explain the last X commands (where X is a number).
- `--git-summary`: Summarize the current git repository.
- `--summarize "<prompt>" <file>`: Summarize the given file with a custom prompt.
- `--chat`: Start an interactive chat session.
- `--help`: Show usage instructions.

Which command should be executed?

Best Practices

- Always activate your virtual environment and install dependencies before running the script.
- Use the provided commands as shown in the help output.
- Keep the script updated if you add new features to your Python backend.

Script Maintenance Cycle

Extending the Script

To add new features:

- Add a new elif block for your new command.
- Validate input as needed.
- Call the Python backend with the appropriate arguments.

Adding New Features to a System

Troubleshooting

- If you see ❌ Python virtual environment not found..., set up your virtual environment as instructed.
- If you see ❌ File not found: ..., check your file path.
- For any other usage errors, run aitalk --help for guidance.

This script is a robust entry point for your aitalk project, ensuring a smooth user experience and reliable backend integration.

Steps to Resolve Aitalk Errors