Openfabric Solution - main.py

```
import logging
from typing import Dict

from ontology_dc8f06af066e4a7880a5938933236037.config import ConfigClass
from ontology_dc8f06af066e4a7880a5938933236037.input import InputClass
from ontology_dc8f06af066e4a7880a5938933236037.output import OutputClass
from openfabric_pysdk.context import AppModel, State
from core.stub import Stub

# Configurations for the app
configurations: Dict[str, ConfigClass] = dict()

############################################################
# Config callback function
############################################################
def config(configuration: Dict[str, ConfigClass], state: State) -> None:
    """
    Stores user-specific configuration data.

    Args:
        configuration (Dict[str, ConfigClass]): A mapping of user IDs to configuration objects.
        state (State): The current state of the application (not used in this implementation).
    """
    for uid, conf in configuration.items():
        logging.info(f"Saving new config for user with id:'{uid}'")
        configurations[uid] = conf


############################################################
# Execution callback function
############################################################
def execute(model: AppModel) -> None:
    """
    Main execution entry point for handling a model pass.

    Args:
        model (AppModel): The model object containing request and response structures.
    """

    # Retrieve input
    request: InputClass = model.request

    # Retrieve user config
    user_config: ConfigClass = configurations.get('super-user', None)
    logging.info(f"{configurations}")

    # Initialize the Stub with app IDs
    app_ids = user_config.app_ids if user_config else []
    stub = Stub(app_ids)
```

```
# ------------------------------
# Simple NLP logic: Reverse prompt & truncate to 10 words
# ------------------------------
words = request.prompt.split()
summary = " ".join(words[:10][::-1])  # Reverse first 10 words
model.response.message = f"Summary: {summary}"
```