**Team 7**

**Team Members:**

1. Desaboina Suren Kumar
2. Shivendra Giri

**Project Statement:**

The purpose of this project is to create a Python function that manages network configurations by reading from a properties file (icfg.conf), modifying specific parameters, and outputting the results in both properties and JSON formats. The project also includes system command execution based on the operating system (Linux or Windows) and appends the output to the JSON file.

**Functionality**  
**Key Features**

**1. Configuration File Handling:**

* Reads network parameters from ifcfg.conf.
* Prompts the user to enter new values for IPADDR and PREFIX.

**2. Parameter Modification:**

* Allows users to modify various network parameters through a menu interface.
* Supports interactive updates for parameters like device name, protocol, network type, UUID, hardware address, etc.

**3. Command Execution Based on OS:**

* Detects the operating system (Linux, Windows, macOS) and executes relevant network commands.
* Captures and stores the output of these commands in the JSON file.

**4. File Output:**

* Writes modified configurations to net\_ifcfg.conf.
* Outputs the entire configuration, including command results, to net\_ifcfg.json.

A screenshot of a computer

Description automatically generated

Fig. Output

**Implementation Details:**

* **Main Function: netinfo()**

The core function, netinfo(), manages the reading, modification, and writing of network configurations.

* **Function Breakdown:**
* **File Existence Check**: Ensures ifcfg.conf exists before proceeding.
* **Read Configurations**: Reads the configuration file and stores parameters in a dictionary.
* **User Input**: Prompts for new IPADDR and PREFIX.
* **Modification Loop**: Allows users to modify parameters interactively.
* **Output Files**: Writes modified configurations to net\_ifcfg.conf and net\_ifcfg.json.

A screenshot of a computer

Description automatically generated

Fig.net\_ifcfg.json file

A screenshot of a computer

Description automatically generated

Fig.net\_ifcfg.conf file

* **Menu Function: menu(choice, config\_key\_list, config\_dict, toggle)**

Handles user input for parameter modifications and updates the configuration dictionary based on the selected parameter.

* **Protocol Option Function: protocol\_option(option)**

Processes user choices for Yes/No parameters, returning the corresponding string value.

**How to Use**

1. **Prepare the Configuration File**: Ensure the ifcfg.conf file is present in the same directory as the script.
2. **Run the Script**: Execute the script using Python 3.x.
3. **Follow Prompts**: Input values as prompted to modify network parameters and complete the configuration.
4. **Review Output Files**: Check net\_ifcfg.conf and net\_ifcfg.json for the updated configurations.

**Dependencies:**

* Python 3.x
* Standard libraries: os, platform, json, subprocess

**Error Handling:**

The tool includes error handling for:

* Checking file existence.
* Validating user input.
* Handling subprocess execution errors.

**Conclusion:**

This project provides a comprehensive solution for managing network configurations through a user-friendly interface. Future enhancements could include additional parameters for modification and improved error handling for a more robust experience.