Step 1: Implement Pre-commit Hook Scripts

Email Validation Script (pre-commit hook):
Trailing Whitespace Detection Script (pre-commit hook):

Create a file named *.git/hooks/pre-commit* in your project repository and add the following script to it.

```
#!/usr/bin/env python3

import re
import subprocess
import sys

def check_email():
    email_regex = r"^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$"
    git_author_email = subprocess.check_output(["git", "config",
"user.email"]).decode().strip()
    if not re.match(email_regex, git_author_email):
        print("Error: Incorrect email format. Please configure correct
email in your Git settings.")
        sys.exit(1)

def check_trailing_whitespaces():
    against = "HEAD" if subprocess.call(["git", "rev-parse", "--verify",
"HEAD"]) == 0 else "4b825dc642cb6eb9a060e54bf8d69288fbee4904"
    if subprocess.call(["git", "diff-index", "--check", "--cached",
against, "--"]) != 0:
        print("Error: Trailing whitespaces found in the changes being
committed.")
        sys.exit(1)

if __name__ == "__main__":
    check_email()
    check_trailing_whitespaces()
    sys.exit(0)
```

If needed, make the file executable.

```
attrib +x .git\hooks\pre-commit
```

Step 2: Implement Pre-commit Hook with Pre-commit Tool
Install the pre-commit tool: pip install pre-commit
Create a file named *.pre-commit-config.yaml* in your project directory and configure your hooks. For example:

```
repos:
-   repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v3.2.0
    hooks:
    -   id: check-yaml
    -   id: end-of-file-fixer
    -   id: trailing-whitespace
    -   id: check-added-large-files
    -   id: debug-statements
        language_version: python3
-   repo: https://github.com/psf/black
    rev: 22.10.0
    hooks:
    -   id: black
        args: [--safe]
```

Run '*pre-commit install'* to set up the hooks.

Step 3: Implement Commit-msg Hook for Commit Message Validation
Create a file named *.git/hooks/commit-msg* and add the following script to it. This script checks if the commit message contains valid tags:

```
#!/bin/bash

# Commit-msg hook to check commit message format for commit types

# Function to check if the commit message contains valid tags
check_commit_message() {
    commit_msg_file="$1"
    commit_msg=$(cat "$commit_msg_file")

    # Regular expression to match valid commit types (feat, fix, refactor, 
etc.)

commit_type_regex='^(feat|fix|refactor|chore|docs|style|test|perf|build|ci|
revert)(\(.+\))?: .+'
```

```
    if [[ ! "$commit_msg" =~ $commit_type_regex ]]; then
        echo "Error: Invalid commit message format. Please include a valid
commit type (feat, fix, refactor, etc.) in the commit message."
        echo "Aborting commit."
        exit 1
    fi
}

# Call the function to check the commit message
check_commit_message "$1"

# Exit with success status if the commit message format is valid
exit 0
```

Make sure to mark the script as executable:

```
attrib +x .git\hooks\commit-msg
```