Project Report

Project Overview

The project aims to create a graphical user interface (GUI) application for managing Docker containers and virtual machines. The application is divided into three main functionalities:

1. **Docker Management:**
    - Create Dockerfile
    - Build Docker Image
    - Create Container
    - List Docker Images
    - List Running Containers
    - Stop Container
2. **Docker Image Search:**
    - Search for Docker images on DockerHub
    - Search for local Docker images
    - Pull Docker images from DockerHub
3. **Virtual Machine Management:**
    - Create a virtual machine interactively
    - Create a virtual machine from a configuration file

Design

The project is implemented in Python using the tkinter library for the graphical interface. It provides a user-friendly interface to interact with Docker and VirtualBox. The main features include creating Dockerfiles, building Docker images, managing containers, searching for Docker images, and managing virtual machines.

Docker Management

The Docker management features allow users to create Dockerfiles, build Docker images, create containers, list Docker images, list running containers, and stop containers. The GUI provides buttons for each functionality, making it easy for users to interact with Docker.

Docker Image Search

The Docker Image Search module enables users to search for Docker images on DockerHub, search for local Docker images, and pull Docker images from DockerHub. The search results are displayed in a new window, and users can pull the selected image with the click of a button.

Virtual Machine Management

The virtual machine management section provides two options: creating a virtual machine interactively and creating a virtual machine from a configuration file. Users can input parameters such as VM name, memory size, and disk size. The configuration file option allows for automating VM creation.

Challenges Faced

1. **User Input Validation:**
    - Validating user input for Docker image names, tags, and container IDs to ensure they meet the required criteria.
2. **Exception Handling:**
    - Handling exceptions when interacting with Docker and VirtualBox, providing informative error messages to the user.
3. **GUI Design:**
    - Designing an intuitive and user-friendly GUI to facilitate easy navigation and interaction.

Solutions Implemented

1. **Input Validation Functions:**
    - Implemented functions to validate Docker image names and tags using regular expressions.
2. **Exception Handling:**
    - Incorporated try-except blocks to catch and handle exceptions during Docker and VirtualBox operations, displaying meaningful error messages.
3. **GUI Design Considerations:**
    - Organized buttons and functionality logically.
    - Used proper font, colors, and layout for a visually appealing interface.

Testing Methodologies

1. **Unit Testing:**

    - Unit tests were conducted for individual functions, ensuring they produced the expected output.

2. **Integration Testing:**

    - Tested the integration of different modules to ensure smooth communication between Docker, Docker Image Search, and Virtual Machine Management components.

3. **User Acceptance Testing:**

    - Conducted testing with potential users to gather feedback on the usability and intuitiveness of the GUI.

Evaluation of System Performance

Docker Management

- **Build Docker Image:**
    - Test Case 1: Successfully built a Python image.
    - Test Case 2: Successfully built a Java image.
    - Test Case 3: Invalid input for image name resulted in an appropriate error message.
- **Create Container:**
    - Test Case 1: Successfully created a container from a valid image.
    - Test Case 2: Attempted to create a container with an invalid image, resulting in an error message.
- **List Docker Images:**
    - Test Case 1: Successfully listed Docker images.
    - Test Case 2: Handled API errors gracefully.
- **List Running Containers:**
    - Test Case 1: Successfully listed running containers.
    - Test Case 2: Handled API errors gracefully.

- **Stop Container:**
    - Test Case 1: Successfully stopped a running container.
    - Test Case 2: Attempted to stop a nonexistent container, resulting in an appropriate error message.

Docker Image Search

- **Search on DockerHub:**
    - Test Case 1: Successfully retrieved search results.
    - Test Case 2: Attempted to search with an empty input, resulting in an error message.
- **Search Image (Local Storage):**
    - Test Case 1: Successfully searched for local images.
    - Test Case 2: Attempted to search with an empty input, resulting in an error message.
- **Download/Pull Image:**
    - Test Case 1: Successfully pulled a valid image from DockerHub.
    - Test Case 2: Attempted to pull an invalid image, resulting in an appropriate error message.

Virtual Machine Management

- **Create VM Interactively:**
    - Test Case 1: Successfully created a virtual machine with valid inputs.
    - Test Case 2: Attempted to create a virtual machine with invalid inputs, resulting in an appropriate error message.
- **Create VM from Config:**
    - Test Case 1: Successfully created a virtual machine from a valid configuration file.
    - Test Case 2: Attempted to create a virtual machine with an invalid configuration file, resulting in an appropriate error message.

Conclusion

The project successfully delivers a user-friendly interface for Docker and VirtualBox management. It provides essential features for Docker image and container management, Docker image search, and virtual machine creation. The implementation is robust, handling various scenarios and providing informative feedback to users.

Testing Report

Testing Environment

- **Operating System:** Windows 10 or above
- **Python Version:** 3.9
- **Docker Version:** 20.10.8
- **VirtualBox Version:** 6.1.26

Test Cases

Docker Management

Build Docker Image

1. **Test Case:** Build Python Image
    - **Input:** Choose Python option, provide a valid directory containing a Dockerfile and requirements.txt.
    - **Expected Output:** Docker image built successfully.
2. **Test Case:** Build Java Image
    - **Input:** Choose Java option, provide a valid directory containing a Dockerfile.
    - **Expected Output:** Docker image built successfully.
3. **Test Case:** Invalid Image Name
    - **Input:** Choose an invalid image name with special characters.
    - **Expected Output:** Display an error message indicating an invalid image name.

Create Container

1. **Test Case:** Create Container Successfully
    - **Input:** Provide a valid image name and container name.
    - **Expected Output:** Container created successfully.
2. **Test Case:** Invalid Image Name
    - **Input:** Provide an invalid image name.
    - **Expected Output:** Display an error message indicating an invalid image name.

List Docker Images

1. **Test Case:** List Images Successfully
    - **Input:** Click on the "List Docker Images" button.
    - **Expected Output:** Display a list of Docker images.
2. **Test Case:** API Error
    - **Input:** Simulate an API error.
    - **Expected Output:** Display an error message indicating an issue listing Docker images.

List Running Containers

1. **Test Case:** List Containers Successfully
    - **Input:** Click on the "List Running Containers" button.
    - **Expected Output:** Display a list of running containers with a "Stop" button for each.
2. **Test Case:** API Error
    - **Input:** Simulate an API error.
    - **Expected Output:** Display an error message indicating an issue listing running containers.

Stop Container

1. **Test Case:** Stop Container Successfully
    - **Input:** Provide a valid container ID.
    - **Expected Output:** Container stopped successfully.
2. **Test Case:** Container Not Found
    - **Input:** Provide a nonexistent container ID.
    - **Expected Output:** Display an error message indicating that the container was not found.

3. **Test Case:** API Error

    - **Input:** Simulate an API error.
    - **Expected Output:** Display an error message indicating an issue stopping the container.

Docker Image Search

Search on DockerHub

1. **Test Case:** Search Successfully
    - **Input:** Provide a valid image name for search.
    - **Expected Output:** Display a list of search results with a "Pull" button for each.
2. **Test Case:** Empty Input
    - **Input:** Attempt to search with an empty input.
    - **Expected Output:** Display an error message indicating that an image name must be entered.

Search Image (Local Storage)

1. **Test Case:** Search Successfully
    - **Input:** Provide a valid image name for local search.
    - **Expected Output:** Display a list of local images.
2. **Test Case:** Empty Input
    - **Input:** Attempt to search with an empty input.
    - **Expected Output:** Display an error message indicating that an image name must be entered.

Download/Pull Image

1. **Test Case:** Pull Successfully

    - **Input:** Provide a valid image name for pulling from DockerHub.
    - **Expected Output:** Display a message indicating that the image is being pulled.

2. **Test Case:** Pull Invalid Image

    - **Input:** Provide an invalid image name.
    - **Expected Output:** Display an error message indicating that the image was not found on DockerHub.

Virtual Machine Management

Create VM Interactively

1. **Test Case:** Create VM Successfully
    - **Input:** Enter valid VM name, memory, and disk size.
    - **Expected Output:** Virtual machine created successfully.
2. **Test Case:** Invalid Inputs
    - **Input:** Enter invalid memory and disk size.
    - **Expected Output:** Display an error message indicating invalid inputs.

Create VM from Config

1. **Test Case:** Create VM Successfully from Config
    - **Input:** Provide a valid configuration file.
    - **Expected Output:** Virtual machine created successfully.
2. **Test Case:** Invalid Config File
    - **Input:** Provide an invalid configuration file.
    - **Expected Output:** Display an error message indicating that the config file was not found or is invalid.

Conclusion

The testing process covered various scenarios for Docker and Virtual Machine management, ensuring the application's robustness and user-friendly behavior. The project demonstrates effective integration of different components, providing a comprehensive solution for managing Docker containers and virtual machines. The application handles errors gracefully and delivers meaningful feedback to users.