

VirtuManager

Virtualization and Containerization GUI Tool

Developed by:

Enjy Ramadan

Mariam Kandeel

Mazen Ashraf

Sohaila Ashraf

Omar Sherif

[Project Demo](#)

Project Overview:

Overview:

VirtuManager is a user-friendly desktop application that simplifies virtualization and containerization tasks using a clean graphical interface. Built with Python's Tkinter and customtkinter, the tool integrates with both VirtualBox and Docker, allowing users to manage virtual machines and build Docker images effortlessly, all from one place.

Phase One: Virtual Machine Management with VirtualBox

In the first phase, VirtuManager focused on simplifying virtual machine lifecycle management using VirtualBox. This included the ability to:

Core Functionalities:

- **List Existing VMs:** View a real-time list of available virtual machines along with their states (Running, Powered Off, etc.).
- **Create VMs Easily:** Input VM name, type, RAM size, disk size, and OS ISO image — all through the interface. The app will configure and create the VM accordingly.
- **Start/Stop/Delete VMs:** Control VM states with a single click, including launching VMs in headless or normal mode.
- **Update VM Settings:** Modify key VM configurations like RAM, CPU, and disk space.
- **Navigation System:** Each major function (Create, List, Update, Delete) is accessible through a streamlined and intuitive navigation bar.

The controller logic for all VirtualBox-related operations is abstracted cleanly in the `controllerVM.py` module, which interacts with system commands using the `subprocess` module. The UI files handle only layout and navigation, following the MVC pattern for better scalability and maintenance.

Phase Two: Docker Container Management Integration

Expanding the virtualization scope, Phase Two introduces Docker support — allowing users to build container images and manage them using the same GUI system.

Docker Features Added:

Build Docker Image GUI:

- Users can specify:
 - Image Name and Tag

- Path to Dockerfile
 - Build Context Folder
- These inputs are handled through a modern UI form with proper labels, spacing, and optional browsing buttons.

Browse Files and Directories:

- Select Dockerfile via `filedialog.askopenfilename`
- Choose context folder via `filedialog.askdirectory`
- Automatically updates the entry field upon selection

Build Progress Bar:

- A dynamic progress bar gives visual feedback during the image build process.
- Simulated progress is shown while Docker builds in the background using a threaded function, ensuring the UI remains responsive.

Notifications and Error Handling:

- After the building finishes, success or failure is displayed via `messagebox.showinfo()` or `messagebox.showerror()`.
- All exceptions and errors during the build are caught and presented clearly.

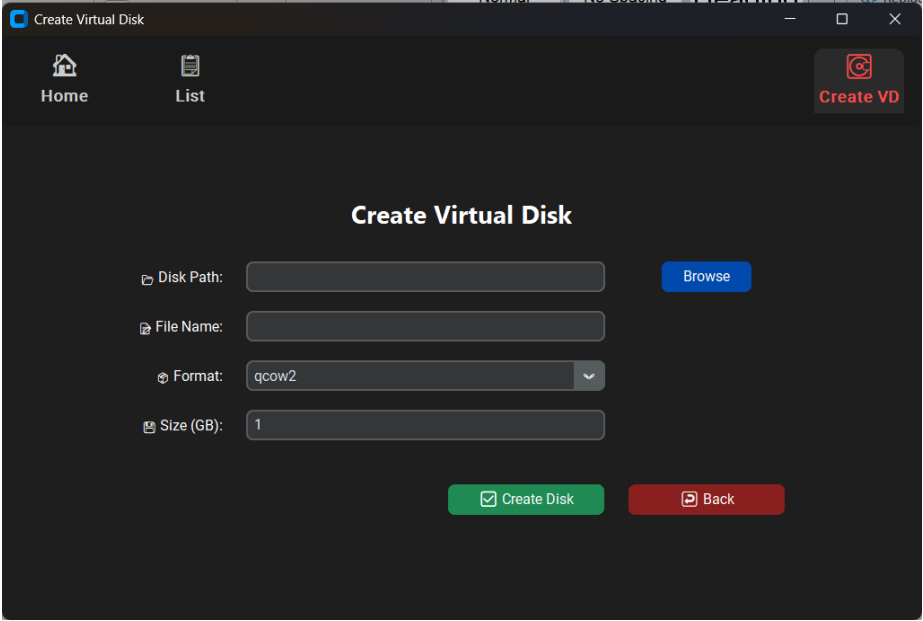
Clean Navigation Flow:

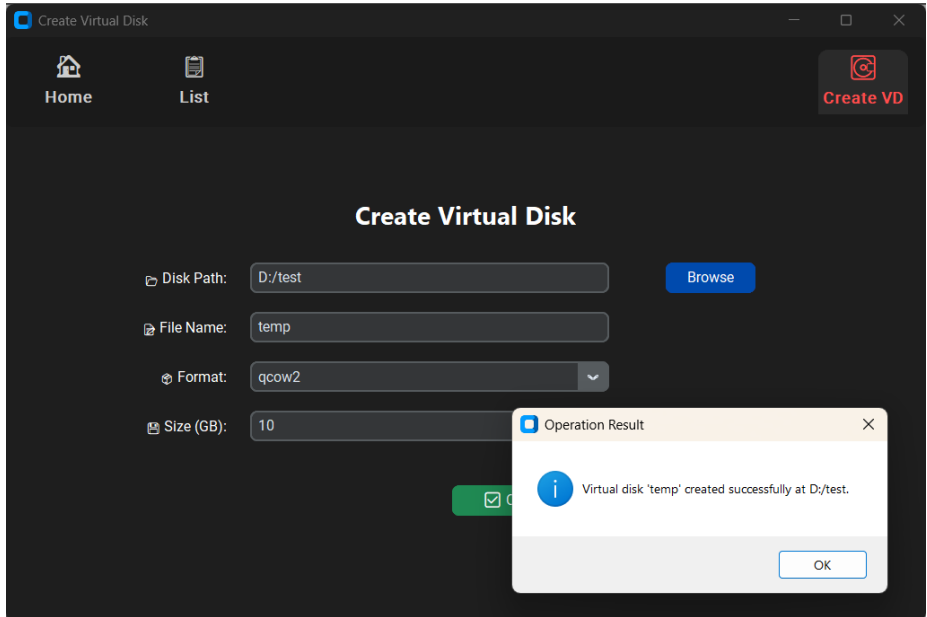
- Users can return to the Dockerfile creation page or main menu using a clear back button.
- The controller logic is delegated to `controllerDocker.py`, preserving the MVC structure.

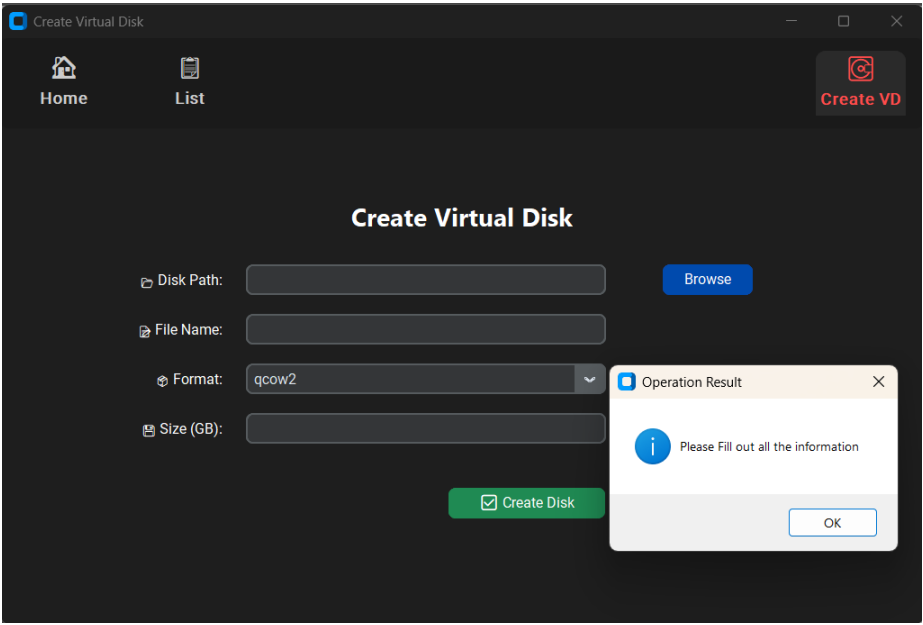
Implemented Features:

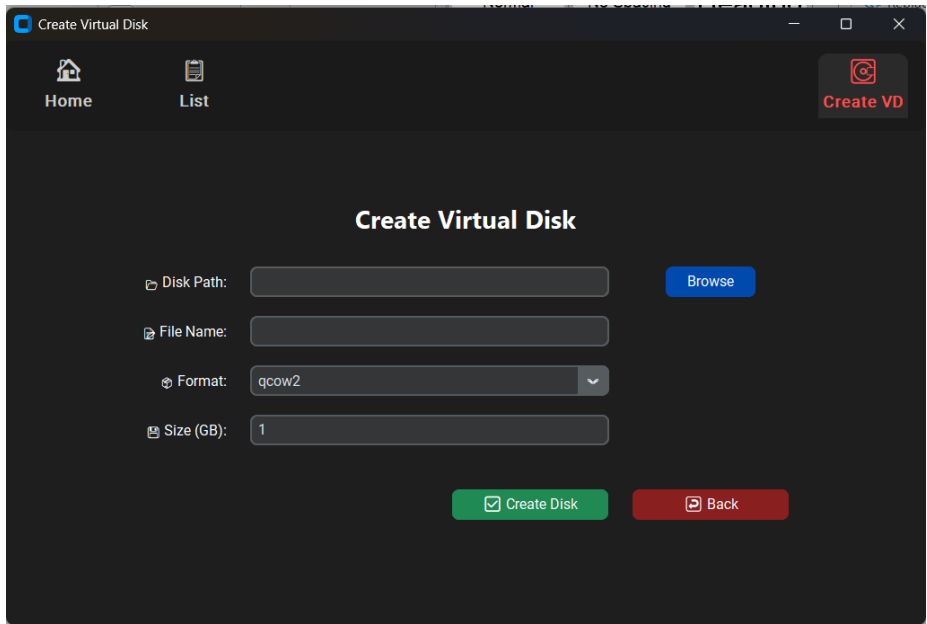
- **Virtual Disk Creation:** Allows users to create virtual disks with specified parameters such as name, path, format, and size.
- **Virtual Machine Creation:** Facilitates creating VMs by selecting the required CPU, RAM, disk, and ISO image.
- **System Resource Validation:** Ensures that the system has sufficient CPU, RAM, and disk space before VM creation to prevent resource shortages.
- **Support for Multiple Disk Formats:** Includes various virtual disk formats like qcow2, vmdk, vdi, raw, vhd, vhdx, and more, enabling flexibility in virtual machine setups.
- **Docker Image Building:** Enables users to build Docker images via a graphical interface by selecting image name, tag, Dockerfile, and context folder.
- **Progress Monitoring for Docker Build:** Includes a dynamic progress bar and real-time feedback for build status.
- **Docker Build Error Handling:** User-friendly messages inform users about successful builds or any issues during the build process.

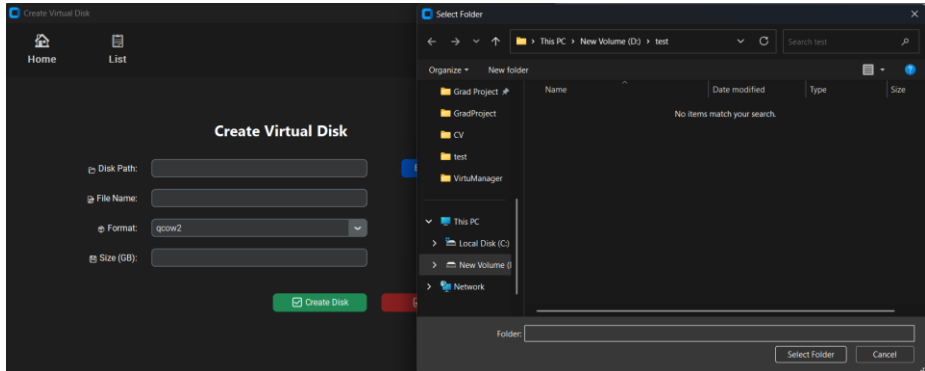
Test Casses:

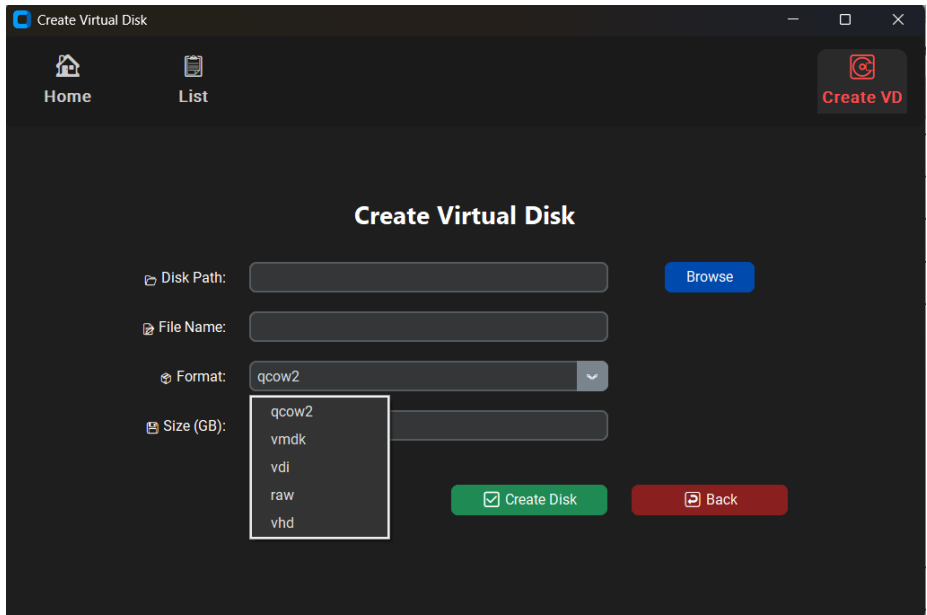
Test Case Name	TVD1: Verify Page Loads Successfully
Test Scenario	Ensure the Create Virtual Disk page opens correctly.
Prerequisites	-
Test Input	Open the Create Disk page
Execution Steps	1. Launch the app. 2. Click on “VD”
Expected Behavior	The Create Virtual Disk form loads without errors
Assumptions	UI and navigation work
Actual Results	UI and navigation work
Status	Pass
Real Life Testing	

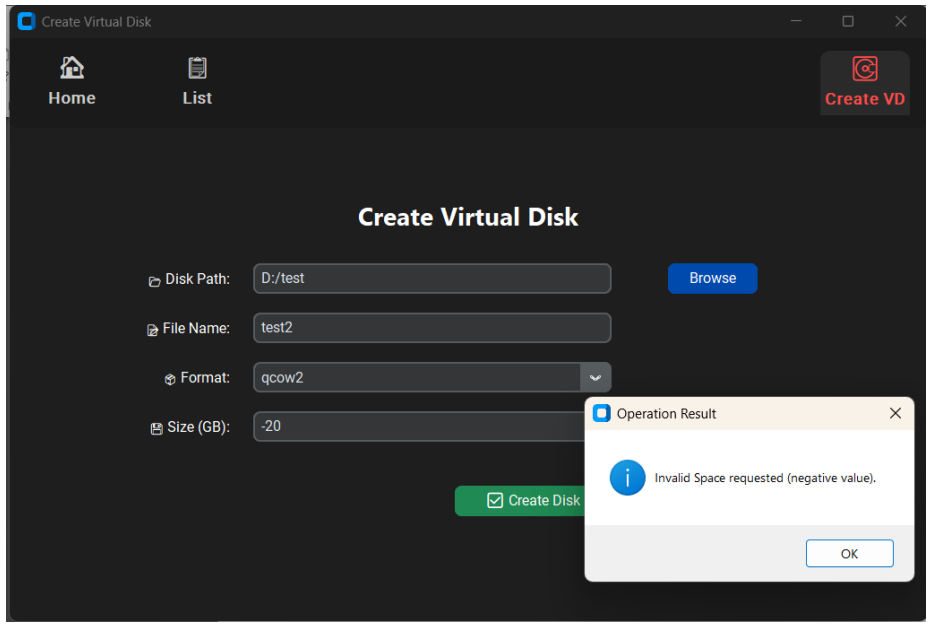
<i>Test Case Name</i>	TVD2: Submit with All Valid Data
<i>Test Scenario</i>	It should create a disk successfully.
<i>Prerequisites</i>	Virtualization engine functional
<i>Test Input</i>	<ol style="list-style-type: none">Valid pathValid formatValid size
<i>Execution Steps</i>	<ol style="list-style-type: none">Fill all fields correctlyClick “Create Disk”
<i>Expected Behavior</i>	Disk is created
<i>Assumptions</i>	Backend supports operation
<i>Actual Results</i>	Disk is created successfully and ready to function
<i>Status</i>	Pass
<i>Real Life Testing</i>	

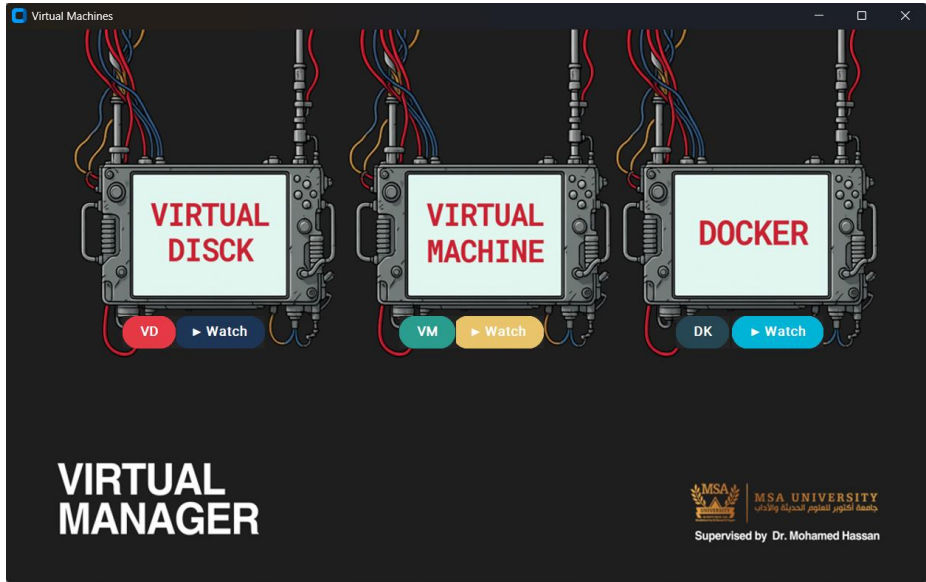
<i>Test Case Name</i>	TVD3: Submit with Empty Fields
<i>Test Scenario</i>	Submit button should validate empty input.
<i>Prerequisites</i>	The form is visible
<i>Test Input</i>	Leave all fields blank
<i>Execution Steps</i>	1. Click “Create Disk” without filling anything
<i>Expected Behavior</i>	Error shown or disk not created
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present
<i>Status</i>	Pass
<i>Real Life Testing</i>	 The screenshot shows a web application titled "Create Virtual Disk". It has a dark theme with a top navigation bar containing "Home" and "List" links, and a "Create VD" button. The main content area is titled "Create Virtual Disk" and contains four input fields: "Disk Path:", "File Name:", "Format:" (set to "qcow2"), and "Size (GB):". There is a "Browse" button next to the "Disk Path:" field and a green "Create Disk" button at the bottom. An "Operation Result" dialog box is open in the foreground, displaying an information icon and the message "Please Fill out all the information", with an "OK" button.

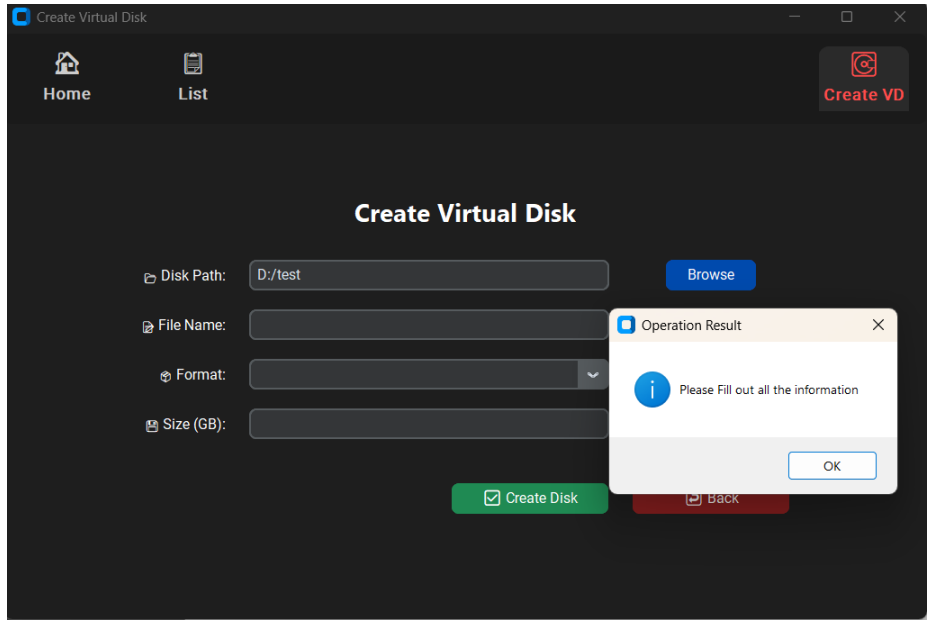
<i>Test Case Name</i>	TVD4: Verify Active States of Buttons
<i>Test Scenario</i>	Ensure that all buttons show appropriate visual feedback when clicked.
<i>Prerequisites</i>	The "Create Virtual Disk" page is fully loaded
<i>Test Input</i>	Interaction clicks with all buttons
<i>Execution Steps</i>	<ol style="list-style-type: none">1. Hover and click on the “Browse” button.2. Hover and click on the “Back” button.3. Hover and click on the “Create Disk” button.
<i>Expected Behavior</i>	Each button should visually be active
<i>Assumptions</i>	UI framework handles button states
<i>Actual Results</i>	Functioning buttons
<i>Status</i>	Pass
<i>Real Life Testing</i>	 A screenshot of a web browser window titled "Create Virtual Disk". The interface has a dark theme. At the top, there are navigation links for "Home" and "List", and a "Create VD" button. The main heading is "Create Virtual Disk". Below it, there are four input fields: "Disk Path:" with a "Browse" button, "File Name:", "Format:" (set to "qcow2"), and "Size (GB):" (set to "1"). At the bottom, there are two buttons: a green "Create Disk" button with a checkmark icon and a red "Back" button with a left arrow icon.

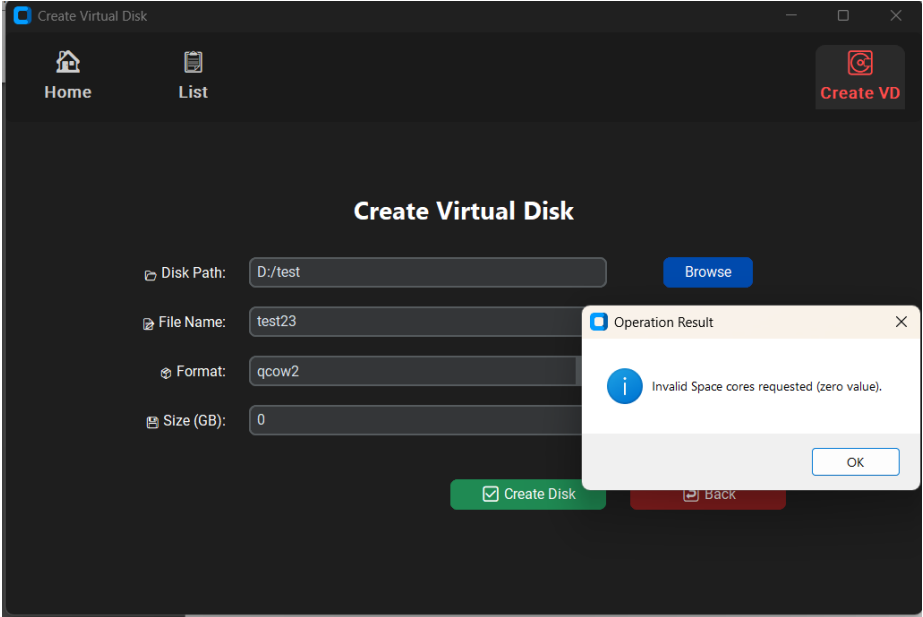
<i>Test Case Name</i>	TVD5: Verify “Disk Path” Browse Button Functionality
<i>Test Scenario</i>	Clicking “Browse” opens laptop search
<i>Prerequisites</i>	The form is visible
<i>Test Input</i>	Click “Browse”
<i>Execution Steps</i>	1. Click on “Browse”
<i>Expected Behavior</i>	File dialog opens for path selection
<i>Assumptions</i>	OS-level file dialog is supported
<i>Actual Results</i>	OS-level file dialog is supported
<i>Status</i>	Pass
<i>Real Life Testing</i>	 The image shows two overlapping windows from a Windows operating system. The background window is titled 'Create Virtual Disk' and contains fields for 'Disk Path', 'File Name', 'Format' (set to 'qcow2'), and 'Size (GB)'. A green 'Create Disk' button is at the bottom. The foreground window is a 'Select Folder' file explorer. The address bar shows the path 'This PC > New Volume (D:) > test'. The left sidebar shows a tree view with 'This PC', 'Local Disk (C:)', 'New Volume (I)', and 'Network'. The main pane shows a search for 'test' with no results found. At the bottom of the file explorer, there is a 'Folder:' text box and 'Select Folder' and 'Cancel' buttons.

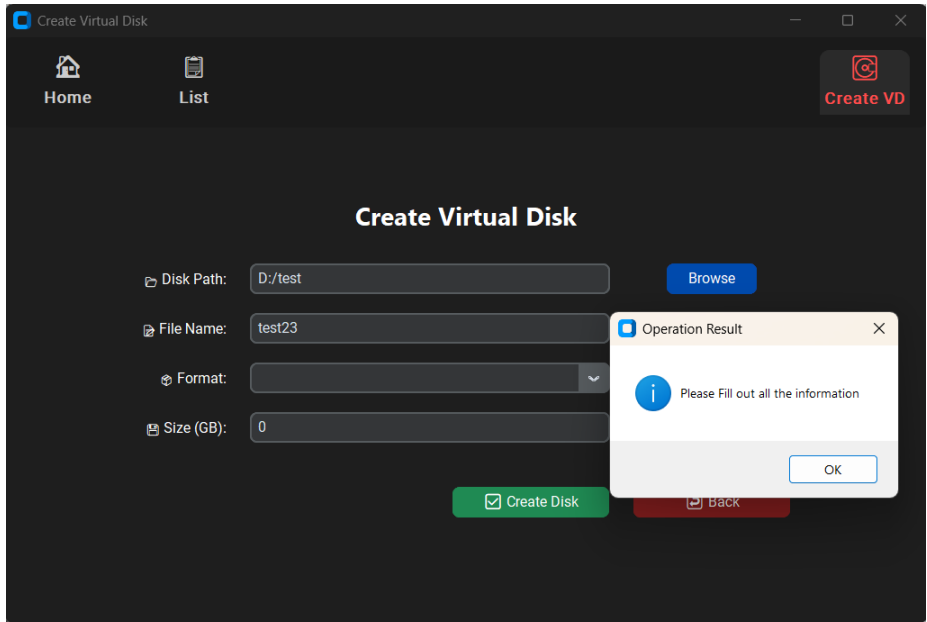
<i>Test Case Name</i>	TVD6: Verify Disk Format Dropdown Options
<i>Test Scenario</i>	Dropdown should list available formats.
<i>Prerequisites</i>	The form is visible
<i>Test Input</i>	Click on the dropdown
<i>Execution Steps</i>	1. Click the “Disk Format” dropdown
<i>Expected Behavior</i>	Shows valid formats like qcow2, vdi, vmdk, raw
<i>Assumptions</i>	Options are hardcoded or loaded correctly
<i>Actual Results</i>	Options are hardcoded or loaded correctly
<i>Status</i>	Pass
<i>Real Life Testing</i>	

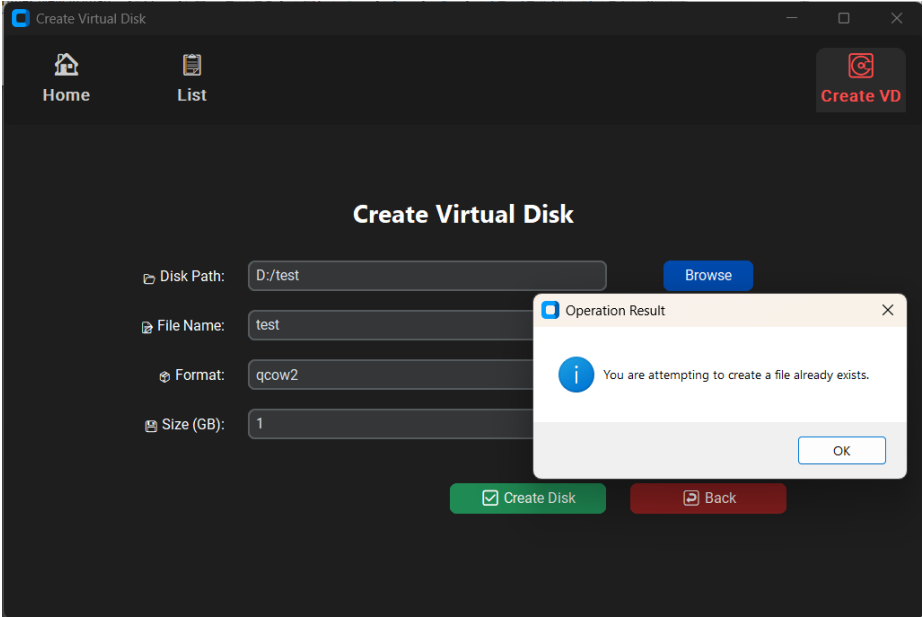
<i>Test Case Name</i>	TVD7: Verify Disk Size Spinner Limits
<i>Test Scenario</i>	Only valid sizes can be selected.
<i>Prerequisites</i>	The form is visible
<i>Test Input</i>	Use arrows to change size
<i>Execution Steps</i>	1. Adjust disk size
<i>Expected Behavior</i>	Only positive integers allowed
<i>Assumptions</i>	Spinner has limits set
<i>Actual Results</i>	Spinner has limits set
<i>Status</i>	Pass
<i>Real Life Testing</i>	

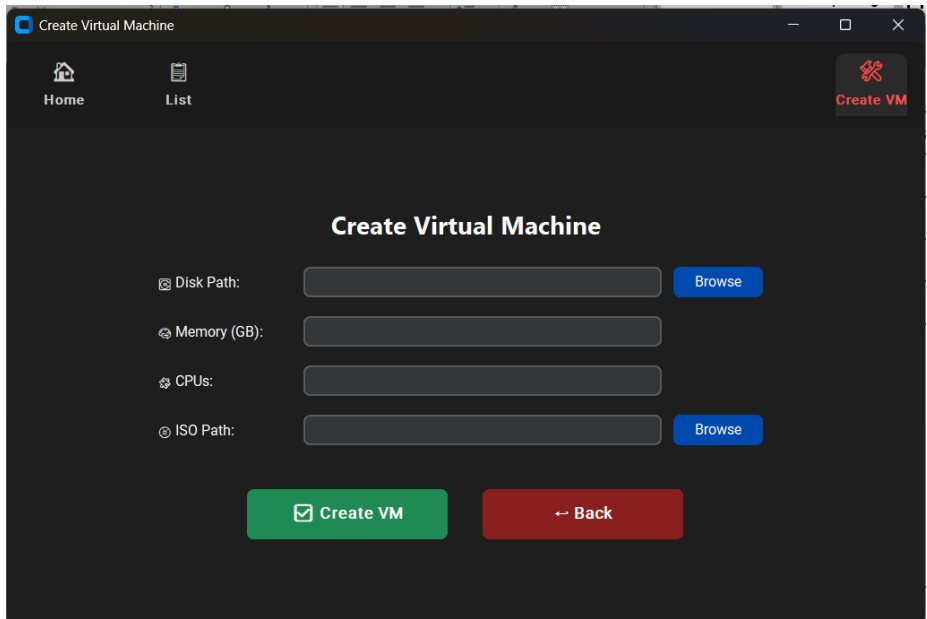
<i>Test Case Name</i>	TVD8: Verify Back Button Functionality
<i>Test Scenario</i>	Clicking “Back” returns to the previous page.
<i>Prerequisites</i>	Page is open
<i>Test Input</i>	Click Back
<i>Execution Steps</i>	1. Click “Back”
<i>Expected Behavior</i>	Redirects back
<i>Assumptions</i>	Navigation handler works
<i>Actual Results</i>	Goes back to the previous page
<i>Status</i>	Pass
<i>Real Life Testing</i>	

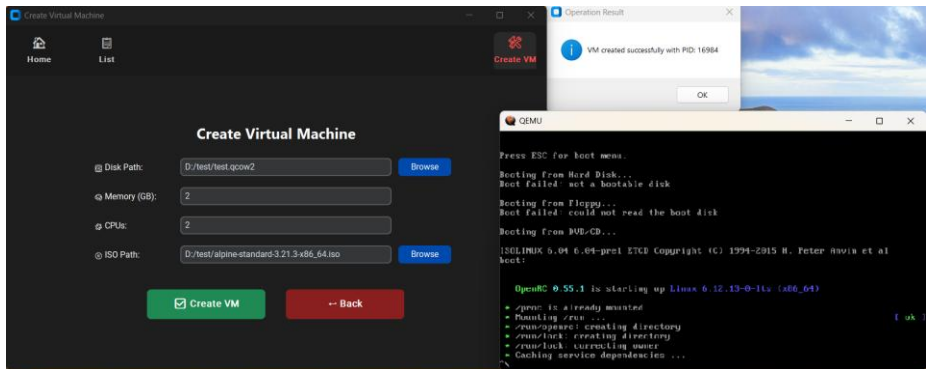
<i>Test Case Name</i>	TVD9: Submit with Only Disk Path Filled
<i>Test Scenario</i>	Incomplete submission should be rejected.
<i>Prerequisites</i>	The form is visible
<i>Test Input</i>	Only browse path
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Enter disk path 2. Click “Create Disk”
<i>Expected Behavior</i>	Error due to missing fields
<i>Assumptions</i>	Form requires all fields
<i>Actual Results</i>	Error message box pops up informing user about missing fields
<i>Status</i>	Pass
<i>Real Life Testing</i>	 <p>The screenshot shows the 'Create Virtual Disk' application interface. It has a dark theme with a top navigation bar containing 'Home' and 'List' icons, and a 'Create VD' button. The main area is titled 'Create Virtual Disk' and contains a form with the following fields: 'Disk Path' (with the value 'D:/test' and a 'Browse' button), 'File Name', 'Format' (a dropdown menu), and 'Size (GB)'. At the bottom of the form are two buttons: 'Create Disk' (green) and 'Back' (red). An 'Operation Result' dialog box is open in the foreground, displaying an information icon and the message 'Please Fill out all the information', with an 'OK' button.</p>

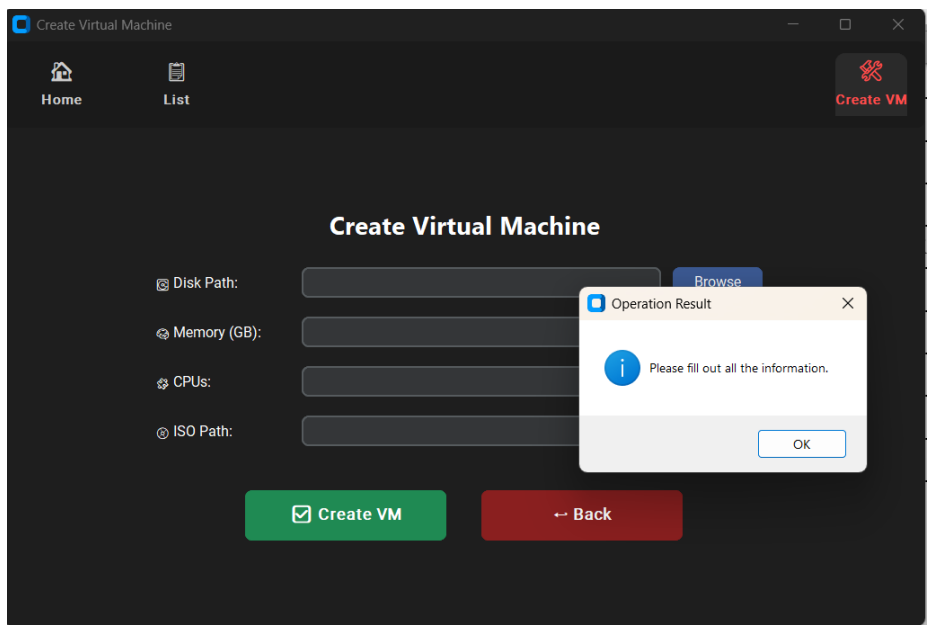
<i>Test Case Name</i>	TVD10: Disk Size Field with Zero
<i>Test Scenario</i>	Zero GB should not be allowed.
<i>Prerequisites</i>	The form is visible
<i>Test Input</i>	0 in Disk Size
<i>Execution Steps</i>	<ol style="list-style-type: none">1. Set size to 02. Submit
<i>Expected Behavior</i>	Validation error
<i>Assumptions</i>	Size must be >0
<i>Actual Results</i>	Message box pops up informing the user about the disk size error
<i>Status</i>	Pass
<i>Real Life Testing</i>	 The screenshot shows the 'Create Virtual Disk' application window. It has a dark theme with a top navigation bar containing 'Home' and 'List' icons, and a 'Create VD' button. The main area is titled 'Create Virtual Disk' and contains four input fields: 'Disk Path' (D:/test), 'File Name' (test23), 'Format' (qcow2), and 'Size (GB)' (0). A 'Browse' button is next to the 'Disk Path' field. At the bottom, there are 'Create Disk' and 'Back' buttons. An 'Operation Result' dialog box is open in the foreground, displaying an information icon and the message 'Invalid Space cores requested (zero value).', with an 'OK' button.

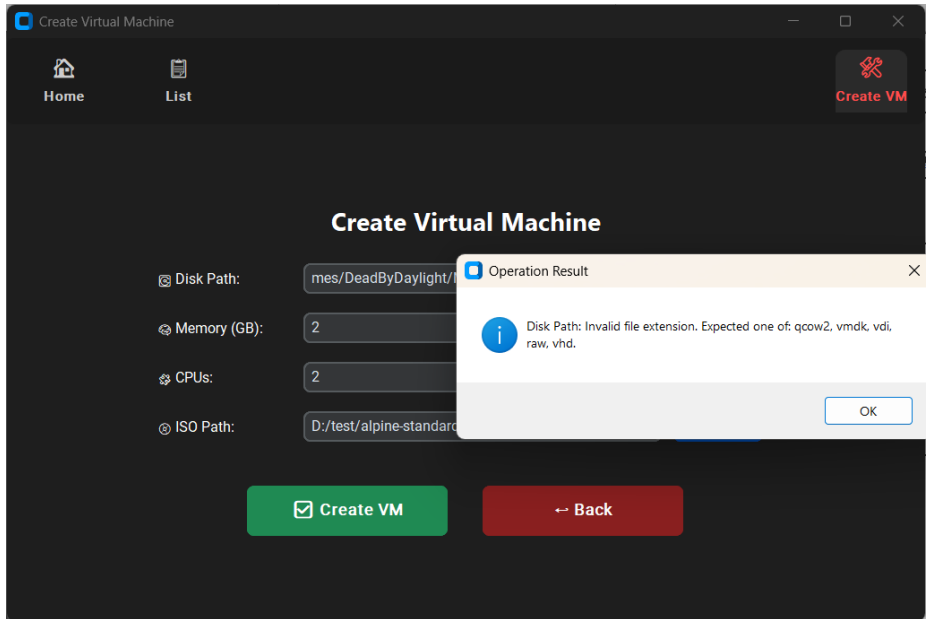
<i>Test Case Name</i>	TVD11: Disk Format Not Selected
<i>Test Scenario</i>	Must require a format.
<i>Prerequisites</i>	Other fields filled
<i>Test Input</i>	Leave format blank
<i>Execution Steps</i>	<ol style="list-style-type: none">1. Enter path and size2. Leave format3. Submit
<i>Expected Behavior</i>	Validation error
<i>Assumptions</i>	Size must be > 0
<i>Actual Results</i>	Message box pops up informing the user about the missing information error
<i>Status</i>	Pass
<i>Real Life Testing</i>	 <p>The screenshot shows the 'Create Virtual Disk' application interface. The title bar reads 'Create Virtual Disk'. The main window has a dark theme with a top navigation bar containing 'Home' and 'List' icons, and a 'Create VD' button. The main content area is titled 'Create Virtual Disk' and contains four input fields: 'Disk Path' (D:/test), 'File Name' (test23), 'Format' (empty), and 'Size (GB)' (0). There is a 'Browse' button next to the 'Disk Path' field. At the bottom, there are 'Create Disk' and 'Back' buttons. An 'Operation Result' dialog box is open in the foreground, displaying an information icon and the message 'Please Fill out all the information', with an 'OK' button.</p>

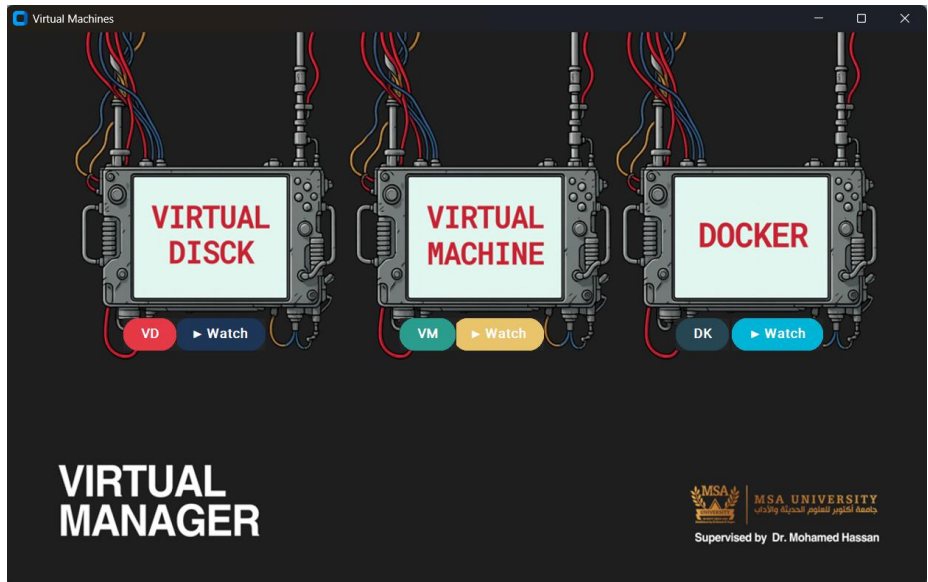
<i>Test Case Name</i>	TVD12: Disk Already Exists at Path
<i>Test Scenario</i>	Duplicate disk path should raise an error.
<i>Prerequisites</i>	File already exists at given path
<i>Test Input</i>	Use the same path as previous disk
<i>Execution Steps</i>	<ol style="list-style-type: none">1. Enter existing path2. Submit form
<i>Expected Behavior</i>	“File already exists” error
<i>Assumptions</i>	System checks for file existence
<i>Actual Results</i>	Message box pops up informing the user about the existing file
<i>Status</i>	Pass
<i>Real Life Testing</i>	

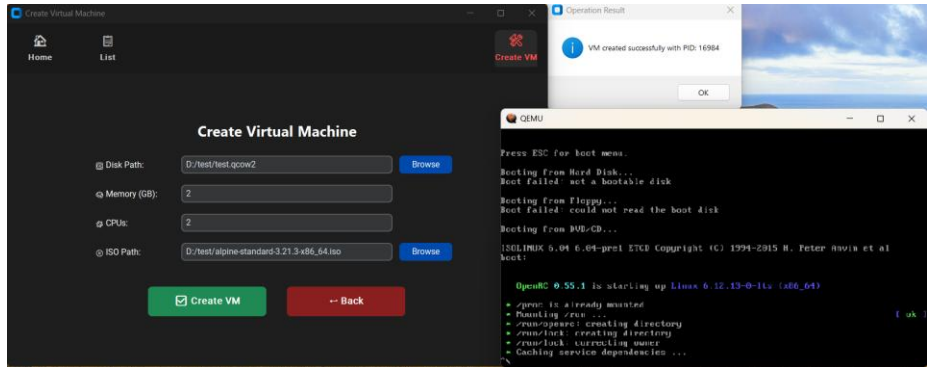
Test Case Name	TVM1: Verify Page Load
Test Scenario	Ensure the "Create Virtual Machine" page loads successfully.
Prerequisites	-
Test Input	Open the "Create VM" tab
Execution Steps	<ol style="list-style-type: none">1. Launch the app.2. Click on "VM"
Expected Behavior	The Create Virtual Machine form loads without errors
Assumptions	UI and navigation work
Actual Results	UI and navigation work
Status	Pass
Real Life Testing	

<i>Test Case Name</i>	TVM2: Submit with All Valid Data
<i>Test Scenario</i>	It should create a machine successfully.
<i>Prerequisites</i>	Virtualization engine functional
<i>Test Input</i>	<ol style="list-style-type: none"> 1. Disk input 2. Memory (GB) 3. Number of CPUs 4. ISO file
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Fill all fields correctly 2. Click “Create Machine”
<i>Expected Behavior</i>	Machine is created
<i>Assumptions</i>	Backend supports operation
<i>Actual Results</i>	Machine is created successfully and ready to function
<i>Status</i>	Pass
<i>Real Life Testing</i>	

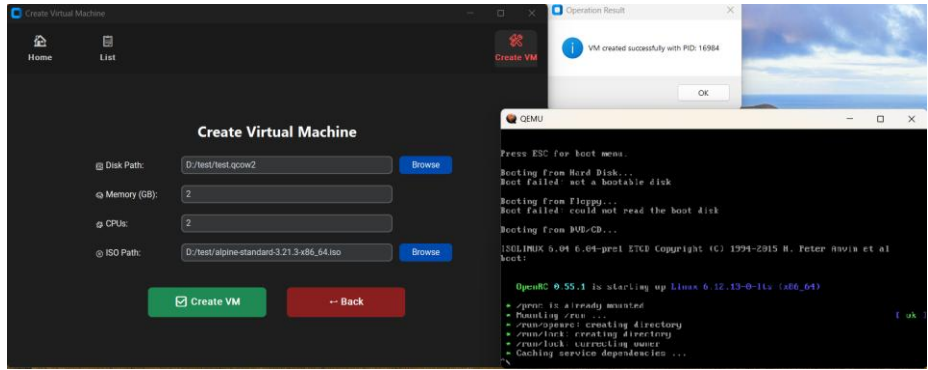
<i>Test Case Name</i>	TVM3: Submit with Empty Fields
<i>Test Scenario</i>	Submit button should validate empty input.
<i>Prerequisites</i>	The Form is visible
<i>Test Input</i>	Leave all fields blank
<i>Execution Steps</i>	1. Click “Create Machine” without filling anything
<i>Expected Behavior</i>	Error shown or machine not created
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present
<i>Status</i>	Pass
<i>Real Life Testing</i>	 A screenshot of a web application titled "Create Virtual Machine". The interface has a dark theme. At the top, there are navigation links for "Home" and "List", and a "Create VM" button. The main form has four input fields: "Disk Path:", "Memory (GB):", "CPUs:", and "ISO Path:". Each field is empty. Below the fields are two buttons: a green "Create VM" button and a red "Back" button. An "Operation Result" dialog box is open in the foreground, displaying an information icon and the text "Please fill out all the information." with an "OK" button.

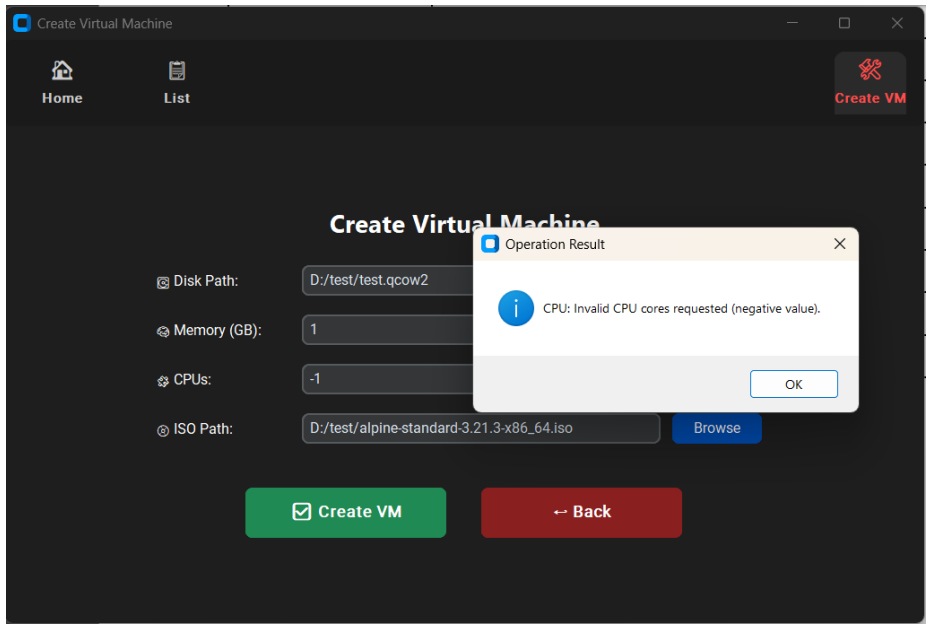
<i>Test Case Name</i>	TVM4: Validate Feedback on VM Creation Failure
<i>Test Scenario</i>	Show clear message if VM creation fails (e.g., missing folder or insufficient disk).
<i>Prerequisites</i>	Known failure scenario exists (like in image: not enough space)
<i>Test Input</i>	Set disk to invalid path ”D:/Games/DeadByDaylight/Manifest_NonUFSFiles_EGS.txt”
<i>Execution Steps</i>	1. Fill form with faulty disk path. 2. Click "Create VM".
<i>Expected Behavior</i>	An error message box pops up informing the user.
<i>Assumptions</i>	Errors returned from backend
<i>Actual Results</i>	Errors returned from backend
<i>Status</i>	Pass
<i>Real Life Testing</i>	

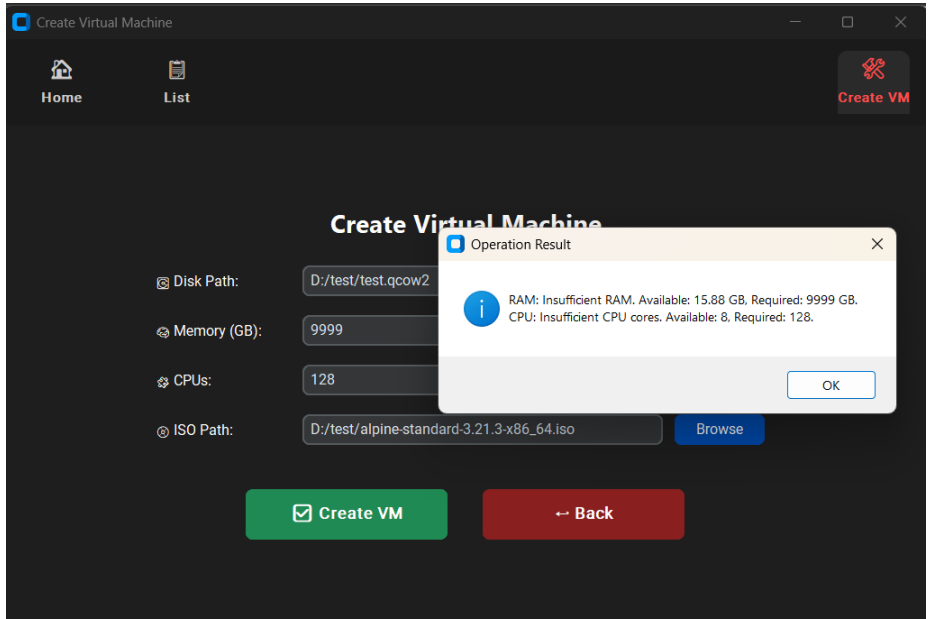
<i>Test Case Name</i>	TVM5: Verify Back Button Functionality
<i>Test Scenario</i>	Clicking “Back” returns to the previous page.
<i>Prerequisites</i>	Page is open
<i>Test Input</i>	Click Back
<i>Execution Steps</i>	2. Click “Back”
<i>Expected Behavior</i>	Redirects back
<i>Assumptions</i>	Navigation handler works
<i>Actual Results</i>	Goes back to the previous page
<i>Status</i>	Pass
<i>Real Life Testing</i>	

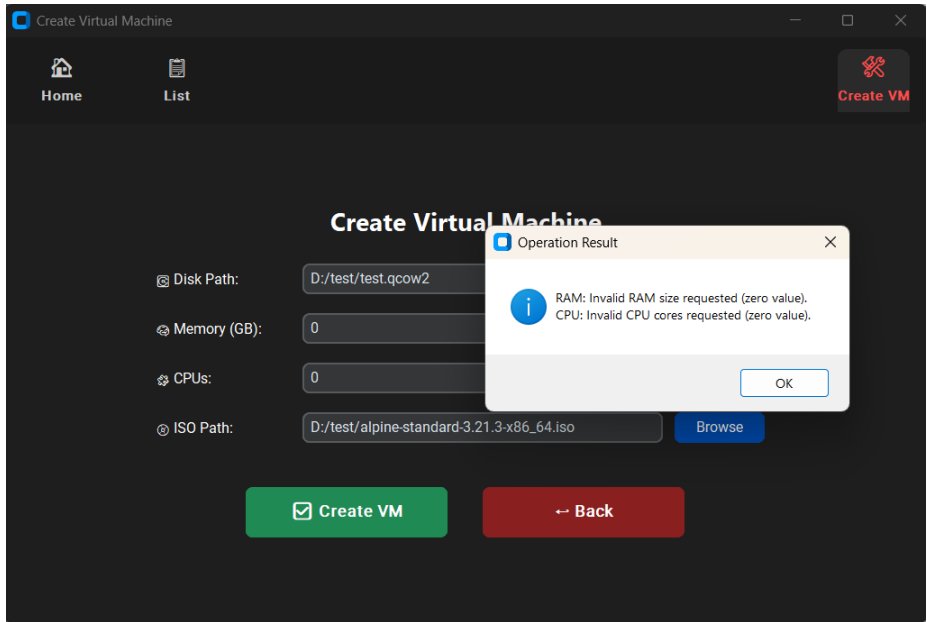
Test Case Name	TVM6: Validate Memory Field (Positive Integer)
Test Scenario	Ensure the user can only enter a positive integer for memory.
Prerequisites	Input box for Memory is active
Test Input	“2”
Execution Steps	<ol style="list-style-type: none">1. Click on the Memory field.2. Enter a value like 2.
Expected Behavior	Field accepts the number and stores it as an integer
Assumptions	Validation is present
Actual Results	Validation is present
Status	Pass
Real Life Testing	

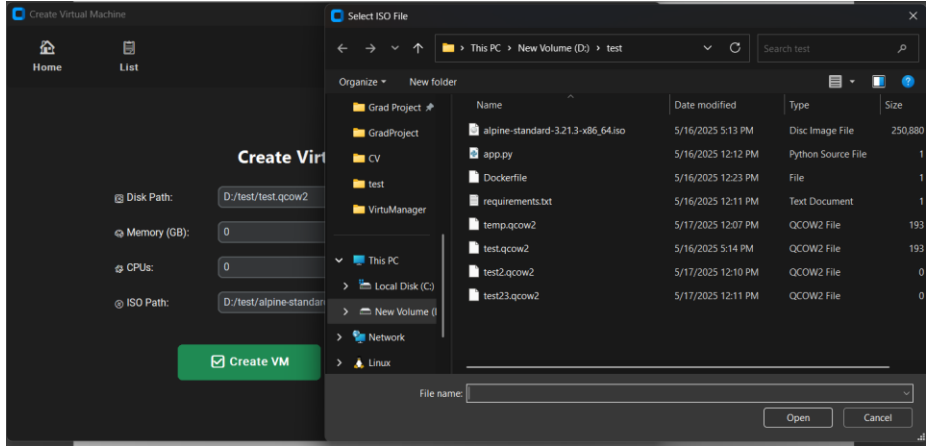
Test Case Name	TVM7: Validate Memory Field (Invalid Input)
Test Scenario	Ensure the field rejects negative values.
Prerequisites	Memory field input not restricted by default
Test Input	“-1”
Execution Steps	<div>1. Click Memory field.</div> <div>2. Enter invalid input.</div>
Expected Behavior	Field blocks invalid input or displays error
Assumptions	Validation is present
Actual Results	Validation is present
Status	Pass
Real Life Testing	<div><div>Create Virtual Machine</div><div><div>HomeList</div><div>Create VM</div></div><div>Create Virtual Machine</div><div><div>Disk Path: D:/test/test.qcow2</div><div>Memory (GB): -1</div><div>CPU: 1</div><div>ISO Path: D:/test/alpine-standard-3.21.3-x86_64.iso</div></div><div>Operation Result</div><div>RAM: Invalid RAM size requested (negative value).</div><div>OK</div><div>Create VMBack</div></div>

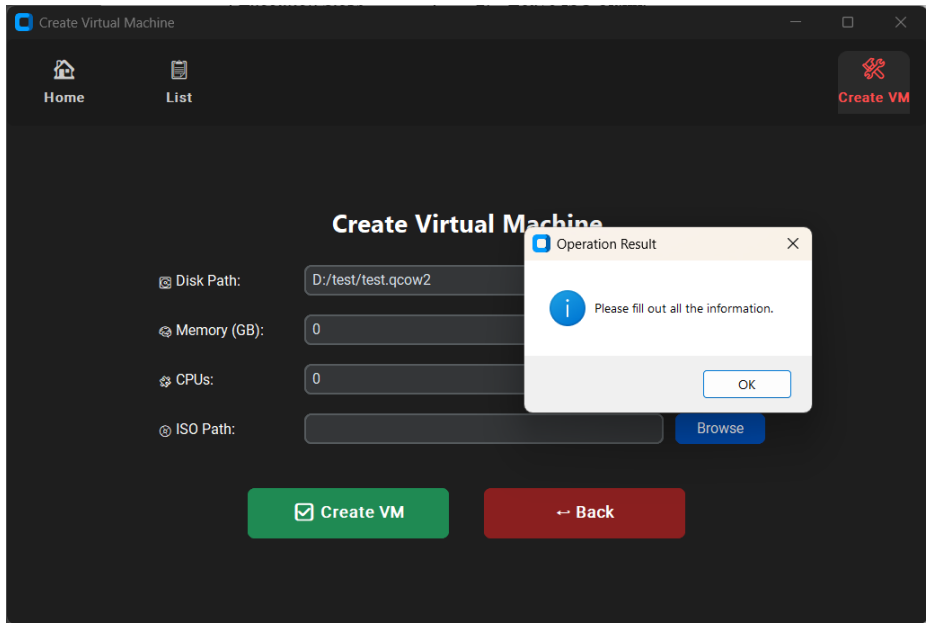
Test Case Name	TVM8: Validate CPUs Field (Positive Integer)
Test Scenario	Ensure the user can only enter valid CPU core counts.
Prerequisites	The CPU field is editable
Test Input	“2”
Execution Steps	1. Enter value in CPUs field.
Expected Behavior	Field accepts integer values
Assumptions	Validation is present
Actual Results	Validation is present
Status	Pass
Real Life Testing	

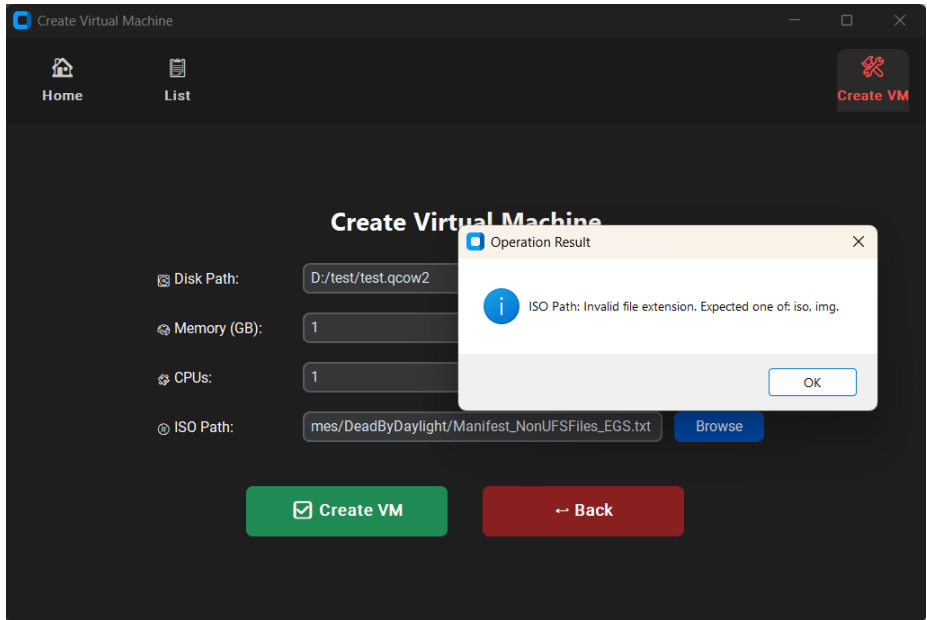
<i>Test Case Name</i>	TVM9: Validate CPUs Field (Negative Integer)
<i>Test Scenario</i>	Ensure the user can only enter valid CPU core counts.
<i>Prerequisites</i>	The CPU field is editable
<i>Test Input</i>	"-1"
<i>Execution Steps</i>	1. Enter value in CPUs field.
<i>Expected Behavior</i>	Field does not accept negative integer values
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present
<i>Status</i>	Pass
<i>Real Life Testing</i>	

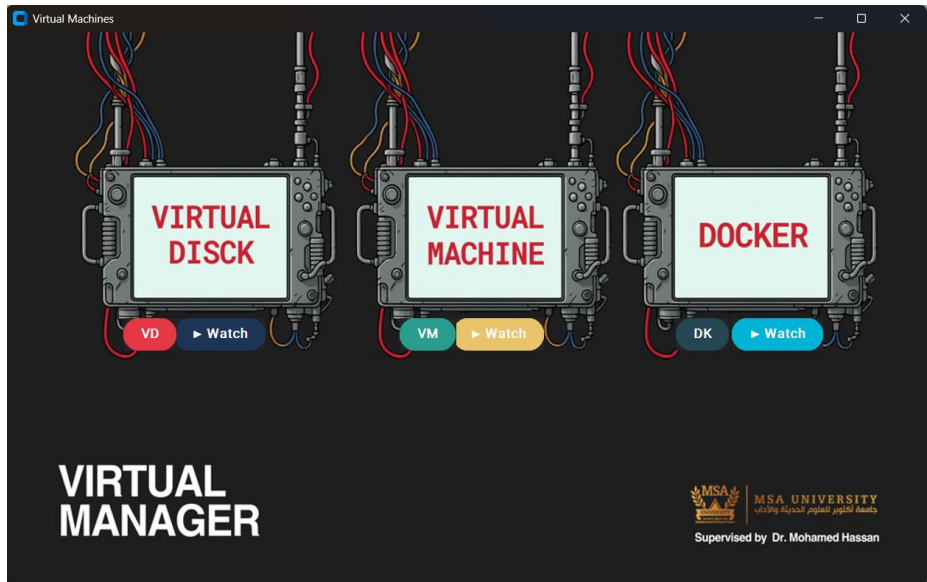
<i>Test Case Name</i>	TVM10: Verify Memory and CPU Limits
<i>Test Scenario</i>	Ensure application restricts memory and CPU input to realistic values.
<i>Prerequisites</i>	Limits defined in code
<i>Test Input</i>	Enter extreme values Memory= “9999”, CPU= “128”
<i>Execution Steps</i>	1. Enter large numbers in both fields.
<i>Expected Behavior</i>	Error message is shown
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present
<i>Status</i>	Pass
<i>Real Life Testing</i>	

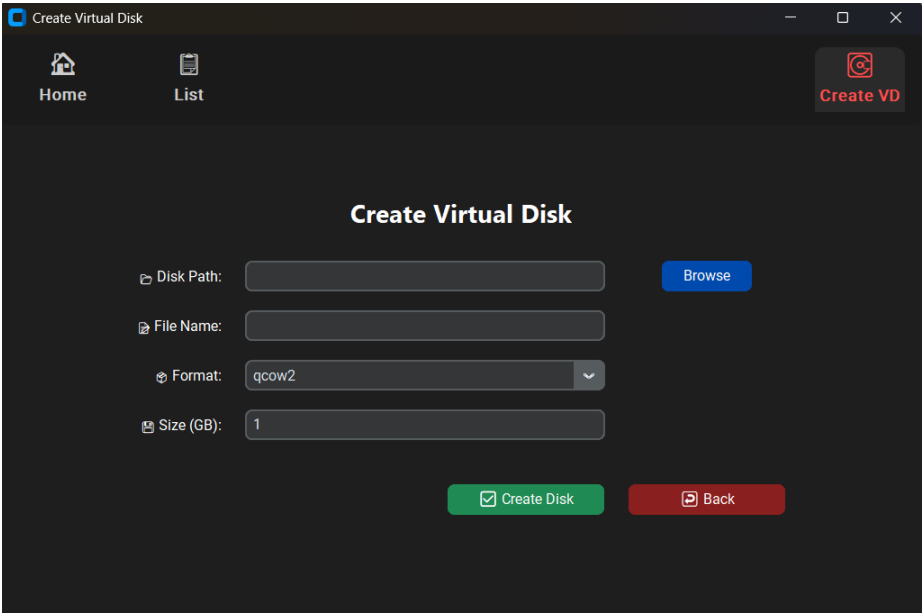
<i>Test Case Name</i>	TVM11: Verify Minimum Memory/CPU Requirement
<i>Test Scenario</i>	Prevent VM creation if memory/CPU value is below minimum allowed.
<i>Prerequisites</i>	Backend enforces min values.
<i>Test Input</i>	Memory: 0 / CPU: 0
<i>Execution Steps</i>	<ol style="list-style-type: none">1. Set memory and CPU values to 0.2. Click Create VM.
<i>Expected Behavior</i>	Error message is shown
<i>Assumptions</i>	Error message informing the user about the incorrect value
<i>Actual Results</i>	Error message informing the user about the incorrect value
<i>Status</i>	Pass
<i>Real Life Testing</i>	 The screenshot shows a web application titled "Create Virtual Machine". It has a dark theme with a top navigation bar containing "Home" and "List" icons, and a "Create VM" button in the top right. The main form has four input fields: "Disk Path" (D:/test/test.qcow2), "Memory (GB)" (0), "CPUs" (0), and "ISO Path" (D:/test/alpine-standard-3.21.3-x86_64.iso). There is a "Browse" button next to the ISO Path field. At the bottom are two large buttons: a green "Create VM" button and a red "Back" button. An "Operation Result" dialog box is open in the center, displaying an information icon and the text: "RAM: Invalid RAM size requested (zero value). CPU: Invalid CPU cores requested (zero value)." with an "OK" button.

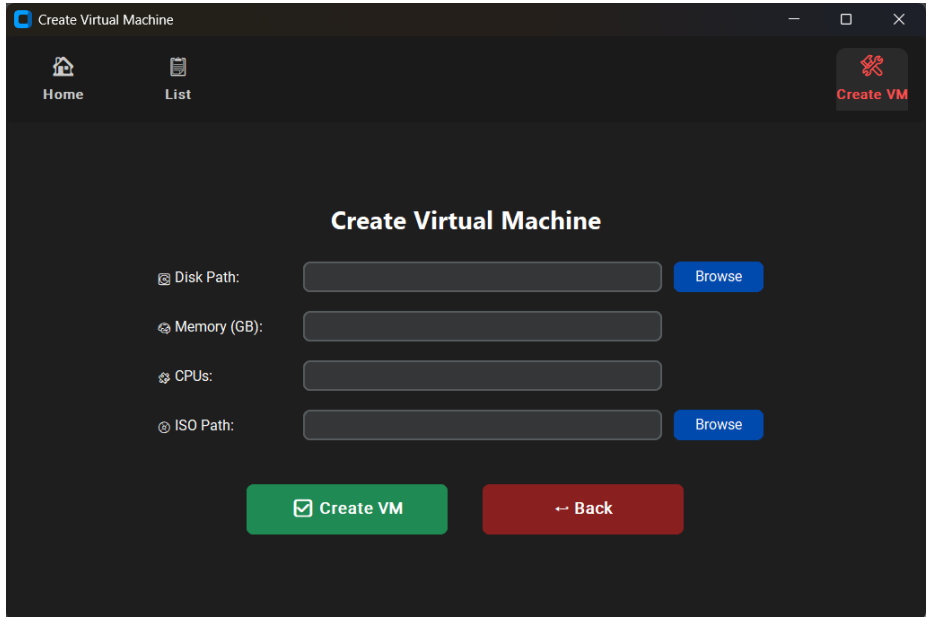
Test Case Name	TVM12: Validate ISO Path Selection
Test Scenario	Ensure users can select ISO file.
Prerequisites	ISO files are available in file system
Test Input	Click "Browse"
Execution Steps	<ol style="list-style-type: none">1. Click "Browse" next to ISO Path.2. Select the ISO file.
Expected Behavior	ISO Path field updates with file location
Assumptions	ISO file format supported
Actual Results	ISO file format supported
Status	Pass
Real Life Testing	 <p>The screenshot displays the 'Create Virtual Machine' wizard on the left and a Windows File Explorer window on the right. The wizard shows fields for Disk Path (D:/test/test.qcow2), Memory (0 GB), CPUs (0), and ISO Path (D:/test/alpine-standard-3.21.3-x86_64.iso), with a 'Create VM' button at the bottom. The File Explorer window is titled 'Select ISO File' and shows the path 'This PC > New Volume (D:) > test'. It lists several files, including 'alpine-standard-3.21.3-x86_64.iso' (250,880 bytes, Disc Image File), 'app.py' (1 byte, Python Source File), 'Dockerfile' (1 byte, File), 'requirements.txt' (1 byte, Text Document), 'temp.qcow2' (193 bytes, QCOW2 File), 'test1.qcow2' (193 bytes, QCOW2 File), 'test2.qcow2' (0 bytes, QCOW2 File), and 'test23.qcow2' (0 bytes, QCOW2 File). The 'File name' field is empty, and 'Open' and 'Cancel' buttons are at the bottom right.</p>

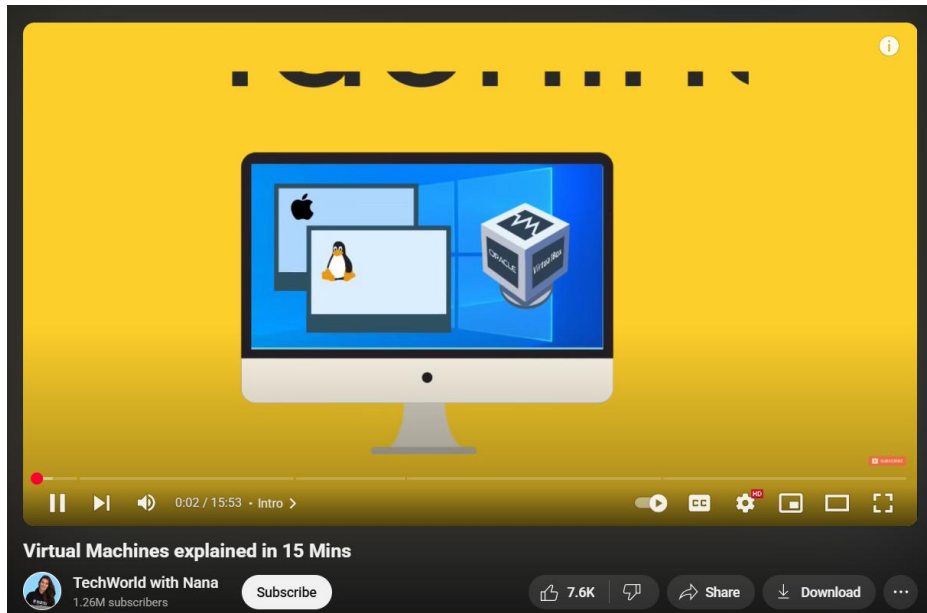
<i>Test Case Name</i>	TVM13: Verify ISO Path is Optional
<i>Test Scenario</i>	Check behavior when ISO path is left blank.
<i>Prerequisites</i>	ISO optional
<i>Test Input</i>	Leave ISO empty, fill rest
<i>Execution Steps</i>	<ol style="list-style-type: none">1. Fill required fields.2. Leave ISO blank.3. Click Create VM.
<i>Expected Behavior</i>	VM creation succeeds without ISO
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present
<i>Status</i>	Pass
<i>Real Life Testing</i>	


<i>Test Case Name</i>	TVM14: Validate File Format of ISO
<i>Test Scenario</i>	Only .iso files are allowed.
<i>Prerequisites</i>	File system has non-ISO files.
<i>Test Input</i>	Select non-ISO file.
<i>Execution Steps</i>	<ol style="list-style-type: none">1. Click Browse.2. Try selecting .txt or .exe file.
<i>Expected Behavior</i>	System restricts or warns user
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present
<i>Status</i>	Pass
<i>Real Life Testing</i>	

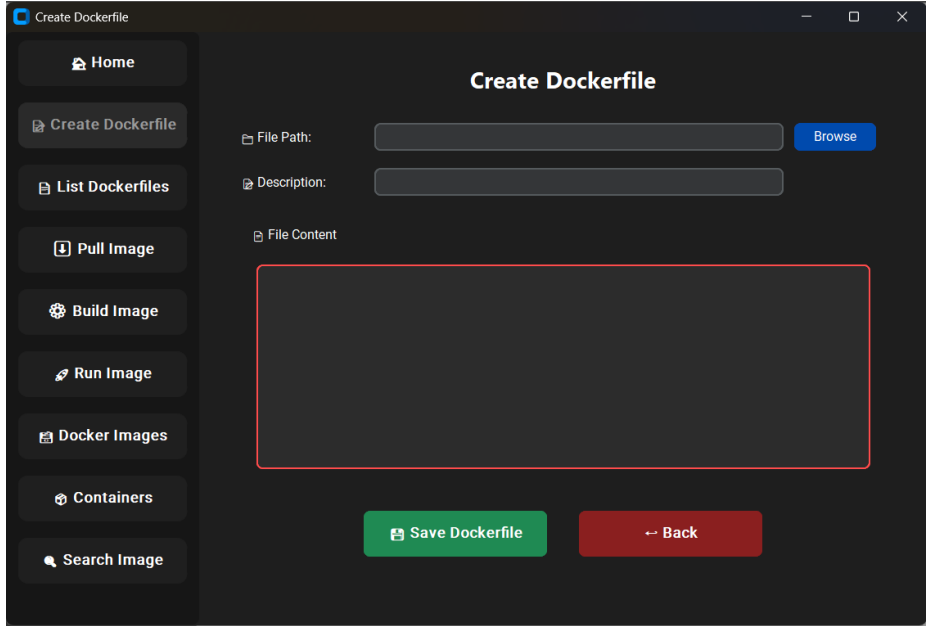
<i>Test Case Name</i>	THP1: Verify Page Load
<i>Test Scenario</i>	Ensure the "HOME" page loads successfully.
<i>Prerequisites</i>	-
<i>Test Input</i>	-
<i>Execution Steps</i>	1. Launch the app.
<i>Expected Behavior</i>	The home page loads without errors
<i>Assumptions</i>	UI and navigation work
<i>Actual Results</i>	UI and navigation work
<i>Status</i>	Pass
<i>Real Life Testing</i>	

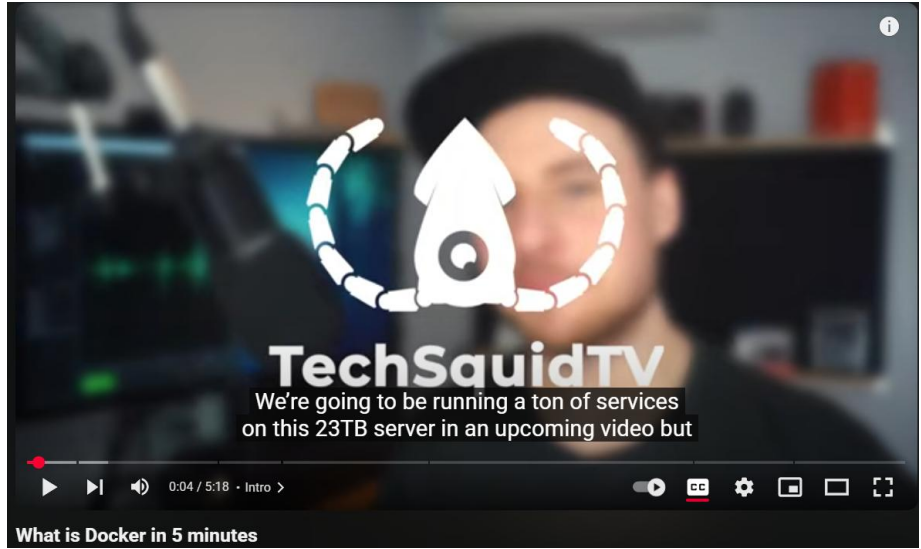
<i>Test Case Name</i>	THP2: Verify “VD” button
<i>Test Scenario</i>	Ensure the “Start VD” button loads the “virtual disk page” correctly.
<i>Prerequisites</i>	-
<i>Test Input</i>	Button input
<i>Execution Steps</i>	1. Press on VD
<i>Expected Behavior</i>	The “virtual disk” page loads without errors
<i>Assumptions</i>	Virtual disk page is loaded
<i>Actual Results</i>	Virtual disk page is loaded
<i>Status</i>	Pass
<i>Real Life Testing</i>	 A screenshot of a web application titled "Create Virtual Disk". The interface has a dark theme. At the top, there are navigation links for "Home" and "List", and a "Create VD" button. The main heading is "Create Virtual Disk". Below it, there are four input fields: "Disk Path:" with a "Browse" button, "File Name:", "Format:" (set to "qcow2"), and "Size (GB):" (set to "1"). At the bottom, there are two buttons: a green "Create Disk" button and a red "Back" button.

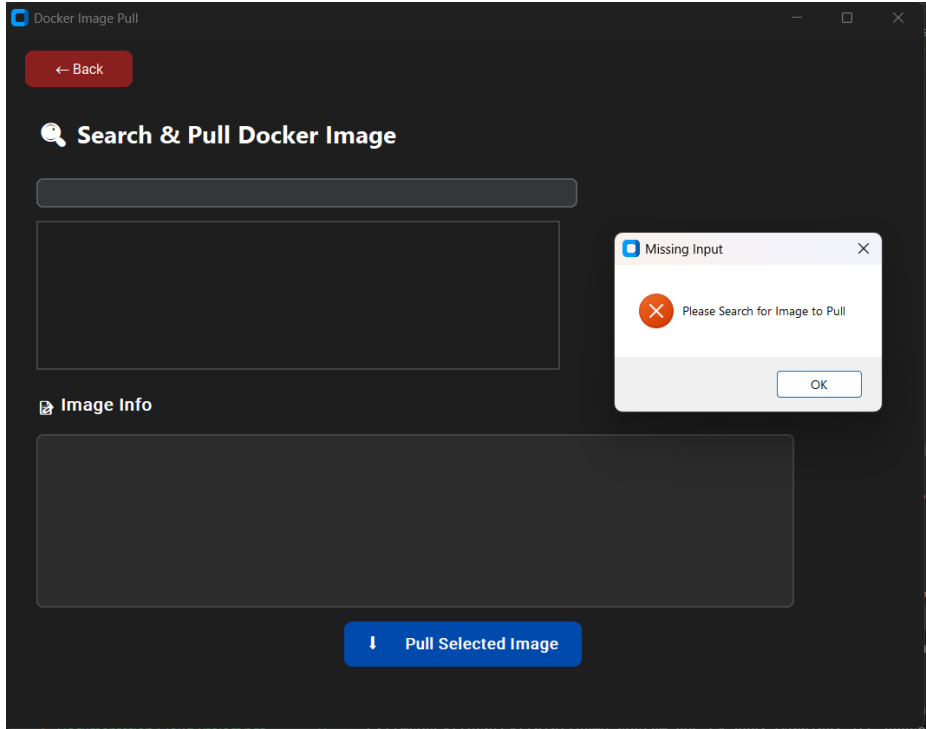
<i>Test Case Name</i>	THP3: Verify “VM” button
<i>Test Scenario</i>	Ensure the “VM” button loads the “virtual machine page” correctly.
<i>Prerequisites</i>	-
<i>Test Input</i>	Button input
<i>Execution Steps</i>	1. Press on VM
<i>Expected Behavior</i>	The “virtual machine” page loads without errors
<i>Assumptions</i>	Virtual machine page is loaded
<i>Actual Results</i>	Virtual machine page is loaded
<i>Status</i>	Pass
<i>Real Life Testing</i>	

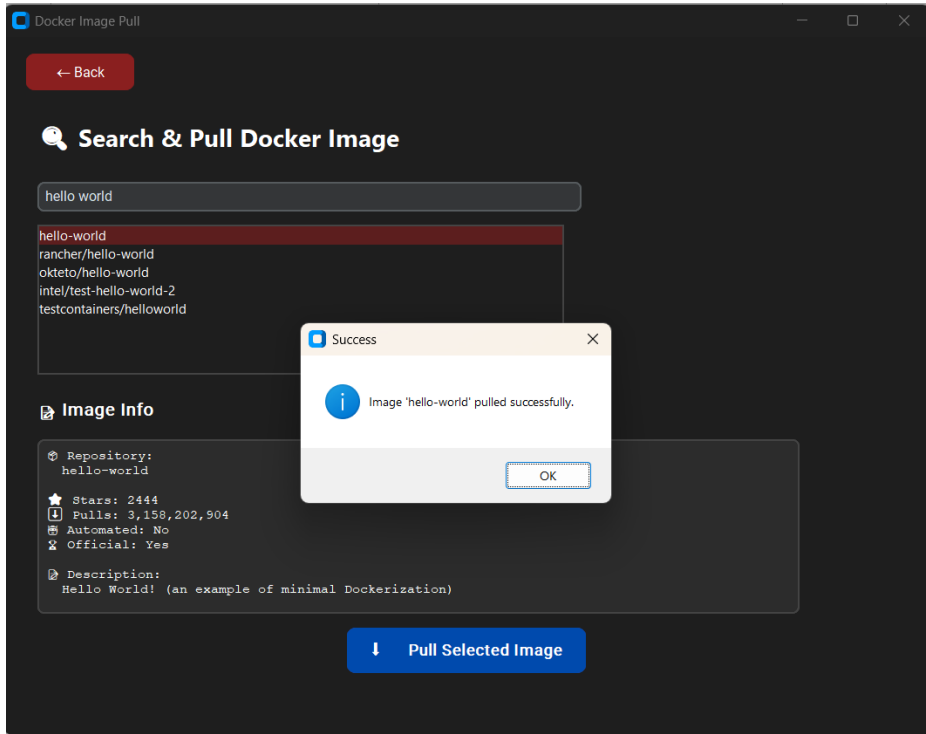
<i>Test Case Name</i>	THP4: Verify the “Watch” button for the VM
<i>Test Scenario</i>	Ensure the “watch” button loads the “YouTube page” correctly.
<i>Prerequisites</i>	-
<i>Test Input</i>	Button input
<i>Execution Steps</i>	1. Press on “watch”
<i>Expected Behavior</i>	The “YouTube” page loads without errors
<i>Assumptions</i>	YouTube page is loaded
<i>Actual Results</i>	YouTube page is loaded
<i>Status</i>	Pass
<i>Real Life Testing</i>	

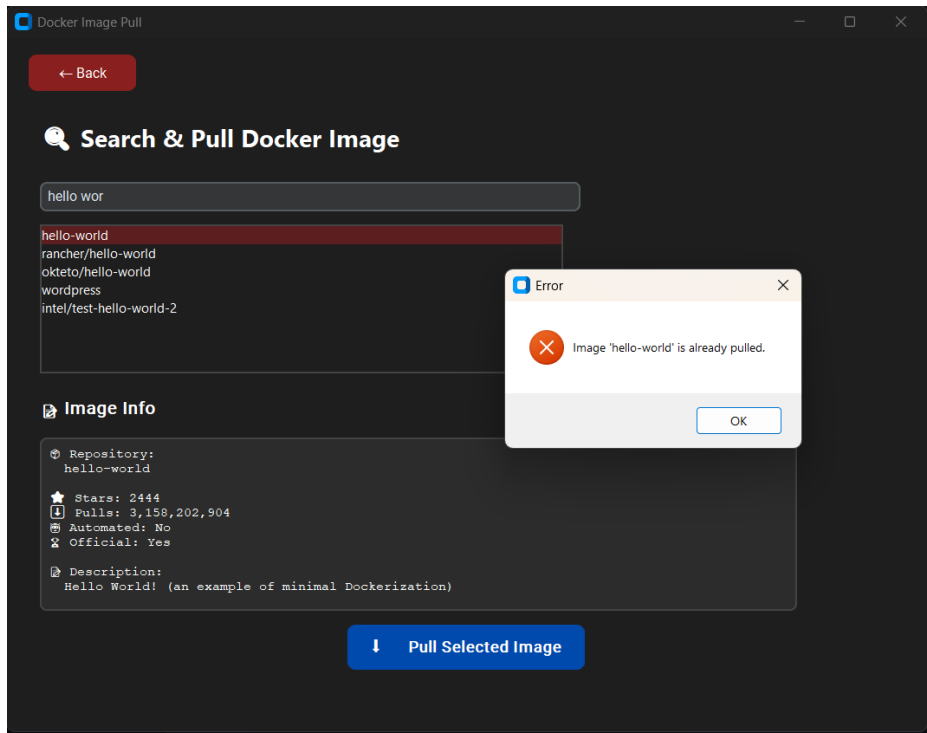
<i>Test Case Name</i>	THP5: Verify the “Watch” button for the VD
<i>Test Scenario</i>	Ensure the “watch” button loads the “YouTube page” correctly.
<i>Prerequisites</i>	-
<i>Test Input</i>	Button input
<i>Execution Steps</i>	2. Press on “watch”
<i>Expected Behavior</i>	The “YouTube” page loads without errors
<i>Assumptions</i>	YouTube page is loaded
<i>Actual Results</i>	YouTube page is loaded
<i>Status</i>	Pass
<i>Real Life Testing</i>	

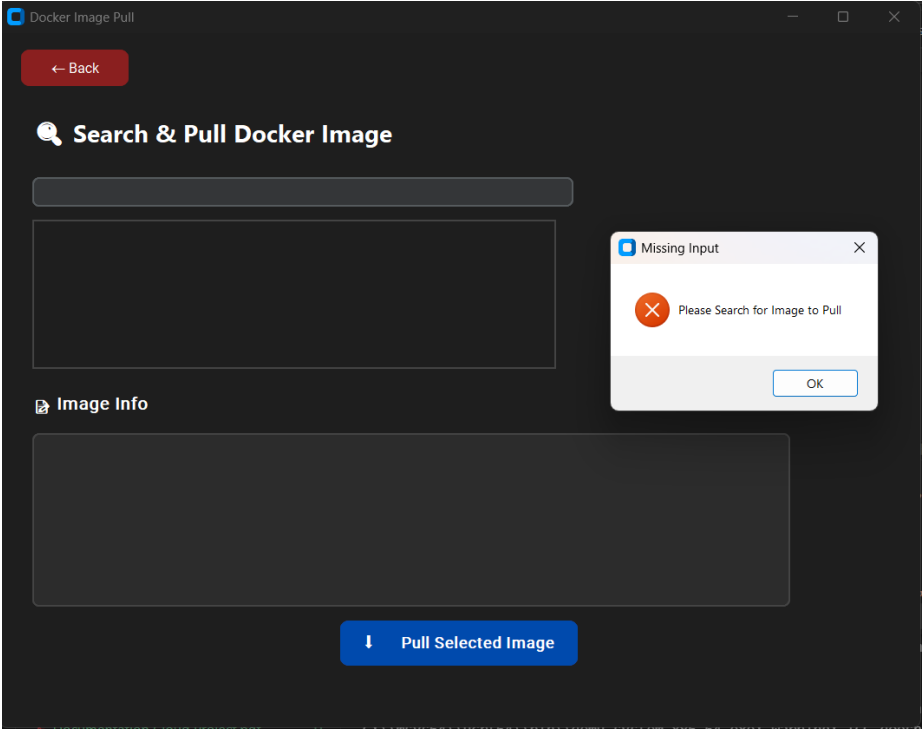
<i>Test Case Name</i>	THP6: Verify “DK” button
<i>Test Scenario</i>	Ensure the “DK” button loads the “docker page” correctly.
<i>Prerequisites</i>	-
<i>Test Input</i>	Button input
<i>Execution Steps</i>	1. Press on start DK
<i>Expected Behavior</i>	The “docker” page loads without errors
<i>Assumptions</i>	Docker page is loaded
<i>Actual Results</i>	Docker machine page is loaded
<i>Status</i>	Pass
<i>Real Life Testing</i>	

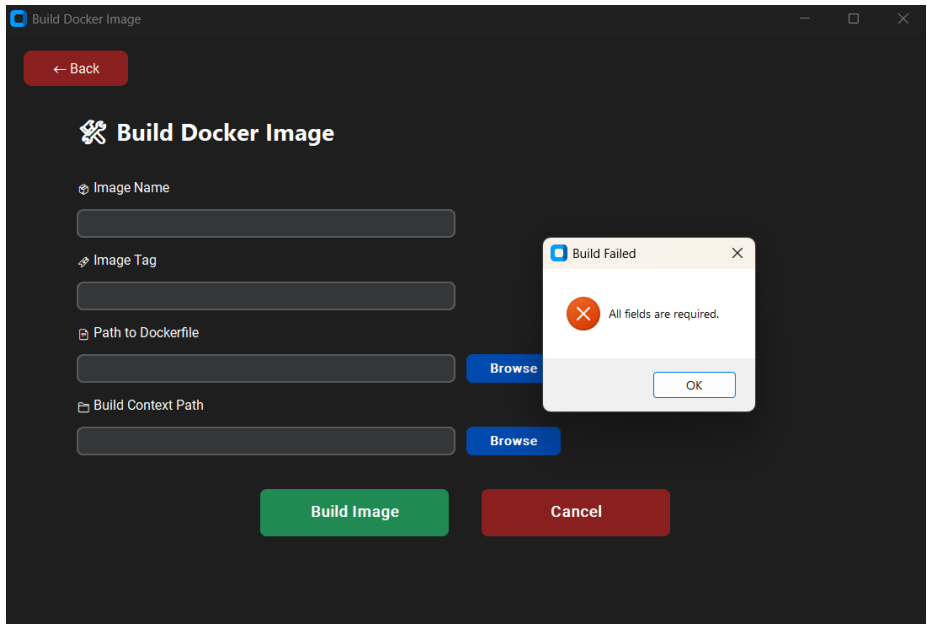
<i>Test Case Name</i>	THP7: Verify the “Watch” button for the DK
<i>Test Scenario</i>	Ensure the “watch” button loads the “YouTube page” correctly.
<i>Prerequisites</i>	-
<i>Test Input</i>	Button input
<i>Execution Steps</i>	1. Press on “watch”
<i>Expected Behavior</i>	The “YouTube” page loads without errors
<i>Assumptions</i>	YouTube page is loaded
<i>Actual Results</i>	YouTube page is loaded
<i>Status</i>	Pass
<i>Real Life Testing</i>	

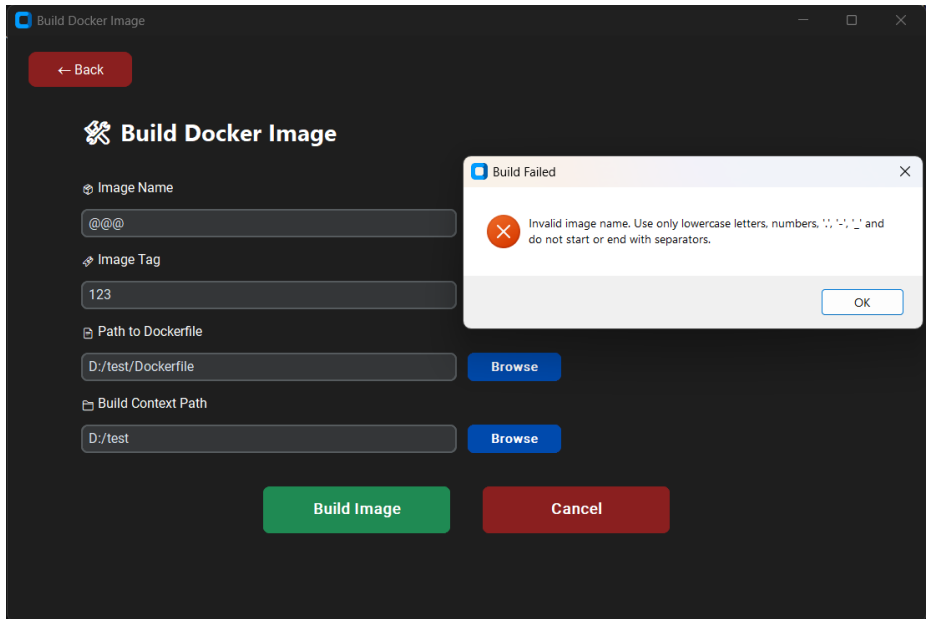
<i>Test Case Name</i>	DK1: Submit with Empty Fields
<i>Test Scenario</i>	Submit button should validate empty input.
<i>Prerequisites</i>	The form is visible
<i>Test Input</i>	Leave all fields blank
<i>Execution Steps</i>	1. Click “Pull Selected Image” without filling anything
<i>Expected Behavior</i>	Error shown or image not pulled
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present
<i>Status</i>	Pass
<i>Real Life Testing</i>	 <p>The screenshot shows the Docker Image Pull application window. At the top, there is a title bar 'Docker Image Pull' and a '← Back' button. Below the title bar, the main heading is 'Search & Pull Docker Image'. There is a search input field and a large rectangular area for image details. Below this, there is an 'Image Info' section with another large rectangular area. At the bottom right, there is a blue button labeled 'Pull Selected Image'. A modal dialog box titled 'Missing Input' is open on the right side of the window. It contains a red 'X' icon and the text 'Please Search for Image to Pull'. There is an 'OK' button at the bottom of the dialog.</p>

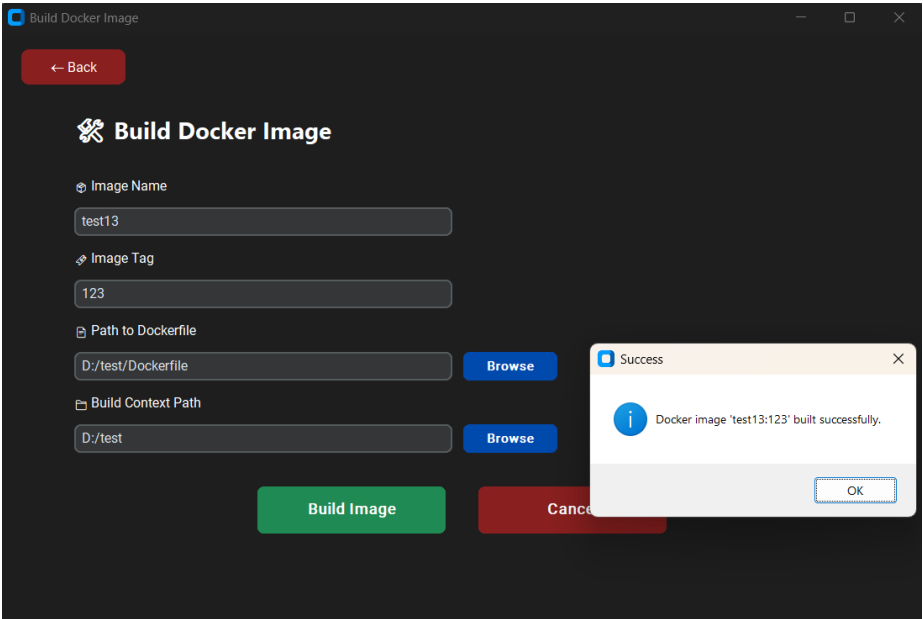
<i>Test Case Name</i>	DK2: Submit with Valid Image Name
<i>Test Scenario</i>	Should pull the image successfully.
<i>Prerequisites</i>	Docker Daemon is running.
<i>Test Input</i>	Valid image name
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Enter a valid image name in the search field. 2. Click "Pull Selected Image."
<i>Expected Behavior</i>	Image is pulled successfully.
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present
<i>Status</i>	Pass
<i>Real Life Testing</i>	

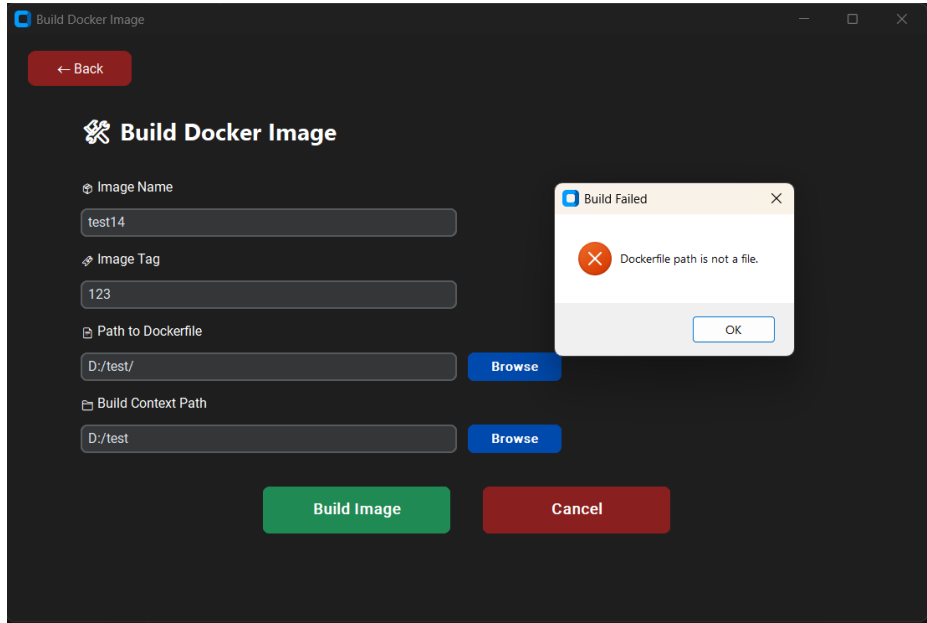
<i>Test Case Name</i>	DK3: Pull Image Multiple Times
<i>Test Scenario</i>	It should handle repeated pulls gracefully.
<i>Prerequisites</i>	Docker Daemon is running.
<i>Test Input</i>	Valid image name
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Enter a valid image name in the search field. 2. Click "Pull Selected Image." 3. Click "Pull Selected Image" again.
<i>Expected Behavior</i>	The system should either pull the image again or show a message indicating it's already present.
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present
<i>Status</i>	Pass
<i>Real Life Testing</i>	 <p>The screenshot shows the Docker Desktop 'Image Pull' window. At the top, there's a search bar with 'hello wor' entered. Below it, a list of search results is shown, with 'hello-world' selected. To the right of the search results, an 'Error' dialog box is displayed with the message 'Image 'hello-world' is already pulled.' and an 'OK' button. Below the search results, there's an 'Image Info' section for 'hello-world', showing details like Stars (2444), Pulls (3,158,202,904), Automated (No), Official (Yes), and a description: 'Hello World! (an example of minimal Dockerization)'. At the bottom right, there's a 'Pull Selected Image' button.</p>

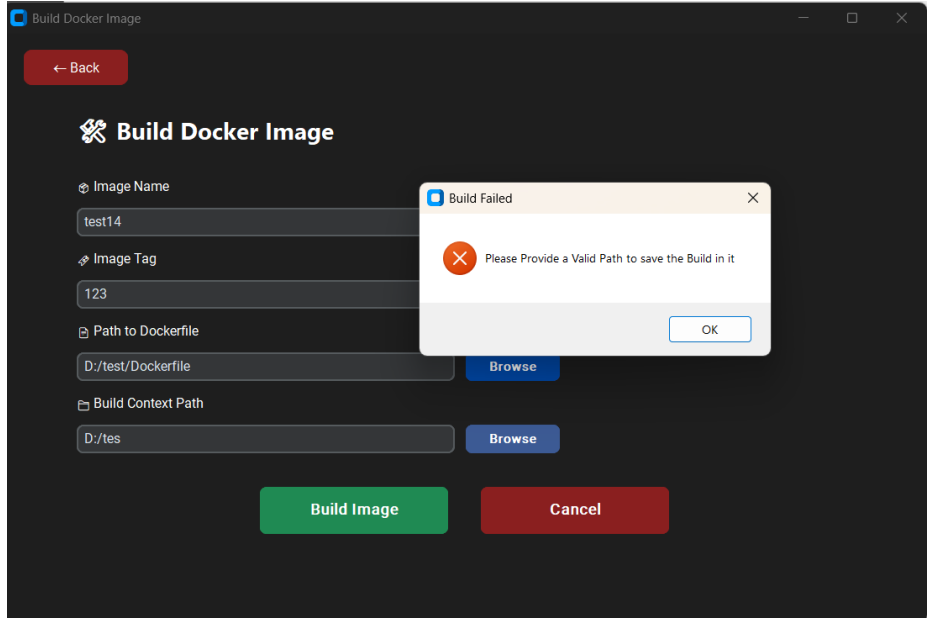
<i>Test Case Name</i>	DK4: Submit with Empty Fields
<i>Test Scenario</i>	Submit button should validate empty input.
<i>Prerequisites</i>	The form is visible
<i>Test Input</i>	Leave all fields blank
<i>Execution Steps</i>	2. Click “Pull Selected Image” without filling anything
<i>Expected Behavior</i>	Error shown or image not pulled
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present
<i>Status</i>	Pass
<i>Real Life Testing</i>	 A screenshot of the Docker Desktop 'Image Pull' window. The window has a dark theme. At the top left is a blue square icon and the text 'Docker Image Pull'. Below it is a red button with a left arrow and the text 'Back'. The main heading is 'Search & Pull Docker Image' with a magnifying glass icon. There is a search input field and a large empty rectangular box below it. To the right of this box is a white error dialog box titled 'Missing Input' with a red 'X' icon and the text 'Please Search for Image to Pull'. Below the error box is an 'OK' button. At the bottom of the main window is a blue button with a right arrow and the text 'Pull Selected Image'. Below the main heading is a section titled 'Image Info' with a document icon and a large empty rectangular box.

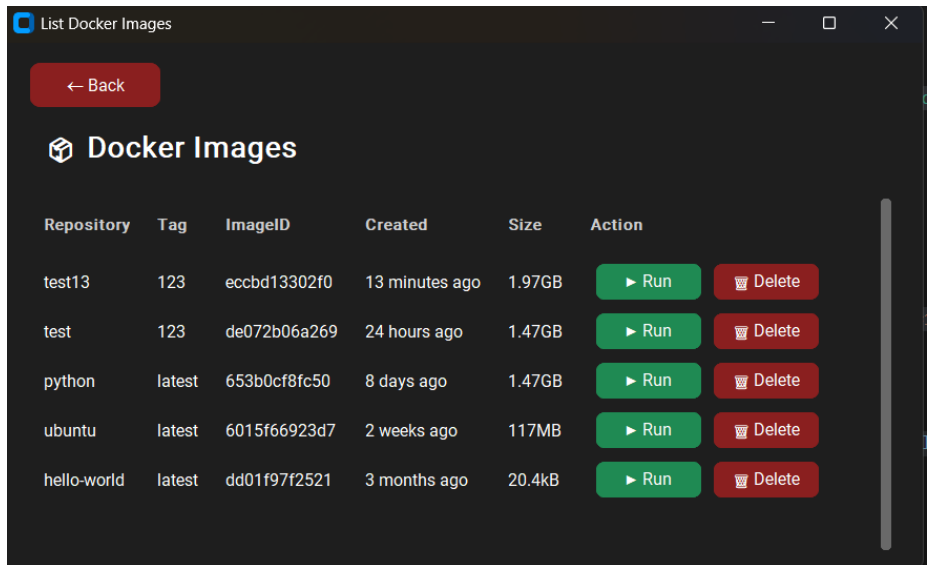
<i>Test Case Name</i>	DKI1: Submit with Empty Fields
<i>Test Scenario</i>	Submit button should validate empty input.
<i>Prerequisites</i>	-
<i>Test Input</i>	Leave all fields blank
<i>Execution Steps</i>	<ol style="list-style-type: none">1. Leave all fields empty.2. Click "Build Image."
<i>Expected Behavior</i>	An error message is displayed indicating that all fields are required.
<i>Assumptions</i>	Validation is present
<i>Actual Results</i>	Validation is present, error message displayed
<i>Status</i>	Pass
<i>Real Life Testing</i>	

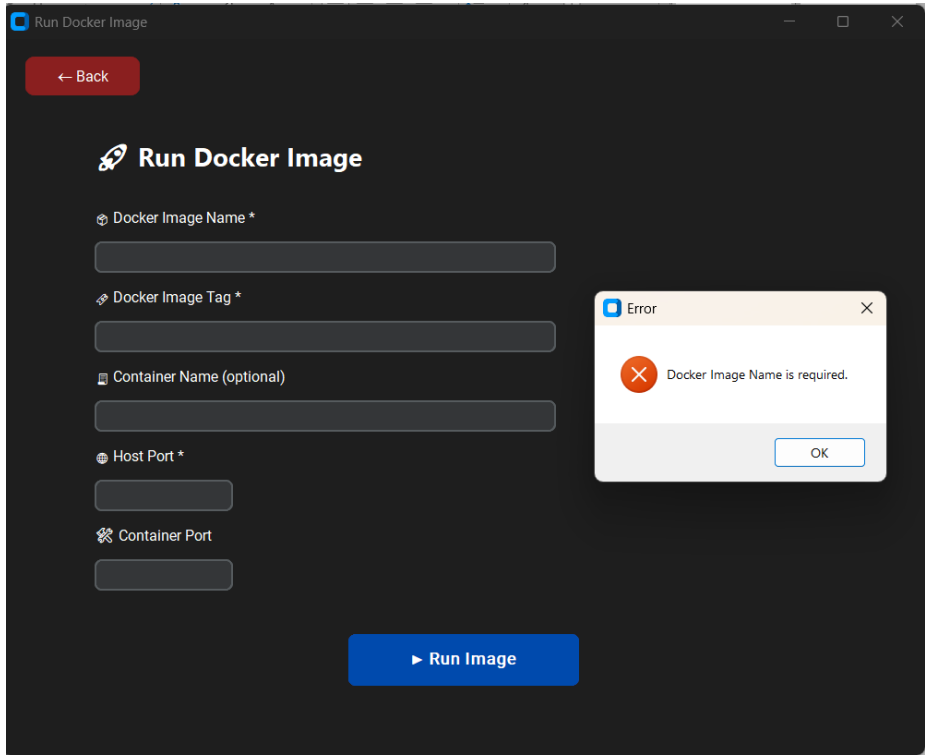
Test Case Name	DKI2: Submit with Invalid Image Name
Test Scenario	Should show an error for invalid image name.
Prerequisites	-
Test Input	Invalid image name (e.g., special characters)
Execution Steps	<ol style="list-style-type: none">1. Enter an invalid image name in the search field.2. Click "Pull Selected Image."
Expected Behavior	An error message is displayed indicating the image name is invalid.
Assumptions	Validation is present
Actual Results	Validation is present
Status	Pass
Real Life Testing	 The screenshot shows a 'Build Docker Image' window with a dark theme. It contains input fields for 'Image Name' (containing '@@@'), 'Image Tag' (containing '123'), 'Path to Dockerfile' (containing 'D:/test/Dockerfile'), and 'Build Context Path' (containing 'D:/test'). There are 'Browse' buttons next to the Dockerfile and context path fields. At the bottom are 'Build Image' and 'Cancel' buttons. An error dialog box titled 'Build Failed' is overlaid on the right, displaying a red 'X' icon and the message: 'Invalid image name. Use only lowercase letters, numbers, '.', ':', '-', '_' and do not start or end with separators.' with an 'OK' button.

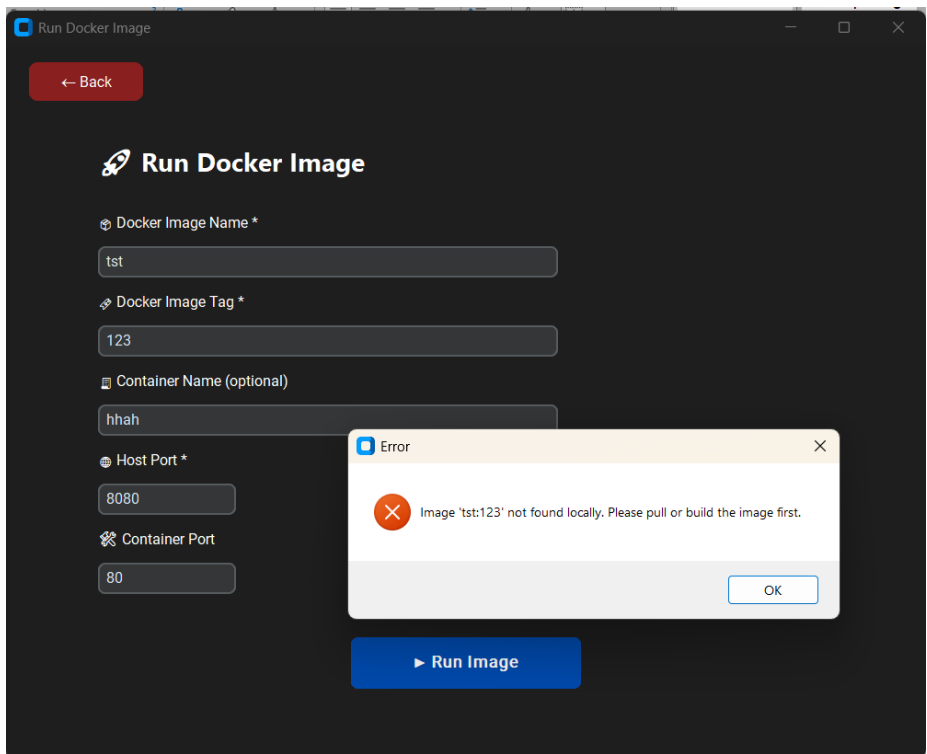
Test Case Name	DKI3: Submit with Valid Data
Test Scenario	Should build the image successfully.
Prerequisites	Docker Daemon is running.
Test Input	Valid image name, tag, and paths
Execution Steps	<ol style="list-style-type: none">1. Enter valid data in all fields.2. Click "Build Image."
Expected Behavior	Image is built successfully.
Assumptions	Validation is present
Actual Results	Image is built and displayed
Status	Pass
Real Life Testing	 A screenshot of the Docker Desktop 'Build Docker Image' window. The window has a dark theme and contains several input fields: 'Image Name' with 'test13', 'Image Tag' with '123', 'Path to Dockerfile' with 'D:/test/Dockerfile', and 'Build Context Path' with 'D:/test'. There are 'Browse' buttons next to the Dockerfile and context path fields. At the bottom, there are 'Build Image' (green) and 'Cancel' (red) buttons. A 'Success' dialog box is overlaid on the right, displaying a blue information icon and the message 'Docker image 'test13:123' built successfully.' with an 'OK' button.

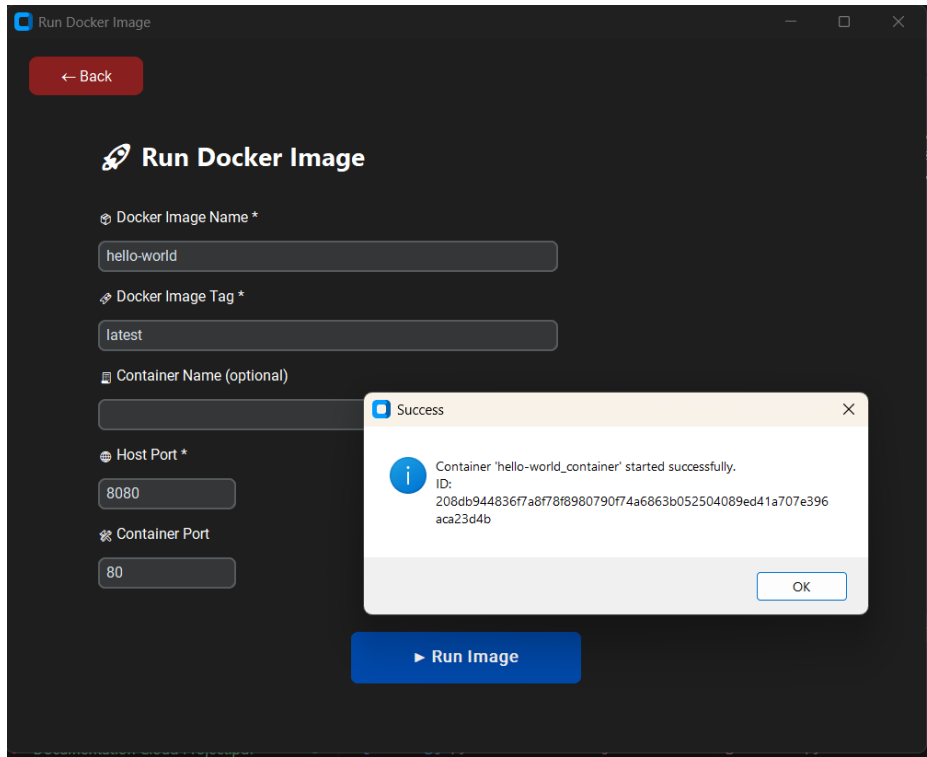
<i>Test Case Name</i>	DKI4: Submit with Invalid Dockerfile Path
<i>Test Scenario</i>	Should show an error for invalid Dockerfile path.
<i>Prerequisites</i>	-
<i>Test Input</i>	Valid image name, tag, and paths
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Enter valid data in "Image Name" and "Image Tag." 2. Provide an invalid path for the "Path to Dockerfile." 3. Click "Build Image."
<i>Expected Behavior</i>	An error message indicating the Dockerfile path is invalid.
<i>Assumptions</i>	Backend checks file existence.
<i>Actual Results</i>	Error messages are shown.
<i>Status</i>	Pass
<i>Real Life Testing</i>	

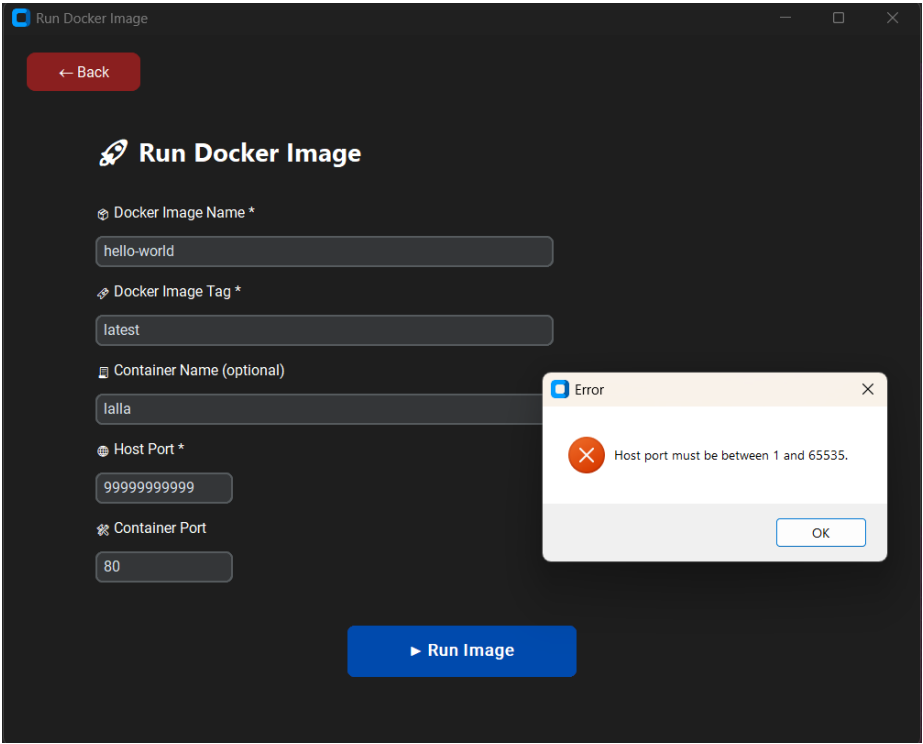
<i>Test Case Name</i>	DKI5: Validate Build Context Path
<i>Test Scenario</i>	Should show an error for invalid build context path.
<i>Prerequisites</i>	-
<i>Test Input</i>	Valid image name, tag, and invalid build context path
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Enter valid data in "Image Name" and "Image Tag." 2. Provide an invalid path for the "Build Context Path." 3. Click "Build Image."
<i>Expected Behavior</i>	An error message indicating the build context path is invalid
<i>Assumptions</i>	Backend checks file existence.
<i>Actual Results</i>	Error messages are shown.
<i>Status</i>	Pass
<i>Real Life Testing</i>	

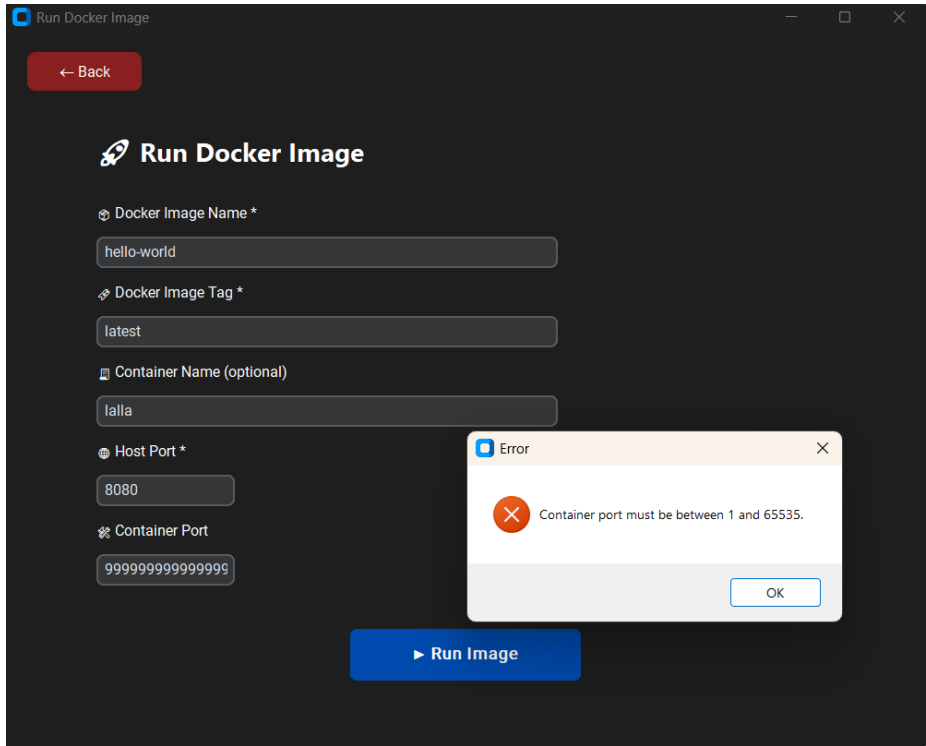
<i>Test Case Name</i>	DKI6: List Image After Build
<i>Test Scenario</i>	Should allow the newly built image.
<i>Prerequisites</i>	Docker Daemon is running.
<i>Test Input</i>	Valid image name of a recently built image
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Build an image successfully. 2. Press Back. 3. Click on Docker Images from the Side.
<i>Expected Behavior</i>	The image is shown in the list.
<i>Assumptions</i>	Backend maintains image availability.
<i>Actual Results</i>	Image is created successfully.
<i>Status</i>	Pass
<i>Real Life Testing</i>	

Test Case Name	RDKI1: Submit with Empty Fields
Test Scenario	Submit button should validate empty input.
Prerequisites	-
Test Input	Leave all fields blank
Execution Steps	<ol style="list-style-type: none">1. Leave all fields empty.2. Click "Run Image."
Expected Behavior	An error message is displayed indicating that all fields are required.
Assumptions	Validation is present
Actual Results	Validation is present, error message displayed
Status	Pass
Real Life Testing	 The screenshot shows a 'Run Docker Image' dialog box with a dark background. It has a 'Back' button in the top left and a 'Run Image' button in the bottom right. The form contains five input fields: 'Docker Image Name *', 'Docker Image Tag *', 'Container Name (optional)', 'Host Port *', and 'Container Port'. All fields are empty. An error message box is overlaid on the right, stating 'Error: Docker Image Name is required.' with an 'OK' button.

Test Case Name	RDKI2: Submit with Invalid Docker Image Name
Test Scenario	Should show an error for invalid Docker Image Name.
Prerequisites	-
Test Input	Invalid Docker image name (e.g., special characters)
Execution Steps	<ol style="list-style-type: none">1. Enter an invalid Docker image name in the "Docker Image Name" field.2. Fill other fields with valid data.3. Click "Run Image."
Expected Behavior	An error message indicating the Docker Image name is invalid.
Assumptions	Validation is present.
Actual Results	Error messages are shown.
Status	Pass
Real Life Testing	 A screenshot of a web application titled "Run Docker Image". The interface has a dark theme. At the top left is a "Back" button. Below it is a rocket icon and the title "Run Docker Image". There are five input fields: "Docker Image Name *" with the value "tst", "Docker Image Tag *" with the value "123", "Container Name (optional)" with the value "hhah", "Host Port *" with the value "8080", and "Container Port" with the value "80". At the bottom right is a "Run Image" button. An error dialog box is open in the foreground, titled "Error", with a red 'X' icon and the message "Image 'tst:123' not found locally. Please pull or build the image first." and an "OK" button.

<i>Test Case Name</i>	RDKI3: Submit with Valid Data
<i>Test Scenario</i>	Should run the image successfully.
<i>Prerequisites</i>	Docker Daemon is running.
<i>Test Input</i>	Valid Docker image name, tag, host port, and optional container name
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Enter valid data in all fields. 2. Click "Run Image."
<i>Expected Behavior</i>	The image is run successfully.
<i>Assumptions</i>	Backend supports the operation.
<i>Actual Results</i>	Image is running as expected
<i>Status</i>	Pass
<i>Real Life Testing</i>	 <p>The screenshot shows a web application titled "Run Docker Image" with a dark theme. It features a "Back" button in the top left and a "Run Image" button at the bottom right. The form contains five input fields: "Docker Image Name *" (filled with "hello-world"), "Docker Image Tag *" (filled with "latest"), "Container Name (optional)" (empty), "Host Port *" (filled with "8080"), and "Container Port" (filled with "80"). A modal dialog box titled "Success" is displayed in the center, containing an information icon, a message stating "Container 'hello-world_container' started successfully.", the container ID "208db944836f7a8f78f8980790f74a6863b052504089ed41a707e396aca23d4b", and an "OK" button.</p>

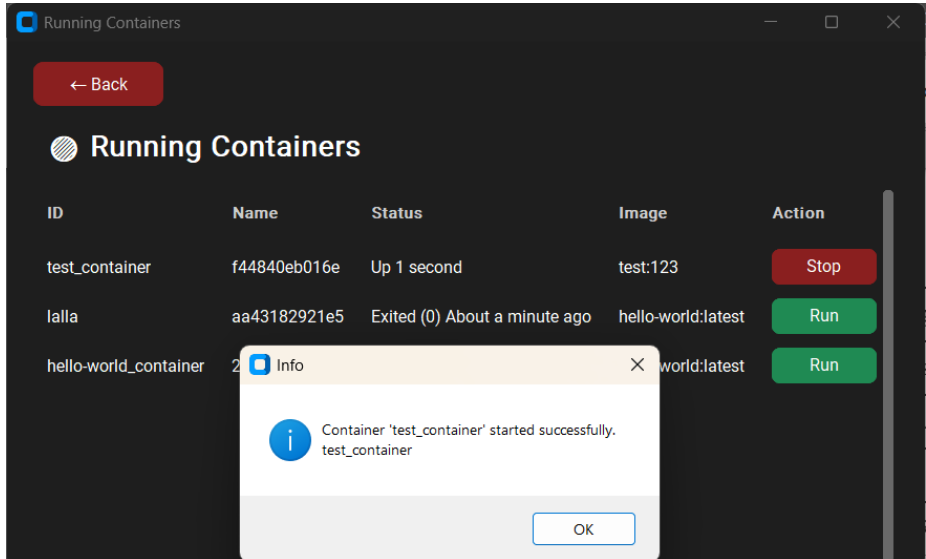
Test Case Name	RDKI4: Submit with Invalid Host Port
Test Scenario	Should show an error for invalid host port.
Prerequisites	-
Test Input	Valid Docker image name and tag, invalid host port
Execution Steps	<ol style="list-style-type: none">1. Enter valid data in all fields.2. Click "Run Image."
Expected Behavior	An error message indicating the host port is invalid.
Assumptions	Backend checks port validity.
Actual Results	Error messages are shown.
Status	Pass
Real Life Testing	

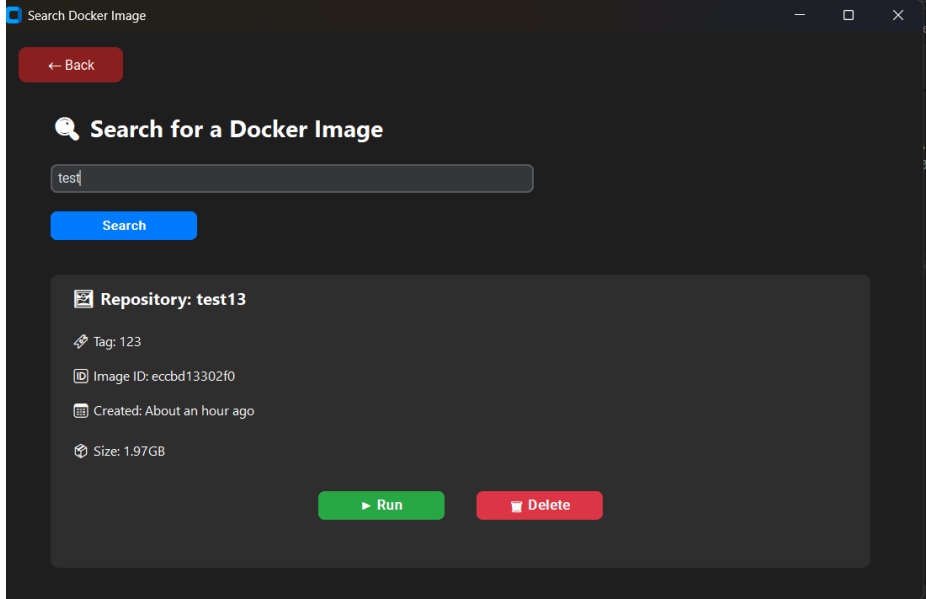
<i>Test Case Name</i>	RDKI5: Submit with Invalid Container Port
<i>Test Scenario</i>	Should show an error for invalid container port.
<i>Prerequisites</i>	-
<i>Test Input</i>	Valid Docker image name and tag, invalid container port
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Enter valid Docker image name and tag. 2. Provide an invalid container port. 3. Click "Run Image."
<i>Expected Behavior</i>	An error message indicating the container port is invalid.
<i>Assumptions</i>	Backend checks port validity.
<i>Actual Results</i>	Error messages are shown.
<i>Status</i>	Pass
<i>Real Life Testing</i>	 <p>The screenshot shows a web application titled "Run Docker Image". It has a "Back" button in the top left. The form contains the following fields: <ul style="list-style-type: none"> Docker Image Name *: Input field with "hello-world" Docker Image Tag *: Input field with "latest" Container Name (optional): Input field with "lalla" Host Port *: Input field with "8080" Container Port: Input field with "9999999999999999" A blue "Run Image" button is at the bottom right. An "Error" dialog box is open in the foreground, displaying a red 'X' icon and the message: "Container port must be between 1 and 65535." with an "OK" button. </p>

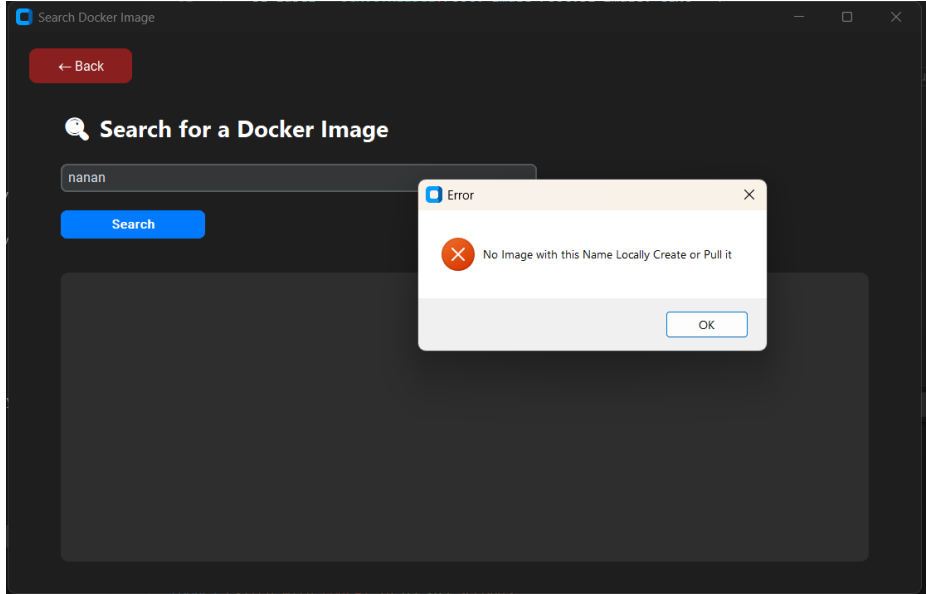
Test Case Name	RCDK1: Display Running Containers															
Test Scenario	You should display a list of containers.															
Prerequisites	At least one container.															
Test Input	-															
Execution Steps	1. Navigate to the "Running Containers" page.															
Expected Behavior	A list of running containers is displayed with their IDs, statuses, images, and action options.															
Assumptions	Backend provides the correct status of running containers.															
Actual Results	Running containers are displayed correctly.															
Status	Pass															
Real Life Testing	<div><div>Running Containers</div><div><div>← Back</div><div><div>Running Containers</div><table><thead><tr><th>ID</th><th>Name</th><th>Status</th><th>Image</th><th>Action</th></tr></thead><tbody><tr><td>lalla</td><td>aa43182921e5</td><td>Exited (0) About a minute ago</td><td>hello-world:latest</td><td>Run</td></tr><tr><td>hello-world_container</td><td>208db944836f</td><td>Exited (0) 1 second ago</td><td>hello-world:latest</td><td>Run</td></tr></tbody></table></div></div></div>	ID	Name	Status	Image	Action	lalla	aa43182921e5	Exited (0) About a minute ago	hello-world:latest	Run	hello-world_container	208db944836f	Exited (0) 1 second ago	hello-world:latest	Run
ID	Name	Status	Image	Action												
lalla	aa43182921e5	Exited (0) About a minute ago	hello-world:latest	Run												
hello-world_container	208db944836f	Exited (0) 1 second ago	hello-world:latest	Run												

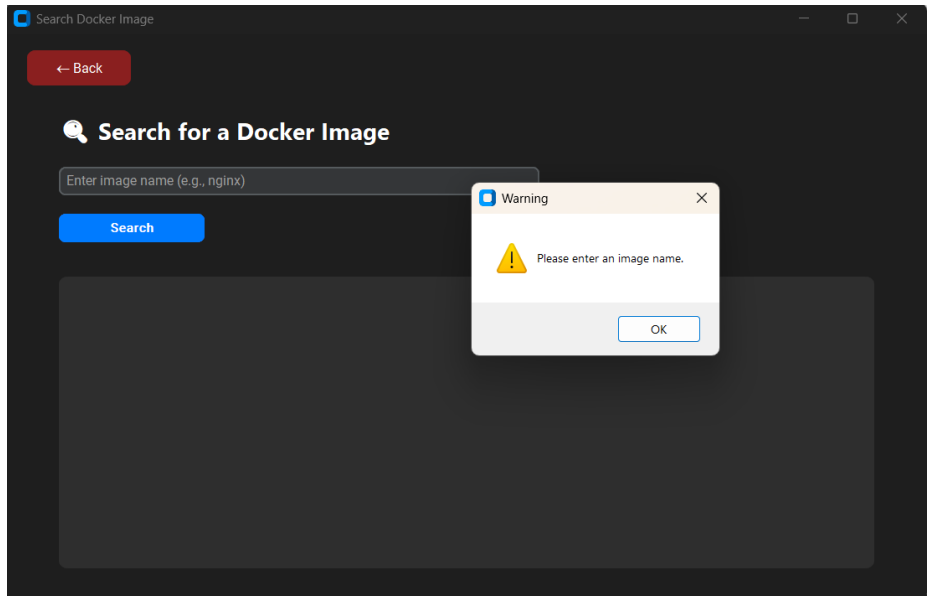
Test Case Name	RCDK2: Run Container from List															
Test Scenario	You should allow running a stopped container from the list.															
Prerequisites	At least one stopped container is available.															
Test Input	-															
Execution Steps	<div>1. Navigate to the "Running Containers" page.</div> <div>2. Click "Run" next to a stopped container.</div>															
Expected Behavior	The selected container is started successfully.															
Assumptions	Backend supports starting containers.															
Actual Results	Container is running as expected.															
Status	Pass															
Real Life Testing	<div><div>Running Containers</div><div><div>← Back</div><div><div>Running Containers</div><table><thead><tr><th>ID</th><th>Name</th><th>Status</th><th>Image</th><th>Action</th></tr></thead><tbody><tr><td>lalla</td><td>aa43182921e5</td><td>Exited (0) 1 second ago</td><td>hello-world:latest</td><td>Run</td></tr><tr><td>hello-world_container</td><td>208db944836f</td><td>Exited (0) About a minute ago</td><td>hello-world:latest</td><td>Run</td></tr></tbody></table></div></div><div><div>Info</div><div><div>i</div><div>Container 'lalla' started successfully. lalla</div></div><div>OK</div></div></div>	ID	Name	Status	Image	Action	lalla	aa43182921e5	Exited (0) 1 second ago	hello-world:latest	Run	hello-world_container	208db944836f	Exited (0) About a minute ago	hello-world:latest	Run
ID	Name	Status	Image	Action												
lalla	aa43182921e5	Exited (0) 1 second ago	hello-world:latest	Run												
hello-world_container	208db944836f	Exited (0) About a minute ago	hello-world:latest	Run												

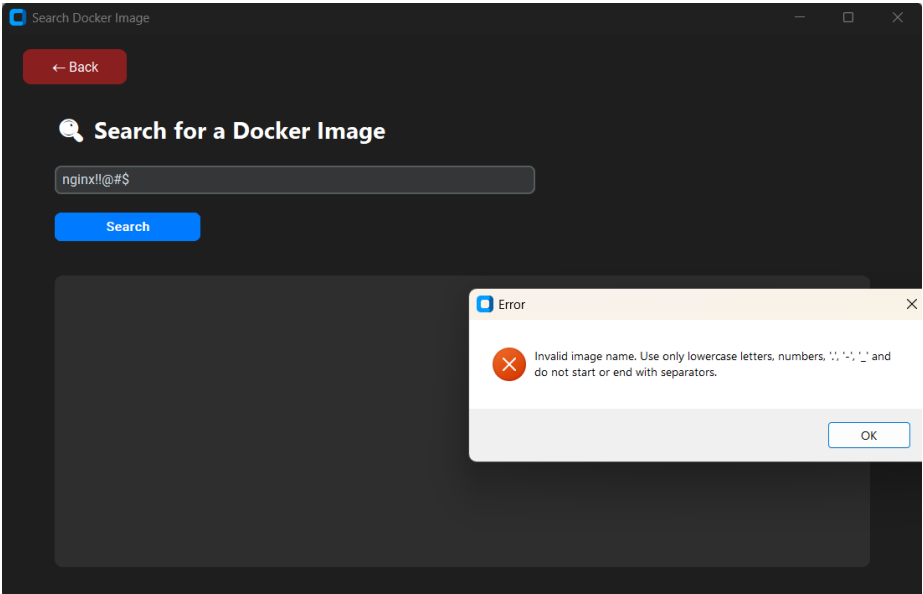
Test Case Name	RCDK3: Check Action Buttons Functionality															
Test Scenario	Should ensure action buttons work as intended. (in this case: RUN)															
Prerequisites	At least one stopped container is available.															
Test Input	-															
Execution Steps	<div>1. Navigate to the "Running Containers" page.</div> <div>2. Click "Run" next to a stopped container.</div> <div>3. Verify the action completes successfully.</div>															
Expected Behavior	The container starts without errors.															
Assumptions	Backend processes requests correctly.															
Actual Results	Container runs successfully.															
Status	Pass															
Real Life Testing	<div><div>Running Containers</div><div><div>← Back</div><div><div>Running Containers</div><table><tr><th>ID</th><th>Name</th><th>Status</th><th>Image</th><th>Action</th></tr><tr><td>lalla</td><td>aa43182921e5</td><td>Exited (0) 1 second ago</td><td>hello-world:latest</td><td>Run</td></tr><tr><td>hello-world_container</td><td>208db944836f</td><td>Exited (0) About a minute ago</td><td>hello-world:latest</td><td>Run</td></tr></table></div></div><div><div>Info</div><div><div>i</div><div>Container 'lalla' started successfully. lalla</div></div><div>OK</div></div></div>	ID	Name	Status	Image	Action	lalla	aa43182921e5	Exited (0) 1 second ago	hello-world:latest	Run	hello-world_container	208db944836f	Exited (0) About a minute ago	hello-world:latest	Run
ID	Name	Status	Image	Action												
lalla	aa43182921e5	Exited (0) 1 second ago	hello-world:latest	Run												
hello-world_container	208db944836f	Exited (0) About a minute ago	hello-world:latest	Run												

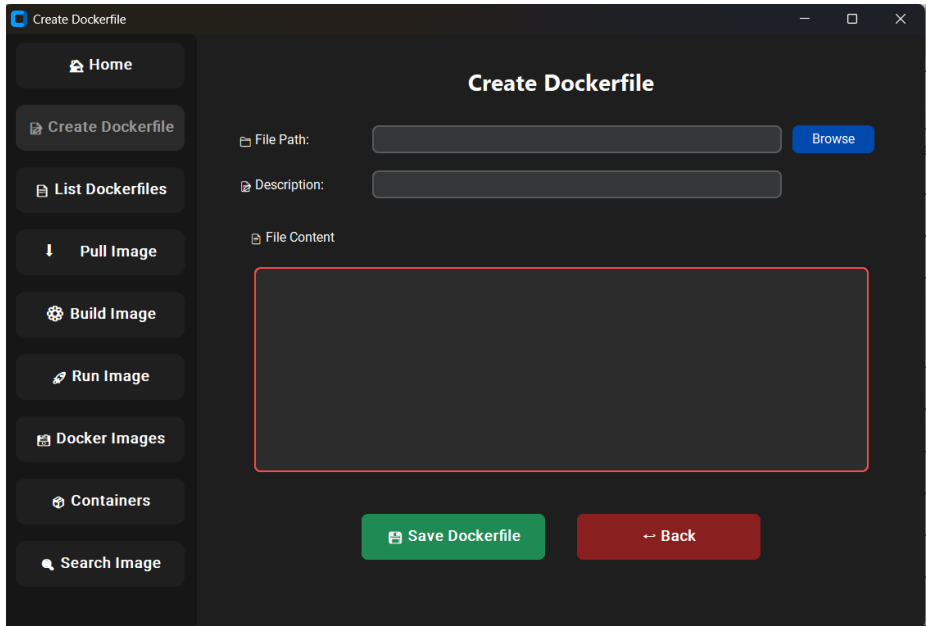
<i>Test Case Name</i>	RCDK4: Refresh Running Containers List
<i>Test Scenario</i>	You should refresh the list of running containers.
<i>Prerequisites</i>	At least one container is running or stopped.
<i>Test Input</i>	-
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Navigate to the "Running Containers" page. 2. Click a refresh button (if available).
<i>Expected Behavior</i>	The list of running containers updates to reflect current statuses.
<i>Assumptions</i>	Backend provides updated container statuses.
<i>Actual Results</i>	The list is updated correctly.
<i>Status</i>	Pass
<i>Real Life Testing</i>	

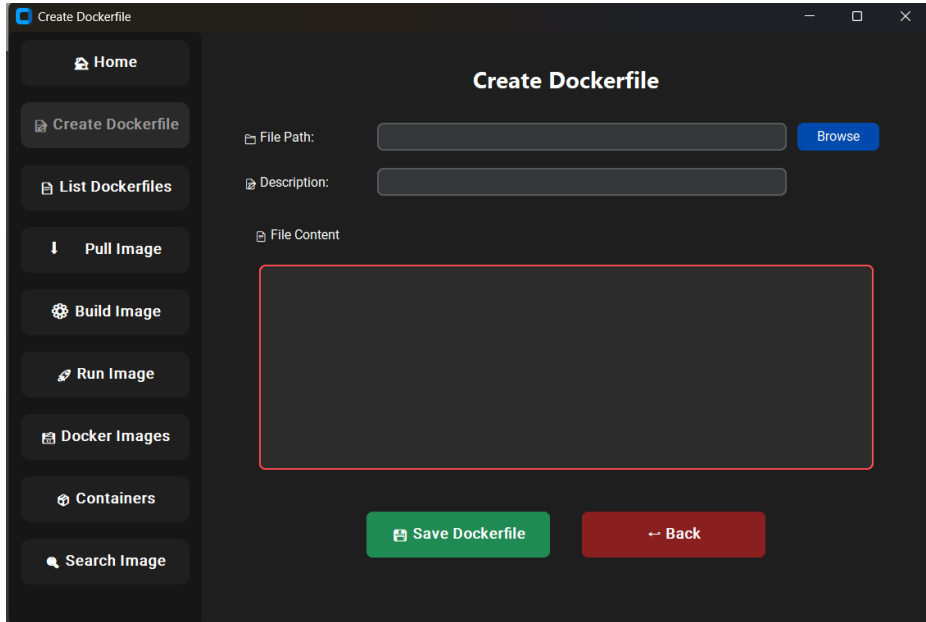
<i>Test Case Name</i>	SDKI1: Search for a Docker Image using valid name
<i>Test Scenario</i>	Users should be able to search for Docker images using valid name.
<i>Prerequisites</i>	Backend is properly integrated with Docker Hub
<i>Test Input</i>	-
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Enter a valid image name in the input field. 2. Click on the Search button.
<i>Expected Behavior</i>	A list of Docker images matching the input should be displayed with relevant details.
<i>Assumptions</i>	Backend handles both valid.
<i>Actual Results</i>	Results are displayed correctly.
<i>Status</i>	Pass
<i>Real Life Testing</i>	

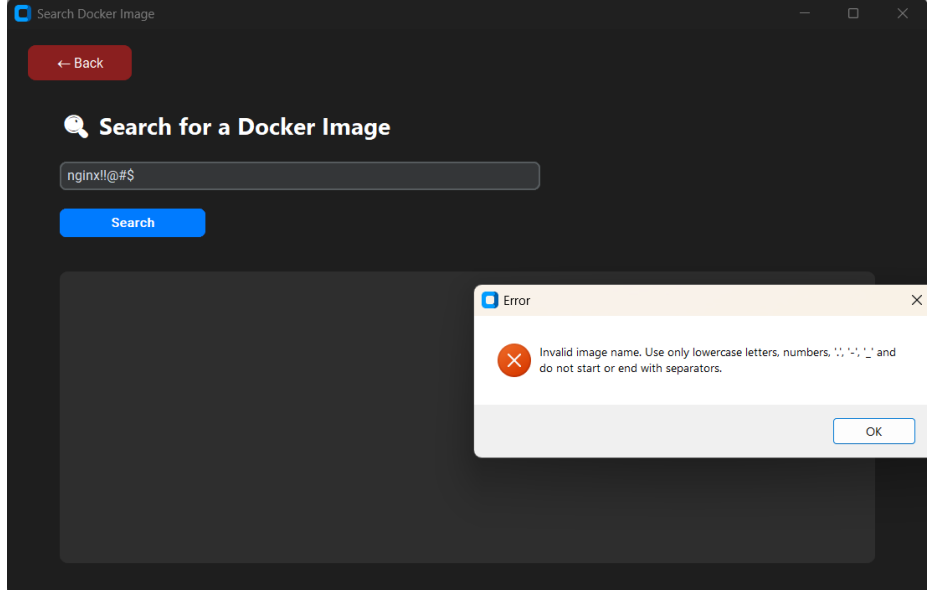
<i>Test Case Name</i>	SDKI2: Search for a Docker Image using invalid name
<i>Test Scenario</i>	Users should be able to search for Docker images using invalid name.
<i>Prerequisites</i>	Backend is properly integrated with Docker Hub
<i>Test Input</i>	-
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Enter a invalid image name in the input field. 2. Click on the Search button.
<i>Expected Behavior</i>	An error message or “No results found” should be shown.
<i>Assumptions</i>	Backend handles both invalid.
<i>Actual Results</i>	Error messages or empty results are shown correctly.
<i>Status</i>	Pass
<i>Real Life Testing</i>	 <p>The screenshot shows a web application titled 'Search Docker Image'. It has a dark theme. At the top left is a red 'Back' button. Below it is a search bar with the text 'nanan' and a blue 'Search' button. An error dialog box is open in the foreground, titled 'Error' with a red 'X' icon. The message inside says 'No Image with this Name Locally Create or Pull it'. There is an 'OK' button at the bottom right of the dialog box.</p>

<i>Test Case Name</i>	SDKI3: Search with Empty Input
<i>Test Scenario</i>	Search should not proceed when no input is provided.
<i>Prerequisites</i>	-
<i>Test Input</i>	-
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Navigate to the Search Docker Image page. 2. Leave the input field empty. 3. Click the Search button.
<i>Expected Behavior</i>	An error message prompting the user to enter an image name.
<i>Assumptions</i>	Input validation is handled on the backend.
<i>Actual Results</i>	Proper validation message shown
<i>Status</i>	Pass
<i>Real Life Testing</i>	

Test Case Name	SDKI4: Search with Special Characters
Test Scenario	The system should handle invalid characters.
Prerequisites	-
Test Input	nginx!!@#\$
Execution Steps	<ol style="list-style-type: none">1. Enter special characters in the search bar.2. Click Search.
Expected Behavior	An error message prompting the user to enter a valid name.
Assumptions	Input validation is handled on the backend.
Actual Results	Proper validation message shown
Status	Pass
Real Life Testing	 A screenshot of a web application titled "Search Docker Image". It features a "← Back" button, a search bar containing "nginx!!@#\$", and a blue "Search" button. An error dialog box is displayed in the foreground with the title "Error" and the message: "Invalid image name. Use only lowercase letters, numbers, ., -, _ and do not start or end with separators." with an "OK" button.

<i>Test Case Name</i>	GTC1: Navigation Consistency
<i>Test Scenario</i>	Verify that all navigation elements (Back button, side nav) work as expected and consistently.
<i>Prerequisites</i>	-
<i>Test Input</i>	-
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Click Back from different pages. 2. Use side navigation to jump between sections.
<i>Expected Behavior</i>	Navigation takes the user to the correct page.
<i>Assumptions</i>	navigation consistency
<i>Actual Results</i>	navigation consistency
<i>Status</i>	Pass
<i>Real Life Testing</i>	

<i>Test Case Name</i>	GTC2: Visual Alignment and Spacing
<i>Test Scenario</i>	Check that all components are well-aligned and spaced correctly.
<i>Prerequisites</i>	-
<i>Test Input</i>	-
<i>Execution Steps</i>	<ol style="list-style-type: none"> 1. Inspect each page layout visually. 2. Resize the window and verify alignment holds.
<i>Expected Behavior</i>	<ul style="list-style-type: none"> - No misalignment, overflow, or awkward spacing. - Text, buttons, and inputs appear balanced.
<i>Assumptions</i>	Consistent Visual Alignment and Spacing
<i>Actual Results</i>	Consistent Visual Alignment and Spacing
<i>Status</i>	Pass
<i>Real Life Testing</i>	

<i>Test Case Name</i>	GTC3: Error Message Clarity
<i>Test Scenario</i>	Users should receive clear, actionable error messages for backend failures or invalid actions.
<i>Prerequisites</i>	-
<i>Test Input</i>	-
<i>Execution Steps</i>	1. Try actions like "Search Image" or "Create Disk" with invalid input.
<i>Expected Behavior</i>	<ul style="list-style-type: none"> - Error message is clear - No raw error logs or developer traces visible.
<i>Assumptions</i>	Clear error messages
<i>Actual Results</i>	Clear error messages
<i>Status</i>	Pass
<i>Real Life Testing</i>	 <p>The screenshot shows a web application titled "Search Docker Image". It has a "← Back" button and a search bar with the text "nginx!!@#\$". Below the search bar is a blue "Search" button. An error dialog box is open in the bottom right corner with the title "Error". The message inside the dialog says: "Invalid image name. Use only lowercase letters, numbers, '.', ':', '-' and do not start or end with separators." There is an "OK" button at the bottom right of the dialog.</p>

Testing:

Testing Approach:

- Unit Testing: Each module, including disk and VM creation, was unit tested to ensure accuracy in functionality.
- System Validation Testing: Ensured that system resource checks (CPU, RAM, disk) work accurately before creating VMs.
- User Interface Testing: Verified the Tkinter GUI for proper user interaction, ensuring inputs are captured correctly.
- Compatibility Testing:
 - Tested with different virtual disk formats (qcow2, vmdk, etc.).
 - Docker image building tested using multiple Dockerfiles and build contexts.

Key Results:

- System resource validation: Prevented VM creation when resources were insufficient.
- Multiple disk format handling: Ensured disk compatibility without failures.
- Docker image build functionality: Verified successful image creation and correct error handling.
- GUI interaction: Inputs were correctly processed, and visual feedback was intuitive across features.

User Manual:

How to Use VirtuManager

1. Installation:

- Install dependencies using the following command:
pip install psutil

2. Create Virtual Disk:

- Launch VirtuManager.
- Enter the required parameters (disk name, path, format, and size).
- Click on "Create Disk" to generate the virtual disk.

3. Create Virtual Machine:

- In the GUI, the CPU, RAM, disk, and ISO image.
- Click "Create VM." The system will validate available resources before proceeding.
- If resources are insufficient, a warning will appear; otherwise, the VM will be created.

4. Build Docker Image:

- Navigate to the Docker section from the main menu.
- Enter the Image Name and Tag.
- Select the Dockerfile via file browser.
- Select the Build Context Folder via directory browser.
- Click "Build" to start image creation.
- A progress bar will display the build process.
- After completion, a dialog will show success or error feedback.

5. Running the Application:

- Open the terminal and write *python main.py*.
- The GUI will appear, allowing interaction with the disk and VM creation features.

Troubleshooting:

• Insufficient Resources (VM Creation):

- Close background apps or increase available resources.

• Invalid Disk Format:

- Only use supported formats like qcow2, vmdk, etc.

• Docker Build Failure:

- Ensure the Dockerfile and context folder are valid.
- Confirm Docker is properly installed and running on your system.