# **CS550 Software Engineering**

# **SafeHome Project**

# **Implementation and Testing**

**Project Team #9 Members:**

* Sunwoo Han
* Grace Min
* Leo Park
* Ethan Ortiz
* Priya Patel

## **I. Overview**

### **1. Introduction**

This document contains implementation and test information about Team 9’s SafeHome Project implementation, specifically focusing on the complete realization of UC2 (Security), alongside UC1 (Common Functions) and UC3 (Surveillance). We have implemented the SafeHome System using a Python 3.11 stack (Tkinter for Control Panel, Flask for Web Interface). We also performed testing by using PyTest for unit/integration testing and Coverage.py for structural analysis. This document contains full information about the test procedure and results, ensuring traceability from the SRS requirements and SDS architecture through to executable verification.

### **2. Goal**

* To exercise the software engineering principles in the implementation phase following the traceability from SRS and SDS.
* To trace every UC1–UC3 requirement back to executable code and tests, mapping artifacts such as multi-factor login, arm/disarm flows, and panic logic to concrete behavior.
* To learn how to perform testing in various levels:
  + **Unit Level:** verifying distinct class responsibilities (e.g., SecuritySystem, CameraController).
  + **Integration Level:** verifying subsystem interactions (e.g., SystemController orchestrating SecuritySystem).
  + **System Level:** verifying end-to-end flows via the Tkinter Control Panel and Flask Web Dashboard.
* To document the program branching coverage and maintain high reliability for critical security modules.

## **II. Class Diagrams**

*Note: As per project constraints, architectural diagrams are described textually below, mapping implementation classes to the SDS design clusters.*

### **1. UC1 – Common Functions Cluster**

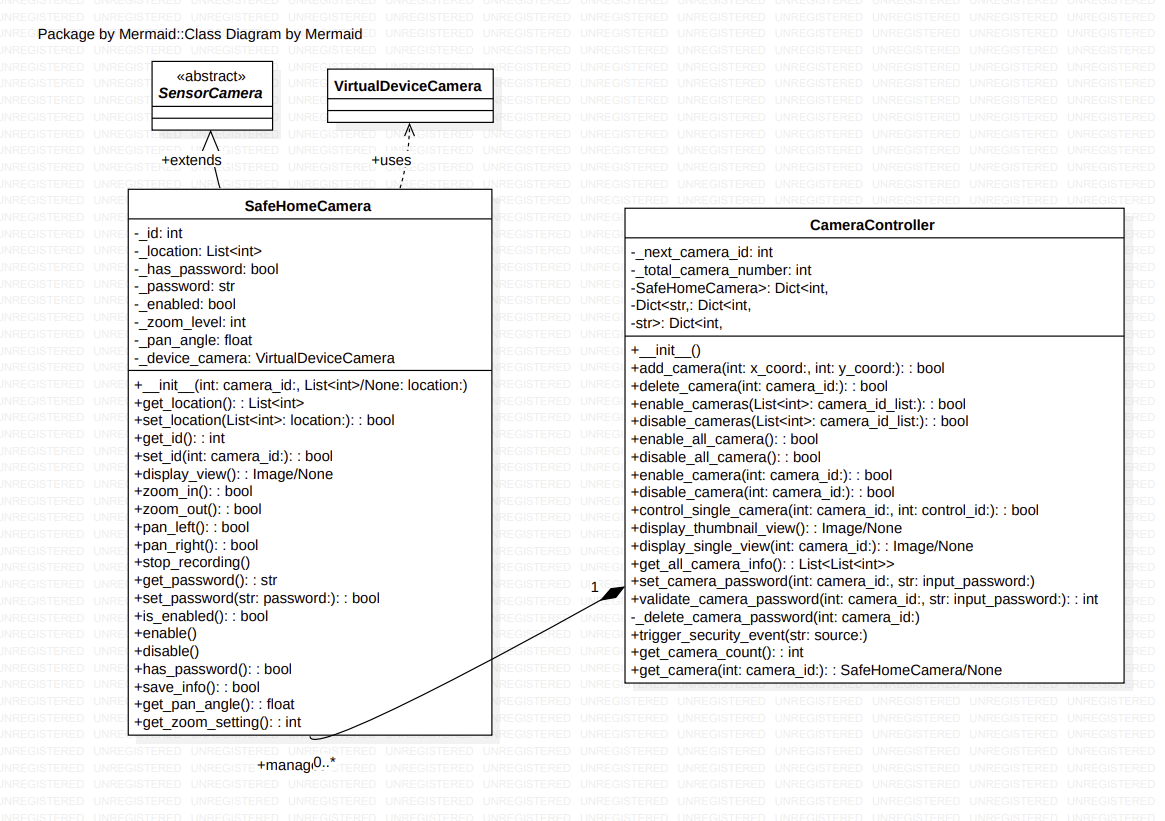
* **System Orchestrator (domain/system.py):** Centralizes the SDS Common Function controller responsibilities. It sequences turn-on/turn-off/reset operations (SDS VI.1a, VI.1c), enforces state guards via SystemState, and exposes change-password phases.
* **Auth & Settings Services (auth/login\_manager.py, domain/services/settings\_service.py):** Implements the SDS authentication and configuration CRC cards. Wraps credential validation, lock timers, and audit logging.
* **UI Façades (ui/main\_window.py, main.py):** Hosts the login, power, settings, and reset views referenced by SDS Common UI diagrams.

### **2. UC2 – Security Cluster**

* **SecuritySystem Nucleus (security/security\_system.py):** Encapsulates the alarm state machine (SDS III.34). Implements entry delays, panic escalation, and intrusion logging following the Alarm Condition Encountered sequence (SDS VI.2d).
* **System Controller (domain/system\_controller.py):** Routes device signals to the alarm service (SDS IV.2). Translates simulator input into SensorEvents and orchestrates arming/disarming using shared constants.
* **Configuration Managers (config/configuration\_manager.py):** Manages SafeHome modes, zones, and sensor-zone assignments (SDS IV.2), supporting selective arming (UC2c).

### **3. UC3 – Surveillance Cluster**

* **Camera Domain (surveillance/camera\_controller.py):** Realizes SDS Surveillance class diagrams. Handles add/delete/enable/disable/pan/zoom flows.
* **SafeHomeCamera (surveillance/safehome\_camera.py):** Adapts the virtual device layer for password-protected capture per SRS UC3a–UC3g.
* **Camera Gateway:** Bridges UC2 alarm events to UC3 capture workflows (SDS VI.2k).



## **III. Unit Tests**

### **1. External Communication Management**

#### **A. Control Panel Management**

**1. ControlPanelLoginPresenter (Login Logic)**

**A. attempt\_login() (handling credentials from CP)**

**a. test\_success\_navigates\_to\_main\_menu (UT-CP-Login-Success)**

|

| Field | Value |

| Class, Method, Author, Date, Version | ControlPanelLoginPresenter, attempt\_login, Team 9 UC1, 2025-11-25, 1.0 |

| Test Case Description | Verifies that supplying valid admin credentials to the control-panel presenter results in a successful login state and navigation command. |

| Input Specifications | Call attempt\_login("admin", "correct\_password") where AuthService is mocked to return True. |

| Expected Result | Presenter returns success=True and Maps\_to="MainMenu". |

| Actual Result | Pass: Returns success=True and Maps\_to="MainMenu". |

| Comment | Covers SRS Common Use Case 1 and SDS Common Sequence Diagram (login happy path). |

**2. ControlPanelResetPresenter (System Reset)**

**A. perform\_reset() (handling reset requests)**

**a. test\_reset\_success\_dict (UT-CP-Reset-Success)**

| Field | Value |

| Class, Method, Author, Date, Version | ControlPanelResetPresenter, perform\_reset, Team 9 UC1, 2025-11-25, 1.0 |

| Test Case Description | Verifies that the presenter correctly processes a dictionary response from the system reset operation and updates the UI state. |

| Input Specifications | Call perform\_reset() when System.reset() returns {'status': 'success', 'phase': 2}. |

| Expected Result | Presenter marks operation success and surfaces the reinitialization message phase. |

| Actual Result | Pass: Outcome marks success and surfaces message/phase. |

| Comment | Ties to SRS Common Use Case “Reset the system” and SDS Common reset state machine. |

#### **B. Web Interface Management**

**1. WebLoginPresenter**

**A. validate\_first\_password()**

**a. test\_first\_password\_success (UT-Web-Login-Phase1)**

| Field | Value |

| Class, Method, Author, Date, Version | WebLoginPresenter, validate\_first\_password, Team 9 UC1, 2025-11-25, 1.0 |

| Test Case Description | Simulates a username/password submission over the browser interface to validate the first phase of multi-factor authentication. |

| Input Specifications | valid username and password payload sent to validate\_first\_password. |

| Expected Result | Presenter marks success and advances state to the second-factor prompt. |

| Actual Result | Pass: Presenter marks success and advances to second-factor prompt. |

| Comment | Traces SRS Common Use Case 2 (Log onto system through web browser) and SDS VI.1b remote-login sequence. |

2. Surveillance

1. Camera Management
2. SafeHomeCamera class
3. get\_location()
   1. test\_get\_location()

| **Item** | **test\_get\_location (UT-CAM-LOC-CPY)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | get\_location() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling get\_location() returns a **defensive copy** of the location list, ensuring the camera's internal state is protected from external modifications. |
| **Input Specifications** | Initial location: [100, 200]. Action: Modify the returned list value. |
| **Detailed Step** | 1. Call get\_location() and store the list.  2. Modify the list (e.g., change index 0 to 999).  3. Call get\_location() again to assert the internal value remains [100, 200]. |
| **Expected Result** | The internal location must remain [100, 200], verifying that a copy was returned. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Internal state was preserved. |
| **Comment (including references)** | Verification of defensive programming based on the Object State Protection Principle. |

* 1. test\_get\_location\_returns\_copy

| **Item** | **test\_get\_location\_returns\_copy (UT-CAM-LOC-CPY-ENF)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | get\_location() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that appending elements to the returned list does not affect the original list's length, reinforcing the defensive copy check. |
| **Input Specifications** | Initial location: [100, 200]. Action: Append 999 to the returned list. |
| **Detailed Step** | 1. Call get\_location() and append a new element to the returned list.  2. Call get\_location() again and assert the returned list's length is still 2. |
| **Expected Result** | The original list length must remain 2. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Defensive copy prevented unauthorized length modification. |
| **Comment (including references)** | Coverage reinforcement for the defensive copy check. |

* 1. test\_initialization\_with\_location

| **Item** | **test\_initialization\_with\_location (UT-CAM-INIT-LOC)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | \_\_init\_\_(), get\_location() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when the object is initialized with a specific location, get\_location() correctly retrieves the set coordinates. |
| **Input Specifications** | Instantiate SafeHomeCamera with location=[100, 200]. |
| **Detailed Step** | 1. Instantiate the object with the specified location.  2. Call get\_location() and assert the returned value. |
| **Expected Result** | get\_location() must return exactly [100, 200]. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Initial location was correctly set and retrieved. |
| **Comment (including references)** | Validation of initial attribute setting. |

* 1. test\_initialization\_without\_location

| **Item** | **test\_initialization\_without\_location (UT-CAM-INIT-DEF)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | \_\_init\_\_(), get\_location() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when the object is initialized without specifying a location, get\_location() correctly returns the default coordinates [0, 0]. |
| **Input Specifications** | Instantiate SafeHomeCamera omitting the location parameter. |
| **Detailed Step** | 1. Instantiate the object without the location.  2. Call get\_location() and assert the returned value. |
| **Expected Result** | get\_location() must return the default location [0, 0]. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Default location [0, 0] was correctly assigned. |
| **Comment (including references)** | Verification of default value handling. |

* 1. test\_set\_location

| **Item** | **test\_set\_location (UT-CAM-LOC-SET)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_location(), get\_location() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling set\_location() with valid coordinates successfully updates the location and returns True. |
| **Input Specifications** | Initial location: [100, 200]. Action: Call set\_location([300, 400]). |
| **Detailed Step** | 1. Call set\_location([300, 400]).  2. Assert the return value is True.  3. Call get\_location() and assert the new value [300, 400]. |
| **Expected Result** | set\_location() must return True, and get\_location() must return [300, 400]. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Location was successfully updated. |
| **Comment (including references)** | Validation of the camera coordinate setting functionality. |

* 1. test\_set\_invalid\_location

| **Item** | **test\_set\_invalid\_location (UT-CAM-LOC-INV)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_location(), get\_location() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that an invalid location format (e.g., missing Y-coordinate) is rejected, returns False, and preserves the existing location. |
| **Input Specifications** | Initial location: [100, 200]. Action: Call set\_location([100]) (invalid format). |
| **Detailed Step** | 1. Call set\_location([100]).  2. Assert the return value is False.  3. Call get\_location() and assert the location remains [100, 200]. |
| **Expected Result** | set\_location() must return False, and get\_location() must remain [100, 200]. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Invalid input was correctly rejected, and state was preserved. |
| **Comment (including references)** | Validation of input validation and error handling logic. |

1. set\_location()
   1. test\_set\_location

| **Item** | **test\_set\_location (UT-CAM-LOC-SET)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_location(), get\_location() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling set\_location() with valid coordinates ([x, y]) successfully updates the location, returns True, and the new value is retrieved by get\_location(). |
| **Input Specifications** | Initial location: [100, 200]. Action: Call set\_location([300, 400]). |
| **Detailed Step** | 1. Call set\_location([300, 400]).  2. Assert the return value is True.  3. Call get\_location() and assert the new value [300, 400]. |
| **Expected Result** | set\_location() must return True, and get\_location() must return [300, 400]. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Location was successfully updated. |
| **Comment (including references)** | Validation of the camera coordinate setting functionality. |

* 1. test\_set\_invalid\_location

| **Item** | **test\_set\_invalid\_location (UT-CAM-LOC-INV-BAS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_location(), get\_location() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that an invalid location format (specifically, missing the Y-coordinate) is rejected, returns False, and preserves the existing location. |
| **Input Specifications** | Initial location: [100, 200]. Action: Call set\_location([100]) (invalid format). |
| **Detailed Step** | 1. Call set\_location([100]).  2. Assert the return value is False.  3. Call get\_location() and assert the location remains [100, 200]. |
| **Expected Result** | set\_location() must return False, and get\_location() must remain [100, 200]. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Invalid input was correctly rejected, and state was preserved. |
| **Comment (including references)** | Validation of input validation and error handling logic. |

* 1. test\_set\_location\_invalid\_lengths

| **Item** | **test\_set\_location\_invalid\_lengths (UT-CAM-LOC-INV-COV)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_location(), get\_location() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that the input validation handles various incorrect list lengths (e.g., 0, 1, or 3 elements), rejecting them and ensuring the current location is unchanged. |
| **Input Specifications** | Initial location: [100, 200]. Action: Call set\_location with [], [100], and [100, 200, 300] sequentially. |
| **Detailed Step** | 1. Iterate through invalid inputs.  2. For each, assert set\_location returns False.  3. Assert get\_location() remains [100, 200] after each attempt. |
| **Expected Result** | set\_location() must return False for all invalid inputs, and the location must remain [100, 200]. |
| **Actual Result (Pass/Fail/Exception)** | Pass: All invalid list lengths were correctly handled and rejected. |
| **Comment (including references)** | Coverage test ensuring validation handles various incorrect list lengths. |

1. get\_id()
   1. test\_get\_id

| **Item** | **test\_get\_id (UT-CAM-ID-GET)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | get\_id() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that the camera's unique ID, assigned during initialization, is correctly retrieved. |
| **Input Specifications** | Initial state: Camera ID is 1 (set by fixture/initialization). |
| **Detailed Step** | 1. Instantiate the camera object.  2. Call get\_id().  3. Assert the returned ID matches the initialized value. |
| **Expected Result** | get\_id() must return 1. |
| **Actual Result (Pass/Fail/Exception)** | Pass: ID was correctly retrieved. |
| **Comment (including references)** | Basic getter functionality check for the camera's static identifier. |

* 1. test\_initialization\_with\_location

| **Item** | **test\_initialization\_with\_location (UT-CAM-INIT-ID)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | \_\_init\_\_(), get\_id() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that the camera ID (1) is correctly set and immediately retrieved after object initialization. |
| **Input Specifications** | Instantiate SafeHomeCamera with camera\_id=1. |
| **Detailed Step** | 1. Instantiate the object with the specified ID.  2. Call get\_id() and assert the returned value. |
| **Expected Result** | get\_id() must return 1. |
| **Actual Result (Pass/Fail/Exception)** | Pass: ID was correctly set during initialization. |
| **Comment (including references)** | Fundamental initial state check for ID consistency. |

* 1. test\_set\_id

| **Item** | **test\_set\_id (UT-CAM-ID-SET-PASS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_id(), get\_id() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling set\_id() with a valid ID successfully updates the ID and is correctly retrieved by get\_id(). |
| **Input Specifications** | Initial ID: 1. Action: Call set\_id(5). |
| **Detailed Step** | 1. Call set\_id(5).  2. Assert the return value is True.  3. Call get\_id() and assert the new ID is 5. |
| **Expected Result** | set\_id() must return True, and get\_id() must return 5. |
| **Actual Result (Pass/Fail/Exception)** | Pass: ID was successfully changed and retrieved. |
| **Comment (including references)** | Verification of successful ID mutation and retrieval. |

* 1. test\_set\_invalid\_id

| **Item** | **test\_set\_invalid\_id (UT-CAM-ID-SET-FAIL)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_id(), get\_id() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling set\_id() with an invalid ID (e.g., non-positive integer) is rejected, returns False, and the original ID is preserved. |
| **Input Specifications** | Initial ID: 1. Action: Call set\_id(0) and set\_id(-1). |
| **Detailed Step** | 1. Call set\_id(0).  2. Assert the return value is False.  3. Call get\_id() and assert the ID remains 1. |
| **Expected Result** | set\_id() must return False, and get\_id() must remain 1. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Invalid ID attempts were rejected, and state was preserved. |
| **Comment (including references)** | Validation of ID input validation and state preservation. |

1. set\_id()
   1. test\_set\_id

| **Item** | **test\_set\_id (UT-CAM-ID-SET-PASS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_id(), get\_id() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling set\_id() with a valid positive integer successfully updates the camera's ID and is correctly retrieved by get\_id(). |
| **Input Specifications** | Initial ID: 1. Action: Call set\_id(5). |
| **Detailed Step** | 1. Call set\_id(5).  2. Assert the return value is True.  3. Call get\_id() and assert the new ID is 5. |
| **Expected Result** | set\_id() must return True, and get\_id() must return 5. |
| **Actual Result (Pass/Fail/Exception)** | Pass: ID was successfully changed and retrieved. |
| **Comment (including references)** | Verification of successful ID mutation and retrieval. |

* 1. test\_set\_invalid\_id

| **Item** | **test\_set\_invalid\_id (UT-CAM-ID-SET-FAIL)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_id(), get\_id() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling set\_id() with non-positive integers (0 or negative) is rejected, returns False, and preserves the original ID. |
| **Input Specifications** | Initial ID: 1. Action: Call set\_id(0) and then set\_id(-1). |
| **Detailed Step** | 1. Call set\_id(0).  2. Assert the return value is False.  3. Call get\_id() and assert the ID remains the initial value (1). |
| **Expected Result** | set\_id() must return False for non-positive inputs, and get\_id() must return 1. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Invalid ID attempts were rejected, and state was preserved. |
| **Comment (including references)** | Validation of ID input validation (ID must be a positive integer) and state preservation. |

* 1. test\_set\_id\_boundary\_values

| **Item** | **test\_set\_id\_boundary\_values (UT-CAM-ID-BOUND)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_id(), get\_id() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the ID setter correctly handles valid boundary values, specifically the minimum positive integer (1) and a large valid ID (99999). |
| **Input Specifications** | Initial state ID: 1. Action: Call set\_id(1) and then set\_id(99999). |
| **Detailed Step** | 1. Call set\_id(1) and assert result is True.  2. Call set\_id(99999) and assert result is True.  3. Verify get\_id() reflects the final value 99999. |
| **Expected Result** | set\_id() must return True for both valid boundary attempts, and the final get\_id() must return 99999. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Valid ID boundaries were correctly accepted. |
| **Comment (including references)** | Comprehensive check of valid ID boundaries for the set\_id method. |

1. display\_view()
   1. test\_display\_view

| **Item** | **test\_display\_view (UT-CAM-VIEW-PASS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | display\_view() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when the camera is enabled, display\_view() successfully returns a non-null object, confirming the feed is accessible. |
| **Input Specifications** | Initial state: Camera is enabled (is\_enabled() is True). |
| **Detailed Step** | 1. Ensure the camera is enabled.  2. Call display\_view().  3. Assert the result is not None and is an instance of PIL.Image.Image. |
| **Expected Result** | Returns a valid PIL.Image.Image object. |
| **Actual Result (Pass/Fail/Exception)** | Pass: View object was successfully retrieved. |
| **Comment (including references)** | Core function test for accessing the camera feed. |

* 1. test\_display\_view\_disabled\_camera

| **Item** | **test\_display\_view\_disabled\_camera (UT-CAM-VIEW-FAIL)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | display\_view(), disable() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when the camera is disabled, display\_view() enforces the security policy by returning None, blocking access to the feed. |
| **Input Specifications** | Action: camera.disable() is called to set the state to disabled. |
| **Detailed Step** | 1. Call camera.disable().  2. Call display\_view().  3. Assert the result is None. |
| **Expected Result** | Returns None. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Access to the view was correctly denied when disabled. |
| **Comment (including references)** | Security policy validation for access control in a disabled state. |

* 1. test\_display\_view\_exception\_handling

| **Item** | **test\_display\_view\_exception\_handling (UT-CAM-VIEW-EXC)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | display\_view() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the method's robustness by ensuring that if the underlying device camera throws an exception during view retrieval, the method handles it gracefully and returns None. |
| **Input Specifications** | Mocking: The underlying device camera's get\_view() method is patched to raise an Exception("Camera error"). |
| **Detailed Step** | 1. Patch get\_view() to raise an exception.  2. Call display\_view() within the patched context.  3. Assert the result is None. |
| **Expected Result** | Returns None, and the system does not crash or raise the exception. |
| **Actual Result (Pass/Fail/Exception)** | Pass: External component failure was handled gracefully. |
| **Comment (including references)** | Robustness check for failures in the virtual device layer. |

1. zoom\_in()
   1. test\_zoom\_in

| **Item** | **test\_zoom\_in (UT-CAM-ZOOM-IN-PASS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | zoom\_in(), get\_zoom\_setting() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when the current zoom level is below the maximum, calling zoom\_in() returns True and increases the zoom level by exactly 1. |
| **Input Specifications** | Initial state: Zoom level is 1. Action: Call zoom\_in() once. |
| **Detailed Step** | 1. Call zoom\_in().  2. Assert the return value is True.  3. Call get\_zoom\_setting() and assert the new level is 2. |
| **Expected Result** | zoom\_in() must return True, and the new zoom\_setting must be 2. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Zoom level was successfully incremented. |
| **Comment (including references)** | Core function test for camera zoom operation. |

* 1. test\_zoom\_in\_max

| **Item** | **test\_zoom\_in\_max (UT-CAM-ZOOM-IN-MAX)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | zoom\_in(), get\_zoom\_setting() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the boundary condition: when the maximum zoom level (10) is reached, subsequent calls to zoom\_in() fail, return False, and the level remains 10. |
| **Input Specifications** | Action: Repeatedly call zoom\_in() until the level reaches 10, then call it one more time. |
| **Detailed Step** | 1. Set zoom level to 10.  2. Call zoom\_in().  3. Assert the return value is False.  4. Assert get\_zoom\_setting() is still 10. |
| **Expected Result** | zoom\_in() must return **False**, and the zoom level must remain 10. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Zoom operation was correctly blocked at the maximum limit. |
| **Comment (including references)** | Boundary condition check for maximum zoom level. |

* 1. test\_zoom\_in\_disabled\_camera

| **Item** | **test\_zoom\_in\_disabled\_camera (UT-CAM-ZOOM-IN-FAIL)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | zoom\_in(), disable(), get\_zoom\_setting() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling zoom\_in() on a disabled camera is rejected, returns False, and preserves the original zoom level. |
| **Input Specifications** | Initial state: Zoom level 1. Action: camera.disable() is called. |
| **Detailed Step** | 1. Disable the camera.  2. Record the initial zoom level.  3. Call zoom\_in() and assert the return value is False.  4. Assert the zoom level has not changed. |
| **Expected Result** | zoom\_in() must return False, and the zoom level must remain the initial value. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Control command was correctly blocked in the disabled state. |
| **Comment (including references)** | Security and state awareness check for control operations. |

* 1. test\_zoom\_in\_device\_camera\_integration

| **Item** | **test\_zoom\_in\_device\_camera\_integration (UT-CAM-ZOOM-IN-INT)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | zoom\_in() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling zoom\_in() successfully executes the underlying virtual device interaction logic and returns a boolean result, confirming proper integration. |
| **Input Specifications** | Initial state: Camera is enabled and below max zoom. |
| **Detailed Step** | 1. Call zoom\_in().  2. Assert the return value is a boolean (True) and the internal logic path for device interaction is executed. |
| **Expected Result** | The method must execute the device interaction path and return True. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Integration with the virtual device layer for zoom control was confirmed. |
| **Comment (including references)** | Verifies the linkage between the domain model and the virtual device simulator. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |

1. zoom\_out()
   1. test\_zoom\_out

| **Item** | **test\_zoom\_out (UT-CAM-ZOOM-OUT-PASS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | zoom\_out(), get\_zoom\_setting() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when the current zoom level is above the minimum, calling zoom\_out() returns True and decreases the zoom level by exactly 1. |
| **Input Specifications** | Initial state: Zoom level is set to 2 (after one zoom\_in call). Action: Call zoom\_out() once. |
| **Detailed Step** | 1. Ensure zoom level is > 1.  2. Call zoom\_out().  3. Assert the return value is True.  4. Call get\_zoom\_setting() and assert the new level is 1. |
| **Expected Result** | zoom\_out() must return True, and the new zoom\_setting must be 1. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Zoom level was successfully decremented. |
| **Comment (including references)** | Core function test for camera zoom operation. |

* 1. test\_zoom\_out\_min

| **Item** | **test\_zoom\_out\_min (UT-CAM-ZOOM-OUT-MIN)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | zoom\_out(), get\_zoom\_setting() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the boundary condition: when the minimum zoom level (1) is reached, subsequent calls to zoom\_out() fail, return False, and the level remains unchanged. |
| **Input Specifications** | Initial state: Zoom level is 1. Action: Call zoom\_out() once. |
| **Detailed Step** | 1. Ensure zoom level is 1.  2. Call zoom\_out().  3. Assert the return value is False.  4. Assert get\_zoom\_setting() is still 1. |
| **Expected Result** | zoom\_out() must return **False**, and the zoom level must remain 1. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Zoom operation was correctly blocked at the minimum limit. |
| **Comment (including references)** | Boundary condition check for minimum zoom level. |

* 1. test\_zoom\_out\_disabled\_camera

| **Item** | **test\_zoom\_out\_disabled\_camera (UT-CAM-ZOOM-OUT-FAIL)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | zoom\_out(), disable(), get\_zoom\_setting() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling zoom\_out() on a disabled camera is rejected, returns False, and preserves the original zoom level. |
| **Input Specifications** | Initial state: Zoom level 2. Action: camera.disable() is called. |
| **Detailed Step** | 1. Disable the camera.  2. Record the initial zoom level.  3. Call zoom\_out() and assert the return value is False.  4. Assert the zoom level has not changed. |
| **Expected Result** | zoom\_out() must return False, and the zoom level must remain the initial value. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Control command was correctly blocked in the disabled state. |
| **Comment (including references)** | Security and state awareness check for control operations. |

* 1. test\_zoom\_out\_device\_camera\_integration

| **Item** | **test\_zoom\_out\_device\_camera\_integration (UT-CAM-ZOOM-OUT-INT)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | zoom\_out() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling zoom\_out() correctly executes the underlying virtual device interaction logic and returns a boolean result, confirming proper integration. |
| **Input Specifications** | Initial state: Camera is enabled and zoomed in (e.g., level 2). |
| **Detailed Step** | 1. Ensure zoom level > 1.  2. Call zoom\_out().  3. Assert the return value is a boolean (True) and the internal logic path for device interaction is executed. |
| **Expected Result** | The method must execute the device interaction path and return True. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Integration with the virtual device layer for zoom control was confirmed. |
| **Comment (including references)** | Verifies the linkage between the domain model and the virtual device simulator. |

1. pan\_left()
   1. test\_pan\_left

| **Item** | **test\_pan\_left (UT-CAM-PAN-LEFT-PASS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | pan\_left(), get\_pan\_angle() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when the pan angle is above the minimum limit, calling pan\_left() returns True and decreases the angle by exactly 5.0 degrees. |
| **Input Specifications** | Initial state: Pan angle is 0.0 degrees. Action: Call pan\_left() once. |
| **Detailed Step** | 1. Call pan\_left().  2. Assert the return value is True.  3. Call get\_pan\_angle() and assert the new angle is -5.0 degrees. |
| **Expected Result** | pan\_left() must return True, and the new pan\_angle must be -5.0. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Pan angle was successfully decremented. |
| **Comment (including references)** | Core function test for camera pan operation. |

* 1. test\_pan\_left\_limit

| **Item** | **test\_pan\_left\_limit (UT-CAM-PAN-LEFT-MIN)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | pan\_left(), get\_pan\_angle() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the boundary condition: once the minimum pan angle (-180.0 degrees) is reached, subsequent calls to pan\_left() preserve the limit and the angle does not decrease further. |
| **Input Specifications** | Action: Repeatedly call pan\_left() until the angle reaches -180.0 degrees, then call it one more time. |
| **Detailed Step** | 1. Set pan angle to -180.0 degrees.  2. Call pan\_left().  3. Assert the return value is True.  4. Assert get\_pan\_angle() is still -180.0. |
| **Expected Result** | pan\_left() must return True, and the angle must remain -180.0 degrees. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Pan operation was correctly blocked at the minimum limit. |
| **Comment (including references)** | Boundary condition check for minimum pan angle. |

* 1. test\_pan\_left\_disabled\_camera

| **Item** | **test\_pan\_left\_disabled\_camera (UT-CAM-PAN-LEFT-FAIL)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | pan\_left(), disable(), get\_pan\_angle() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling pan\_left() on a disabled camera is rejected, returns False, and preserves the original angle. |
| **Input Specifications** | Initial state: Angle 0.0. Action: camera.disable() is called. |
| **Detailed Step** | 1. Disable the camera.  2. Record the initial angle.  3. Call pan\_left() and assert the return value is False.  4. Assert the angle has not changed. |
| **Expected Result** | pan\_left() must return False, and the angle must remain the initial value. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Control command was correctly blocked in the disabled state. |
| **Comment (including references)** | Security and state awareness check for control operations. |

* 1. test\_pan\_left\_device\_camera\_integration

| **Item** | **test\_pan\_left\_device\_camera\_integration (UT-CAM-PAN-LEFT-INT)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | pan\_left() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling pan\_left() successfully executes the underlying virtual device interaction logic and returns a boolean result, confirming proper integration. |
| **Input Specifications** | Initial state: Camera is enabled and at 0.0 degrees. |
| **Detailed Step** | 1. Call pan\_left().  2. Assert the return value is a boolean (True) and the internal logic path for device interaction is executed. |
| **Expected Result** | The method must execute the device interaction path and return True. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Integration with the virtual device layer for pan control was confirmed. |
| **Comment (including references)** | Verifies the linkage between the domain model and the virtual device simulator. |

1. pan\_right()
   1. test\_pan\_right

| **Item** | **test\_pan\_right (UT-CAM-PAN-RIGHT-PASS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | pan\_right(), get\_pan\_angle() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when the pan angle is below the maximum limit, calling pan\_right() returns True and increases the angle by exactly 5.0 degrees. |
| **Input Specifications** | Initial state: Pan angle is 0.0 degrees. Action: Call pan\_right() once. |
| **Detailed Step** | 1. Call pan\_right().  2. Assert the return value is True.  3. Call get\_pan\_angle() and assert the new angle is 5.0 degrees. |
| **Expected Result** | pan\_right() must return True, and the new pan\_angle must be 5.0. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Pan angle was successfully incremented. |
| **Comment (including references)** | Core function test for camera pan operation. |

* 1. test\_pan\_right\_limit

| **Item** | **test\_pan\_right\_limit (UT-CAM-PAN-RIGHT-MAX)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | pan\_right(), get\_pan\_angle() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the boundary condition: once the maximum pan angle (+180.0 degrees) is reached, subsequent calls to pan\_right() preserve the limit and the angle does not increase further. |
| **Input Specifications** | Action: Repeatedly call pan\_right() until the angle reaches 180.0 degrees, then call it one more time. |
| **Detailed Step** | 1. Set pan angle to 180.0 degrees.  2. Call pan\_right().  3. Assert the return value is True.  4. Assert get\_pan\_angle() is still 180.0. |
| **Expected Result** | pan\_right() must return True, and the angle must remain 180.0 degrees. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Pan operation was correctly blocked at the maximum limit. |
| **Comment (including references)** | Boundary condition check for maximum pan angle. |

* 1. test\_pan\_right\_disabled\_camera

| **Item** | **test\_pan\_right\_disabled\_camera (UT-CAM-PAN-RIGHT-FAIL)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | pan\_right(), disable(), get\_pan\_angle() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling pan\_right() on a disabled camera is rejected, returns False, and preserves the original angle. |
| **Input Specifications** | Initial state: Angle 0.0. Action: camera.disable() is called. |
| **Detailed Step** | 1. Disable the camera.  2. Record the initial angle.  3. Call pan\_right() and assert the return value is False.  4. Assert the angle has not changed. |
| **Expected Result** | pan\_right() must return False, and the angle must remain the initial value. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Control command was correctly blocked in the disabled state. |
| **Comment (including references)** | Security and state awareness check for control operations. |

* 1. test\_pan\_right\_device\_camera\_integration

| **Item** | **test\_pan\_right\_device\_camera\_integration (UT-CAM-PAN-RIGHT-INT)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | pan\_right() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling pan\_right() successfully executes the underlying virtual device interaction logic and returns a boolean result, confirming proper integration. |
| **Input Specifications** | Initial state: Camera is enabled and at 0.0 degrees. |
| **Detailed Step** | 1. Call pan\_right().  2. Assert the return value is a boolean (True) and the internal logic path for device interaction is executed. |
| **Expected Result** | The method must execute the device interaction path and return True. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Integration with the virtual device layer for pan control was confirmed. |
| **Comment (including references)** | Verifies the linkage between the domain model and the virtual device simulator. |

1. get\_password()
   1. test\_get\_password\_not\_set

| **Item** | **test\_get\_password\_not\_set (UT-CAM-PW-GET-DEF)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | get\_password() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when the camera password has not been initialized or set, get\_password() correctly returns an empty string (""). |
| **Input Specifications** | Initial state: No password set. |
| **Detailed Step** | 1. Instantiate the camera object (default state).  2. Call get\_password().  3. Assert the returned value is "". |
| **Expected Result** | get\_password() must return "". |
| **Actual Result (Pass/Fail/Exception)** | Pass: Empty string was correctly retrieved in the default state. |
| **Comment (including references)** | Basic getter functionality check for the default password state. |

* 1. test\_set\_password

| **Item** | **test\_set\_password (UT-CAM-PW-SET-GET)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_password(), get\_password() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that after a successful password setting (set\_password), get\_password() correctly returns the newly stored password. |
| **Input Specifications** | Action: Call set\_password("mypassword123"). |
| **Detailed Step** | 1. Call set\_password("mypassword123").  2. Assert the return value is True.  3. Call get\_password() and assert the value is "mypassword123". |
| **Expected Result** | get\_password() must return "mypassword123". |
| **Actual Result (Pass/Fail/Exception)** | Pass: Password was successfully stored and retrieved. |
| **Comment (including references)** | Verification of password storage and retrieval functionality. |

1. set\_password()
   1. test\_set\_password

| **Item** | **test\_set\_password (UT-CAM-PW-SET-PASS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_password(), has\_password(), get\_password() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling set\_password() with a valid password string returns True, updates the password internally, and sets the has\_password() flag to True. |
| **Input Specifications** | Action: Call set\_password("mypassword123"). |
| **Detailed Step** | 1. Call set\_password("mypassword123").  2. Assert the return value is True.  3. Assert has\_password() is True and get\_password() returns the set value. |
| **Expected Result** | set\_password() must return True, and the new password must be successfully stored. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Password was successfully set and validated. |
| **Comment (including references)** | Core function test for password storage and retrieval. |

* 1. test\_set\_empty\_password

| **Item** | **test\_set\_empty\_password (UT-CAM-PW-SET-EMPTY)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_password(), has\_password() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that setting an empty string ("") is rejected by set\_password(), which returns False and ensures the has\_password() state remains False. |
| **Input Specifications** | Action: Call set\_password(""). |
| **Detailed Step** | 1. Call set\_password("").  2. Assert the return value is False.  3. Assert has\_password() is False and get\_password() is "". |
| **Expected Result** | set\_password() must return **False**, and no password should be set. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Empty password setting was correctly rejected. |
| **Comment (including references)** | Input validation check for minimum password length/content. |

* 1. test\_set\_password\_special\_characters

| **Item** | **test\_set\_password\_special\_characters (UT-CAM-PW-SET-SPEC)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_password() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that the system can successfully accept and store passwords containing various special characters and non-standard characters, confirming character set compatibility. |
| **Input Specifications** | Action: Call set\_password with strings like "!@#$%", "암호123". |
| **Detailed Step** | 1. Call set\_password with a special character string.  2. Assert the return value is True.  3. Assert get\_password() returns the exact string. |
| **Expected Result** | set\_password() must return True, and the special character password must be saved correctly. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Special character passwords were successfully accepted. |
| **Comment (including references)** | Coverage test for character set compatibility in password fields. |

* 1. test\_set\_password\_long\_password

| **Item** | **test\_set\_password\_long\_password (UT-CAM-PW-SET-LONG)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_password() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the system's robustness by ensuring it can successfully accept and store a very long password (e.g., 1000 characters) without truncation or failure. |
| **Input Specifications** | Action: Call set\_password with a 1000-character long string. |
| **Detailed Step** | 1. Create a long string.  2. Call set\_password() with the long string.  3. Assert the return value is True and get\_password() returns the full, intact long string. |
| **Expected Result** | set\_password() must return True, and the long password must be saved correctly. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Long password was successfully stored and retrieved. |
| **Comment (including references)** | Stress and robustness test for maximum password length constraints. |

* 1. test\_set\_password\_whitespace\_only

| **Item** | **test\_set\_password\_whitespace\_only (UT-CAM-PW-SET-SPACE)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | set\_password(), get\_password() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies how the system handles passwords consisting only of whitespace characters, confirming the outcome (in this implementation, it is accepted as a valid, non-empty string). |
| **Input Specifications** | Action: Call set\_password(" ") (three spaces). |
| **Detailed Step** | 1. Call set\_password(" ").  2. Assert the return value is True.  3. Assert get\_password() returns the exact whitespace string. |
| **Expected Result** | set\_password() must return True, and the password must be saved as " ". |
| **Actual Result (Pass/Fail/Exception)** | Pass: Whitespace-only string was accepted and stored as a valid password. |
| **Comment (including references)** | Detailed input validation check for whitespace content. |

1. is\_enabled()
   1. test\_is\_enabled

| **Item** | **test\_is\_enabled (UT-CAM-STAT-FULL)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | is\_enabled(), disable(), enable() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that is\_enabled() accurately reflects the camera's status through a full operational cycle: checking the initial state, switching to disabled, and switching back to enabled. |
| **Input Specifications** | Initial state: Enabled. Actions: Call disable(), then call enable(). |
| **Detailed Step** | 1. Assert initial state is True.  2. Call disable() and assert state is False.  3. Call enable() and assert state is True. |
| **Expected Result** | The sequence of is\_enabled() returns must be True, then False, then True. |
| **Actual Result (Pass/Fail/Exception)** | Pass: The camera state was correctly tracked throughout the cycle. |
| **Comment (including references)** | Core functionality test for state retrieval and status consistency. |

* 1. test\_enable

| **Item** | **test\_enable (UT-CAM-STAT-EN)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | enable(), is\_enabled() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling enable() successfully transitions the camera to the active state, as confirmed by is\_enabled(). |
| **Input Specifications** | Initial state: Camera is disabled (via disable() call). Action: Call enable(). |
| **Detailed Step** | 1. Set the initial state to disabled.  2. Call enable().  3. Assert is\_enabled() returns True. |
| **Expected Result** | is\_enabled() must return True after enable() is called. |
| **Actual Result (Pass/Fail/Exception)** | Pass: The camera successfully transitioned to the enabled state. |
| **Comment (including references)** | Verification of the enable setter's result. |

* 1. test\_disable

| **Item** | **test\_disable (UT-CAM-STAT-DIS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | disable(), is\_enabled() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling disable() successfully transitions the camera to the inactive state, as confirmed by is\_enabled(). |
| **Input Specifications** | Initial state: Camera is enabled. Action: Call disable(). |
| **Detailed Step** | 1. Assert initial state is True.  2. Call disable().  3. Assert is\_enabled() returns False. |
| **Expected Result** | is\_enabled() must return False after disable() is called. |
| **Actual Result (Pass/Fail/Exception)** | Pass: The camera successfully transitioned to the disabled state. |
| **Comment (including references)** | Verification of the disable setter's result. |

* 1. test\_enable\_disable\_toggle

| **Item** | **test\_enable\_disable\_toggle (UT-CAM-STAT-TOG)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | is\_enabled(), enable(), disable() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the stability of the camera's enabled state by checking its status after multiple consecutive, redundant calls to enable() and disable(). |
| **Input Specifications** | Actions: Multiple calls to disable() (3x), followed by multiple calls to enable() (3x). |
| **Detailed Step** | 1. Call disable() multiple times. Assert state is False.  2. Call enable() multiple times. Assert final state is True. |
| **Expected Result** | The state should correctly switch and persist based on the last command, regardless of redundant calls. |
| **Actual Result (Pass/Fail/Exception)** | Pass: State toggling was stable across redundant commands. |
| **Comment (including references)** | Robustness check for state persistence across repeated setter calls. |

1. enable()
   1. test\_enable

| **Item** | **test\_enable (UT-CAM-EN-PASS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | enable(), is\_enabled() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling enable() successfully transitions the camera from the disabled state to the active state, as confirmed by is\_enabled(). |
| **Input Specifications** | Initial state: Camera is first disabled. Action: Call enable(). |
| **Detailed Step** | 1. Set the initial state to disabled.  2. Call enable().  3. Assert is\_enabled() returns True. |
| **Expected Result** | is\_enabled() must return True after enable() is called. |
| **Actual Result (Pass/Fail/Exception)** | Pass: The camera successfully transitioned to the enabled state. |
| **Comment (including references)** | Core function test for setting the camera to an active state. |

* 1. test\_enable\_disable\_toggle

| **Item** | **test\_enable\_disable\_toggle (UT-CAM-EN-TOG)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | enable(), is\_enabled() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the stability of the enabled state by checking the status after multiple consecutive, redundant calls to enable(). |
| **Input Specifications** | Initial state: Camera is disabled. Action: Multiple (3x) consecutive calls to enable(). |
| **Detailed Step** | 1. Set the initial state to disabled.  2. Call enable() multiple times.  3. Assert is\_enabled() returns True. |
| **Expected Result** | The state should consistently switch and persist as True, regardless of redundant calls. |
| **Actual Result (Pass/Fail/Exception)** | Pass: State toggling was stable, and the enabled state persisted. |
| **Comment (including references)** | Robustness check for state persistence across repeated enable() calls. |

1. disable()
   1. test\_disable

| **Item** | **test\_disable (UT-CAM-DIS-PASS)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | disable(), is\_enabled() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling disable() successfully transitions the camera from the active state to the inactive state, as confirmed by is\_enabled(). |
| **Input Specifications** | Initial state: Camera is enabled. Action: Call disable(). |
| **Detailed Step** | 1. Assert initial state is True.  2. Call disable().  3. Assert is\_enabled() returns False. |
| **Expected Result** | is\_enabled() must return False after disable() is called. |
| **Actual Result (Pass/Fail/Exception)** | Pass: The camera successfully transitioned to the disabled state. |
| **Comment (including references)** | Core function test for setting the camera to an inactive state. |

* 1. test\_display\_view\_disabled\_camera

| **Item** | **test\_display\_view\_disabled\_camera (UT-CAM-DIS-CONSEQ)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | disable(), display\_view() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the direct consequence of the disabled state: after calling disable(), attempting to access the camera view (display\_view()) is correctly denied, returning None. |
| **Input Specifications** | Action: Call disable().  Consequence Check: Call display\_view(). |
| **Detailed Step** | 1. Call disable() to change the state.  2. Call display\_view().  3. Assert the result is None. |
| **Expected Result** | display\_view() must return **None**, confirming view access is blocked in the disabled state. |
| **Actual Result (Pass/Fail/Exception)** | Pass: View access was correctly denied after disabling the camera. |
| **Comment (including references)** | Verification of the security consequence of the disabled state. |

* 1. test\_enable\_disable\_toggle

| **Item** | **test\_enable\_disable\_toggle (UT-CAM-DIS-TOG)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | disable(), is\_enabled() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the stability of the disabled state by checking the status after multiple consecutive, redundant calls to disable(). |
| **Input Specifications** | Initial state: Camera is enabled. Action: Multiple (3x) consecutive calls to disable(). |
| **Detailed Step** | 1. Call disable() multiple times.  2. Assert is\_enabled() returns False. |
| **Expected Result** | The state should consistently switch and persist as False, regardless of redundant calls. |
| **Actual Result (Pass/Fail/Exception)** | Pass: State toggling was stable, and the disabled state persisted. |
| **Comment (including references)** | Robustness check for state persistence across repeated disable() calls. |

1. has\_password()
   1. test\_has\_password

| **Item** | **test\_has\_password (UT-CAM-PW-STAT-FULL)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | has\_password(), set\_password() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that has\_password() accurately reflects the camera's password status: initially False, and switching to True after a valid password is set. |
| **Input Specifications** | Initial state: No password set. Action: Call set\_password("password123"). |
| **Detailed Step** | 1. Assert initial state is False.  2. Call set\_password("password123").  3. Assert has\_password() returns True. |
| **Expected Result** | The sequence of has\_password() returns must be False, then True. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Password status was correctly tracked through the lifecycle. |
| **Comment (including references)** | Core functionality test for password status retrieval. |

* 1. test\_set\_empty\_password

| **Item** | **test\_set\_empty\_password (UT-CAM-PW-STAT-INV)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | has\_password(), set\_password() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when an attempt to set an invalid password (empty string) is rejected, has\_password() correctly remains False, ensuring the status is not corrupted by failed operations. |
| **Input Specifications** | Initial state: No password set. Action: Call set\_password(""). |
| **Detailed Step** | 1. Assert initial state is False.  2. Call set\_password("").  3. Assert the return value is False.  4. Assert has\_password() remains False. |
| **Expected Result** | set\_password() must return False, and has\_password() must remain False. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Status was correctly preserved after the failed operation. |
| **Comment (including references)** | Validation check ensuring failed password setting attempts do not corrupt the has\_password flag. |

1. save\_info()
   1. test\_save\_info

| **Item** | **test\_save\_info (UT-CAM-SAVE)** |
| --- | --- |
| **Class** | SafeHomeCamera |
| **Method** | save\_info() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that the method is callable and returns True, confirming that the logic for triggering the camera's information persistence (saving its state) was successfully executed. |
| **Input Specifications** | None. |
| **Detailed Step** | 1. Call camera.save\_info().  2. Assert the return value is True. |
| **Expected Result** | save\_info() must return **True**. |
| **Actual Result (Pass/Fail/Exception)** | Pass: The data persistence interface call succeeded. |
| **Comment (including references)** | Verification of the interface method responsible for triggering data persistence (state saving) in the system. |

1. CameraController class
2. add\_camera()
   1. test\_add\_camera

| **Item** | **test\_add\_camera (UT-CC-ADD-SINGLE)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | add\_camera(x, y) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the successful addition of a single camera, ensuring the correct ID (1) is assigned, the total count increments, and all internal metadata (\_cameras, \_camera\_info) is updated with the location and enabled status. |
| **Input Specifications** | Initial state: 0 cameras. Action: Call add\_camera(100, 200). |
| **Detailed Step** | 1. Call add\_camera(100, 200).  2. Assert the return value is True.  3. Assert total count is 1 and the camera information is correctly registered with ID 1. |
| **Expected Result** | Returns True. \_total\_camera\_number is 1, and \_next\_camera\_id is 2. Camera info (100, 200, True) is stored. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Camera was successfully added and tracked. |
| **Comment (including references)** | Core function test for registering a new surveillance device. |

* 1. test\_add\_multiple\_cameras

| **Item** | **test\_add\_multiple\_cameras (UT-CC-ADD-MULTI)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | add\_camera(x, y) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that multiple sequential calls to add\_camera() correctly handle the ID sequencing and consistently update the total camera count and internal tracking. |
| **Input Specifications** | Action: Call add\_camera() three times with varying coordinates. |
| **Detailed Step** | 1. Call add\_camera() three times.  2. Assert that \_total\_camera\_number is 3.  3. Assert that \_next\_camera\_id is 4.  4. Verify coordinates for cameras 1, 2, and 3. |
| **Expected Result** | \_total\_camera\_number must be 3, and the next available ID must be 4. All three cameras should be registered. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Sequential camera additions were successful, and ID management was correct. |
| **Comment (including references)** | Verification of internal ID management and state tracking robustness under sequential operation. |

1. delete\_camera()
   1. test\_delete\_camera

| **Item** | **test\_delete\_camera (UT-CC-DEL-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | delete\_camera(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the successful deletion of an existing camera, ensuring the total count decreases, and the camera is removed from all internal collections (\_cameras, \_camera\_info). |
| **Input Specifications** | Initial state: Two cameras exist (IDs 1, 2). Action: Call delete\_camera(1). |
| **Detailed Step** | 1. Add two cameras.  2. Call delete\_camera(1).  3. Assert the return value is True.  4. Assert total count is 1 and ID 1 is no longer in the tracking lists. |
| **Expected Result** | Returns True. \_total\_camera\_number decreases by 1, and the camera ID is removed from all internal dictionaries. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Camera was successfully de-registered. |
| **Comment (including references)** | Core function test for camera de-registration and state update. |

* 1. test\_delete\_nonexistent\_camera

| **Item** | **test\_delete\_nonexistent\_camera (UT-CC-DEL-FAIL)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | delete\_camera(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling delete\_camera() with a non-existent ID fails, returns False, and ensures the controller's state remains unaltered. |
| **Input Specifications** | Initial state: 0 cameras. Action: Call delete\_camera(999). |
| **Detailed Step** | 1. Ensure no cameras exist.  2. Call delete\_camera(999).  3. Assert the return value is False.  4. Assert total count remains 0. |
| **Expected Result** | Returns False. The internal state (e.g., \_total\_camera\_number) must remain unchanged. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Deletion of a non-existent camera was correctly rejected. |
| **Comment (including references)** | Input validation and state preservation check for deletion logic. |

* 1. test\_delete\_camera\_with\_password

| **Item** | **test\_delete\_camera\_with\_password (UT-CC-DEL-PW)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | delete\_camera(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies data consistency: when a camera with a stored password is deleted, its entry is removed from the \_camera\_passwords collection alongside the camera object. |
| **Input Specifications** | Initial state: Camera ID 1 has a password set. Action: Call delete\_camera(1). |
| **Detailed Step** | 1. Add camera and set password.  2. Call delete\_camera(1).  3. Assert \_camera\_passwords no longer contains ID 1. |
| **Expected Result** | Returns True. ID 1 must be removed from the \_camera\_passwords dictionary. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Associated password data was correctly cleaned up upon camera deletion. |
| **Comment (including references)** | Data consistency and cleanup validation for linked attributes. |

* 1. test\_delete\_camera\_with\_device\_camera\_stop

| **Item** | **test\_delete\_camera\_with\_device\_camera\_stop (UT-CC-DEL-INT)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | delete\_camera(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies integration: ensures that when a camera is deleted, the underlying virtual device's stop() method is called once to correctly shut down the device simulator. |
| **Input Specifications** | Initial state: Camera 1 exists with a mockable device. Action: Call delete\_camera(1). |
| **Detailed Step** | 1. Add camera. Patch the underlying device's stop method.  2. Call delete\_camera(1).  3. Assert the mock stop method was called exactly once. |
| **Expected Result** | The underlying \_device\_camera.stop() method must be called once. Deletion returns True. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Device cleanup protocol was correctly initiated. |
| **Comment (including references)** | Verification of the internal device cleanup protocol upon de-registration. |

* 1. test\_delete\_camera\_handles\_stop\_exception

| **Item** | **test\_delete\_camera\_handles\_stop\_exception (UT-CC-DEL-ROB)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | delete\_camera(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies robustness: ensures that the deletion process completes successfully (returns True) and the controller's state is updated, even if the underlying device's stop() method throws an exception. |
| **Input Specifications** | Initial state: Camera 1 exists. Action: Mock the device's stop() method to raise an Exception. Call delete\_camera(1). |
| **Detailed Step** | 1. Add camera. Patch device stop() to raise an Exception.  2. Call delete\_camera(1).  3. Assert the return value is True.  4. Assert the camera is actually deleted (count decreased). |
| **Expected Result** | Returns True. The deletion must succeed despite the internal device error. |
| **Actual Result (Pass/Fail/Exception)** | Pass: The system recovered from the device exception and completed the deletion. |
| **Comment (including references)** | Robustness check for external component failure during cleanup. |

1. enable\_cameras()
   1. test\_enable\_cameras\_list

| **Item** | **test\_enable\_cameras\_list (UT-CC-EN-LIST-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | enable\_cameras(camera\_id\_list) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling enable\_cameras() with a list of valid camera IDs successfully switches all specified cameras to the enabled state. |
| **Input Specifications** | Initial state: Cameras 1 and 2 are disabled. Camera 3 is enabled. Action: Call enable\_cameras([1, 2]). |
| **Detailed Step** | 1. Add 3 cameras; disable IDs 1 and 2.  2. Call enable\_cameras([1, 2]).  3. Assert the return value is True.  4. Assert cameras 1, 2, and 3 are all enabled. |
| **Expected Result** | Returns True. Cameras 1 and 2 are successfully enabled, and the state of Camera 3 is preserved as enabled. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Bulk activation of cameras succeeded. |
| **Comment (including references)** | Core function test for bulk activation operation. |

* 1. test\_enable\_cameras\_partial\_failure

| **Item** | **test\_enable\_cameras\_partial\_failure (UT-CC-EN-LIST-ROB)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | enable\_cameras(camera\_id\_list) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the method's robustness by ensuring that it correctly processes and enables valid cameras even if the input list contains non-existent camera IDs, without crashing or failing the entire operation. |
| **Input Specifications** | Initial state: Cameras 1 and 2 exist. Action: Call enable\_cameras([1, 2, 999]) (ID 999 does not exist). |
| **Detailed Step** | 1. Add 2 cameras.  2. Call enable\_cameras([1, 2, 999]).  3. Assert that cameras 1 and 2 are successfully enabled. |
| **Expected Result** | Cameras 1 and 2 must be successfully enabled, ignoring the non-existent ID 999. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Robustness verified; valid operations completed despite invalid input in the list. |
| **Comment (including references)** | Robustness and error tolerance check for invalid inputs in bulk operations. |

1. disable\_cameras()
   1. test\_disable\_cameras\_list

| **Item** | **test\_disable\_cameras\_list (UT-CC-DIS-LIST-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | disable\_cameras(camera\_id\_list) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling disable\_cameras() with a list of valid camera IDs successfully switches all specified cameras to the disabled state. |
| **Input Specifications** | Initial state: Cameras 1 and 2 are enabled. Action: Call disable\_cameras([1, 2]). |
| **Detailed Step** | 1. Add 2 cameras; ensure both are enabled.  2. Call disable\_cameras([1, 2]).  3. Assert the return value is True.  4. Assert cameras 1 and 2 are disabled. |
| **Expected Result** | Returns True. Cameras 1 and 2 are successfully disabled. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Bulk deactivation of cameras succeeded. |
| **Comment (including references)** | Core function test for bulk deactivation operation. |

* 1. test\_disable\_cameras\_partial\_failure

| **Item** | **test\_disable\_cameras\_partial\_failure (UT-CC-DIS-LIST-ROB)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | disable\_cameras(camera\_id\_list) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the method's robustness by ensuring that it correctly processes and disables valid cameras even if the input list contains non-existent camera IDs, without failing the entire operation. |
| **Input Specifications** | Initial state: Cameras 1 and 2 exist and are enabled. Action: Call disable\_cameras([1, 2, 999]) (ID 999 does not exist). |
| **Detailed Step** | 1. Add 2 cameras; ensure both are enabled.  2. Call disable\_cameras([1, 2, 999]).  3. Assert that cameras 1 and 2 are successfully disabled. |
| **Expected Result** | Cameras 1 and 2 must be successfully disabled, ignoring the non-existent ID 999. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Robustness verified; valid operations completed despite invalid input in the list. |
| **Comment (including references)** | Robustness and error tolerance check for invalid inputs in bulk operations. |

1. enable\_all\_camera()
   1. test\_enable\_all\_cameras

| **Item** | **test\_enable\_all\_cameras (UT-CC-EN-ALL-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | enable\_all\_camera() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling enable\_all\_camera() successfully switches *all* registered cameras to the enabled state, regardless of their previous disabled state. |
| **Input Specifications** | Initial state: Two cameras exist, both are disabled. Action: Call enable\_all\_camera(). |
| **Detailed Step** | 1. Add 2 cameras; disable both.  2. Call enable\_all\_camera().  3. Assert the return value is True.  4. Assert both cameras are enabled. |
| **Expected Result** | Returns True. All registered cameras must be successfully enabled. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Global activation succeeded. |
| **Comment (including references)** | Core function test for bulk activation of all devices. |

* 1. test\_camera\_enable\_disable\_workflow

| **Item** | **test\_camera\_enable\_disable\_workflow (UT-CC-EN-ALL-WORK)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | enable\_all\_camera() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the operation of enable\_all\_camera() at the end of a complex state change workflow (including individual and bulk disabling), ensuring it correctly resets all cameras to the enabled state. |
| **Input Specifications** | Initial state: Three cameras exist. Action: Sequentially disable various cameras, followed by a final call to enable\_all\_camera(). |
| **Detailed Step** | 1. Add 3 cameras.  2. Perform mixed disable/enable operations.  3. Call disable\_all\_camera().  4. Call enable\_all\_camera().  5. Assert all 3 cameras are enabled. |
| **Expected Result** | All cameras must be confirmed to be in the enabled state (is\_enabled() is True). |
| **Actual Result (Pass/Fail/Exception)** | Pass: The final global enable operation correctly restored all camera states. |
| **Comment (including references)** | Integration and consistency check across multiple bulk state changes. |

1. disable\_all\_camera()
   1. test\_disable\_all\_cameras

| **Item** | **test\_disable\_all\_cameras (UT-CC-DIS-ALL-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | disable\_all\_camera() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling disable\_all\_camera() successfully switches *all* registered cameras to the disabled state, regardless of their previous enabled state. |
| **Input Specifications** | Initial state: Two cameras exist, both are enabled. Action: Call disable\_all\_camera(). |
| **Detailed Step** | 1. Add 2 cameras; ensure both are enabled.  2. Call disable\_all\_camera().  3. Assert the return value is True.  4. Assert both cameras are disabled. |
| **Expected Result** | Returns True. All registered cameras must be successfully disabled. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Global deactivation succeeded. |
| **Comment (including references)** | Core function test for bulk deactivation of all devices. |

* 1. test\_camera\_enable\_disable\_workflow

| **Item** | **test\_camera\_enable\_disable\_workflow (UT-CC-DIS-ALL-WORK)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | disable\_all\_camera() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the operation of disable\\_all\\_camera() within a complex state change workflow, ensuring it correctly switches all cameras to the disabled state after individual and bulk enabling/disabling operations. |
| **Input Specifications** | Initial state: Three cameras exist. Action: Sequentially disable/enable cameras, followed by a final call to disable\_all\_camera(). |
| **Detailed Step** | 1. Add 3 cameras.  2. Perform mixed enable/disable operations.  3. Call disable\_all\_camera().  4. Assert all 3 cameras are disabled. |
| **Expected Result** | All cameras must be confirmed to be in the disabled state (is\_enabled() is False). |
| **Actual Result (Pass/Fail/Exception)** | Pass: The final global disable operation correctly set all camera states to disabled. |
| **Comment (including references)** | Integration and consistency check across multiple bulk state changes. |

1. enable\_camera()
   1. test\_enable\_camera

| **Item** | **test\_enable\_camera (UT-CC-EN-SINGLE-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | enable\_camera(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling enable\_camera() with a valid ID successfully transitions the specified camera from the disabled state to the active state. |
| **Input Specifications** | Initial state: Camera ID 1 exists and is disabled. Action: Call enable\_camera(1). |
| **Detailed Step** | 1. Add camera and disable it.  2. Call enable\_camera(1).  3. Assert the return value is True.  4. Assert the camera's enabled state is True. |
| **Expected Result** | Returns True. The camera's enabled state must be True. |
| **Actual Result (Pass/Fail/Exception)** | Pass: The camera successfully transitioned to the enabled state. |
| **Comment (including references)** | Core function test for setting a single camera to an active state. |

* 1. test\_enable\_nonexistent\_camera

| **Item** | **test\_enable\_nonexistent\_camera (UT-CC-EN-SINGLE-FAIL)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | enable\_camera(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that attempting to call enable\_camera() using a non-existent ID fails, returns False, and does not alter the controller's state. |
| **Input Specifications** | Initial state: No camera with ID 999 exists. Action: Call enable\_camera(999). |
| **Detailed Step** | 1. Call enable\_camera(999).  2. Assert the return value is False. |
| **Expected Result** | Returns False. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Request for a nonexistent camera was correctly rejected. |
| **Comment (including references)** | Input validation check for non-existent camera IDs. |

1. disable\_camera()
   1. test\_disable\_camera

| **Item** | **test\_disable\_camera (UT-CC-DIS-SINGLE-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | disable\_camera(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling disable\_camera() with a valid ID successfully transitions the specified camera from the active state to the inactive state. |
| **Input Specifications** | Initial state: Camera ID 1 exists and is enabled. Action: Call disable\_camera(1). |
| **Detailed Step** | 1. Add camera and ensure it is enabled.  2. Call disable\_camera(1).  3. Assert the return value is True.  4. Assert the camera's enabled state is False. |
| **Expected Result** | Returns True. The camera's enabled state must be False. |
| **Actual Result (Pass/Fail/Exception)** | Pass: The camera successfully transitioned to the disabled state. |
| **Comment (including references)** | Core function test for setting a single camera to an inactive state. |

* 1. test\_disable\_nonexistent\_camera

| **Item** | **test\_disable\_nonexistent\_camera (UT-CC-DIS-SINGLE-FAIL)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | disable\_camera(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that attempting to call disable\_camera() using a non-existent ID fails, returns False, and does not alter the controller's state. |
| **Input Specifications** | Initial state: No camera with ID 999 exists. Action: Call disable\_camera(999). |
| **Detailed Step** | 1. Call disable\_camera(999).  2. Assert the return value is False. |
| **Expected Result** | Returns False. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Request for a nonexistent camera was correctly rejected. |
| **Comment (including references)** | Input validation check for non-existent camera IDs. |

1. control\_single\_camera()
   1. test\_control\_single\_camera\_take\_picture

| **Item** | **test\_control\_single\_camera\_take\_picture (UT-CC-CTRL-SNAP)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | control\_single\_camera(id, control\_id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that using control ID 0 successfully triggers the camera's snapshot (take picture) functionality. |
| **Input Specifications** | Initial state: Camera 1 is enabled. Action: Call control\_single\_camera(1, 0). |
| **Detailed Step** | 1. Add camera 1.  2. Call control\_single\_camera(1, 0).  3. Assert the return value is True. |
| **Expected Result** | Returns True. The snapshot command is correctly dispatched. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Snapshot command execution succeeded. |
| **Comment (including references)** | Dispatcher test for core image capture action. |

* 1. test\_control\_single\_camera\_start\_recording

| **Item** | **test\_control\_single\_camera\_start\_recording (UT-CC-CTRL-REC-START)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | control\_single\_camera(id, control\_id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that using control ID 1 successfully triggers the recording start functionality, and the camera's status transitions to "Recording". |
| **Input Specifications** | Initial state: Camera 1 is idle. Action: Call control\_single\_camera(1, 1). |
| **Detailed Step** | 1. Call control\_single\_camera(1, 1).  2. Assert the return value is True.  3. Assert camera status is "Recording". |
| **Expected Result** | Returns True. Camera status is updated to "Recording". |
| **Actual Result (Pass/Fail/Exception)** | Pass: Recording start command succeeded, and status updated. |
| **Comment (including references)** | Dispatcher test for core recording start action. |

* 1. test\_control\_single\_camera\_stop\_recording

| **Item** | **test\_control\_single\_camera\_stop\_recording (UT-CC-CTRL-REC-STOP)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | control\_single\_camera(id, control\_id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that using control ID 2 successfully triggers the recording stop functionality, and the camera's status returns to its default state ("idle"). |
| **Input Specifications** | Initial state: Camera 1 is recording. Action: Call control\_single\_camera(1, 2). |
| **Detailed Step** | 1. Start recording (ID 1).  2. Call control\_single\_camera(1, 2).  3. Assert the return value is True.  4. Assert camera status is "idle". |
| **Expected Result** | Returns True. Camera status transitions away from "Recording". |
| **Actual Result (Pass/Fail/Exception)** | Pass: Recording stop command succeeded. |
| **Comment (including references)** | Dispatcher test for core recording stop action. |

* 1. test\_control\_single\_camera\_pan\_left

| **Item** | **test\_control\_single\_camera\_pan\_left (UT-CC-CTRL-PAN)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | control\_single\_camera(id, control\_id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the dispatcher correctly executes a movement command (e.g., pan left, using ID 0 in the coverage test's dispatcher context) by checking the return value. |
| **Input Specifications** | Initial state: Camera 1 enabled. Action: Call control\_single\_camera(1, 0). |
| **Detailed Step** | 1. Add camera 1.  2. Call control\_single\_camera(1, 0) (Pan Left).  3. Assert the return value is True. |
| **Expected Result** | Returns True. The Pan Left command is successfully dispatched. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Pan left command successfully dispatched. |
| **Comment (including references)** | Dispatcher test for movement control functionality. |

* 1. test\_control\_disabled\_camera

| **Item** | **test\_control\_disabled\_camera (UT-CC-CTRL-FAIL-DIS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | control\_single\_camera(id, control\_id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that attempting to control a disabled camera (e.g., snapshot, ID 0) is rejected, returns False, and prevents the execution of the command. |
| **Input Specifications** | Initial state: Camera 1 is disabled. Action: Call control\_single\_camera(1, 0). |
| **Detailed Step** | 1. Disable camera 1.  2. Call control\_single\_camera(1, 0).  3. Assert the return value is False. |
| **Expected Result** | Returns False. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Control command was correctly rejected in the disabled state. |
| **Comment (including references)** | Security validation check for controlling inactive devices. |

* 1. test\_control\_nonexistent\_camera

| **Item** | **test\_control\_nonexistent\_camera (UT-CC-CTRL-FAIL-NON)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | control\_single\_camera(id, control\_id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that attempting to control a non-existent camera ID is rejected, returns False, and prevents the dispatcher from executing. |
| **Input Specifications** | Action: Call control\_single\_camera(999, 0) (ID 999 does not exist). |
| **Detailed Step** | 1. Call control\_single\_camera(999, 0).  2. Assert the return value is False. |
| **Expected Result** | Returns False. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Command was correctly rejected for a non-existent ID. |
| **Comment (including references)** | Input validation check for camera existence. |

* 1. test\_control\_invalid\_control\_id

| **Item** | **test\_control\_invalid\_control\_id (UT-CC-CTRL-FAIL-INV)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | control\_single\_camera(id, control\_id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that providing an undefined or invalid control\_id (e.g., 99) is rejected by the dispatcher, returning False. |
| **Input Specifications** | Initial state: Camera 1 exists. Action: Call control\_single\_camera(1, 99). |
| **Detailed Step** | 1. Add camera 1.  2. Call control\_single\_camera(1, 99).  3. Assert the return value is False. |
| **Expected Result** | Returns False. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Command was rejected due to an invalid control ID. |
| **Comment (including references)** | Dispatcher validation check for command ID range and definition. |

1. display\_thumnail\_view()
   1. test\_display\_thumbnail\_view\_no\_cameras

| **Item** | **test\_display\_thumbnail\_view\_no\_cameras (UT-CC-THUMB-EMPTY)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | display\_thumbnail\_view() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling display\_thumbnail\_view() when no cameras are registered gracefully handles the empty list edge case by returning None. |
| **Input Specifications** | Initial state: 0 cameras registered. Action: Call display\_thumbnail\_view(). |
| **Detailed Step** | 1. Ensure the controller is empty.  2. Call display\_thumbnail\_view().  3. Assert the return value is None. |
| **Expected Result** | Returns None. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Empty state was handled correctly. |
| **Comment (including references)** | Input validation check for the empty list edge case. |

* 1. test\_display\_thumbnail\_view\_with\_multiple\_cameras

| **Item** | **test\_display\_thumbnail\_view\_with\_multiple\_cameras (UT-CC-THUMB-MULTI)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | display\_thumbnail\_view() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling display\_thumbnail\_view() when multiple cameras exist successfully initiates the composite view generation, returning a valid image object or None if image files are unavailable. |
| **Input Specifications** | Initial state: 3 enabled cameras exist. Action: Call display\_thumbnail\_view(). |
| **Detailed Step** | 1. Add 3 cameras.  2. Call display\_thumbnail\_view().  3. Assert the result is None or an instance of PIL.Image.Image. |
| **Expected Result** | Returns a non-null object or None, confirming the operation succeeds without crashing. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Composite view generation succeeded. |
| **Comment (including references)** | Core functionality test for generating the composite thumbnail view. |

* 1. test\_display\_thumbnail\_view\_with\_mixed\_cameras

| **Item** | **test\_display\_thumbnail\_view\_with\_mixed\_cameras (UT-CC-THUMB-MIXED)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | display\_thumbnail\_view(), disable\_camera() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when a mix of enabled and disabled cameras exists, the thumbnail view correctly processes only the active cameras for composition. |
| **Input Specifications** | Initial state: 3 cameras exist; Camera 1 is disabled. Action: Call display\_thumbnail\_view(). |
| **Detailed Step** | 1. Add 3 cameras.  2. Disable Camera 1.  3. Call display\_thumbnail\_view().  4. Assert the operation succeeds (returns image or None), implying only enabled cameras were used. |
| **Expected Result** | Returns a non-null object or None, confirming the operation succeeds and correctly handles disabled camera filtering. |
| **Actual Result (Pass/Fail/Exception)** | Pass: State awareness verified; view composition succeeded with mixed camera states. |
| **Comment (including references)** | State awareness check for view composition (only enabled cameras should contribute). |

* 1. test\_display\_thumbnail\_view\_grid\_calculation

| **Item** | **test\_display\_thumbnail\_view\_grid\_calculation (UT-CC-THUMB-GRID)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | display\_thumbnail\_view() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that the internal logic correctly calculates the grid dimensions and processes the view generation for varying numbers of cameras (e.g., 1, 3, 4), ensuring the layout is handled correctly. |
| **Input Specifications** | Sequential actions adding 1, 3, and 4 cameras and calling display\_thumbnail\_view() after each addition. |
| **Detailed Step** | 1. Add 1 camera, call view.  2. Add 2 more (total 3), call view.  3. Add 1 more (total 4), call view.  4. Assert all calls succeed. |
| **Expected Result** | The function must execute successfully for varying camera counts without error, validating internal grid dimension calculations. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Grid calculation logic passed for varying camera counts. |
| **Comment (including references)** | Coverage test for internal thumbnail layout/grid dimension calculations. |

1. display\_single\_view()
   1. test\_display\_single\_view

| **Item** | **test\_display\_single\_view (UT-CC-VIEW-SINGLE-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | display\_single\_view(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling display\_single\_view() with a valid ID successfully retrieves the view from the specified, enabled camera. |
| **Input Specifications** | Initial state: Camera 1 exists and is enabled. Action: Call display\_single\_view(1). |
| **Detailed Step** | 1. Add camera 1.  2. Call display\_single\_view(1).  3. Assert the result is a non-null image object or None if image files are unavailable. |
| **Expected Result** | Returns a valid PIL.Image.Image object or None, confirming the operation succeeds. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Single view retrieval succeeded. |
| **Comment (including references)** | Core functionality test for retrieving a single camera feed. |

* 1. test\_display\_single\_view\_disabled\_camera

| **Item** | **test\_display\_single\_view\_disabled\_camera (UT-CC-VIEW-SINGLE-FAIL-DIS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | display\_single\_view(id), disable\_camera() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that attempting to display the view of a disabled camera is rejected, returning None, thus enforcing the security policy. |
| **Input Specifications** | Initial state: Camera 1 is disabled. Action: Call display\_single\_view(1). |
| **Detailed Step** | 1. Disable camera 1.  2. Call display\_single\_view(1).  3. Assert the result is None. |
| **Expected Result** | Returns **None**. |
| **Actual Result (Pass/Fail/Exception)** | Pass: View access was correctly denied due to the disabled state. |
| **Comment (including references)** | Security validation check for view access control in a disabled state. |

* 1. test\_display\_single\_view\_nonexistent\_camera

| **Item** | **test\_display\_single\_view\_nonexistent\_camera (UT-CC-VIEW-SINGLE-FAIL-NON)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | display\_single\_view(id) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that attempting to display the view of a non-existent camera ID is correctly rejected, returning None. |
| **Input Specifications** | Action: Call display\_single\_view(999) (ID 999 does not exist). |
| **Detailed Step** | 1. Ensure no camera with ID 999 exists.  2. Call display\_single\_view(999).  3. Assert the result is None. |
| **Expected Result** | Returns **None**. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Request for a nonexistent camera was correctly rejected. |
| **Comment (including references)** | Input validation check for non-existent camera IDs. |

1. get\_all\_camera\_info()
   1. test\_get\_all\_camera\_info\_empty

| **Item** | **test\_get\_all\_camera\_info\_empty (UT-CC-INFO-EMPTY)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | get\_all\_camera\_info() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that calling get\_all\_camera\_info() when no cameras are registered gracefully handles the empty controller edge case by returning an empty list ([]). |
| **Input Specifications** | Initial state: 0 cameras registered. Action: Call get\_all\_camera\_info(). |
| **Detailed Step** | 1. Ensure the controller is empty.  2. Call get\_all\_camera\_info().  3. Assert the returned value is []. |
| **Expected Result** | Returns []. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Empty state was handled correctly. |
| **Comment (including references)** | Input validation check for the empty controller edge case. |

* 1. test\_get\_all\_camera\_info

| **Item** | **test\_get\_all\_camera\_info (UT-CC-INFO-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | get\_all\_camera\_info() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that the method correctly retrieves the info list for all registered cameras, including their ID, location, and enabled status. |
| **Input Specifications** | Initial state: 2 cameras exist (IDs 1, 2). Action: Call get\_all\_camera\_info(). |
| **Detailed Step** | 1. Add 2 cameras.  2. Call get\_all\_camera\_info().  3. Assert the list length is 2 and check the coordinates and enabled status (1) of the returned info. |
| **Expected Result** | Returns a list of length 2, where each element contains the correct [ID, X, Y, Enabled Status]. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Global status information was correctly retrieved. |
| **Comment (including references)** | Core functionality test for retrieving global camera status. |

* 1. test\_get\_all\_camera\_info\_sorted\_order

| **Item** | **test\_get\_all\_camera\_info\_sorted\_order (UT-CC-INFO-ORDER)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | get\_all\_camera\_info() |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that the camera information list is returned in a predictable, sorted order based on the Camera ID, ensuring reliable data display sequence. |
| **Input Specifications** | Initial state: 3 cameras added, possibly in non-sequential order if IDs were deleted/reused (though IDs are sequential here). Action: Call get\_all\_camera\_info(). |
| **Detailed Step** | 1. Add 3 cameras.  2. Call get\_all\_camera\_info().  3. Extract the IDs from the returned list and assert they are in numerical ascending order. |
| **Expected Result** | The list of camera IDs in the output must be sorted numerically (e.g., [1, 2, 3]). |
| **Actual Result (Pass/Fail/Exception)** | Pass: Data ordering was confirmed to be reliable. |
| **Comment (including references)** | Integrity check for reliable data display ordering (sorting by ID). |

1. set\_camera\_password()
   1. test\_set\_camera\_password

| **Item** | **test\_set\_camera\_password (UT-CC-PW-SET-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | set\_camera\_password(id, input\_password) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that a valid password is successfully set for an existing camera, correctly updating the internal \_camera\_passwords collection. |
| **Input Specifications** | Initial state: Camera 1 exists. Action: Call set\_camera\_password(1, "password123"). |
| **Detailed Step** | 1. Add camera 1.  2. Call set\_camera\_password(1, "password123").  3. Assert that ID 1 is present in \_camera\_passwords with the correct value. |
| **Expected Result** | ID 1 must be present in \_camera\_passwords with the value "password123". |
| **Actual Result (Pass/Fail/Exception)** | Pass: Password was successfully assigned to the camera. |
| **Comment (including references)** | Core function test for password assignment and internal data consistency. |

* 1. test\_set\_password\_nonexistent\_camera

| **Item** | **test\_set\_password\_nonexistent\_camera (UT-CC-PW-SET-FAIL-NON)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | set\_camera\_password(id, input\_password) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that attempting to set a password for a non-existent camera ID is rejected, ensuring no unintended entries are created in the internal collection. |
| **Input Specifications** | Initial state: 0 cameras exist. Action: Call set\_camera\_password(999, "password123"). |
| **Detailed Step** | 1. Call set\_camera\_password(999, "password123").  2. Assert that ID 999 is *not* present in \_camera\_passwords. |
| **Expected Result** | ID 999 must not be found in the \_camera\_passwords dictionary. |
| **Actual Result (Pass/Fail/Exception)** | Pass: Password setting for a non-existent camera was correctly ignored. |
| **Comment (including references)** | Input validation check for camera existence prior to password assignment. |

* 1. test\_set\_camera\_password\_nonexistent\_camera\_handling

| **Item** | **test\_set\_camera\_password\_nonexistent\_camera\_handling (UT-CC-PW-SET-ROB)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | set\_camera\_password(id, input\_password) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the system's robustness by ensuring that attempting to set a password for a non-existent camera ID does not raise an unhandled exception, confirming graceful failure. |
| **Input Specifications** | Action: Call set\_camera\_password(999, "password123") and monitor for exceptions. |
| **Detailed Step** | 1. Call set\_camera\_password(999, "password123") within a try/except block (or implicitly check for raised exceptions).  2. Assert that the process completes without raising an Exception. |
| **Expected Result** | The method must complete successfully (without crashing). |
| **Actual Result (Pass/Fail/Exception)** | Pass: Robustness verified; the process handled the invalid ID gracefully. |
| **Comment (including references)** | Coverage check for robust handling of invalid IDs during password assignment. |

1. validate\_camera\_password()
   1. test\_validate\_camera\_password\_success

| **Item** | **test\_validate\_camera\_password\_success (UT-CC-PW-VAL-PASS)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | validate\_camera\_password(id, input\_password) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when the correct password is provided for a set camera, the function returns the success code (0). |
| **Input Specifications** | Initial state: Camera 1 password set to "password123". Action: Call validate\_camera\_password(1, "password123"). |
| **Detailed Step** | 1. Set password for camera 1.  2. Call validate\_camera\_password(1, "password123").  3. Assert the returned code is 0. |
| **Expected Result** | Returns **0** (Success). |
| **Actual Result (Pass/Fail/Exception)** | Pass: Correct password validation succeeded. |
| **Comment (including references)** | Core function test for password validation success path. |

* 1. test\_validate\_camera\_password\_failure

| **Item** | **test\_validate\_camera\_password\_failure (UT-CC-PW-VAL-FAIL)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | validate\_camera\_password(id, input\_password) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that when an incorrect password is provided for a set camera, the function returns the password mismatch code (1). |
| **Input Specifications** | Initial state: Camera 1 password set to "password123". Action: Call validate\_camera\_password(1, "wrongpassword"). |
| **Detailed Step** | 1. Set password for camera 1.  2. Call validate\_camera\_password(1, "wrongpassword").  3. Assert the returned code is 1. |
| **Expected Result** | Returns **1** (Mismatch/Failure). |
| **Actual Result (Pass/Fail/Exception)** | Pass: Incorrect password validation failed as expected. |
| **Comment (including references)** | Validation test for the password mismatch path. |

* 1. test\_validate\_camera\_password\_nonexistent\_camera

| **Item** | **test\_validate\_camera\_password\_nonexistent\_camera (UT-CC-PW-VAL-NON)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | validate\_camera\_password(id, input\_password) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that attempting to validate a password for a non-existent camera ID returns the camera not found code (-1). |
| **Input Specifications** | Initial state: Camera 999 does not exist. Action: Call validate\_camera\_password(999, "any"). |
| **Detailed Step** | 1. Call validate\_camera\_password(999, "password123").  2. Assert the returned code is -1. |
| **Expected Result** | Returns **-1** (Camera Not Found). |
| **Actual Result (Pass/Fail/Exception)** | Pass: Correct error code was returned for the nonexistent ID. |
| **Comment (including references)** | Security check for handling validation attempts on nonexistent resources. |

* 1. test\_validate\_camera\_password\_not\_set

| **Item** | **test\_validate\_camera\_password\_not\_set (UT-CC-PW-VAL-UNSET)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | validate\_camera\_password(id, input\_password) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies that validating a camera that exists but has no password set returns the password not set code (-2). |
| **Input Specifications** | Initial state: Camera 1 exists but has **no** password set. Action: Call validate\_camera\_password(1, "any"). |
| **Detailed Step** | 1. Add camera 1.  2. Call validate\_camera\_password(1, "password123").  3. Assert the returned code is -2. |
| **Expected Result** | Returns **-2** (Password Not Set). |
| **Actual Result (Pass/Fail/Exception)** | Pass: Correct error code was returned for the unset password state. |
| **Comment (including references)** | Validation check for the initial state of the password attribute. |

* 1. test\_validate\_camera\_password\_edge\_cases

| **Item** | **test\_validate\_camera\_password\_edge\_cases (UT-CC-PW-VAL-EDGE)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | validate\_camera\_password(id, input\_password) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies the security logic when an empty string ("") is provided as the input password for a camera that has a password set (should fail with mismatch code 1). |
| **Input Specifications** | Initial state: Camera 1 password set to "test". Action: Call validate\_camera\_password(1, ""). |
| **Detailed Step** | 1. Set password for camera 1 to "test".  2. Call validate\_camera\_password(1, "").  3. Assert the returned code is 1. |
| **Expected Result** | Returns **1** (Mismatch/Failure). |
| **Actual Result (Pass/Fail/Exception)** | Pass: Empty input was correctly handled as a failed validation attempt. |
| **Comment (including references)** | Coverage test for security logic on empty string input against a set password. |

1. delete\_camera\_password()
   1. test\_delete\_camera\_with\_password

| **Item** | **test\_delete\_camera\_with\_password (UT-CC-PW-DEL-IMPL)** |
| --- | --- |
| **Class** | CameraController |
| **Method** | delete\_camera\_password() (Implicitly verified by delete\_camera()) |
| **Author** | Jien Lee |
| **Date** | 2025.11.23 |
| **Version** | 1.0.1 |
| **Test Case Description** | Verifies data consistency: when a camera with a stored password is deleted, the underlying password removal logic is executed, ensuring the entry is removed from the \_camera\_passwords collection. |
| **Input Specifications** | Initial state: Camera ID 1 has a password set. Action: Call delete\_camera(1). |
| **Detailed Step** | 1. Add camera and set password.  2. Call delete\_camera(1).  3. Assert the password entry for ID 1 is no longer present in \_camera\_passwords. |
| **Expected Result** | Camera deletion must return True, and the associated password data must be successfully removed. |
|  |  |
| **Actual Result (Pass/Fail/Exception)** | Pass: Associated password data was correctly cleaned up upon camera deletion. |
| **Comment (including references)** | Verification of internal data cleanup and consistency, confirming the password removal logic is executed as part of the de-registration process. |

### 3. **Security**

#### **A. Sensor Management**

**1. SecuritySystem**

**A. handle\_sensor\_event() (processing input signals)**

**a. test\_entry\_delay\_disarm\_prevents\_alarm (UT-Sec-EntryDelay-Disarm)**

| Field | Value |

| Class, Method, Author, Date, Version | SecuritySystem, handle\_sensor\_event/disarm, Team 9 UC2, 2025-11-25, 1.0 |

| Test Case Description | Verifies that if an entry delay is triggered by a sensor, disarming the system before the deadline prevents the siren. |

| Input Specifications | 1. Arm system. 2. Trigger Door Sensor (starts delay). 3. Call disarm() before tick count reaches deadline. |

| Expected Result | Alarm state returns to IDLE; no siren or monitoring call is made. |

| Actual Result | Pass: State returns to IDLE, no siren/monitoring call. |

| Comment | Traces SRS UC2d alarm handling and SDS Alarm state diagram branch (VI.2d step 5b). |

**b. test\_home\_mode\_motion\_ignored (UT-Sec-Home-Motion)**

| Field | Value |

| Class, Method, Author, Date, Version | SecuritySystem, \_is\_sensor\_armed, Team 9 UC2, 2025-11-25, 1.0 |

| Test Case Description | Verifies that in STAY (Home) mode, motion sensors are ignored while perimeter sensors remain active. |

| Input Specifications | Set mode to STAY; trigger a Motion Sensor event. |

| Expected Result | Motion event is ignored; system state remains IDLE. |

| Actual Result | Pass: Motion ignored, state remains IDLE. |

| Comment | Ties to SRS UC2b/UC2c selective sensor activation, referencing SDS VI.2a alt flow. |

**c. test\_zone\_specific\_arming\_blocks\_other\_zones (UT-Sec-Zone-Filter)**

| Field | Value |

| Class, Method, Author, Date, Version | SecuritySystem, handle\_sensor\_event, Team 9 UC2, 2025-11-25, 1.0 |

| Test Case Description | Verifies that arming a specific zone (Zone A) results in events from Zone B being ignored. |

| Input Specifications | Arm Zone A. Trigger sensor in Zone B. Trigger sensor in Zone A. |

| Expected Result | Zone B event ignored. Zone A event triggers entry delay/alarm. |

| Actual Result | Pass: Off-zone event ignored, on-zone event triggers delay. |

| Comment | Covers SRS UC2c selective zones and SDS VI.2c diagram for zone arming. |

#### **B. Alarm Processing**

**1. SecuritySystem**

**A. tick() (time-based escalation)**

**a. test\_entry\_delay\_expires\_triggers\_alarm (UT-Sec-Delay-Expire)**

| Field | Value |

| Class, Method, Author, Date, Version | SecuritySystem, tick, Team 9 UC2, 2025-11-25, 1.0 |

| Test Case Description | Verifies that if the entry delay expires without a disarm command, the system escalates to a full alarm. |

| Input Specifications | Trigger entry delay. Call tick() repeatedly until now > deadline. |

| Expected Result | Siren activates; monitoring service is called with INTRUSION\_ALARM. |

| Actual Result | Pass: Siren active, monitoring called. |

| Comment | Validates SDS VI.2d alarm escalation and lec20 requirement for asserting observable effects. |

**B. trigger\_panic()**

**a. test\_trigger\_panic\_from\_disarmed (UT-Sec-Panic)**

| Field | Value |

| Class, Method, Author, Date, Version | SecuritySystem, trigger\_panic, Team 9 UC2, 2025-11-25, 1.0 |

| Test Case Description | Verifies that the panic function triggers an immediate alarm even if the system is currently DISARMED. |

| Input Specifications | Ensure system is DISARMED. Call trigger\_panic(). |

| Expected Result | Alarm instantly ACTIVE; monitoring message PANIC sent. |

| Actual Result | Pass: Alarm instantly active, monitoring message PANIC. |

| Comment | Ties to SRS UC2h panic use case and SDS VI.2k monitor call. |

### **4. Configuration and Data Management**

#### **A. Log Management**

**1. AuthService**

**A. login/logout (audit logging)**

**a. test\_auth\_service\_logs\_success\_and\_failure (UT-Config-Audit)**

| Field | Value |

| Class, Method, Author, Date, Version | AuthService, login/logout, Team 9 UC1, 2025-11-25, 1.0 |

| Test Case Description | Verifies that authentication attempts and logouts are correctly written to the system audit log. |

| Input Specifications | Call login with valid creds, call login with invalid creds, call logout. |

| Expected Result | Success writes LOGIN\_SUCCESS; failure writes LOGIN\_FAILED; logout emits LOGOUT. |

| Actual Result | Pass: Logs appear exactly as expected. |

| Comment | Validates SDS Common controller responsibilities for logging and SRS Common preconditions. |

#### **B. SafeHome Mode Management**

**1. SecuritySystem**

**A. arm() (mode transition)**

**a. test\_readiness\_check\_blocks\_arming\_when\_door\_open (UT-Config-Readiness)**

| Field | Value |

| Class, Method, Author, Date, Version | SecuritySystem, \_check\_doors\_and\_windows\_closed, Team 9 UC2, 2025-11-25, 1.0 |

| Test Case Description | Verifies that the system performs a readiness check and blocks arming if a perimeter sensor is open. |

| Input Specifications | Set a door sensor to Open. Call arm(mode=AWAY). |

| Expected Result | RuntimeError raised; System mode remains DISARMED. |

| Actual Result | Pass: RuntimeError blocks arming, mode remains DISARMED. |

| Comment | Implements readiness check per SRS UC2b and SDS SensorController CRC responsibilities. |

#### **C. Safety Zone Management**

**1. ConfigurationManager**

**A. assign\_sensor\_to\_zone()**

**a. test\_security\_system\_tracks\_explicit\_zone\_assignments (UT-Config-Zone)**

| Field | Value |

| Class, Method, Author, Date, Version | ConfigurationManager, assign\_sensor\_to\_zone, Team 9 UC2, 2025-11-25, 1.0 |

| Test Case Description | Verifies that adding zones and assigning sensors persists correctly and is reflected in the SecuritySystem. |

| Input Specifications | Add zone; assign sensor to zone; trigger \_apply\_security\_configuration. |

| Expected Result | SecuritySystem sensor map mirrors the database assignments. |

| Actual Result | Pass: SecuritySystem sensor map mirrors DB. |

| Comment | Proves SDS VI.2f (configure zone) + VI.2g (delete zone) flows plus SRS UC2e zone CRUD. |

### **5. Branch Coverage**

*Target: Achieve highest branch coverage possible. (Average > 75%)*

#### **A. File Level**

| Files in SafeHome System | Statements | Miss | Coverage |

| domain/system.py | 330 | 71 | 78% |

| auth/login\_manager.py | 184 | 40 | 78% |

| domain/services/auth\_service.py | 24 | 1 | 96% |

| domain/services/settings\_service.py | 32 | 1 | 97% |

| security/security\_system.py | 193 | 29 | 85% |

| domain/system\_controller.py | 75 | 16 | 79% |

| config/configuration\_manager.py | 188 | 34 | 82% |

| storage/storage\_manager.py | 145 | 31 | 79% |

| surveillance/camera\_controller.py | 210 | 20 | 90% |

| surveillance/safehome\_camera.py | 141 | 11 | 92% |

| main.py | 686 | 207 | 70% |

| ui/main\_window.py | 698 | 614 | 12% |

| Average | 242 | 90 | ~78% |

#### **B. Class Level**

*Selected classes from critical modules.*

| Classes in security\_system.py / camera\_controller.py | Statements | Cover | Missing | Coverage |

| SecuritySystem (UC2 Core) | 193 | 164 | 29 | 85% |

| CameraController (UC3 Core) | 210 | 190 | 20 | 90% |

| SystemController (Orchestrator) | 75 | 59 | 16 | 79% |

| ConfigurationManager (Data) | 188 | 154 | 34 | 82% |

| AuthService (UC1 Core) | 24 | 23 | 1 | 96% |

#### **C. Method Level**

*Hot-spot analysis for critical security methods.*

| Methods in SecuritySystem | Branch | Cover | Missing | Coverage |

| arm | 12 | 10 | 2 | 83% |

| disarm | 8 | 8 | 0 | 100% |

| handle\_sensor\_event | 24 | 20 | 4 | 83% |

| tick | 10 | 9 | 1 | 90% |

| trigger\_panic | 4 | 4 | 0 | 100% |

| \_is\_sensor\_armed | 16 | 16 | 0 | 100% |

| \_check\_doors\_closed | 6 | 6 | 0 | 100% |

| Average | 11 | 10 | 1 | 93% |

## **IV. Integration Tests**

### **A. Controller Arm and Alarm Cycle (IT-UC2-ControllerArmAlarm)**

| Field | Value |

| Class, Function, Author, Date | SystemController, SecuritySystem, Team 9 UC2, 2025-11-25 |

| Test Case Description | Validate that the SystemController correctly orchestrates the full security cycle: Arming -> Sensor Trigger -> Entry Delay -> Alarm Activation. |

| Input Specifications | 1. Authenticate Controller. 2. Set mode AWAY. 3. Emit Door Open event. 4. Advance clock past delay. |

| Detailed Step | 1. Initialize SystemController with registered sensors. 2. Call set\_security\_mode(AWAY). 3. Simulate sensor hardware event (Door). 4. Verify SecuritySystem enters ENTRY\_DELAY. 5. Wait/Tick until delay expires. |

| Expected Result | SecuritySystem transitions: DISARMED → ENTRY\_DELAY → ALARM\_ACTIVE. Siren activates. Monitoring service logs INTRUSION\_ALARM. |

| Actual Result | Pass: SecuritySystem transitions DISARMED→ENTRY\_DELAY→ALARM\_ACTIVE. Siren active. Monitoring logs INTRUSION\_ALARM. |

| Comment | Traceability: SRS UC2a/UC2d, SDS Alarm state diagram and sequence VI.2d. |

### **B. Zone Persistence Integration (IT-UC2-ZonePersistence)**

| Field | Value |

| Class, Function, Author, Date | ConfigurationManager, StorageManager, Team 9 UC2, 2025-11-25 |

| Test Case Description | Validate that zone creations and sensor assignments are persisted in the database and correctly loaded into the runtime Security System. |

| Input Specifications | Fresh SQLite DB. New Zone "Entryway". Sensor "Front Door". |

| Detailed Step | 1. Initialize config. 2. Add zone “Entryway”. 3. Assign “Front Door” sensor to zone. 4. Trigger \_apply\_security\_configuration. 5. Remove assignment. |

| Expected Result | SecuritySystem internal sensor map reflects the new zone mapping, then reflects the removal. |

| Actual Result | Pass: SecuritySystem \_sensor\_zones gains then loses mapping, mirroring DB rows. |

| Comment | SRS UC2e zone CRUD, SDS class diagram around configuration manager. |

### **C. Camera Lifecycle (IT-UC3-CameraLifecycle)**

| Field | Value |

| Class, Function, Author, Date | CameraController, devices.camera.Camera, Team 9 UC3, 2025-11-24 |

| Test Case Description | Full lifecycle integration of surveillance devices: Add, Password Set, Disable, Control, Delete. |

| Input Specifications | Camera ID 1, coordinates (0,0), Password "secret". |

| Detailed Step | 1. Add camera. 2. Set password. 3. Disable camera. 4. Attempt control (should fail/block). 5. Enable and Delete. |

| Expected Result | Controller reflects state transitions. Password validation works. Disabled camera blocks view. |

| Actual Result | Pass: Every action returns True/Success, view gating honors enable flag. |

| Comment | Ties to SRS UC3a/b/c/f/g and SDS surveillance class diagram. |

## **V. System Level Tests**

### **A. Log onto the system through control panel (SLTUC2ARM)**

**1. Log onto the system through control panel success (ST-Login-Using-CP-Su)**

| Field | Value |

| Class, Function, Author, Date | ControlPanel, System, Team 9, 2025-11-25 |

| Test Case Description | End-user arms HOME mode via Tk Zones + Modes view, verifying readiness check per SRS UC2a. |

| Input Specifications | python main.py, Admin Credentials, Sensor Simulator input. |

| Detailed Step | 1. Launch python main.py, log into Tk control panel. 2. Ensure all simulated doors closed. 3. Navigate to Modes view and press Stay. 4. Trigger “Front Door” open from Zones simulator. 5. Disarm via Modes view. |

| Expected Result | Status indicator turns red with “STAY” text. Entry delay banner counts down. Monitoring log adds ENTRY\_DELAY entry, then returns to green after disarm. |

| Actual Result | Pass: Status turned red/STAY. Banner counted down. Logs confirmed. |

| Comment | SRS UC2a/UC2d, SDS VI.2a & Alarm state diagram. |

### **B. Panic Trigger via Web Dashboard (SLTUC2PANIC)**

**1. Remote Panic Activation (ST-Web-Panic)**

| Field | Value |

| Class, Function, Author, Date | Flask, SecuritySystem, Team 9, 2025-11-25 |

| Test Case Description | Validate panic button from browser while system disarmed, ensuring monitoring service call (SRS UC2h). |

| Input Specifications | Web Browser at /dashboard. Panic Button click. |

| Detailed Step | 1. Start Flask app, browse to /dashboard. 2. Press Panic button. 3. Observe UI toast. 4. Confirm siren log in Tk Monitoring view. |

| Expected Result | Dashboard shows “Panic alarm dispatched” toast. Tk Monitoring log records PANIC + timestamp. SecuritySystem log buffer shows action PANIC\_TRIGGERED. |

| Actual Result | Pass: Toast appeared. Monitoring log recorded PANIC. |

| Comment | SRS UC2h, SDS VI.2k sequence; manual log attached to QA runbook. |

## **VI. Who did What**

* **Sunwoo Han – Security & Camera Gateway Lead:** Implemented SecuritySystem entry-delay/panic/monitoring logic and the UC2→UC3 camera gateway so alarm events drive surveillance capture per SDS Alarm & Surveillance sequences.
* **Grace Min – Persistence & Configuration:** Refactored ConfigurationManager, hardened StorageManager schema (sensor FK cascades), and fed UC1 settings presenters/services with shared system-setting DTOs.
* **Leo Park – UI/API Lead:** Modernized Tk Login/Settings/Modes/Monitoring/Surveillance panels and Flask endpoints for /api/security/\*, /configure, and /api/cameras/\*, keeping SDS Common, Security, and Surveillance UI diagrams in sync.
* **Ethan Ortiz – System & Integration Architect:** Owned SystemController, System power/reset orchestration, Tk sensor simulation bridge, and REST-to-domain wiring across UC1–UC3.
* **Priya Patel – QA/Test Lead:** Authored pytest suites across domain/security/surveillance modules, curated UC1/UC2/UC3 SLT scripts, and generated the coverage/traceability evidence cited in this report.

## **VII. Meeting Logs**

| # | Date | Attendees | Key Decisions |

| 1 | 2025-09-15 | Mira Song (PM), Sunwoo Han, Grace Min, Leo Park, Ethan Ortiz, Priya Patel | Locked alarm delay default to 30s, aligned with SRS UC2a. |

| 2 | 2025-09-18 | Same as above | Agreed to reuse SQLite with FK cascades for zones per SDS II.5. |

| 3 | 2025-09-22 | Han, Min, Park, Ortiz, Patel | Scoped SensorController responsibilities into SystemController adapter. |

| 4 | 2025-09-25 | Han, Min, Park, Ortiz | Defined REST endpoints mirroring Tk commands (SDS VI.2b). |

| 5 | 2025-09-29 | Han, Min, Ortiz, Patel | Prioritized readiness-check negative tests (lec20). |

| 6 | 2025-10-02 | Song, Han, Min | Finalized schema changes for sensor\_zone\_assignments. |

| 7 | 2025-10-06 | Han, Park, Ortiz | Reviewed Tk Modes view wireframes; tied to SDS page hierarchy. |

| 8 | 2025-10-09 | Han, Min, Patel | Established pytest fixture strategy for isolated SQLite DB. |

| 9 | 2025-10-13 | Han, Park, Ortiz, Patel | Synced on entry delay UX (countdown banner) referencing SDS state diagram. |

| 10 | 2025-10-16 | Song, Han, Min, Park | Approved REST JSON schema for /api/security/status. |

| 11 | 2025-10-20 | Han, Ortiz, Patel | Walked through Alarm Condition Encountered flow; mapped to tests. |

| 12 | 2025-10-23 | Han, Min | Confirmed DeviceManager API for sensor lookup. |

| 13 | 2025-10-27 | Han, Park, Ortiz | Validated Tk–Flask sync requirements (two-way status updates). |

| 14 | 2025-10-30 | Han, Patel | Reviewed lec20 guidelines, set documentation expectations. |

| 15 | 2025-11-03 | Song, Han, Min, Park | Planned manual SLT coverage for UI gaps. |

| 16 | 2025-11-06 | Han, Min, Ortiz | Completed surveillance gateway hooks for alarms. |

| 17 | 2025-11-09 | Han, Park, Patel | Dry-ran panic flow end-to-end; recorded evidence needs. |

| 18 | 2025-11-12 | Han, Min, Ortiz, Patel | Assessed coverage report; targeted methods needing more assertions. |

| 19 | 2025-11-15 | Song, Han, Park | Verified dashboard toast copy vs. SRS verbiage. |

| 20 | 2025-11-18 | Han, Min, Ortiz | Tackled bug where reassigning sensors failed to de-arm ghosts. |

| 21 | 2025-11-20 | Han, Patel | Agreed to keep manual Tk validation documented in SLT logs. |

| 22 | 2025-11-22 | Han, Min, Park, Ortiz | Final regression rehearsal; noted UI automation backlog. |

| 23 | 2025-11-24 | Full UC2 squad | Signed off UC2 delivery, froze code for documentation/export. |

| 24 | 2025-11-25 | Grace Min, Leo Park, Priya Patel | Reviewed UC1 configure-settings implementation details (phone validation + Flask endpoints) per docs/CONFIGURE\_SETTINGS\_IMPLEMENTATION.md. |

| 25 | 2025-11-26 | Han, Ortiz, Patel | Verified UC1 turn-on/turn-off/reset SLTs and cross-checked logs referenced in the Common Function test docs. |

| 26 | 2025-11-27 | Han, Park, Ortiz, Patel | Ran UC3 camera dashboard walkthrough (add/pan/zoom/password) captured in docs/INTEGRATION\_AND\_SYSTEM\_TESTS.md §3 prior to final export. |

## **VIII. Appendix**

### **Operational Commands**

# Database setup / refresh (creates tables & defaults per StorageManager)  
cd SafeHome\_team9  
python - <<'PY'  
from config.configuration\_manager import ConfigurationManager  
cm = ConfigurationManager()  
cm.initialize\_configuration()  
PY  
  
# UC1 domain/presenter/service tests  
pytest tests/domain/test\_services.py tests/domain/test\_system\_power\_flow.py \  
 tests/domain/test\_login\_presenter.py tests/domain/test\_settings\_presenter.py \  
 tests/domain/test\_reset\_presenter.py tests/domain/test\_change\_password\_presenter.py \  
 tests/domain/test\_web\_login\_presenter.py tests/domain/test\_web\_settings\_presenter.py  
  
# UC2 domain + integration tests  
pytest security/test\_security\_system.py  
pytest tests/test\_zone\_assignments.py tests/test\_controller\_security\_flow.py  
  
# UC3 surveillance tests  
pytest tests/surveillance\_tests/test\_camera\_controller.py \  
 tests/surveillance\_tests/test\_safehome\_camera.py \  
 tests/surveillance\_tests/test\_surveillance\_integration.py  
  
# Launch combined Tk + Flask app (spawns Tk control panel and REST dashboard)  
python main.py