RIPHAH INTERNATIONAL UNIVERSITY GG CAMPUS

Software Testing Report: Weather App

Submitted to: Ma'am Shummaila Iqbal

Course: Software Quality Engineering

Submitted by:

Hanzla Alvi – 47583
Department - BSSE
47583@studets.riphah.edu.pk

Department of Computing

Date: 23rd May, 2025

Contents

Weather App

1. Introduction:

This smart weather app gives you everything you need to know about current and upcoming weather conditions in the simplest way possible. Just enter any city name, and it instantly shows you accurate weather information including temperature (which you can see in both Celsius and Fahrenheit), humidity levels, wind speed, visibility, and atmospheric pressure. The app automatically detects whether it's sunny, cloudy, rainy, or stormy, and displays beautiful matching icons so you can understand the weather at just a glance.

One of the best features is the 5-day forecast that helps you plan your week ahead. You'll see how the temperature changes day by day, whether rain is expected, and even the exact times for sunrise and sunset. The app can also warn you about extreme weather conditions like heat waves, heavy rains, or strong winds through instant alerts.

For those who prefer listening, there's a voice report feature that speaks the current weather conditions aloud in clear English. You can even email the full weather report to your friends or family with just one click. The interface is super user-friendly with options to switch between light and dark themes, making it comfortable to use any time of day. All your frequently checked locations are saved automatically, so you don't have to search them again and again.

Unlike complicated weather websites, WeatherVision keeps things simple while still providing all the important details you need to dress appropriately, plan outdoor activities, or just decide whether to carry an umbrella. It's like having a personal weather assistant right on your computer

2. Features:

Feature	**Description**
Real-Time Weather	Shows current temperature (°C/°F), humidity, wind speed, and conditions (sunny/rainy)
5-Day Forecast	Scrollable daily predictions with high/low temps and weather icons
Voice Reports	Speaks weather aloud in English (temperature + conditions)
Weather Alerts	Notifies about extreme conditions (heatwaves, storms, heavy rain)
Sunrise/Sunset	Displays exact day/night transition times
Light/Dark Mode	Toggle between white (day) and black (night) themes
Email Reports	Sends full weather details via email (Gmail/Outlook)
Temperature Graph	Weekly trend chart showing rising/falling temperatures
Saved Locations	Remembers frequently searched cities for quick access
Unit Converter	Switch between Celsius (°C) and Fahrenheit (°F) instantly

Offline Access	Shows last fetched data without internet
No Ads	Clean interface without pop-ups or banners
Data Backup & Export	Saves location in a table and can be exported to device in csv file.

3. Software Requirements

F.E-1. Real-Time Weather:

SRS.1.1 System shall display live temperature, humidity, wind speed, pressure, and visibility within 3 seconds of API fetch.
SRS.1.2 All numerical values shall update dynamically when unit conversion is triggered (°C/°F for temperature, m/s/kmph for wind speed).

F.E-2. 5-Day Forecast:

SRS.2.1 The system shall display forecast data for 5 consecutive days with date, temperature range, and weather condition icons.
SRS.2.2 The forecast panel shall support horizontal scrolling when content exceeds container width.

F.E-3. Voice Reports:

SRS.3.1 The system shall generate audible weather reports in English when the voice button is clicked.
SRS.3.2 Audio output shall include city name, temperature, humidity, and wind speed.

F.E-4. Weather Alerts:

SRS.4.1 System shall trigger alerts for Temperature (<5°C or >35°C).
SRS.4.2 System shall trigger alerts for Wind >10 m/s.
SRS.4.2 System shall trigger alerts for Thunderstorm/Extreme conditions.
SRS.4.4 – Alerts shall display as desktop notifications with 10s timeout.
SRS.4.5 Users shall view all alerts in a consolidated dialog.

F.E-5. Sunrise/Sunset:

SRS.5.1 The system shall display accurate sunrise/sunset times in HH:MM format, calculated from the city's geographical coordinates.

F.E-6. Light/Dark Mode:

SRS.6.1 The system shall provide theme toggling with smooth transitions between color schemes with all UI elements (text, cards, buttons) within 0.5 seconds.
SRS.6.2 Weather icons shall invert colors in dark mode.

F.E-7. Email Reports:

SRS.7.1 System shall generate emails with City name, country, Current weather metrics, Timestamp in ISO format.

SRS.7.2 The system shall compose HTML emails containing current weather data when the email report feature is activated.

SRS.7.3 Users shall input recipient email and subject before sending.

F.E-8. Temperature Graph:

SRS.8.1 The system shall plot 30-day temperature trends using matplotlib.
SRS.8.2 The system shall Graph window include title, labeled axes, and grid lines.

F.E-9. Saved Location:

SRS.9.1 The system shall store locations in a file (i.e. saved_locations.json) that should be a JSON file with case-insensitive deduplication.

F.E-10. Unit Convertor:

SRS.10.1 The system shall instantly convert the temperature from Celsius °C to Fahrenheit °F within 1 second of user request.
SRS.10.2 The system shall instantly convert the Wind from meter per second (m/s) to kilometer per hour (km/h) within 1 second of user request.

F.E-11. Offline Access:

SRS.11.1 The system shall display cached data from weather_history.csv with an "Offline Mode" watermark.
SRS.11.2 Cached data shall show its last update timestamp.

F.E-12. No Ads

SRS.12.1 The application shall not contain any advertising SDKs, pop-ups, or sponsored content.

F.E-13. Data Backup & Export

SRS.13.1 The system shall automatically create ZIP backups (saved locations + weather history) and export data to Excel/CSV with timestamped filenames.

SRS.13.2 Users shall view historical data in a scrollable table.

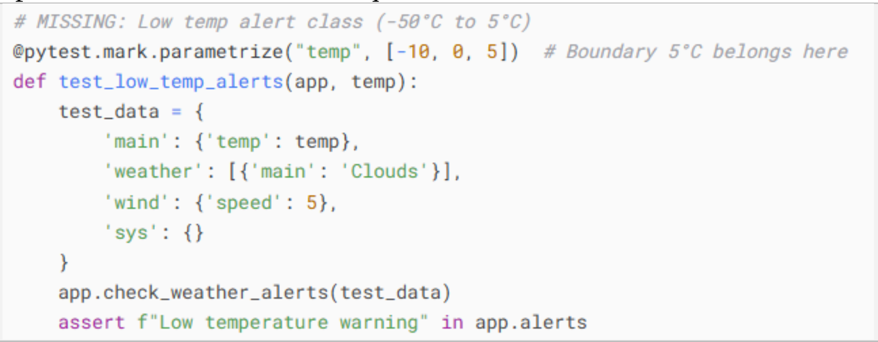
4. Equivalence Partitioning (ECP) for Weather Alerts:

Feature: Weather Alert System

Purpose: Automatically detect and notify users about extreme weather conditions (temperature, wind speed, severe weather) to ensure safety and preparedness.

Input Condition	Valid Equivalence Classes	**Invalid Equivalence** Classes
Temperature	1. -50°C to 5°C (Low temp alert)	1. Below-50°C(Invalid)
	2. 5°C to 35°C (Normal)	2. Non-numeric values
	3. 35°C to 60°C (High temp alert)	3. Missing temperature data
Wind Speed	1. 0-10 m/s (Normal)	1. Negative values
	2. >10 m/s (High wind alert)	2. Non-numeric values
Weather Condition	1. Clear/Clouds (Normal)	1. Empty array
	2. Thunderstorm/Extreme (Alert)	2. Invalid condition types
Data Completeness	1. All required fields present	1. Missing mandatory fields
Temperature Unit	1. "metric" (Celsius)	1.Unsupported unit strings
	2. "imperial" (Fahrenheit)	2. Null/undefined

Bug Report:

Bug ID	**ECP_WeatherAlerts_001**
Tester	Quality Assurance Team
Date (submitted)	2023-05-12
Title	WEATHER ALERT SYSTEM - Incomplete ECP Test Coverage
Bug Description	
URL	ECP.py
Summary	The Equivalence Partitioning tests for weather alerts do not fully cover all specified valid and invalid equivalence classes.
Screenshot	

```python
# MISSING: Low temp alert class (-50°C to 5°C)
@pytest.mark.parametrize("temp", [-10, 0, 5])  # Boundary 5°C belongs here
def test_low_temp_alerts(app, temp):
    test_data = {
        'main': {'temp': temp},
        'weather': [{'main': 'Clouds'}],
        'wind': {'speed': 5},
        'sys': {}
    }
    app.check_weather_alerts(test_data)
    assert f"Low temperature warning" in app.alerts
```

Platform	Python 3.12.5
Browser	File://weatherapp

Administrative	
Assigned To	Hanzla Alvi
Assigned At	2025-05-15
Priority	High
Severity	Medium

5. Boundary Value Analysis (BVA) for Weather Alerts

1. Temperature Boundary Tests

Low Temperature Alert (5°C Boundary)

Test Case	Input Data (**temp**)	Expected Alert	Reason
test_temp_4.9	4.9°C	True	4.9 < 5 ⟶ Alert
test_temp_5.0	5.0°C	False	5.0 is **not** < 5
test_temp_5.1	5.1°C	False	Normal range

High Temperature Alert (35°C Boundary)

Test Case	Input Data (**temp**)	Expected Alert	Reason
test_temp_34.9	34.9°C	False	Normal range
test_temp_35.0	35.0°C	False	35.0 is **not** > 35
test_temp_35.1	35.1°C	True	35.1> 35 ⟶ Alert

2. Wind Speed Boundary Tests (10 m/s)

Test Case	Input Data (**wind_speed**)	Expected Alert	Reason
test_wind_9.9	9.9 m/s	False	Normal range
test_wind_10.0	10.0 m/s	False	10.0 is **not** > 10
test_wind_10.1	10.1 m/s	True	10.1 > 10 ⟶ Alert

3. Weather Condition Tests

Test Case	Input Data (**weather**)	Expected Alert	Reason
test_weather_thunderstorm	"Thunderstorm"	True	Thunderstorm ⟶Alert
test_weather_clear	"Clear"	False	No alert
test_weather_empty	[]	Error	IndexError (no [0])

Bug Report:

Field	**Details**
Bug ID	BVA-WeatherAlerts-001

Tester	Hanzla Alvi
Date (Submitted)	2025-05-12
Title	BVA Implementation Missing Proper Boundary Handling in Weather Alerts
Bug Description	check_weather_alerts() function does not correctly handle **exact boundary values** for temperature and wind speed. Edge inputs cause alerts to misfire or go untriggered.
URL / File	src/gui.py (inside WeatherApp class – check_weather_alerts() method)
Summary	Current BVA tests miss verifying exact thresholds (like 5.0°C, 35.0°C, 10.0 m/s). Also, the app doesn't handle edge cases like empty weather data and negative wind speed.
Screenshot	if temp < 5 or temp > 35: *(Boundary values like 5.0 or 35.0 are not included in alerts)*
	condition = data['weather'][0]['main'] *(Crashes if weather list is empty)*
	wind_speed = data['wind']['speed'] *(No check for negative wind speed)*
Platform	Python 3.12.5
Browser	file://weatherApp
Assigned To	Hanzla Alvi
Assigned At	2025-05-15
Priority	High
Severity	Medium

6. Test cases:

Test Case ID	Test Case Scenario	Test Steps	Test Data	Expected Output	Actual Output	Test Status
TC-01	No alerts for normal conditions	1. Create test data2. Call check_weather_alerts()	{ 'main': {'temp': 20}, 'weather': [{'main': 'Clear'}], 'wind': {'speed': 5} }	No alerts generated	No alerts generated	PASS
TC-02	High temperature alert (>35°C)	1. Create test data2. Call check_weather_alerts()	{ 'main': {'temp': 36}, 'weather': [{'main': 'Clear'}], 'wind': {'speed': 5} }	"High temperature warning: 36°C"	"High temperature warning: 36°C"	PASS
TC-03	Low temperature alert (<5°C)	1. Create test data2. Call check_weather_alerts()	{ 'main': {'temp': 4}, 'weather': [{'main': 'Clear'}], 'wind': {'speed': 5} }	"Low temperature warning: 4°C"	"Low temperature warning: 4°C"	PASS
TC-04	Thunderstorm alert	1. Create test data2. Call check_weather_alerts()	{ 'main': {'temp': 20}, 'weather': [{'main': 'Thunderstorm'}], 'wind': {'speed': 5} }	"Weather alert: Thunderstorm"	"Weather alert: Thunderstorm"	PASS
TC-05	Extreme weather alert	1. Create test data2. Call check_weather_alerts()	{ 'main': {'temp': 20}, 'weather': [{'main': 'Extreme'}], 'wind': {'speed': 5} }	"Weather alert: Extreme"	"Weather alert: Extreme"	PASS
TC-06	High wind alert (>10 m/s)	1. Create test data2. Call check_weather_alerts()	{ 'main': {'temp': 20}, 'weather': [{'main': 'Clear'}], 'wind': {'speed': 11} }	"High wind warning: 11 m/s"	"High wind warning: 11 m/s"	PASS
TC-07	Multiple simultaneous alerts	1. Create test data2. Call check_weather_alerts()	{ 'main': {'temp': 36}, 'weather': [{'main': 'Thunderstorm'}], 'wind': {'speed': 11} }	3 alerts: temp, thunderstorm, wind	3 alerts: temp, thunderstorm, wind	PASS

TC-08	Alerts button style changes	1. Create test data2. Call check_weather_alerts()3. Check button	{ 'main': {'temp': 36}, 'weather': [{'main': 'Clear'}], 'wind': {'speed': 5} }	Button style = 'Warning.TButton'	Button style = 'Warning.TButton'	PASS
TC-09	No alerts button style	1. Create test data2. Call check_weather_alerts()3. Check button	{ 'main': {'temp': 20}, 'weather': [{'main': 'Clear'}], 'wind': {'speed': 5} }	Button style = 'TButton'	Button style = 'TButton'	PASS
TC-10	Notification generated	1. Mock notification2. Create test data3. Call check_weather_alerts()	{ 'main': {'temp': 36}, 'weather': [{'main': 'Clear'}], 'wind': {'speed': 5} }	Notification sent once	Notification sent once	PASS
TC-11	Incomplete weather data (missing 'main')	1. Create test data2. Call check_weather_alerts()	{ 'weather': [{'main': 'Clear'}], 'wind': {'speed': 5}, 'sys': {} }	No alerts generated, no crash	No alerts generated	PASS
TC-12	Unexpected weather condition	1. Create test data with unknown condition2. Call check_weather_alerts()	{ 'main': {'temp': 20}, 'weather': [{'main': 'Alien Invasion'}], 'wind': {'speed': 5}, 'sys': {} }	No alerts generated	No alerts generated	PASS

7. Usecase Diagram:

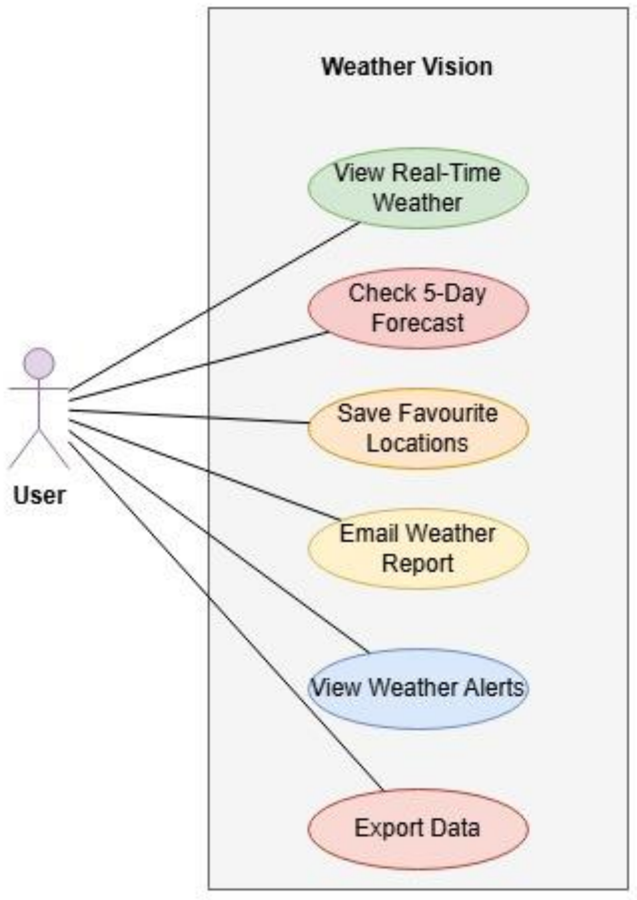


Figure 1 Weather Vision Use Case Diagram

8. Use Case Fully Dress for "Receive Weather Alerts"

Section	**Content**
Designation	UC-07
Name	Receive Weather Alerts
Authors	Hanzla Alvi
Priority	High
Criticality	Medium
Source	System detects extreme weather conditions

Responsible	WeatherAPI, Notification System
Description	System monitors weather conditions and alerts user when extreme conditions are detected
Trigger Event	Weather data update with extreme conditions
Actors	User, System
Pre-condition	1. Weather data is available 2. Notification system is operational
Post-condition	User is notified of extreme weather conditions
Result	User is aware of potential weather hazards
Main Scenario	1. System checks weather data 2. Detects extreme conditions 3. Generates alert notification 4. Updates alerts button 5. User views alerts when clicked
Alternative Scenario	1a. No extreme conditions detected - continue monitoring
Exception Scenario	3a. Notification fails - log error and retry
Qualities	Reliability, Timeliness, Visibility

Bug Report:

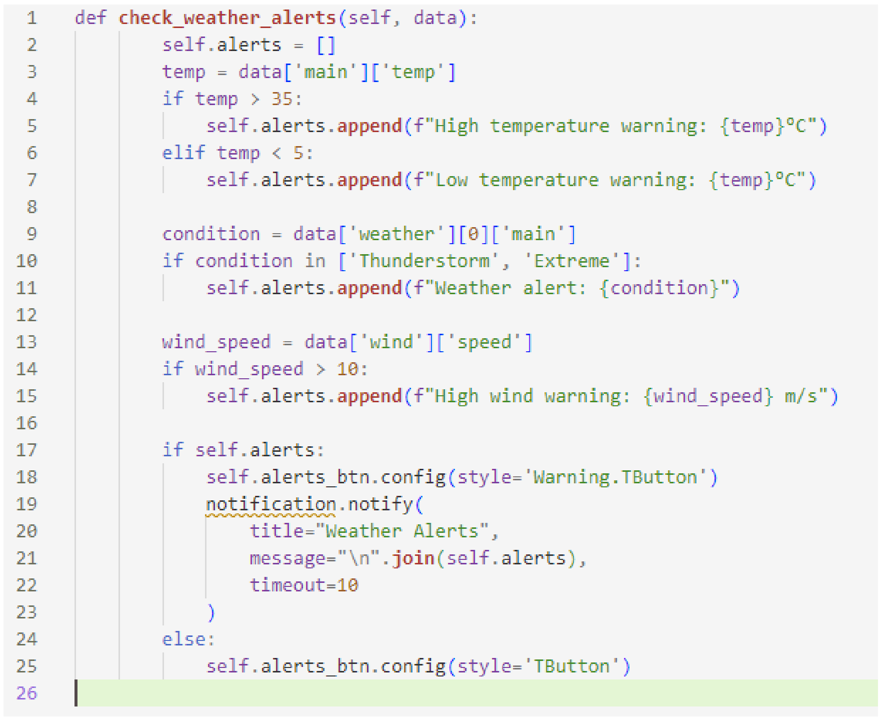
Field	**Details**
Bug ID	UC07-AlertFailure-001
Tester	Hanzla Alvi
Date (Submitted)	2025-05-23
Title	USE CASE UC-07: Weather Alerts Not Triggering on Boundary or Invalid Input
Bug Description	The weather alert system fails to trigger notifications when weather conditions are at the exact boundary values or contain invalid data such as empty weather list or negative wind speed.
Related Use Case	UC-07 – Receive Weather Alerts
Module	Notification System, WeatherAPI
Summary	Use Case defines timely notification to user, but system fails to alert at 5.0°C, 35.0°C, 10.0 m/s, and on missing/invalid input, which contradicts the post-condition and main scenario.
Source	System detects extreme weather conditions
Responsible	WeatherAPI, Notification System

Trigger Event	Weather data update with extreme conditions
Expected Result	User receives clear and timely alert if extreme weather is detected
Actual Result	No alert is triggered at boundary values; app crashes on empty weather[]; negative wind speed is accepted silently
Steps to Reproduce	1. Send weather data with temp = 5.0, 35.0, or wind = 10.0 2. Send data with empty weather[] or wind speed = -5

Screenshot Code	if temp < 5 or temp > 35:
	condition = data['weather'][0]['main']
	wind_speed = data['wind']['speed']

Platform	Python 3.12.5
Browser	file://weatherapp
Priority	High
Severity	Medium
Post-condition Failure	User is **not** notified of extreme weather due to logic issues
Exception Failure	Notification failure not handled properly when weather data is incomplete
Qualities Affected	Reliability, Timeliness, Visibility

9. White Box Testing:



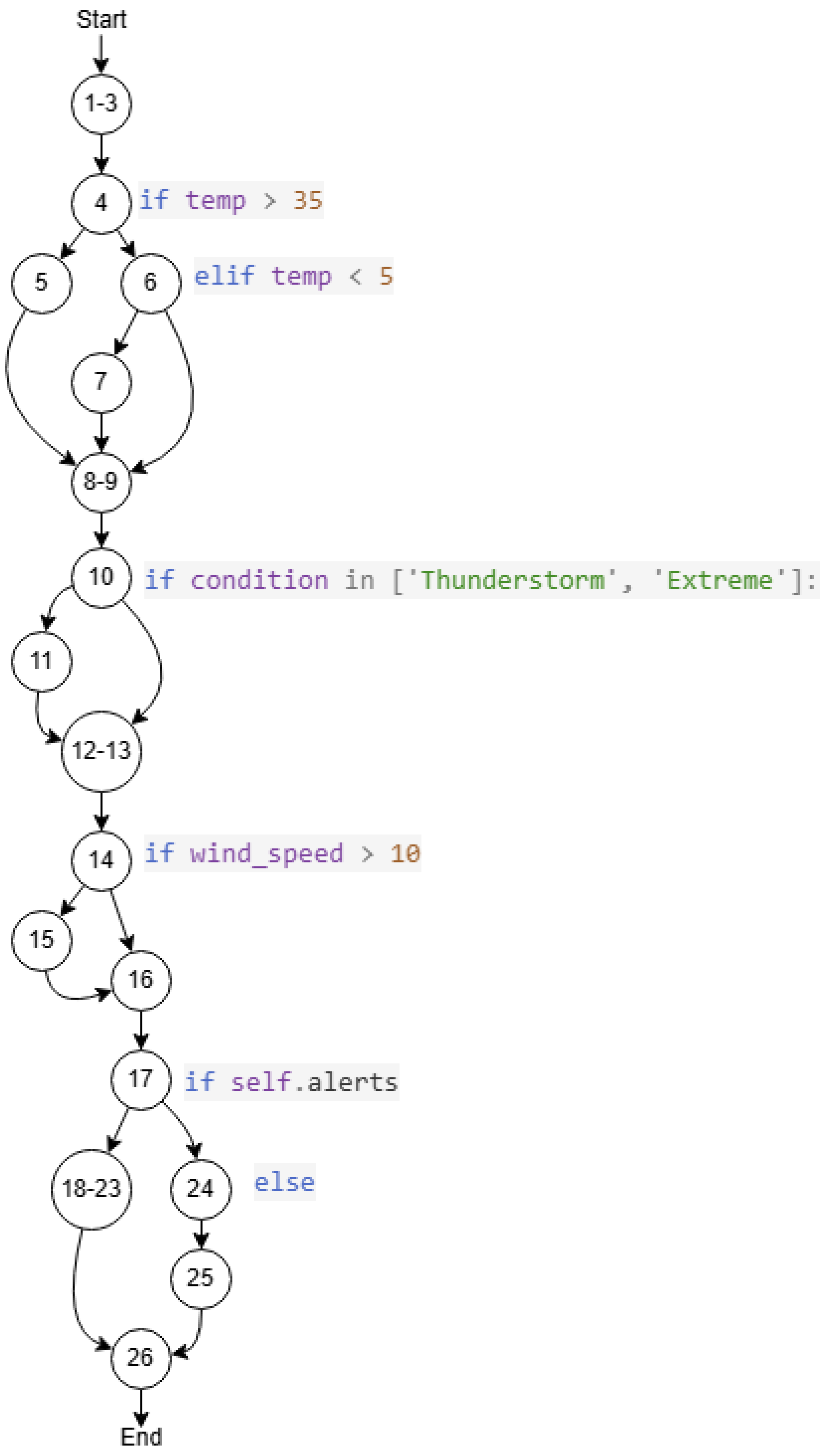
```python
def check_weather_alerts(self, data):
        self.alerts = []
        temp = data['main']['temp']
        if temp > 35:
            self.alerts.append(f"High temperature warning: {temp}°C")
        elif temp < 5:
            self.alerts.append(f"Low temperature warning: {temp}°C")

        condition = data['weather'][0]['main']
        if condition in ['Thunderstorm', 'Extreme']:
            self.alerts.append(f"Weather alert: {condition}")

        wind_speed = data['wind']['speed']
        if wind_speed > 10:
            self.alerts.append(f"High wind warning: {wind_speed} m/s")

        if self.alerts:
            self.alerts_btn.config(style='Warning.TButton')
            notification.notify(
                title="Weather Alerts",
                message="\n".join(self.alerts),
                timeout=10
            )
        else:
            self.alerts_btn.config(style='TButton')
```

1. **CFG:**


Start
1-3
4
if temp > 35
5
6
elif temp < 5
7
8-9
10
if condition in ['Thunderstorm', 'Extreme']:
11
12-13
14
if wind_speed > 10
15
16
17
if self.alerts
18-23
24
else
25
26
End

2. Perform decision coverage

Test ID	Test Data	Steps	Expected Outputs	Actual Outputs
T-1	1,2,3,4,5,8,9,10,12,13,14,16,17,18,19,20,21,22,23,26	Temp = 36, Condition = “Clear”, Wind=5	High temp alert only	High temp alert only
T-2	1,2,3,4,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,26	Temp = 4, Condition = “Thunderstorm”, Wind = 11	Low temp + weather alert	Low temp + weather alert + wind alert
T-3	1,2,3,4,6,8,9,10,12,13,14,16,17,24,25,26	Temp = 10, Condition = “Clear”, Wind=5	No alerts	No alerts

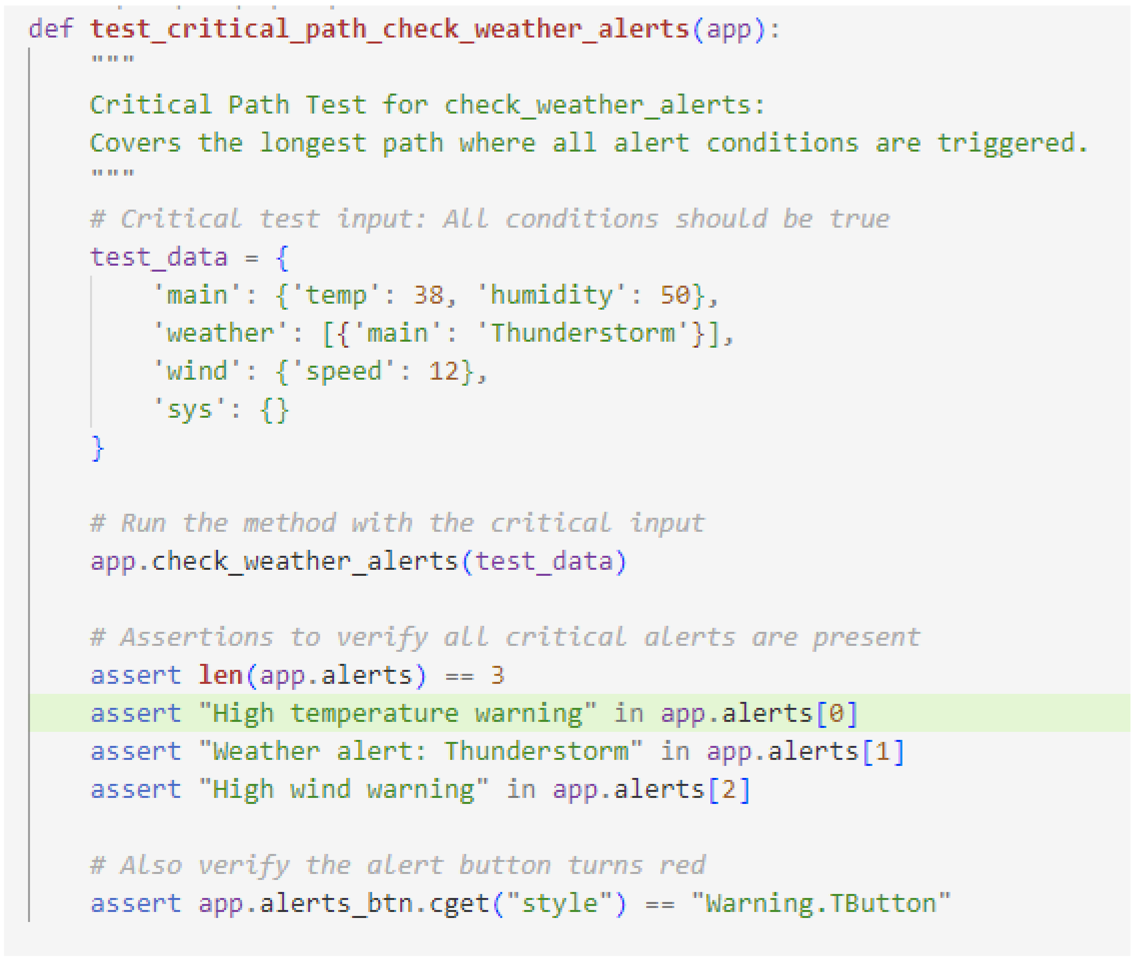
Statement Coverage: (Number of statements Exercised / Total number of statements) *100

Statement Coverage: (26 / 26) *100% = 100%

3. Critical path coverage

Step	Condition Checked	Input Value	Result	Action Taken
1	temp > 35	temp = 38	True	Add "High temperature warning"
2	temp < 5 (skipped due to el-if)	temp = 38	False	No action
3	condition in ['Thunderstorm', 'Extreme']	condition = Thunderstorm	True	Add "Weather alert"
4	wind_speed > 10	wind_speed = 12	True	Add "High wind warning"
5	if self.alerts	3 alerts in list	True	Style button red + show notification

Test data:



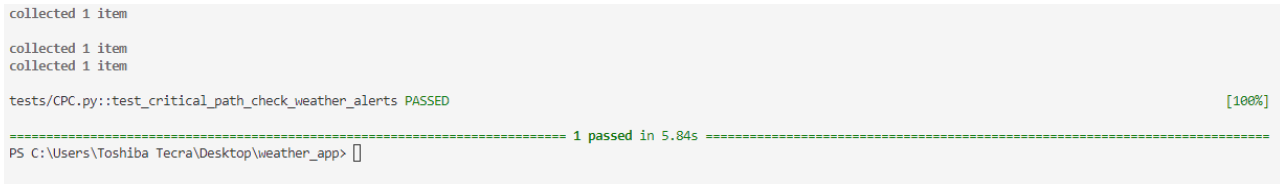
```python
def test_critical_path_check_weather_alerts(app):
    """
    Critical Path Test for check_weather_alerts:
    Covers the longest path where all alert conditions are triggered.
    """
    # Critical test input: All conditions should be true
    test_data = {
        'main': {'temp': 38, 'humidity': 50},
        'weather': [{'main': 'Thunderstorm'}],
        'wind': {'speed': 12},
        'sys': {}
    }

    # Run the method with the critical input
    app.check_weather_alerts(test_data)

    # Assertions to verify all critical alerts are present
    assert len(app.alerts) == 3
    assert "High temperature warning" in app.alerts[0]
    assert "Weather alert: Thunderstorm" in app.alerts[1]
    assert "High wind warning" in app.alerts[2]

    # Also verify the alert button turns red
    assert app.alerts_btn.cget("style") == "Warning.TButton"
```

Test Result:

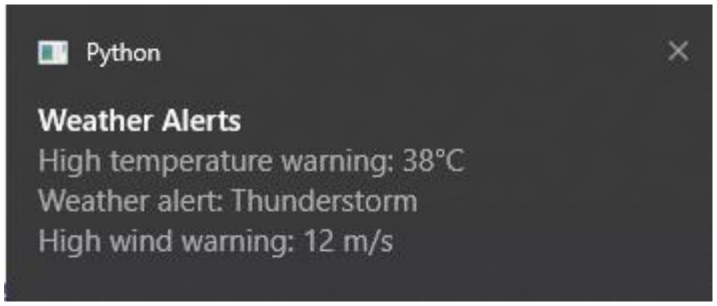


```
collected 1 item

collected 1 item
collected 1 item

tests/CPC.py::test_critical_path_check_weather_alerts PASSED                                                    [100%]

================================================ 1 passed in 5.84s ================================================
PS C:\Users\Toshiba Tecra\Desktop\weather_app> []
```



Python

Weather Alerts
High temperature warning: 38°C
Weather alert: Thunderstorm
High wind warning: 12 m/s

4. Data Flow Analysis:

Variable	Definition Node	Usage Nodes	Anomalies Detected
`temp`	3	4,5	None
`condition`	8	10	No check for `weather[0]` existence
`wind_speed`	12	14	Negative values not handled
`self.alerts`	1	17,18,23	Duplicates possible

Github link: https://github.com/HanzlaAlvi/Weather-App.git