Kyle Cole, Nea Culbreath, Alvaro de Landaluce,
Logan Ford, Sawyer Jones, & Amshah Mushtaq
COMP 330/474 001
12/11/2021

T5 - Use Case Narratives

ShutdownDueToInactivity

Use case: ShutdownDueToInactivity	
Actors: User, ThermometerSystem	
Purpose: Determine inactivity and turn off ThermometerSystem when thirty seconds of inactivity is detected	
Requirements Implemented: 6.1, 6.2, 6.3	
Overview: ThermometerSystem starts a timer that resets every time the user interacts with the ThermometerSystem, and the ThermometerSystem turns off after 30 seconds of no user input	
Type: Essential	
Preconditions: ThermometerSystem is currently turned off	
Postconditions: N/A	
Special Requirements: Ensure timer is set to thirty seconds	
Flow of Events	
Actor Action	**System Response**
1. This use case begins when the user turns on the **ThermometerSystem**.	2. The **ThermometerSystem** starts a 30 second long timer.
	3. When the timer reaches 0 seconds, the **ThermometerSystem** turns off.
Alternative Flow of Events	
Line 2: If the user performs any other action, and the thermometer receives any input,	

the **ThermometerSystem** resets the timer back to 30 seconds.

MeasureTemperature and CalculateFever

Use case: MeasureTemperature and CalculateFever	
Actors: User, ThermometerSystem	
Purpose: Measure and display the average temperature of the user	
Requirements Implemented: 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6	
Overview: ThermometerSystem reads user temperature, calculates the average temperature, and then compares the AverageTemperatureValue to the user's PresetValueTemperature.	
Type: Essential	
Preconditions: ThermometerSystem has been turned on and has passed the self-test, user has set the PresetTemperatureValue, and user has been given access to the MainMenu	
Postconditions: ThermometerSystem displays the MainScreen and waits for user input	
Special Requirements: Message displayed along with the AverageTemperatureValue needs to be above or below the value, and it must not crowd out the temperature reading	
Flow of Events	
Actor Action	**System Response**
1. This use case begins when the user requests a temperature calculation by inputting 3 in the **MainMenu**.	2. The **ReadPeriod** begins; during which the **ThermometerSystem** repeatedly reads the temperature from the sensor until 10 successive readings fall within a range of 0.1 degrees Celsius within ten seconds.
3. The user waits for the **ThermometerSystem** to calculate and display a temperature.	4. The **ThermometerSystem** adds all of the successful temperature readings together and divides them by 10 in order to calculate the **AverageTemperatureValue**.

9. The user views their temperature reading, which is the **AverageTemperatureValue.**	5. The **ThermometerSystem** produces two **BeepSounds** and displays “Reading Successful” on the **MainScreen** for five seconds as an indication of a successful **ReadPeriod**.
	6. Following the two **BeepSounds**, the **ThermometerSystem** displays a single value to the user, which is the **AverageTemperatureValue** to the nearest tenth degree on the **MainScreen** of the **ThermometerSystem**. The ten values in the snapshot will not be made available for the user and are solely for calculation purposes.
	7. The **ThermometerSystem** compares the newly calculated **AverageTemperatureValue** to the user’s previously defined **PresetTemperatureValue**.
	8. A message stating “No Fever Detected” and the **AverageTemperatureValue** is displayed on the **MainScreen** for 5 seconds. The **ThermometerSystem** does not produce a **BeepSound** in the case of no detected fever.
Alternative Flow of Events	
Line 1. If the user inputs a different number, temperature calculation will not begin.	
Line 2: If the **ThermometerSystem** is unable to collect 10 successful readings within 10 seconds, the **ThermometerSystem** will produce one **BeepSound** and display a message stating “Reading Failed” on the **MainScreen** for 5 seconds.	
Line 7: If the user wants to select a different **PresetTemperatureValue,** the user may change the **PresetValueTemperature** through the **MainMenu** of the **ThermometerSystem** to a value that is considered a fever temperature by the user.	
Line 8: If the user has a fever, which occurs when the **AverageTemperatureValu**e is greater than the	

PresetValueTemperature, then the **ThermometerSystem** will make one **BeepSound** and display a message stating “Fever Detected” above the **AverageTemperatureValue** that’s displayed on the **MainScreen** for 5 seconds.
Line 9: If the user is unable to view their reading because of a **ThermometerSystem** failure, then the events will be repeated beginning with **Event 1.**

AccessTemperatureLog and AccessMainMenu

Use case: AccessTemperatureLog and AccessMainMenu	
Actors: User, ThermometerSystem	
Purpose: Allow the user to access the MainMenu and view a log of the last five AverageTemperatureValues the ThermometerSystem has recorded	
Requirements Implemented: 5.1, 5.2, 5.3	
Overview: A log accessed through the MainMenu shows the last five AverageTemperatureValues recorded in memory by the ThermometerSystem, and the MainMenu allows users to select whether they want to view the temperature readings in Celsius or Fahrenheit	
Type: Essential	
Preconditions: ThermometerSystem is currently turned on and functioning properly	
Postconditions: N/A	
Special Requirements: Program is written to appropriately delete the correct temperature entry and append the log	
Flow of Events	
Actor Action	**System Response**
1. This use case begins when the user desires to access the log of the last five **AverageTemperatureValue** that are stored in the **ThermometerSystem’s** memory.	2. The **ThermometerSystem** records in memory the last five temperatures the user has been shown, the **AverageTemperatureValues**, in order of newest to oldest temperature reading.

3. On the **MainMenu** of the **ThermometerSystem**, the user views a log of the last five **AverageTemperatureValues** the **ThermometerSystem** has recorded in Celsius.	
Alternative Flow of Events	
Line 3. If the user wants to view the temperature log as well as all other temperatures within the **ThermometerSystem** in Fahrenheit, the user can select to view the temperature readings in Fahrenheit in the **MainMenu** of the **ThermometerSystem**, which modifies the last five **AverageTeperatureValues** to meet the user's requests. The user can switch back to Celsius through the **MainMenu** as well.	

PerformPowerOnOff and PerformSelfTest

Use case: PerformPowerOnOff and PerformSelfTest
Actors: User, ThermometerSystem
Purpose: Allow users to turn the ThermometerSystem on and off, and ensure that the ThermometerSystem performs a self-test when it is first turned on to check for battery power and sensor responding
Requirements Implemented: 1.1, 1.2, 1.3, 1.4, 1.5, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6
Overview: ThermometerSystem is turned on by the press of a button, and ThermometerSystem proceeds by performing a self-test to check for battery life and sensor responsiveness until both tests pass
Type: Essential
Preconditions: ThermometerSystem is currently turned off
Postconditions: ThermometerSystem will have been turned on with the self-test conducted, and the ThermometerSystem will be ready to begin taking temperatures.
Special Requirements: ThermometerSystem must check for sensor response and battery and present messages to the users in a timely manner
Flow of Events

Actor Action	System Response
1. This use case begins when the user presses the **PowerButton** on the thermometer while the thermometer is turned off.	2. The thermometer turns on, and the **MainScreen** of the thermometer displays a welcome message for five seconds, which is “Welcome to the Electronic Thermometer.”
5. The user is able to view the **MainScreen** and the **ThermometerSystem’s BatteryLife**.	3. The **ThermometerSystem** displays a self-test message on the **MainScreen** when the thermometer is first turned on. The self-test message explicitly states “Self-Test Beginning Now” to the user on the **MainScreen** for five seconds.
7. The user is continued to be able to view the **MainScreen**.	4. The **ThermometerSystem** checks the **BatteryLife** to determine if the battery has at least a 20% charge remaining, in which case the **MainScreen** is displayed. The **ThermometerSystem** also displays **BatteryLife** as a graphical battery icon on the **MainScreen** to update the user on the **ThermometerSystem’s BatteryLife.**
	6. The **ThermometerSystem** checks if the sensor is able to detect the temperature, in which case the **MainScreen** is displayed.
	8. Since the battery power and the sensor response tests pass, the **ThermometerSystem** produces two **BeepSounds**.
Alternative Flow of Events	
Line 2: If the user presses the PowerButton while the thermometer is currently turned on, the **ThermometerSystem** turns off. When the **ThermometerSystem** is turned off, the **AverageTemperat**	

is deleted automatically from the **MainScreen** of the **ThermometerSystem**. Only the **PresetValueTemperature** and the temperature readings in the log will be stored when the **ThermometerSystem** is turned off.
Line 4: If the thermometer **BatteryLife** has less than a 20% charge remaining, a warning message is displayed for five seconds, then the **MainScreen** is displayed. The battery alert message explicitly states "Low Battery" to the user.
Line 6: If the sensor is not able to detect the temperature, an error message is displayed for five seconds, and the user is not allowed access to the **MainScreen** until a sensor response is verified. The error message explicitly states “Sensor Error.