Implementation Deliverables and Complete System Materials

Team V

Alexis Carranza, Aditya Dhungana, Bhargvi Handa, Emily Meister, Kiarra Villaraza, Robert Xie

Table of Contents:

Purpose

The purpose of this document is to create a list of requirements for building an Electronic Thermometer that can measure temperature from a distance.

Requirements

Identifier	***Title***	***Requirement Statements***
Req 1: Importance: Essential	Initial Set-Up Process	The very first time the user opens the app, the system shall prompt the user to set up the thermometer to their liking. I.e. customize **Unit of measurement**, **Fever limit**, **additional important temperatures**, **Display** options, and **Temperature log**. All of these customization settings can be accessed through a ‘Settings’ button when the thermometer has been successfully **initialised**.
Req 1.1	Initial Temperature Set-Up	The system shall prompt the user to select their desired **Unit of measurement**.

Req 1.2	Initial Fever Limit Set-Up	The system shall prompt the user to select their desired **Fever limit**.
Req 1.3	Initial Display Set-Up	The system shall prompt the user to select their desired **Display**.
Req 1.4	Initial Temperature Log Set-Up	The system shall prompt the user to select whether or not they wish to create a **Temperature log**.
Req 1.5	Initial Display Check	If the system is first powered on, then **Display** shall showcase a **TemperatureInput** of zero and a green background color; otherwise **Display** will show no background color and there would be no **TemperatureInput** on the screen .
Req 1.6	Change Settings	The system shall allow the user to return to the settings section and customize/change anything they initialized in Req 1.

Req 2 Importance: Essential	System Reset	After the system is **initialised,** it shall provide users an option to reset the system by pressing the **SystemReset** button on the **Display.**
Req 2.1	System Reset Pressed	The system shall check if the **SystemReset** button has been pressed to which the device will reset **SystemIdle** and **TemperatureInput** to zero and will make a **BeepingSound** one time.
Req 2.2	Return to Initialization	On reset, the system shall return to Initial startup.
Req 3 Importance: Essential	Track System non-Active state.	When the thermometer is on and in idle state, the system shall record the time period during which the system is inactive (**SystemIdle)** as the duration of no activity, i.e.,

		the user pressing a MeasureButton or **SystemReset** on the **Display.**
Req 3.1	System Active	If the user presses **MeasureButton** or **SystemReset** at any time then, **the SystemIdle** counter is set to zero; otherwise, the SystemIdle counter will increase by 1 every second.
Req 3.2	Automatic Power Off	If the SystemIdle counter is at 30 seconds, then the system will power off.

Req 4 Importance: Essential	System Self-Test	On system startup the system shall calibrate the app and make sure that the app is able to take correct readings. If it is successfully **calibrate**d, the app will produce the **Beeping Sound** once; otherwise, an error code will pop up and prompt the user to restart the app so the **self-test** can run again.
Req 5 Importance: Essential	Check Temperature	If *Req. 4* passes and the device is ready to be used then, the thermometer shall read the temperature when the user presses and holds the **MeasureButton.**
Req 5.1	Average Detected Temperatures	The thermometer shall read temperatures repeatedly until 10 values are within a range of 0.1 degrees and then calculate the average of the 10 values. Once the data is available, the

		thermometer shall display the value in the **Display**.
Req 5.2	Detected Fever Limit	The thermometer will produce **BeepingSound** three times if a person has a temperature over the **Fever limit**.
Req 5.3	Detected Additional Important Temperatures	The thermometer will produce **Beeping Sound** two times if a person has a temperature that matches the **User specified temperature range.**
Req 6 Importance: Essential	Store Temperature	After the measurement has been done, **Temperature input** will be stored in the Temperature **log** if it has been set up by the user; otherwise, the **Temperature input** will stay on the screen until the user takes another temperature or remains idle. See requirement 3.2 for more details.

Req 7 Importance: Essential	Power State	When the system is off and **PowerButton** is pressed, then the thermometer will power on; otherwise, when the system is on and **PowerButton** is pressed, then the thermometer will power off
Req 7.1	Thermometer Interactability	On power on, the thermometer's features are accessible and interactable; otherwise, on power off, the thermometer's features are not accessible or interactable

Use Case Narratives

Use Case: **Settings Configuration**
Actors: User
Purpose: Configure Thermometer's settings to the User's preference. Standardizes Unit of measurement, fever limit, Display options, temperature log, and User specified temperature range.
Requirements Implemented: 1-1.6 , 3-3.2
Overview: The Thermometer is turned ON for the first time and prompts the user to configure its settings. The User sets available options according to their preference.
Type: Slightly Essential
Preconditions: The Thermometer is OFF and the User presses the **PowerButton**.
Postcondition: **Unit of measurement**, **fever limit**, **Display** options, **temperature log**, and **User specified temperature range** have been initialized based on the User's preferences.
Special Requirements: The Thermometer has never been turned ON before. It's the first time a User has powered the Thermometer ON.

Flow of Events

Actor Action	**System Response**
1. The User wishes to interact with the Thermometer or turn it on	
2. The User presses the **PowerButton**	3. Thermometer is set to an ON state. Initial Set-Up Process is performed. Thermometer prompts the User to customize the Unit **of measurement, Fever limit, Display** options**,** and **Temperature log.**
4. The User Initialized Thermometer settings	5. Initial Display Check is performed. **Display** showcases **TempertureInput** of 0 with a green background color. Thermometer state is set to idle. **The SystemIdle** counter is set to 0 and begins incrementing by each second that passes in idle state.

Alternative Flow of Events

Use Case: **Reset System**
Actors: User
Purpose: Allow the user to press the reset button to have the thermometer reset
Requirements Implemented: 2-2.2, 3-3.2
Overview: The user presses the **SystemReset** button. The thermometer resets the **SystemIdle** and **TemperatureInput** and it will return to the Initial Startup.
Type: Essential
Preconditions: The Thermometer is on and the user can access the thermometer's features.
Postconditions: The Thermometer's device is on the initial setup.
Special Requirements:

Flow of Events

Actor Action	System Response
1. The User wishes to reset the thermometer. 2. The User presses the **SystemReset** button.	3. The Thermometer system prompts the user to confirm reset 5a. The Thermometer system receives no from the User and the system returns to the ON Display and resets **SystemIdle** to zero.
4.User confirms yes or no to **SystemReset**	5b. The Thermometer system receives a yes input from the User and the Thermometer system resets **SystemIdle** and **TemperatureInput** to zero. It will produce a beeping sound one time. 6. The thermometer system goes to the initial startup.

Alternative Flow of Events

Use Case: **Check Temperature**
Actors: User, Subject
Purpose: Record the subject's temperature and determine if they have a fever
Requirements Implemented: 3-3.2, 5-5.3
Overview: The User presses and holds the **MeasureButton**. The Thermometer reads temperature values, stores an average from those values, calculates whether it is within the Fever **limit** and then displays that average to the User.
Type: Essential
Preconditions: The Thermometer is ON and the User has access to the Thermometer's features.
Postcondition: The Subject's temperature is stored
Special Requirements: User has already configured and initialized Thermometer settings during the Initial Set-Up Process.

Flow of Events

Actor Action	**System Response**
1. The User wishes to measure someone's temperature	
	3. **SystemIdle** counter is reset to 0. Thermometer performs Check Temperature. Thermometer reads 10 consecutive temperatures within a range of 0.1 degrees, calculates their average, stores into **Temperature log**, and displays the value to the **Display.** If Temperature value is over the **Fever limit,** Thermometer produces **BeepingSound** three times and two times if it is within **User Specified Range.**
2. The User presses and holds **MeasureButton**	
4. The User lets go of **MeasureButton**	5. **SystemIdle is incremented** by 1. **SystemIdle** continues to increment by 1 by each second passed in idle state. **TemperatureInput** remains on the **Display**

Alternative Flow of Events

Use Case: **Thermometer ON**

Actors: User
Purpose: Allow the user to access the Thermometer's features and interact with them
Requirements Implemented: 7-7.1
Overview: The User presses the PowerButton and is granted access to the Thermometer's features. The Thermometer's features are then interactable.
Type: Essential
Preconditions: The Thermometer is OFF. User is unable to check temperature
Postcondition: User has access to the Thermometer's features. User is able to check temperature
Special Requirements: The Thermometer has been previously turned ON before. This is not the first time a User has turned ON the Thermometer.

Flow of Events

Actor Action	**System Response**
1. The User wishes to Interact with the Thermometer or turn it ON 2. The User presses the **PowerButton**	3. Thermometer is set to an ON state. Initial Display Check is performed. Display showcases **TempertureInput** of 0 with a green background color. Thermometer state is set to idle. **SystemIdle** counter is set to 0 and begins incrementing by each second that passes in idle state.

Alternative Flow of Events

Line 3-When the user presses the power button and the thermometer fails to self diagnose, it will display the error code on the screen.

Use Case: **Thermometer OFF**
Actors: User

Purpose: Leave the Thermometer in an OFF state and conserve resources i.e, power and memory.
Requirements Implemented: 1.6, 7-7.1
Overview: The User presses the PowerButton and is no longer able to access the Thermometer's features.
Type: Essential
Preconditions: The Thermometer is ON and the user is able to access the Thermometer's features.
Postcondition: The Thermometer is left in an OFF state. User is unable to check temperature
Special Requirements:

Flow of Events

Actor Action	**System Response**
1. The User wishes to no longer interact with the Thermometer	
2. The User presses the **PowerButton**	3. Thermometer is set to an OFF state. System configures the Thermometer **Display** to show no **TemperatureInput** or background. **Display** will remain empty or blank.

Alternative Flow of Events

Traceability Table

Req #	Use case	UML Class diagram	Sequence diagram
Req 1 Initial Set-Up Process	Settings Configuration	ThermometerDisplay prompt() setUnitOfMeasurement() setFeverLimit()	thermometerDisplay(), setUnitOfMeasurement(), setFeverLimit()
Req 1.1	Settings Configuration	ThermometerDisplay setUnitOfMeasurement	thermometerDisplay(), setUnitOfMeasurement
Req 1.2	Settings Configuration	ThermometerDisplay setFeverLimit()	thermometerDisplay, setFeverLimit()
Req 1.3	Settings Configuration	ThermometerDisplay	thermometerDisplay()
Req 1.4	Settings Configuration	ThermometerDisplay optionLog()	thermometerDisplay()
Req 1.5	Settings Configuration	ThermometerDisplay ThermometerDisplay() shownTemperature()	thermometerDisplay(), ThermometerDisplay()

Req 1.6	Settings Configuration and Thermometer OFF	Thermometer Display settingsSection() setUnitOfMeasurement() setFeverLimit() optionLog() shownTemperature() Thermometer System	thermometerDisplay(), settingsSection(), setUnitOfMeasurement(), setFeverLimit()
Req 2 System Reset	Reset System	Thermometer Display getInitialized() systemResetButton() Thermometer System resetSystem() Reset resetSystem() SystemCounter resetCounter()	resetSystem()
Req 2.1	Reset System	Thermometer Display() systemResetButton() shownTemper	resetSystem()

		ature() produceBeepingSound() ThermometerSystem resetSystem() ResetState resetSystem() SystemCounter resetCounter() TempCalculator setTemperatureInput()	
Req 2.2	Reset System	ThermometerDisplay systemResetButton() prompt() produceBeepingSound() ThermometerSystem resetSystem() ResetState resetSystem() SystemCount	resetSystem()

		er resetCounter( ) TempCalculat or setThermome terInput()	
Req 3 Track System non-Active state	Reset System	Thermometer Display idleState() systemReset Button() Thermometer System getSystemCo unter() SystemCount er getCounter()	resetSystem()
Req 3.1	Reset System	Thermometer Display systemReset Button idleState() Thermometer System resetSystem() ResetState resetSystem() SystemCount	resetSystem()

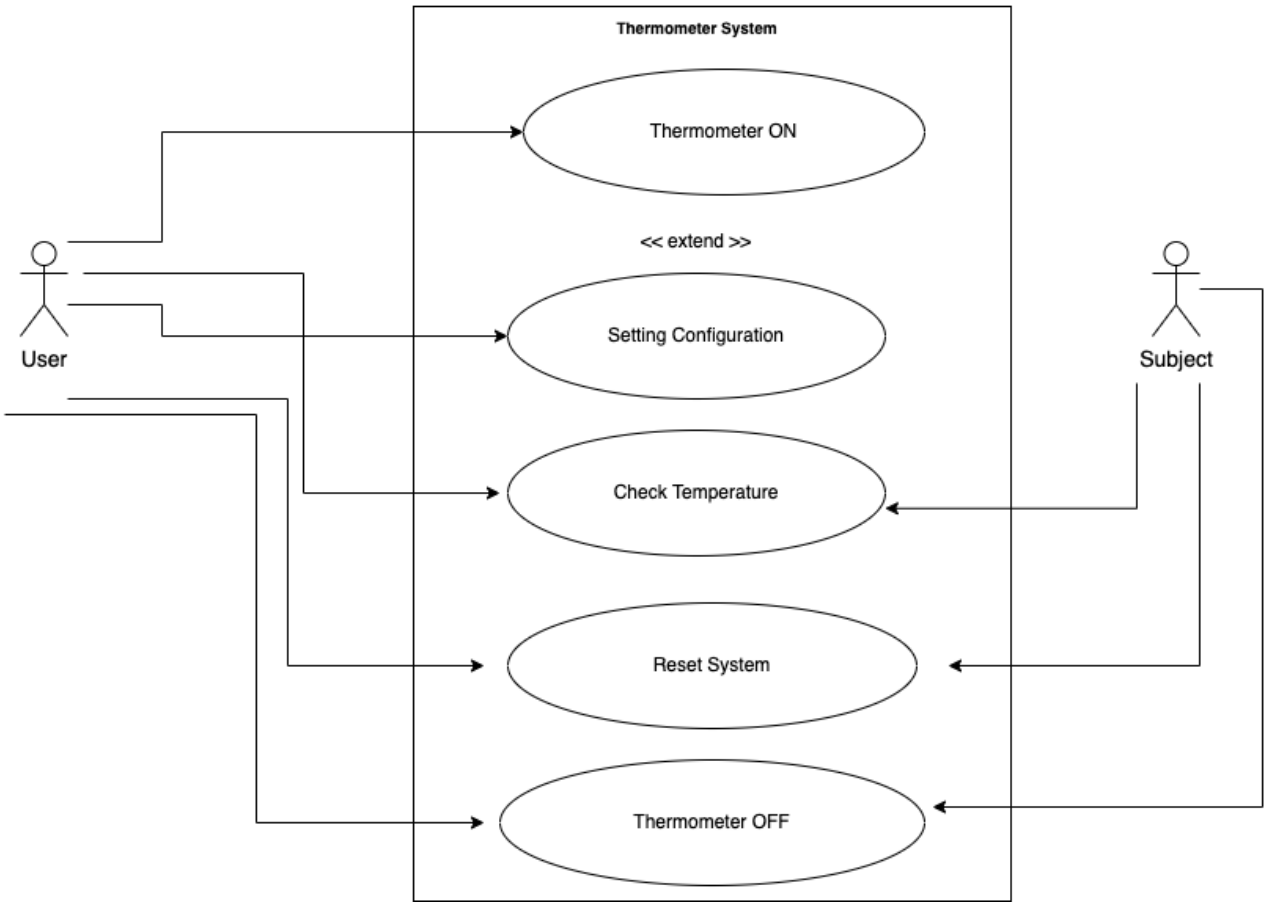
		er resetCounter( ) Counter()	
Req 3.2	Reset System	SystemCount er Counter() Thermometer System turnOff()	resetSystem()
Req 4 System Self-Test		Self-Test checkSystem Calculator() checkSystem Counter() checkSystem Display() Thermometer System getSystemCo unter() getThermome terDisplay() getTempCalcu lator()	selfTest()
Req 5 Check Temperature	Check Temperature,	Thermometer System TempCheckSt ate checkTemp() TempCalculat or	checkTemp(), measureTemp erature(), setLog()

		SystemCounter	
Req 5.1	Check Temperature	ThermometerDisplay MeasureButton() ThermometerSystem checkTemp() TempCheckState checkTemp() TempCalculator measureTemperature() setTemperatureInput() SystemCounter	checkTemp(), measureTemperature(), setLog()
Req 5.2	Check Temperature	TempCalculator measureTemperature() produceBeepingSound() SystemCounter	checkTemp(), measureTemperature(), setLog()
Req 5.3	Check Temperature	TempCalculator measureTemperature()	checkTemp(), measureTemperature(), setLog()

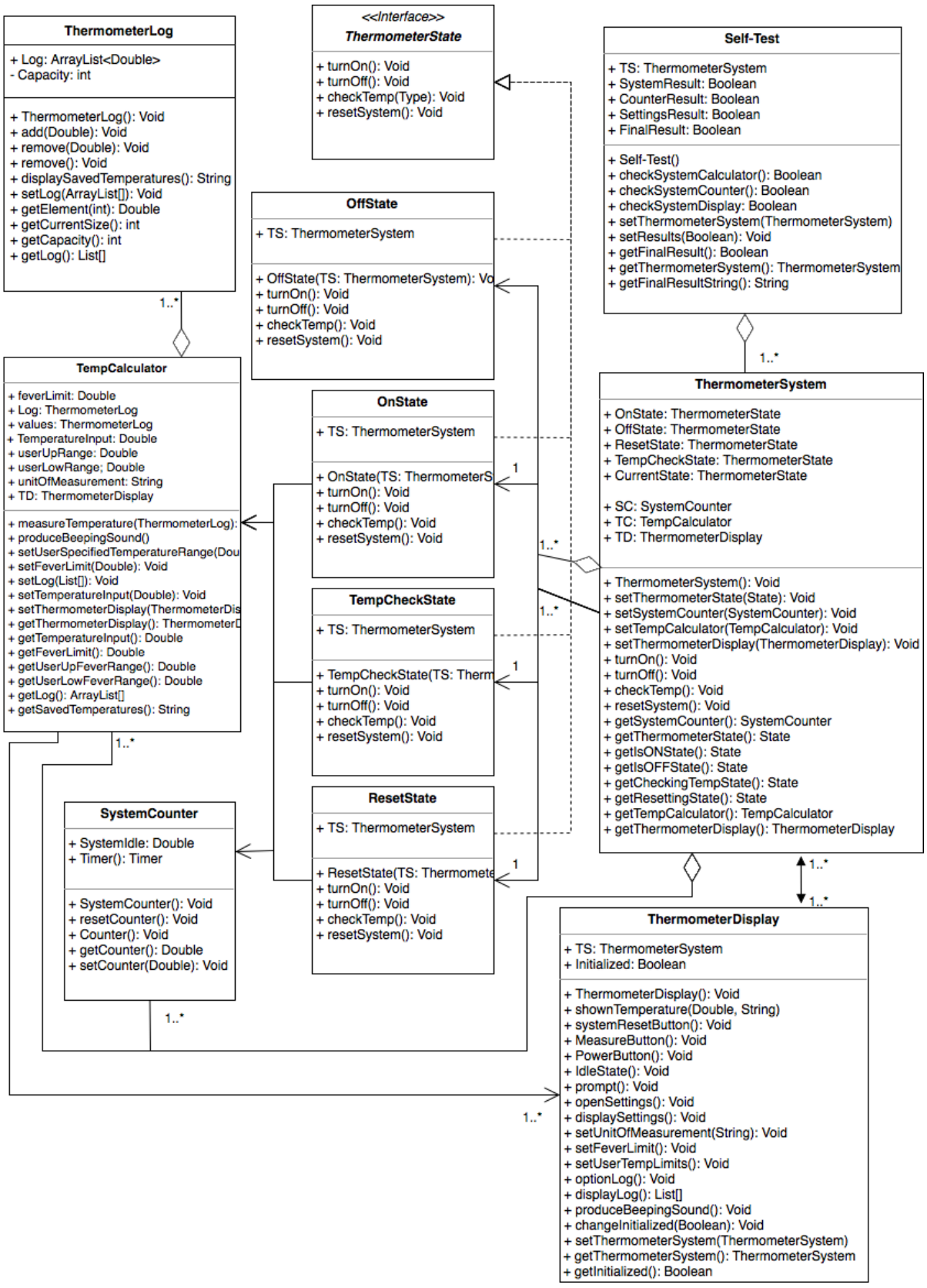
		produceBeepingSound() SystemCounter	
Req 6 Store Temperature		TempCalculator getLog() getSavedTemperatures() ThermometerLog add()	checkTemp(), measureTemperature(), setLog()
Req 7 Power State	Thermometer ON and Thermometer OFF	ThermometerDisplay PowerButton() ThermometerSystem turnOn() turnOff() OnState turnOn() turnOff() OffState turnOn() turnOff()	turnOn(), turnOff()
Req 7.1	Thermometer ON and Thermometer OFF	ThermometerDisplay ThermometerSystem	turnOn(), turnOff()

		TempCalculator	

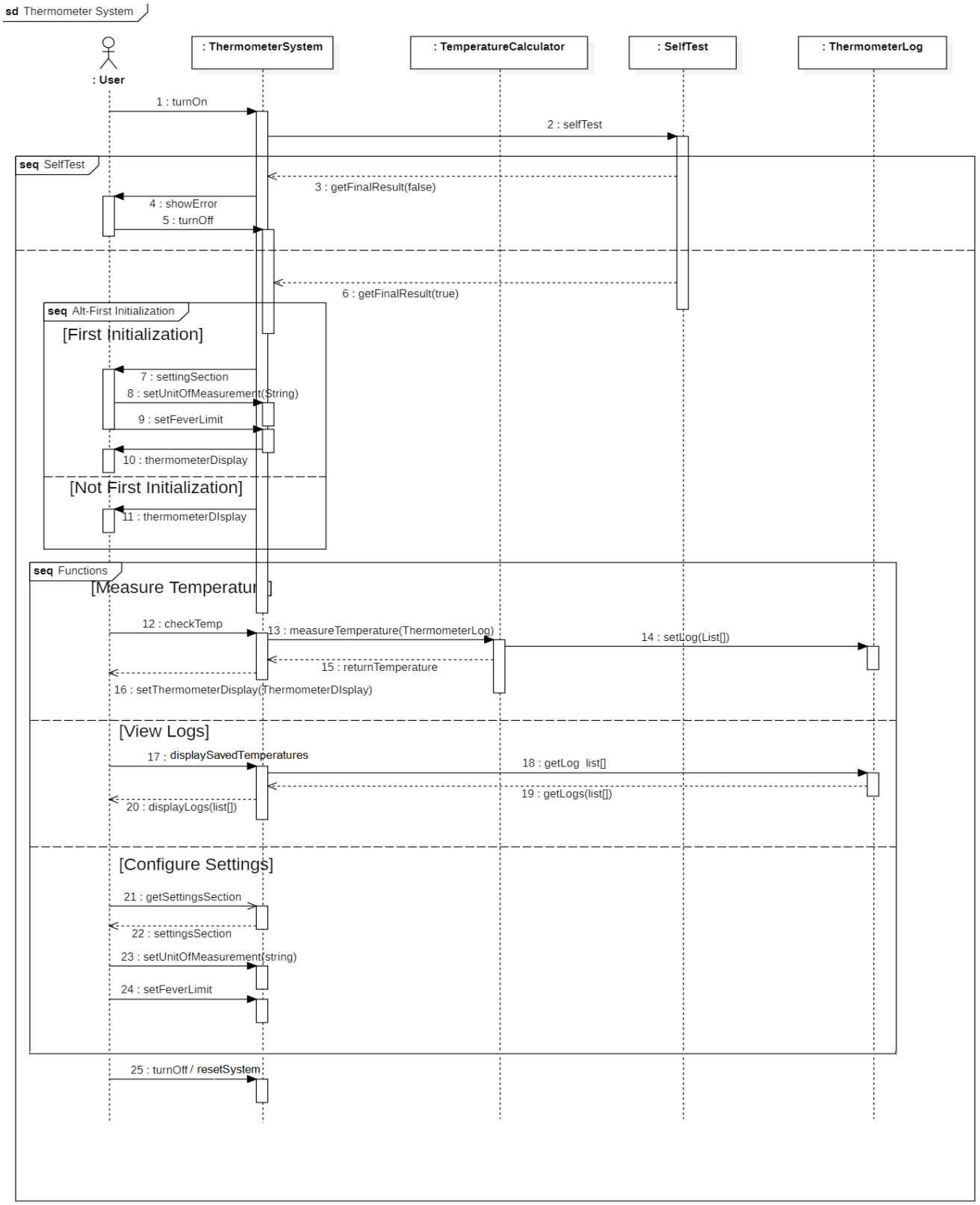
Use Case Model:



UML Class Diagram:



UML Sequence Diagram



Team Metrics

Date	Meeting Method	Attendance	Start time and end time	Next Steps	Key Topics Discussed
10/6/21	Zoom	Bhargvi Handa, Alexis Carranza, Robert Xie, Kiarra Villaraza, Emily Meister	7pm - 9pm	Use the requirements metrics to review all of the requirements and see which ones need to be revised.	Everyone shared their individual and atomic requirements with each other. Kiarra shared her screen and we started to pick which requirements we wanted to use. We worked on organizing the requirements in a logical way that could easily be followed.
10/27/21	Zoom	Bhargvi Handa, Alexis Carranza, Robert Xie, Kiarra Villaraza, Emily Meister	1pm - 2pm	Finish writing out the use case narratives and have everyone check it over.	We discussed everyone's drafts for the use case narratives. We started to organize what use case's we wanted to write in the narrative. Kiarra created the use case model.
11/24/21	Zoom	Bhargvi Handa and Alexis Carranza	12pm - 1:30pm	Get the UML Class Diagram finished so we can post it on the forum. Get started on the sequence diagram.	We started to create the UML Class Diagram. We went over the mistakes that were in the requirements and made corrections to them. Bhargvi revised the Use Case Model so it did not have the mistakes that Dr.Honig mentioned.
12/9/21	Zoom	Bhargvi Handa, Alexis Carranza, Robert Xie, Kiarra Villaraza,	1:30pm - 2:30pm	Finish the sequence diagram column of the traceability table and check the sequence	We discussed what we were going to put into each column in the traceability table and then started to work on the table. We also added the use case narratives, use case

		Emily Meister		diagram to make sure all the methods and classes match the UML Class Diagram.	model to our requirements document so we'd have everything in one spot. We reorganized our table of contents so that it has all of the work we've done so far.
12/13/21	Zoom	Bhargvi Handa, Alexis Carranza, Robert Xie, Kiarra Villaraza, Aditya Dhungana	10am - 10:39am	Finish the last column of the traceability table, add use case narratives to the presentation, finish the code and add the code and test results to the slides. Then once all of those parts are done, check everything over so there's no errors.	We discussed what information is going on each slide in the presentation. We also discussed who is presenting what slide. We discussed what parts we have left to write in the reports and the slides.

Code Metrics and Test results

1. Does the code measure temperature?
   Yes, we receive temperature as an ArrayList of type ThermometerLog containing multiple “temperature” inputs as a double value.

2. Does the code perform the correct measure calculation?
   Yes. Based on the initial value, we receive the next temperature inputs until 10 are received. Then we average those 10 values and return the temperature if it’s below the fever limit. Otherwise, we stop receiving inputs when a fever is detected, and a beeping sound is made.

3. Does the temperature get added to the log?
   Yes, for our coding simulation, TempCalculator holds the first ten appropriate temperature inputs into value.

4. Is a sound made when a temperature is above the fever limit?
   Yes, produceBeepingSound() is called from measureTemperature() and plays a sound when the temperature is above the fever limit

5. How many lines of code are there?
   For our program called TempCalculator, there are 187 lines. For our ThermometerLog program, there are 50 lines. This is not including the tests.

Glossary

PowerButton : Virtual button on the **Display.** Prompts the user to access its features when pressed. Works as a power on/off feature where if you power on you'll be able to customize the thermometer to your liking and then after that you'll be able to conduct the self test and be able to access its other features.

User Specified Temperature Range : Any additional degrees/range of degrees the user will select to indicate importance. I.e. If the user wants to detect any temperature/range of temperatures in addition to the **fever limit**.

Initialised: State of the thermometer when it has successfully performed a self test after being turned on and is ready to use.

Beeping Sound: audible pre-recorded sound; duration of about 1 second.

Display: Virtual screen about 3in. by 2in. Will showcase **temperature input**, **temperature log**, and **System Idle State**.

Fever Limit : The degree/range of degrees the user will select to indicate what temperature a fever is.

Calibrate: Adjust measured values to take into account the external factors like **ambient temperature**.

Ambient Temperature: Temperature of the surrounding in which the thermometer is.

MeasureButton: Virtual button on the **Display**; clickable; is either idle or processing; allows the user to receive/send temperature as an input to the system.

Temperature Input: to its degree (tenth, hundredth – ie 97, 97.0, 97.00); can be negative or positive; measured in Celsius and Fahrenheit; is zero on initial startup.

Temperature Log: A log that will allow users to save any of their **Temperature Input(s)** for future reference. Found in the **temperature log** section of the app. **Temperature log** customization will be found in the **settings** section of the app.

Unit of Measurement : The unit of measurement (Celsius or Fahrenheit) the user will select for the thermometer. Found in the **settings** section of the app.

Settings : Section of the app that will house all of the customizable features for the user such as **unit of measurement, fever limit, temperature display, temperature log permission**.

SystemIdle: Value between 0 and 30 seconds; keeps track of the amount of time the system has been idle.

SystemReset: Visual button on the **Display**.

Self-Test : The tests the system runs through to calibrate the app and make sure that the app is able to take correct readings.

Link to Repository

Here is the link to our repo where you can view our code.
https://github.com/a-carranza/TempCalculator