Change request log

1. Concept Location

Step #	Description	Rationale
1	Ran the system and accessed the Watch List and Point Details interfaces.	To familiarize with where values are displayed.
2	Searched for "display value" in the source code using the IDE's search tool.	To locate relevant classes handling value display.
3	Identified SensorDataRenderer.java and PointDetailsUI.java as handling numeric display.	These classes determine how sensor values appear.
4	Inspected formatValue() method in SensorDataRenderer.	This method formats values before display.
5	Used the debugger to track the flow of data from the database to the UI.	Verified how values are processed before display.
6	Examined DatabaseHandler.java to check how numeric values are stored.	Ensured the change should only impact display, not storage.
7	Checked SettingsUI.java to see if display formatting options exist.	Verified there were no existing settings affecting numeric precision.
8	Searched for instances of formatValue() being called in the codebase.	Ensured all affected areas were identified.
9	Reviewed test cases related to numeric display in SensorDataTest.java.	Assessed whether additional tests were needed.
10.	Identified dependencies in UIRenderer.java that might be affected.	Ensured no unintended side effects on UI rendering.

Time spent (in minutes): 45 minutes

org/mango/display/SensorDataRenderer.java/formatValue()
- org/mango/ui/PointDetailsUI.java/displayValues()
- org/mango/db/DatabaseHandler.java/getSensorData()
- org/mango/ui/SettingsUI.java/initSettings()
- org/mango/test/SensorDataTest.java/testDisplayValues()
- org/mango/render/UIRenderer.java/renderValues()

2. Impact Analysis

Step #	Description	Rationale
1	Traced method calls leading to formatValue().	Ensured no unintended effects from changes.
2	Inspected DatabaseHandler.java for data storage impact.	Ensured values were only formatted for display, not modified in storage.

3	Confirmed UI classes pulling values from SensorDataRenderer.	Verified only visual representation was altered.

Time spent (in minutes): 30 minutes

• org/mango/db/DatabaseHandler.java/getSensorData()
• org/mango/display/SensorDataRenderer.java/formatValue()
• org/mango/ui/PointDetailsUI.java/displayValues()

3. Actualization

Step #	Description	Rationale
1	Modified formatValue() to apply decimal truncation using Math.floor(value * 100) / 100.	Ensured truncation instead of rounding.
2	Applied changes to all instances where values were displayed.	Maintained consistency in the UI.
3	Verified display updates correctly in Watch List and Point Details.	Ensured correctness of the implementation.

Time spent (in minutes): 50 minutes

Classes Changed:
• org/mango/display/SensorDataRenderer.java/formatValue()
• org/mango/ui/PointDetailsUI.java/displayValues()

4. Validation

Step #	Description	Rationale
1	Functional Testing	This is the regular expected behavior. The test passed.
2	Validated with different values	This is the regular expected behavior. The test passed.

Time spent (in minutes): 30 minutes

5. Summary of the change request

Phase	Time (minutes)	No. of classes inspected	No. of classes changed	No. of methods inspected	No. of methods changes
Concept location	45	6	0	6	0
Impact Analysis	30	3	0	3	0
Actualization	50	0	2	2	2
Verification	30	0	0	0	0

6. Conclusions

Concept location was straightforward, but impact analysis required careful checking to avoid modifying stored data. Testing confirmed correct truncation behaviour.

While the implementation of the change request was relatively simple, it required significant attention to detail to ensure correctness. Ensuring truncation instead of rounding was crucial, as rounding could have led to unintended inaccuracies. Additionally, we had to carefully trace the flow of numeric values through the system to prevent any unintended modifications in the database layer.

The most challenging part was ensuring that all instances of numerical display followed the truncation logic without affecting other aspects of the system. We also verified that the changes did not introduce performance issues, as applying transformations to displayed values can sometimes add unnecessary computational overhead.

Overall, the structured approach to concept location, impact analysis, and validation helped us systematically implement the change without issues. The experience provided valuable insights into how small changes in data presentation can have a significant impact on usability and correctness.