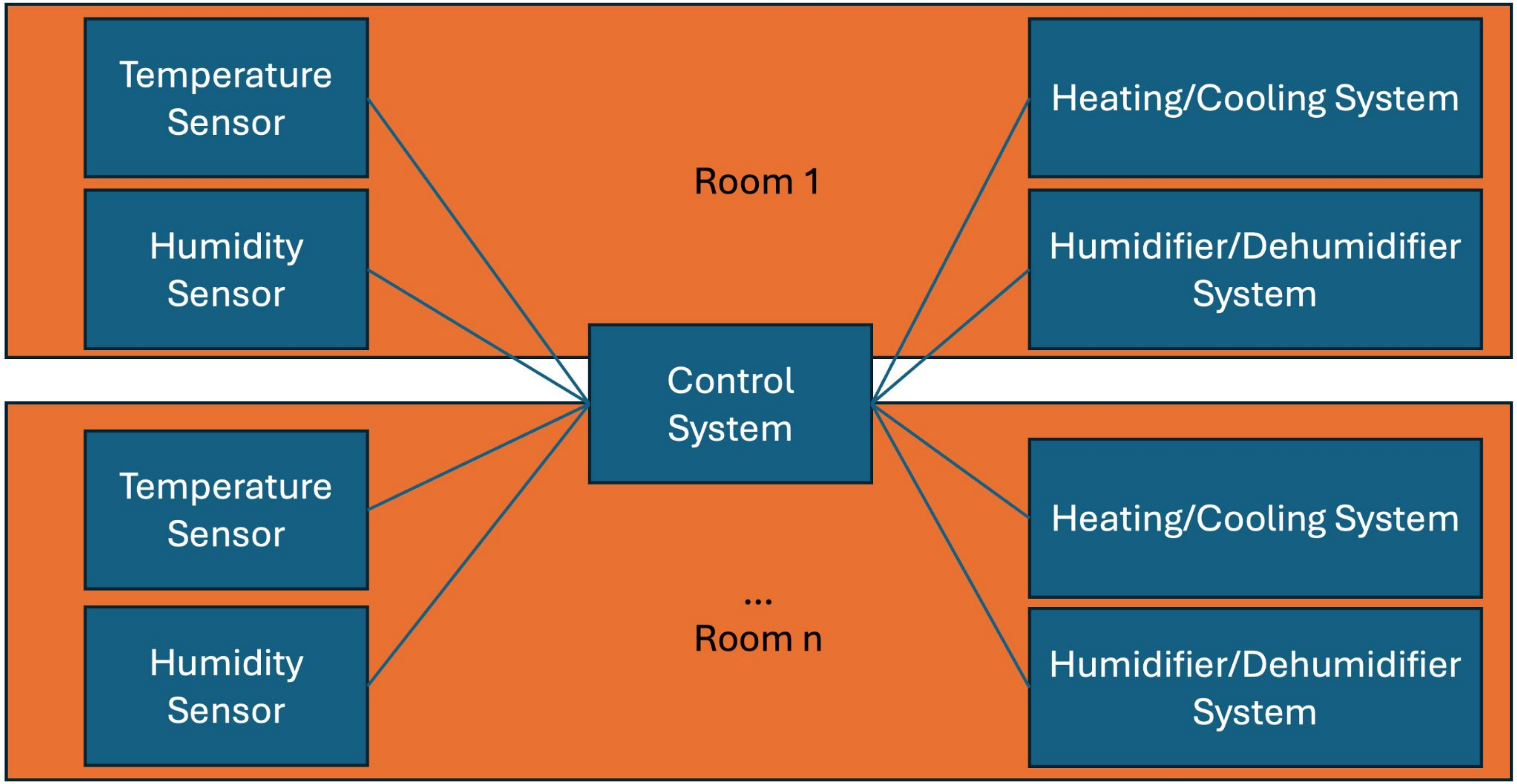
17-423/723: Designing Large-scale Software Systems

Midterm Review

Mar 15, 2024

Recall: Case Study System



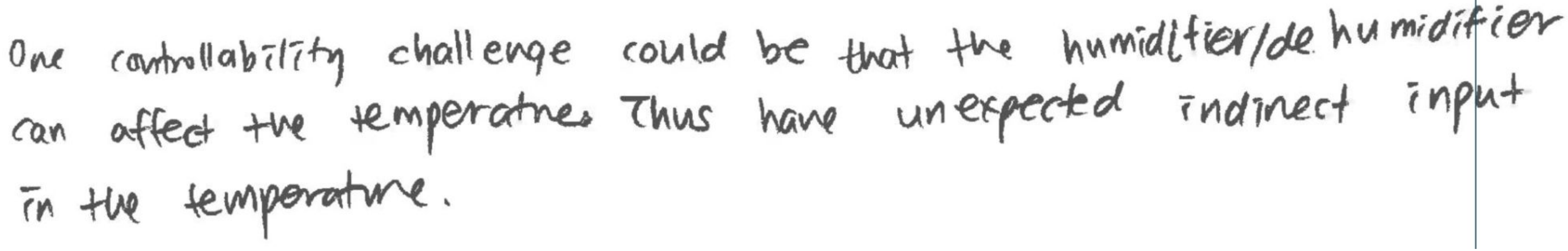
Task 1: Controllability & Observability

Common mistakes

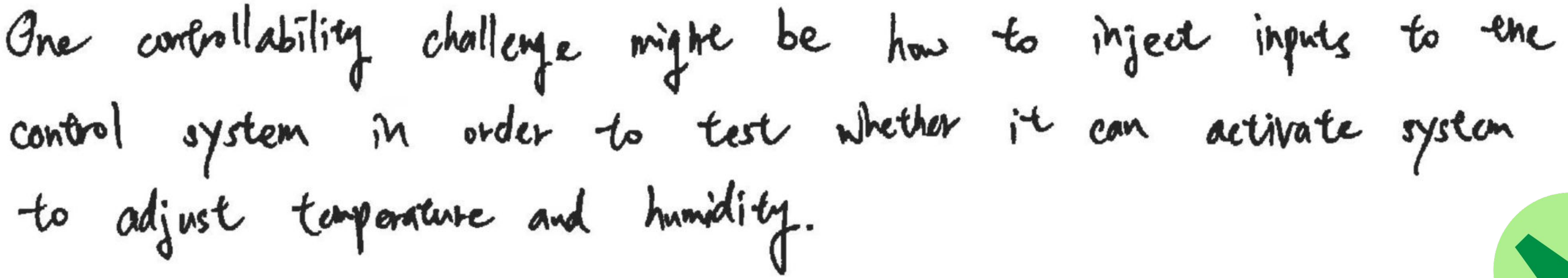
- Answer is too vague
- Answer does not apply to the case study system
- Answer does not really describe controllability or observability in the context of testability
  - e.g., Potential bugs rather than challenges of testing
  - e.g., End-user-facing properties rather than testing challenges
- Controllability is mixed up with observability

Task 1: Controllability & Observability

1.1. Controllability Challenge (4pts)

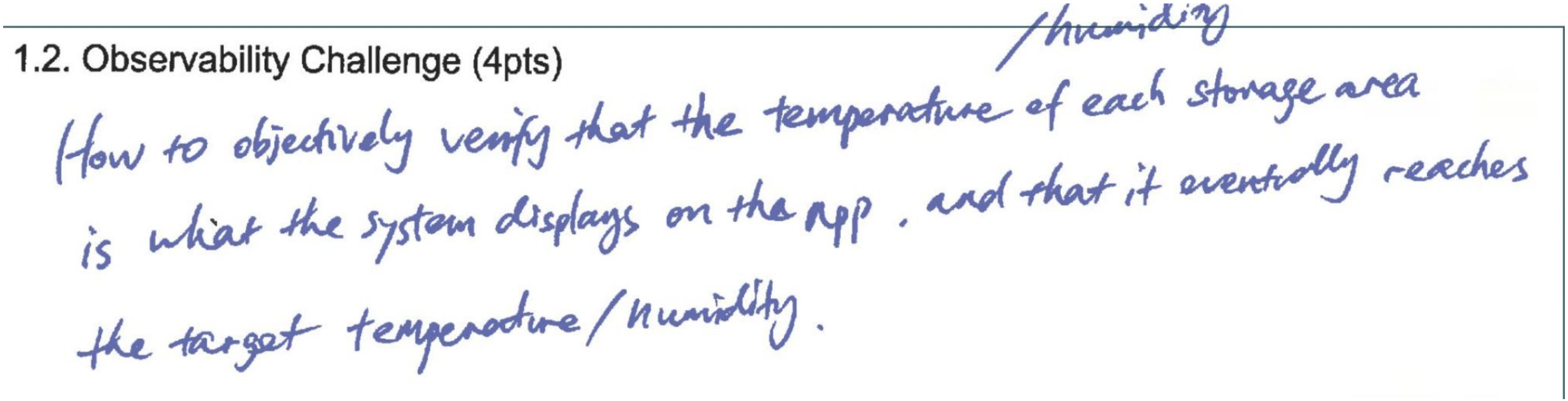


One controllability challenge could be that the humidifier/dehumidifier can affect the temperature. Thus have unexpected indirect input in the temperature.



One controllability challenge might be how to inject inputs to the control system in order to test whether it can activate system to adjust temperature and humidity.

Task 1: Controllability & Observability



1.2. Observability Challenge (4pts)

How to objectively verify that the temperature/humidity of each storage area is what the system displays on the app, and that it eventually reaches the target temperature/humidity.

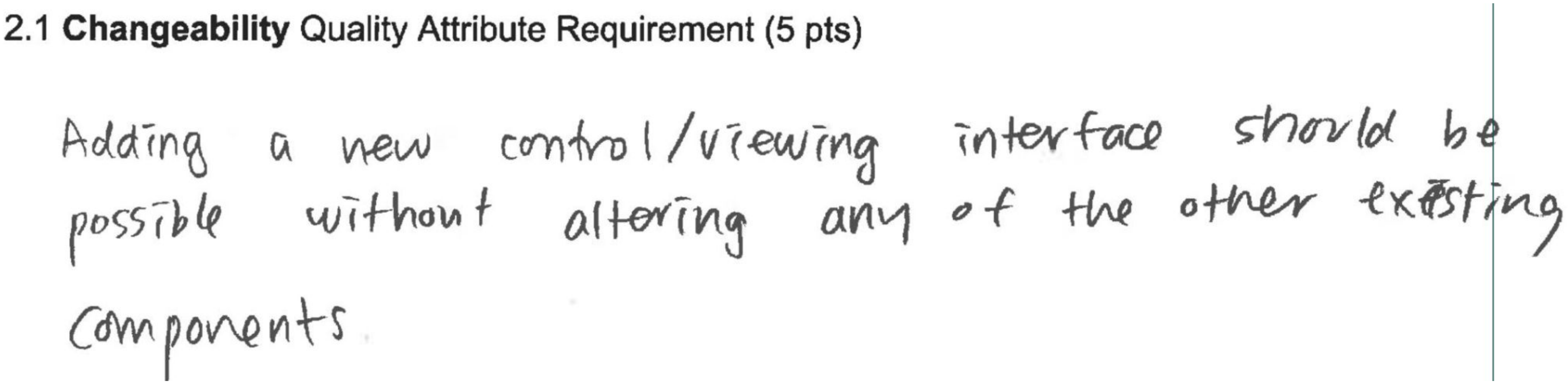
✓

Task 2: Quality Attribute Specifications

Common mistakes

- Answer does not actually describe changeability/interoperability
  - “Should be able to add new sensors without data corruption” -> This is about reliability, not changeability!
  - “Should be able to add a new room without affecting other rooms” -> This is about changeability, not interoperability!
- Answer is not a quality that is measurable (binary or quantitative)
  - “Should be able to add new types of sensors to the system” -> How much effort does this change require?

Task 2: Quality Attribute Specifications

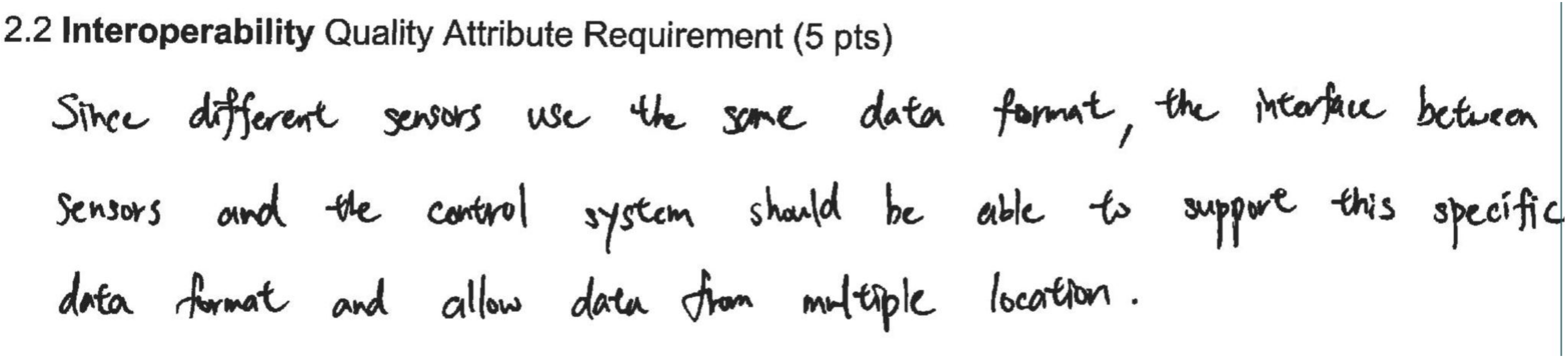


2.1 **Changeability** Quality Attribute Requirement (5 pts)

Adding a new control/viewing interface should be possible without altering any of the other existing components.

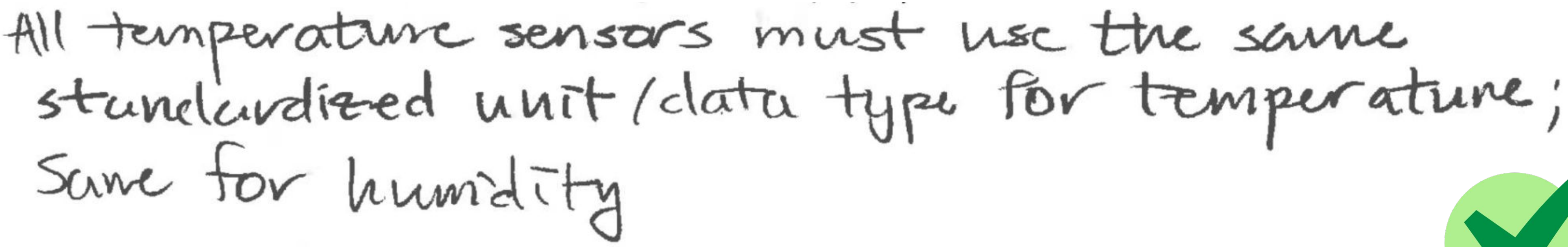
✓

Task 2: Quality Attribute Specifications



2.2 **Interoperability** Quality Attribute Requirement (5 pts)

Since different sensors use the same data format, the interface between sensors and the control system should be able to suppore this specific data format and allow data from multiple location.



All temperature sensors must use the same standardized unit/data type for temperature;
Same for humidity

Task 3: Design Options

Some interesting options

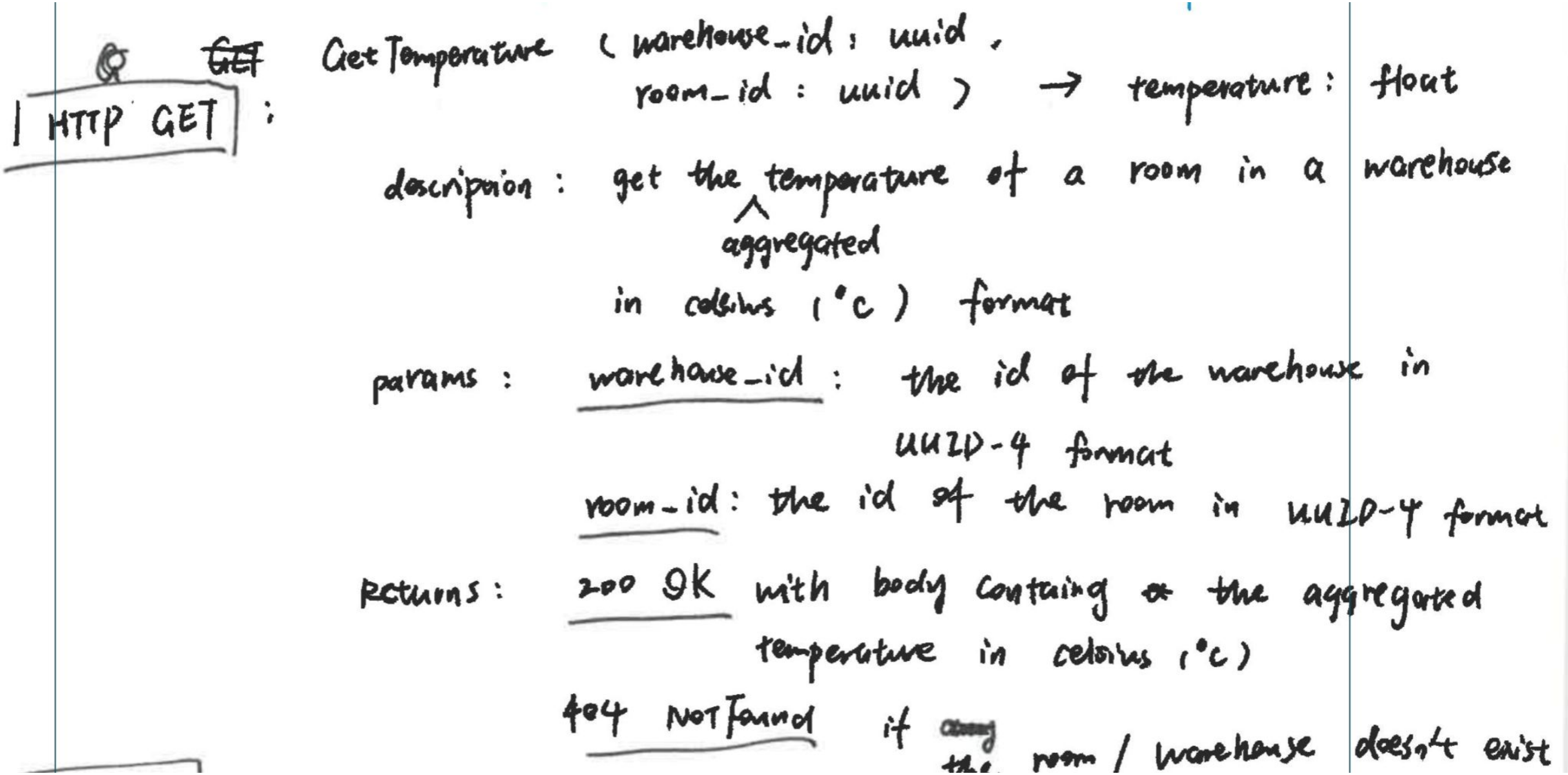
- A separate management system for each room vs. a global sensor/actuator management system
- Poll vs. push notification for temperature/humidity changes
- Splitting the control system into multiple components (e.g., data processor and actuation controller), each with a single responsibility
- Send aggregate data vs. stream all data to the user through mobile app

Task 3: Interface Description

Common mistakes

- Missing semantics (meaning) of interface parameters
- Missing units
  - e.g., temperature in ºC, ºF, or K?
  - Avoid the same mistake as in the Mars Climate Orbiter Failure!
- Answer is not really an interface description

Task 3: Interface Description



~~GET~~ GetTemperature (warehouse_id : uuid,
room_id : uuid) → temperature : float

HTTP GET :

descripion : get the aggregated temperature of a room in a warehouse
in celsius (°c) format

params : warehouse_id : the id of the warehouse in
UUID-4 format
room_id : the id of the room in UUID-4 format

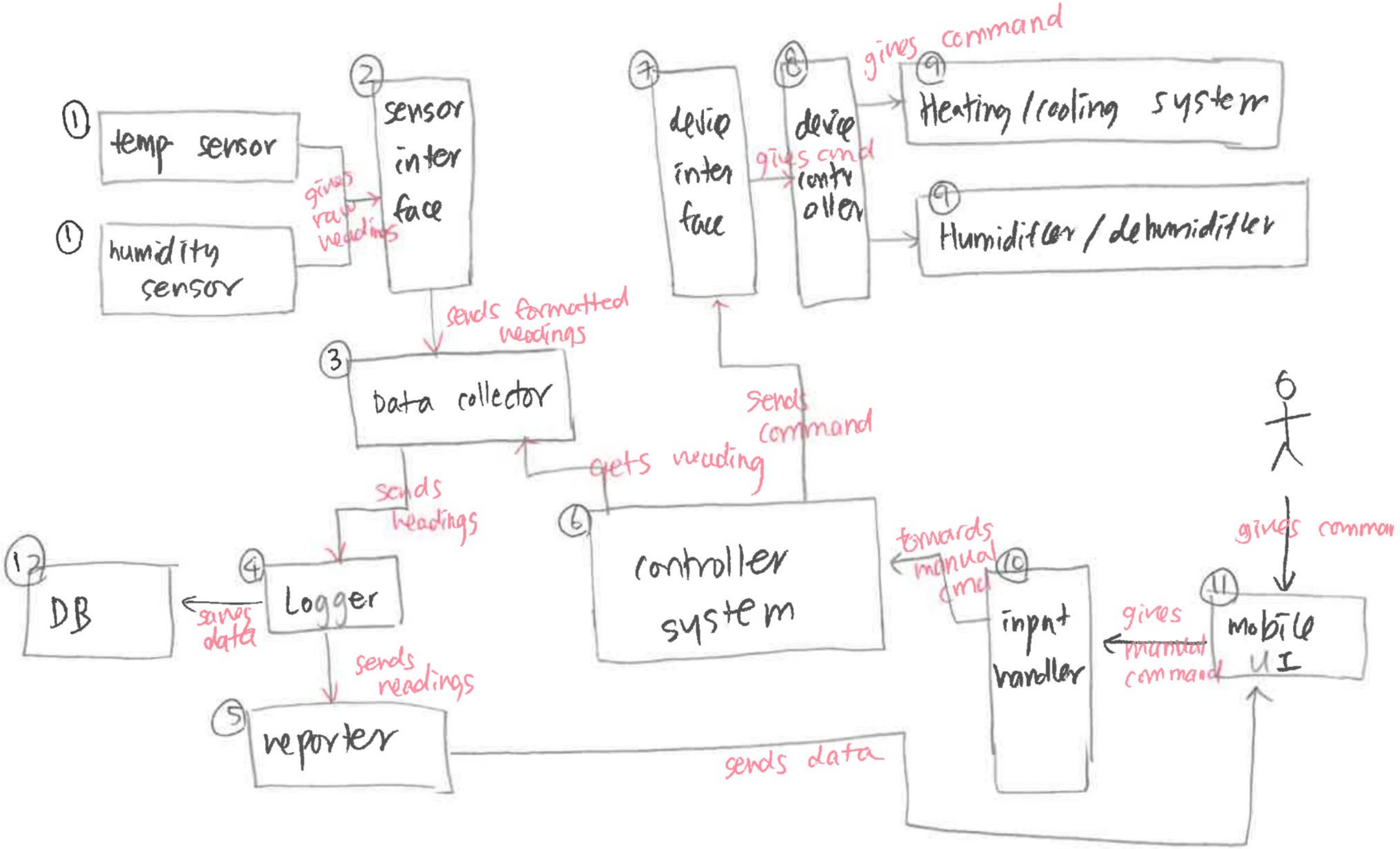
Returns : 200 OK with body containg ~~a~~ the aggregated
temperature in celsius (°c)
404 NotFound if the room / warehouse doesn't exist

Task 3: Component Diagram

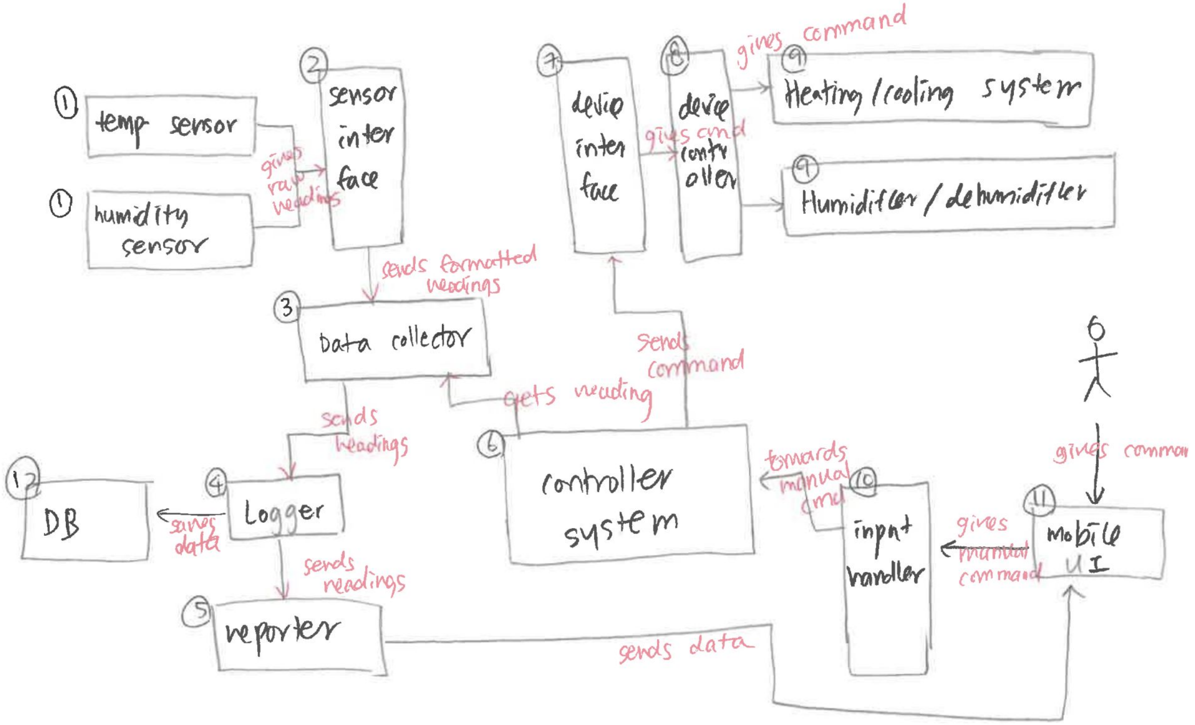
Common mistakes

- Missing components (that show up in the textual design description or in the sequence diagram)
- Missing labels on connections
- Missing responsibility annotation/description

Task 3: Component Diagram



Task 3: Component Diagram



① in charge of measuring environment

② converts the readings into acceptable format

③ collects multiple sensor datas

④ logs the data into JSON/SQL format

⑤ generats report for mobile UI

⑥ gets input from actual sensors & manual and decides what and to send to device interface

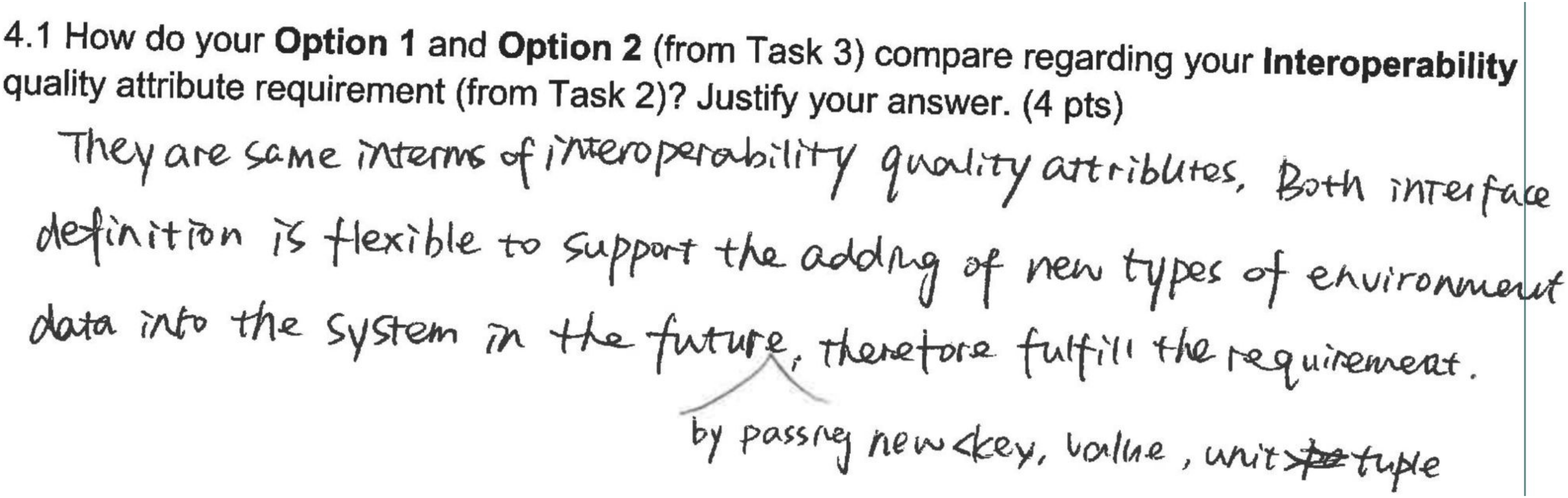
⑫ stores data for future use.

Task 4: Design Comparison

Common mistakes

- Answer refers to a different scenario than the one described in Task 2 (e.g., changeability of adding sensors vs. changeability of the control logic)
- Answer is just rephrasing the quality attribute instead of providing an **argument** for why one is better than the other.

Task 4: Design Comparison



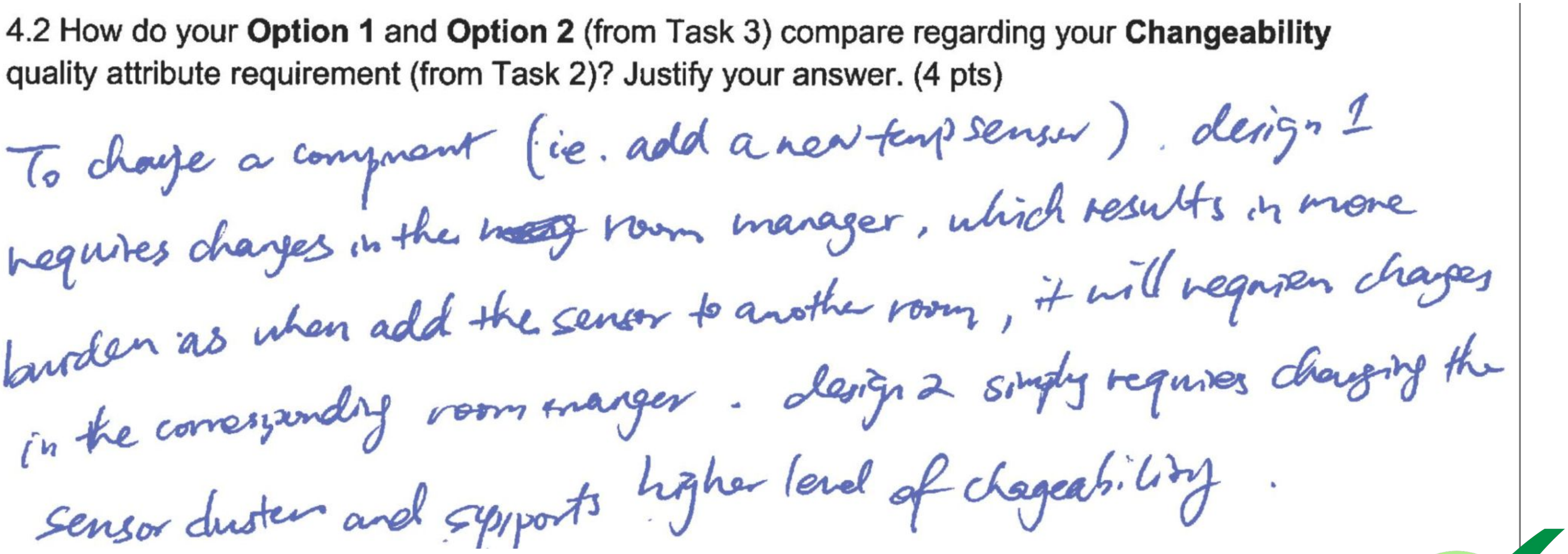
4.1 How do your **Option 1** and **Option 2** (from Task 3) compare regarding your **Interoperability** quality attribute requirement (from Task 2)? Justify your answer. (4 pts)

They are same interms of interoperability quality attributes. Both interface definition is flexible to support the adding of new types of environment data into the system in the future, therefore fulfill the requirement.

by passng new <key, value, unit> tuple

✓

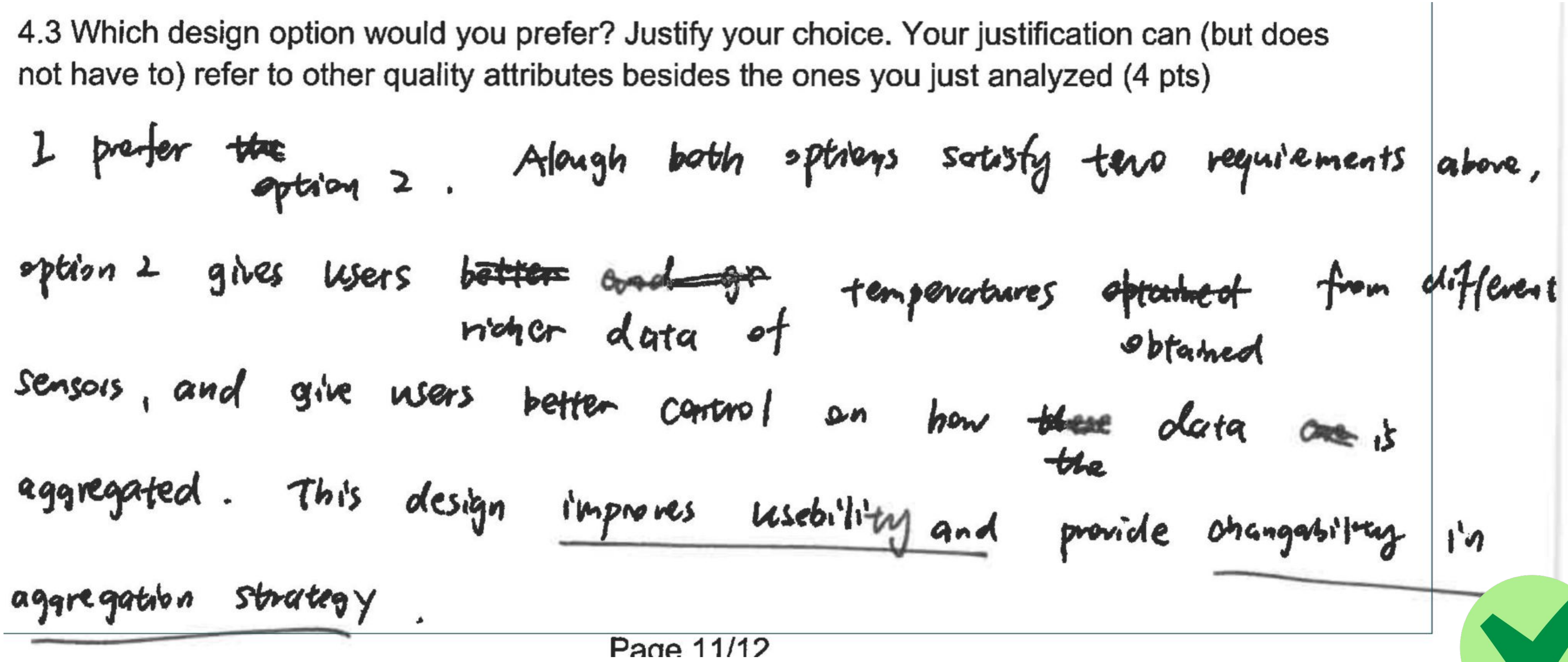
Task 4: Design Comparison



4.2 How do your **Option 1** and **Option 2** (from Task 3) compare regarding your **Changeability** quality attribute requirement (from Task 2)? Justify your answer. (4 pts)

To change a component (ie. add a new temp sensor). design 1 requires changes in the room manager, which results in more burden as when add the sensor to another room, it will require changes in the corresponding room manager. design 2 simply requires changing the sensor cluster and supports higher level of changeability.

Task 4: Design Comparison

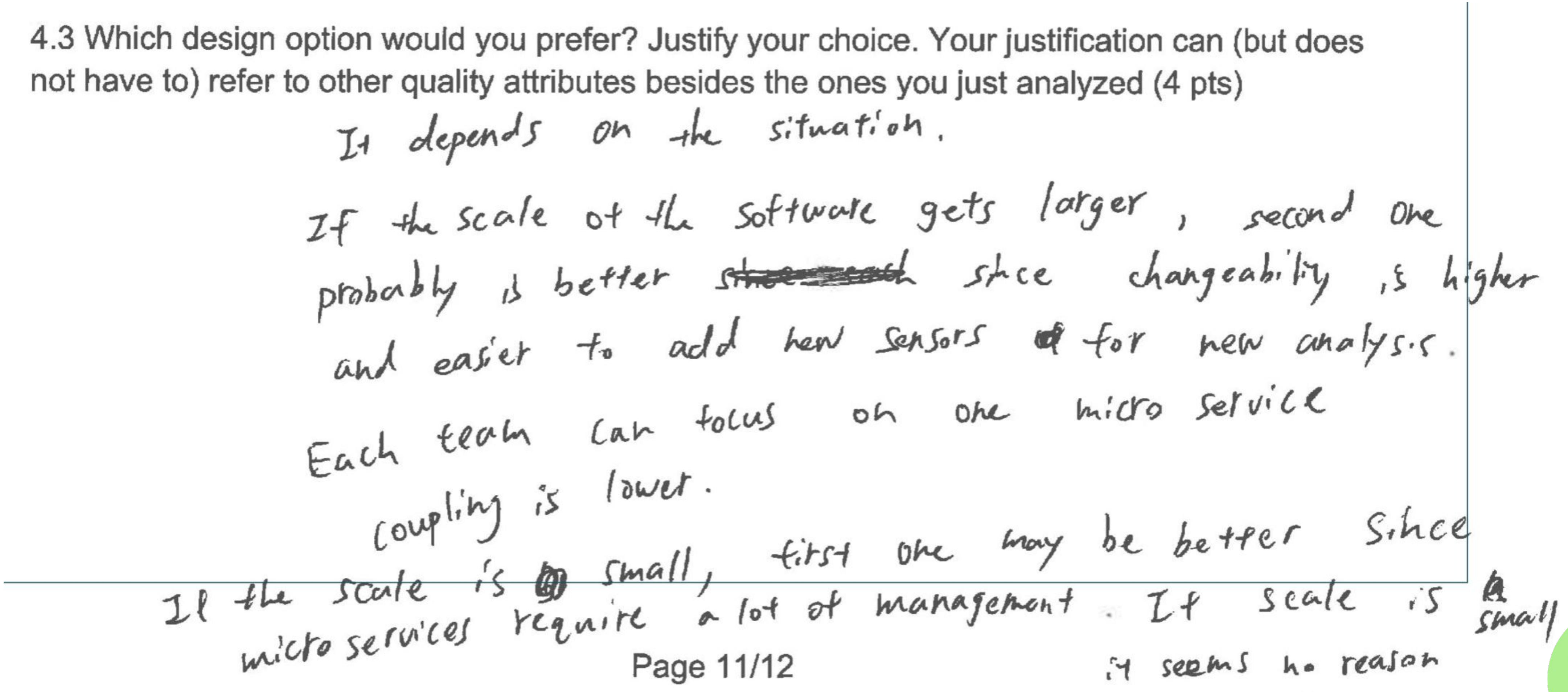


4.3 Which design option would you prefer? Justify your choice. Your justification can (but does not have to) refer to other quality attributes besides the ones you just analyzed (4 pts)

I prefer ~~the~~ option 2. Alough both options satisfy two requirements above, option 2 gives users ~~better~~ ~~and gr~~ richer data of temperatures ~~optained~~ obtained from different sensors, and give users better control on how ~~these~~ the data ~~are~~ is aggregated. This design improves usebility and provide changability in aggregation strategy.

Page 11/12

Task 4: Design Comparison



4.3 Which design option would you prefer? Justify your choice. Your justification can (but does not have to) refer to other quality attributes besides the ones you just analyzed (4 pts)

It depends on the situation.

If the scale of the software gets larger, second one probably is better since changeability is higher and easier to add new sensors for new analysis.

Each team can focus on one micro service

coupling is lower.

If the scale is small, first one may be better since micro services require a lot of management. If scale is small it seems no reason

Page 11/12