**Step-by-Step Guide for Ontology Modeling (G3-C1)**

| **Step** | **Action** | **Example/Code** | **Purpose** |
| --- | --- | --- | --- |
| **1. Create Classes** | Define WearableButton, TactileFeedback, and WearableDevice. | **In Protégé:** - Go to Classes → Add Subclass. - Create WearableButton (under ButtonComponent). - Create TactileFeedback (under Feedback). - Create WearableDevice (under DevicePlatform). | Model wearable-specific buttons and feedback mechanisms. |
| **2. Define Properties** | Create object/data properties for relationships. | **Object Property:** - hasFeedbackMechanism (domain: WearableButton, range: TactileFeedback). **Data Property:** - hasHapticOption (domain: TactileFeedback, range: xsd:boolean). | Link buttons to feedback mechanisms with haptic support. |
| **3. Add Restriction to**WearableButton | Enforce that wearable buttons must use tactile feedback with haptic enabled. | **Manchester Syntax:** hasFeedbackMechanism some (TactileFeedback and (hasHapticOption some boolean[= "true"^^xsd:boolean]) | Ensure wearable buttons have haptic-enabled tactile feedback. |
| **4. Create Instances** | Add a wearable button and feedback instance. | **Turtle Syntax:** :SmartwatchButton a :WearableButton ; :hasFeedbackMechanism :HapticVibration . :HapticVibration a :TactileFeedback ; :hasHapticOption true . | Test ontology consistency and reasoning. |
| **5. Run Reasoner** | Validate the ontology. | **Steps:** 1. Go to Reasoner → HermiT. 2. Click Start Reasoner. | Check for inconsistencies (e.g., missing haptic feedback). |
| **6. SPARQL Query** | Retrieve compliant wearable buttons. | **SPARQL Query:** sparql<br> SELECT ?button ?feedback<br> WHERE {<br> ?button a :WearableButton ;<br> :hasFeedbackMechanism ?feedback .<br> ?feedback :hasHapticOption true .<br> }<br> | Verify wearable buttons with valid haptic feedback. |

**Key Takeaways**

1. **Haptic Feedback**: Enforced via TactileFeedback with hasHapticOption true.
2. **Device Context**: Buttons are restricted to WearableDevice platforms.
3. **SPARQL Use Case**: Efficiently retrieves wearable components with compliant feedback.