Ontology Management System

Md Rofiqul Islam
Nurul Karim Rafi
Sarjan Kabir

02 October 2020

1 Project Vision

The vision of Ontology Management System (OMS) is to enable special type of users called experts to build an ontology of all items of information regarding the construction and design of buildings, where every piece of information is connected with one another.

2 Requirements

Functional Requirements

1. Experts will be able to build ontologies.
2. Experts will be able to add nodes.
3. Experts will be able to delete nodes.
4. Experts will be able to modify nodes.
5. Experts will be able to add edges.
6. Experts will be able to delete edges.
7. Experts can change properties to nodes and edges.
8. Each category should define the type of a node or edge.
9. User will be able to login to the system with appropriate credentials.
10. System should check if user has access as 'expert'.
11. User can change password for logging into the system.
12. User can get new password if they forget their current password.
13. System will allow only directed acyclic graph compliant ontologies.
14. Web-socket must be implemented for giving access to databases to multiple experts.

Non-functional Requirements

1. Authentication through user login.
2. System should handle concurrent operation of multiple experts.
3. Ontology must be Directed Acyclic Graph.
4. Creation process of node, edge and full ontology should be designed for only expert users.
5. User should need special training to become expert user.
6. Session recovery.

3 Use Cases

Since the main 3 modules of OMS are : node, edge, and ontology, we have divided the Use cases related to each module to each group member respectively.

- Nurul Karim Rafi:
  Use cases and other artifacts related to Nodes: UC1, UC2, UC3, UC4
- Md Rofiqul Islam:
  Use cases and other artifacts related to Edges: UC5, UC6, UC7, UC8, UC13
- Sarjan Kabir:
  Use cases and other artifacts related to Ontology: UC9, UC10, UC11, UC12

Use Case UC1: Create Node Category

Primary actor
Expert

Stakeholders and Interests
-Expert : wants to create a category for node.

Preconditions
-The expert should be logged in the system.

Success Guarantee (Postconditions)
- Expert successfully create the node category.

Main Success Scenario
1. Expert defines the name of the node category.
2. Add the type for node category.

Extensions
1.a. If already type is saved in the node category then system shows an alert
1.b. System checks for valid type for node category.

Use Case UC2: Create Node

Primary actor
Expert

Stakeholders and Interests
-Expert : wants to create node

Preconditions
-Predefined node category and other property should be defined. -The expert should be logged in the system.

Success Guarantee (Postconditions)
-Expert can create a node.

Main Success Scenario
1. Expert set the name of the node.
2. Define the type or category of the node.
3. Define the other properties for that node.

Extensions
1.a System shows alert if valid name is not given
2.a. Define type which is not present then system shows the alert and available types
3.a. Define invalid property which is not present then system shows alert and available properties.

Use Case UC3: Modify Node

Primary actor
Expert

Stakeholders and Interests
-Expert : Wants to modify existing node.

Preconditions
-There should be a predefined node in the system.
-The expert should be logged in the system.

Success Guarantee (Postconditions)
- Successfully modify the node

Main Success Scenario
1. Expert change the name of the node.
2. Expert change the type of the node.
3. Change the existing other properties.
4. Add other properties to the existing list.

Extensions
1.a. System shows alert if already used name is given for the node.
1.b. Alert for invalid name.
2.a. Shows alert for invalid changed type.
4.a. Show alert for invalid other properties.

4.b. If modification doesn't happen then node rolls back to the previous position.

Use Case UC4: Delete Node

Primary actor
Expert

Stakeholders and Interests
-Expert: wants to delete existing node.
who is using this system

Preconditions
-The expert should be logged in the system. Node that expert wants to delete must exist.

Success Guarantee (Postconditions)
- Successfully delete the node

Main Success Scenario
1. Expert delete the node.

Extensions
1.a. If delete is not happened then ontology remains in it's previous condition. 1.b. Corresponding edge connections will be removed.

Use Case UC5: Create Edge Category

- Primary Actor: Expert
- Stakeholder and Interests:
  Expert: Wants to create category for edge.
- Preconditions: Expert must be logged in to the system.
- Success Guarantee : Successfully created category for edges.
- Success Scenario:
  1. Expert enters label for category.
  2. New category is created.
- Extensions:
  2. a. Edge cannot be created if same category with same label already exists.

Use Case UC6: Create Edge between nodes

- Primary Actor: Expert
- Stakeholder and Interests:
  Expert: Wants to create edge between nodes.
- Preconditions: Expert must be logged in to the system. Nodes must exist for which user wants to create edge.Edge categories must be predefined.

- Success Guarantee : Successfully created edge between nodes.
- Success Scenario:
  1. Expert selects ”From” node.
  2. Expert selects ”To” node.
  3. Expert selects edge category.
  4. Expert saves the information.
  5. Edge is created .
- Extensions:
  5. a. Edge is not created as it violated DAG compliant graph.

Use Case UC7: Modify Edge

- Primary Actor: Expert
- Stakeholder and Interests:
  Expert: Wants to modify edge between nodes.
- Preconditions: Expert must be logged in to the system. Nodes must exist for which user wants to modify edge .Edge categories must be predefined.
- Success Guarantee : Successfully modified edge between nodes.
- Success Scenario:
  1. Expert modifies ”To” node.
  2. Expert modifies ”From” node.
  3. Expert modifies edge category from list.
  4. Expert saves the information.
  5. Edge is modified .
- Extensions:
  5. a. Edge is not created as it violated DAG compliant graph.

Use Case UC8: Delete Edge

- Primary Actor: Expert
- Stakeholder and Interests:
  Expert: Wants to delete edge between nodes.
- Preconditions: Expert must be logged in to the system. Nodes must exist for which user wants to delete edge.
- Success Guarantee : Successfully deleted edge between nodes.
- Success Scenario:
  1. Expert selects edge which he wants to delete.
  2. Edge is deleted.
- Extensions:
  2. a. Edge deletion can be unsuccessful due to concurrent modification.

Use case UC9 : Create Ontology

- Primary Actor: Expert
- Stakeholder and Interests:
  Expert: Wants to connect pre - built nodes and edges. Wants to build ontology (which is a directed acyclic graph consisting of nodes, edges, and properties) of all items of information regarding the design of buildings.
- Preconditions: Nodes of particular category and edges have been created.
  Each node and edge has been assigned properties.
- Success Guarantee : Successfully created Ontology that is DAG (directed acyclic graph) compliant.
- Success Scenario:
  1. Expert creates nodes with predefined node category.
  2. Expert creates edges with predefined edge category.
  3. Expert connects edge to nodes, by specifying "From" node to "To" node.
  4. A DAG compliant ontology is created.
- Extensions:
  2. a. Expert can change orientation of the ontology.
  3. a. A DAG compliant ontology may not be created. System will give error.
  b. Expert will change the edge orientation to change the ontology to make it DAG complaint.

Use Case UC10: Modify Ontology

- Primary Actor: Expert
- Stakeholder and Interests:
  Expert: Wants to modify ontology that remains DAG compliant.
- Preconditions: The ontology that the expert wants to modify must exist in the system.
- Success Guarantee : Successfully modified existing Ontology that is DAG (directed acyclic graph) compliant.
- Success Scenario:
  1. Expert selects preexisting ontology.
  2. Expert modifies node(s).
  3. Expert modifies edges(s).
  4. Expert connects edge to nodes, by specifying "From" node to "To" node.
  5. The ontology is modified , which remains DAG compliant.
- Extensions:
  In any time, if some other user is modifying the ontology, system will notify current user. 2. a. Expert can add a new node.
  2. b. Expert can delete a node.
  3. a. Expert can add a new edge.
  3. b. Expert can delete an edge.
  4. a. Expert can change orientation of the edge.

Use Case UC11: Delete Ontology

- Primary Actor: Expert
- Stakeholder and Interests:
  Expert: Wants to delete ontology that preexists.
- Preconditions: The ontology that the expert wants to delete must exist in the system.
- Success Guarantee : Successfully deleted existing Ontology.
- Success Scenario:
  1. Expert selects an ontology from a list of ontologies.
  2. Expert deletes the ontology.
  3. System asks expert to confirm if he wants to delete.
  4. Expert confirms that he wants to delete the ontology.
  5. Ontology is deleted.
- Extensions:
  5. a. Ontology deletion can be unsuccessful due to any concurrent modification.

Use Case UC12: View Ontology

- Primary Actor: Expert
- Stakeholder and Interests:
  Expert: Wants to view ontology that preexists.
- Preconditions: The ontology that the expert wants to view must exist in the system.
- Success Guarantee : Successfully view existing Ontology.
- Success Scenario:
  1. Expert selects an ontology from a list of ontologies.
  2. Expert views the ontology.
- Extensions:
  2. a . The ontology is not displayed due to some error.

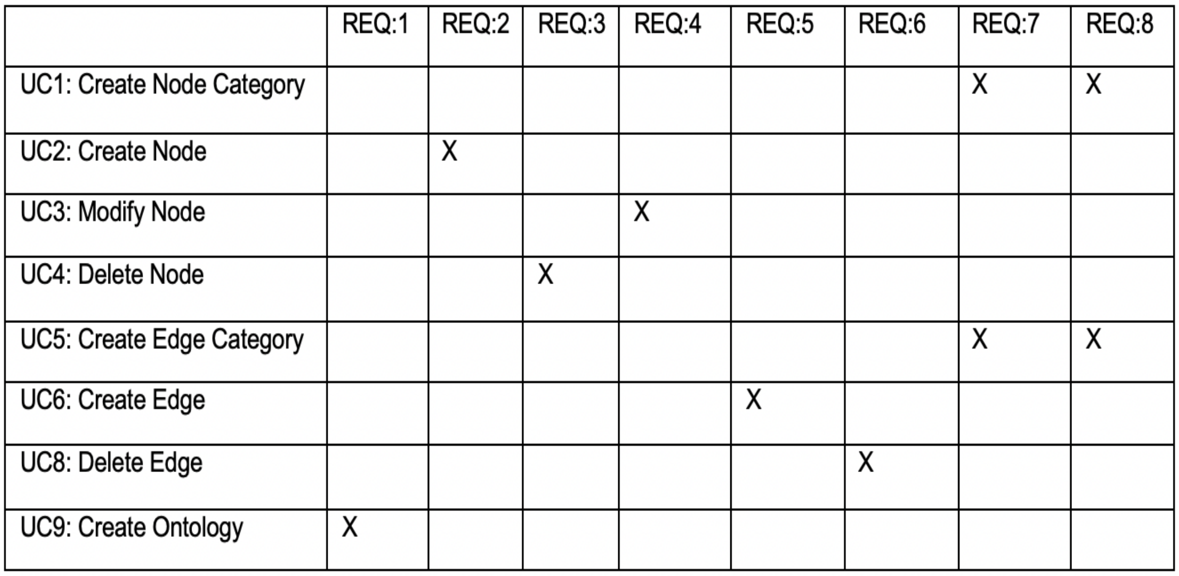
Use Case UC13: Create Other Property

- Primary Actor: Expert
- Stakeholder and Interests:
  Expert: Wants to create property which will later be used in nodes and edges.
- Preconditions: Expert must be logged in to the system
- Success Guarantee : Successfully created property.
- Success Scenario:

1. Expert starts to create a property.
2. Expert fills in id, name and type for the property.
3. Expert saves the information.
4. Property is created.

- Extensions:
  2. a . Property name may already exist. Expert cannot create property with existing name.

4 Traceability Matrix

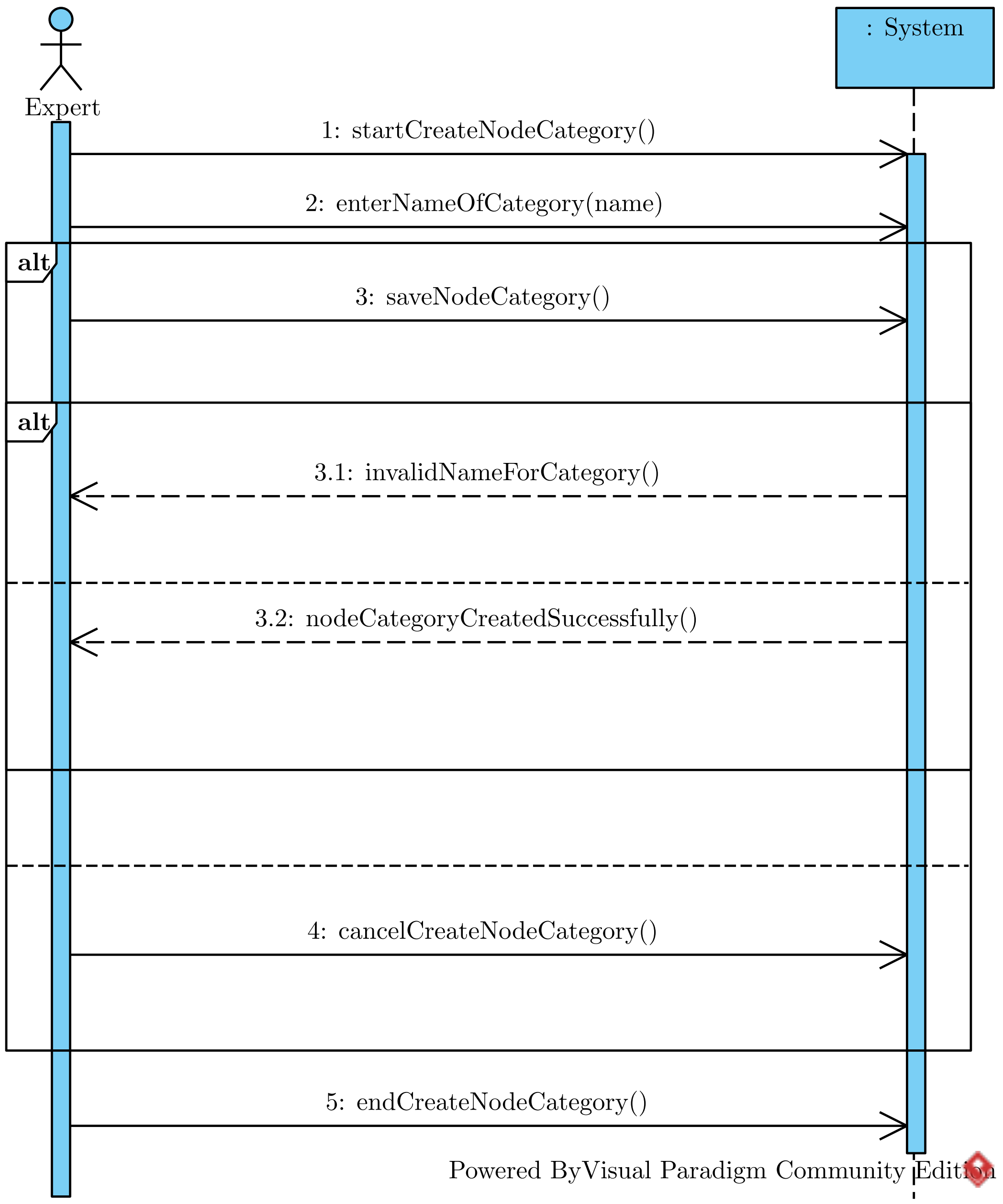


	REQ:1	REQ:2	REQ:3	REQ:4	REQ:5	REQ:6	REQ:7	REQ:8
UC1: Create Node Category							X	X
UC2: Create Node		X						
UC3: Modify Node				X				
UC4: Delete Node			X					
UC5: Create Edge Category							X	X
UC6: Create Edge					X			
UC8: Delete Edge						X		
UC9: Create Ontology	X							

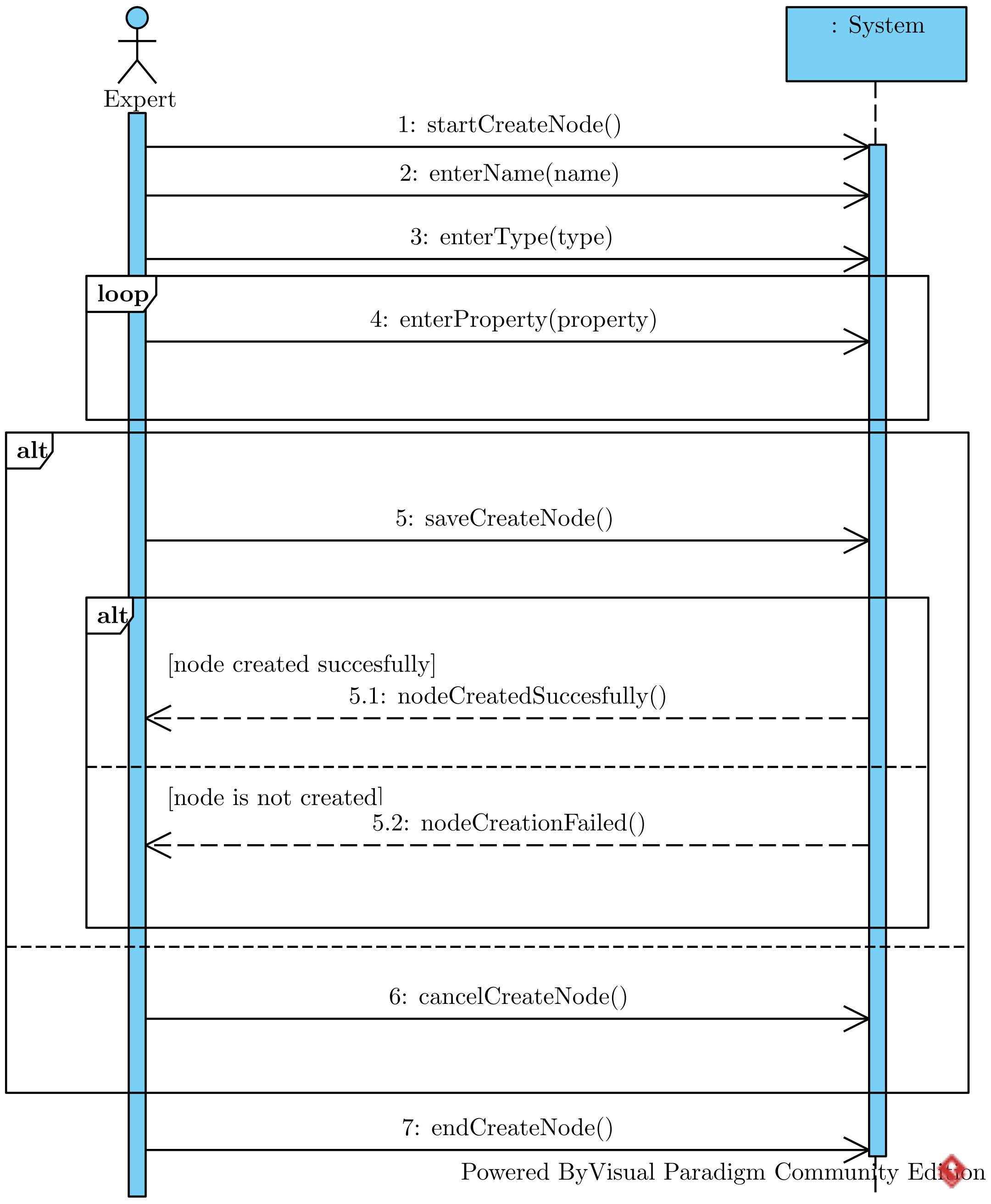
Requirements 9-14 are not actually represented in any use cases. They will be implemented in the project and handled there.

5 System Sequence Diagrams

Create Node Category

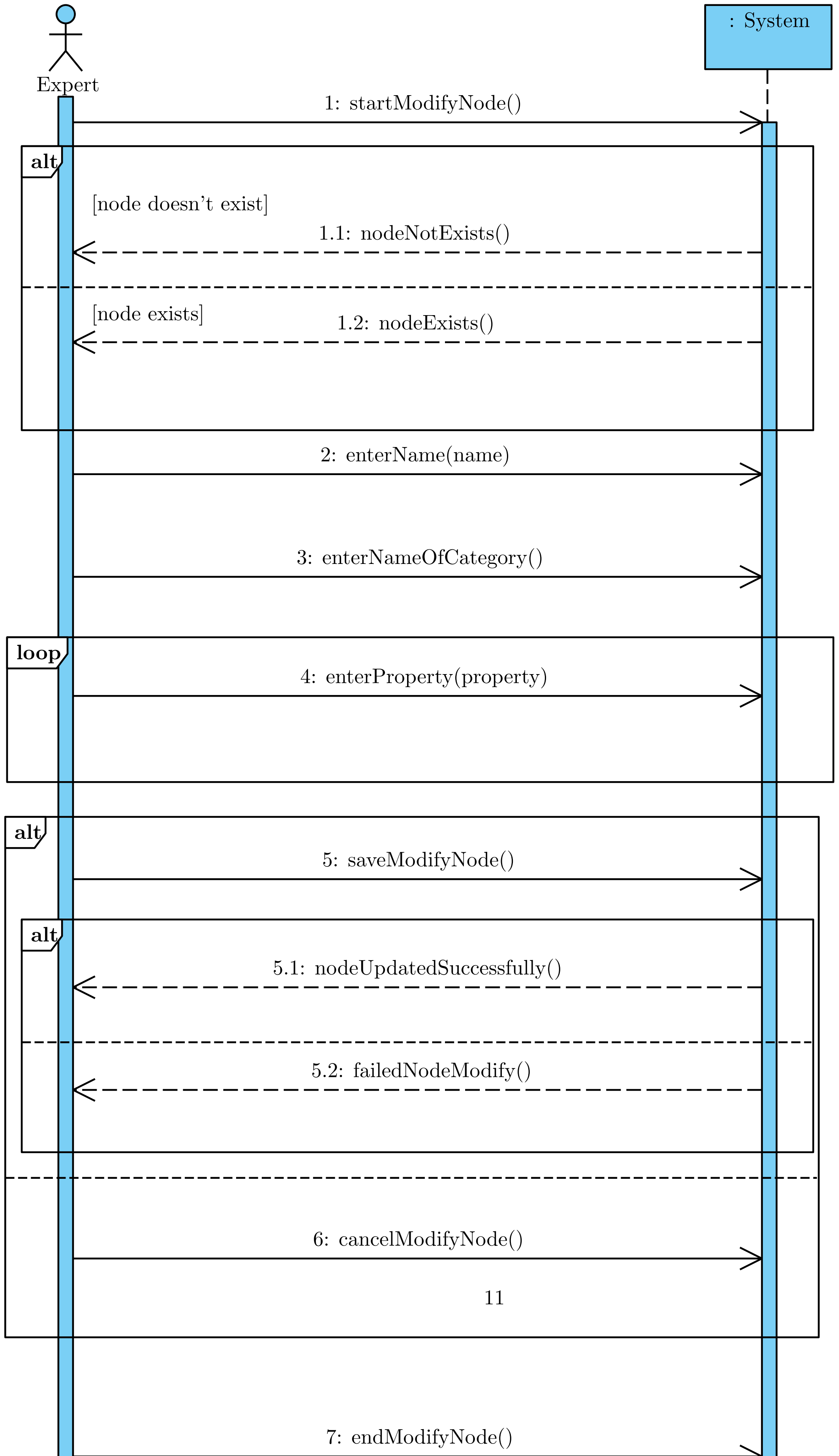

Expert
: System
1: startCreateNodeCategory()
2: enterNameOfCategory(name)
alt
3: saveNodeCategory()
alt
3.1: invalidNameForCategory()
3.2: nodeCategoryCreatedSuccessfully()
4: cancelCreateNodeCategory()
5: endCreateNodeCategory()
Powered ByVisual Paradigm Community Edition

Create Node



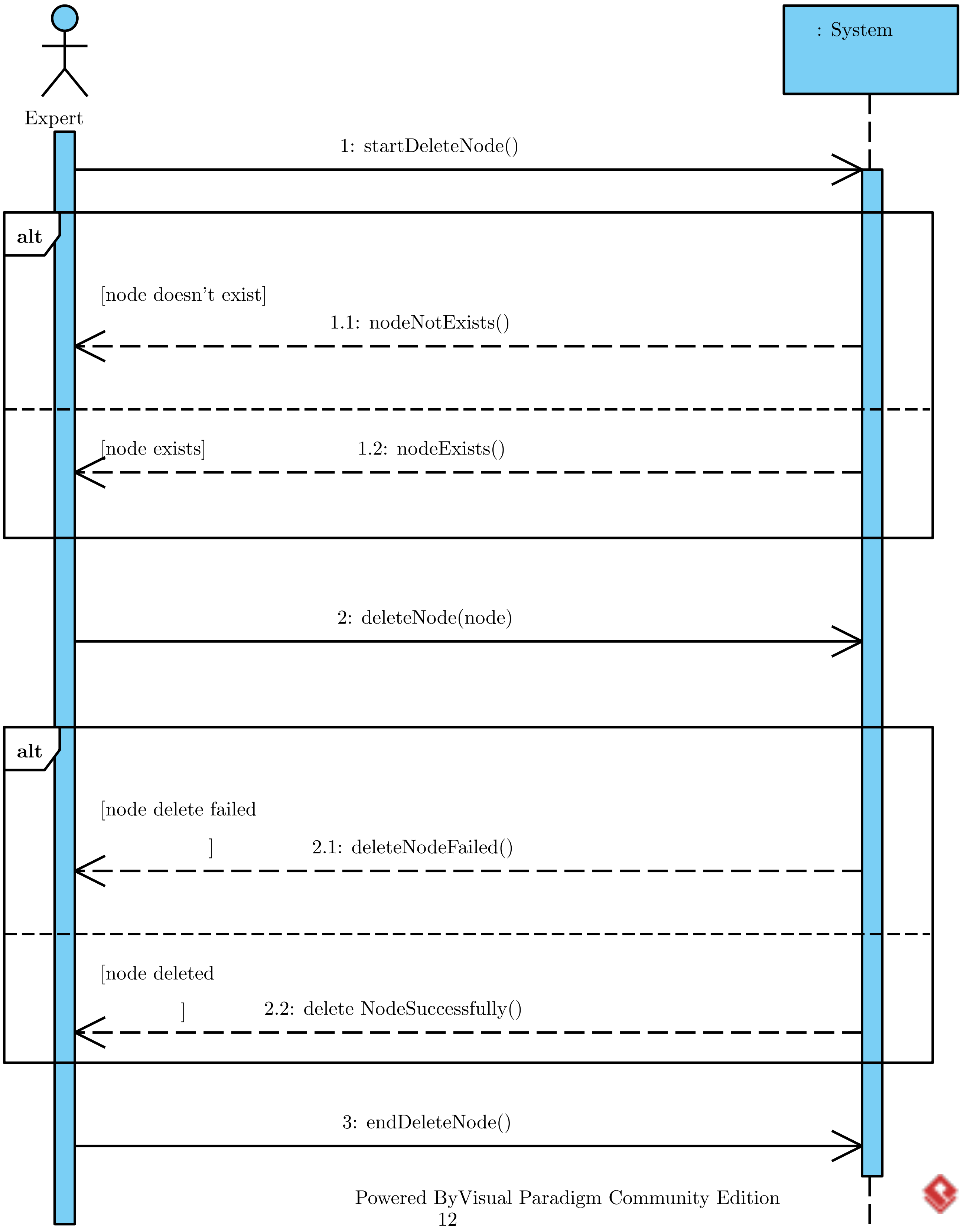
Expert
: System
1: startCreateNode()
2: enterName(name)
3: enterType(type)
loop
4: enterProperty(property)
alt
5: saveCreateNode()
alt
[node created succesfully]
5.1: nodeCreatedSuccesfully()
[node is not created]
5.2: nodeCreationFailed()
6: cancelCreateNode()
7: endCreateNode()
Powered ByVisual Paradigm Community Edition

Modify Node



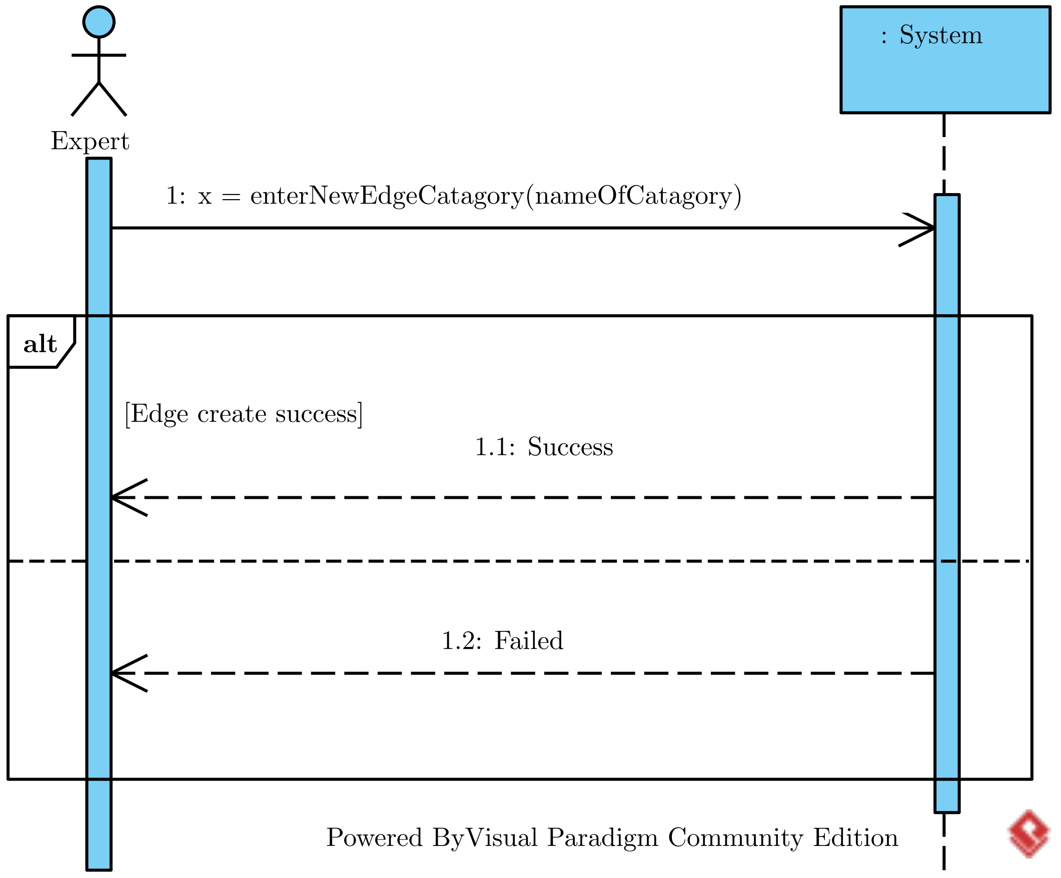
Expert
: System
1: startModifyNode()
alt
[node doesn't exist]
1.1: nodeNotExists()
[node exists]
1.2: nodeExists()
2: enterName(name)
3: enterNameOfCategory()
loop
4: enterProperty(property)
alt
5: saveModifyNode()
alt
5.1: nodeUpdatedSuccessfully()
5.2: failedNodeModify()
6: cancelModifyNode()
7: endModifyNode()

Delete Node

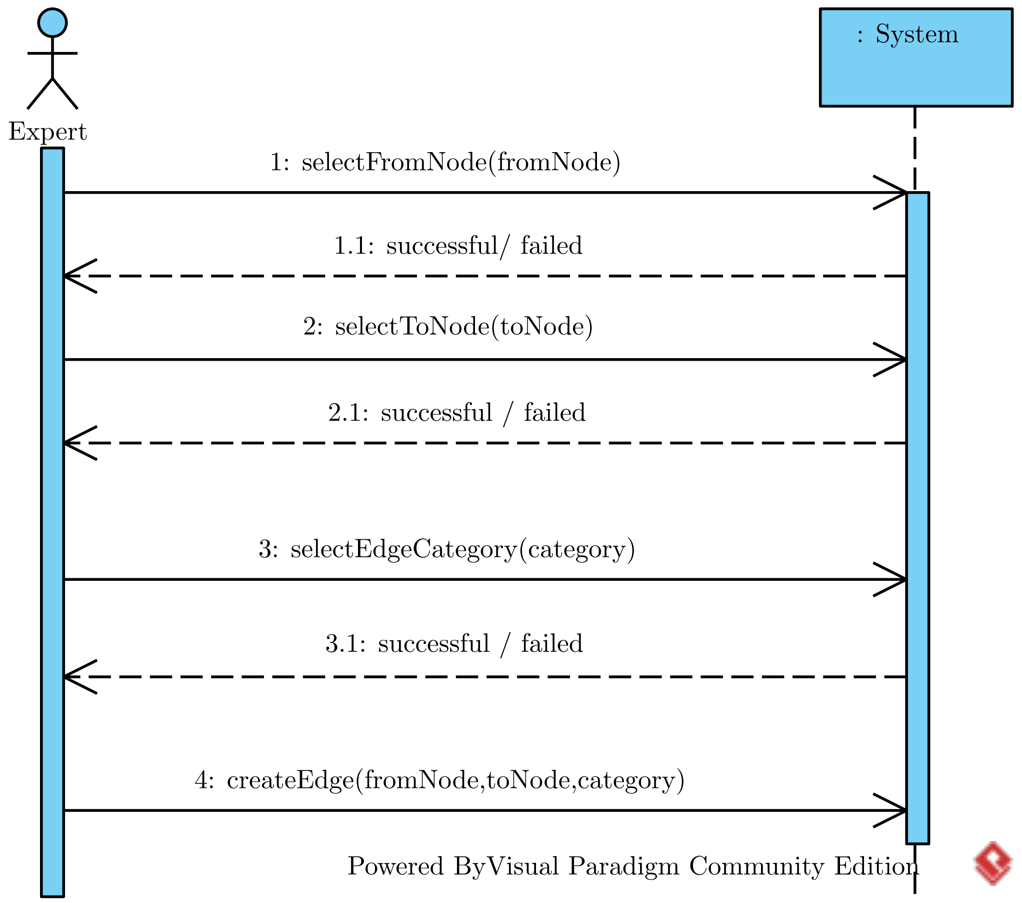


Powered ByVisual Paradigm Community Edition

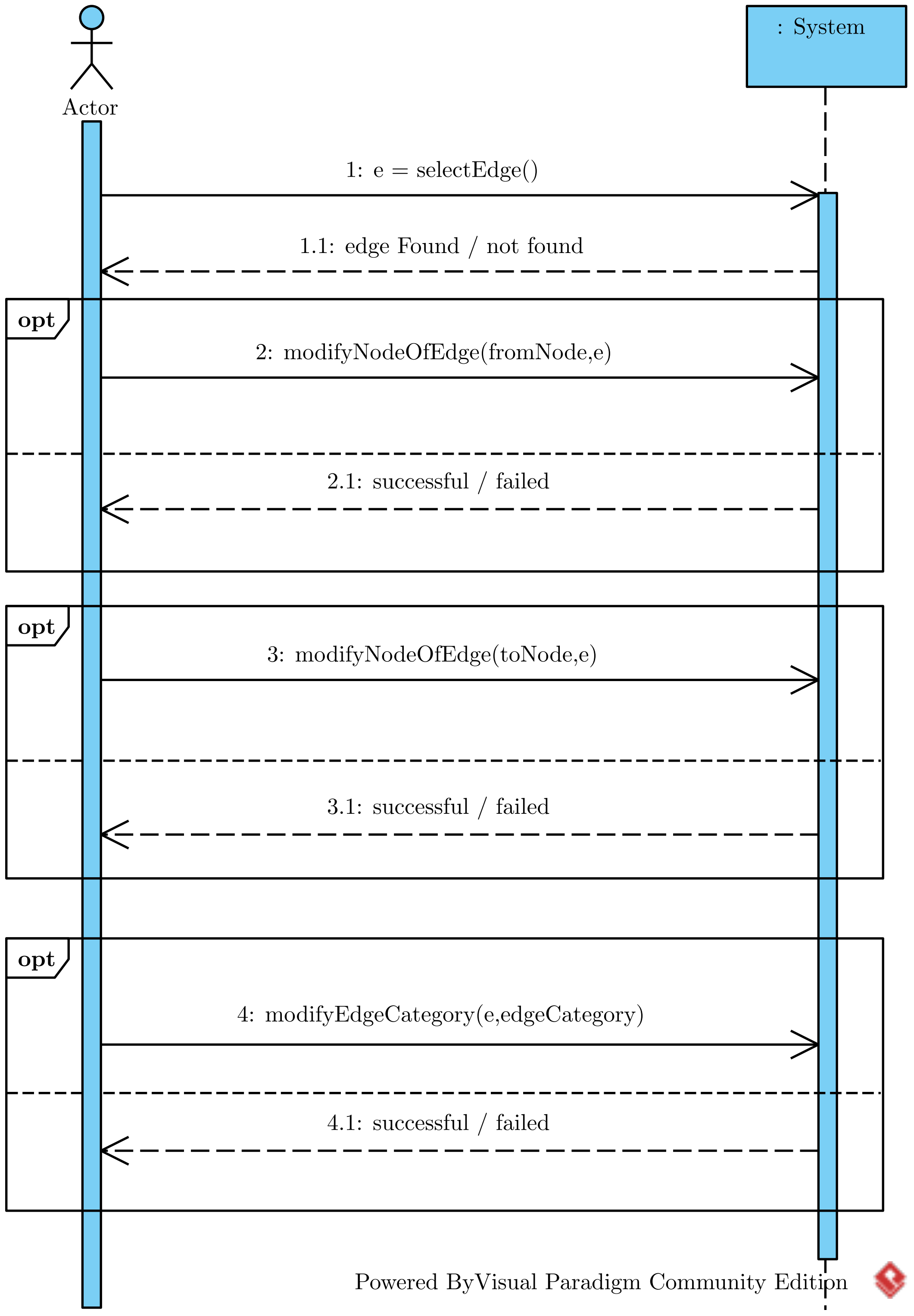
Create Edge Category


Expert
: System
1: x = enterNewEdgeCatagory(nameOfCatagory)
alt
[Edge create success]
1.1: Success
1.2: Failed
Powered ByVisual Paradigm Community Edition

Create Edge

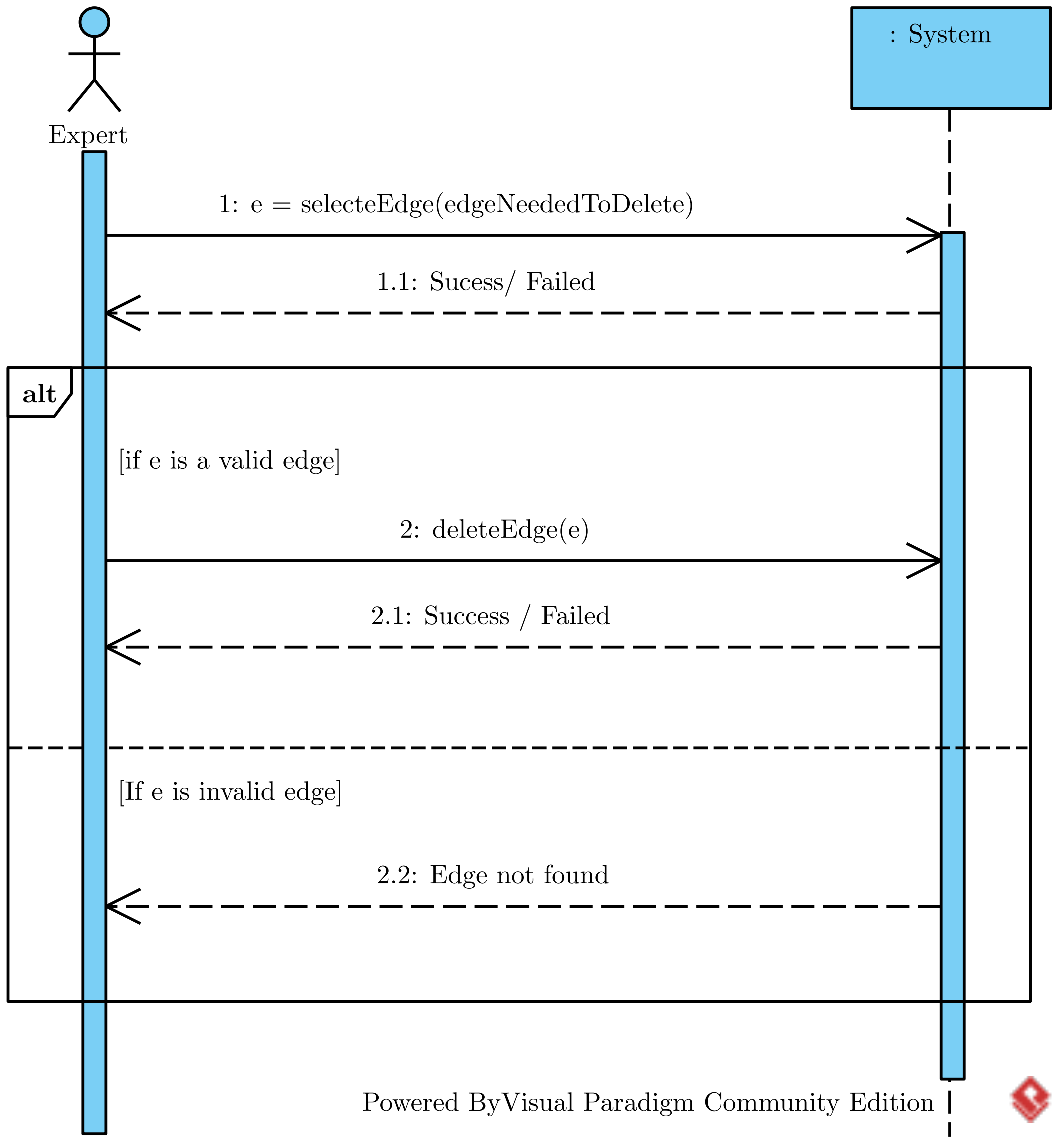

Expert
: System
1: selectFromNode(fromNode)
1.1: successful/ failed
2: selectToNode(toNode)
2.1: successful / failed
3: selectEdgeCategory(category)
3.1: successful / failed
4: createEdge(fromNode,toNode,category)
Powered ByVisual Paradigm Community Edition

Modify Edge

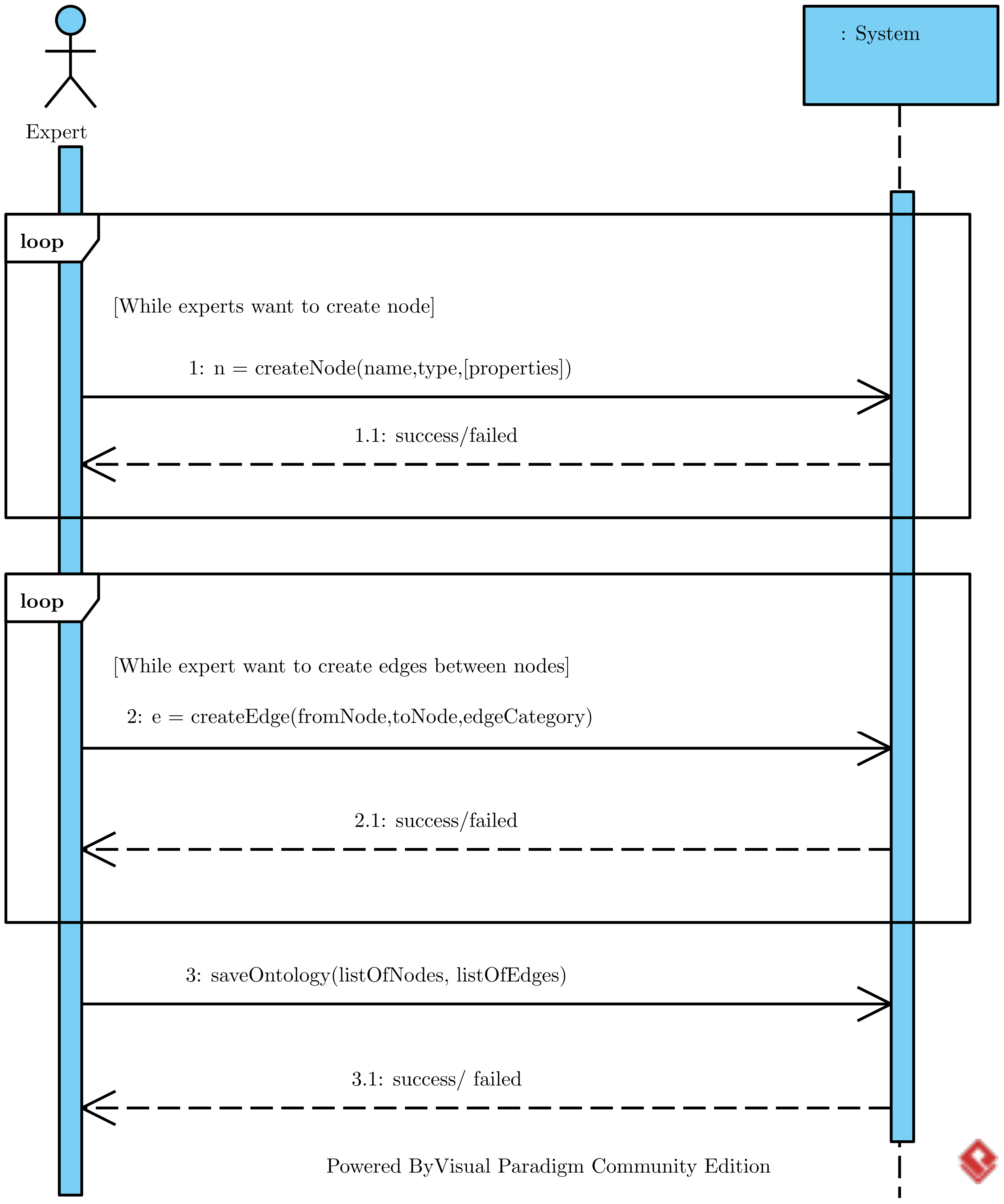


Actor
: System
1: e = selectEdge()
1.1: edge Found / not found
opt
2: modifyNodeOfEdge(fromNode,e)
2.1: successful / failed
opt
3: modifyNodeOfEdge(toNode,e)
3.1: successful / failed
opt
4: modifyEdgeCategory(e,edgeCategory)
4.1: successful / failed
Powered ByVisual Paradigm Community Edition

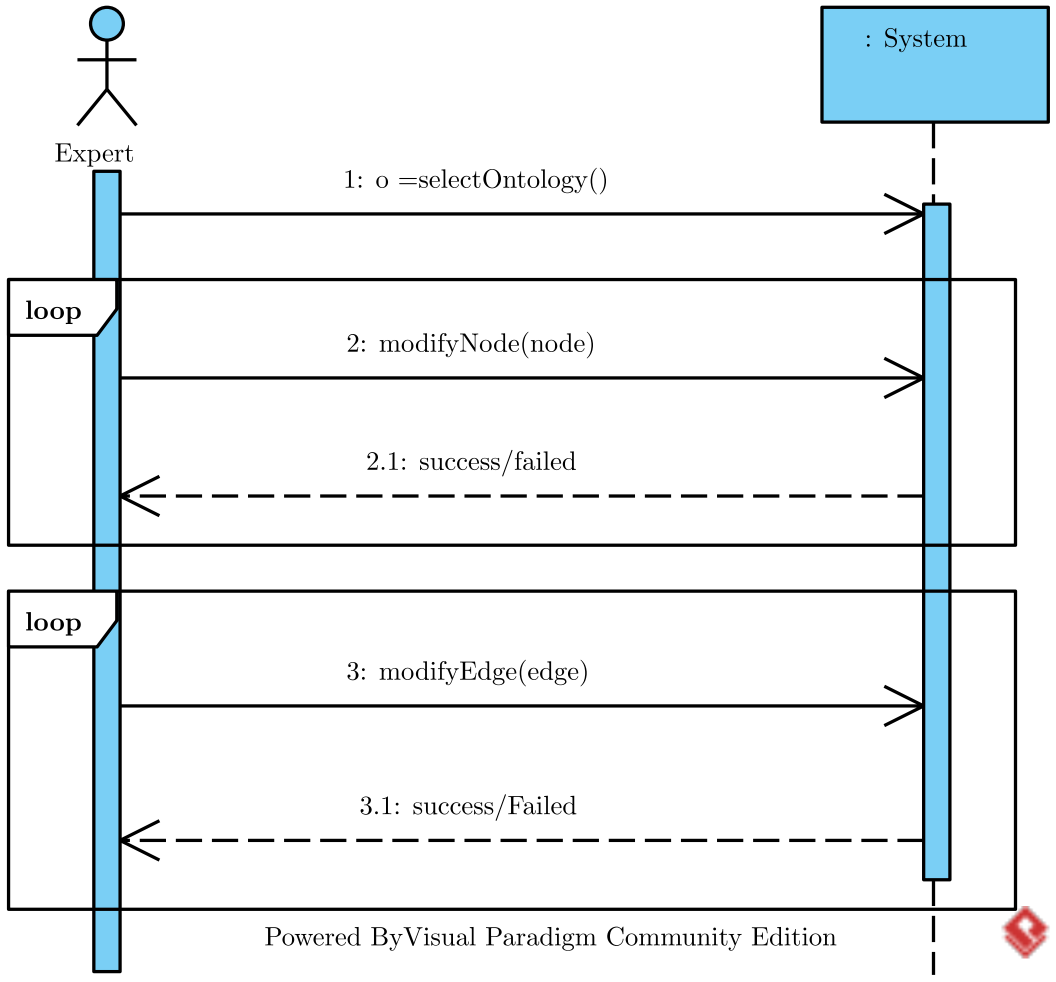
Delete Edge


Expert
: System
1: e = selecteEdge(edgeNeededToDelete)
1.1: Sucess/ Failed
alt
[if e is a valid edge]
2: deleteEdge(e)
2.1: Success / Failed
[If e is invalid edge]
2.2: Edge not found
Powered ByVisual Paradigm Community Edition

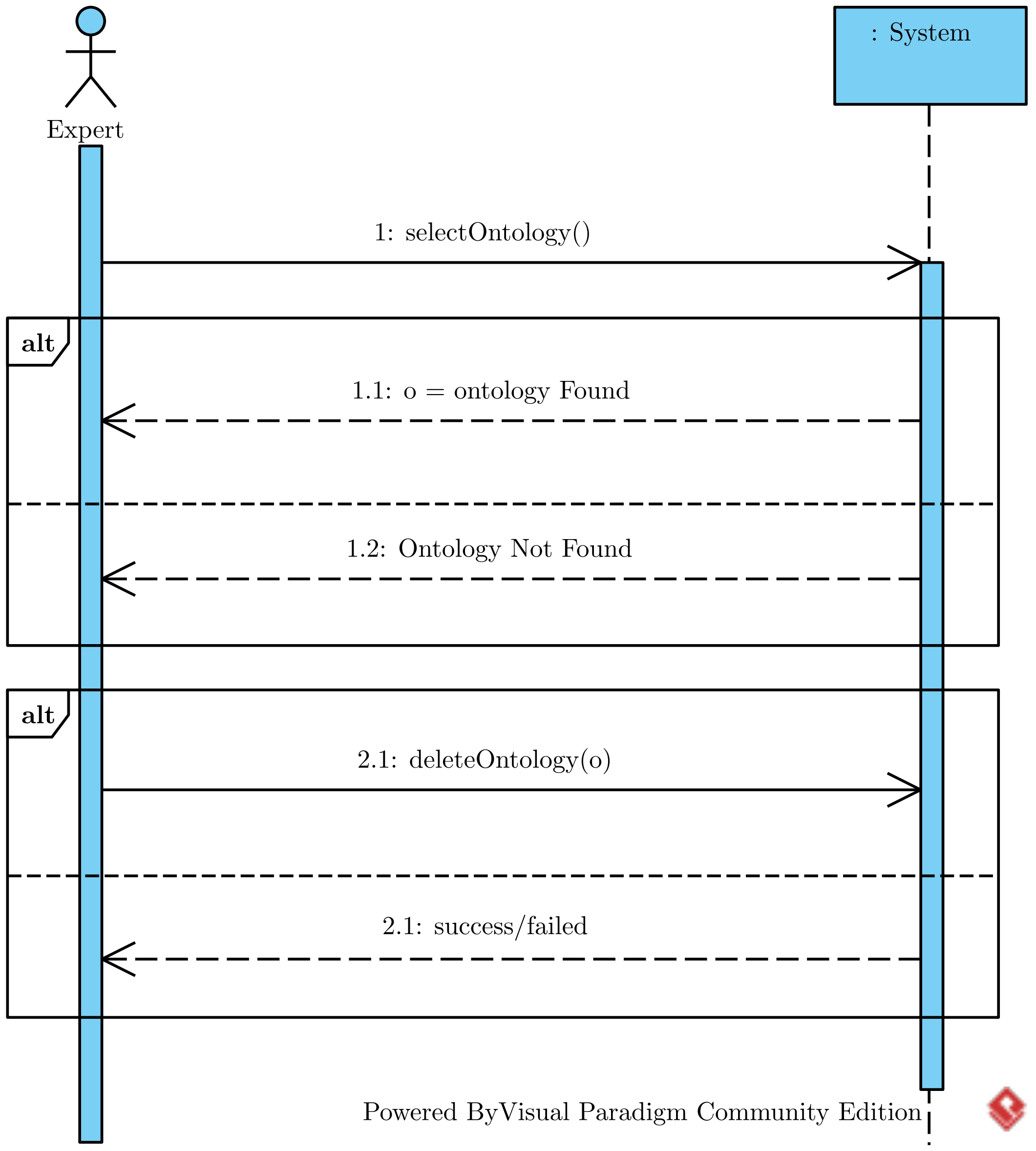
Create Ontology


Expert
: System
loop
[While experts want to create node]
1: n = createNode(name,type,[properties])
1.1: success/failed
loop
[While expert want to create edges between nodes]
2: e = createEdge(fromNode,toNode,edgeCategory)
2.1: success/failed
3: saveOntology(listOfNodes, listOfEdges)
3.1: success/ failed
Powered ByVisual Paradigm Community Edition

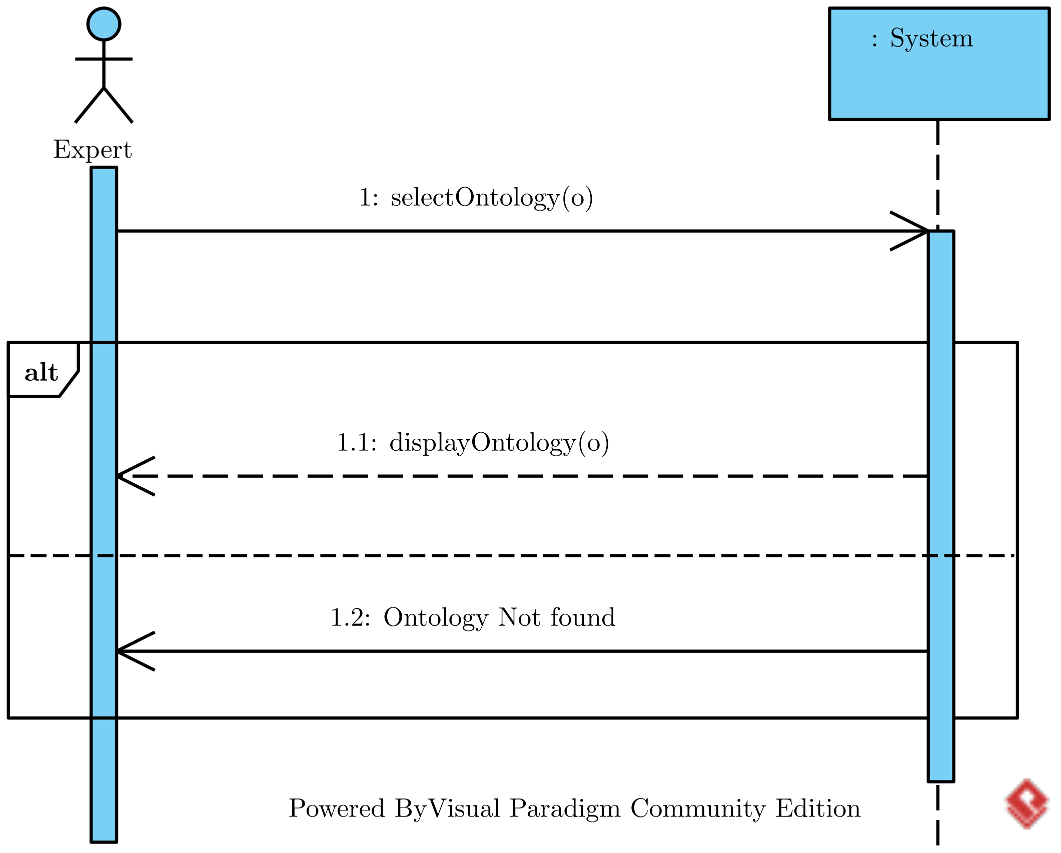
Modify Ontology



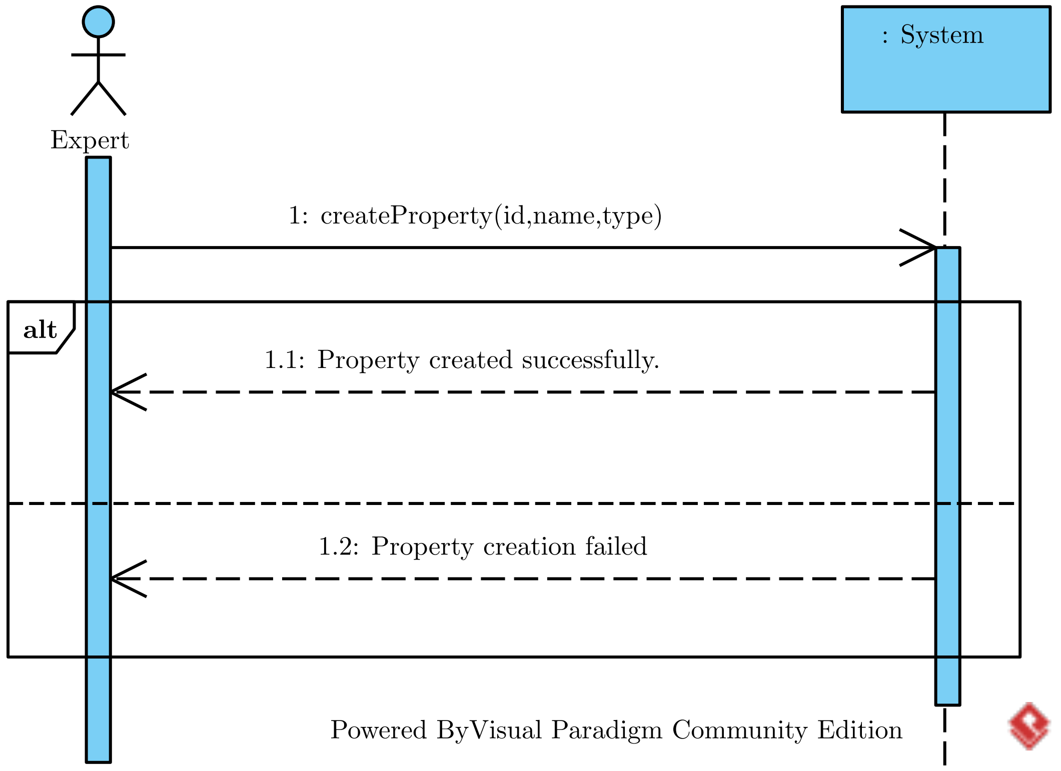
Delete Ontology



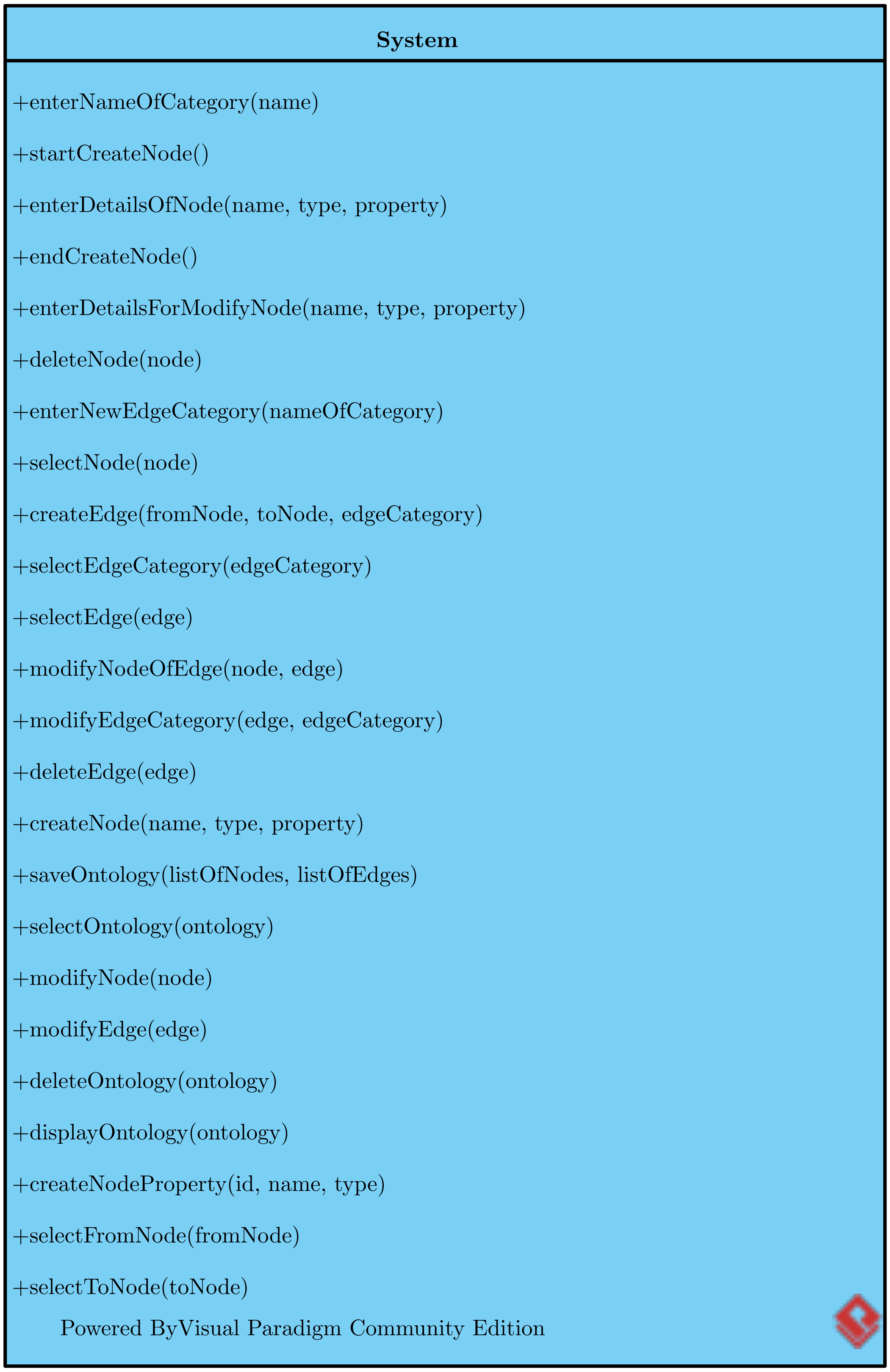
View Ontology



Create Other Property


Expert
: System
1: createProperty(id,name,type)
alt
1.1: Property created successfully.
1.2: Property creation failed
Powered ByVisual Paradigm Community Edition

6 System operations


System
+enterNameOfCategory(name)
+startCreateNode()
+enterDetailsOfNode(name, type, property)
+endCreateNode()
+enterDetailsForModifyNode(name, type, property)
+deleteNode(node)
+enterNewEdgeCategory(nameOfCategory)
+selectNode(node)
+createEdge(fromNode, toNode, edgeCategory)
+selectEdgeCategory(edgeCategory)
+selectEdge(edge)
+modifyNodeOfEdge(node, edge)
+modifyEdgeCategory(edge, edgeCategory)
+deleteEdge(edge)
+createNode(name, type, property)
+saveOntology(listOfNodes, listOfEdges)
+selectOntology(ontology)
+modifyNode(node)
+modifyEdge(edge)
+deleteOntology(ontology)
+displayOntology(ontology)
+createNodeProperty(id, name, type)
+selectFromNode(fromNode)
+selectToNode(toNode)
Powered ByVisual Paradigm Community Edition

7 Operation Contracts

1. **Contract C01: enterNameOfCategory**
   - Operation : enterNameOfCategory(name)
   - Cross References : Use Case UC1: Create Node Category
   - Preconditions:
     A category instance is created.
   - Postconditions:
     - category was associated with the current Category (association formed).
     - category.name became name (attribute modification).

2. **Contract C02: startCreateNode**
   - Operation : startCreateNode()
   - Cross References : Use Case UC2: Create Node
   - Preconditions:
     None.
   - Postconditions:
     - A Node instance node was created (instance creation).
     - Attributes of node were initialized.

3. **Contract C03: enterDetailsOfNode**
   - Operation : enterDetailsOfNode(name, type, property)
   - Cross References : Use Case UC2: Create Node
   - Preconditions:
     A node instance is created.
   - Postconditions:
     - node was associated with the current Node (association formed).
     - node.name became name (attribute modification).
     - node.type became type (attribute modification).
     - node.property became property (attribute modification).

4. **Contract C04: endCreateNode**
   - Operation : endCreateNode()
   - Cross References : Use Case UC2: Create Node
   - Preconditions:
     A node creation is underway.
   - Postconditions:
     - node creation successfully become true (condition).

5. **Contract C05: enterDetailsForModifyNode**
   - Operation : enterDetailsForModifyNode(name, type, property)
   - Cross References : Use Case UC3: Modify Node

- Preconditions:
  -Selected node was created before.
  -node instance is created and equal to selected node.
- Postconditions:
  - node was associated with the previous Node (association formed).
  - node.name became name (attribute modification).
  - node.type became type (attribute modification).
  - node.property became property (attribute modification).

6. **Contract C06: deleteNode(node)**
   - Operation : deleteNode(node)
   - Cross References : Use Case UC4: Delete Node
   - Preconditions:
     -Node deletion is under way
   - Postconditions:
     - given node instance was inserted as node instance (instance assertion).
     - node was selected for deletion.
     - node deletion is underway.

7. **Contract C07: enterNewEdgeCategory(nameOfCategory)**
   - Operation : enterNewEdgeCategory(nameOfCategory : string)
   - Cross References :
     Use Case UC5: Create Edge Category
   - Preconditions:

     Creation of edge category is underway.
   - Postconditions:
     - An edgeCategory instance ec was created.
     - ec was associated with preexisting list of edgeCategory.

8. **Contract C08: selectNode(node: Node)**
   - Operation : selectNode(node: Node)
   - Cross References :
     Use Case UC6: Create Edge between nodes
     Use Case UC3: Modify Node
     Use Case UC4: Delete Node
   - Preconditions: Operation with node is going on.
   - Postconditions: A node instance n was selected.

9. **Contract C09: selectEdge(edge: Edge)**
   - Operation : selectEdge(edge: Edge)
   - Cross References :
     Use Case UC7: Modify Edge
     Use Case UC8: Delete Edge

- Preconditions:

  Modification of edge is underway. OR
  Deletion of edge is underway.
- Postconditions:
  - An edge instance e was selected.

10. **Contract C10: createEdge(fromNode: Node, toNode: Node, edgeCategory: EdgeCategory)**

    - Operation : createEdge(fromNode: Node, toNode: Node, edgeCategory: EdgeCategory)
    - Cross References :
      Use Case UC6: Create Edge between nodes
      Use case UC9 : Create Ontology
      Use Case UC10: Modify Ontology
    - Preconditions:
      Edge creation is underway. OR
      Ontology creation is underway. OR
      Ontology modification is underway.
    - Postconditions:
      - An edge instance e was created.
      - e became associated with fromNode and toNode.
      - Type of e is set by instance of edgeCategory.
      - e became associated with current Ontology.

11. **Contract C11: selectEdgeCategory(edgeCategory)**

    - Operation : selectEdgeCategory(edgeCategory: EdgeCategory)
    - Cross References :
      Use Case UC6: Create Edge between nodes
      Use Case UC7: Modify Edge
    - Preconditions: Operation with edge is going on.
    - Postconditions: An edgeCategory instance e was selected from list of edge category list.

12. **Contract C12: modifyNodeOfEdge(node edge)**

    - Operation : modifyNodeOfEdge(node : Node, edge:Edge)
    - Cross References : Use Case UC7: Modify Edge
    - Preconditions:
      Modification of edge is underway.
    - Postconditions:
      - Association relation between instance of node n and associated edge e was modified.

13. **Contract C13: modifyEdgeCategory(edge,edgeCategory)**

    - Operation : modifyEdgeCategory(edge:Edge : edgeCategory : EdgeCategory)
    - Cross References : Use Case UC7: Modify Edge
    - Preconditions:
      Modification of edge is underway.

- Postconditions:
  - Association relation between instance of node n1 and node n2 was modified.
  - Associated ontology was modified.

14. **Contract C14: deleteEdge(edge)**
    - Operation : deleteEdge(edge: Edge)
    - Cross References :

      Use Case UC8: Delete Edge
    - Preconditions:
      Deletion of edge is underway.
    - Postconditions:
      - Association relation between edge instance e and associated node of e was discarded.
      - Association relation between e and associated ontology of e was discarded.
      - The edge instance e was removed from system.

15. **Contract C15: createNode(name,type,properties)**
    - Operation : createNode(name: String, type : NodeCategory, properties: ListOfProperties)
    - Cross References :
      Use Case UC2: Create Node
      Use Case UC9 : Create Ontology
      Use Case UC10: Modify Ontology
    - Preconditions:
      Node creation is underway. OR
      Ontology creation is underway. OR
      Ontology modification is underway.
    - Postconditions:
      - An node instance n was created.
      - Name of n is set by name.
      - Type of n is set by instance of nodeCategory.
      - Properties of n is set with properties.
      - n became associated with current Ontology.

16. **Contract C16: saveOntology(listOfNodes, listOfEdges)**
    - Operation : saveOntology(listOfNodes: ListOfNodes, listOfEdges : ListOfEdges)
    - Cross References :
      Use Case UC9 : Create Ontology
      Use Case UC10: Modify Ontology
    - Preconditions:

      Ontology creation is underway. OR
      Ontology modification is underway.
    - Postconditions:
      - An ontology instance o was created.
      - List of nodes of o is set by listOfNodes.

    - List of edges of o is set by listOfNodes.

17. **Contract C17: selectOntology(ontology)**
    - Operation : selectOntology(ontology: Ontology)
    - Cross References :
      Use Case UC10: Modify Ontology
      Use Case UC11: Delete Ontology
      Use Case UC12: View Ontology
    - Preconditions:

      Modification of existing ontology is underway. OR
      Deletion of existing ontology is underway.
    - Postconditions:
      - An ontology instance o was selected.

18. **Contract C18: modifyNode(node)**
    - Operation : modifyNode(node : Node)
    - Cross References : Use Case UC10: Modify Ontology
      Use Case UC3: Modify Node
    - Preconditions:
      Modification of ontology is underway.
    - Postconditions:
      - Association relation between instance of node n and associated edge was modified.
      - Name of n could be modified.
      - Node category of n could be modified.
      - Node properties of n could be modified.
      - Association relation between instance of node n and associated ontology was modified.

19. **Contract C19: modifyEdge(edge)**
    - Operation : modifyEdge(edge : Edge)
    - Cross References : Use Case UC10: Modify Ontology
      Use Case UC7: Modify Edge
    - Preconditions:
      Modification of ontology is underway.
    - Postconditions:
      - Association relation between instance of edge e and associated node was modified.
      - Association relation between instance of edge e and associated ontology was modified.

20. **Contract C20: deleteOntology(ontology)**
    - Operation : deleteOntology(ontology: Ontology)
    - Cross References :

      Use Case UC11: Delete Ontology
    - Preconditions:
      Deletion of ontology is underway.

- Postconditions:
  - Association relation between ontology instance o and associated nodes of o was discarded.
  - Association relation between o and associated edges of e was discarded.
  - The ontology instance o was removed from system.

21. **Contract C21: displayOntology(ontology)**
    - Operation : displayOntology(ontology : Ontology)
    - Cross References : Use Case UC12: View Ontology
    - Preconditions:
      View operation of ontology is underway.
    - Postconditions:
      - An instance of ontology o is fetched from the system.

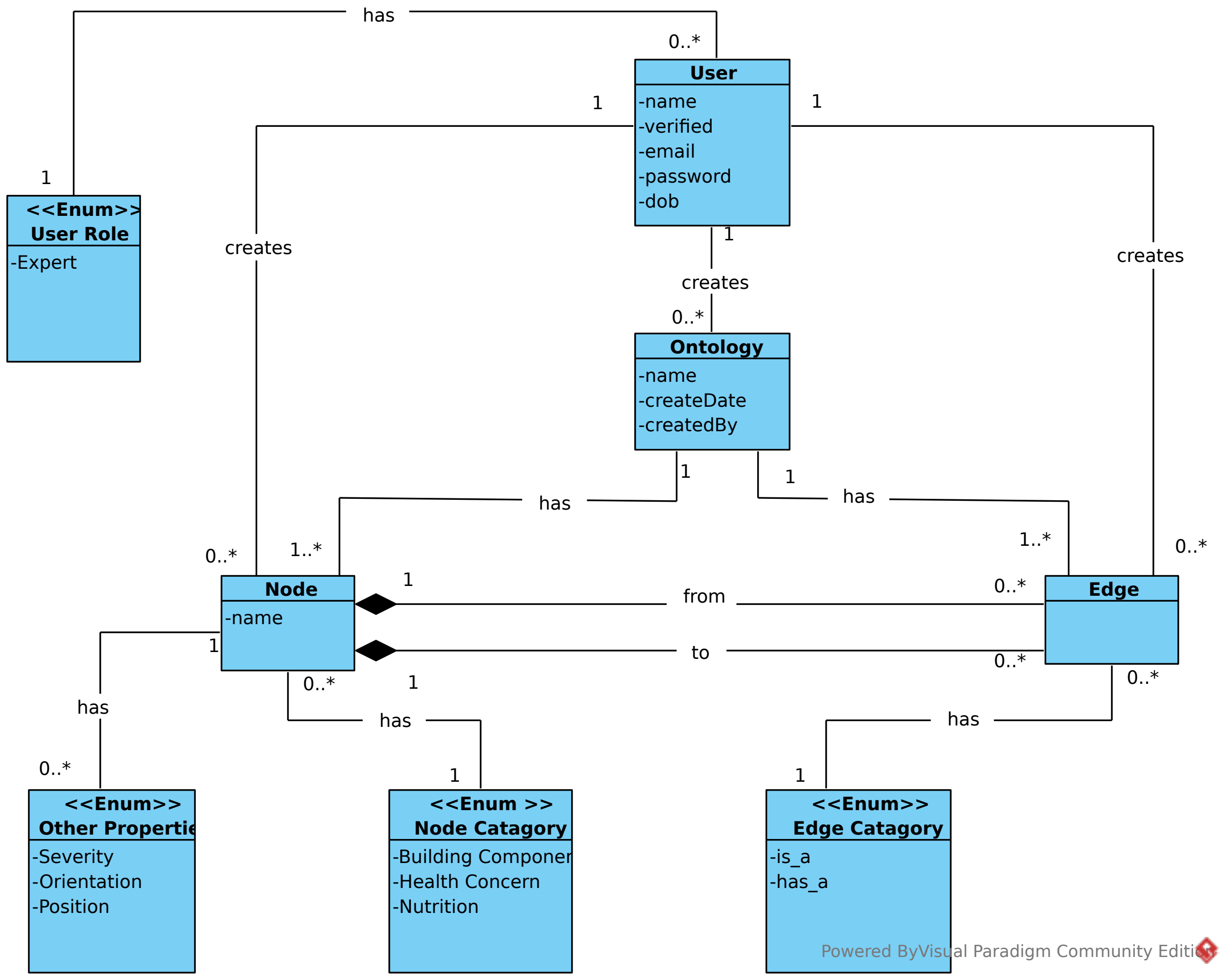
22. **Contract C22: createNodeProperty(id,name,type)**
    - Operation : createNodeProperty(id: string,name : string ,type : : string)
    - Cross References :
      Use Case UC13: Create Other Property
    - Preconditions:

      Creation of other property of node is underway.
    - Postconditions:
      - An otherProperty instance op was created.
      - Name of op is set by name.
      - Id of op is set by id.
      - Type of op is set by type.
      - ec was associated with preexisting list of otherProperty.

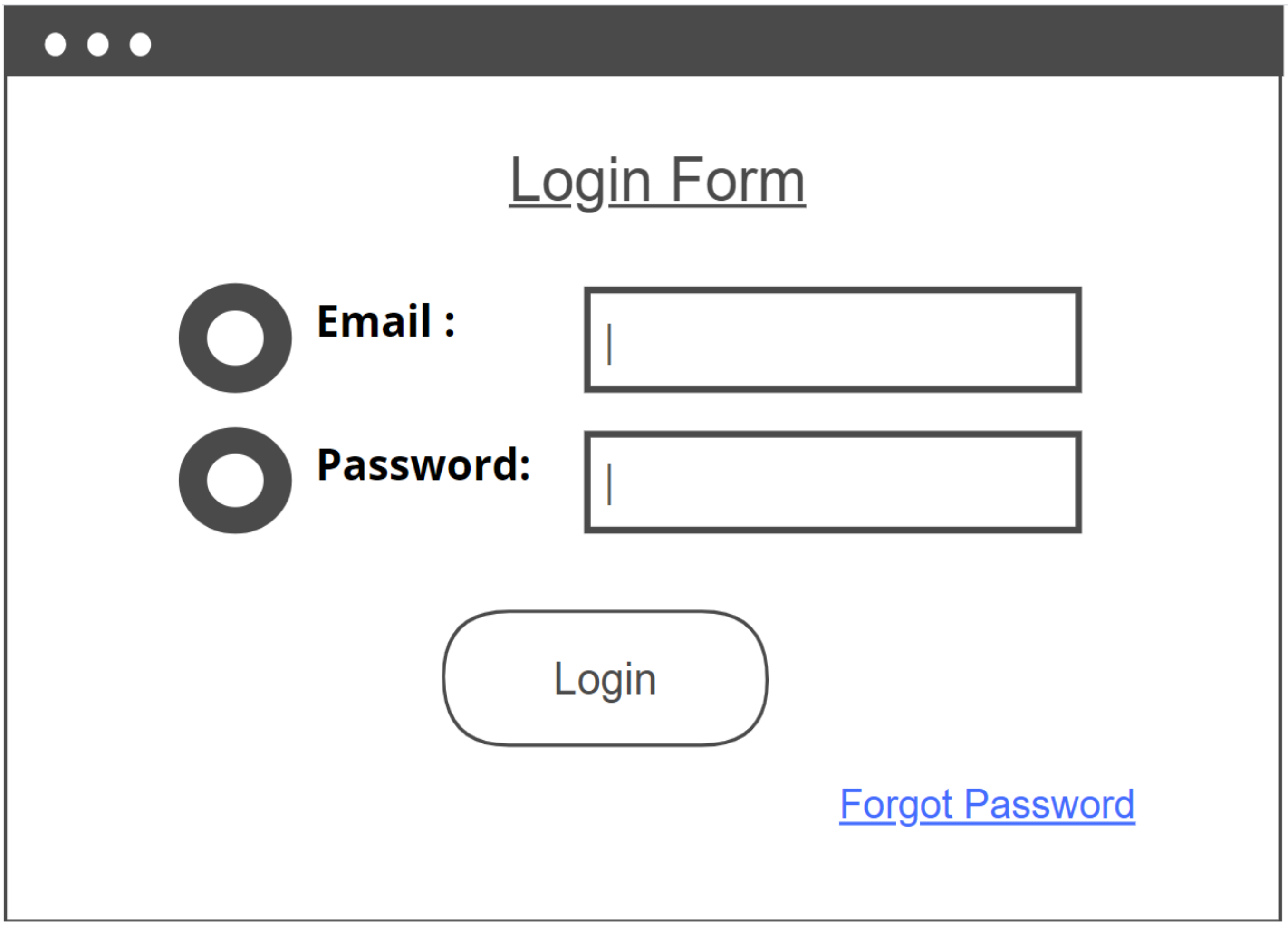
23. **Contract C23: selectFromNode(fromNode: Node)**
    - Operation : selectFromNode(fromNode: Node)
    - Cross References :
      Use Case UC6: Create Edge between nodes
    - Preconditions: Edge creation is underway.
    - Postconditions: A node instance n was selected.

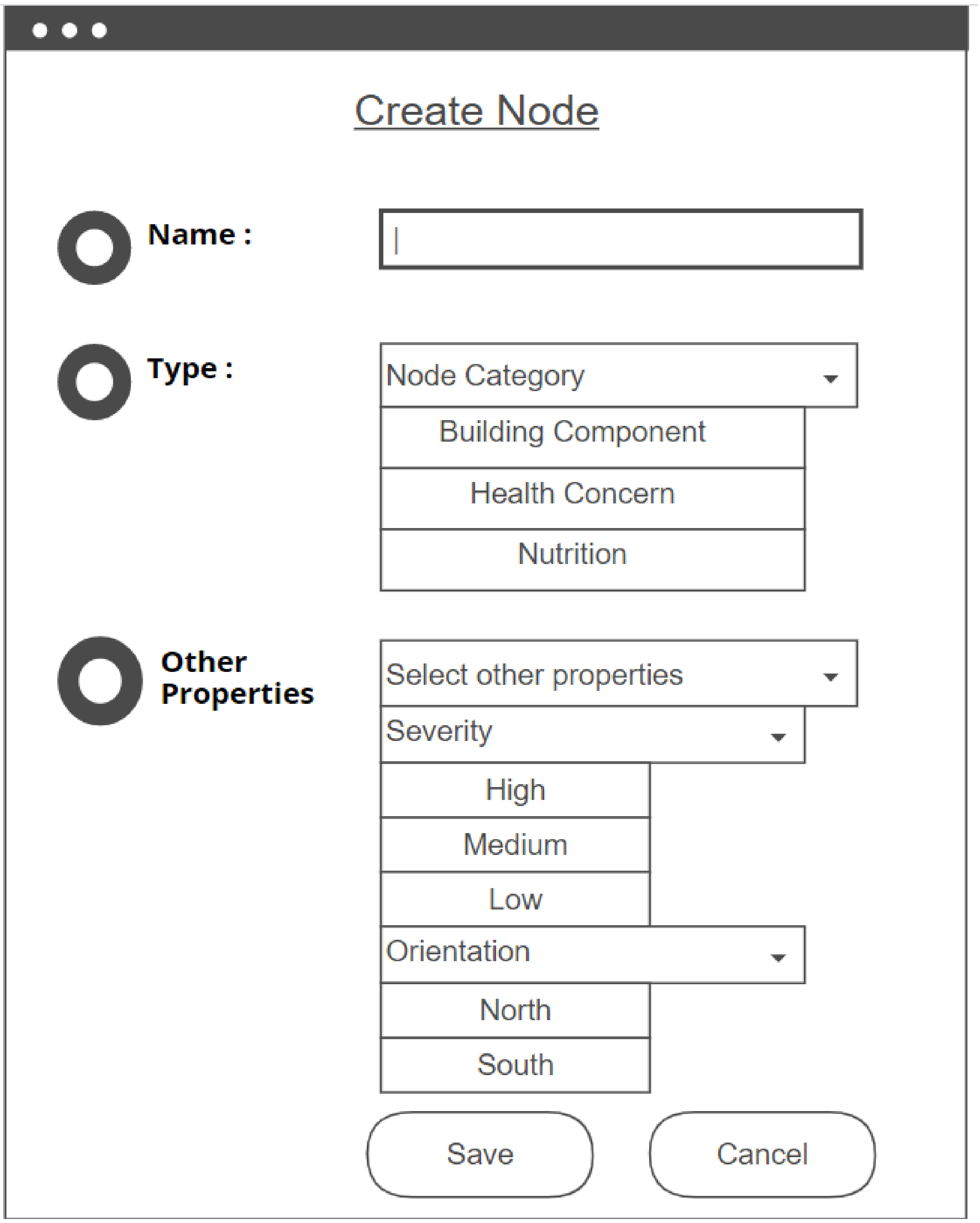
24. **Contract C24: selectToNode(toNode: Node)**
    - Operation : selectToNode(toNode: Node)
    - Cross References :
      Use Case UC6: Create Edge between nodes
    - Preconditions: Edge creation is underway.
    - Postconditions: A node instance n was selected.

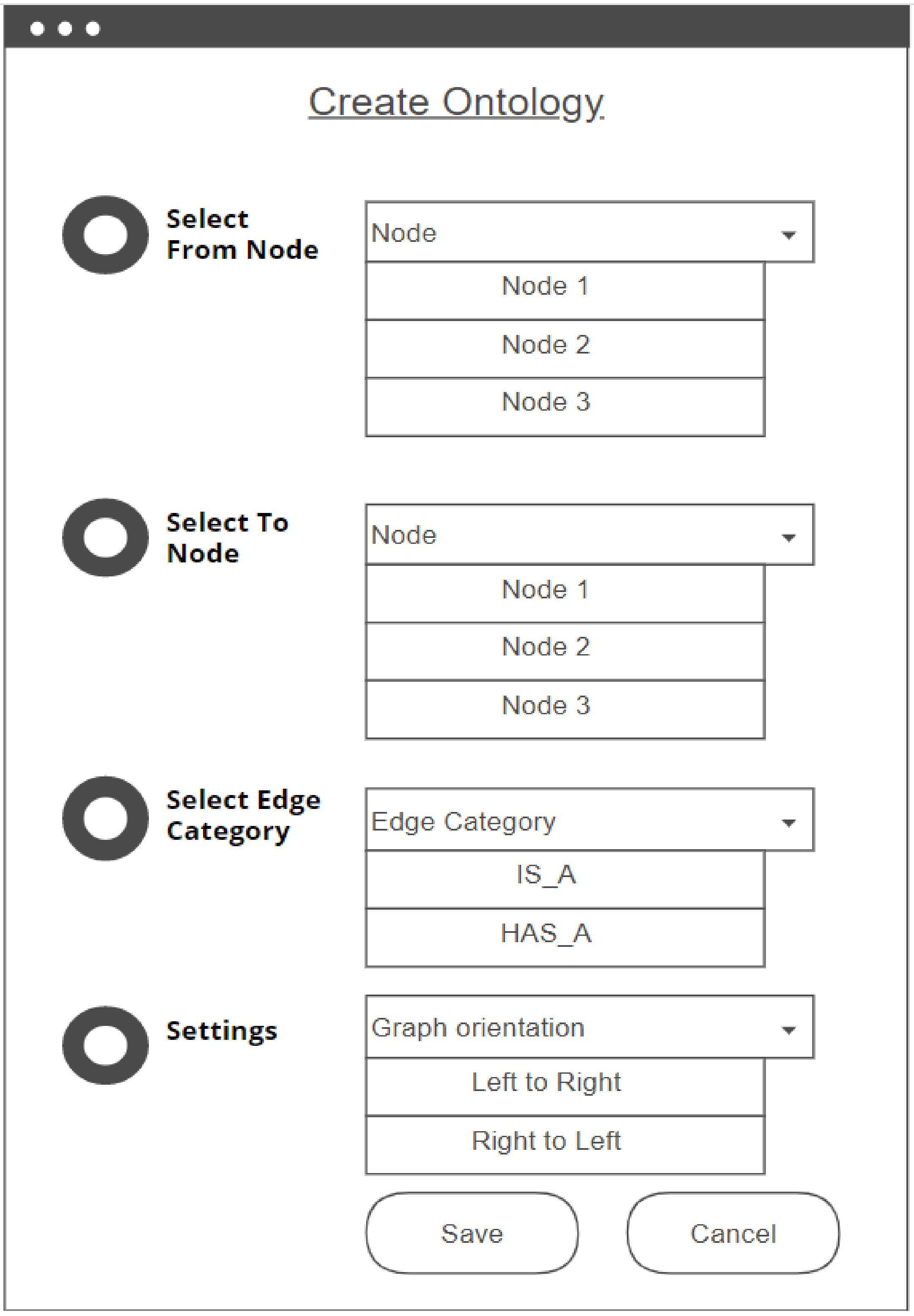
8 Domain Model

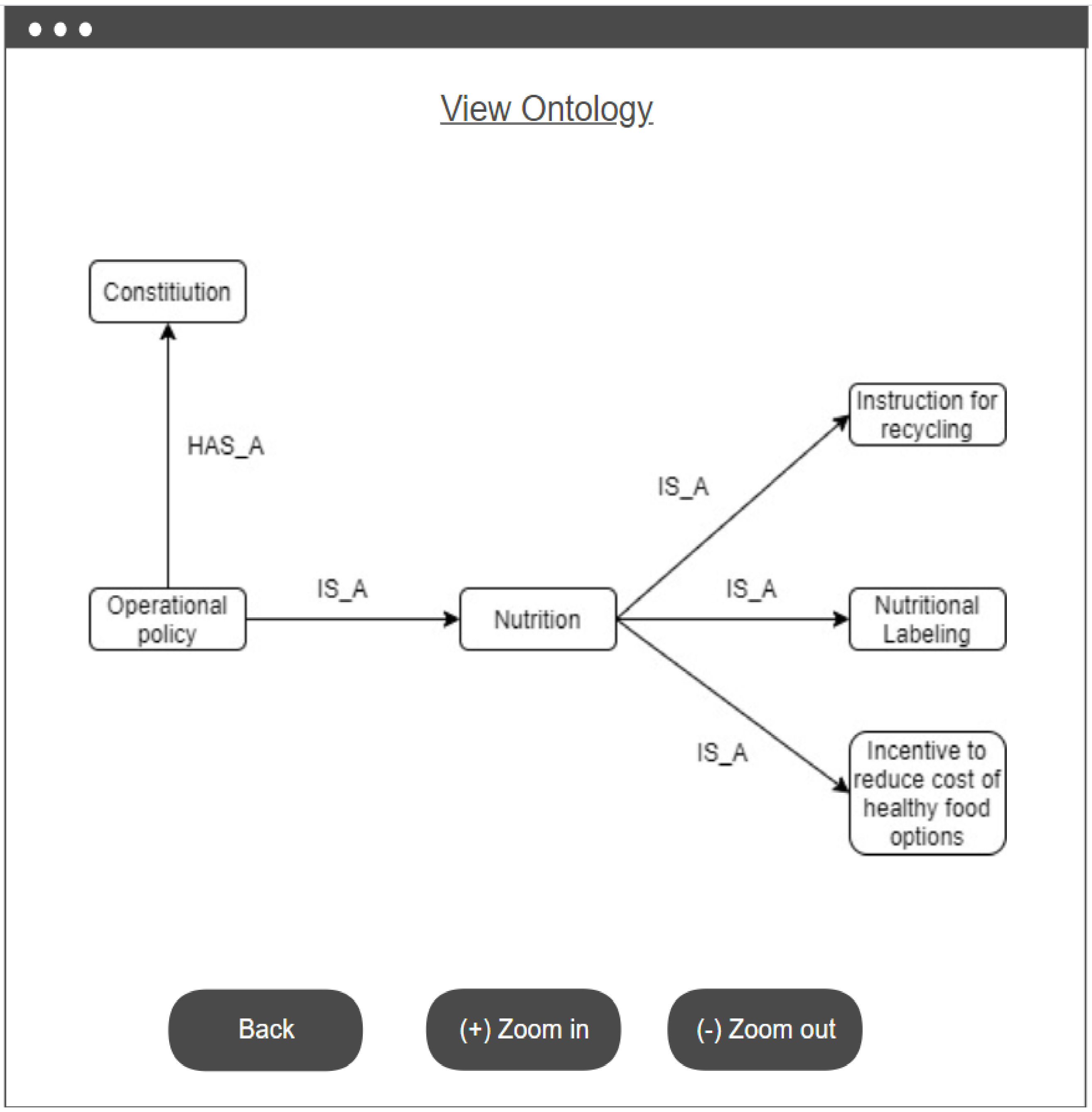


9 WireFrames

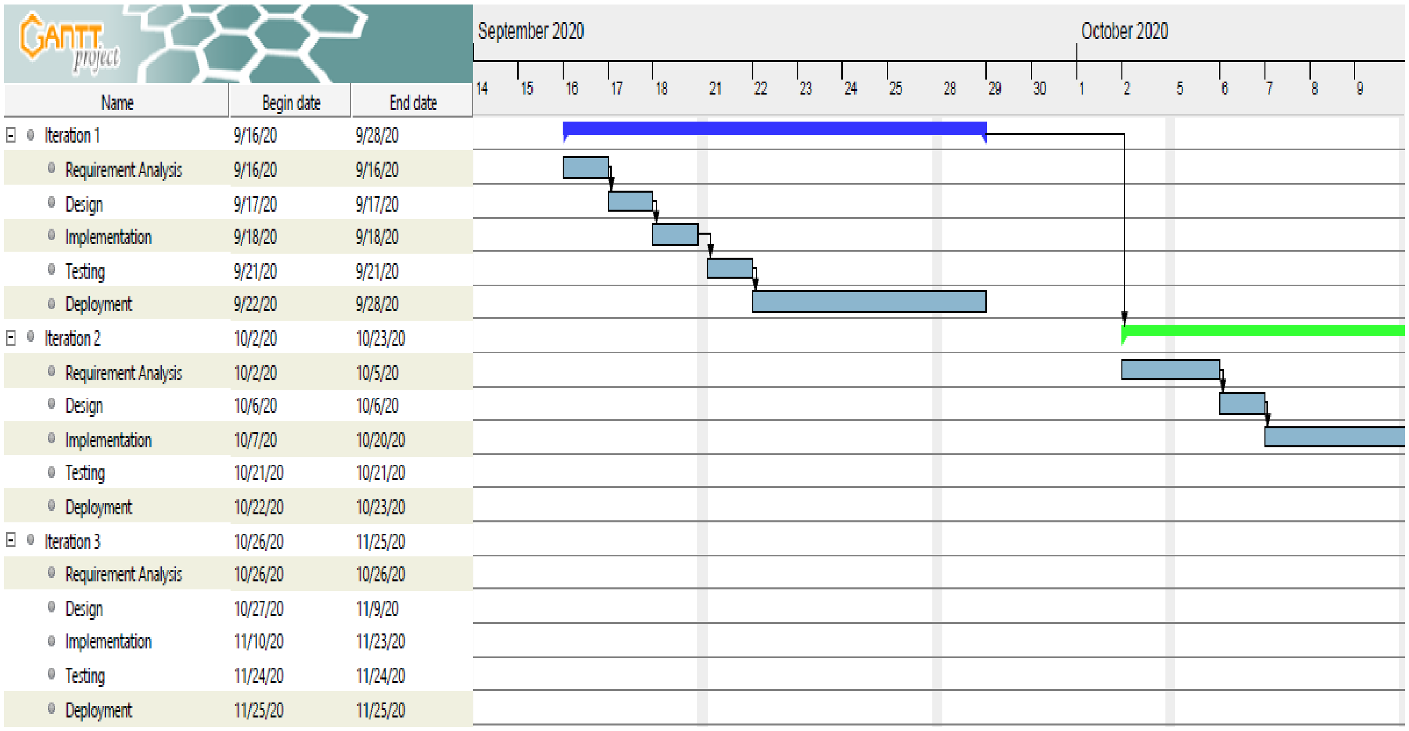



Create Node
Name :
Type :
Node Category
Building Component
Health Concern
Nutrition
Other Properties
Select other properties
Severity
High
Medium
Low
Orientation
North
South
Save
Cancel


Create Ontology
Select From Node
Node
Node 1
Node 2
Node 3
Select To Node
Node
Node 1
Node 2
Node 3
Select Edge Category
Edge Category
IS_A
HAS_A
Settings
Graph orientation
Left to Right
Right to Left
Save
Cancel


View Ontology
Constitiution
HAS_A
Instruction for recycling
IS_A
IS_A
IS_A
Operational policy
Nutrition
Nutritional Labeling
IS_A
Incentive to reduce cost of healthy food options
Back
(+) Zoom in
(-) Zoom out

10 Gantt Chart



Name	Begin date	End date
Iteration 1	9/16/20	9/28/20
Requirement Analysis	9/16/20	9/16/20
Design	9/17/20	9/17/20
Implementation	9/18/20	9/18/20
Testing	9/21/20	9/21/20
Deployment	9/22/20	9/28/20
Iteration 2	10/2/20	10/23/20
Requirement Analysis	10/2/20	10/5/20
Design	10/6/20	10/6/20
Implementation	10/7/20	10/20/20
Testing	10/21/20	10/21/20
Deployment	10/22/20	10/23/20
Iteration 3	10/26/20	11/25/20
Requirement Analysis	10/26/20	10/26/20
Design	10/27/20	11/9/20
Implementation	11/10/20	11/23/20
Testing	11/24/20	11/24/20
Deployment	11/25/20	11/25/20