Product Design

Team 15 : **Piyush Bansal, Chandan Singh, Gv Karthik**

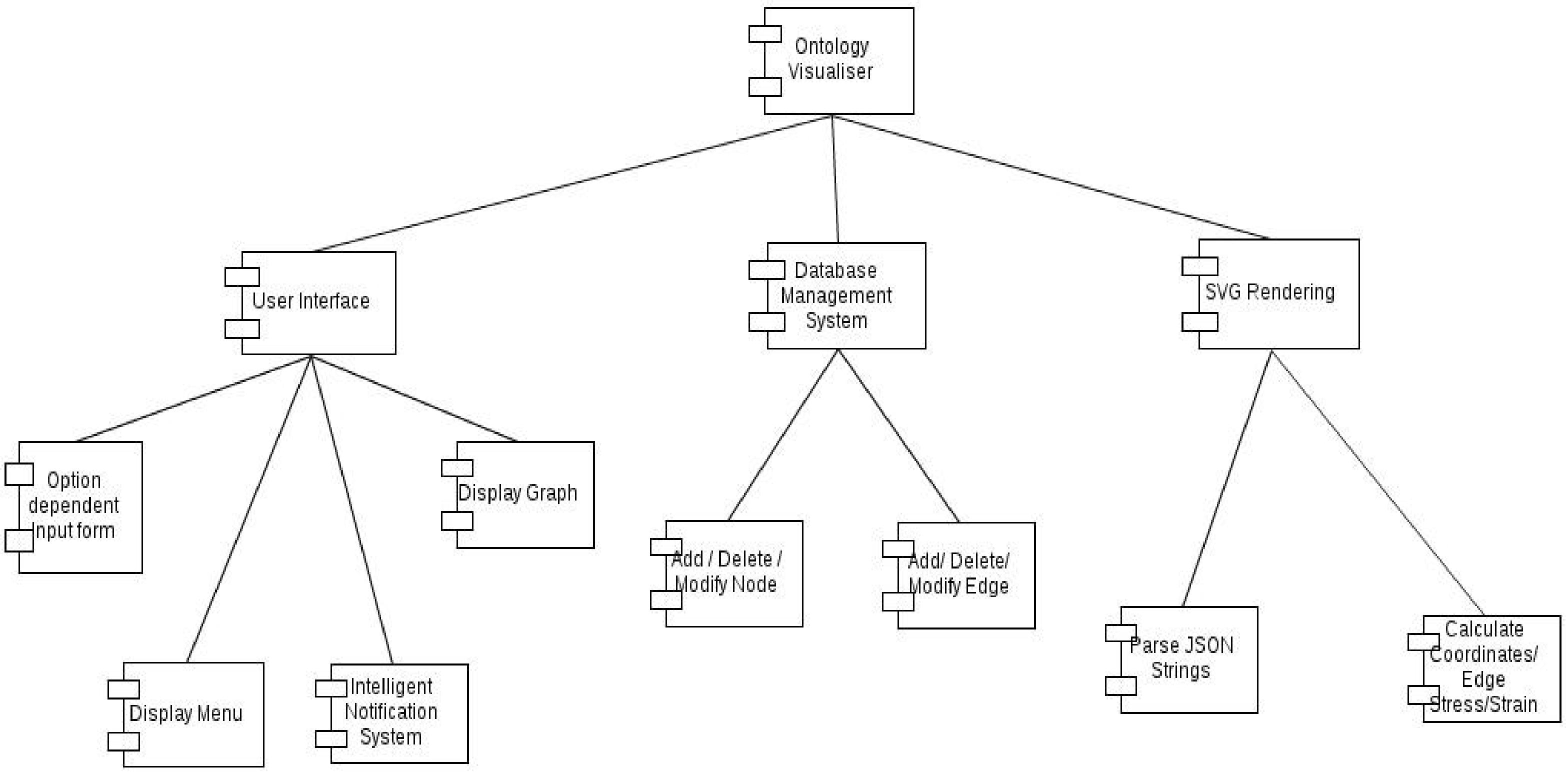
Architectural Model

COMPONENTS TABLE:

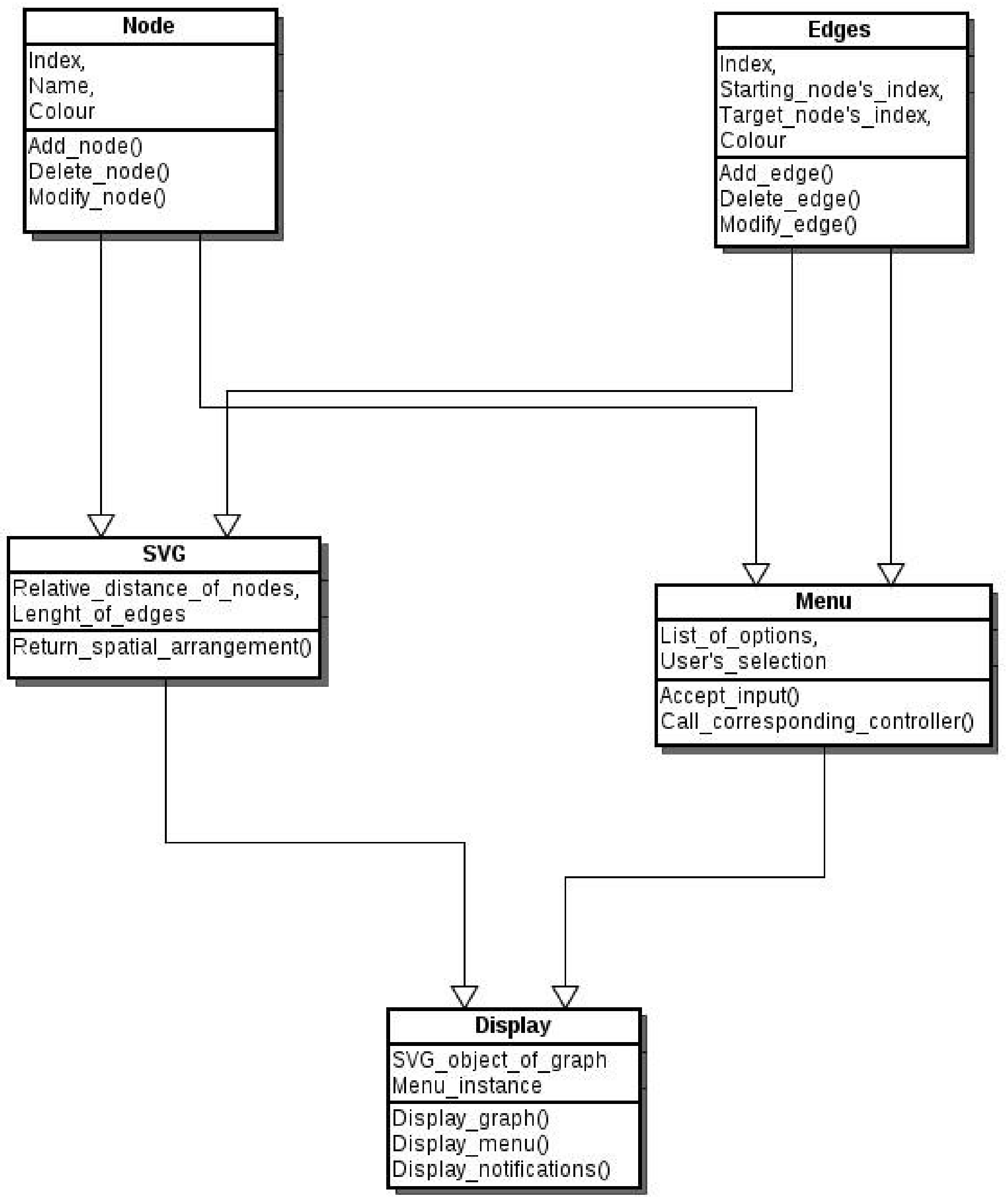
1. User Interface	**Component state** • This Module is responsible for the view part of the application. It provides the user with an input form along with options menu . **Component Behavior** • This component helps Database Management System component with Input values on which the database is supposed to be updated.
1.1 Option Dependent Form	**Component state** • This form is responsible for obtaining input according to the option selected by the user **Component Behavior** • This subcomponent helps passing input values to database management component .
1.2 Display Graph	**Component state** • This Subcomponent helps render values returned by the SVG component into a force based graph. **Component Behavior** • This Subcomponent , does not provide service to any other component , its an independent module.
1.3 Display Menu	**Component state** • This Subcomponent helps user chose various functions such as add/delete/modify a node , add/delete/modify edge. Component Behavior • This Subcomponent is independent of other components.
1.4 Intellige Notification system.	**Component state** • This Subcomponent is responsible for giving out intelligent alerts and notifications to the user regarding his activity. **Component Behavior** • This Subcomponent is also independent .

2. Database management system	**Component state** • This component is the basic database interacting module. **Component behavior** • This component helps get input values to the SVG Rendering Module.
2.1 Add/ Delete/ Modify node	**Component state** • This Subcomponent helps to add a new node to the graph by passing value to the database. **Component Behavior** • This Subcomponent helps SVG rendering Module obtain values to function on.
2.2 Add/ Delete/ Modify Edge	**Component state** • This Subcomponent helps add a new edge to the graph by passing the values inputted by the user , i.e. passed on from the UI module to the Database. **Component Behavior** • This Subcomponent helps SVG rendering module get the values so as to render coordinates etc.
3. SVG Rendering Module	**Component state** • This is the main subcomponent where various force based algorithms are applied to the graph inputs and a spring-charge modeled graph coordinates and edge behavior is returned. **Component behavior** • This module helps UI component to a very great deal . UI renders all the values that it gets from this module. This , however should not the thought of a coupling .
3.1 Parsing JSON strings	**Component state** • This subcomponent helps stringify the values stored in the database(JSON) and further render it to the 3.2 Subcomponent so as to apply various physics based algorithms. **Component Behavior** • This subcomponent helps the subcomponent 3.2 for obtaining values , in which is easier for it to function (i.e. strings)
3.2 Calculate Coordinates, Edges Stress / Strain	**Component state** • This subcomponent is Important to model the data in the required (force based Directed graph format). It stores in it the position of the nodes/edges. **Component Behavior** • This subcomponent helps render results to the UI module , which in turn renders the graph in the final format.

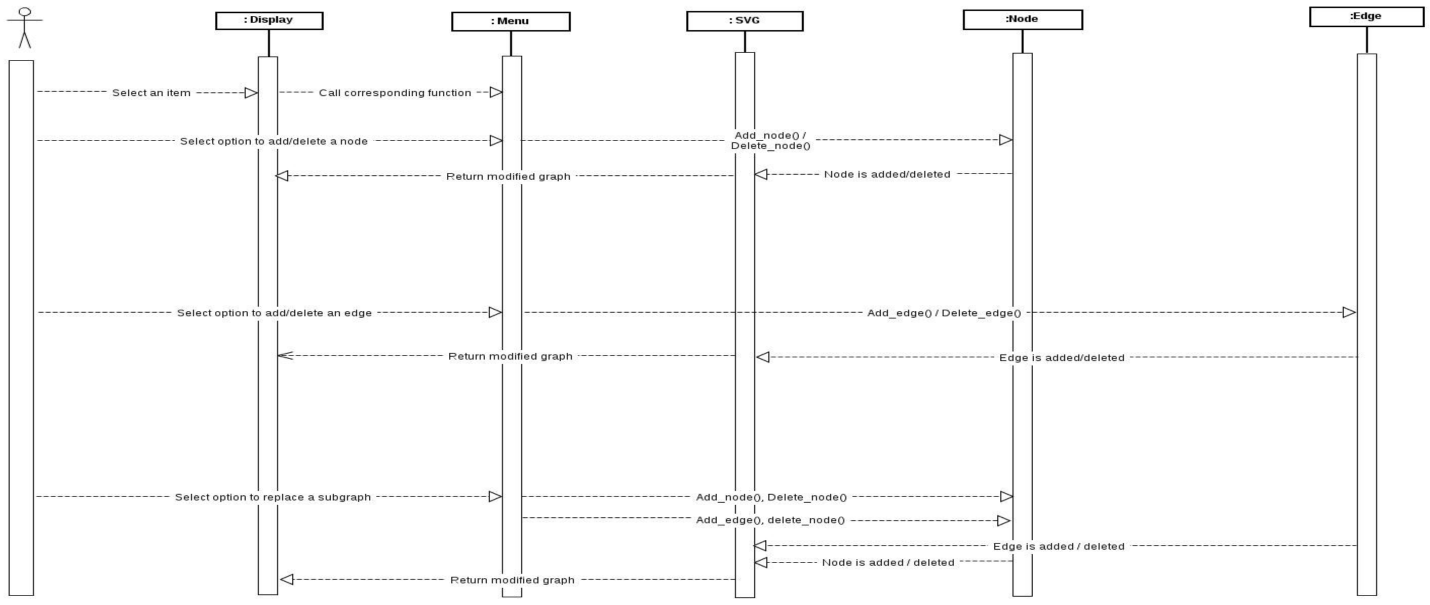
COMPONENT DIAGRAM:


Ontology Visualiser
User Interface
Database Management System
SVG Rendering
Option dependent Input form
Display Graph
Display Menu
Intelligent Notification System
Add / Delete / Modify Node
Add/ Delete/ Modify Edge
Parse JSON Strings
Calculate Coordinates/ Edge Stress/Strain

Class Diagram



Sequence Diagram


: Display
: Menu
: SVG
:Node
:Edge
Select an item
Call corresponding function
Select option to add/delete a node
Add_node() / Delete_node()
Return modified graph
Node is added/deleted
Select option to add/delete an edge
Add_edge() / Delete_edge()
Return modified graph
Edge is added/deleted
Select option to replace a subgraph
Add_node(), Delete_node()
Add_edge(), delete_node()
Edge is added / deleted
Node is added / deleted
Return modified graph

Link for full-sized image

Design Rationale

- Currently SVG class is explicitly distinguished from display class, which however is only partially true. A better Class implementation of an agglomeration of the two needs to be devised.
- These seems to be some amount of coupling between the Intelligent Notification system , inherited from the User Interface Component (Diagram) with SVG rendering component and Database Management System Component which needs to be managed properly.
- Class Node has been defined separate from class Edge. However they could be merged into one, if time permits and coupling issues arise. This would significantly reduce interdependencies between other modules as well.
- The current model has used JSON as database,which however can be changed if size of database increases, and JSON proves inefficient . The chances of this happening are quite less as Clients dont have a very large database as told.
- Current method for taking input is a basic input form . This doesn't appeal us very much as there can be better interactive methods for the same purpose. This however is purely upto our discretion to come up with alternate UI model , as client dint have any preferences.