**Design Documentation for Venn Diagram Application**

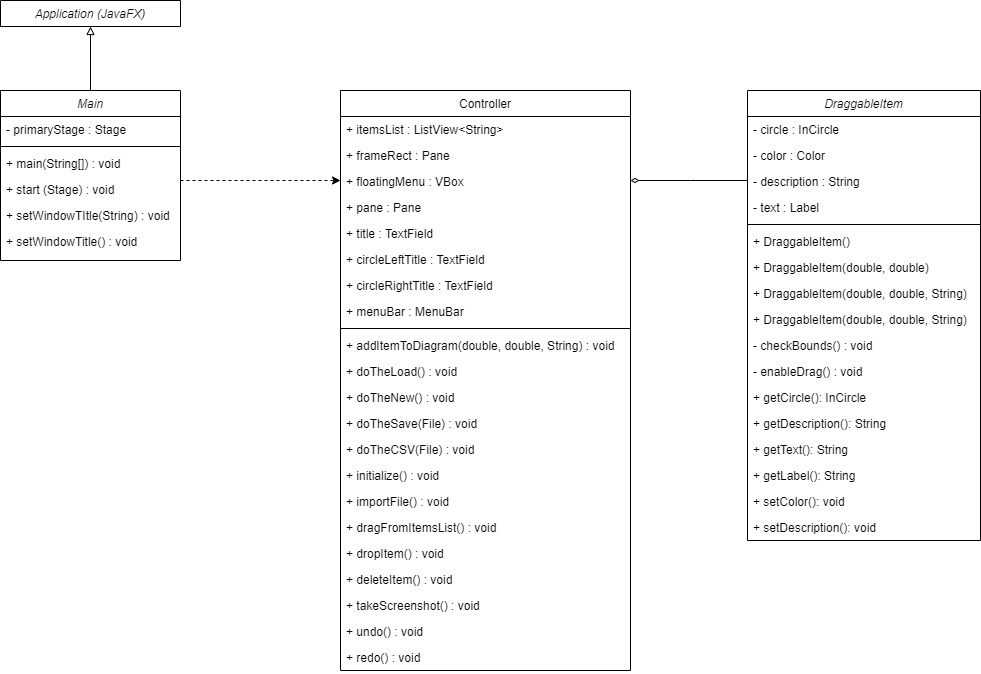
**Group 4**

**Class Diagram of the Venn Diagram**

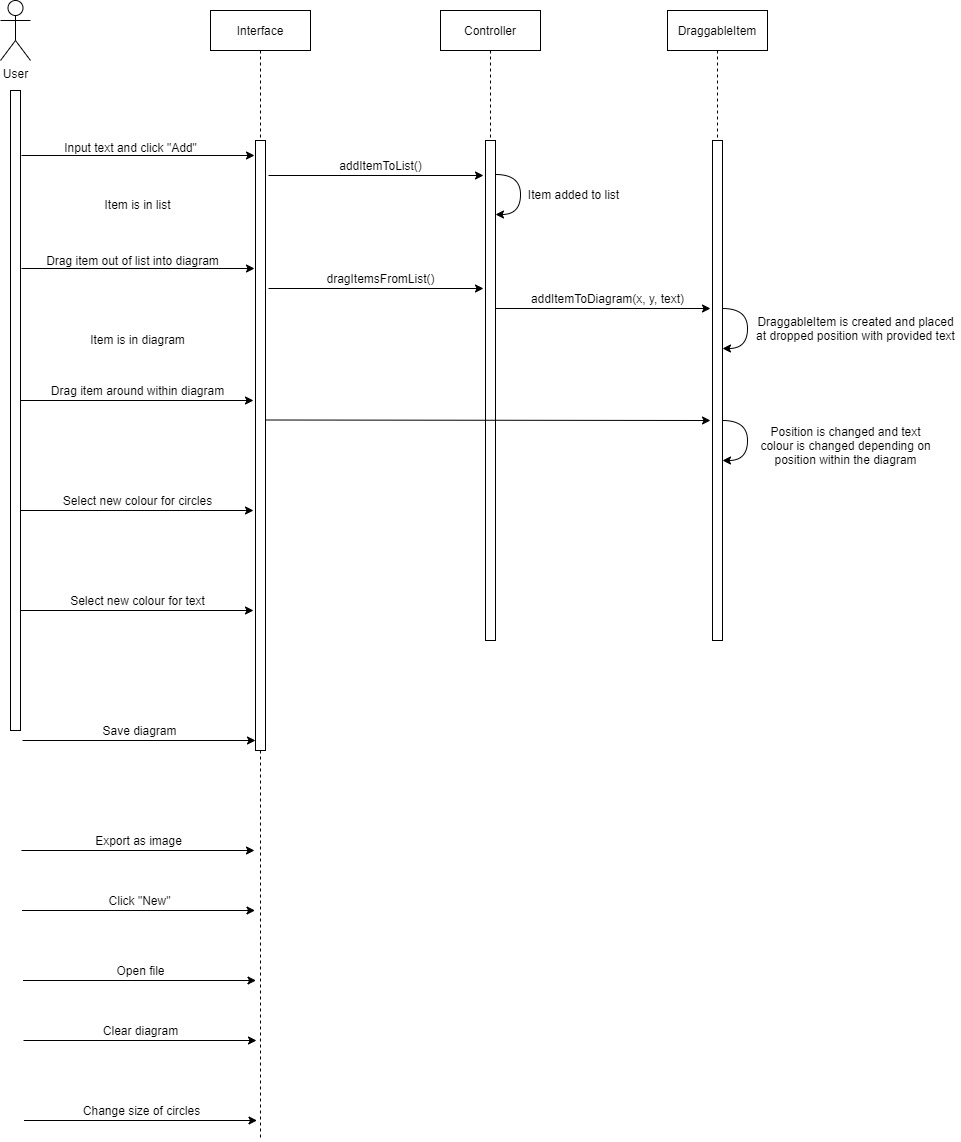
This UML Diagram shows the main classes of the appplication and their methods. Main class contains the UX application window.

The Controller class contains all the methods (this diagram contains the key methods). The Main class has directional connection with the Controller class.

The controller class calls the DraggableItem class. It depicts a ‘has-a’ relationship (aggregation).



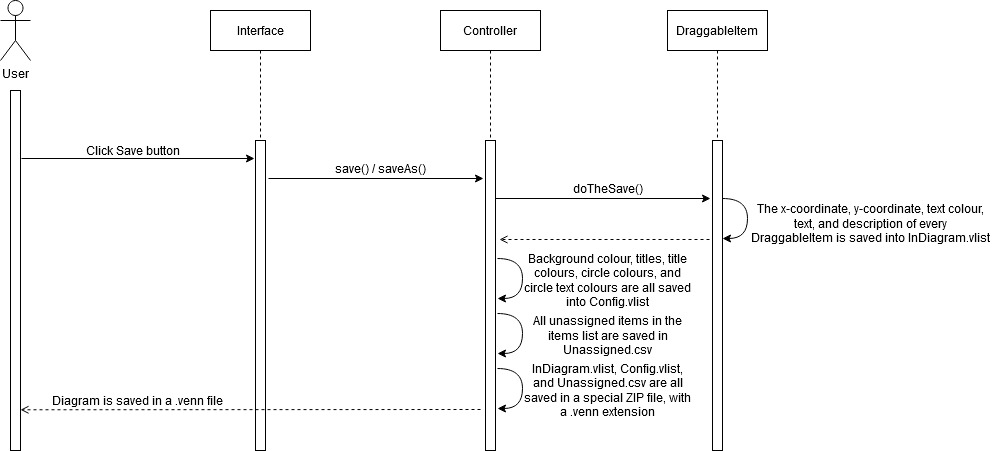
**Sequence Diagram for Activity 1:**

The following Sequence diagram depicts the sequence of inputting text in the list, and then drag the text in the VennDiagram.

draggableItem()

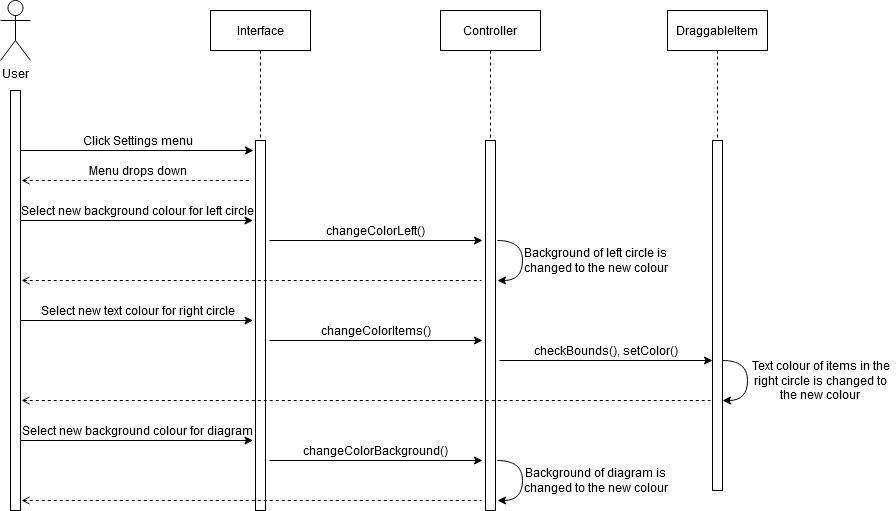
**Sequence Diagram Activity 2:**

The following Diagram depicts the sequence of Save As and/or Save function in the Venn Diagram. It shows the stages of method calls in various classes after the user selects to perform these functions from the interface.



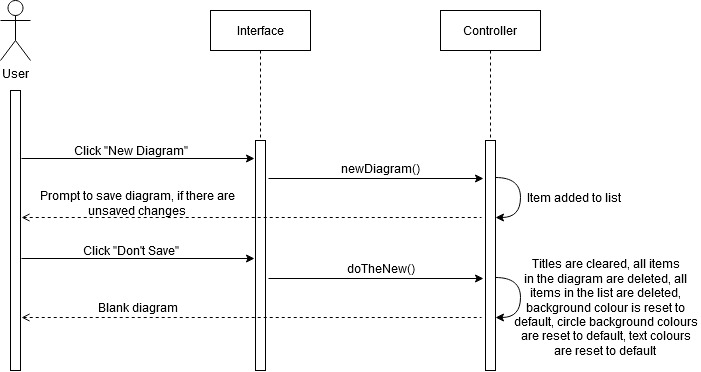
**Sequence Diagram Activity 3:**

The following sequence diagram shows the stages of customizing colour of the left and right circles.



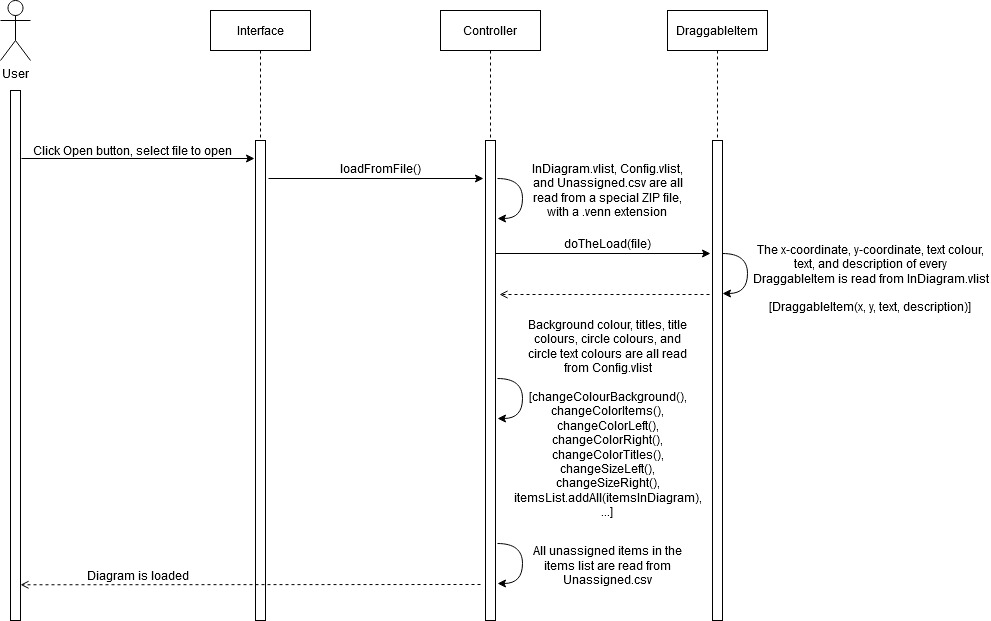
**Sequence Diagram Activity 4:**

The stages of creating a new Venn Diagram.



**Sequence Diagram Activity 5:**

The stages of importing loading a file into the Venn Diagram application.



**Software Maintenance Scenarios:**

This section includes the Corrective Maintenance issues that addresses the existing bugs that we aim to fix in the next release.

It also includes Perfective Maintenance scenarios that we can include to perfect the existing application.

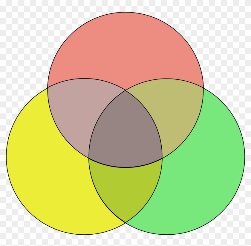
**Corrective Maintenance Scenarios:**

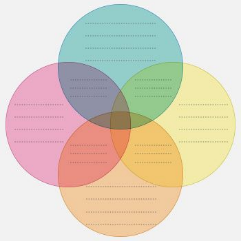
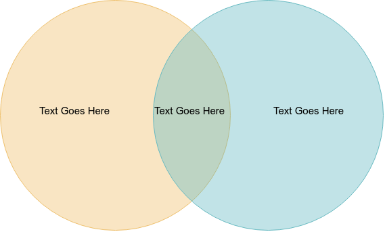
* If the application is opened in a full screen mode, the bottom of the Venn diagram is cropped out of the full screen mode.   
  To fix this issue, we can add scroll bars in the application window. Implement zooming function in the previous release.
* The textboxes may overlap if one textbox is dragged on top of the other. Add a condition before the setDropCompleted method to see if the borders of the textboxes overlap. If they do, use the setVisible(false) method to make the textbox added later invisible.

**Perfective Maintenance Scenario:**

Feature 1: Multiple Venn templates

* An updated release of the app would have several layouts of a Venn. For example, a three circle and a four circle Venn Diagram.
* To implement this feature, we need to change the class hierarchy of the java application, we need to have a 2 more classes, each of which contains the appropriate UI components. These classes will be called by the Main class. Each of these classes will share the same controller class to navigate the components and deploy the features.
* The same feature can also be implemented using a add circle feature. The add circle allows us to add 2/3/4 circles depending on the demand of the user. The circles can be dragged to create an overlapping zone. This will not require a complete reformation of the class hierarchy but we only need to add an addCircle() method in the Controller class. In addition, we need to tailor this change into the existing the methods and UI components.





Feature 2: Importing Image into Venn Diagram:

* Another feature that would enhance the application is to be able to import image and drag and drop them within the Venn Diagram.
* To execute the feature, we need to create a DraggableImage class that extends DraggableItem and override methods in Draggable Item to handle image format.
* Finally make an addImageToDiagram method in the Controller class

This is what the feature might look like:

