Design Document
EECS2311

Group 10

Thanh Nguyen 213 507 231

Mahamed Hassan - 216536476

Esra Kastrati - 215507205

Derui Liu 215608946

Purpose:

The is to provide a description of the **design** of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to built. So, this document will allow an experienced developer that is unfamiliar with the system to get a clear idea of the system's design (ideally also the rationale for the design).

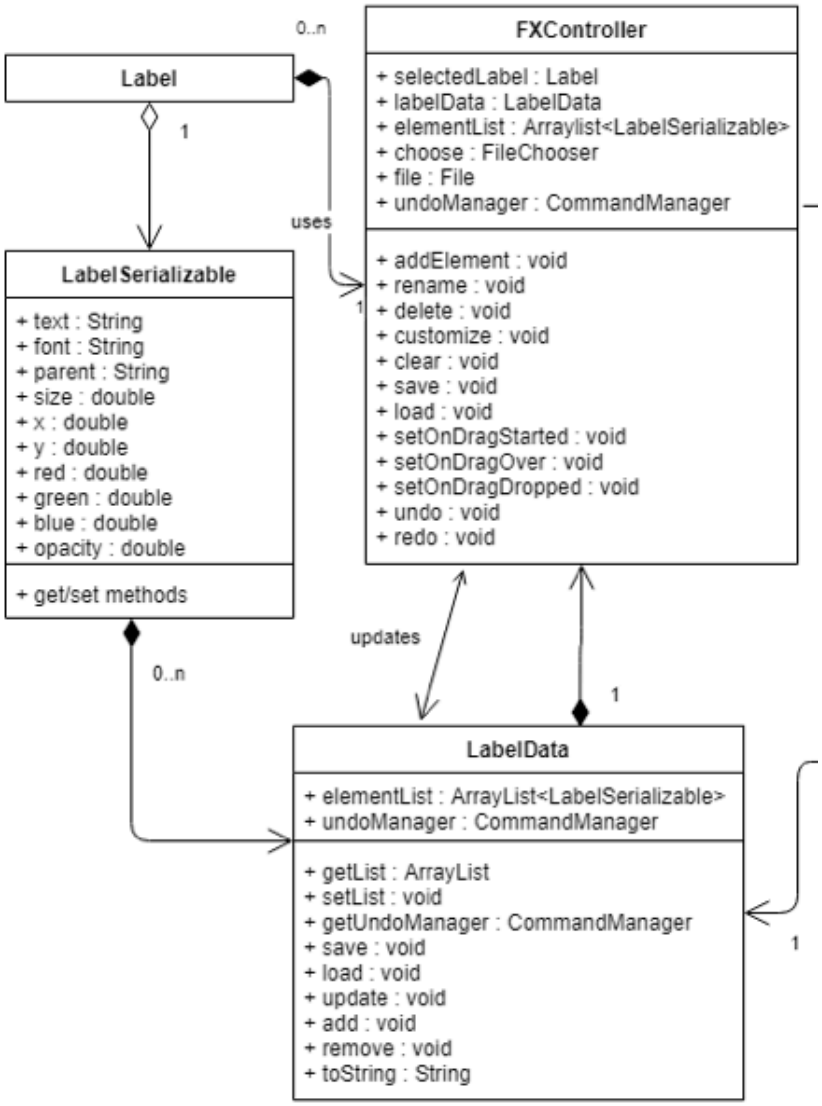
Overview

This app allows the user to represent ideas or datas in a more visual form. It has several features, like add/remove, save/saveAs, and undo/redo and features that allow the user to customize each part of the Venn diagram.

Class Diagrams

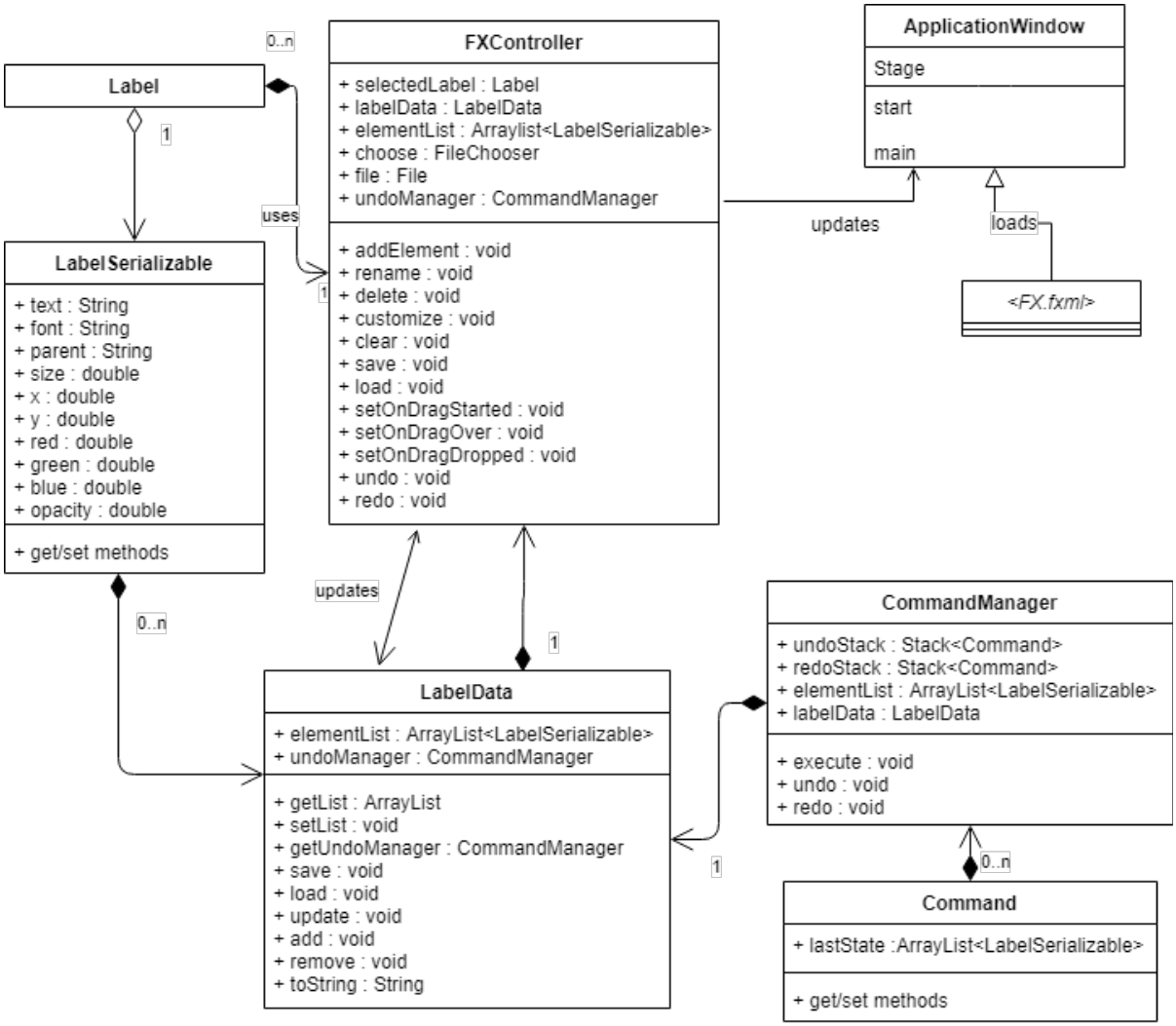
Customization with Venn diagram

It is very important for the user to be comfortable with the font, size and the colour of the elements. In our app, user can change how each element looks. Each element is created as a label in the FXController and the class LabelSerializable makes it possible for the user to customize list items. In order to display the customization options, user should right click and select ‘Customize’.



Complete Class Diagram:

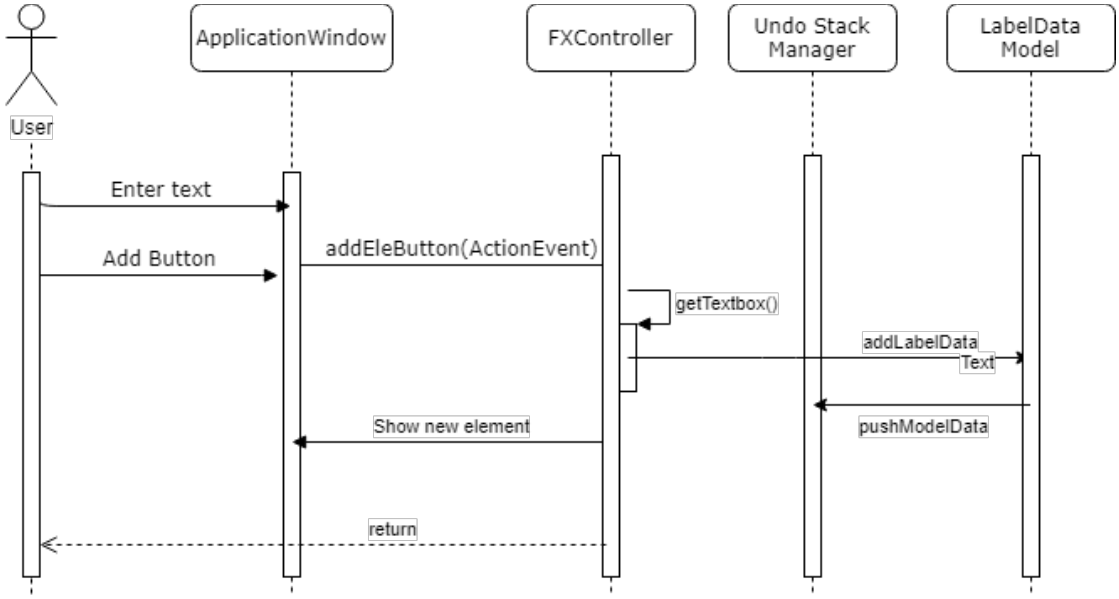
This diagram is a representation of what classes are used and how they interact with each other. The ApplicationWindow is initialized by the FXML file, then is updated with the FXController. The controller creates an instance of LabelData, which stores labels as LabelSerializable. This allows us to save and load the state of the ApplicationWindow. Every change done to LabelData is saved as a Command, which is handled by CommandManager. CommandManager will push a new Command onto its undo stack, and every undo operation both pop that Command and push it onto the redoStack.



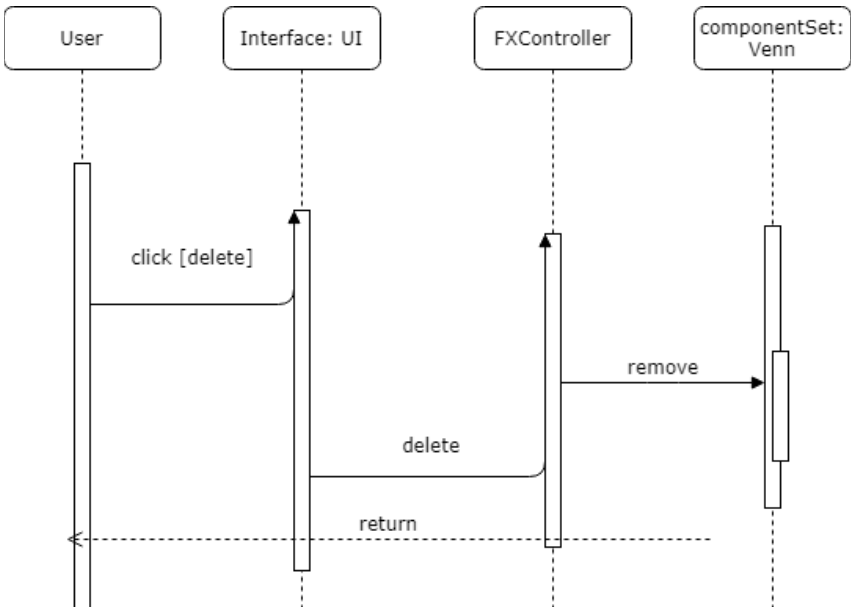
Sequence Diagrams

These diagrams are a representation of how the end user can interact with the program, and how the program responds accordingly. Each diagram represents an action performed by the user, and the methods involved with that action.

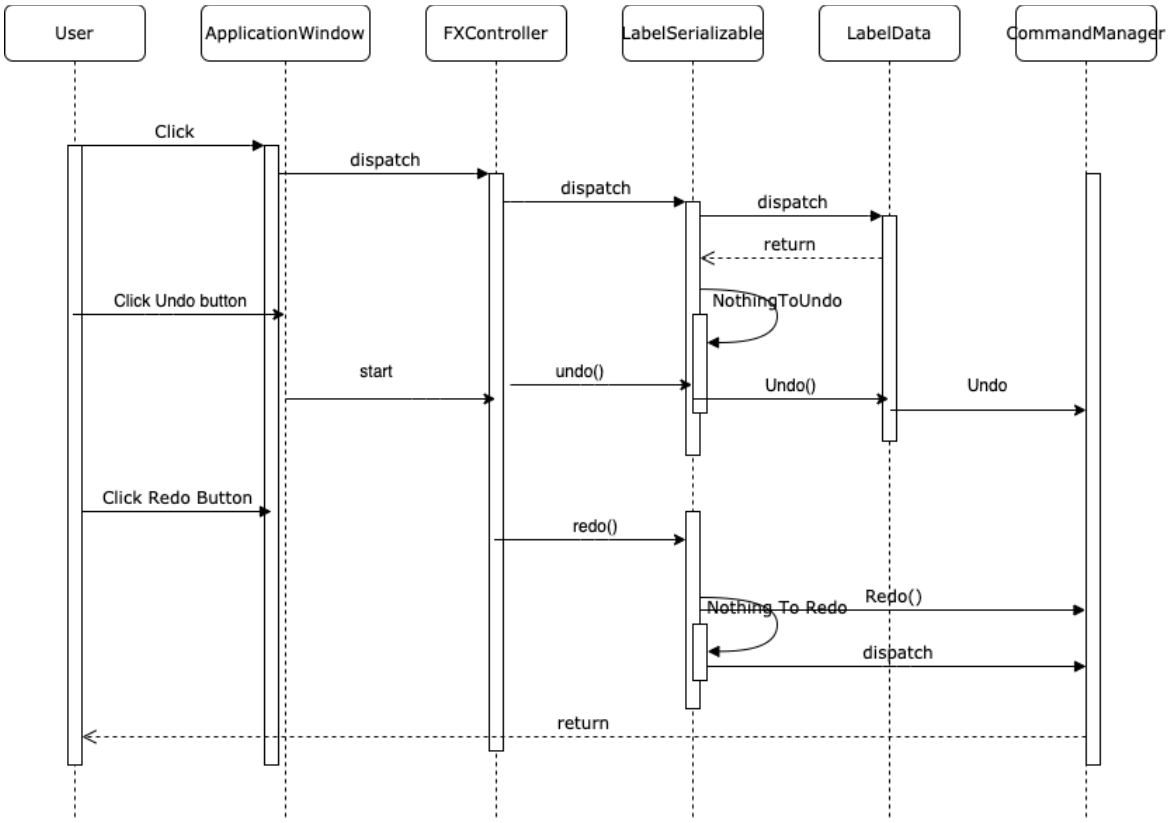
Add



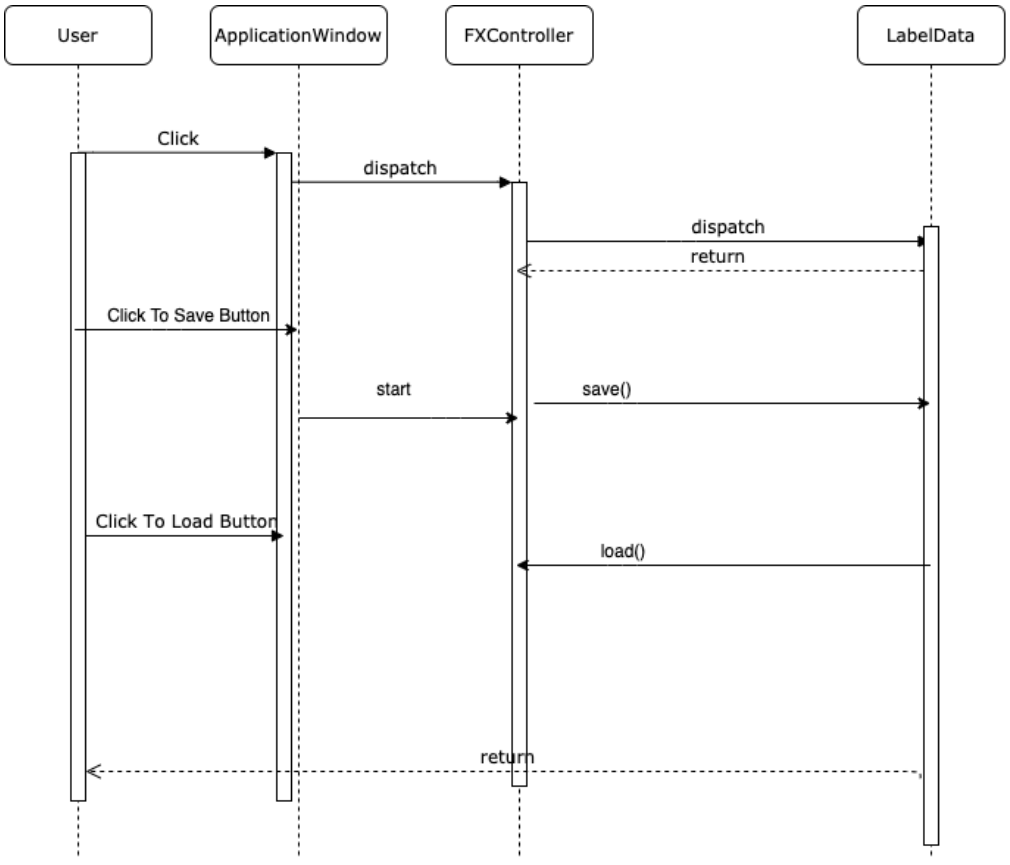
Delete



Undo/Redo



Save/Load



Maintenance Scenarios

Adding Additional Circles

If we want to add more circles to the diagram in the future, a simple way would be to make a copy of the two-circle FXML file, modify it to add a third or fourth circle.
Then we would load each additional FXML in the application class as an individual scene. Finally, we would add a button to the application and accompanying method that would switch scenes to the desired number of circles, and reload the elements created by the user.

Opening Hyperlinks in Elements

In the future, we may want to allow the program to accept hyperlinks in elements, then allow the user to click them to open it in their browser.
This could be achieved by creating a hyperlink attribute for each element, and creating a method that will open the link in their default browser using the Hyperlink class. Then we would add a button in the elementContextMenu associated with that method.