2020

VennCreate Design Document

AUTHOURS:

CHIDALU AGBAKWA (216337784), SHANGRU LI (214488993), JIHAL PATEL (216376436), ROBERT SUWARY (215446016)

LASSONDE SCHOOL OF ENGINEERING | EECS2311 W2020

Contents

List of Figures

1 Introduction

1.1 Purpose

The purpose of this document is to describe the implementation of the VennCreate venn diagram desktop tool as described in the VennCreate requirements document. The VennCreate application is designed to allow users to create customizable venn diagrams.

1.2 Scope

This document describes the implementation details of VennCreate. The software will consist of three major functions. First to allow users to create customizable, elegant, and easy-to-read venn diagrams. Secondly, to be able to export them as a CSV file or as a PNG/JPG file. And lastly to be able to simulate a test environment to allow a user to test themselves on the content of a .txt file of choice.

2 Design Overview

2.1 Description of Problem

Finding similarities and differences between different ideas has been limited in the past to t-charts and other less visual graphical components. VennCreate has provided a unique and fun way for users to find similarities and differences between ideas. Simply launch the application and start entering text to compare.

2.2 Technologies Used

VennCreate is a Java project built on Gradle. It uses the JavaFX (8) framework to handle the GUI building and components.

The target platform will be all OS's, and the development environment is Eclipse IDE.

2.3 System Architecture

Figure 1 depicts the high-level system architecture. The system will be constructed from multiple distinct components:

- **VennCreate UI –** The user interface for creating, editing, and manipulating Venn diagrams.
- **Data Storage –** The users hard drive that will accept the Venn Diagrams as CSV files.
- **Controller (ShapeSceneController, MenuSceneController, TestModeController) –** The Java back-end classes that tell the front-end how to react when the user has made changes.
- **Data Model –** The back-end interface for temporarily storing and organizing data in mathematical sets before they are sent to data storage.

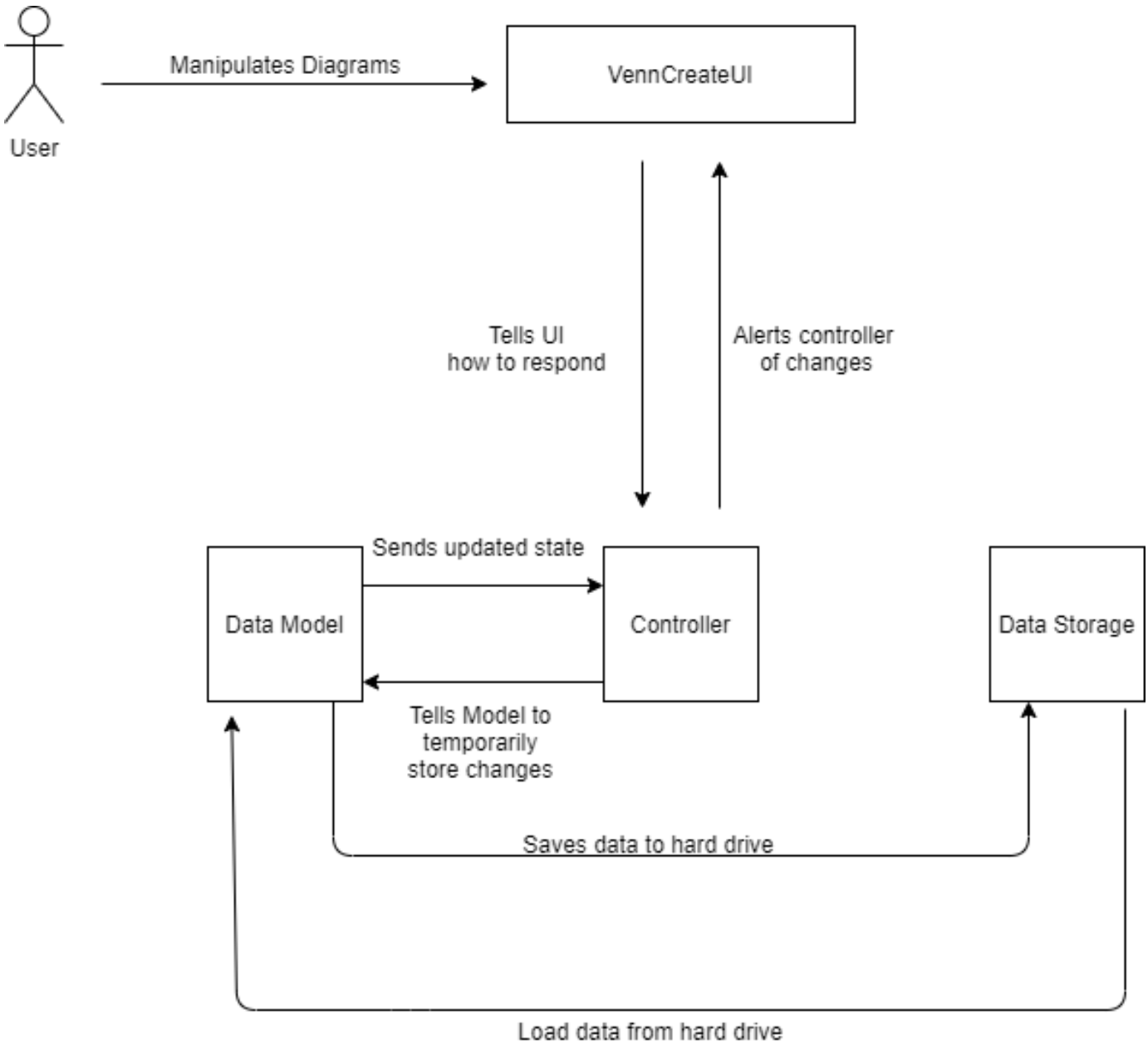

Figure 1: A High-Level Overview of VennCreate Architecture

3 System Operation

The following sequence diagrams depict sequences of events that may occur during a VennCreate session.

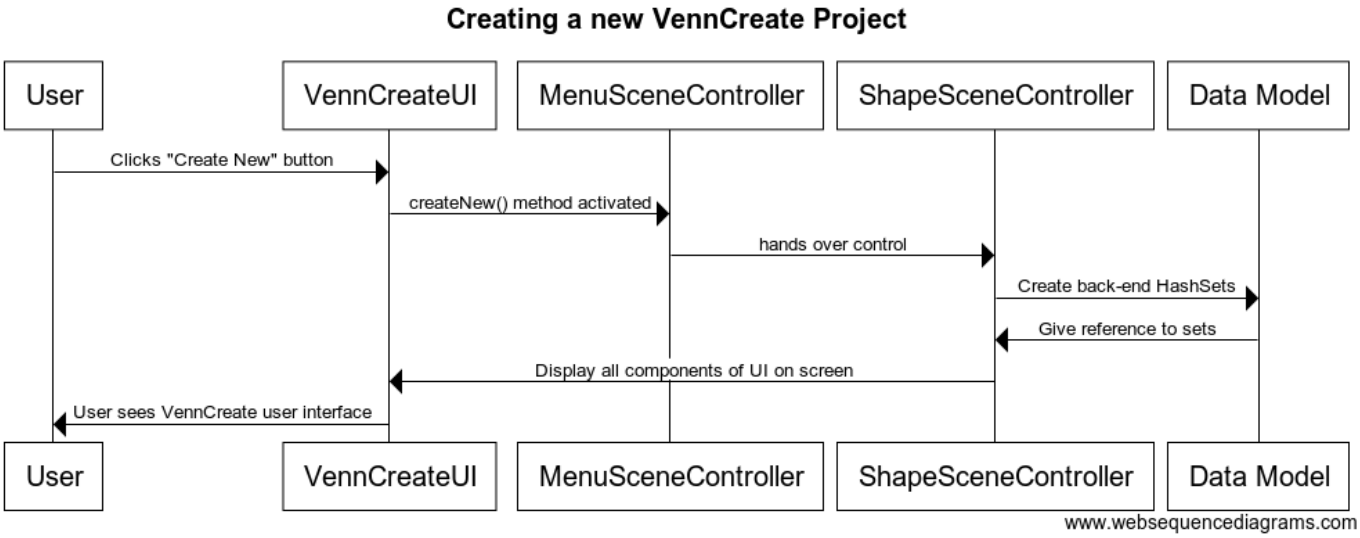

Figure 2: Creating a new Venn Diagram

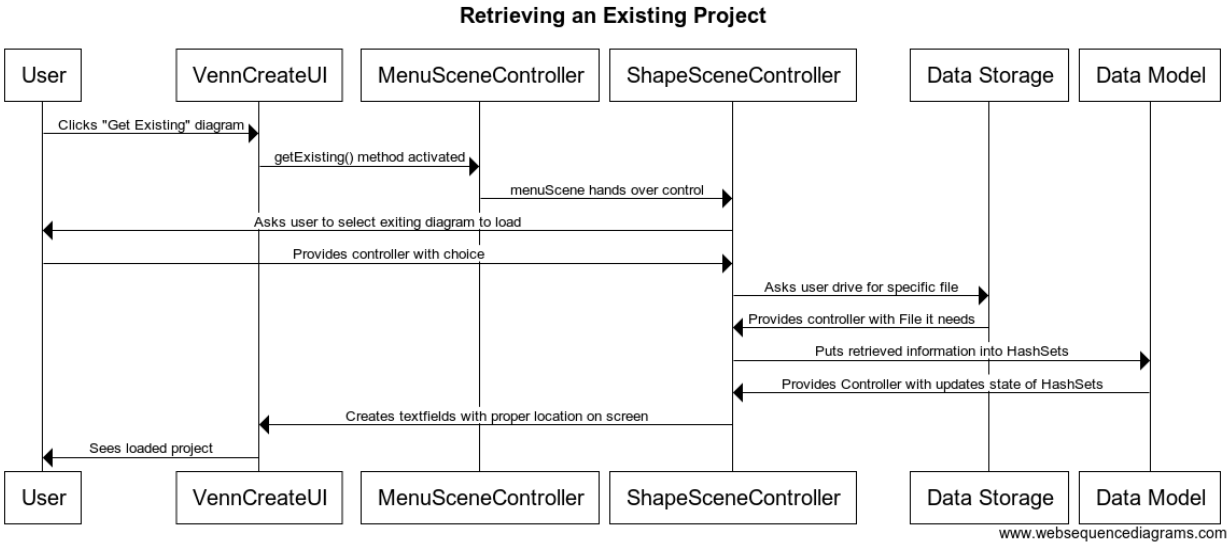

Figure 3: Retrieving an existing diagram from storage

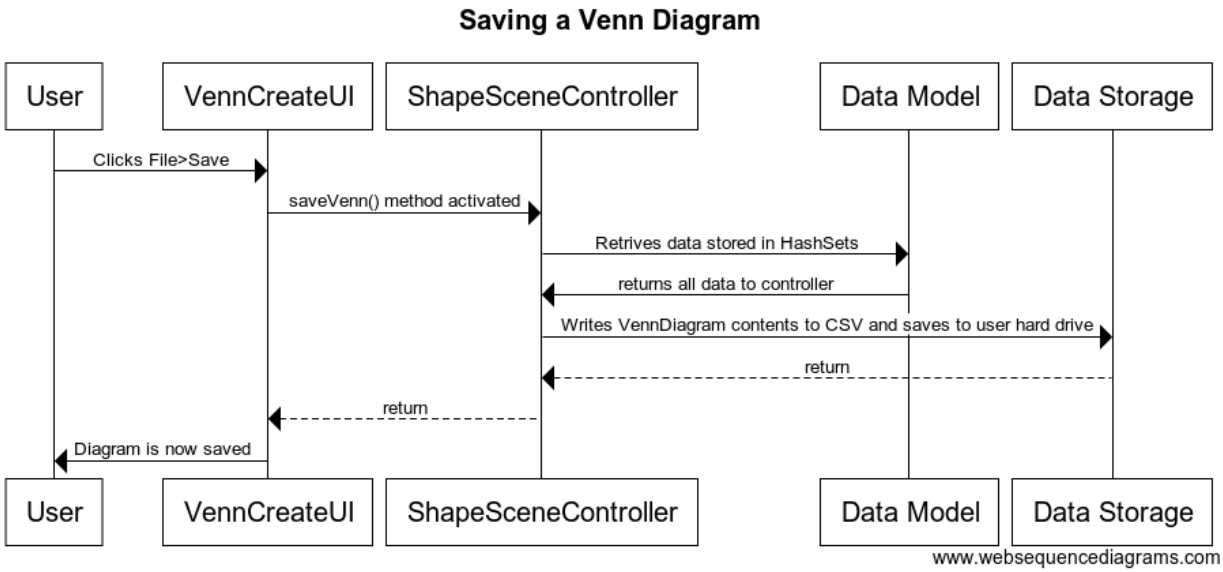

Figure 4: Saving a Venn Diagram

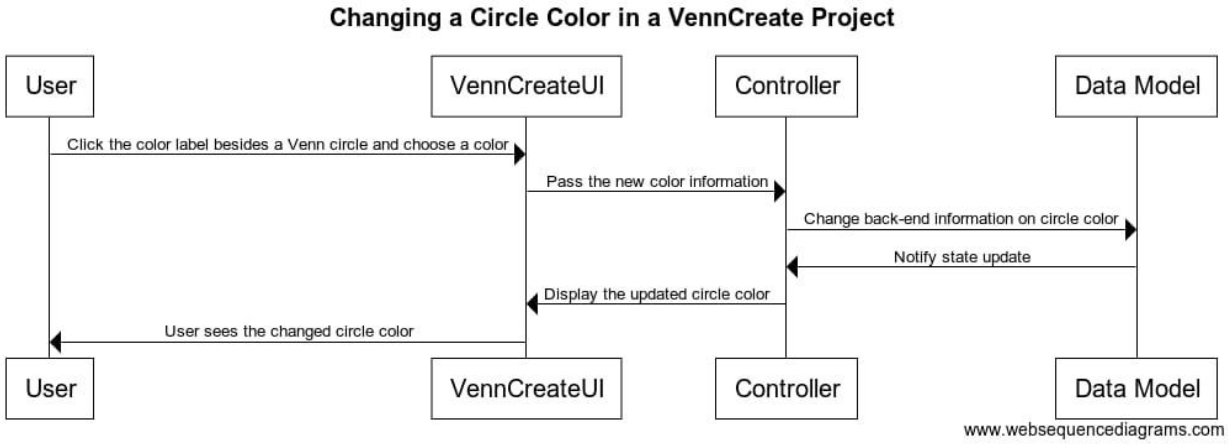

Figure 5: Changing Colors in a Venn Diagram

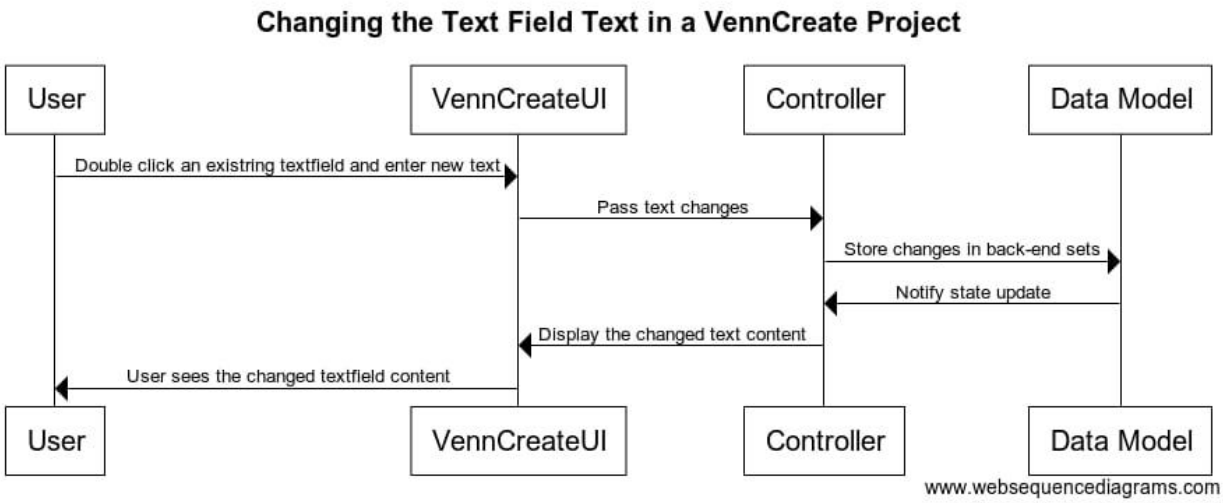

Figure 6: Changing Text Field text

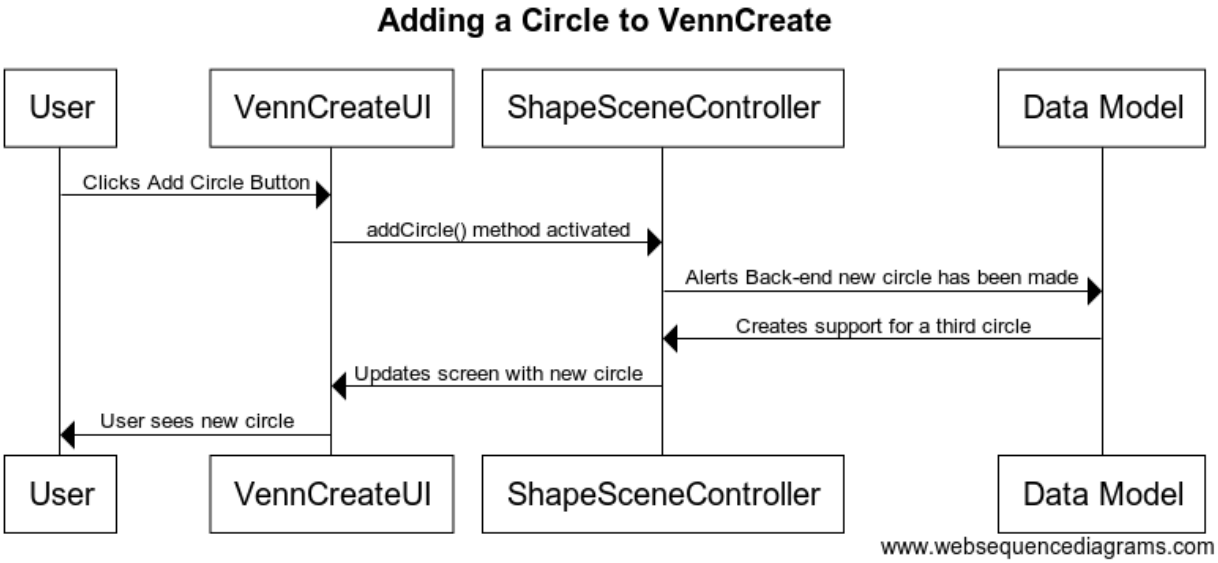

Figure 7 - Adding Circles in VennCreate

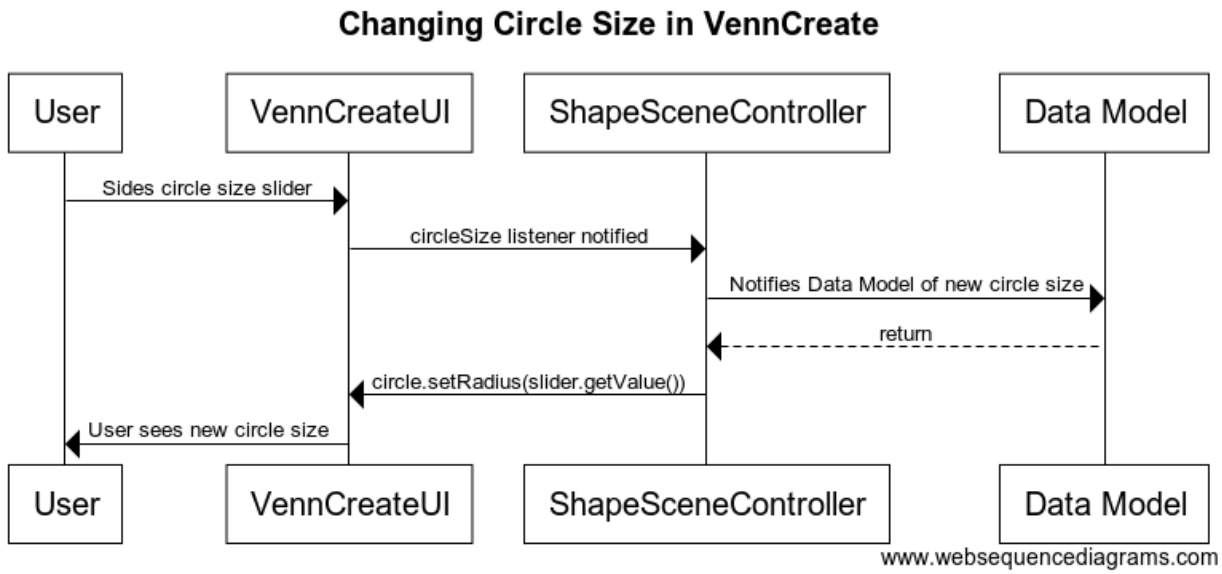

Figure 8 - Changing circle size in VennCreate

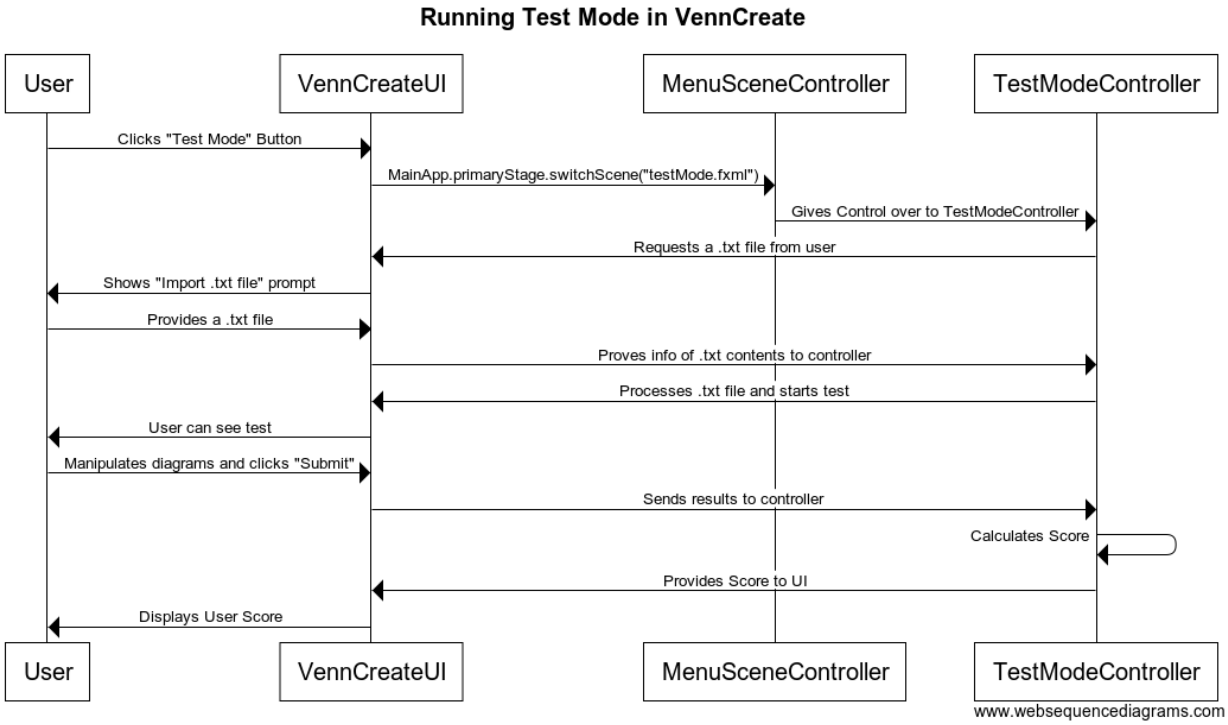

Figure 9 - Running a test in VennCreate

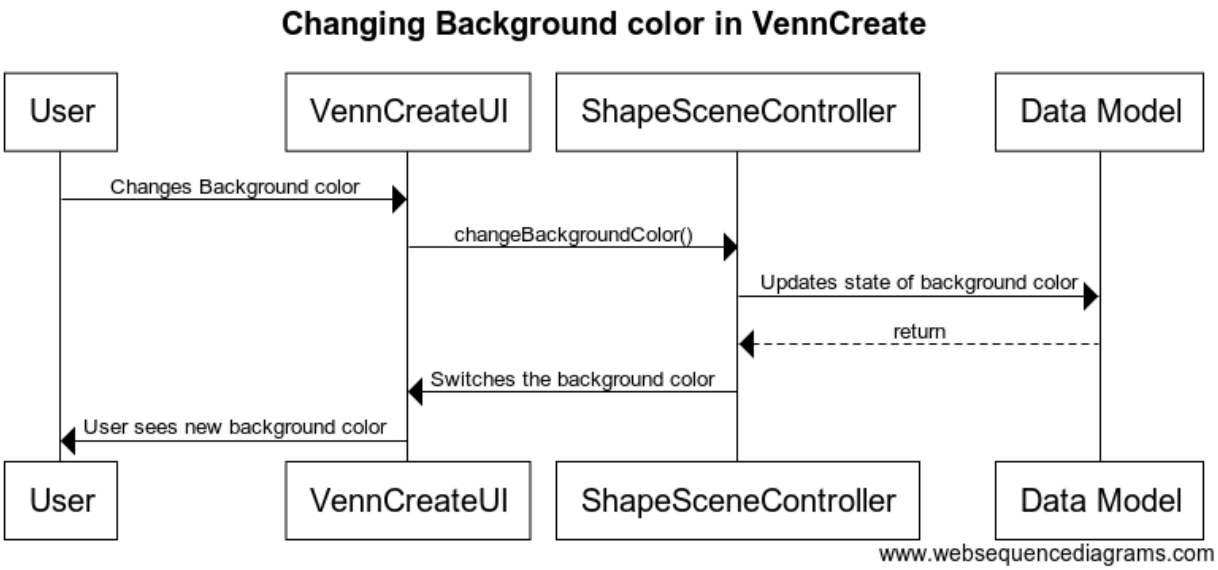

Figure 10 - Changing background color in VennCreate

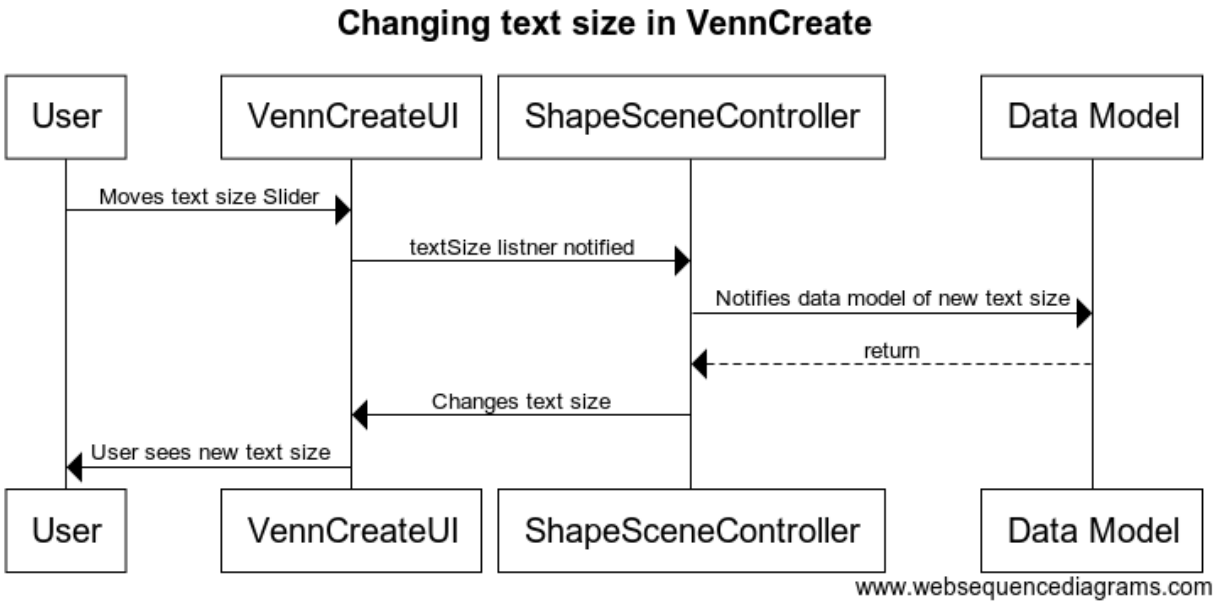

Figure 11 - Changing text size in VennCreate

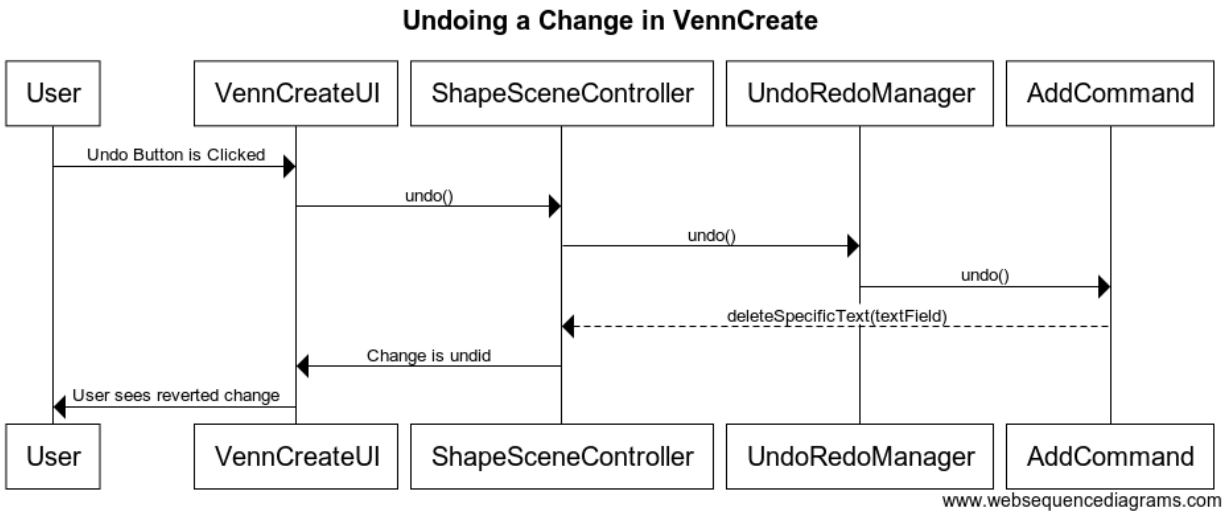

Figure 12 - Undoing a change in VennCreate

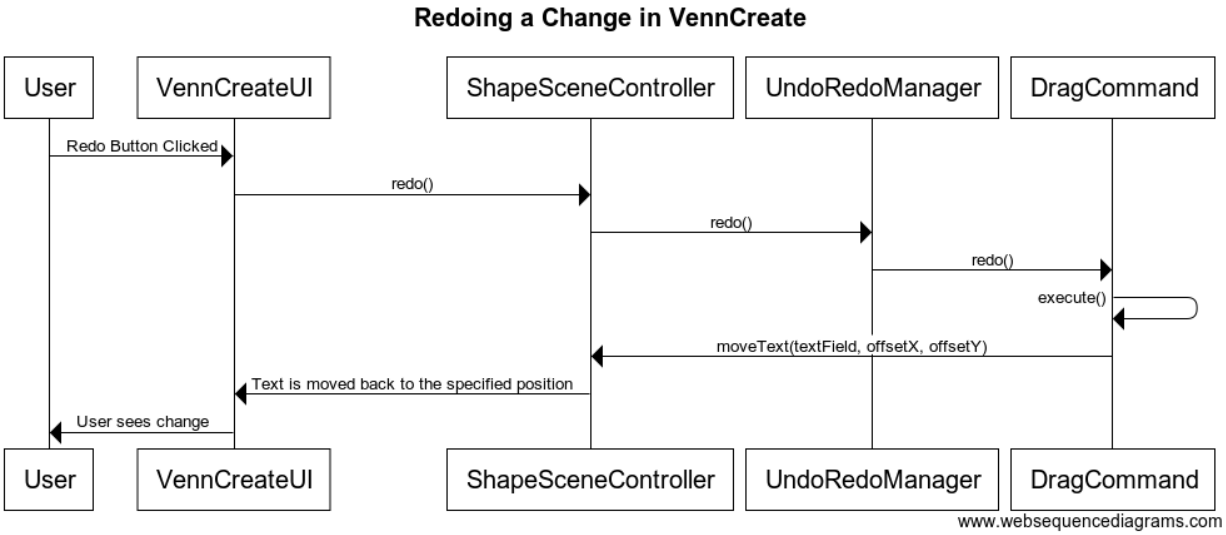

Figure 13 - Redoing a change in VennCreate

4 VennCreate Application Class Diagram

The following figure shows the class layout of VennCreate. The UML diagram only shows important classes and methods. Less significant classes and methods have been omitted.

Figure 14: VennCreate Class Diagram

5 Important Methods

The following tables describe the functionality of important methods that can be found within VennCreate classes.

5.1 MainApp methods

Method: void start(Stage primaryStage)	**Class:** MainApp
Input:	A reference to the main stage of the application
Output:	Void
Description:	This method starts and launches the entire JavaFX application.

Method: void loadRootLayout()	**Class:** MainApp
Input:	Void
Output:	Void
Description:	This method uses an FXMLLoader and loads the rootLayout.fmxl file.

Method: void loadShapeScene()	**Class:** MainApp
Input:	Void
Output:	Void
Description:	This method uses an FXMLLoader and loads the shapeScene.fmxl file, which is the main Scene of the entire application.

Method: void switchScene(String sceneNew, File file)	**Class:** MainApp
Input:	A String object and a File object
Output:	Void
Description:	This method accepts a string with the name of the scene you would like to switch to, and a File if the user is retrieving an existing project. File is null if the user is creating a new project.

Method: void loadMenuScene()	**Class:** MainApp
Input:	Void
Output	Void
Description	This method uses an FXMLLoader and loads the menuScene.fmxl file.

5.2 MenuSceneController methods

Method: void createNew()	**Class:** MenuSceneController
Input:	Void
Output:	Void

Description:	This method switches the current scene on the stage to shapeScene, and hands over control to shapeSceneController

Method: void getExisting(ActionEvent event)	**Class:** MenuSceneController
Input:	ActionEvent e
Output:	Void
Description:	This method opens the user file system so that the user can locate the CSV file they would like to load into VennCreate. When the user has selected a file, it loads the file and hands over control to shapeSceneController.

5.3 ShapeSceneController methods

Method: void addTextToDiagram()	**Class:** ShapeSceneController
Input:	Void
Output:	Void
Description:	Once the user enters text in the main TextField and clicks enter, this method creates a new TextField and places gives it the ability to be dragged.

Method: void loadVenn(File fileToLoad)	**Class:** ShapeSceneController
Input:	Void
Output:	Void
Description:	This method uses an FXMLLoader and loads the rootLayout.fmxl file.

Method: void loadShapeScene()	**Class:** ShapeSceneController
Input:	File fileToLoad
Output:	Void
Description:	If the user chose “Get Existing” as the start option, this method reads all content from the CSV file (fileToLoad) and prints it correctly on a Venn Diagram.

Method: void saveVenn(ArrayList<TextField> write)	**Class:** ShapeSceneController
Input:	ArrayList<TextField> write
Output:	Void
Description:	The input arraylist contains all the text fields the user would like to save to the CSV file. This methods loops through the array list and writes

	the text field text, and text field x and y coordinates to the CSV file.

5.4 VennSet methods

Method: Location getLocation(TextField tf)	**Class:** VennSet
Input:	TextField tf
Output:	Location
Description:	This method uses simple distance formula calculates to decide whether a textfield has been placed in a certain location (Location is an Enum). Location.MIDDLE, Location.LEFT, Location.RIGHT

5.5 TestModeController methods

Method: void importTXT()	**Class:** TestModeController
Input:	Void
Output:	Void
Description:	This method opens the users file explorer and asks the user to import a .txt file needed to run test mode. It then calls all the necessary helper methods in order to do so.

Method: readAndParseTXT(File file)	**Class:** TestModeController
Input:	File file
Output:	int
Description:	This method is called by importTXT to help parse the .txt file imported by the user. It returns an int representing the number of elements that are to be imported into this test, and 0 if an error occurred parsing the txt file. An error will occur if the imported .txt file does not abide to VennCreate standards as specified in the VennCreate user manual.

Method: boolean assignLocation(String text, String location)	**Class:** TestModeController
Input:	String text, String location
Output:	boolean
Description:	This method processes a single line of the .txt file imported by the user and puts the function arguments in a hashmap. This function returns true if the line was successfully parsed and false otherwise.

6 Maintenance Scenarios

6.1 Adding a Vertical Navigation Drawer to shapeScene.fxml file

(I) Add jfoenix dependencies to the project.

(II) Create a new FXML file with the main component being a VBox and style it as you please. This will be the vertical navigation drawer.

(III) Add a JFXDrawer component to the left-hand side of shapeScene.fxml file and make sure the drawer has the same dimensions as the VBox you created.

(IV) Add another VBox field to ShapeSceneController class. This will be the field that holds reference to the VBox you have created in your fxml file.

(V) In the initialize() method in ShapeSceneController, call another method such as loadNavDrawer() that uses an FXMLLoader to load the fxml file containing the VBox you have just created. And add assign it to the new VBox field.

6.2 Implementing an "Add Circle" feature

(I) Decide on a set Location for the new Circle to be added to reside. The optimal location would be in a place that creates an intersection of three circles in the project.

(II) Create a new Circle field for the new circle in ShapeSceneController.

(III) Create an add circle button on the main UI.

(IV) Create an addCircle() method.

(V) On button press, create a new circle object and set its location to the optimal x,y coordinates.

6.3 Implementing an Export as PNG/JPG feature

(I) Add an Export as PNG option to the main menu bar.

(II) Use the CaptureWindow class to capture the users current window.

(III) Activate a FileChooser object that opens the users file system to allow the user to save the PNG anywhere they choose.

6.4 Implementing an Export as PDF Feature

(I) Add an Export as PDF option to the main menu bar.

(II) Use the CaptureWindow class to capture the users current window.

(III) Activate a FileChooser object that opens the users file system to allow the user to save the PDF anywhere they choose.

6.5 Adding Tabs to the application so the user can use more than one VennDiagram at once

(I) Add a TabView object to shapeScene.fxml.

(II) Load a fresh shapeScene.fxml in the other tab.

(III) Create a new ShapeSceneController object for the new tab to use.

6.6 Creating multiple text field entry in application

(I) Add a ListView object to the vertical navigation drawer in scenario 6.1.

(II) When users enter text into the text field above the listview, add the text into the listview.

(III) Add a drag feature so that a user may drag text from the listview into the main venn diagrams.

6.7 Adding a "Tagging" feature so that entries can be grouped

(I) Add a legend section somewhere on the scene, this legend will depict a color with a tag beside it.

(II) Add a right-click option to added text fields to give them a tag.

(III) Change the color of the text fields text to match the tag legend.

6.8 Supporting additional shapes in VennCreate

(I) Turn ShapeSceneController into an abstract class, all other shapes would behave the same as shapeScene, the only thing that would change is the shape they use.

(II) Instead of the leftCircle and rightCircle field in ShapeSceneController, instead use:

Shape leftShape;

Shape rightShape;

as you can now use any shapes as your main two shapes because Circle, Rectangle, Triangle, all inherit from the Shape class.

(III) Say you wanted to make a Rectangle version of shapeScene.fxml, make a RectangleSceneController that extends ShapeSceneController.

6.9 Adding Auto-resizing to scene components

Say you wanted to make the components on the screen such as the circles grow as the user expands and shrinks the window how could you do this?

(I) Add a change listener to the window to listen for changes to the window size, you can do this buy adding a change listener to this.mainApp in ShapeSceneController in the initialize method

(II) As the window is changing you must scale the circle dimensions in the same way