

Software for Editing Multiple Image Document

PROJECT REPORT

Acharya Narendra Dev College

under

ELITE Scholarship Program

[10th June 2019 – 31st July 2019]

Submitted by

Prince Sharma (Dept. of Electronics)

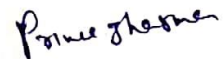
Ankit Negi (Dept. of Electronics)

Guidance:

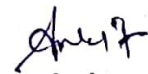
Ms. Vandita Grover (Dept. of Computer Science)

Certificate

The Project report titled 'Software for Editing Multiple Images' is a GUI based software for generating unique documents in image format as per the inputs of document template in image format and CSV file that contains the data. This project carried out by the undersigned students of the Department of Electronics is a record of original and bonafide work done under the Department of Computer Science, Acharya Narendra Dev College, University of Delhi, New Delhi during 4th June – 31st July 2019.



Prince Sharma



Ankit Negi

 - 02/09

Supervisor
Vandita Grover

Principal
Dr. Ravi Toteja

Acknowledgment

We would like to express our gratitude to Ms. Vandita Grover for her invaluable contribution and generous suggestions towards the completion of this project. We feel honoured and privileged to work under her guidance.

We are thankful to our principal, Dr. Ravi Toteja for running ELITE scholarship programme during 5th June – 31st July that helps students to learn new things beyond the scope of the curriculum.

Prince Sharma
Ankit Negi

Contents

1. INTRODUCTION	
1.1 Overview.....	5
1.2 Objective.....	5
2. TOOLS AND TECHNOLOGY USED IN BUILDING THE PROJECT	
2.1 Tools and Technology Used.....	6
2.2 Minimum Hardware and Software Requirement.....	7
3. DESIGN OF THE SOFTWARE.	
3.1 DFDs of The Software.....	8
3.2 Data Dictionary.....	9
3.3 Description of Module used in Level 1 DFD.....	10
4. GUI DESIGN OF THE SOFTWARE	
4.1 Front-end of the Application.....	12
5. BACK END PROCESSING	
5.1 Back End of the Software was Implemented in the Following Manner.....	15
6. CODING.....	16
7. FUTURE SCOPE.....	18
8. USER MANUAL	
8.1 Hardware Requirements.....	20
8.2 Software Requirements.....	20
8.3 Steps to Use the Software.....	20
9. References.....	22

Chapter 1

Introduction

The use of computers has made cumbersome, repetitive, and time-consuming tasks a lot easier, efficient, and productive. Take for instance, when the users need to create similar documents with unique information on each or if the user wants to generate marriage invitation cards for the thousands of people. Writing the names of all of them is no doubt very tedious and inefficient task. In such cases, a user may either buy a paid software like Microsoft Word or hire a programmer who can write a script to complete that task, both of which are quite expensive. The goal of this project was to automate customized documents, where the tool can generate documents, like certificates in bulk, customized with the awardee information on each certificate.

1.1 Overview

The software has the potential of fulfilling the requirement of image editing at a large scale as it eases a computer user to add unique pieces of information by running a simple image editing algorithms. Also, this project presents an opportunity for potential developers to add some extra features to it which can support a wide range of formatting within the image.

1.2 Objective

The goal of this project was to create a free GUI based software to automate the task of mass production of certificates (or Image document) as per the input of large data in CSV format and eliminate the tedious task of manually writing on each certificate. Editing an image is largely a time-consuming task for a computer user. Users sometimes come across problem when an image needs to be edited such as writing some information on it. This can be done easily for fewer numbers of different pieces of information. But definitely tiresome for the larger one. So, this software automates the tasks of writing on the image.

Chapter 2

Tools and Technology Used

2.1 Tools and Technology used

This section gives a brief description of the various tools and techniques learned and used to implement the software successfully.

- i. **Java:** Java is a general-purpose programming language that is class-based, object-oriented (although not a pure object-oriented language, as it contains primitive types), and designed to have as few implementation dependencies as possible. It has cross platform operability features meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but it has fewer low-level facilities than either of them. Java is used to develop mobile apps, web apps, desktop apps, games and much more. While building the software Java 12 version has been used.
- ii. **JavaFX:** JavaFX is an open-source, next-generation client application platform for desktop, mobile and embedded systems built on Java. It is a software platform for creating and delivering desktop applications, as well as rich Internet applications (RIAs) that can run across a wide variety of devices. JavaFX is intended to replace Swing as the standard GUI library for Java SE, but both will be included for the foreseeable future. JavaFX has support for desktop computers and web browsers on Microsoft Windows, Linux, and macOS.
- iii. **FXML:** FXML is an XML-based user interface Mark-up language created by Oracle Corporation for defining the user interface of a JavaFX application. It provides a convenient alternative to constructing such graphs in procedural code and is ideally suited to defining the user

interface of a JavaFX application since the hierarchical structure of an XML document closely parallels the structure of the JavaFX scene graph. However, anything that is created or implemented in FXML can be expressed using JavaFX directly.

- iv. Eclipse: Eclipse is an integrated development environment (IDE) used in computer programming, and in 2014 was the most widely used Java IDE in one website's poll. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins.
- v. Scene builder: JavaFX Scene Builder is a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding. Users can drag and drop UI components to a work area, modify their properties, apply style sheets, and the FXML code for the layout that they are creating is automatically generated in the background. The result is an FXML file that can then be combined with a Java project by binding the UI to the application's logic.

2.2 Minimum Hardware and Software Requirement

The software requires a minimum of 2 GB RAM equipped with compatible Windows or Linux operating system. At least Intel 4004 Microprocessor should be there for smooth running of the application.

Chapter 3

Software Design

This chapter describes the design and implementation of the software highlighting the important components and functionality.

3.1 Design Constraints

Size of the image viewer is fixed. Whenever an image is imported it first displays half portion of the image that is, $\frac{1}{2}$ width and $\frac{1}{2}$ height. This is done so that viewport do not remain null. Doing this eliminates the null condition of the viewport. This constraint has been compensated by adding pan and normal zoomable and effect. So, image can be viewed wholly fitting the image viewer's default. This enables the user to click on a certain point more precisely.

3.2 System Design

The complete functioning of the software is explained in the Level 1 DFD below. Data-flow diagram (DFD) is a way of representing a flow of data of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself.

3.2.1 Level 1 DFD

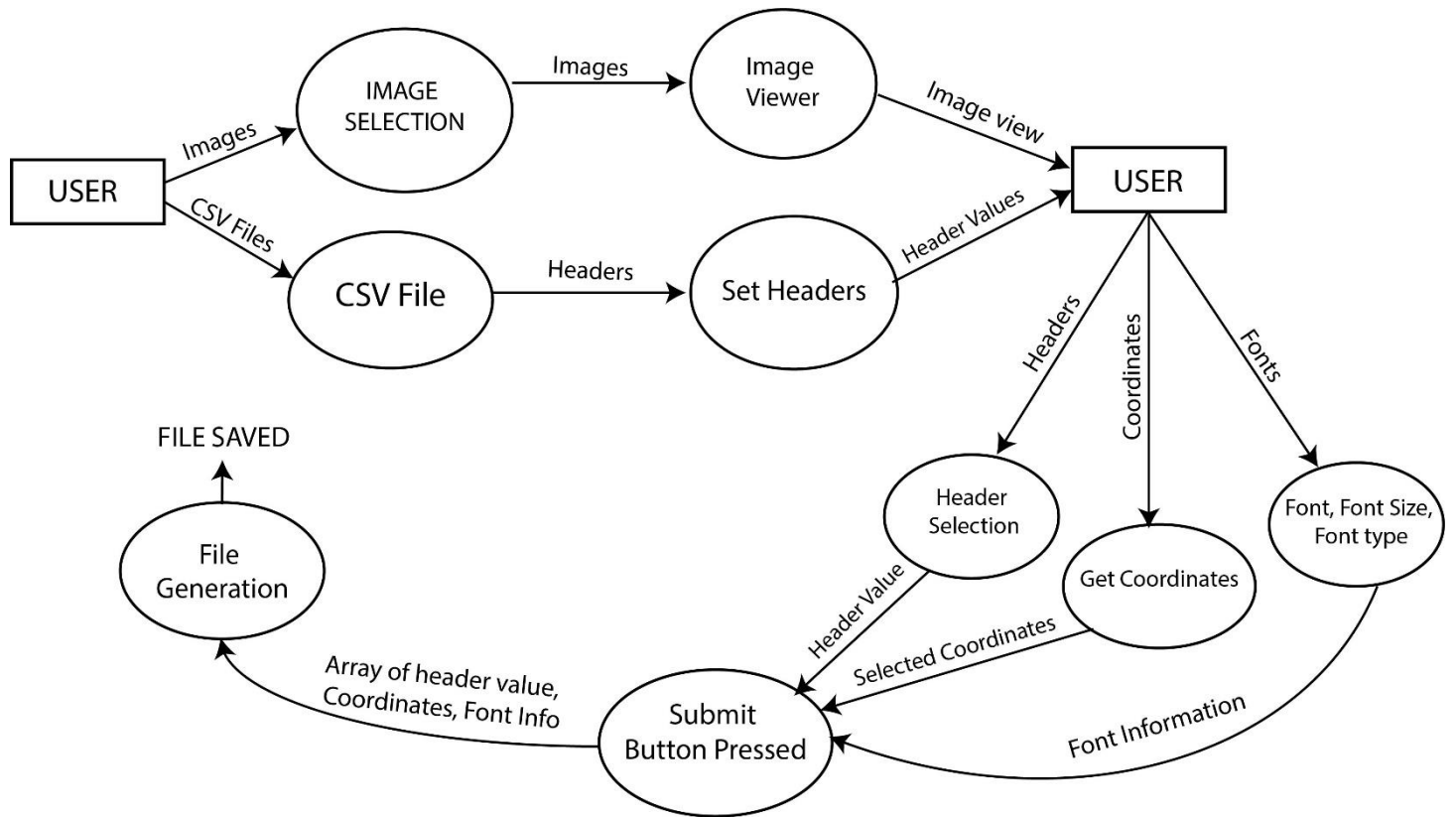


Figure 3.1: Level 1 DFD

3.2.2 Data Dictionary

Field Name	Data Type	Description
Images	Image	The Image class represents graphical images and is used for loading images from a specified URL.
Header	String	It is the first row of the CSV file which describes each column of the CSV file.
Coordinates	Integer	It contains a pair of integers which represent x and y coordinate respectively.

ImageView	Node	This class allows resizing the displayed image (with or without preserving the original aspect ratio) and specifying a viewport into the source image for restricting the pixels displayed by this ImageView.
CSV File	csv	CSV is a simple file format used to store tabular data, such as a spreadsheet or database. Files in the CSV format can be imported to and exported from programs that store data in tables, such as Microsoft Excel or OpenOffice Calc.

3.3 Component Design and Module Description

3.3.1 Image and CSV File Selection

Input Click on button event received command

Processing These two functions open file chooser Dialog Box for the user and ask the user to select the files.

Output File path is returned in the output in the form of string.

3.3.2 Coordinates Selection

Input Clicked on a position of the image event received.

Processing This module detects the point where the mouse is clicked on the image viewer and displays the corresponding coordinates of the real Image.

Output X and Y coordinates in the form of Integer is returned.

3.3.3 Headers Selection

Input Click event on the required header from the dropdown list.

Processing This module extracts all the headers from the CSV file and stores in Combo Box lists. Later when any value is selected. It is further stored in an array.

Output Value of selected header in the form of String.

3.3.4 Joining coordinates with headers

Input X, Y coordinates integer value, the header value

Processing In this module, header and its corresponding Font Size, Font and Coordinates on the image where it is to be printed are stored a dynamic array. This helps to join to all the properties with the headers of CSV.

Output Structured dynamic array containing integer values of x and y coordinates and header value such that they are at the same index.

3.3.5 File Generations

Input Structured dynamic array containing headers, fonts, font size and x-y coordinates of the image.

Processing Once the dynamic array is stored with a required data which includes headers, Font, Font Size and X-Y Coordinates of the Image where the text is to be written. This module runs the editing script of the software and edits the imported image and save all the image files in a folder of user's choice.

Output File get saved in the chosen location.

Chapter 4

Front End of The Software

Front end of the software was implemented in the following manner:

- Step 1. JavaFX is used over Swing because JavaFX has a decent number of UI components available but lesser than what Swing provides.
- Step 2. A program is written with a name 'Controller' which contains all the algorithm that runs the software in the back end.
- Step 3. SceneBuilder is used to write the FXML code and embedded that with our controller program.
- Step 4. The design is included with various buttons which runs various functions in the controller program and ImageViewer which shows the imported image.
- Step 5. ImageViewer was further upgraded to have a zoom in and out effect which gave the precise coordinates of the point where the mouse is clicked. Also, it improved the image quality that viewed earlier.
- Step 6. File Chooser initially used JFileChooser which was quite slow and had a little control over the customization. So, we switched to Window's File Chooser which opens in a new stage.

The front end of the application

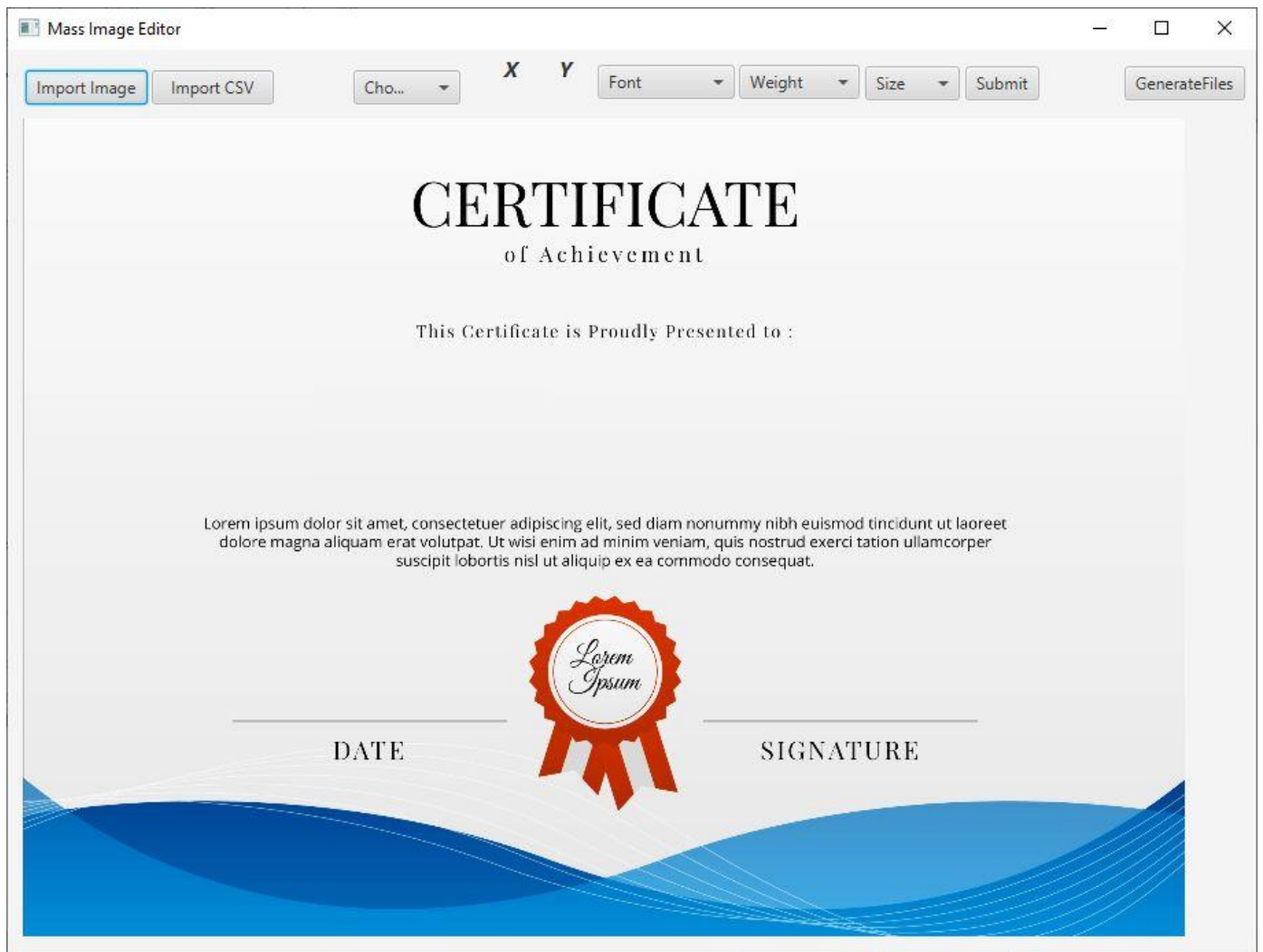


Figure 4.1: Front end (1/2) of the application

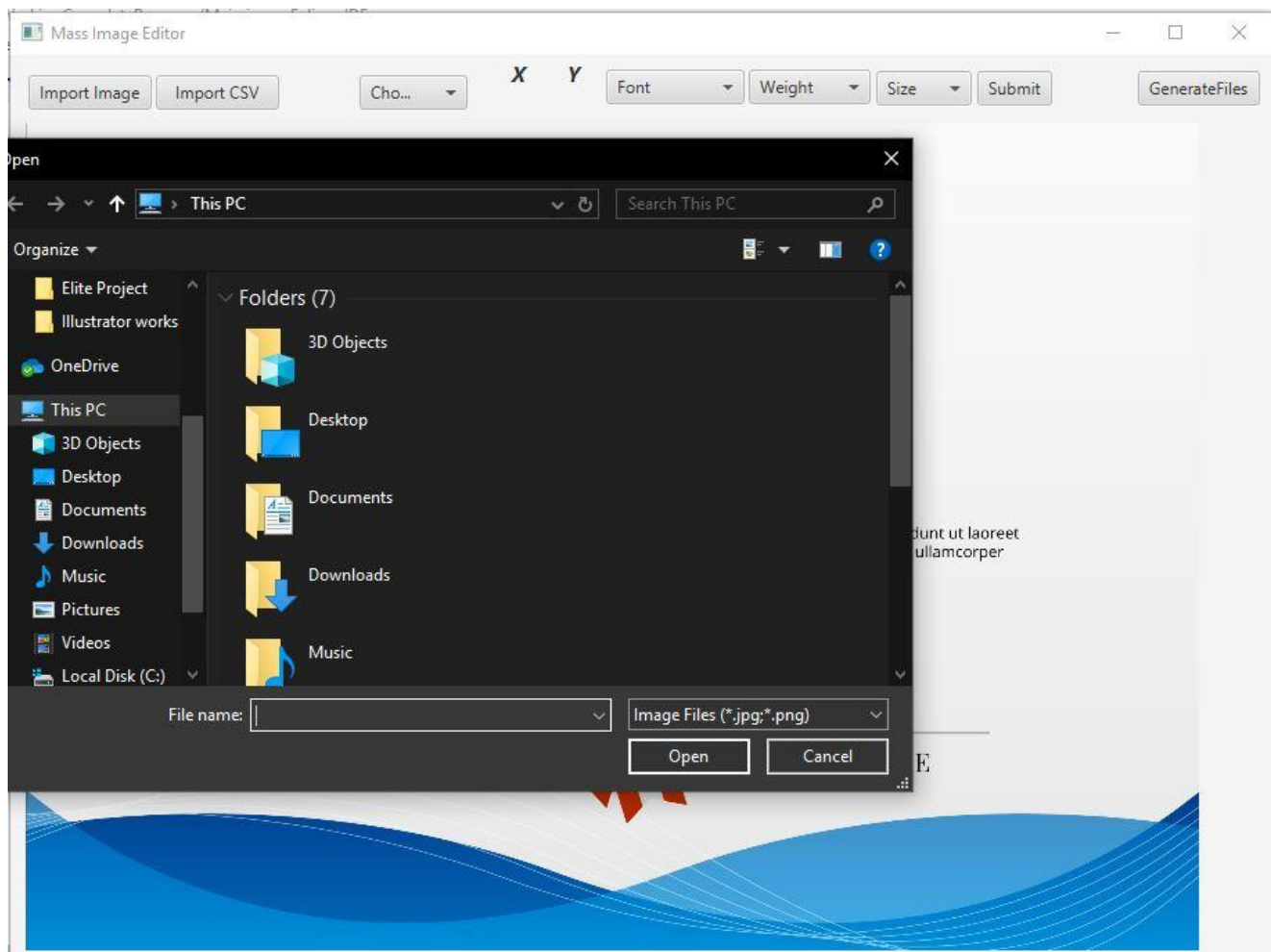


Figure 4.2: Front end (2/2) of the application

Chapter 5

Back End Processing

Back end of the software was implemented in the following manner

1. The program contains four files Design.fxml, Controller.java, Main.java, setComboBoxFontFamilies.java, ImageViewer.java
2. Main.java makes an interconnection between Controller.java and Design.fxml file. Main creates a stage and places the design over there.
3. Controller.java performs most of the tasks like editing and have a connection with other modules to perform various tasks like image viewing with zoom in and out effect, finding coordinates of the image, setting ComboBox values in the dropdown, etc. Objects of other classes is created here and their function is taken is used. Additionally it performs the main task of editing the image under the function ImageEdit()
4. ImageViewer is the class which is called by controller.java for viewing the image and featuring the zoom in and out effect.

Chapter 6

Coding

Figure 6.1: Shows the back end working in the console

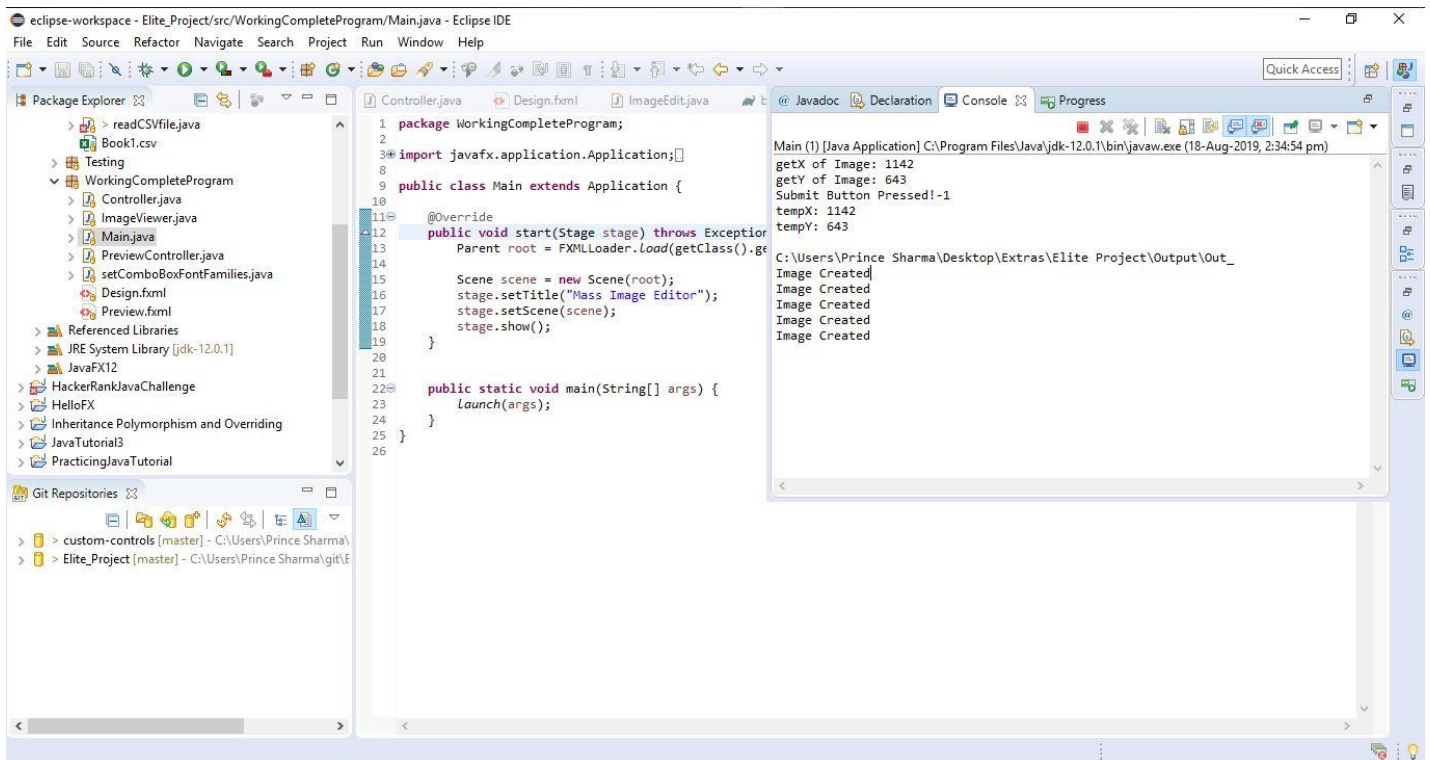


Figure 6.2: Main.java

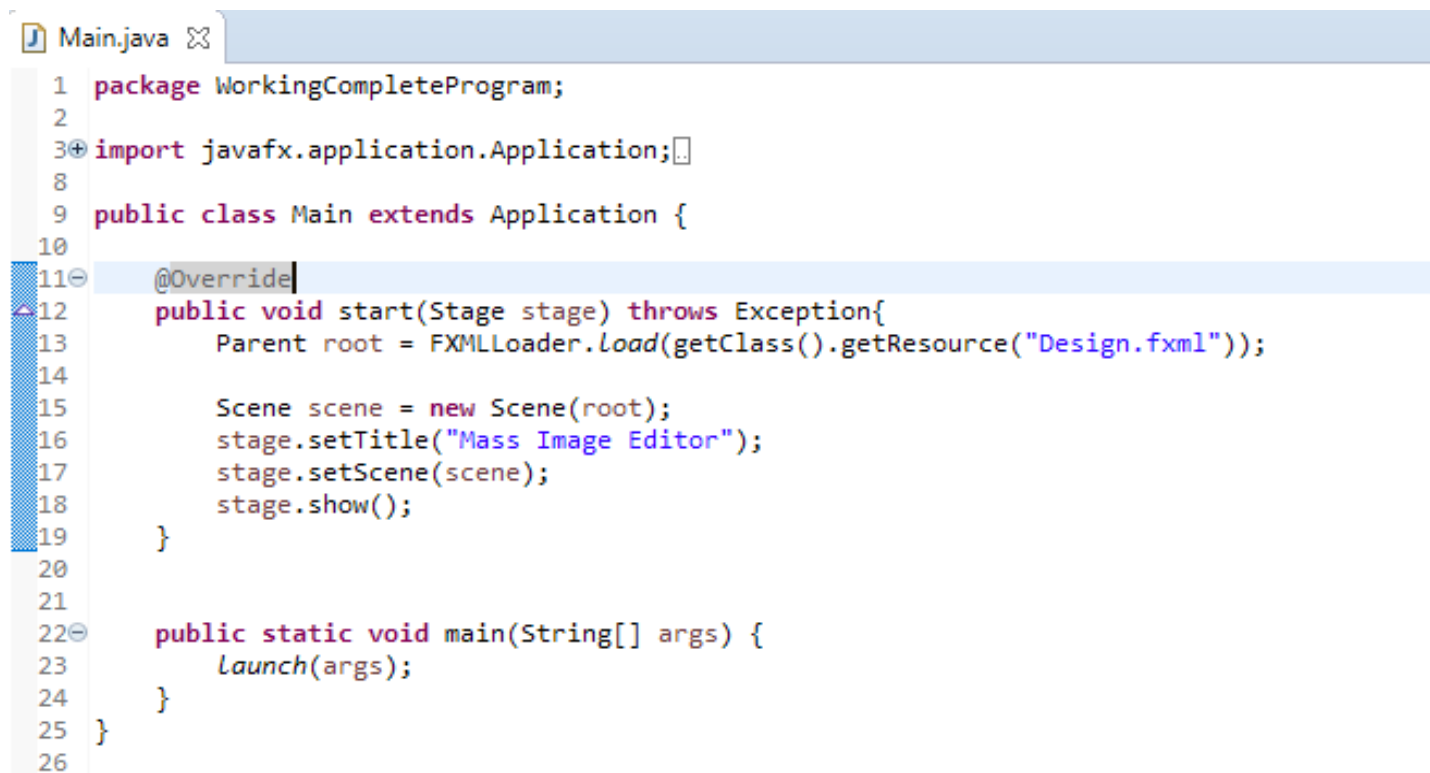


Figure 6.2: Design.fxml

```

Design.fxml
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <?import javafx.scene.control.Button?>
4 <?import javafx.scene.control.ComboBox?>
5 <?import javafx.scene.control.Label?>
6 <?import javafx.scene.image.ImageView?>
7 <?import javafx.scene.layout.AnchorPane?>
8 <?import javafx.scene.text.Font?>
9
10 <AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="662.0" prefWidth="910.0" xmlns="http://javafx.com/javafx/11.0.1"
11 <children>
12 <Button layoutX="13.0" layoutY="15.0" mnemonicParsing="false" onAction="#ImportButtonPressed" prefHeight="17.0" prefWidth="89.0" text="Import Image" />
13 <ImageView fx:id="imageview" fitHeight="597.0" fitWidth="889.0" layoutX="11.0" layoutY="50.0" pickOnBounds="true" preserveRatio="true" />
14 <ComboBox fx:id="comboBox" layoutX="252.0" layoutY="15.0" prefHeight="25.0" prefWidth="78.0" />
15 <Label layoutX="352.0" layoutY="23.0" text="Label" fx:id="x" />
16 <Label fx:id="y" layoutX="393.0" layoutY="23.0" text="Label" />
17 <Button layoutX="698.0" layoutY="12.0" mnemonicParsing="false" onAction="#SubmitButtonPressed" text="Submit" />
18 <Label layoutX="362.0" layoutY="4.0" text="X">
19 <font>
20 <font name="System Bold Italic" size="15.0" />
21 </font>
22 </Label>
23 <Label layoutX="402.0" layoutY="4.0" text="Y">
24 <font>
25 <font name="System Bold Italic" size="15.0" />
26 </font>
27 </Label>
28 <Button layoutX="814.0" layoutY="12.0" mnemonicParsing="false" onAction="#GenerateFiles" text="Generate Files" />
29 <Button layoutX="105.0" layoutY="15.0" mnemonicParsing="false" onAction="#ImportCSV" prefHeight="17.0" prefWidth="89.0" text="Import CSV" />
30 <ComboBox fx:id="Fonts" layoutX="430.0" layoutY="11.0" prefHeight="25.0" prefWidth="100.0" promptText="Font" />
31 <ComboBox fx:id="Weight" layoutX="533.0" layoutY="11.0" prefHeight="25.0" prefWidth="88.0" promptText="Style" />
32 <ComboBox fx:id="size" layoutX="625.0" layoutY="12.0" prefHeight="25.0" prefWidth="69.0" promptText="Size" />
33 </children>
34 </AnchorPane>
35

```

Figure 6.3: Controller

```

Controller.java
153 // This function will let the use choose the csv file and add the headers in comboBox
154 public void ImportCSV() {
155     strCSV = FileChooser2(null);
156     ComboBoxSETTER();
157     Alert a = new Alert(AlertType.NONE);
158
159     // set alert type
160     a.setAlertType(AlertType.INFORMATION);
161     a.setHeaderText(null);
162
163     a.setContentText("CSV File Added Successfully \n\nPlease select header and choose the position in image");
164     // a.setContentText("Please add CSV File");
165
166     // show the dialog
167     a.show();
168 }
169
170 // This function is used to store the header and the pixel position in ArrayList and add the default font if not entered by the user
171 public void SubmitButtonPressed(ActionEvent event) {
172     String SelectedHeader = comboBox.getValue();
173     str.add(SelectedHeader);
174     //this.x.setText(Integer.toString(tempX));
175     //this.y.setText(Integer.toString(tempY));
176     j++;
177     Point2D point = img.GetThePoint();
178     int X = (int)point.getX();
179     int Y = (int)point.getY();
180     tempXarray.add(X);
181     tempYarray.add(Y);
182
183     String Fontvalue = "";
184
185     if(!Fonts.getSelectionModel().isEmpty()) {
186         Fontvalue = Fonts.getValue();
187     }
188     else {
189         Fontvalue = "Calibri";
190     }
191     Font.add(Fontvalue);
192

```

```
Controller.java  ImageViewer.java  setComboBoxFontFamilies.java

1 package WorkingCompleteProgram;
2
3 import java.awt.Color;
4
44
45 public class Controller implements Initializable{
46     @FXML private ImageView imageview, imageview1;
47     @FXML private ComboBox<String> comboBox;
48     @FXML private ComboBox<String> Fonts;
49     @FXML private ComboBox<String> Weight;
50     @FXML private ComboBox<Integer> size;
51     @FXML private Label x;
52     @FXML private Label y;
53     private List<Integer> tempXarray = new ArrayList<>();
54     private List<Integer> tempYarray = new ArrayList<>();
55     private List<Integer> fontsize = new ArrayList<>();
56     private List<Integer> style = new ArrayList<>();
57     private List<String> Font = new ArrayList<>();
58     private BufferedReader br;
59     private int j = 0;
60     private List<String> str = new ArrayList<>();
61     private String strImage="", strCSV="";
62     Integer size1[] = {8,9,10,11,12,14,16,18,20,22,24,26,28,36,48,72};
63     ImageViewer img = new ImageViewer();
64     String filename = "";
65
66     // This will align the header in center of the given pixel position properly
67     public void DrawCenterAlignedString(String key, int x, int y, Graphics graphics) {
68         int stringlen = (int)graphics.getFontMetrics().getStringBounds(key, graphics).getWidth();
69         int reduce = stringlen/2; //alignment
70         x = x-reduce;
71         graphics.drawString(key, x, y);
72     }
73
74
75     setComboBoxFontFamilies set = new setComboBoxFontFamilies();
76
```

Figure 6.4: setComboBoxFontFamilies.java

```
Controller.java  Design.fxml  ImageEdit.java  build.gradle  Main.java  setComboBoxF...  PreviewCont...  11

1 package WorkingCompleteProgram;
2
3 import javafx.collections.FXCollections;
4 import javafx.collections.ObservableList;
5 import javafx.scene.text.Font;
6
7 public class setComboBoxFontFamilies {
8     private ObservableList<String> list1 = FXCollections.observableArrayList(Font.getFontNames());
9
10    public ObservableList<String> setComboBox(){
11        list1 = FXCollections.observableArrayList(Font.getFontNames());
12        return list1;
13    }
14    public int Font(String str) {
15        if(str.equals("PLAIN"))
16            return 0;
17        if(str.equals("ITALIC"))
18            return 2;
19        if(str.equals("BOLD"))
20            return 1;
21        return -1;
22    }
23 }
24
```

Chapter 6

Future Scope

The software can be improved to perform various other tasks like adding a series of different and unique images and vectors in addition to mere fonts and texts. The software can be tweaked to work for various other document formats like pdf, Docx, XPS, etc. Also, providing some additional formatting tools and designing. The project has been uploaded on GitHub so that other developers can contribute to the projects and provide additional features.

Chapter 7

User Manual

7.1 Hardware Requirements

- (i) Keyboard
- (ii) Mouse
- (iii) CPU
- (iv) Monitor

7.2 Software Requirements

User must have installed the operating system (Windows OS, Linux OS, Mac OS). User must have installed Java in their operating system.

7.3 Steps to use the software

Step 1: Click on Import Image. File Chooser is popped up to select an image. The selected image is displayed in the ImageViewer area. Wherever mouse is clicked on the image, it will display the corresponding coordinates under X and Y label.

Step 2: The user clicks on Import CSV. This puts all the headers into the ComboBox dropdown list. This facilitates the user to choose from the list of available headers corresponding to the CSV file imported.

Step 3: Click on the ComboBox to select the header. (The data under this column will be written on the image coordinate selected once the submit button is pressed.)

Step 4: Mouse is clicked on the position of the image where the selected header is to be written.

Step 5: The user then selects the font, font size, font style.

Step 6: Now after going through the previous steps, the user clicks on the submit button which submits the required information including header, header position on the image, font size, font and style of the header.

Step 7: The user repeats step 3-6 until all the required headers to be printed on the imported image template is selected.

Step 8: The user clicks on 'Generate Files' button. This pops up a path chooser. Here the user selects the path where the generated image documents are to be saved. Once the path is selected, a name is given without extension and clicked 'save'.

References

1. <https://www.youtube.com/watch?v=STIHzuVmIG4&list=PLoodc-fmtJNYbs-gYCdd5MYS4CKVbGHv2>
2. <https://stackoverflow.com/questions/45027087/how-to-edit-image-by-java-program>
3. <https://openjfx.io/openjfx-docs/>
4. <https://gist.github.com/james-d/ce5ec1fd44ce6c64e81a>
5. <https://www.coursera.org/specializations/java-programming?>