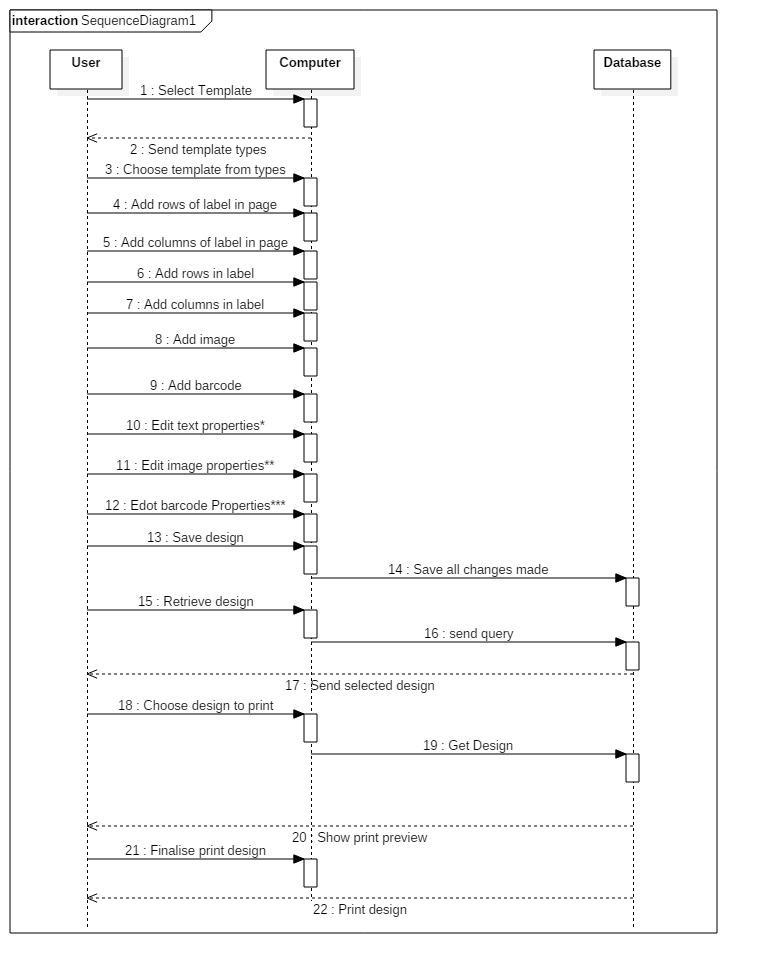
1. **Introduction**
   1. **Purpose of this document:** The document is intended to provide Software Design Specifications of the Label and Barcode Printer. It displays the class diagrams, the state diagram and the sequence diagram. It helps the user understand the design and the procedure of the software.
   2. **Product Scope:** Primarily, the scope of this product is to generate different type of labels. Labels are most commonly used among many organisations in order to keep organised their product/item and serialised. Our software targets many device families, like Laptops/Desktops, Mobile phones and Tablets. All these device families should be running with Windows 10 or above to support our product.
   3. **References**

(a)<https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjU15SL35vPAhUFeD4KHaBXC-cQFggeMAA&url=https%3A%2F%2Fchannel9.msdn.com%2FSeries%2FWindows-10-development-for-absolute-beginners%2FUWP-001-Series-Introduction>

(b)<https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&sqi=2&ved=0ahUKEwihn5Sa35vPAhWEjZQKHbI3A-QQFggcMAA&url=https%3A%2F%2Fmsdn.microsoft.com%2Fen-us%2Fwindows%2Fuwp%2Fget-started%2Funiversal-application-platform-guide&uh>

**1.4 SDS Organisation:** The Document is divided into 4 parts. The first is the introduction which gives an idea as to what is to be expected in the document. The second part is System Architecture which has the various diagrams needed in the project and they show the flow of the software. The third part gives the details about the components of the software and the fourth part talks about the design trade-offs.

1. **System Architecture**
   1. **Overview of Modules/Components**
      1. **Sequence Diagram**



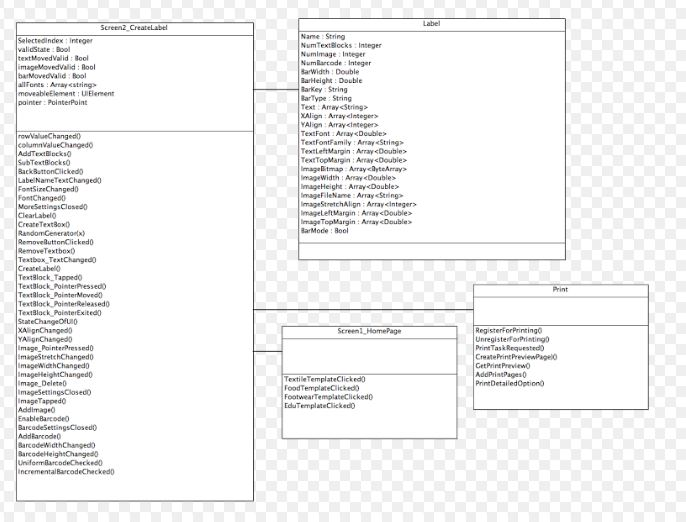
Text properties\* : editing properties like font, size, alignment

Image properties\*\*:editing properties like size, alignment

Barcode properties\*\*\*: editing properties like type, key, width, height,mode

The sequence diagram has 3 lifelines, User, Computer and the Database that will carry out the entire process of creating a label and printing it.

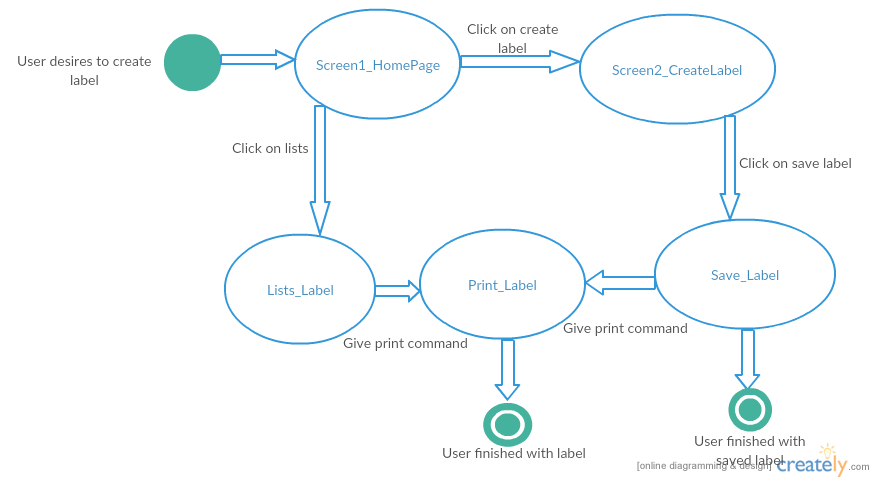
**2.1.2 Class Diagram**



There are 4 classes:

|  |  |  |
| --- | --- | --- |
| **Class** | **Variables** | **Methods** |
| Screen2\_CreateLabel | * SelectedIndex:Integer * validState:Bool * textMovedValid:Bool * barMovedValid:Bool * allFonts:Array<string> * moveableElement: UIElement * Pointer: PointerPoint | * rowValueChanged() * columnValueChanged() * AddTextBlocks() * SubTextBlocks() * BackButtonClicked() * LabelNameTextChanged() * FontSizeChanged() * FontChanged() * MoreSettingsClosed() * ClearLabel() * CreateTextBox() * RandomGenerator(x) * RemoveButtonClicked() * RemoveTextbox() * Textbox\_TextChanged() * CreateLabel() * TextBlock\_Tapped() * TextBlock\_PointerPressed() * TextBlock\_PointerMoved() * TextBlock\_Released() * TextBlock\_PointerExited() * StateChangeOfUI() * XAlignChanged() * YAlignChanged() * Image\_PointerPressed() * ImageStretchChanged() * ImageWidthChanged() * ImageHeightChanged() * Image\_Delete() * ImageSettingsClosed() * ImageTapped() * AddImage() * Enablebarcode() * BarcodeSettingsClosed() * AddBarcode() * BarcodeWidthChanged() * BarcodeHeightChanged() * UniformBarcodeChecked() * IncrementalBarcodeChecked() |
| Label | * Name:String * NumTextBlocks:Integer * NumImage: Integer * NumBarcode: Integer * BarWidth: Double * Barheight:Double * BarKey: String * BarType: String * Text: Array<string> * XAlign: Array<Integer> * YAlign: Array<Integer> * TextFont:Array<Double> * TextFontFamily:Array<String> * TextLeftMargin:Array<Double> * TextTopMargin: Array<Double> * ImageBitmap:Array<ByteArray> * ImageWidth:Array<Double> * ImageHeight:Array<Double> * ImageFileName:Array<string> * ImageStretchAlign:Array<integer> * ImageLeftMargin:Array<string> * ImageTopMargin:Array<Double> * BarMode:Bool |  |
| Screen1\_HomePage |  | * TextileTemplateClicked() * FoodTemplateClicked() * FootwearTemplateClicked() * EduTemplateClicked() |
| Print |  | * RegisterForPrinting() * UnregisterForPrinting() * PrintTaskRequested() * CreatePrintReviewPage() * GetPrintPreview() * AddPrintPages() * PrintDetailedOption() |

**2.1.3 State Diagram**

Screen1\_HomePage- Start State

Save\_Label- Final State

Print\_Label- Final State

The state diagram is representing the process of creating, saving and printing of labels.

# Transition Table

|  |  |  |
| --- | --- | --- |
| Initial State | Action | Next State |
| Screen1\_HomePage | Click on create label button. | Screen2\_CreateLabel |
| Screen2\_CreateLabel | Click on save button | Save Labels |
| Save Labels | Give print command | Print Label |
| Screen1\_HomePage | Click on Lists | Lists\_Label |
| Lists\_Label | Choose any label and give print command | Print Label |

**3. Detailed Description of Components**

**3.1 Components according to Class Diagram**

There are 4 classes in the class diagram made, so there are 4 components.

**3.1.1 Sreen2\_CreateLabel**

This is the screen 2 of the software where the user will be working on the design of their label after choosing the template in screen 1. In this the user will customise the label with the various methods like adding text rows and columns, adding image, adding a barcode and editing their properties.

The variables in the this component are SelectedIndex:Integer, validState:Bool ,textMovedValid:Bool, barMovedValid:Bool, allFonts:Array<string>,moveableElement: UIElement,Pointer: PointerPoint.

There are various functions listed in the above table which belong to this component.

**3.1.2 Label**

The instance of this class will hold all the attributes of a label that we make using this software. All the information from this class will be stored in the database as it is given.

The variables in this component are NumTextBlocks:IntegerNumImage:Integer,NumBarcode:Integer,BarWidth:Double,BarHeight:Double,BarKey:String,BarType:String,Text:Array<string>,XAlign:Array<Integer>YAlign:Array<Integer>,TextFont:Array<Double>,TextFontFamily:Array<String>,TextLeftMargin:Array<Double>,TextTopMargin:Array<Double>,ImageBitmap:Array<ByteArray>,ImageWidth:Array<Double>,ImageHeight:Array<Double>,ImageFileName:Array<string>,ImageStretchAlign:Array<integer>,ImageLeftMargin:Array<string>,ImageTopMargin:Array<Double>,BarMode:Bool.

**3.1.3 Screen1\_Homepage**

This is the first screen of the software and the user will decide what template they will choose to proceed for designing the label. It has functions like TextileTemplateClicked(),FoodTemplateClicked(),FootwearTemplateClicked(),EduTemplateClicked().

**3.1.4 Print**

The software will ask the user for all the settings to print to the user like how many pages to print, the printer to choose,etc. After this page the user will click print and the labels and barcodes will get printed. It has functions like RegisterForPrinting()

UnregisterForPrinting(),PrintTaskRequested(),CreatePrintReviewPage(),GetPrintPreview(),AddPrint,Pages(),PrintDetailedOption().

**4. Design Decision & Trade-offs**

There were many design decisions in the process of designing the UI. The major task was to keep all the settings on the same page of our activity. Same page will give users of mobile a ease of working with a preview above. However, there was no chance of compacting the properties required in the process of creating a label. Therefore we decided to create our own way to show settings. We change the visibility of the UI components, only the required component’s visibility is on and rest are off. Hence we managed to set all the components in the same activity which would become customer friendly User Interface.

We decided to implement Barcode with Uniform Barcode mode and Incremental Barcode mode. Barcode with incremental mode requires a base key and further barcode will be in incremented form in a page. But the idea of incremental barcode is only applicable for a single page print out. Multi page will require addition of multi pages which is for now not applicable through our design process. However, users can use incremental mode for a single page and for next page users can create another labels.

**5. Pseudo Code**

using ZXing; //Barcode Library

namespace Label\_v1

{

class CreateLabel

{

private void CreateTextBlock()

{

var textblock = new TextBlock();

textblock.text = “Text Line” + textlineNumber.toString();

textblock.Color = Black;

textblock.Font = 20;

PreviewGrid.children.add(textblock); //Adding TextBlock to Grid

}

private void CreateImage()

{

var Image = new Image();

Image.Source = “/Assets/imageFIle.png”

Image.Width = 70; //pixels

Image.Height = 70;

Image.Stretch = Stretch.Uniform;

PreviewGrid.children.add(Image);//Adding Image to Grid

}

private void CreateBarcode()

{

if(BarcodeEnable == true)

{

var barcode = new Image();

barcode.source = ConvertStringToBarcode(barkey);

//barkey is string input by user.

barcode.Width = 70;

barcode.Height = 70;

barcode.Stretch = Stretch.Uniform;

PreviewGrid.children.add(barcode);//Adding barcode to grid;

}

}

private void deleteText(UIElement textblock)

{

PreviewGrid.Children.Remove(textblock);

}

private void deleteImage(UIElement Image)

{

PreviewGrid.Children.Remove(Image);

}

private void deleteBarcode(UIElement barcode)

{

PreviewGrid.Children.Remove(barcode);

BarcodeEnable = false;

}

private void UpdateTextBlock(UIElement textblock, int Index)

{

double newWidth=100, newHeigth = 90;

textblock.Text = “New Text”;

textblock.Width = newWidth; //by user

textblock.Height = newHeight; //by user

PreviewGrid.Children.Insert(Index, textblock);

//Updating textblock again to Grid

}

private void UpdateImage(UIElement image, int Index)

{

double newWidth = 200, newHeight = 70;

WritableBitmap newSource;

image.Width = newWidth; //by user

image.Height = newHeight; //by user

image.source = newSource // by user;

image.Stretch = Stretch.UniformToFill;

PreviewGrid.Children.Insert(Index, image);

//Updating image again to Grid

}

private void UpdateBarcode(UIElement barcode, int Index)

{

if(BarcodeEnabled == true)

{

double newWidth =130, newHeight = 60;

string newBarkey = “12223”;

barcode.Width = newWidth;

barcode.Height = newHeight;

barcode.Source = ConvertStringToBarkey(newBarkey);

barcode.Stretch = Stretch.Fill;

PreviewGrid.Children.Insert(Index, barcode);

//Updating barcode Image to Grid

}

}

}

class Label

{

public long numTextblock, numImage, previewGridWidth, previewGridHeight;

public int numRows, numColumns, numPage;

public bool BarcodeEnabled = false;

public ObservableCollection<string> text;

public ObservableCollection<double> textFont;

public ObservableCollection<double> textwidth;

public ObservableCollection<double> textHeight;

public ObservableCollection<WriteableBitmap> Image;

public ObservableCollection<double> imageWidth;

public ObservableCollection<double> imageHeight;

public ObservableCollection<int> imageStretch;

public ObservableCollection<double> barcodeHeight;

public ObservableCollection<double> barcodeWidth;

public ObservableCollection<WriteableBitmap> Barcode;

public ObservableCollection<int> barcodeStretch;

}

Class Database

{

SqliteConnection conn;

public void LoadDatabase()

{

conn = new SqliteConnection(“LabelDatabase.db”);

string Query = “

CREATE TABLE ‘Label’ (

‘Label\_ID’ varchar(50) NOT NULL ,

`Label\_Name’ varchar(20) NOT NULL,

‘Rows’ int(15) NOT NULL,

`Columns’ int(10) NOT NULL,

`Num\_Txt\_Blocks` int(100) NULL,

‘Num\_Img’ int(50) NOT NULL,

‘Barcode\_Valid’ tinyint(1) NOT NULL,

PRIMARY KEY (`Label\_ID`)

); ”;

conn.Prepare(Query);

conn.Step(); //Execute Query

Query = “CREATE TABLE ‘Barcode’ (

‘Label\_ID’ varchar(50) NOT NULL,

‘Bar\_Type’ varchar(20) NOT NULL,

‘Bar\_Key’ varchar(100) NOT NULL,

‘Bar\_Width’ real(3) NOT NULL,

‘Bar\_Height real(3) NOT NULL,

FOREIGN KEY(‘Label\_ID’)

PRIMARY KEY(‘Label\_ID’)

); ”;

conn.Prepare(Query);

conn.Step(); //Execute Query

Query = “CREATE TABLE ‘Text’ (

‘Label\_ID’ varchar(50) NOT NULL,

‘Txt\_ID’ varchar(4) NOT NULL,

‘Array\_Txt’ varchar(200) NOT NULL,

‘ArrayAlign\_X’ int(2) NOT NULL,

‘ArrayAlign\_Y’ int(2) NOT NULL,

‘Font\_Size’ real(3) NOT NULL,

‘Font\_Type’ varchar(20) NOT NULL,

‘Left\_Margin’ real(3) NOT NULL,

‘Top\_Margin’ real(3) NOT NULL,

FOREIGN KEY(‘Label\_ID’)

PRIMARY KEY(‘Txt\_ID’)

);”;

conn.Prepare(Query);

conn.step();

Query = “CREATE TABLE ‘Image’(

‘Lable\_ID’ varchar(50) NOT NULL,

‘Img\_ID’ varchar(4) NOT NULL,

‘Img\_Bitmap’ blob NOT NULL,

‘Img\_Width’ real(3) NOT NULL,

‘Img\_Height’ real(3) NOT NULL,

‘Img\_File’ varchar(20) NOT NULL,

‘Stretch\_Align’ real(3) NOT NULL,

‘Left\_Margin’ real(3) NOT NULL

‘Right\_Margin’ real(3) NOT NULL

FOREIGN KEY(‘Label\_ID’)

PRIMARY KEY(‘Img\_ID’)

);”;

conn.Prepare(Query);

conn.step();// Execute Query

}

}

}