ANALYSIS OF POPULAR WEB FRAMEWORKS FOR FEATURES SUPPORTED IN SKIN, LAYOUT AND NAVIGATION AND DESIGN OF IMPROVED LAYOUT SUPPORT IN AN EXISTING FRAMEWORK

SUBMITTED

BY

SUVRA NANDI (MUKHOPADHYAY)

EXAMINATION ROLL NUMBER:

REGISTRATION NUMBER: 117100 of 2011-12

CLASS ROLL NUMBER: 001111003004

THIS THESIS SUBMITTED TO

THE FACULTY OF ENGINEERING & TECHNOLOGY OF JADAVPUR UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING

IN

SOFTWARE ENGINEERING

UNDER THE SUPERVISION

OF

Mr. DIPTENDU DUTTA

DEPARTMENT OF INFORMATION TECHNOLOGY

JADAVPUR UNIVERSITY

2015

Department of Information Technology

Faculty of Engineering & Technology

JADAVPUR UNIVERSITY

Certificate of Submission

I hereby recommend that the thesis, entitled "Analysis of popular web frameworks for features supported in skin, layout and navigation and design of improved layout support in an existing framework", prepared by Suvra Nandi (Mukhopadhyay) (Registration. No.117100 of 2011-12) under my supervision, be accepted in partial fulfillment of the requirement for the degree of Master of Engineering in Software Engineering from the Department of Information Technology under Jadavpur University.

____

(MR. DIPTENDU DUTTA)

Department of Information Technology

Jadavpur University

Countersigned by:

____

(Head of the Department)

Information Technology

Jadavpur University

____

(Dean)

Jadavpur University

Faculty of Engineering & Technology

JADAVPUR UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY

FACULTY OF ENGINEERING & TECHNOLOGY

CERTIFICATE OF APPROVAL

The thesis at instance is hereby approved as a creditable study of an Engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis for the purpose for which it is submitted.

______________________________ ______________________________

(Signature of the Examiner) (Signature of the Supervisor)

Declaration of Compliance of Academic Ethics

I hereby declare that this thesis contains literature survey work by me, as a part of my Master of Engineering in Software Engineering studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

____

(Signature with Date)

Name : **SUVRA NANDI (MUKHOPADHYAY)**

Roll Number : **001111003004**

Thesis Title : **Analysis of popular web frameworks for features supported in skin, layout and navigation and design of improved layout support in an existing framework**

Acknowledgement

I would like to thank a lot of people who gave me unending support and inspiration from the beginning and without their help this thesis and project would not have been completed.

First and foremost, I would like to thank my guide Mr. DIPTENDU DUTTA, whose suggestions, guidance and encouragement have helped me immensely in understanding the subject. I would like to thank my classmate and colleague Mr. Joydip Das for his support and cooperation.

I would also like to thank all of my classmates/colleague for the constant support and help they provided all the time.

Location:

Date:

Regards,

____

SUVRA NANDI (MUKHOPADHYAY)

M.E. in Software Engineering

Class Roll No: 001111003004

Exam Roll No:

Registration No: 117100 of 2011-12

ABSTRACT

The goal of a framework is to allow designers and developers to focus on building the unique features for their project, rather than re-inventing the wheel by coding common, familiar features found across many websites and web applications. An application framework is basically a standard programming structure that already pre-defines those repetitive stuffs as well as any other necessary functions so that developer easily and effectively writes their web applications in less time. Usage of web framework have also advantage in different other aspects like efficiency, security, integration etc.

Whenever we want to build web based visualization application / GUI we want to search for an efficient and helpful framework to use to ease our development work. While searching a perfect fit framework according to developer requirement criteria we need to study explore and hands on with different frameworks to understand different features and identify strength and weakness of different available features.

This thesis work tries to identify the features available in a set of popular web frameworks (via analysis and hands-on implementation) with respect to style, layout and navigation features and identify their strengths and weaknesses and suggest their usability in specific scenarios. Later we identify a framework, explore its features with respect to skin, layout, and navigation and suggest an improved design for laying out components addressing common layout challenges and basic requirements of users in a single package.

Table of Contents

1. Introduction 1
1.1. Motivation and Focus of the Thesis 2
1.2. Related Work 3
1.2.1 GUI Design Frameworks 3
1.2.2 Layout Managers 4
1.2.3 Difference between Imperative and Declarative Approach to UI Design 7
1.2.4 Declarative Approaches to UI Design – UIML 8
1.2.5 Layout Resizing 10
1.2.6 Web Page Navigation – Useful Strategies 13
1.2.7 Existing Navigation Approaches 15
1.2.8 The Basic Skinning Strategies 22
1.2.9 Existing Skinning Approaches 23
1.3. Contributions of the Thesis 26
1.4. Organization of the Thesis 27
2. Study on Different Web Frameworks via Implementation of a GUI Application 29
2.1 Identify Framework for Study and Hands-On 30
2.2 Identify Target GUI Application for Hands-On using Different Web Frameworks 31
2.3 Study Outcome of Swing Framework 31
2.3.1 Layout Support in SWING 32
2.3.2 Navigation Support in SWING 39
2.3.3 Theme Support in SWING 40
2.3.4 Sample Screens Implemented in SWING 41
2.3.5 Evaluation through Implementation Experience 47
2.4 Study Outcome of SWT Framework 49
2.4.1 Layout Support in SWT 49
2.4.2 Navigation Support in SWT 54
2.4.3 Theme Support in SWT 55
2.4.4 Sample Screens Implemented in SWT 55
2.4.5 Evaluation through Implementation Experience 59

2.5 Study outcome of JSF Framework 61
2.5.1 Layout Support in JSF 61
2.5.2 Navigation Support in JSF 77
2.5.3 Theme Support in JSF 81
2.5.4 Sample Screens Implemented in JSF 86
2.5.5 Evaluation through Implementation Experience 91
2.6 Study outcome of Windows Store Apps Framework 93
2.6.1 Layout Support in Windows Store Apps 93
2.6.2 Navigation Support in Windows Store Apps 98
2.6.3 Theme Support in Windows Store Apps 99
2.6.4 Sample Screens Implemented in Windows Store Apps 100
2.6.5 Evaluation through Implementation Experience 105
2.7 Study outcome of SiteMesh Framework 107
2.7.1 Layout Support in SiteMesh 107
2.7.2 Navigation Support in SiteMesh 112
2.7.3 Theme Support in SiteMesh 112
2.7.4 Sample Screens Implemented in SiteMesh 114
2.7.5 Evaluation through Implementation Experience 119
2.8 Study outcome of QT QML Framework 120
2.8.1 Layout Support in QT QML 120
2.8.2 Navigation Support in QT QML 127
2.8.3 Theme Support in QT QML 130
2.8.4 Sample Screens Implemented in QT QML 132
2.8.5 Evaluation through Implementation Experience 136
2.9 Study outcome of CSS3 FLEXBOX Framework 138
2.9.1 Layout Support in CSS3 FLEXBOX 138
2.9.2 Navigation Support in CSS3 FLEXBOX 140
2.9.3 Theme Support in CSS3 FLEXBOX 141
2.9.4 Sample Screens Implemented in CSS3 FLEXBOX 143
2.9.5 Evaluation through Implementation Experience 147
3. Framework Comparison 149
4. Layout Design Patterns – Step by Step Approach 156

4.1 Page View Patterns 158
4.1.1 Grid Based Pages 158
4.1.2 Point Based Pages 159
4.2 Patterns in Application Symmetricity 160
4.2.1 Template Based Page Structure 160
4.3 Patterns in Positioning Approach 163
4.3.1 Layout Managers or Class Based Layout 163
4.3.2 Anchor Based Layout 166
4.3.3 Nested Table based Static layout 168
4.3.4 DIV, CSS & JS based layout 170
4.3.5 Absolute Positioning 171
4.4 Patterns in Layout Re-rendering Approach 173
5. Understanding the LearnITy Framework 174
5.1 Basic Concepts of LearnITy Framework 175
5.2 Why use LearnITy Framework 176
5.3 Main Components of LearnITy Framework 176
5.3.1 Manifest.xml 177
5.3.2 Interfacerole.xml 177
5.3.3 Interface.xml 178
5.4 How to Use LearnITy Framework 180
5.5 Sample Screens Implemented in LearnITy Framework 182
6. Improved Design Implementation in the LearnITy Framework 184
6.1 LearnITy Framework Layout Design 185
6.1.1 Container Layout 185
6.1.2 Component Layout 195
6.2 LearnITy Framework XML Validation Feature 208
6.3 Comparison with Baseline Version and Advantages of Enhanced Design in LearnITy Framework 210
6.3.1 Baseline Version of LearnITy Framework 210
6.3.2 Improved Design in LearnITy Framework 211
6.4 Implementation Notes 218
6.4.1 Existing Structure of the Backend Code 218

6.4.2 Modified Structure of the Backend Code 221
7. Future Research Work 227
7.1 Implementation of Theme Based Styling 228
7.2 Implementation of Rule Based Navigation Support 228
7.3 Implementation of Responsive Layout with Component Resizing 228
8. References 229
Appendix 1: Live Implementation of the LearnITy Framework 236
Manifest.xml 237
Interfacerole.xml 237
Interface.xml for NewDesignPageCreation 238
Interface.xml for AccountSummaryCenterPanel 249
Manifest.xsd structure 256
Interfacerole.xsd structure 257
Interface.xsd structure 258
Appendix 2: Layout Rendering Approach 270
Static Layout 271
Liquid Layout 272
Adaptive Layout 273
Responsive Layout 274
Mostly Fluid 274
Column Drop 275
Layout Shifter 276
Tiny Tweaks 277
Off Canvas 277

List of Figures

Figure 1: Swing Implementation Screen - Current/Savings Account 41
Figure 2: Swing Implementation Screen - View Account Balance-Savings 42
Figure 3: Swing Implementation Screen - View Account Balance-Current........ 42
Figure 4: Swing Implementation Screen - Funds Transfer - Select Transaction Type........ 43
Figure 5: Swing Implementation Screen - Funds Transfer 43
Figure 6: Swing Implementation Screen - Funds Transfer – Confirm 44
Figure 7: Swing Implementation Screen - Transaction Success Page 44
Figure 8: Swing Implementation Screen - Account Summary after Fund Transfer 45
Figure 9: Swing Implementation Screen - Look and Feel Menu 45
Figure 10: Swing Implementation Screen - Default Metal LAF 46
Figure 11: Swing Implementation Screen - System LAF........ 46
Figure 12: Swing Implementation Screen - Motif LAF 47
Figure 13: SWT Implementation Screen - Current/Savings Account 55
Figure 14: SWT Implementation Screen - View Account Balance – Savings........ 56
Figure 15: SWT Implementation Screen - View Account Balance – Current 56
Figure 16: SWT Implementation Screen - Funds Transfer - Transaction Type 57
Figure 17: SWT Implementation Screen - Funds Transfer 57
Figure 18: SWT Implementation Screen - Funds Transfer – Confirm 58
Figure 19: SWT Implementation Screen - Transaction Success Page 58
Figure 20: SWT Implementation Screen - Account Summary after Funds Transfer 59
Figure 21: JSF Implementation Screen - Account Summary 86
Figure 22: JSF Implementation Screen - View Account Balance - Savings Account........ 87
Figure 23: JSF Implementation Screen - View Account Balance - Current Account 87
Figure 24: JSF Implementation Screen - Fund Transfer - Transaction Type 88
Figure 25: JSF Implementation Screen - Fund Transfer 88
Figure 26: JSF Implementation Screen - Fund Transfer – Confirm 89
Figure 27: JSF Implementation Screen - Fund Transfer after Back Navigation........ 89
Figure 28: JSF Implementation Screen - Transaction Success Page........ 90
Figure 29: JSF Implementation Screen - Page Navigation Demo from Help Link 90
Figure 30: Windows Store Apps Implementation Screen - Current/Savings Account 100
Figure 31: Windows Store Apps Implementation Screen - View Account Balance – Savings 101
Figure 32: Windows Store Apps Implementation Screen - View Account Balance – Current 101
Figure 33: Windows Store Apps Implementation Screen - Funds Transfer - Transaction Type 102
Figure 34: Windows Store Apps Implementation Screen - Funds Transfer 102
Figure 35: Windows Store Apps Implementation Screen - Funds Transfer – Confirm 103
Figure 36: Windows Store Apps Implementation Screen - Funds Transfer Success Page 103
Figure 37: Windows Store Apps Implementation Screen - Blank Help Page with "Back" Link - for showing Page Navigation 104
Figure 38: Windows Store Apps Implementation Screen - Blank Help page in Dark theme 104

Figure 39: Windows Store Apps Implementation Screen - Account Summary page in Dark Theme 105
Figure 40: SiteMesh Implementation Screen - Account Summary 114
Figure 41: SiteMesh Implementation Screen - View Account Balance – Savings 115
Figure 42: SiteMesh Implementation Screen - View Account Balance – Current 115
Figure 43: SiteMesh Implementation Screen - Fund Transfer - Tran Type 116
Figure 44: SiteMesh Implementation Screen - Fund Transfer 116
Figure 45: SiteMesh Implementation Screen - Fund Transfer – Confirm 117
Figure 46: SiteMesh Implementation Screen - Fund Transfer after Back Navigation 117
Figure 47: SiteMesh Implementation Screen - Transaction Success Page 118
Figure 48: SiteMesh Implementation Screen - Blank Help Page on click of "Help" link for Page Navigation Demo 118
Figure 49: Qt QML Implementation Screen - Account Summary 132
Figure 50: Qt QML Implementation Screen - View Account Balance – Savings 133
Figure 51: Qt QML Implementation Screen - View Account Balance – Current 133
Figure 52: Qt QML Implementation Screen - Fund Transfer - Transaction Type 134
Figure 53: Qt QML Implementation Screen - Fund Transfer 134
Figure 54: Qt QML Implementation Screen - Fund Transfer Confirm 135
Figure 55: Qt QML Implementation Screen - Fund Transfer - after navigating back through "Back" link 135
Figure 56: Qt QML Implementation Screen - Transaction Success Page 136
Figure 57: CSS3 FlexBox Implementation Screen - Account Summary 143
Figure 58: CSS3 FlexBox Implementation Screen - View Account Balance 144
Figure 59: CSS3 FlexBox Implementation Screen - Fund Transfer - Tran Type 144
Figure 60: CSS3 FlexBox Implementation Screen - Fund Transfer 145
Figure 61: CSS3 FlexBox Implementation Screen - Fund Transfer – Confirm 145
Figure 62: CSS3 FlexBox Implementation Screen - Fund Transfer after Back Navigation 146
Figure 63: CSS3 FlexBox Implementation Screen - Fund Transfer Success 146
Figure 64: Page View Patterns - Probable Grid Structure - 1 158
Figure 65: Page View Patterns - Probable Grid Structure – 2 158
Figure 66: Page View Patterns - Probable Grid Structure – 3 159
Figure 67: Page View Patterns - Point based page view - 1 159
Figure 68: Page View Patterns - Point based page view – 2 160
Figure 69: Template based Page Structure - One Column Page Layout 160
Figure 70: Template based Page Structure - Two Column Page Layout 161
Figure 71: Template based Page Structure - Three Column Page Layout 161
Figure 72: Template based Page Structure - Layout with Footer 161
Figure 73: LearnITy Framework Implementation Screen - Account Summary Screen 182
Figure 74: LearnITy Framework Implementation Screen - Select Account 182
Figure 75: LearnITy Framework Implementation Screen - View Account Balance - Account1 183
Figure 76: LearnITy Framework Implementation Screen - View Account Balance - Account2 183
Figure 77: Enhanced Layout Design - HorizontalContainer, VerticalContainer, RightDiagonalContainer, LeftDiagonalContainer 186

Figure 78: Enhanced Layout Design - Recursive Container - Red Bordered Horizontal Container containing one verticalContainer and one Button 193
Figure 79: Enhanced Layout Design - Recursive Container: Red Bordered Horizontal Container containing one horizontalContainer and one Button 193
Figure 80: Enhanced Layout Design - Recursive Container - Black Bordered Horizontal Container, inside Red Bordered Horizontal Container........ 194
Figure 81: Enhanced Layout Design - Recursive Container - Black Bordered Vertical Container, inside Red Bordered Horizontal Container........ 194
Figure 82: Enhanced Layout Design - Recursive Container - Black Bordered LeftDiagonalContainer, inside Red Bordered Horizontal Container........ 194
Figure 83: Enhanced Layout Design - Recursive Container - Black Bordered RightDiagonalContainer, inside Red Bordered Horizontal Container 195
Figure 84: Enhanced Layout Design - Recursive Container - Red Bordered Horizontal Container containing one RightDiagonalContainer and one Button 195
Figure 85: Enhanced Layout Design - Recursive Container - Red Bordered Horizontal Container containing one LeftDiagonalContainer and one Button 195
Figure 86: Enhanced Layout Design - Border Layout - Buttons Positioned at North West, North Center, West Center, Center 196
Figure 87: Enhanced Layout Design - Border Layout - Buttons Positioned at South West, South Center, West Center, and Center 197
Figure 88: Enhanced Layout Design - Border Layout - Buttons Positioned at North East, North Center, East Center, and Center 198
Figure 89: Enhanced Layout Design - Border Layout - Buttons Positioned at South East, South Center, East Center, and Center 198
Figure 90: Enhanced Layout Design - Border Layout - North, South, East and West Bars placed in page using Border Layout 199
Figure 91: Enhanced Layout Design - Anchor Layout - Placing Block to the Right of Fixed Black Block, with indentation “\\n\\t”........ 203
Figure 92: Enhanced Layout Design - Anchor Layout - Placing Block to the Right of Fixed Black Block, with indentation “\\t” 204
Figure 93: Enhanced Layout Design - Anchor Layout - Placing Block to the Right of Fixed Black Block, with indentation “\\p\\n” 205
Figure 94: Enhanced Layout Design - Anchor Layout - Demo of different anchor positioned blocks 206
Figure 95: XSD Validation - Uploading Manifest.xml 209
Figure 96: XSD Validation - Manifest.xml uploaded successfully 209
Figure 97: XSD Validation - XML validation through XSD failed for Manifest.xml 210
Figure 98: Bank screen implemented using enhanced layout design of LearnITy Framework 255
Figure 99: eFiling screen implemented using enhanced layout design of LearnITy Framework........ 256
Figure 100: Layout Rendering Approach - Static Page Layout 272
Figure 101: Layout Rendering Approach - Liquid Page Layout 273
Figure 102: Layout Rendering Approach - Adaptive Page Layout 274
Figure 103: Layout Rendering Approach - Responsive Design - Mostly Fluid Layout........ 275

Figure 104: Layout Rendering Approach - Responsive Design - Column Drop Layout 275
Figure 105: Layout Rendering Approach - Responsive Design - Layout Shifter 276
Figure 106: Layout Rendering Approach - Responsive Design - Tiny Tweaks Layout 277
Figure 107: Layout Rendering Approach - Responsive Design - Off Canvas 277

1. Introduction

1.1.Motivation and Focus of the Thesis

A web application framework is a type of framework or foundation, specifically designed to help developers build web applications. These frameworks typically provide core functionality common to most web applications, such as user session management, data persistence, and template creating systems. By using an appropriate framework, a developer can often save a significant amount of time building a web site. The framework aims to alleviate the overhead associated with common activities performed in web development, and they often promote code reuse.

Most GUI architects will agree that it is difficult to design highly customizable user interfaces. Let's consider a scenario where the Application Users have multiple roles (Supervisor, Administrator, Order Representative, Customer Service Representative, and Bookkeeper). Developers are tasked with designing User Interfaces for their application and each role requires different subset of data to be displayed in a different layout.

If we plan to develop a new set of interfaces for each role, here are few reasons which make it not a good approach:

- If the customers define a new role, then developers need to build a new set of interfaces.
- If the data model changes, developers are required to update at least some set of user interfaces to display the new data elements.
- If permissions associated with a role changes, developers will have to modify code to display/hide data elements for a user.

Architecting a solution that solves for customizable UI with different role support would include inclusion of a framework that caters to common customization needs.

In general, the basic GUI frameworks can be categorized into the following types:

- Content management or data management frameworks: These frameworks enable developers to customize the data/content displayed in the UI.
- Execution/Controller framework: This framework essentially maps to the Controller and related component functionality in an MVC model. It is the glue between Presentation tier and Business Logic tier of an application. This is a non-visual GUI framework.
- Flow framework: This framework enables developers to customize navigation/workflow (it can be interpreted as screen/task flow) for an application. This may include inter-screen navigation, dynamically choosing the next screen to display.

- Layout framework: This is a pure visual framework that enables design of reusable User Interfaces. This framework enables a visual designer to choose layouts for content to be displayed in a container.

So we are studying a handful of frameworks, identify its support in style, layout and navigation perspective and finally use a proprietary framework named LearnITy Framework (Reference[40]) where we find a high level abstraction provided to developer with respect to style, layout and navigation. We also suggest improvements on existing features of layout or component positioning on it.

1.2.Related Work

As per the recent study and investigation on existing framework layout designs and ongoing initiatives, we found relevance to below areas while working on this thesis:

1.2.1 GUI Design Frameworks

Existing GUI design frameworks or widget toolkits can be categorized in multiple ways. Each of these toolkits provides its own layout, navigation and theme supports in various ways with pros and cons associated to each approach.

- First way of categorization is segregating frameworks as OS Dependent Toolkits & Cross Platform Toolkits. (Reference[17])

  OS Dependent Toolkits are of mainly four types – On Amiga, On Macintosh, On Microsoft Windows, and On Unix – under the X Windows system.

  Cross Platform Toolkits are many, based on no. of different languages like C, C++, OpenGL, Flash, XML, JavaScript, SVG, Java, Object Pascal, Ada, Objective-C, Eiffel, Ruby etc.

- Second way of categorization can be Monolithic (Desktop) environments, Client Server Systems, Sever Side Technologies, and Client Side Frameworks etc.

  Monolithic environments may encounter Native Windows systems on Visual C++, C#, XAML etc. (References[18-19]), Mac OS / OS X based Cocoa Framework – Auto Layout (Refererence[20]), Cross Platform frameworks include C++ based QT QML, Java Swing SWT etc.

  Client Server systems encounter C/C++ based X Windows Xt, Motif frameworks (Reference[21])

  Server Side Technologies encounter Java View Technologies and Frameworks (Reference[22]) which consist of different toolkits like Servlet API, JSP, Apache Struts, Apache Tapestry, Apache Wicket, JSF and Facelets, SiteMesh etc.

Client Side GUI Frameworks encounter JavaScript based JQuery UI (Reference[23]), Sencha (Reference[24]) and Silverlight (Reference[25])

- Rich Internet Application (RIA) Development frameworks can also be categorized as Java-Script based (JQuery, Sencha, Prototype, X-Library etc.), Non Java-Script based (Adobe Flex, Java FX, Silverlight, OpenLaszlo etc.) & Multi-Device frameworks (Application Craft, QT, AppMobi etc.) GUI Frameworks

1.2.2 Layout Managers

Layout Managers are software components used in widget toolkits which have the ability to lay out graphical control elements by their relative positions without using distance units. It is often more natural to define component layouts in this manner then to define their positions in pixels or common distance units. So a number of popular widget toolkits include this ability by default.

Toolkits that provide this functionality can be classified into two groups:

- Layout behaviour is coded in Special Graphic Containers. E.g. XUL, .NET Framework widget toolkit
- Layout behaviour is coded in Layout Managers that can be applied to graphical container. E.g. Swing widget toolkit in Java

In XUL, the vbox container is used to stack components on top of each other. Below piece of code creates 3 buttons stacked on top of each other in a vertical box:

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<window id="vbox example" title="Example"
xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

<vbox>
<button id="yes" label="Yes"/>
<button id="no" label="No"/>
<button id="maybe" label="Maybe"/>
</vbox>
</window>
```

(Reference[27])

In XAML, the DockPanel container lays out children components on top of each other in a vertical box. Below code shows four text blocks on top of each other:

```
<Page xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
WindowTitle="myDock Panel">
<DockPanel>
<TextBlock DockPanel.Dock="Top">Top 1</TextBlock>
<TextBlock DockPanel.Dock="Top">Top 2</TextBlock>
<TextBlock DockPanel.Dock="Top">Top 3</TextBlock>
<TextBlock DockPanel.Dock="Top">Top 4</TextBlock>
</DockPanel>
</Page>
```

(Reference[28])

In Java, the FlowLayout layout manager arranges components in a directional flow, much like lines of text in a paragraph. It arranges components horizontally until no more components fit on the same line, and then places them on another line. Several AWT, Swing classes provide layout managers for general use.

Below code shows five buttons alongside each other on the same link:

```
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.FlowLayout;
import java.awt.Container;
public class Example {
private JFrame frame;
public Example() {
frame = new JFrame("FlowLayout Demo");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setLayout(new FlowLayout());
frame.add((new JButton("Button 1")));
frame.add((new JButton("Button 2")));
frame.add((new JButton("Button 3")));
frame.add((new JButton("Long-Named Button 4")));
frame.add((new JButton("5")));
frame.pack();
frame.setVisible(true);
}
public static void main(String[] args) {
Example ex = new Example();
}
}
```

(Reference[1])

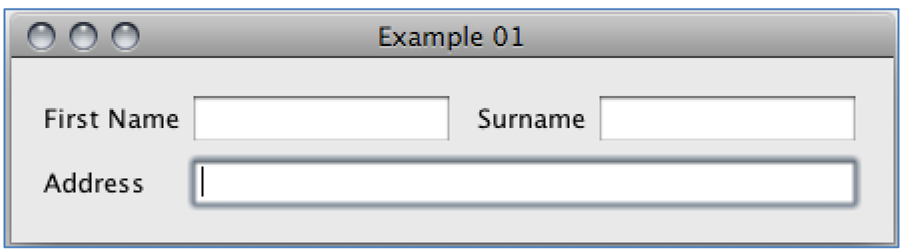
Java Layout Manager MiGLayout is another easy to use and powerful Java Swing, JavaFX 2 and SWT layout manager. For Java developers writing GUI layouts by hand that wants simplicity, power and automatic per platform fidelity, that are dissatisfied with the current layout managers in Swing, JavaFX 2 and SWT, MigLayout solves layout problems. User Interface created with MiGLayout is easy to maintain. It is using String or API type-checked constraints to format the layout. MiGLayout can produce flowing, grid based, absolute, grouped and docking layouts. It can also be used for other toolkits by writing the glue code.

(Reference[29])

Below is a code snippet using MigLayout layout manager:

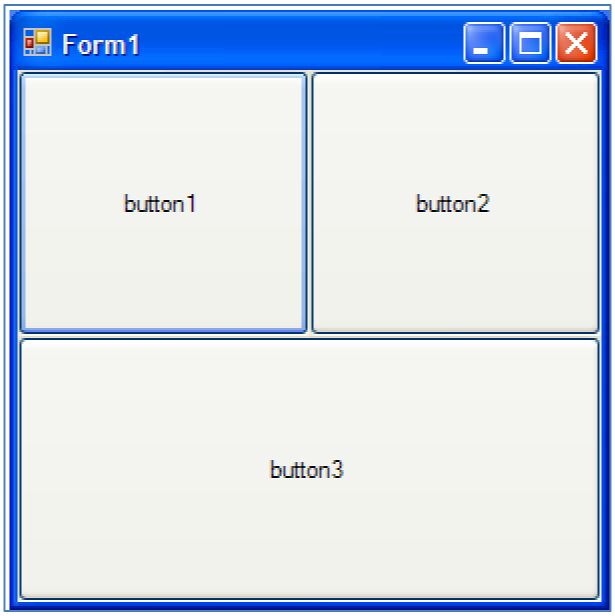
```
JPanel panel = new JPanel(new MigLayout());

panel.add(firstNameLabel);
panel.add(firstNameTextField);
panel.add(lastNameLabel,       "gap unrelated");
panel.add(lastNameTextField,   "wrap");
panel.add(addressLabel);
panel.add(addressTextField,    "span, grow");
```

It produces the below panel. All the gaps (white spaces) are added automatically. The gaps will be correct for the platform it is run. Even the white space around the component or border is automatic.



The Auckland Layout Model (ALM) is another technique for specifying 2D layout as it is used for arranging the controls in a GUI. The model allows the specification of constraints based on linear algebra, and an optimal layout is calculated using linear programming. Linear equalities and inequalities can be specified on horizontal and vertical tabstops, which are virtual lines that form a grid to which all the elements of the GUI are aligned. The described technique supports the dynamic adaptation of a layout to changes in the environment, e.g. window size, and offers more flexibility than other layout managers. (Reference[31])

Few examples of Linear Programming based ALM Layout manager implemented screen are depicted below:




```
LayoutSpec ls = new LayoutSpec();
XTab x1 = ls.AddXTab();
YTab y1 = ls.AddYTab();

ls.AddArea(ls.Left, ls.Top, x1, y1, button1);
ls.AddArea(x1, ls.Top, ls.Right, y1, button2);
ls.AddArea(ls.Left, y1, ls.Right, ls.Bottom, button3);

// give the two columns the same width and the two rows the same
height
ls.AddConstraint(new double[] { 2, -1 }, new Variable[] { x1, ls.Right },
OperatorType.EQ, 0);
ls.AddConstraint(new double[] { 2, -1 }, new Variable[] { y1, ls.Bottom },
OperatorType.EQ, 0);
```

This example shows how three buttons are arranged using three areas. The buttons have already been added to the window, but have not been arranged. Their location and size is determined by the ALM specification. No matter how we resize the window, the two columns will always have the same width, and the two rows will always have the same height due to the two linear constraints. On a Pentium 4 with 3.4 GHz layout calculation takes about 0.3 milliseconds. Interestingly, on a Pentium M with 1.6 GHz the calculation takes also about 0.3 milliseconds. On a Pentium 3 with 788 MHz the calculation takes about 0.7 milliseconds.

1.2.3 Difference between Imperative and Declarative Approach to UI Design

With Imperative programming developers need to tell the compiler what's needs to be done, step by step. Below is an example of imperative coding:

```
List<int> collection = new List<int> { 1, 2, 3, 4, 5 };
List<int> results = new List<int>();
foreach(var num in collection)
{
if (num % 2 != 0)
results.Add(num);  }
```

In the above code snippet below steps are mentioned:

- Create a result collection
- Step through each number in the collection
- Check the number, if it's odd, add it to the results

With Declarative programming, on the other hand, developers write code that describes what they want, but not necessarily how to get it. Below is the example of related declarative code:

```
var results = collection.Where( num => num % 2 != 0);
```

Here, developer is saying "Give us everything where it's odd", not "Step through the collection. Check this item, if it's odd; add it to a result collection."

Imperative approach being more specific language based, developers need to learn the language before using the frameworks. E.g. Swing, SWT etc. are imperative frameworks, where underlying logic needs to be written in backend Java classes. Hence for using those developers first need to be expert in Java language coding.

On the contrary, Declarative approaches are mainly XML based and doesn't require any specific programming language usage knowledge. Hence this approach is more easy to use for developers where all the methodical details are taken care by underlying code. So declarative programming was considered for improvement in Layout, Navigation and Theme designs because that may result in simplified framework usage with powerful features availability, which is a pain area in current UI design approaches.

1.2.4 Declarative Approaches to UI Design – UIML

(References[11-14])

Most devices with visual capabilities can display a graphical user interface (GUI), which was built using a high level tool. Modern GUI environments enable the user to communicate efficiently and intuitively with an application. The complexity in communication, however, remains and has been transferred from the user to the programmer. Several GUI builder tools have been developed to relieve this burden and even automate the GUI creation process (e.g., Visual Studio). Novices and highly skilled programmers alike use such tools to increase productivity.

Unfortunately these tools, tailored for GUIs, fall short when confronted with the new breed of devices (e.g., Palm, WebTV, cell phones, etc.) now appearing in the marketplace. The underlying assumption is that the interface is generated for one platform and is presented on a fixed type of display (e.g., computer monitor). Some

Tools provide support for multiple platforms, but they require significant code modification when the interface is presented to a new type of device (e.g., palm screen). Even on the desktop platform, different screen resolutions violate this assumption. Developing cross-device user interfaces requires both programming and artistic skills, as well as, knowledge about the particular device and application domain. However, few people have the necessary skills and knowledge to build and deliver these interfaces. Most people master a single platform on a single device and rely on others to port their interfaces to other platforms or devices.

Information publishing and the Internet faced a similar problem. Journalist and writers, the people who generate the information, did not have the computer skills to distribute their work on the Internet. HTML, a simple markup language that can be used by non-experts, revolutionized electronic publishing by enabling non-professional programmers to distribute on the Internet information in electronic form. This dissertation attempts to enable nonprofessional programmers to build user interfaces by giving a theoretical foundation for device-independent user interfaces and embodying this foundation in a declarative language, the User Interface Markup Language (or UIML).

Rendering is the process of converting a UIML document into a form that can be displayed to the end user. Rendering can be accomplished in two ways:

- By compiling UIML into another language (e.g. Java) which allows to display and interact with the interfaces created by UIML. Compilation can be accomplished by any program written in traditional programming language.
- By interpreting UIML, meaning that a program reads UIML and makes call to an API that displays the user interface and allows interaction with the interface. Interpretation is the same process which any web browser uses when presented with an HTML document.

Many of these markup languages are dialects of XML and are dependent upon a pre-existing scripting language engine, usually a JavaScript engine, for rendering of controls and extra script ability. There are several User interface Markup Languages like QML, MXML, UIML, XUL, XAML, OpenLaszlo(LZX) etc.

1.2.5 Layout Resizing

(Reference[15])

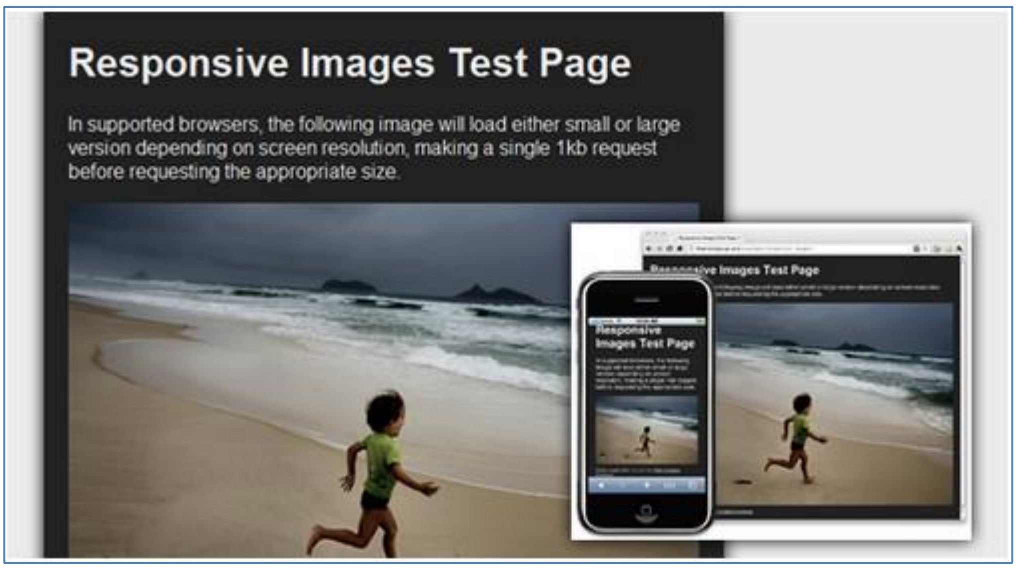
(References[31-32]) - Concept of Linear Programming

The most effective way of placing a widget on a page is resizing the component based on browser or page size, to accommodate the positioning of the component across all devices of varying screen size. Almost every new client these days wants a mobile version of their website. It's practically essential after all: one design for the BlackBerry, another for the iPhone, the iPad, netbook, Kindle — and all screen resolutions must be compatible, too.

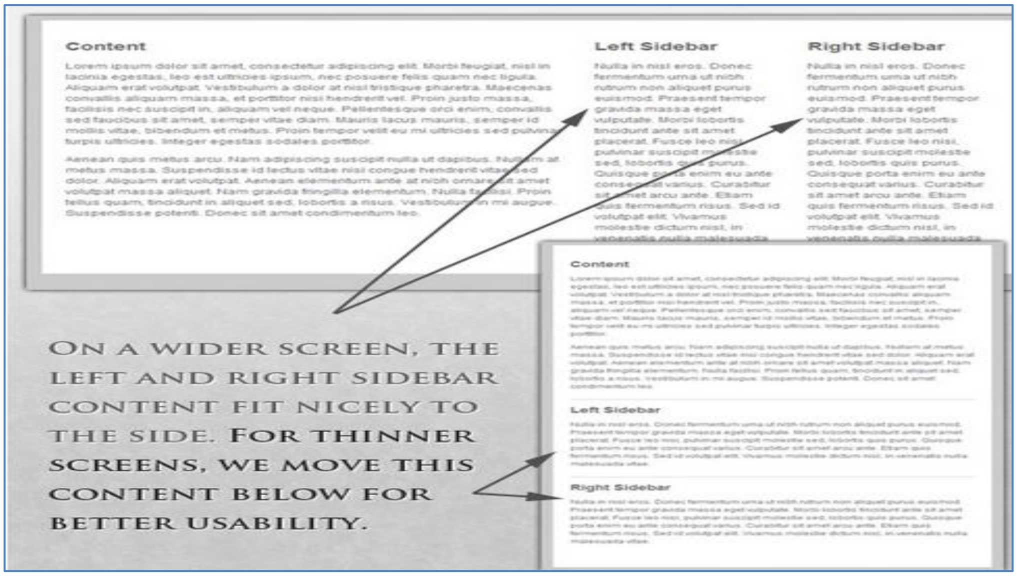
Responsive Web design is the approach that suggests that design and development should respond to the user's behavior and environment based on screen size, platform and orientation. The practice consists of a mix of flexible grids and layouts, images and an intelligent use of CSS media queries. As the user switches from their laptop to iPad, the website should automatically switch to accommodate for resolution, image size and scripting abilities. In other words, the website should have the technology to automatically respond to the user's preferences. This would eliminate the need for a different design and development phase for each new gadget on the market. Responsive Web design is not only about adjustable screen resolutions and automatically resizable images, but rather about a whole new way of thinking about design. Few of the common responsive layout pain areas are outlined below:

- Adjusting Screen Resolution

- Flexible Images



- Custom Layout Structure – with Media Queries & JavaScript



CSS3 Media Queries – Detecting Browser Width

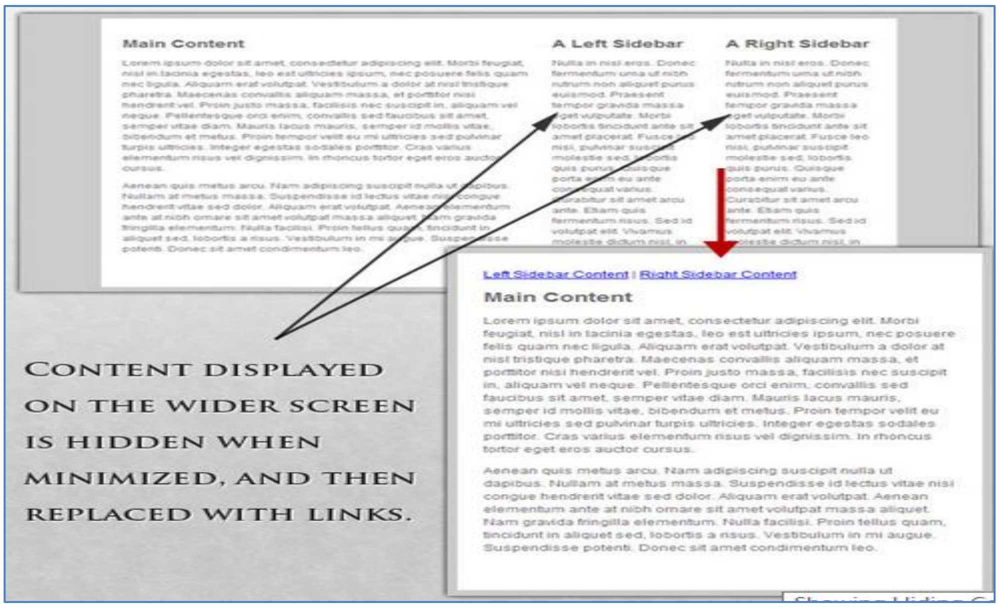
```
/* Smartphones (portrait and landscape) ----------- */
@media only screen
and (min-device-width : 320px)
and (max-device-width : 480px) {
/* Styles */
}

/* Smartphones (landscape) ----------- */
@media only screen
and (min-width : 321px) {
/* Styles */
}

/* Smartphones (portrait) ----------- */
@media only screen
and (max-width : 320px) {
/* Styles */
}
```

JavaScript – Detecting Browser Width for Devices Not Supporting CSS

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>

<script type="text/javascript">
$(document).ready(function(){
```

```
$(window).bind("resize", resizeWindow);
function resizeWindow(e){
var newWindowWidth = $(window).width();

// If width width is below 600px, switch to the mobile stylesheet
if(newWindowWidth < 600){       $("link[rel=stylesheet]").attr({href : "mobile.css"});       }
// Else if width is above 600px, switch to the large stylesheet       else if(newWindowWidth >
600){
$("link[rel=stylesheet]").attr({href : "style.css"});
}
}
});
</script>
```

- Showing or Hiding Content

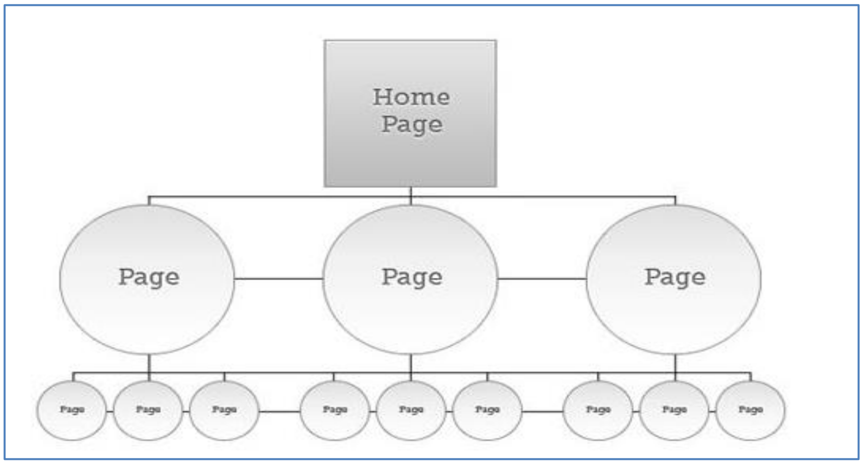


Responsive Web design and the techniques discussed above are not the final answer to the ever-changing mobile world. Responsive Web design is a mere concept that when implemented correctly can improve the user experience, but not completely solve it for every user, device and platform. We will need to constantly work with new devices, resolutions and technologies to continually improve the user experience as technology evolves in the coming years.

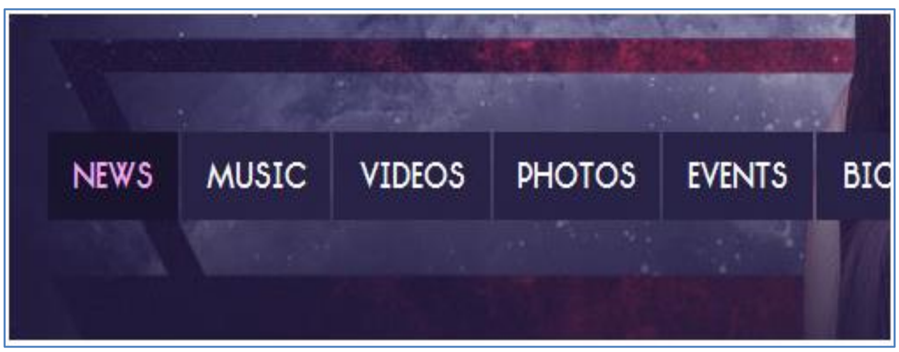
1.2.6 Web Page Navigation – Useful Strategies

The navigation menu on a website is like a road sign on a street or a level directory in a shopping mall. Like in real life, navigation in web design is very important and plays a major role in a website's usability as well as in user experience. There are few reusable tips available in order to make the navigation effective and usable:

- Navigation planning starts with information architecture. It is vital to sit down and brainstorm about a website's information architecture. As a result below is something which may come out of this architecture study:



- It is better to use simple, obvious and terms that are easy to figure out than to keep to industry-only terms for the navigation menu. Any link that takes users more than a second or two to figure out is probably unsuitable for use.



- Standardizing Navigation design by using the same navigation model in all pages. It is very important because without a consistent design, a user may actually think he is in another website.
- It is crucial to let the user know where he is at all times. It can be done by changing the link's background, color of the page name or turning the text bold in the navigation menu to make it different from others
- Using common web design conventions like
    - ✓ Placing logo at top left position of the page
    - ✓ Logo should link back to Homepage
    - ✓ Links should be underlined and differently colored than text

- ✓ Visited links should be differently colored than non-visited links
- ✓ Navigation should be consistent across pages
- ✓ Search facility should be provided
- ✓ Breadcrumb navigation should be present to help visualizing current positions
- ✓ If something looks like button, it should also act like button
- ✓ Form fields should be easily clickable
- ✓ Website should be accessible with and without www

Multiple reusable navigation components are available on CSS and jQuery (Reference[33])

Few Responsive Navigation approach and Pattern identification artifacts are also available at (References[34-35])

1.2.7 Existing Navigation Approaches

Based on study and implementation of UI screens through multiple web design frameworks, below categories are identified in Navigation design approach:

- Navigation approaches are mainly of seven types:
    - ✓ Programmatic DIV rendering
        - Normal DIV Render
        - JS Based DIV Render
    - ✓ Implicit Navigation
    - ✓ Managed Bean Navigation
        - Auto Navigation
        - Conditional Navigation
    - ✓ Rule Based Navigation
    - ✓ Class Based Navigation
    - ✓ Page Stack Navigation
    - ✓ State Based Navigation
- In Programmatic DIV rendering approach separate panels or DIV s are created which are to be shown on a common UI area. At a time only one DIV can be displayed. DIV transitions will result depending upon button click or any kind of other command events
- In normal DIV render sub category, DIVs can be shown or hidden by calling methods from code. This approach mostly works with Imperative Programming

Example from SWING framework:

There will be a common panel available. For navigating through various pages of the application, firstly old panel contents need to be removed, by calling the dispose command like this :

```
center.remove();
```

Then elements can be added to the panel as new contents as follows, layout defined :

```
center.add(contentGrid, BorderLayout.NORTH);
```

Finally, panel needs to be validated, repaint and packed for proper display.

```
center.validate();
center.repaint();
pack();
setVisible(true);
```

- In JS based DIV rendering category different DIVs are rendered or hidden from within JavaScript functions. DIVs are addressed either by Id, or by name. E.g. JQuery, CSS3 FlexBox Page Navigation.

Example from JQUERY:

```
<div id="tabs-22">
   <div class = "ex" ><input type="radio" name="sex" value="male">Transfer same bank other A/C<br></div>
   <div class = "ex" ><input type="radio" name="sex" value="female">Transfer other bank  A/C</div>
   <div class = "ex" ><button style="border-radius:4px" onclick="FetchData()">Continue</button></div>
</div>
```

where class ex represents

```
div.ex
{
width:700px;
padding:10px;
border:1px  solid blue;
margin:0px;
}
```

To hide the div : $("#tabs-22").hide();
To show the Div : $("#tabs-22").show();

These functions are called from a java script function.

Examples from CSS3 Flexbox:

```
<script type="text/javascript" language="javascript">
//document.getElementById("savingsHead").style.visibility = "hidden";
function showAccountSummary(){
document.getElementById("centerPanel").style.display="block";
document.getElementById("accountBalance").style.display="none";
document.getElementById("fundXferTranType").style.display="none";
document.getElementById("fundXfer").style.display="none";
document.getElementById("fundXferConfirm").style.display="none";
document.getElementById("fundXferSuccess").style.display="none";
}
</script>
```

- In Implicit Page Navigation approach developer needs to provide the Page Name in the Action attribute of the command widget (like Button, Link etc.) and the navigation strategy will search for the Page Name in current directory in order to load it on command action.

  E.g. JSF, Adobe Flex, Apache Tapestry, SiteMesh Page Navigation.

Examples from JSF:

```
<h:form>
<h3>Using JSF outcome</h3>
<h:commandButton action="page2" value="Page2" />
</h:form>
```

- Managed Bean Navigation strategy involves Managed Bean classes for page redirection. This approach is mainly applicable for JAVA based applications
- In Auto Navigation of Managed Bean Developer needs to define a method in managed bean to return a view name. The method name in turn gets called from the Page Widget's Action Attribute.
  E.g. JSF Page Navigation

Examples from JSF:

```
Page Code Snippet :
<h:form>
<h3>Using Managed Bean</h3>
<h:commandButton action="#{navigationController.moveToPage1}"
value="Page1" />
</h:form>
Managed Bean Code Snippet :
```

```
@ManagedBean(name = "navigationController", eager = true)
@RequestScoped
public class NavigationController implements Serializable {
public String moveToPage1(){
return "page1";
}
}
```

- Condition Navigation is also another Managed Bean navigation approach where conditioning is done within Managed Bean function based on parameters passed from the page.

  E.g. JSF Navigation

Examples from JSF:

```
Code Snippet from Page :
<h:form>
<h:commandLink action="#{navigationController.showPage}"
value="Page1">
<f:param name="pageId" value="1" />
</h:commandLink>
<h:commandLink action="#{navigationController.showPage}"
value="Page2">
<f:param name="pageId" value="2" />
</h:commandLink>
<h:commandLink action="#{navigationController.showPage}"
value="Home">
<f:param name="pageId" value="3" />
</h:commandLink>
</h:form>

Code Snippet from Managed Bean :
@ManagedBean(name = "navigationController", eager = true)
@RequestScoped
public class NavigationController implements Serializable {
//this managed property will read value from request parameter pageId
@ManagedProperty(value="#{param.pageId}")
private String pageId;
//condional navigation based on pageId
//if pageId is 1 show page1.xhtml,
//if pageId is 2 show page2.xhtml
//else show home.xhtml
public String showPage(){
if(pageId == null){
return "home";
```

```
}
if(pageId.equals("1")){
return "page1";
}else if(pageId.equals("2")){
return "page2";
}else{
return "home";
}
}
}
```

- In Rule-Based Navigation strategy Navigation Rules are set at configuration level. These rules get fired when command action is fired from the page. The rule contains characteristics like From Page, To Page, and From Action etc. This is a centralized location where page navigation rules are present for all the pages throughout the entire application.
  E.g. JSF Page Navigation

Examples from JSF:

```
<navigation-rule>
<from-view-id>home.xhtml</from-view-id>
<navigation-case>
<from-action>#{navigationController.processPage1}</from-action>
<from-outcome>page</from-outcome>
<to-view-id>page1.jsf</to-view-id>
</navigation-case>
<navigation-case>
<from-action>#{navigationController.processPage2}</from-action>
<from-outcome>page</from-outcome>
<to-view-id>page2.jsf</to-view-id>
</navigation-case>
</navigation-rule>
```

- In Class Based navigation certain navigation classes are defined within the framework, which provide page navigation functionalities like Go Back, Go Forward, State Management, Keyboard & Mouse Navigation etc.

  E.g. Windows Store App Navigation

Example from Windows Store App:

```
Code snippet of Page :
<HyperlinkButton x:Name="help" Content="Help" HorizontalAlignment="Left"
VerticalAlignment="Top"  FontSize="10" FontFamily="Segoe UI" FontWeight="Bold"
Click="Help_OnClick"/>

Help_OnClick function defined in Backend CPP file:
void TestApp::MainPage::Help_OnClick(Platform::Object^ sender,
Windows::UI::Xaml::RoutedEventArgs^ e)
{
if (this->Frame != nullptr)
{
this->Frame->Navigate(HelpPage::typeid);
}
}
This function will search for HelpPage.xaml in the current Directory and will render that
page. Here Frame class is having Navigation Functionality written in it.
```

- A Page Stack data structure consists of one or more pages, defined by Page Objects. A Page may be pushed onto the stack to place it on top of the stack, or popped to remove it from the stack.

  This stack-based navigation model makes it simple to provide hierarchical navigation within an application. When the application is required to provide a page of content, a new page can be pushed onto the stack; if the user requests to return to the previous page, the current page can be popped to reveal the previous page. Otherwise, another page can be pushed onto the stack to navigate deeper into the user interface hierarchy, and so on.

  E.g. QT QML Page Navigation

Examples from Qt Qml:

```
The QStackedWidget class provides a stack of widgets where only one widget is visible at a
time. QStackedWidget can be constructed and populated with a number of child widgets
("pages") like below:
QWidget *firstPageWidget = new QWidget;
QWidget *secondPageWidget = new QWidget;
QWidget *thirdPageWidget = new QWidget;
QStackedWidget *stackedWidget = new QStackedWidget;
stackedWidget->addWidget(firstPageWidget);
stackedWidget->addWidget(secondPageWidget);
stackedWidget->addWidget(thirdPageWidget);
QVBoxLayout *layout = new QVBoxLayout;
```

```
layout->addWidget(stackedWidget);
setLayout(layout);

QStackedWidget provides no intrinsic means for the user to switch page. This is typically done
through a QComboBox or a QListWidget that stores the titles of the QStackedWidget's pages.
For example:
QComboBox *pageComboBox = new QComboBox;
 pageComboBox->addItem(tr("Page 1"));
 pageComboBox->addItem(tr("Page 2"));
 pageComboBox->addItem(tr("Page 3"));
 connect(pageComboBox, SIGNAL(activated(int)),
        stackedWidget, SLOT(setCurrentIndex(int)));
```

- In State Based Navigation states are a set of property configurations defined in a State element. Different configurations are possible with this approach, for example:
  - ✓ Show some UI elements and hide others
  - ✓ Present different available actions to the user
  - ✓ Start, stop, or pause animations
  - ✓ Execute some script required in the new state
  - ✓ Change a property value for a particular item
  - ✓ Show a different view or screen

  E.g. QT QML Navigation

Example from QT QML:

```
states :[
State{
name: "accountSummary"
when: leftPanel.activeButton == "accountSummary"
PropertyChanges { target: centerPanelAccountSummary; visible:true; }
PropertyChanges { target: centerPanelViewAccountBalance; visible:false; }
PropertyChanges { target: centerPanelFundXfer; visible:false; }
},
State{
name: "viewAccountBalance"
when: leftPanel.activeButton == "viewAccountBalance"
PropertyChanges { target: centerPanelAccountSummary; visible:false; }
PropertyChanges { target: centerPanelViewAccountBalance; visible:true; }
PropertyChanges { target: centerPanelFundXfer; visible:false; }
```

```
},
State{
name: "fundXfer"
when: leftPanel.activeButton == "fundXfer"
PropertyChanges { target: centerPanelAccountSummary; visible:false; }
PropertyChanges { target: centerPanelViewAccountBalance; visible:false; }
PropertyChanges { target: centerPanelFundXfer; visible:true; }
} ]
```

This code can be placed in the main.qml file, where specified targets "centerPanelAccountSummary", "centerPanelViewAccountBalance", "centerPanelFundXfer" are different qml pages that gets called on fulfilling the "when" criteria.

1.2.8 The Basic Skinning Strategies

In computing, a skin is a custom graphical appearance preset package achieved by the use of a graphical user interface (GUI) that can be applied to specific computer software, operating system, and websites to suit the purpose, topic, or tastes of different users. A skin can completely change look and feel and navigation interface of a piece of application software or operating system. A skin may also be associated with themes, which usually only implies part changes and smaller differences, such as colors.

Software that is capable of having a skin applied is referred to as being skinnable, and the process of writing or applying such a skin is known as skinning. Applying a skin changes a piece of software's look and feel – some skins merely make the program more aesthetically pleasing, but others can rearrange elements of the interface, potentially making the program easier to use. Although often used simply as a synonym for skin, the term theme normally refers to less-complex customizations, such as a set of icons and matching colour scheme for an operating system.

The most popular skins are for instant messaging clients, media center, and media player software, such as Trillian and Winamp, due to the association with fun that such programs try to encourage

Skinning is typically implemented with a model–view–controller architecture, which allows for a flexible structure in which the interface is independent from and indirectly linked to application functionality, so the GUI can be easily customized. This allows the user to select or design a different skin at will, and also allows for more deep changes in the position and function of the interface elements.

Different available set of skinning and CSS styling best practices can be access through the referred links: (References[36-39])

1.2.9 Existing Skinning Approaches

Based on study and implementation of UI screens through multiple web design frameworks, below categories are identified in Skin design approach:

- Styling approaches are mainly of three types in existing frameworks in place:
  - ✓ Standard Theme based Styling
  - ✓ CSS Style-sheet based Styling
  - ✓ Pluggable or Custom Skin based Styling
- In the first approach various Standard Themes are provided by the Framework, which has certain built-in configuration settings. Developer can use any one of the built-in standards in their application. However, there is very small or absolutely no room for change, or scope for creating Custom Themes.
  e.g., SWING provides various standard themes like Motif, Metal, GTK+, System etc., Windows Store Apps provide two standard themes Dark & Light, JSF Richfaces library provides built-in themes like Plain, EmeraldTown, BlueSky, Wine etc.

Example from Swing:

```
UIManager.setLookAndFeel(lookAndFeel);
// If L&F = "Metal", set the theme
if (LOOKANDFEEL.equals("Metal")) {
if (THEME.equals("DefaultMetal"))
MetalLookAndFeel.setCurrentTheme(new  DefaultMetalTheme());
else if (THEME.equals("Ocean"))
MetalLookAndFeel.setCurrentTheme(new OceanTheme());
else
MetalLookAndFeel.setCurrentTheme(new CustomTheme());
UIManager.setLookAndFeel(new MetalLookAndFeel());
}
```

- In CSS Style sheet based approach of skinning separate style sheet classes need to be created which in turn gets included in the page components. Style attributes defined at component level can override general styling, which are provided through general style classes.
  e.g., SiteMesh, QT QML, CSS3 FlexBox

Example from SiteMesh:

```
<?xml version="1.0" encoding="UTF-8" ?>
<%@ taglib uri="http://www.opensymphony.com/sitemesh/decorator" prefix="decorator" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<style>
.loginLink {
color: #002862;
font-size: 11px;
line-height: 13px;
padding-left: 0;
padding-top: 25px;
width: 300px;
}
.buttonBold {
display: block;
width: 160px;
height: 25px;
background: #c9dfef;
padding: 10px;
text-align: center;
border-radius: 5px;
color: black;
font-weight: bold;
}
…
…
<div id="leftPanel" style="width: 200px; height: 400px; overflow: scroll">
<table width="100%" border="1">
<tbody>
<tr><td>Account Summary</td></tr>
<tr><td><a href="accountSummary.jsp" class="buttonBold">Current/Savings
Account</a></td></tr>
<tr><td><a href="viewAccountBalance.jsp" class="button">View Account
Balance</a></td></tr>
<tr><td><a href="" class="button">View/Download Account Statement</a></td></tr>
<tr><td><a href="fundXferTranType.jsp" class="button">Fund Transfer</a></td></tr>
<tr><td><a href="" class="button">CASA Sweepin</a></td></tr>
<tr><td><a href="" class="button">Download Historical Statement</a></td></tr>
<tr><td><a href="" class="button">Mailbox</a></td></tr>
<tr><td><a href="" class="buttonBold">Fixed Deposit</a></td></tr>
<tr><td><a href="" class="button">Fixed Deposits Summary</a></td></tr>
<tr><td><a href="" class="button">Open Fixed Deposit &lt Rs 1 Cr</a></td></tr>
<tr><td><a href="" class="button">Open Fixed Deposit &gt  Rs 1 Cr</a></td></tr>
```

```
<tr><td><a href="" class="button">Fixed Deposit Sweep-in</a></td></tr>

</tbody>
</table>
</div>
```

- In Pluggable Custom Skin based approach skins are packaged in external jar files that must be added into the project in order to be able to use them. These skins can be edited by editing the skin ".properties" file kept inside the jar.
  E.g. JSF RichFaces Library

Example from JSF RichFaces:

```
Context parameter :
<context-param>
<param-name>org.richfaces.SKIN</param-name>
<param-value>emeraldTown</param-value>
</context-param>

".properties" file :
#Colors
headerBackgroundColor=#005000
headerGradientColor=#70BA70
headerTextColor=#FFFFFF
headerWeightFont=bold
generalBackgroundColor=#f1f1f1
generalTextColor=#000000
generalSizeFont=18px
generalFamilyFont=Arial, Verdana, sans-serif
controlTextColor=#000000
controlBackgroundColor=#ffffff
additionalBackgroundColor=#E2F6E2
shadowBackgroundColor=#000000
shadowOpacity=1
panelBorderColor=#C0C0C0
subBorderColor=#ffffff
tabBackgroundColor=#ADCDAD
tabDisabledTextColor=#67AA67
trimColor=#BBECBB
tipBackgroundColor=#FAE6B0
tipBorderColor=#E5973E
selectControlColor=#FF9409
generalLinkColor=#43BD43
hoverLinkColor=#FF9409
visitedLinkColor=#43BD43
# Fonts
```

```
headerSizeFont=18px
headerFamilyFont=Arial, Verdana, sans-serif
tabSizeFont=11
tabFamilyFont=Arial, Verdana, sans-serif
buttonSizeFont=18
buttonFamilyFont=Arial, Verdana, sans-serif
tableBackgroundColor=#FFFFFF
tableFooterBackgroundColor=#cccccc
tableSubfooterBackgroundColor=#f1f1f1
tableBorderColor=#C0C0C0
tableBorderWidth=2px
#Calendar colors
calendarWeekBackgroundColor=#f5f5f5
calendarHolidaysBackgroundColor=#FFEBDA
calendarHolidaysTextColor=#FF7800
calendarCurrentBackgroundColor=#FF7800
calendarCurrentTextColor=#FFEBDA
calendarSpecBackgroundColor=#E2F6E2
calendarSpecTextColor=#000000
warningColor=#FFE6E6
warningBackgroundColor=#FF0000
editorBackgroundColor=#F1F1F1
editBackgroundColor=#FEFFDA
#Gradients
gradientType=plain
```

1.3.Contributions of the Thesis

The thesis is aimed at addressing common pain-points in existing Layout, Navigation & Theme design while creating a web page. Below are the outcomes which are achieved at current point of time:

- Identifying common structural elements in GUI frameworks and using it to create a set of criteria for comparison of such frameworks
    - ✓ Implementation based analysis of seven commonly used UI design frameworks
    - ✓ Analysis covering Layout, Navigation and Theme implementation approaches
    - ✓ Framework comparison on few major pain areas or attributes
- Creating a taxonomy of Layout Patterns
    - ✓ Layout design patterns identification based on a four step approach
    - ✓ Page view patterns
    - ✓ Patterns in application symmetric view design requirements
    - ✓ Patterns in positioning approach
    - ✓ Patterns in layout re-rendering or adjustment with screen size

- Design and Implementation of improved Layout support in an existing framework
  - ✓ Target UI design framework is named as LearnITy Framework
  - ✓ LearnITy Framework encourages declarative approach of page creation
  - ✓ The baseline version having provision of Absolute Positioning only
  - ✓ Enhanced layout design created two modes of layout – Container and Component
  - ✓ Container layout having the ability to render elements horizontally, vertically or diagonally – concept of layout managers are taken into consideration
  - ✓ Component layout providing provision for Border Layout and Anchor Layout
  - ✓ Border layout supports thirteen different border positions on a page
  - ✓ Anchor layout encourages reference based positioning with provision of providing required indentation
  - ✓ Container layout supersedes Component layout
- Improvement in other design aspects of the target framework
  - ✓ Baseline version of LearnITy Framework not having any XML content validation in place
  - ✓ XML page validation via XSD structure is incorporated in LearnITy Framework

1.4.Organization of the Thesis

This thesis covers study of some of existing framework and then study, implementation and improvement of LearnITy Framework.

- **Chapter 2:** Contains the discussion about different web design frameworks by implementation of a sample GUI application. It elaborates the seven existing frameworks – Swing, SWT, JSF, Windows Store Apps, SiteMesh, QT QML and CSS3 Flexbox – which were chosen for hands-on. Each framework contains its study outcome in terms of Layout Support, Navigation Support, Theme Support, and Sample Implementation of chosen Bank screens and Evaluation through Implementation Experiences.

- **Chapter 3**: Contains Framework Comparison among all the above mentioned seven frameworks in terms of below attributes :
  - ✓ Support for Form Layout
  - ✓ Support for Menu Layout
  - ✓ Support for Multi-Column Page Layout
  - ✓ Support for Liquid or Fluid Layout
  - ✓ Browser Support
  - ✓ Deployment
  - ✓ Browser Execution Code and Page Quality

- ✓ Performance

- **Chapter 4**: Contains layout design patterns with steps by step approach of deciding on layout designs and positioning elements. It depicts a four steps process towards identifying appropriate GUI design framework as a solution based on perceptions and constraints of individual developer at each step existing patterns are also elaborated.

- **Chapter 5**: Contains the discussion on basic concepts of LearnITy Framework, its motivation, best possible usage, main components of the framework, its existing usage workflow diagram and sample screens implemented using the framework.

- **Chapter 6**: Contains the improved design details of LearnITy Framework. The improvement implemented in terms of enhanced layout support, enhanced skin design, enhanced navigation design and inclusion of XML validation via XSD approach is elaborated in this chapter. This chapter also provides the details of Java implementation pseudo code in existing and modified design of the targeted framework.

- **Chapter 7**: Contains further scope of improvement opportunities in LearnITy Framework.

- **Chapter 8**: Contains References

- **Appendix 1**: Contains live implementation of LearnITy Framework with configuration file details which a developer may refer to and reuse for new application UI development.

- **Appendix 2**: Contains different approach and classification of layout patterns in terms of flexibility in positioning components on screens depending upon varying size of screens. Static Positioning, Fluid Positioning, Adaptive Positioning and Responsive Positioning are few identified patterns.

2. Study on Different Web Frameworks via Implementation of a GUI Application

2.1 Identify Framework for Study and Hands-On

We have identified around 16 frameworks for our initial study. Those are as follows:

Based on Monolithic Environments: These are designated by single-tiered software applications in which the user interface and data access code are combined into a single program. A software system is called "monolithic" if it has a monolithic architecture, in which functionally distinguishable aspects (for example data input and output, data processing, error handling, and the user interface), are not architecturally separate components but are all interwoven.

Below frameworks are examples of Monolithic Architecture based UI design Framework:

- Windows Store apps using C#/VB/C++ and XAML
- Cocoa
- Swing
- SWT
- Qt with QML

Based on Client Server System: The client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service which are called servers and service requesters which are called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

Below framework is an example of Client-Server Model based UI design Framework:

- X Windows Motif

Server side System: Server-centric frameworks keep a component UI tree on the server. The state of input fields and UI components are synchronized with the server side representation. Usually it provides data binding for free and can easily perform server side validations as well as, access any backend services. Server side validation is implemented in Java and is easier unit testable. In addition, client side validation can be implemented to avoid server round trips.

Below frameworks are examples of Server side UI design Framework:

- Java Server Faces(JSF)
- SiteMesh(Web Template based Framework)

- Apache Tapestry
- Apache Tiles
- Vaadin Framework

Client side System: Client-centric web frameworks don't have to interact with the server. Its entire code is translated into JavaScript and lives self-contained in the browser. It is very convenient to build one-page, offline, office-like applications with only little communication with the server is required. Client-centric framework is entirely executed on the client and thus it should not consume any resources (CPU / memory) on the server.

Below frameworks are examples of Client side UI design Framework:

- JQuery
- Sencha
- Adobe Flex
- MXML ActionScript
- CSS3 Flexbox

Out of which we have selected 4 Monolithic (Windows Store App, Swing, SWT & QT with QML) and 2 Server side (JSF and SiteMesh) and 1 Client side (CSS3 Flexbox) frameworks considering the widespread acceptability and usage in developer community and time constraint.

2.2 Identify Target GUI Application for Hands-On using Different Web Frameworks

We have identified the banking application as a target GUI application for hands-on with the different frameworks. The reason for this is:

The banking application is one of the real-life business applications. The screens have diverse set of components with different layouts. It poses a sufficiently complicated example to be implemented using various existing layout methods and helps in experiencing the pros and cons of different approaches. Moreover, the selection of screens are such that it covers the back and forth navigation facility also within it.

2.3 Study Outcome of Swing Framework

Swing(Reference[1]) is a platform-independent, Model-View-Controller GUI framework for Java, which follows a single-threaded programming model. Additionally, this framework provides a layer of abstraction between the code structure and graphic presentation of a Swing-based GUI.

2.3.1 Layout Support in SWING

Different kinds of positioning can be achieved in SWING framework in following ways:

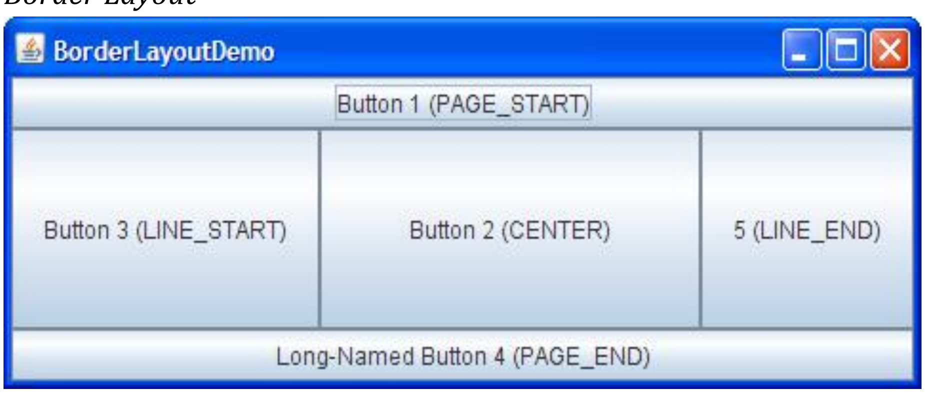
- Available Layout Managers
- Custom Layout Manager Creation
- Absolute Positioning

a) Available Layout Managers

Swing framework provides below Layout Managers to work with:

- Border Layout
- Box Layout
- Card Layout
- Flow Layout
- Grid Bag Layout
- Grid Layout
- Group Layout
- Spring Layout

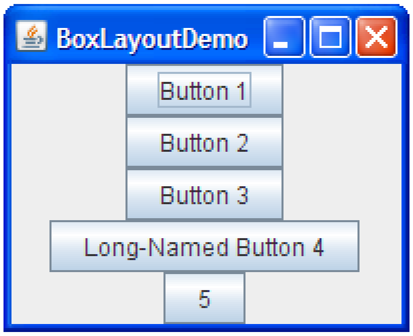
Border Layout



- Every content pane is initialized to use a Border Layout
- A Border Layout places components in up to five areas: Top, Bottom, Left, Right and Center. These areas can be specified by Border Layout constants:

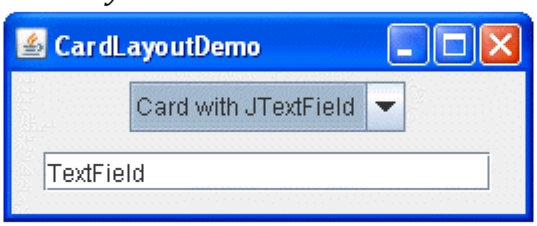
PAGE_START, PAGE_END, LINE_START, CENTER and LINE_END – as shown in figure above

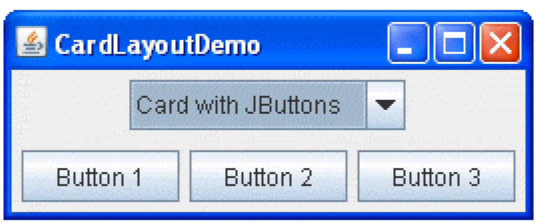
Box Layout



- Box Layout class puts components in a single row or column
- It respects the components' requested maximum size and lets developer align the components

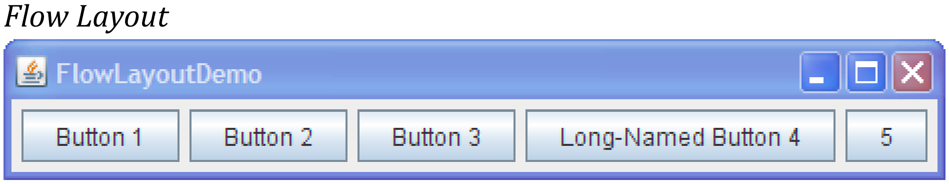
Card Layout





- Card Layout lets developer implement an area that contains different components at different times
- Card Layout is often controlled by a combo box, with the state of the combo box determining which panel the Card Layout displays

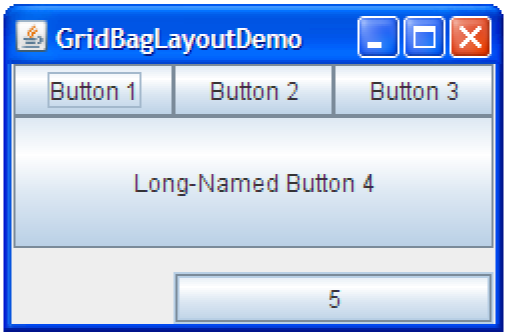
Flow Layout



- Flow Layout is the default layout manager for every JPanel

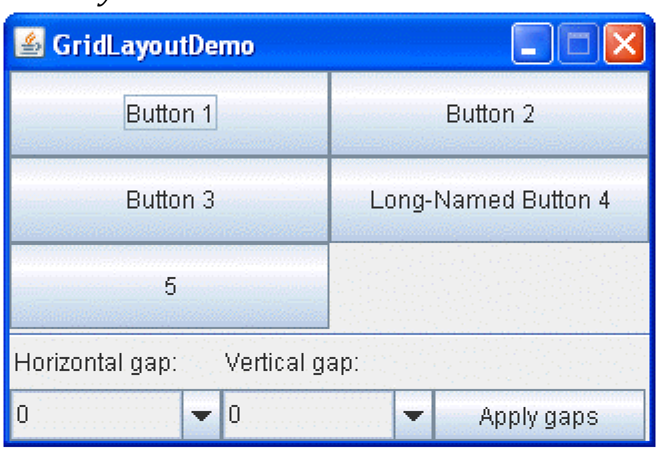
- Flow Layout lays out components in a single row, starting a new row if its container is not sufficiently wide

Grid Bag Layout



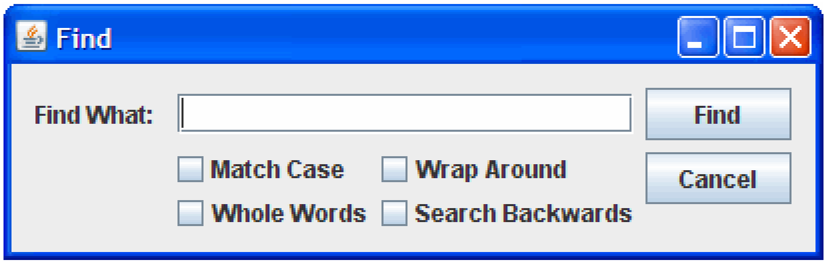
- Grid Bag Layout is a sophisticated, flexible layout manager
- It aligns components by placing them within a Grid of cells, allowing components to span more than one cell
- The rows in the grid can have different heights and columns can have different width

Grid Layout



- Grid Layout simply makes a bunch of components equal in size and displays them in requested number of rows and columns

Group Layout



- Group Layout is a Layout Manager that was developed for GUI builders such as Matisse, the GUI builder provided with NetBeans IDE

- Group Layout works with Horizontal and Vertical layouts separately. The layout is defined for each dimension independently. Each component needs to be defined twice in the layout

- Group Layout used two types of arrangement: Sequential and Parallel, combining with hierarchical composition

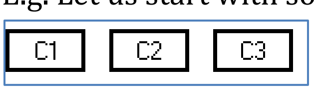
- With sequential arrangement, the components are simply placed one after another, just like BoxLayout or FlowLayout would do along one axis. The position of each component is defined as being relative to the preceding component

- The second way places the components in parallel—on top of each other in the same space. They can be baseline-, top-, or bottom-aligned along the vertical axis. Along the horizontal axis, they can be left-, right-, or center-aligned if the components are not all the same size

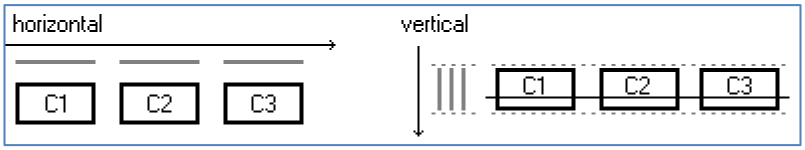
- The size of a sequential group is the sum of the sizes of the contained elements, and the size of a parallel group corresponds to the size of the largest element

- Defining a layout means defining how the components should be grouped by combining the sequential and parallel arrangements

E.g. Let us start with something simple, just three components in a row:



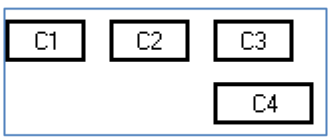
We will express this layout using groups. Starting with the horizontal axis it is easy to see there is a sequential group of 3 components arranged from left to right. Along the vertical axis there is a parallel group of the same 3 components with the same location, size, and baseline:



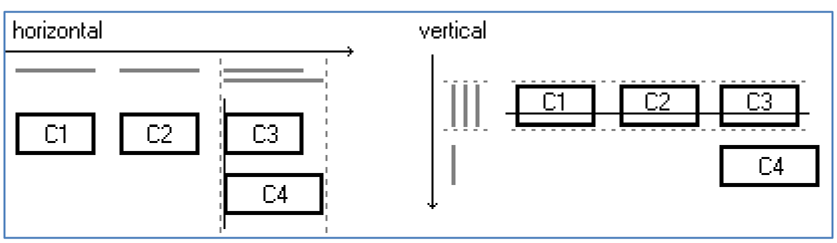
In pseudo code, the layout specification might look like this (the real code is in the Writing Code section below):

horizontal layout = sequential group { c1, c2, c3 } *vertical layout = parallel group (BASELINE) { c1, c2, c3 }*

Now adding one more component, C4, left-aligned with C3:



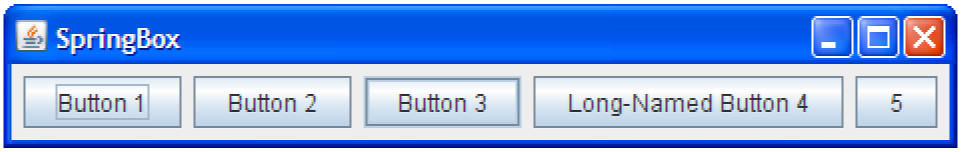
Along the horizontal axis the new component occupies the same horizontal space as C3 so that it forms a parallel group with C3. Along the vertical axis C4 forms a sequential group with the original parallel group of the three components.

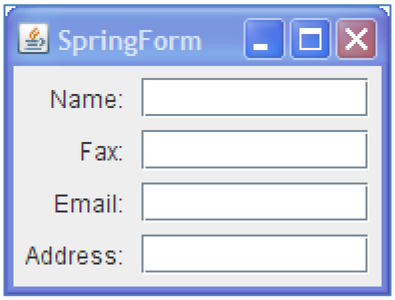


In pseudo code, the layout specification now looks like this:

horizontal layout = sequential group { c1, c2, parallel group (LEFT) { c3, c4 } } *vertical layout = sequential group { parallel group (BASELINE) { c1, c2, c3 }, c4 }*

Spring Layout





- Spring layouts do their job by defining directional relationships, or constraints, between the edges of components. For example, one might define that the left edge of one component is a fixed distance (5 pixels, say) from the right edge of another component

- Unlike other layout managers, Spring Layout does not automatically set the location of the components it manages. Component locations need to be initialized by constraining the west/east and north/south locations

b) Custom Layout Manager Creation

To create a custom layout manager, developers must create a class that implements the LayoutManager interface. It can be implemented directly, or its sub interface, LayoutManager2 can also be implemented.

Every layout manager must implement at least the following five methods, which are required by the LayoutManager interface:

- **void addLayoutComponent(String, Component)**
Called by the Container class's add methods. Layout managers that do not associate strings with their components generally do nothing in this method.

- **void removeLayoutComponent(Component)**
Called by the Container methods remove and removeAll. Layout managers override this method to clear an internal state they may have associated with the Component.

- **Dimension preferredLayoutSize(Container)**
Called by the Container class's getPreferredSize method, which is it called under a variety of circumstances. This method should calculate and return the ideal

size of the container, assuming that the components it contains will be at or above their preferred sizes. This method must take into account the container's internal borders, which are returned by the getInsets method.

- **Dimension minimumLayoutSize(Container)**

Called by the Container getMinimumSize method, which itself is called under a variety of circumstances. This method should calculate and return the minimum size of the container, assuming that the components it contains will be at or above their minimum sizes. This method must take into account the container's internal borders, which are returned by the getInsets method.

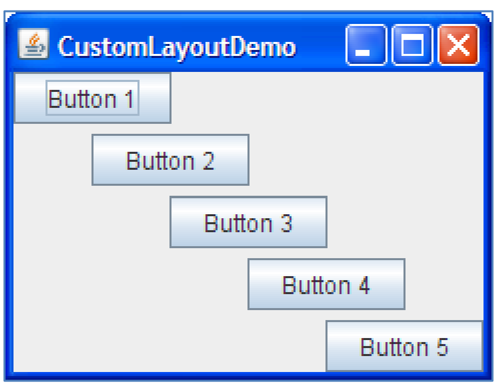
- **void layoutContainer(Container)**

Called to position and size each of the components in the container. A layout manager's layoutContainer method does not actually draw components. It simply invokes one or more of each component's setSize, setLocation, and setBounds methods to set the component's size and position.

To support component constraints, maximum sizes, or alignment, custom Layout manager should implement LayoutManager2 interface, which add following five methods to those required by LayoutManager:

- ✓ addLayoutComponent(Component, Object)
- ✓ getLayoutAlignmentX(Container)
- ✓ getLayoutAlignmentY(Container)
- ✓ invalidateLayout(Container)
- ✓ maximumLayoutSize(Container)

Example of a Custom Layout



c) Doing Without a Layout Manager (Absolute Positioning)

Although it is possible to do without a layout manager, developers should use a layout manager if at all possible. A layout manager makes it easier to adjust to look-and-feel-dependent component appearances, to different font sizes, to a container's changing size, and to different locales. Layout managers also can be reused easily by other containers, as well as other programs.

If a container holds components whose size is not affected by the container's size or by font, look-and-feel, or language changes, then absolute positioning might make sense. Desktop panes, which contain internal frames, are in this category. The size and position of internal frames does not depend directly on the desktop pane's size. The programmer determines the initial size and placement of internal frames within the desktop pane, and then the user can move or resize the frames. A layout manager is unnecessary in this situation.

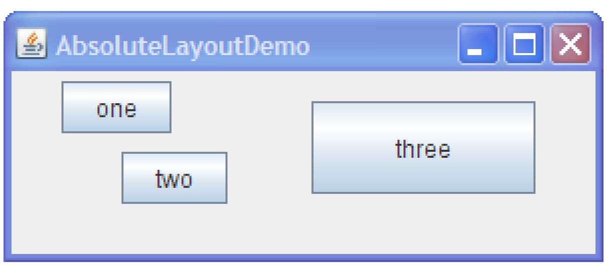
Another situation in which absolute positioning might make sense is that of a custom container that performs size and position calculations that are particular to the container, and perhaps require knowledge of the container's specialized state. This is the situation with split panes.

Creating a container without a layout manager involves the following steps.

- Set the container's layout manager to null by calling setLayout(null)
- Call the Component class's setbounds method for each of the container's children
- Call the Component class's repaints method

However, creating containers with absolutely positioned containers can cause problems if the window containing the container is resized.

Example of Absolute Positioning



2.3.2 Navigation Support in SWING

There is no specific Navigation support in SWING. There will be a common panel available. For navigating through various pages of the application, firstly old panel contents need to be removed, by calling the dispose command like this:

```
center.remove();
```

Then elements can be added to the panel as new contents as follows, layout defined:

```
center.add(contentGrid, BorderLayout.NORTH);
```

Finally, panel needs to be validated, repaint and packed for proper display.

```
center.validate();
center.repaint();
pack();
setVisible(true);
```

2.3.3 Theme Support in SWING

Look-and-Feel comes in two flavors in SWING framework:

- Platform Look-and-Feel
- Third-Party Look-and-Feel

a) Platform Look-and-Feel

Default Cross-Platform LAF is called Metal. This LAF comes with several Themes:

- DefaultMetalTheme, which was historically the first Swing default theme
- Ocean Theme, which is smoother than DefaultTheme and became the default in Java 5.0

Other cross-platform LAFs:

- Synth, a skinned LAF called which is configured with an XML property file
- Nimbus, which is added for the Java SE 6 Update

Platform-dependent LAF, which aims to be the closest as possible to the platform native GUI:

- Windows, which depends on the version of the Windows OS
- Motif or GTK+, which are LAFs for Linux or Solaris
- Specific vendor LAF for IBM AIX, HP-UX, Mac OS X

b) Third-Party Look-and-Feel

Numerous other LAFs are developed by Third-Party:

- Insubstantial, a maintenance fork of Substance
- JGoodies Windows and plastic look and feels
- Liquid, which aim to look like the Liquid theme for KDE
- Napkin, which can be used to make GUI work look provisional
- Substance, a skinned look and feel
- Synthetica, which is based on synth

2.3.4 Sample Screens Implemented in SWING

Few pages of a sample Bank Website was built using SWING framework, taking help of different inbuilt layout managers:

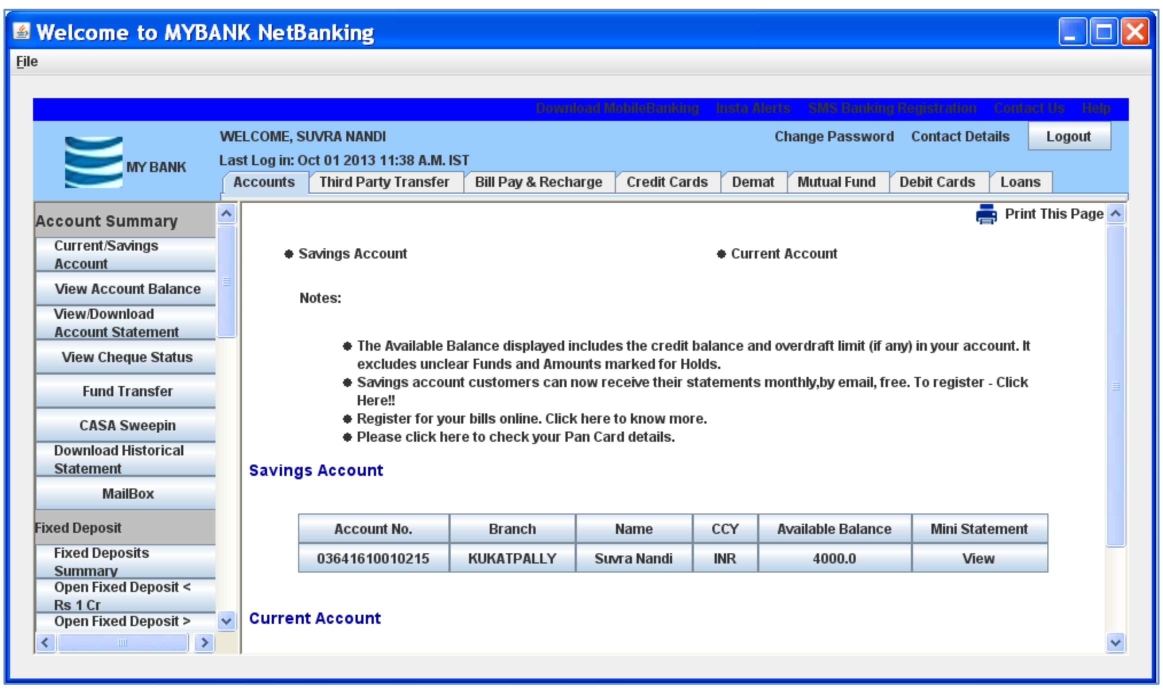


Figure 1: Swing Implementation Screen - Current/Savings Account

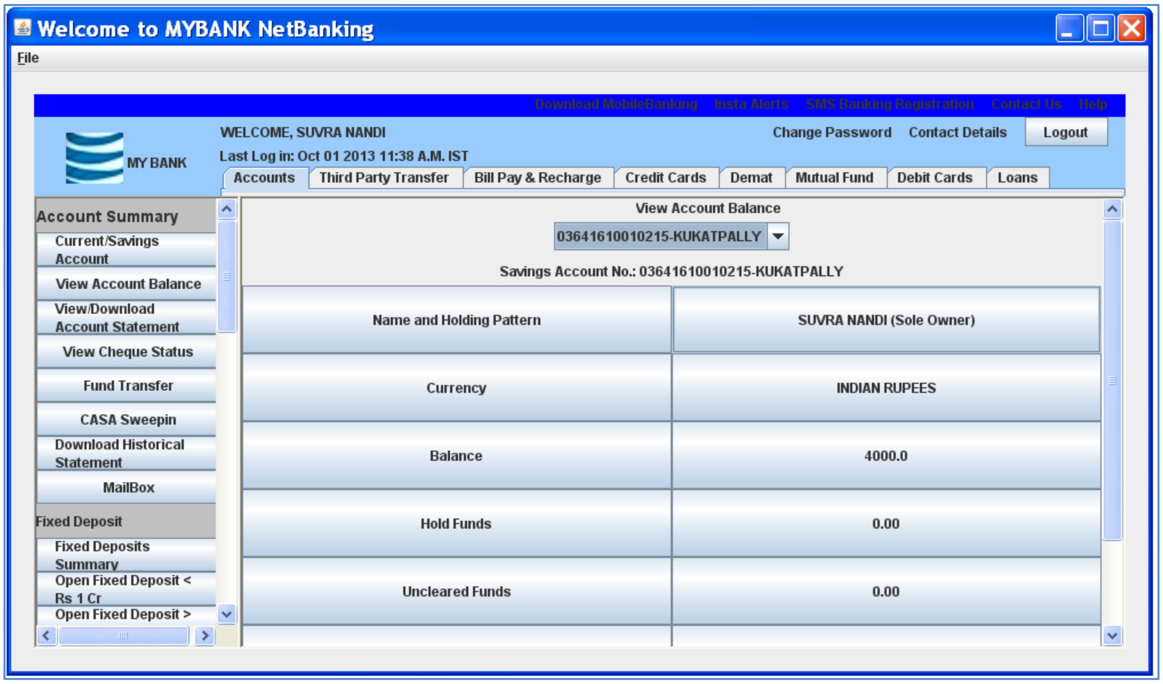


Figure 2: Swing Implementation Screen - View Account Balance-Savings

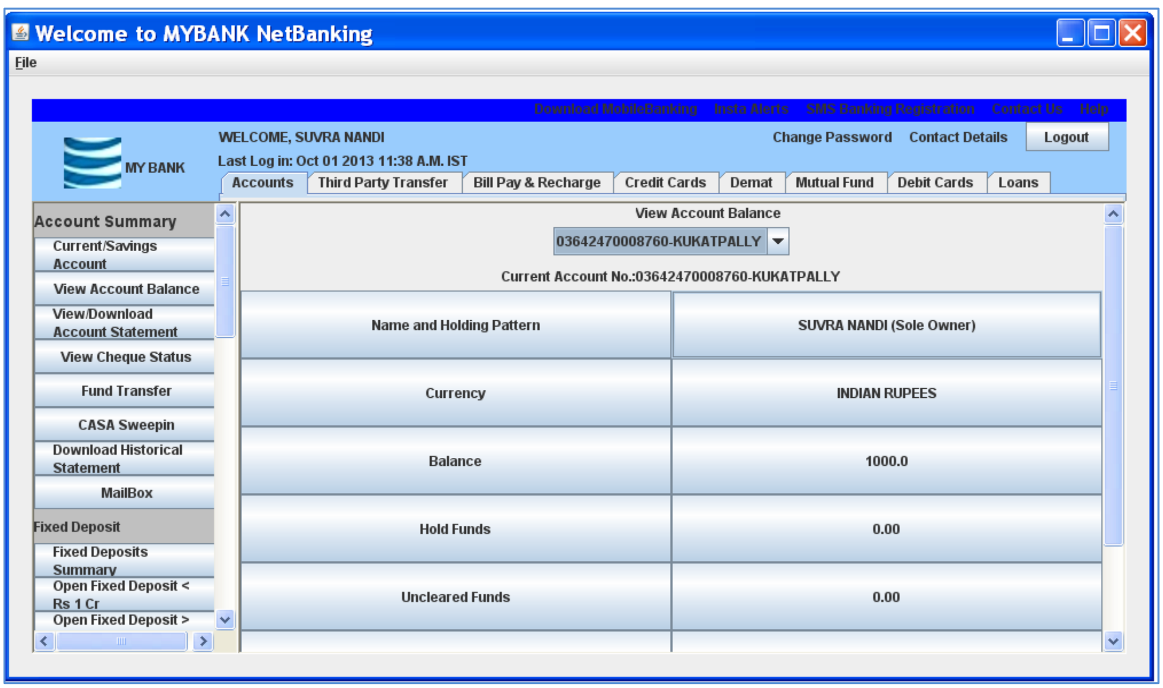


Figure 3: Swing Implementation Screen - View Account Balance-Current

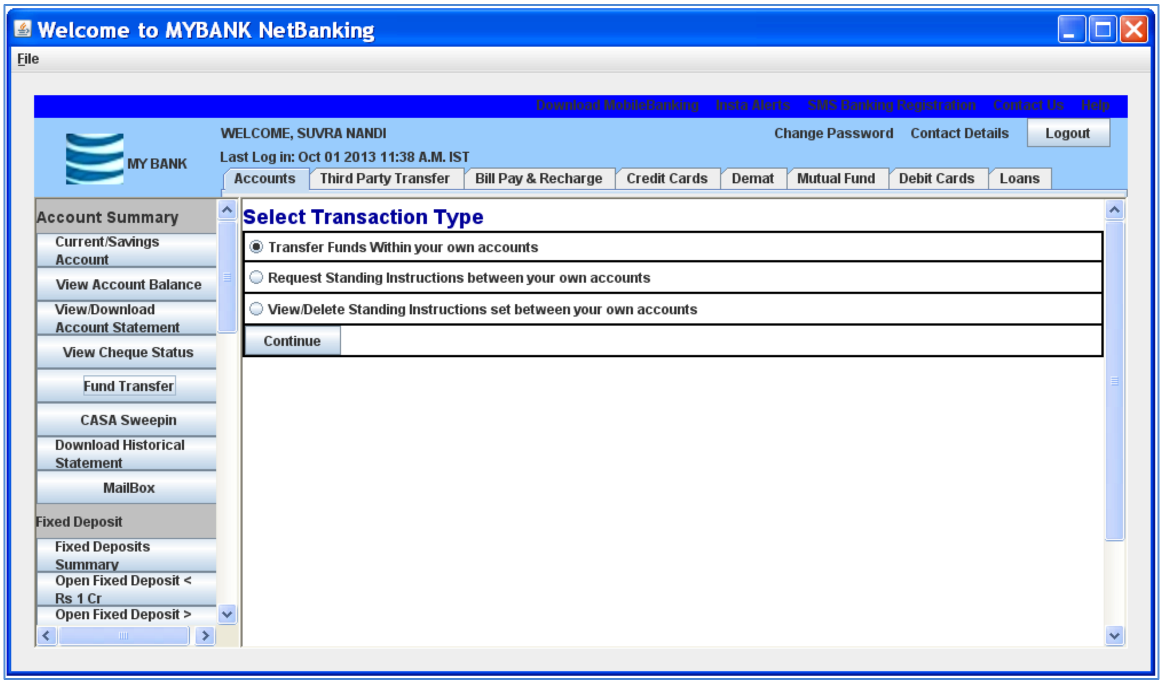


Figure 4: Swing Implementation Screen - Funds Transfer - Select Transaction Type

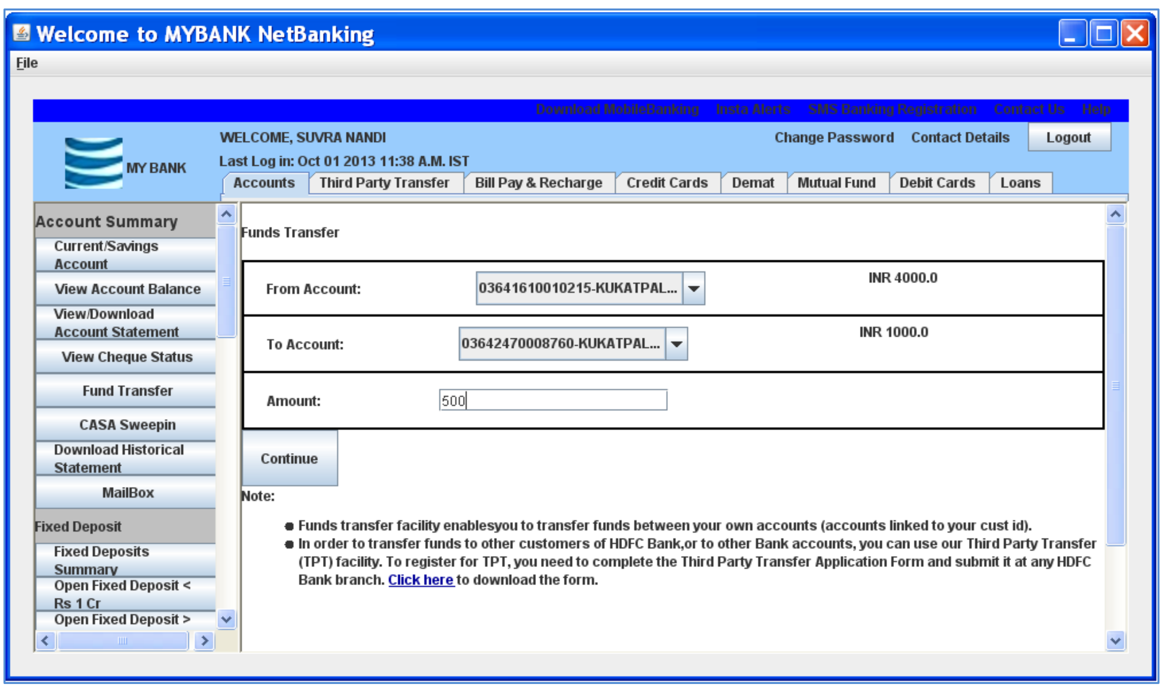


Figure 5: Swing Implementation Screen - Funds Transfer

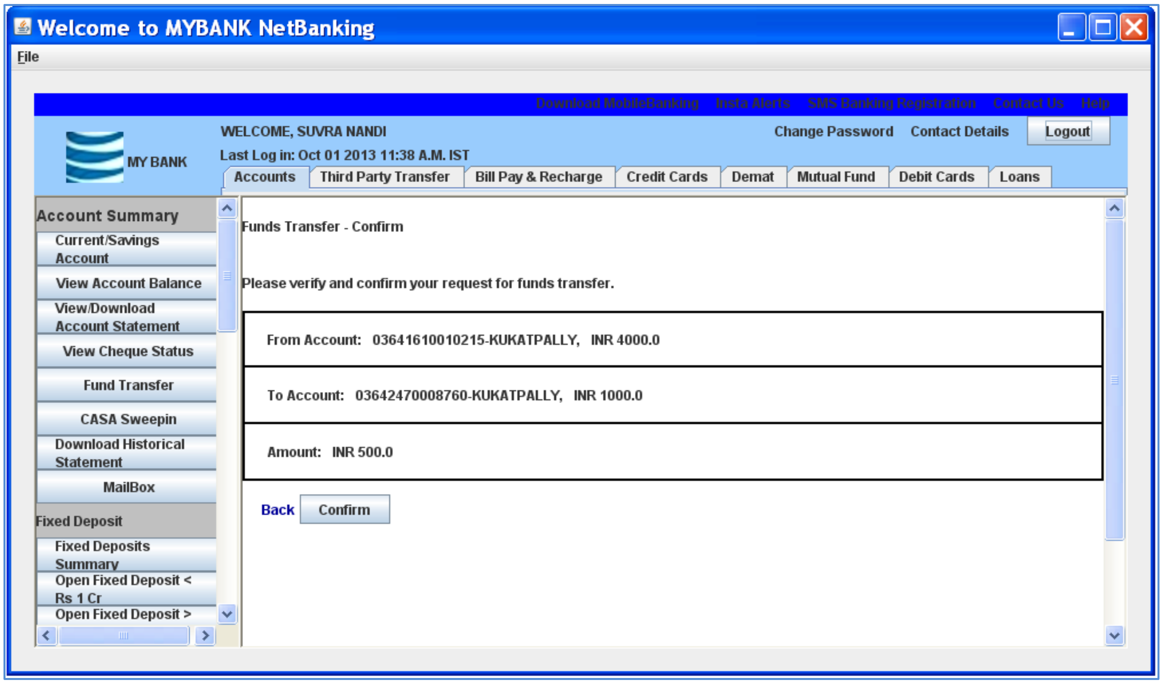


Figure 6: Swing Implementation Screen - Funds Transfer – Confirm

Figure 7: Swing Implementation Screen - Transaction Success Page

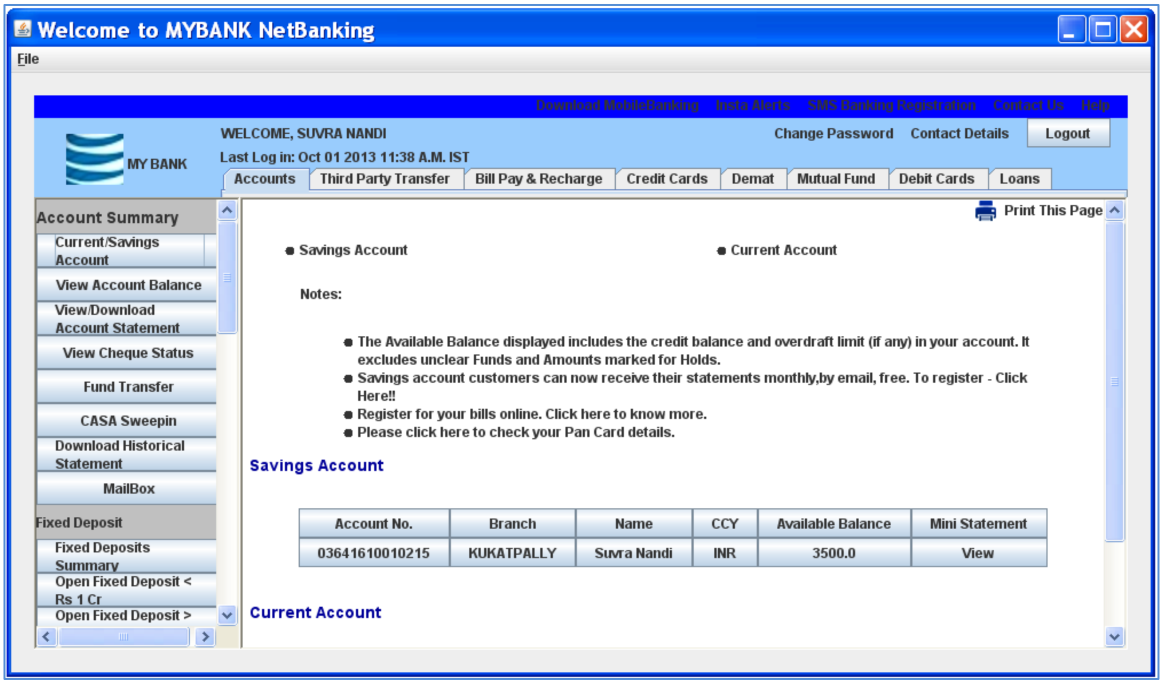


Figure 8: Swing Implementation Screen - Account Summary after Fund Transfer

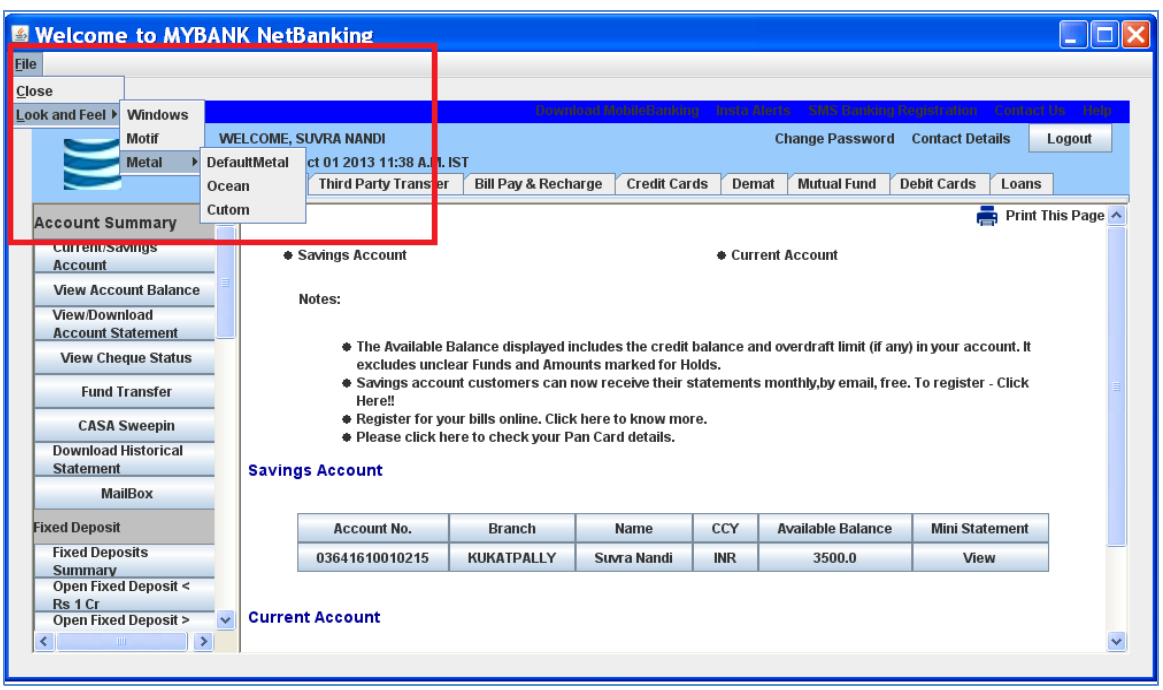


Figure 9: Swing Implementation Screen - Look and Feel Menu

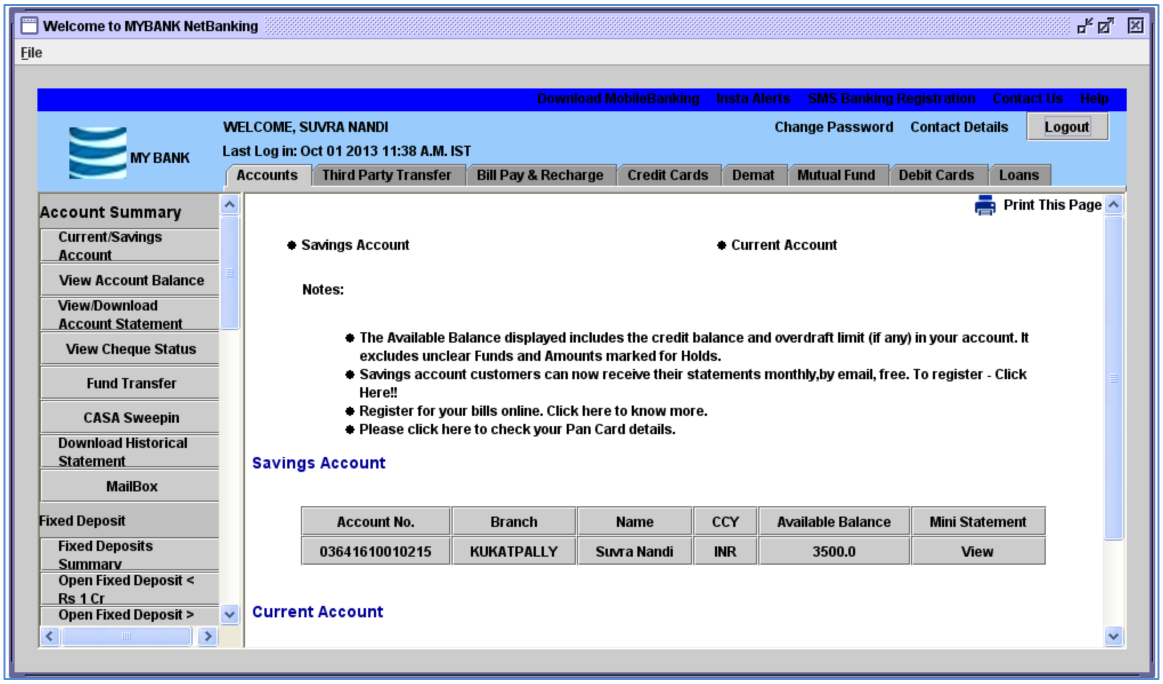


Figure 10: Swing Implementation Screen - Default Metal LAF

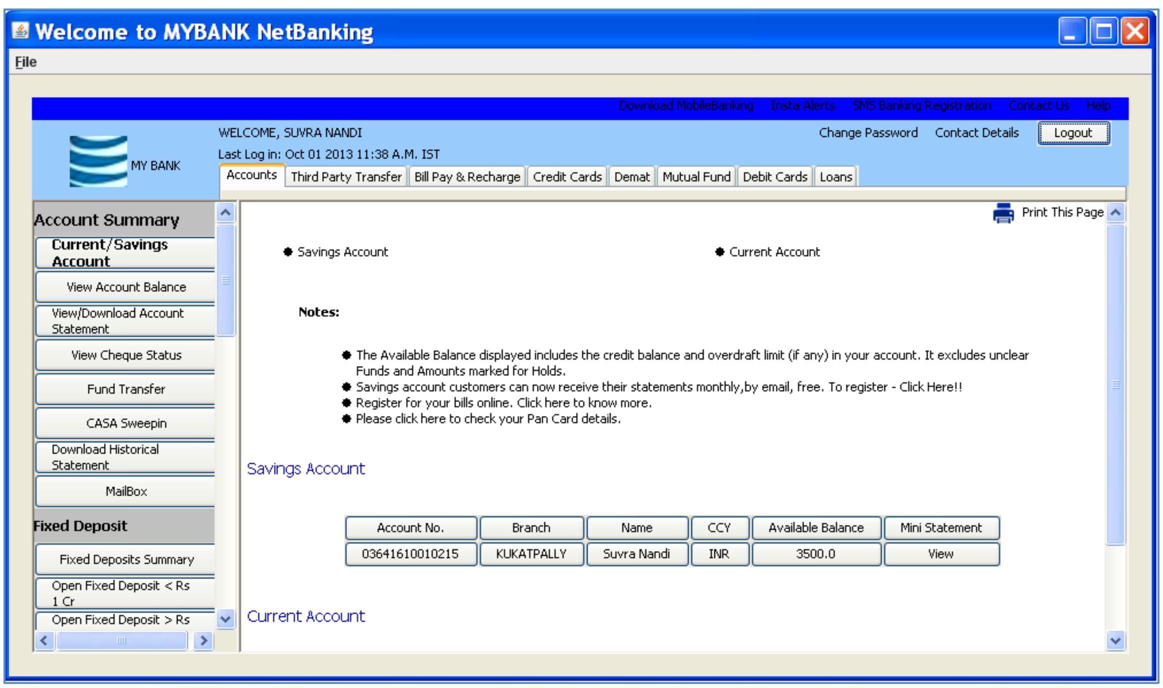


Figure 11: Swing Implementation Screen - System LAF

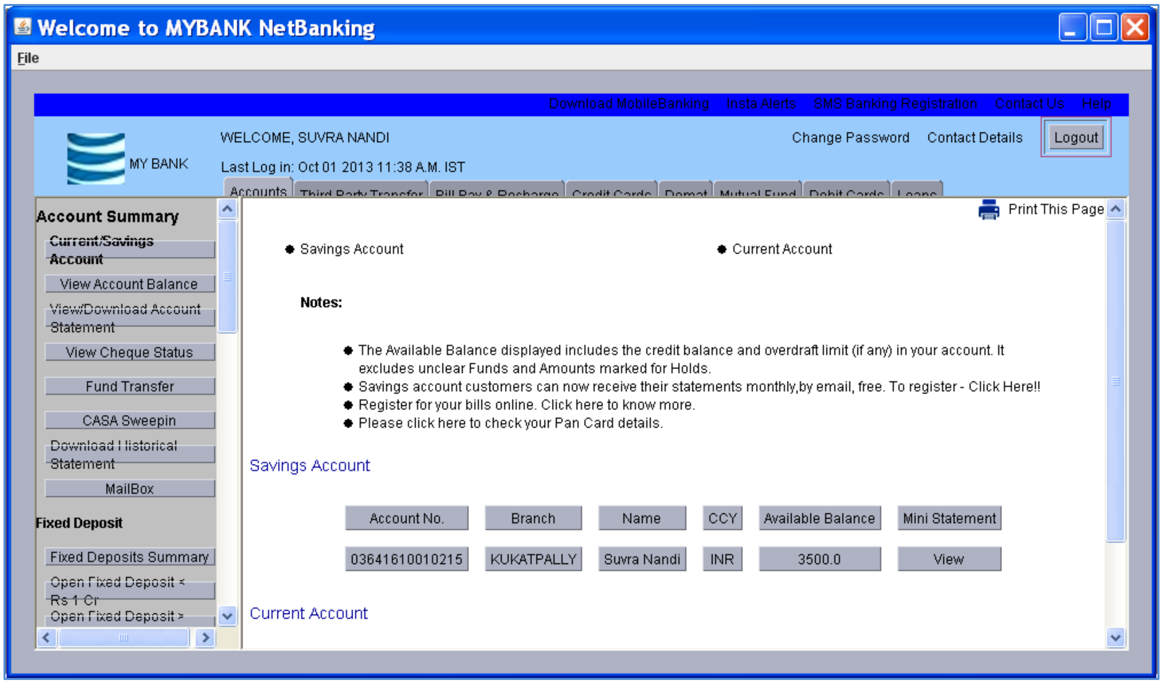


Figure 12: Swing Implementation Screen - Motif LAF

2.3.5 Evaluation through Implementation Experience

Swing is the primary Java GUI widget toolkit. It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs. Swing is the reference GUI toolkit for J2SE.

a) Implementation Notes

Swing provides varieties of **layout managers**, which mostly fits with any type of complex screen designs. Apart from that, developers have the flexibility of creating **Custom Layout Managers** on their own. Swing provides very good **documentation** on layout managers, which help in implementing those efficiently.

While implementing the Bank Application Screens:

- Flow Layout was the Default Layout
- BorderLayout was useful in defining the outer structure of the page
- BoxLayout was used in no. of places for creating widgets in single row, or column
- Card Layout was useful in creating ComboBoxes
- Grid Layout was useful in table structures like View Account Balance tables

- GridBagLayout was used in the Account Summary tables, where column width were not same
- Skinning was implemented by creating Menu, SubMenu and Nested SubMenu Structures, which can be selected by user for changing the skin of the screens
- System, Motif, Metallic Look and Feel were used, with Default, Ocean and Custom Themes for Metallic LAF
- Swing provides different inbuilt Themes and "Look and Feel". Custom Look and Feel can also be created
- Some of the Layout Managers , like GridBagLayout is unnecessarily complex and hard to use
- Swing doesn't provide any support for inbuilt Page Navigation. It needs to be implemented programmatically

b) When SWING should be used?

Developers should choose SWING as GUI Development Framework when they need

- Java Coding

Swing is a Java based framework. As Java is managed code and there are a lot of documentation too.

- MVC architecture is preferred
- Complex Layout

Swing provides a larger set of features, it is much more elegant, and gives a higher abstraction level (that turns helpful when developing complex GUIs with custom components).

2.4 Study Outcome of SWT Framework

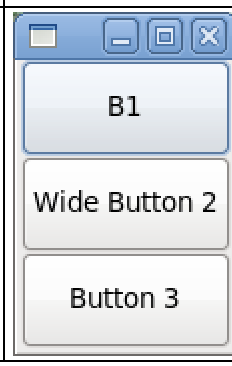
2.4.1 Layout Support in SWT

SWT(Reference[2]) provides few standard layout classes. Layout classes are subclasses of the abstract class Layout. Custom layouts can also be created in SWT.

a) Available Layout Classes

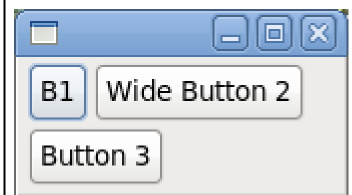
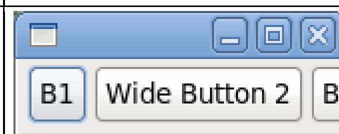
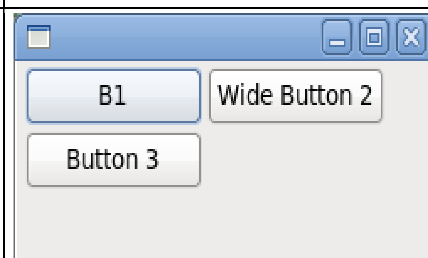
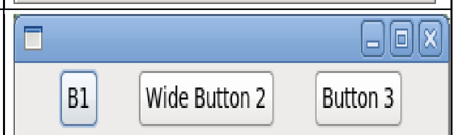
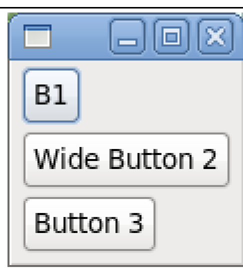
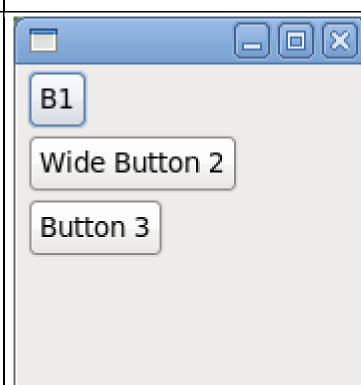
- Fill Layout
- Row Layout
- Grid Layout
- Form Layout

Fill Layout

	Initial	After resize
fillLayout.type = SWT.HORIZONTAL (default)		
fillLayout.type = SWT.VERTICAL		

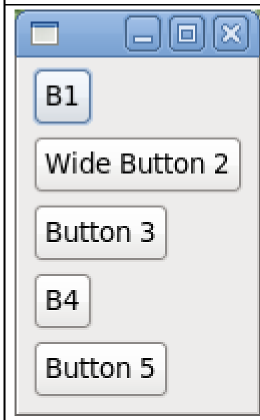
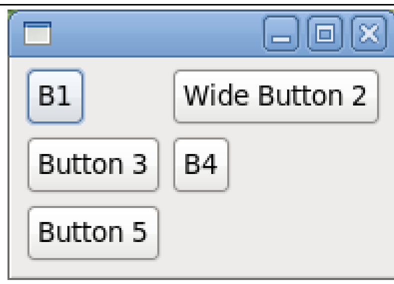
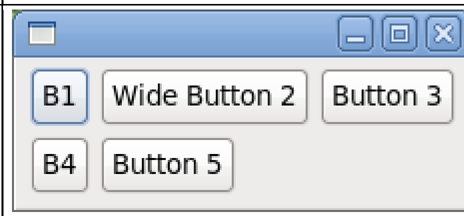
- Fill Layout is the simplest Layout Class
- It lays out widgets in a single row or column, forcing them to be the same size
- Initially, the widgets will all be as tall as the tallest widget, and as wide as the widest
- Fill Layout does not wrap, and Margins or Spacing cannot be provided
- It can be used to lay out buttons in a task bar or tool bar, or to stack checkboxes in a Group. Fill Layout can also be used when a Composite only has one child

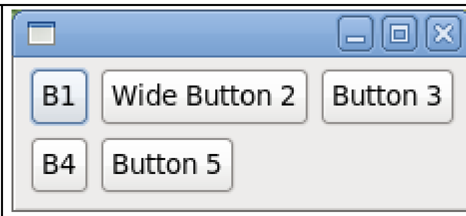
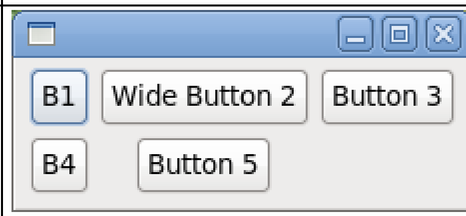
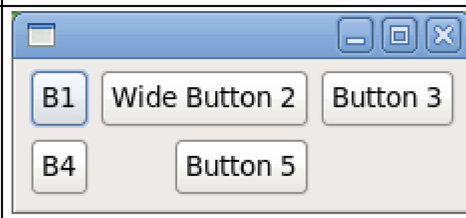
Row Layout

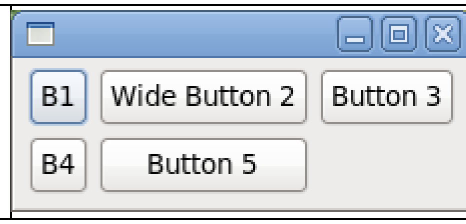
	Initial	After resize
rowLayout.wrap = true; *rowLayout.pack = true;* *rowLayout.justify = false;* *rowLayout.type = SWT.HORIZONTAL;* *(defaults)*		 and 
wrap = false *(clips if not enough space)*		
pack = false *(all widgets are the same size)*		
justify = true *(widgets are spread across the available space)*		
type = SWT.VERTICAL *(widgets are arranged vertically in columns)*		

- RowLayout is more commonly used than FillLayout because of its ability to wrap, and because it provides configurable margins and spacing
- In addition, the height and width of each widget in a RowLayout can be specified by setting the widget's RowData object using setLayoutData

Grid Layout

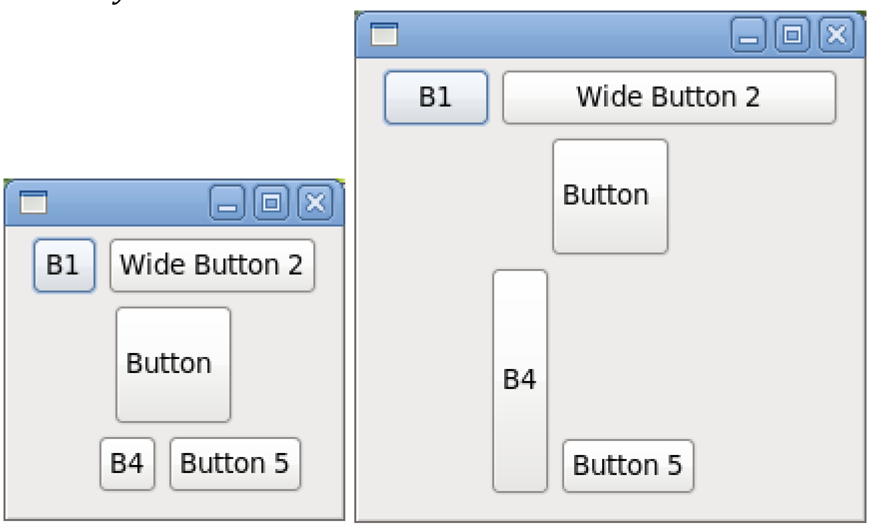
numColumns = 1	*numColumns = 2*	*numColumns = 3*
		

horizontalAlignment = GridData.BEGINNING *(default)*	
horizontalAlignment = GridData.CENTER	
horizontalAlignment = GridData.END	

horizontalAlignment = GridData.FILL	

- With a Grid Layout, the widget children of a Composite are laid out in a grid.
- Grid Layout has a number of configuration fields, and—like Row Layout—the widgets it lays out can have an associated layout data object, called Grid Data. The power of Grid Layout lies in the ability to configure Grid Data for each widget controlled by the Grid Layout.

Form Layout



- Form Layout works by creating FormAttachments for each side of the widget, and storing them in the layout data.
- An attachment 'attaches' a specific side of the widget either to a position in the parent Composite or to another widget within the layout. This provides tremendous flexibility when laid out, as it allows developer to specify the placement of individual widgets within the layout.
- The marginWidth and MarginHeight fields in FormLayout are similar to those in GridLayout. Left and right margins are defined by margin Width, and top and bottom margins are defined by marginHeight. Margins can also be defined on a per-widget basis in the attachments. FormLayout margins are zero by default.

b) Custom Layout Class Creation

Custom Layouts can also be created in SWT, as per developer's need. Before creating Custom Layout, developer must have the knowledge of How Layouts work in SWT:

Layout is the abstract superclass of all layouts. It only has two methods: computeSize and layout. The class is defined as follows:

```
public abstract class Layout {

protected abstract Point computeSize(Composite composite, int widthHint, int heightHint, boolean flushCache);

protected abstract void layout(Composite composite, boolean flushCache);
}
```

The computeSize method calculates the width and height of a rectangle that encloses all of the Composite's children once they have been sized and placed according to the layout algorithm encoded in the Layout class. The hint parameters allow the width and/or height to be constrained. For example, a layout may choose to grow in one dimension if constrained in another. A hint of SWT.DEFAULT means to use the preferred size.

The layout method positions and sizes the Composite's children. A Layout can choose to cache layout-related information, such as the preferred extent of each of the children. TheflushCache parameter tells the Layout to flush cached data.

Since a Layout controls the size and placement of widgets in a Composite, there are several methods in Composite that are used with Layouts.
The first two methods allow setting and getting a Layout object in a Composite.

```
public void setLayout(Layout layout);

public Layout getLayout();
```

An application can force a Layout to recalculate the sizes of and reposition children by sending layout() to the parent Composite.

```
public void layout(boolean changed);

public void layout();
```

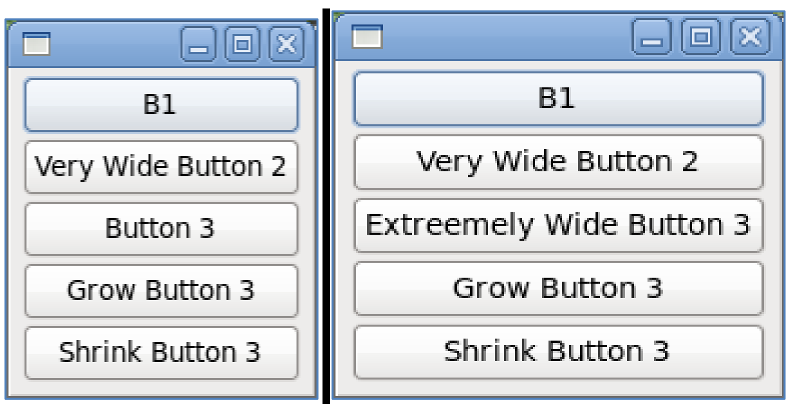
The clientArea of a Composite is the rectangle that will contain all of the children. A Layout positions the children inside the client area.

```
public Rectangle getClientArea ();
```

The trim of a Composite is the area outside the client area. For some composites, the size of the trim is zero. The trim can be computed by passing the dimensions of the client area into the method computeTrim.

```
public Rectangle computeTrim (int x, int y, int width, int height);
```

Below is the demo screen of sample custom layout, called Column Layout:



2.4.2 Navigation Support in SWT

There is no specific Navigation support in SWT, like SWING. There will be a common panel available. For navigating through various pages of the application, firstly old panel contents need to be disposed, by calling the dispose command like this:

```
centerPanel.dispose();
```

Secondly, the panel needs to be reinitialized:

```
centerPanel = new Group(shell, SWT.NONE);
```

Then proper layout needs to be provided to the common panel:

```
centerPanel.setLayout(new FillLayout(SWT.VERTICAL));
```

Now the panel is ready for holding new contents, as if it is getting navigated to a new page.

2.4.3 Theme Support in SWT

SWT doesn't provide any specific support of skinning. SWT widgets do not have skins, unlike SWING widgets. As SWT supports native look and feel, widgets are drawn by the Native Windowing Toolkit like Win32, GTK+ etc.

Themes in SWT depend on the native platform on which its widgets are being drawn.

2.4.4 Sample Screens Implemented in SWT

Few pages of a sample Bank Website was built using SWT framework, taking help of different inbuilt layout classed.

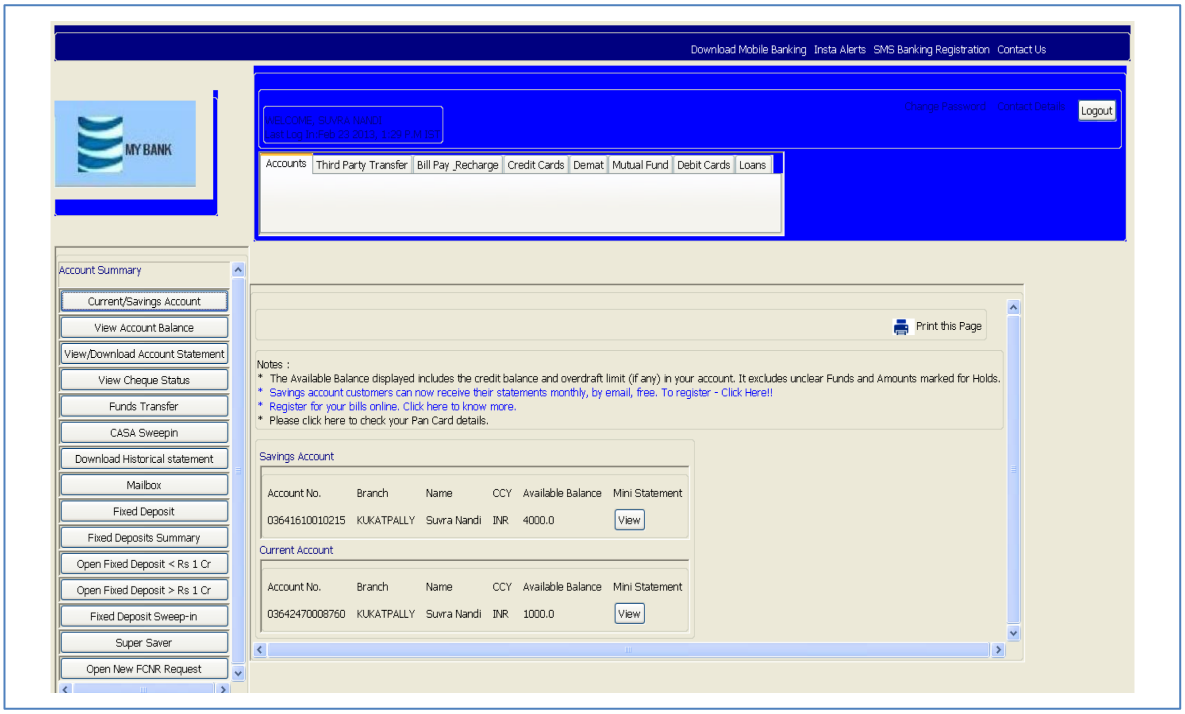


Figure 13: SWT Implementation Screen - Current/Savings Account

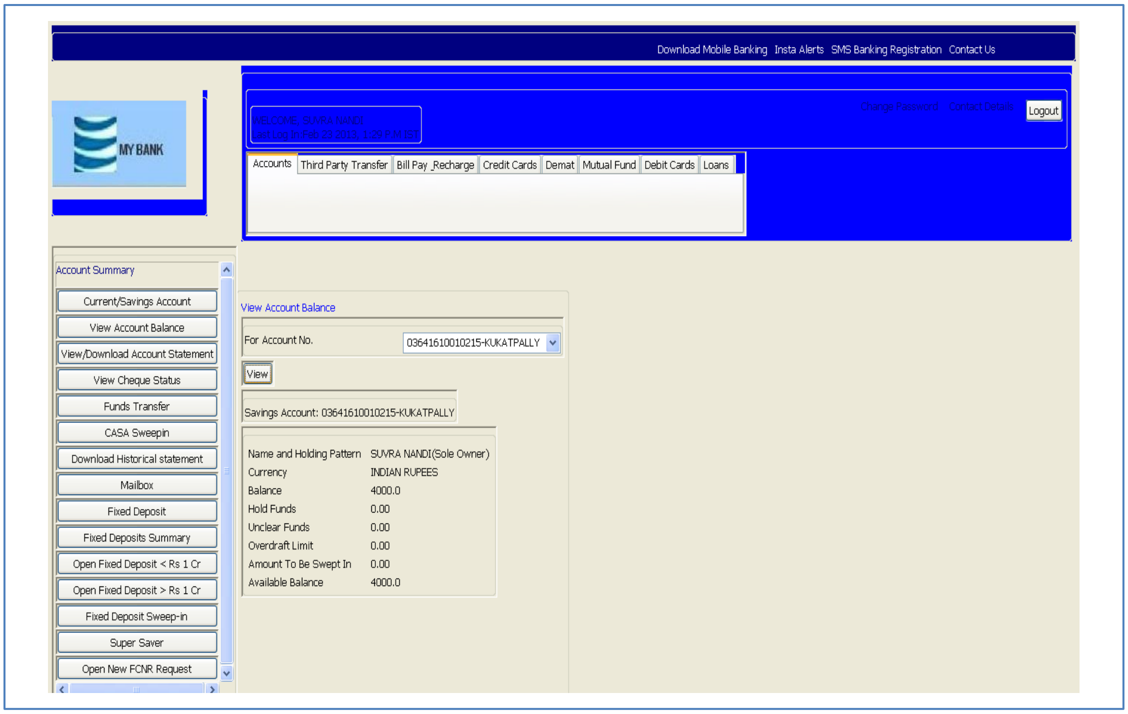


Figure 14: SWT Implementation Screen - View Account Balance – Savings

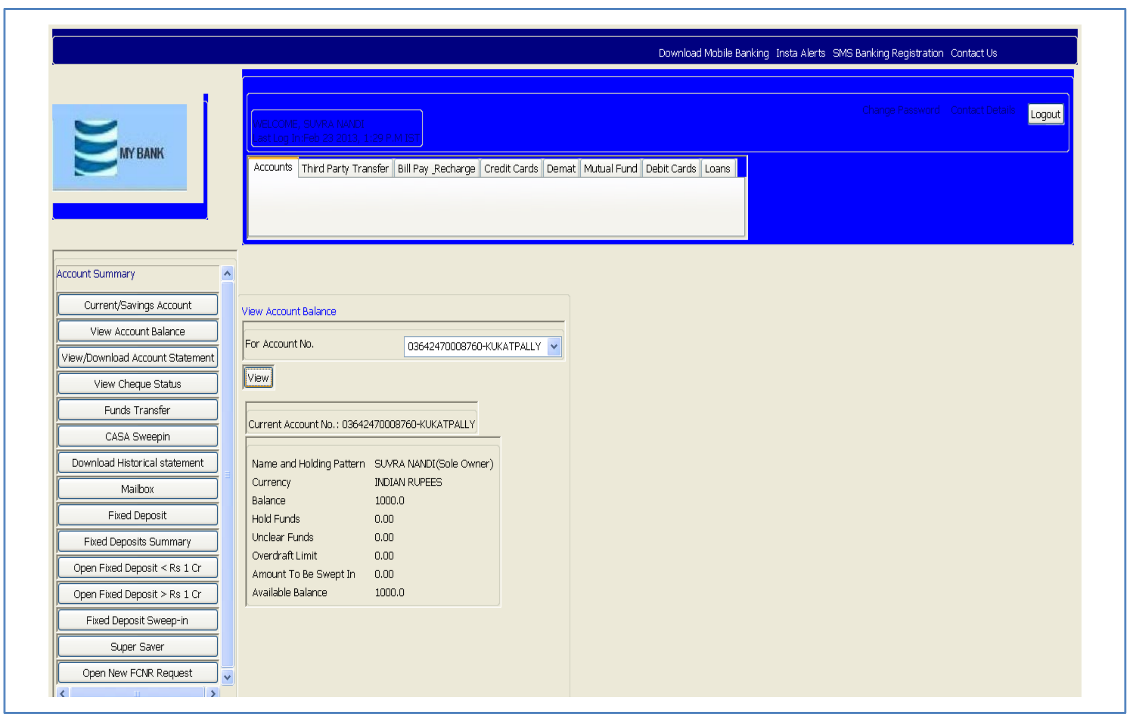


Figure 15: SWT Implementation Screen - View Account Balance – Current

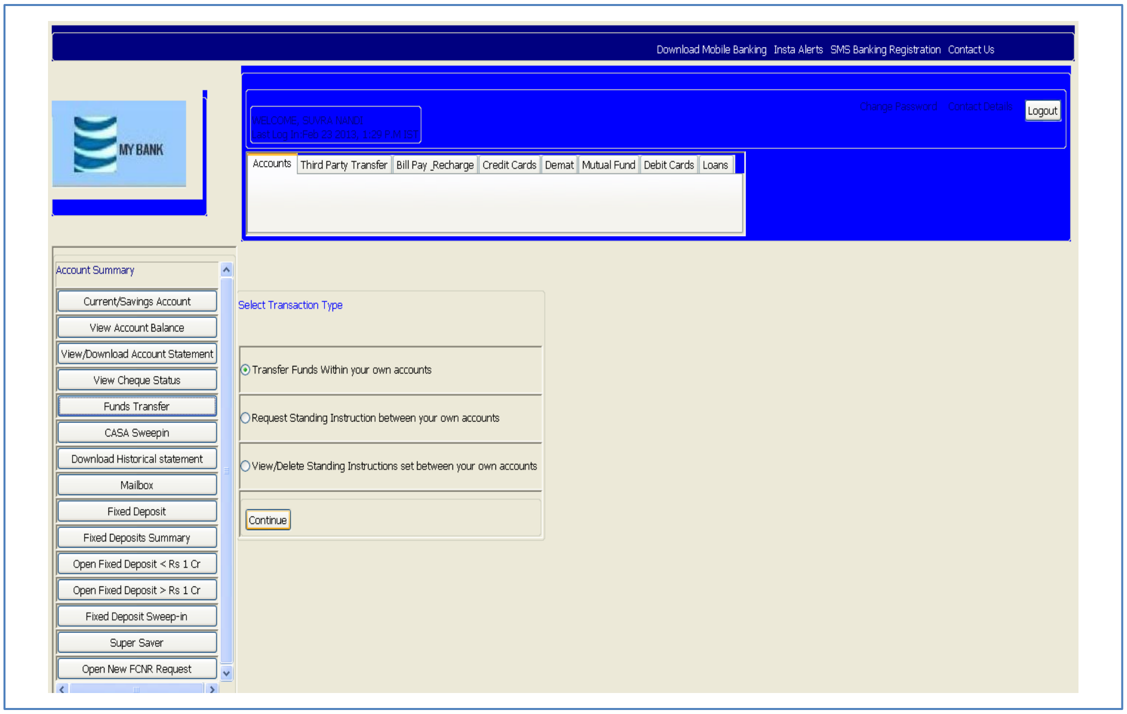


Figure 16: SWT Implementation Screen - Funds Transfer - Transaction Type

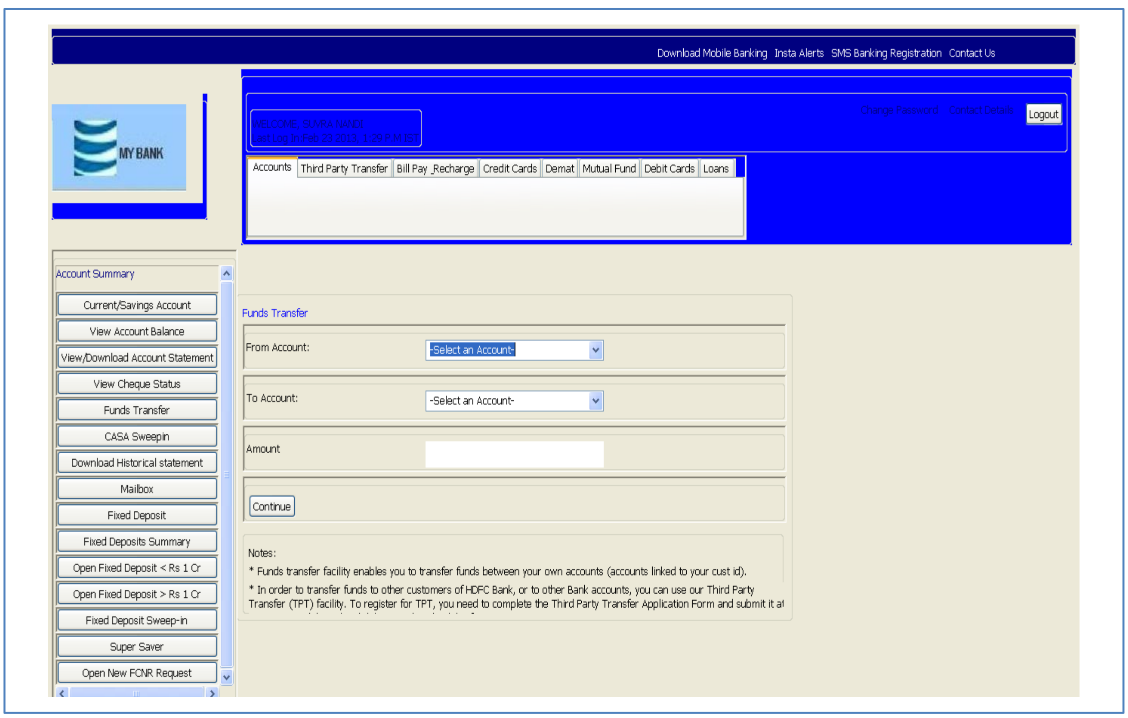


Figure 17: SWT Implementation Screen - Funds Transfer

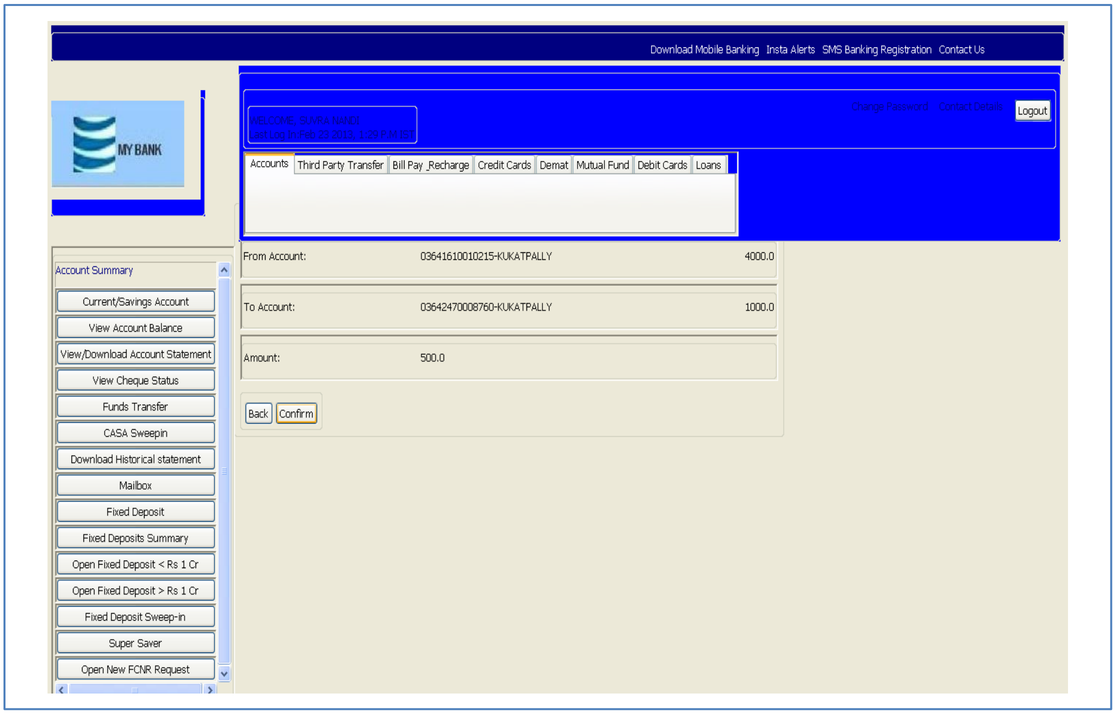


Figure 18: SWT Implementation Screen - Funds Transfer – Confirm

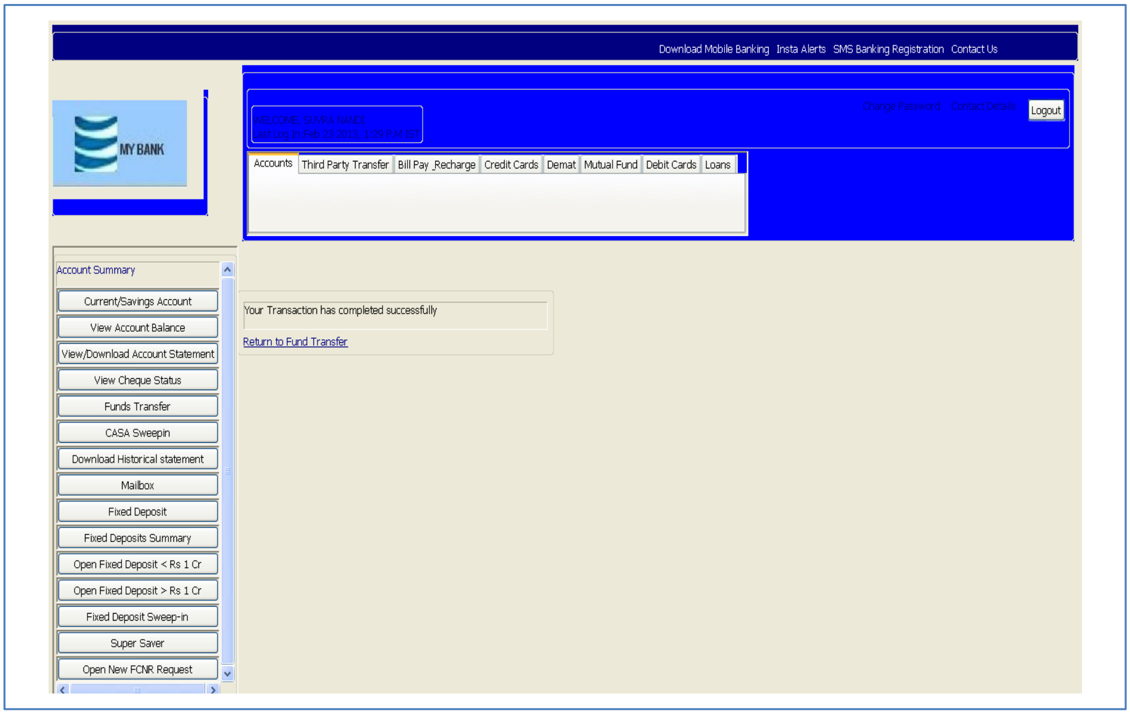


Figure 19: SWT Implementation Screen - Transaction Success Page

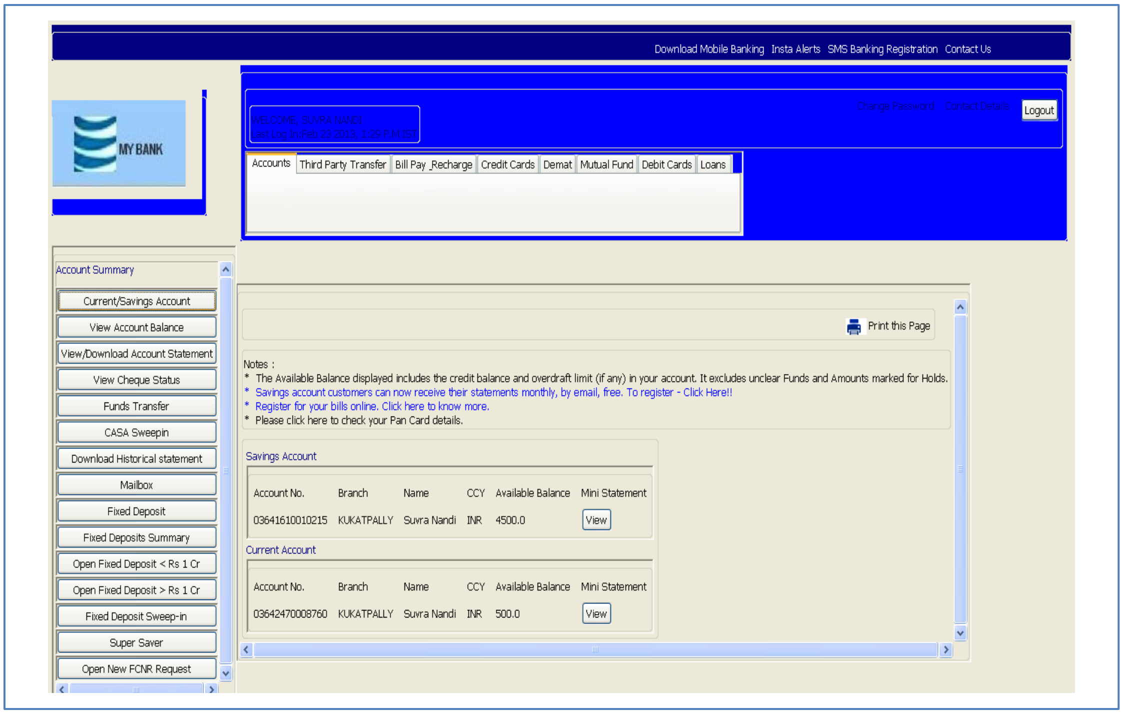


Figure 20: SWT Implementation Screen - Account Summary after Funds Transfer

2.4.5 Evaluation through Implementation Experience

SWT is an open source widget toolkit for Java designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented. SWT library has been developed by IBM as a part of the Eclipse platform.

a) Implementation Notes

While implementing Bank Application's pages by use of SWT:

- RowLayout, FillLayout and GridLayout were used heavily
- FillLayout has the flexibility of spanning the entire screen with window resize.
- Layout data objects like GridData, RowData, FormData provides flexibility to position individual component separately on a single layout
- FormLayout offers flexibility to control over the sizing behavior of each of the four sides of a widget. This is a powerful feature of SWT Layouts
- Familiar short cuts, keyboard navigation etc., get automatically available to user in SWT. For example user can resize a SWT table control (on Windows) to re-adjust the columns according to the data, just by double

clicking between the columns. This functionality comes out of the box, as it is provided by MFC, on which SWT is written.

- Page Navigation support is available only for Browser widget. Browser address needs to be set using setUrl method. Back and Forth navigation is inbuilt.
- Implementation of Scrolling is somewhat difficult and didn't work properly in implemented solution
- As SWT toolkit is unique for each platform, platform specific skinning and LAF can be achieved by separately importing native libraries.

b) When SWT should be used?

Developer may choose SWT as a GUI framework when they need:

- Tight integration with the native platform

If one wants to take advantage of the Java language but needs to deploy only on Windows (or some of the few other supported platforms), SWT (and the Eclipse platform) is a better choice than Swing.

- Cross-Platform development is not needed
- MVC architectural pattern
- Java Coding
- Easy Learning Curve

SWT is easier than SWING in terms of coding and learning, when complex UI is not required.

2.5 Study outcome of JSF Framework

2.5.1 Layout Support in JSF

JSF framework (Reference[3]) uses XHTML as its front-end language. Layouts in XHTML can be achieved in number of different ways:

- Using Basic JSF Layout Tags
- Using JSF Facelets Templating
- Using HTML Layout Tag Library
- Using JSF Open source component libraries, like PrimeFaces, RichFaces etc.

a) Basic JSF Layout Tags

JSF provides a standard HTML tag library. These tags get rendered into corresponding html output.

For these tags we need to use the following namespaces of URI in html node.

```
<html

xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html">
```

Following are important Basic Tags in JSF 2.0:

S.N.	Tag & Description
1	*h:inputText* Renders a HTML input of type="text", text box.
2	*h:inputSecret* Renders a HTML input of type="password", text box.
3	*h:inputTextarea* Renders a HTML textarea field.
4	*h:inputHidden* Renders a HTML input of type="hidden".
5	*h:selectBooleanCheckbox* Renders a single HTML check box.

6	*h:selectManyCheckbox* Renders a group of HTML check boxes.
7	*h:selectOneRadio* Renders a single HTML radio button.
8	*h:selectOneListbox* Renders a HTML single list box.
9	*h:selectManyListbox* Renders a HTML multiple list box.
10	*h:selectOneMenu* Renders a HTML combo box.
11	*h:outputText* Renders a HTML text.
12	*h:outputFormat* Renders a HTML text. It accepts parameters.
13	*h:graphicImage* Renders an image.
14	*h:outputStylesheet* Includes a CSS style sheet in HTML output.
15	*h:outputScript* Includes a script in HTML output.
16	*h:commandButton* Renders a HTML input of type="submit" button.
17	*h:Link* Renders a HTML anchor.
18	*h:commandLink* Renders a HTML anchor.
19	*h:outputLink* Renders a HTML anchor.

20	*h:panelGrid* Renders an HTML Table in form of grid.
21	*h:message* Renders message for a JSF UI Component.
22	*h:messages* Renders all message for JSF UI Components.
23	*f:param* Pass parameters to JSF UI Component.
24	*f:attribute* Pass attribute to a JSF UI Component.
25	*f:setPropertyActionListener* Sets value of a managed bean's property

b) JSF Facelets Templating

Standard Syntax:

```
<%@ taglib prefix="ui" uri="http://xmlns.jcp.org/jsf/facelets" %>
```

XML Syntax:

```
<anyxmlelement xmlns:ui="http://xmlns.jcp.org/jsf/facelets" />
```

The tags in this library add templating, a powerful view composition technique to JSF. Four Facelets Tags are mainly used to build a page from a template:

- ui:insert

Used in template file, it defines content that is going to replace by the file that load the template. The content can be replace with “ui:define” tag.

- ui:define

Defines content that is inserted into template with a matching “ui:insert” tag.

- ui:include

Similar to JSP's "jsp:include", includes content from another XHTML page.

- ui:composition

If used with "template" attribute, the specified template is loaded, and the children of this tag defines the template layout; Otherwise, it's a group of elements, that can be inserted somewhere. In addition, JSF removes all tags "outside" of "ui:composition" tag.

A Templating Example

First, we define a template:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets"

<head>

<link href="styles.css" rel="stylesheet" type="text/css"/>

<title><ui:insert name="title">Default Title</ui:insert></title>

</head>

<body>

<ui:debug/>

<div class="heading">

<ui:insert name="heading"/>

</div>

<div class="content">

<ui:insert name="content"/>

</div>

</body></html>
```

In the preceding listing, we've defined a layout, also known as a template. That template uses the ui:insert tag to insert pieces of a page —namely, title, heading, and content— defined in a composition. Notice that on line 8, we define a default title, in case one isn't provided by the composition. Also note that on line 12 we have the ui:debug tag, which lets the user activate a popup window with debugging information by typing CTRL + Shift + d.

The title, heading, and content pieces of the page referenced in the template are defined in a separate XHTML file in a composition, like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<body>

<ui:composition template="/layout.xhtml">

<ui:define name="title">A List of Contacts</ui:define>

<ui:define name="heading">Contacts</ui:define>

<ui:define name="content">

<ui:include src="contactsTable.xhtml" />

</ui:define>

</ui:composition>

</body></html>
```

At runtime, JSF synthesizes the two previous XHTML pages to create a single JSF view by inserting the pieces defined in the composition into the template (that template is layout.xhtml, which is the first listing above). JSF also disregards everything outside of the composition tag so that we don't wind up with two body elements in the view. Also, note that we use the ui:include tag on line 14 to include content (which happens to be a table) from another XHTML page, so that we can reuse that table in other views.

So why do we have two XHTML pages to define a single view? Why not simply take the pieces and manually insert them into the layout so that we have only a single XHTML page? The answer is simple: we have separated layout from the content so that we can reuse that layout among multiple compositions. For example, now we can define another composition that uses the same layout:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<body>

<ui:composition template="/layout.xhtml">

<ui:define name="title">Create a Contact</ui:define>

<ui:define name="heading">Create Contact</ui:define>

<ui:define name="content">

<ui:include src="createContactForm.xhtml"/>

</ui:define>

</ui:composition>

</body>

</html>
```

By encapsulating the layout, we can reuse that layout among multiple compositions. Just like ui:include lets us encapsulate and reuse content, JSF compositions let us encapsulate and reuse layout, so that changes to a single layout can affect multiple views.

c) HTML Layout Tag Library

The HTML Layout Tag Library contains components for almost all of the standard HTML tags and attributes based on the HTML 4.01 specification. The excluded tags for the library are the interaction tags, such as form, input, select, and textarea. The library allows you to use old-fashioned HTML layout tags mixed in with JSF components without having to use the f:verbatim tag.

This resolves a problem developers face who begin working with standard JSF libraries and then notice that some features are missing that they had used previously in working with other JSP-centric technologies such as Struts. The stumbling block is that most JSF tags will not allow you to insert a pure HTML tag as a child node. One way around this is to use the f:verbatim tag, but this tag does not allow JSF components inside (as a descendent of the f:verbatim element). This has the unfortunate result of discouraging developers from using JSF tags that provide layout such as h:panelGrid, h:panelGroup, h:column. The HTML Layout Tag Library eliminates this problem, easing the transition to JSF for newbies.

The library contains tags that are rendered into their HTML analogs. You use the tags in the library as you would their HTML analogs (per the W3C documentation) with the following differences:

- Each library tag contains a rendered attribute. If it is set to false, the tag and its children are not rendered
- If the library tag contains an id attribute, the standard JSF convention for id attributes is applied. The id's value will be replaced with the component clientId's value
- If you want to keep the same value for the id in the rendered result, use an sid attribute instead
- Each tag has a binding attribute. So, you can use the component binding. The component class name for each tag is built based on tag name using the com.exadel.htmLib.components.UI<TagName> pattern. Where the<TagName> is a name of the tag starts with a capital letter. Example:
  htm:table - com.exadel.htmLib.components.UITable
  htm:b - com.exadel.htmLib.components.UIB
- Each attribute, except the id attribute, supports value binding. You can use something like #{foo.bar} instead of a static literal value
- Each attribute, except the id attribute, supports jsp scriplets. You can use something like <%=foo%> instead of a static literal value
- The following attribute names are changed due the name convention problem:

  - ✓ class with styleClass
  - ✓ for with forAttribute
  - ✓ char with charAttribute

d) JSF Open Source Component Libraries like PrimeFaces, RichFaces etc.

PrimeFaces Layout

Layout is a border layout panel that can be either applied to a full page or a specific element. Layout can respond to expand, collapse, close and resize events of each layout unit with Ajax listeners.

PrimeFaces provide layout in different ways as follows:

- PrimeFaces Full Page Layout
- PrimeFaces Element Layout
- PrimeFaces Event Listeners
- PrimeFaces Nested Layout
- PrimeFaces Complex Layout
- PrimeFaces Mailbox

PrimeFaces Full Page Layout

This fullPage layout consists of five different layoutUnits which are resizable and closable by default.

```
<p:layout fullPage="true">

<p:layoutUnit position="north" size="100" header="Top" resizable="true" closable="t
rue" collapsible="true">

<h:outputText value="North unit content." />

</p:layoutUnit>

<p:layoutUnit position="south" size="100" header="Bottom" resizable="true" closabl
e="true" collapsible="true">

<h:outputText value="South unit content." />

</p:layoutUnit>

<p:layoutUnit position="west" size="200" header="Left" resizable="true" closable="tr
ue" collapsible="true">

<h:outputText value="West unit content." />

</p:layoutUnit>

<p:layoutUnit position="east" size="200" header="Right" resizable="true" closable="t
rue" collapsible="true" effect="drop">

<h:outputText value="Right unit content." />

</p:layoutUnit>

<p:layoutUnit position="center">

<h:form>

This fullPage layout consists of five different layoutUnits which are resizable and clos
able by default.

</h:form>

</p:layoutUnit>

</p:layout>
```

PrimeFaces Element Layout

Layouts can also be created at element level. Example below is a simple split panel implementation.

```
<p:layout style="min-width:400px;min-height:200px;" id="layout">

<p:layoutUnit position="west" resizable="true" size="100" minSize="40" maxSize="20
0">    West

</p:layoutUnit>

<p:layoutUnit position="center">

Center

</p:layoutUnit>

</p:layout>
```

PrimeFaces Event Listeners

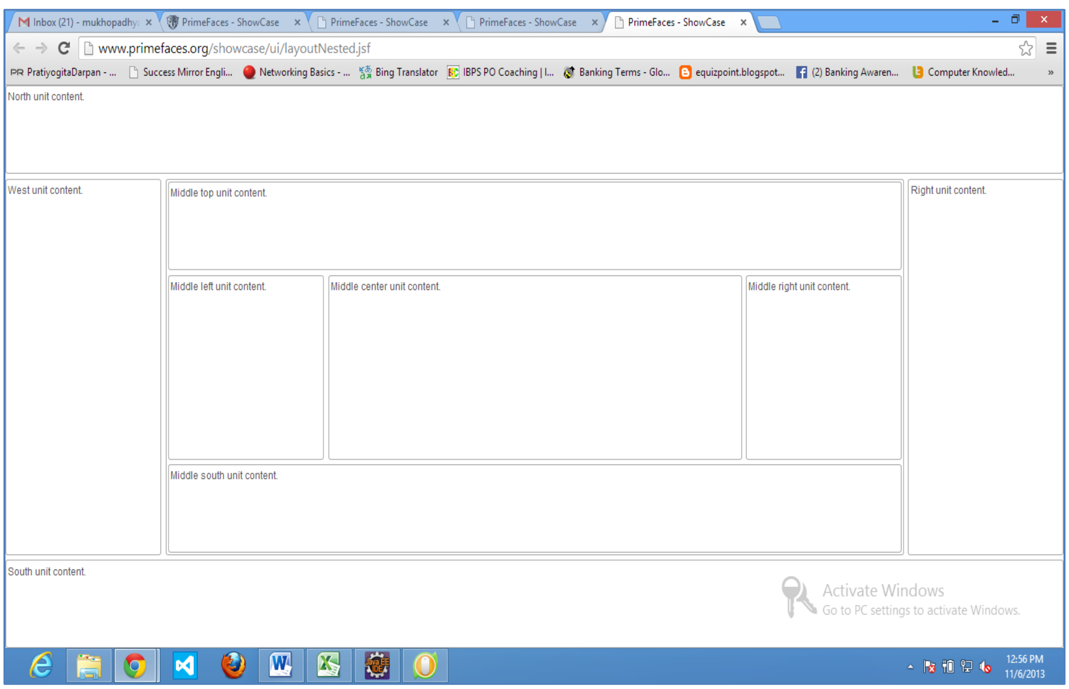
Layout responds to toggle, close and resize events by invoking server side listeners.

```
<p:layout fullPage="true">

<p:ajax event="toggle" listener="#{layoutBean.handleToggle}" update="msg" />

<p:ajax event="close" listener="#{layoutBean.handleClose}" update="msg" />

<p:layoutUnit position="north" size="100" header="Top" resizable="true" closable="true"
collapsible="true">

<h:outputText value="North unit content." />

</p:layoutUnit>

<p:layoutUnit position="south" size="100" header="Bottom" resizable="true" closable="tr
ue" collapsible="true">

<h:outputText value="South unit content." />

</p:layoutUnit>

<p:layoutUnit position="west" size="200" header="Left" resizable="true" closable="true" c
```

```
ollapsible="true">

<h:outputText value="West unit content." />

</p:layoutUnit>

<p:layoutUnit position="east" size="200" header="Right" resizable="true" closable="true" collapsible="true">

<h:outputText value="Right unit content." />

</p:layoutUnit>

<p:layoutUnit position="center">

//content

</p:layoutUnit>

</p:layout>
```

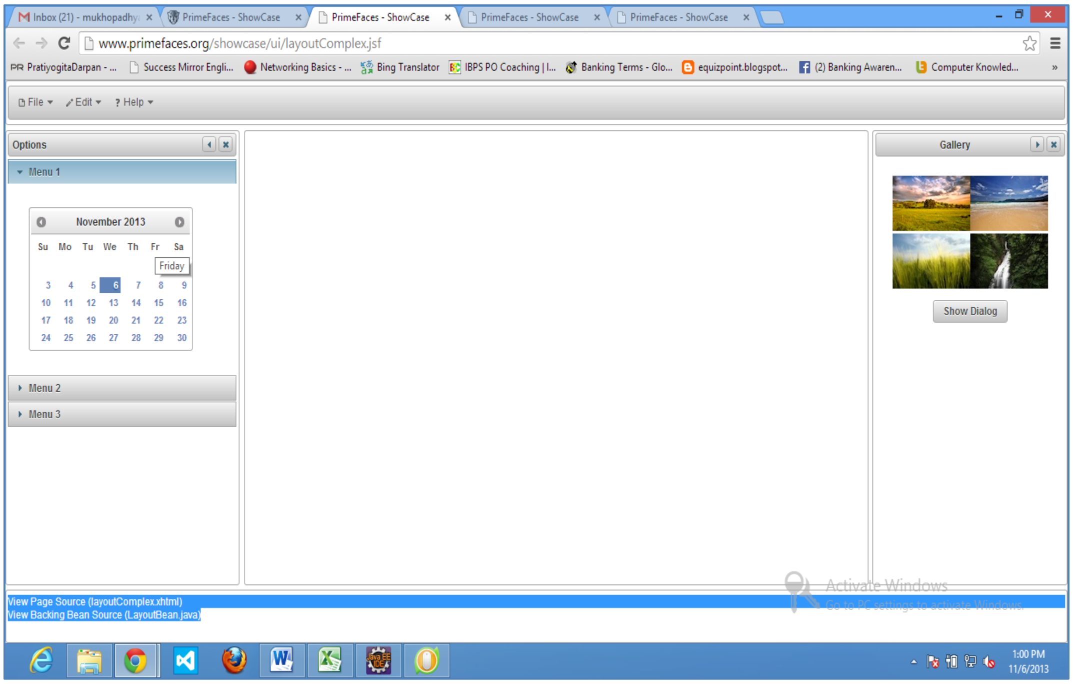
PrimeFaces Nested Layout

Primefaces can build Nested Layouts with its layoutUnit tags as displayed below :



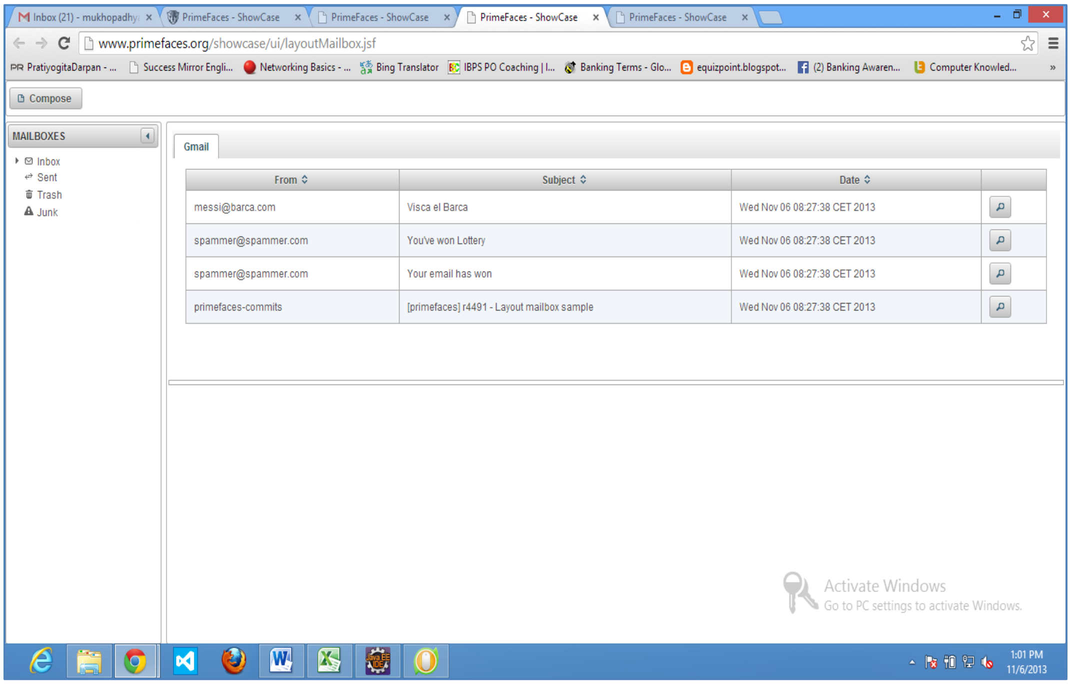
PrimeFaces Complex Layout

Complex Layouts with Elegant look can be built using PrimeFaces. A sample is displayed below :



PrimeFaces Mailbox

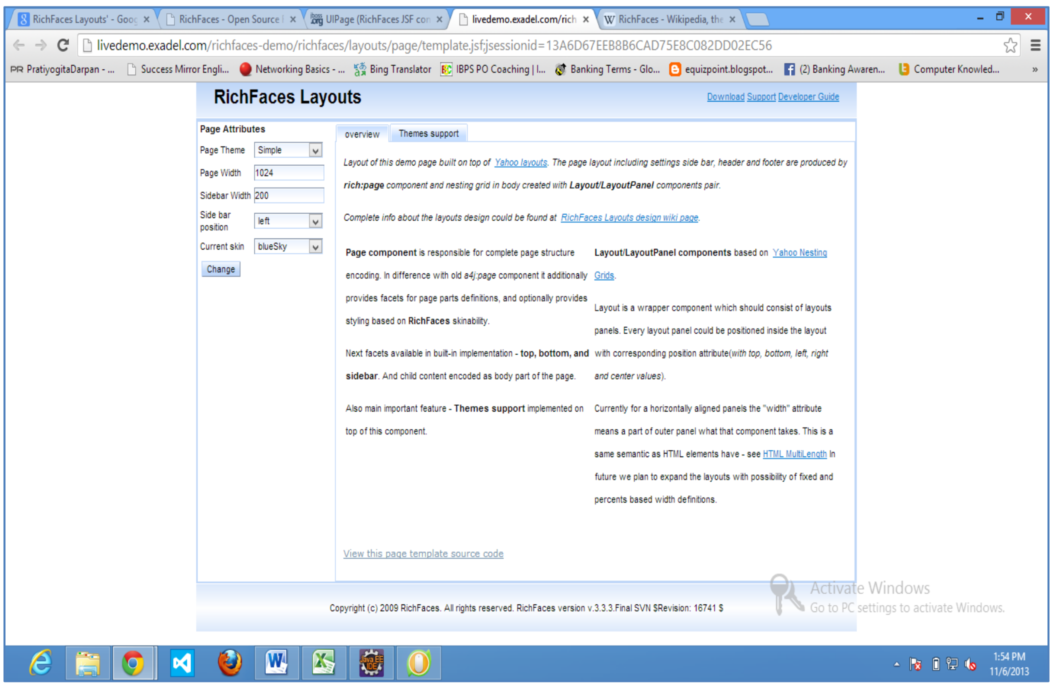
Mailbox Layouts can also be created using PrimeFaces :



RichFaces Layout

RichFaces component library of JSF provides simple Layouts based on layout panel position: top, left, right and center. The <rich:layoutPanel> tag should be enclosed within a <rich:layout> tag.

Below is a sample screen built with RichPanel Layout and the underlying coding:




```
<rich:page xmlns="http://www.w3.org/1999/xhtml"

xmlns:ui="http://java.sun.com/jsf/facelets"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:f="http://java.sun.com/jsf/core"

xmlns:rich="http://richfaces.org/rich"

xmlns:a4j="http://richfaces.org/a4j" markupType="xhtml"

contentType="text/html" theme="#{layoutBean.theme}"

width="#{layoutBean.width}" sidebarWidth="#{layoutBean.sidebarWidth}"

sidebarPosition="#{layoutBean.position}">

<f:facet name="sidebar">
```

```
<ui:include src="/richfaces/layouts/page/includes/sidebar.xhtml" />

</f:facet>

<f:facet name="header">

<h:panelGroup>

<table width="100%"><tbody><tr>

<td><h2>RichFaces Layouts</h2></td>

<td align="right">

<h:outputLink
value="http://labs.jboss.com/portal/jbossrichfaces/downloads">Download</h:outp
utLink> 

<h:outputLink
value="http://jboss.com/index.html?module=bb&amp;op=viewforum&amp;f=261">
Support</h:outputLink> 

<h:outputLink value="http://labs.jboss.com/file-
access/default/members/jbossrichfaces/freezone/docs/devguide/en/html/index.ht
ml">Developer Guide</h:outputLink>

</td>

</tr></tbody></table>

</h:panelGroup>

</f:facet>

<f:facet name="footer">

<h:outputText

value="Copyright (c) 2009 RichFaces. All rights reserved. RichFaces version
#{environment.version}" />

</f:facet>

<rich:tabPanel switchType="client">

<rich:tab label="overview">

<rich:layout>
```

```
<rich:layoutPanel position="top" width="100%">
<ui:include src="/richfaces/layouts/page/includes/gridtop.xhtml" />
</rich:layoutPanel>
<rich:layoutPanel position="left" width="50%" style="padding:3px;">
<ui:include src="/richfaces/layouts/page/includes/gridleft.xhtml" />
</rich:layoutPanel>
<rich:layoutPanel position="center" width="50%" style="padding:3px;">
<ui:include src="/richfaces/layouts/page/includes/gridcenter.xhtml" />
</rich:layoutPanel>
<rich:layoutPanel position="bottom">
<ui:include src="/templates/include/sourceview.xhtml">
<ui:param name="sourcepath" value="/richfaces/layouts/page/template.xhtml"/>
<ui:param name="openlabel" value="View this page template source code"/>
</ui:include>
<br/>
</rich:layoutPanel>
</rich:layout>
</rich:tab>
<rich:tab label="Themes support">
<ui:include src="/richfaces/layouts/page/includes/themes.xhtml" />
</rich:tab>
</rich:tabPanel>
</rich:page>
```

2.5.2 Navigation Support in JSF

JSF provides four kinds of navigation support:

- Implicit or Auto Navigation : JSF 2.0 provides implicit navigation as well in which there is no need to define navigation rules as such.
- Auto Navigation in Managed Bean: Navigation rules can be defined in managed beans.
- Conditional Navigation: Navigation rules can contain conditions based on which resulted view can be shown
- Configuration Rule-Based Navigation: Navigation rules can be defined in JSF configuration file named faces-config.xml.

a) Implicit Navigation

JSF 2.0 provides auto view page resolver mechanism named implicit navigation. In this case developer only needs to put view name in action attribute and JSF will search the correct view page automatically in the deployed application.

```
<h:form>

<h3>Using JSF outcome</h3>

<h:commandButton action="page2" value="Page2" />

</h:form>
```

b) Auto Navigation in Managed Bean

In this approach developer need to define a method in managed bean to return a view name.

```
@ManagedBean(name = "navigationController", eager = true)

@RequestScoped

public class NavigationController implements Serializable {

public String moveToPage1(){

return "page1";

} }
```

Also, developer needs to put view name in action attribute of any JSF UI Component using managed bean.

```
<h:form>

<h3>Using Managed Bean</h3>

<h:commandButton action="#{navigationController.moveToPage1}"

value="Page1" />

</h:form>
```

c) *Conditional Navigation*

This is also a Managed Bean approach as above, where conditioning is done based on parameters passed from the JSF. Below is the sample Managed Bean code:

```
@ManagedBean(name = "navigationController", eager = true)

@RequestScoped

public class NavigationController implements Serializable {

//this managed property will read value from request parameter pageId

@ManagedProperty(value="#{param.pageId}")

private String pageId;

//conditional navigation based on pageId

//if pageId is 1 show page1.xhtml,

//if pageId is 2 show page2.xhtml

//else show home.xhtml

public String showPage(){

if(pageId == null){

return "home";

}
```

```
if(pageId.equals("1")){

return "page1";

}else if(pageId.equals("2")){

return "page2";

}else{

return "home";

}

}

}
```

The JSF code looks like below:

```
<h:form>

<h:commandLink action="#{navigationController.showPage}"
value="Page1">

<f:param name="pageId" value="1" />

</h:commandLink>

<h:commandLink action="#{navigationController.showPage}"
value="Page2">

<f:param name="pageId" value="2" />

</h:commandLink>

<h:commandLink action="#{navigationController.showPage}"
value="Home">

<f:param name="pageId" value="3" />

</h:commandLink>

</h:form>
```

d) Configuration Rule-Based Navigation

JSF provides navigation resolution option even if managed bean different methods returns same view name.

Below is the sample code snippet from Managed Bean :

```
public String processPage1(){

return "page";

}

public String processPage2(){

return "page";

}
```

To resolve views, developer needs to define following navigation rule in faces-config.xml:

```
<navigation-rule>

<from-view-id>home.xhtml</from-view-id>

<navigation-case>

<from-action>#{navigationController.processPage1}</from-action>

<from-outcome>page</from-outcome>

<to-view-id>page1.jsf</to-view-id>

</navigation-case>

<navigation-case>

<from-action>#{navigationController.processPage2}</from-action>

<from-outcome>page</from-outcome>

<to-view-id>page2.jsf</to-view-id>

</navigation-case></navigation-rule>
```

The JSF code looks like below :

```
<h:form>

<h:commandLink action="#{navigationController.processPage1}" value="Page1">

</h:commandLink>

<h:commandLink action="#{navigationController.processPage2}" value="Page2">

</h:commandLink>

</h:form>
```

2.5.3 Theme Support in JSF

Basic JSF doesn't provide any specific theme support. However, there are few JSF libraries available, like RichFaces which provide very good support of Skinning and Themes in JSF applications.

Every RichFaces component gives the support for skinning and it means that just by changing the skin, users change the look for all of the components. That's very good for giving an application a consistent look and not repeating the same CSS values for each component every time.

RichFaces still uses CSS, but it also enhances it in order to make it simpler to manage and maintain.

RichFaces skins come in two flavors :

- Built-In Skins
- Pluggable Skins

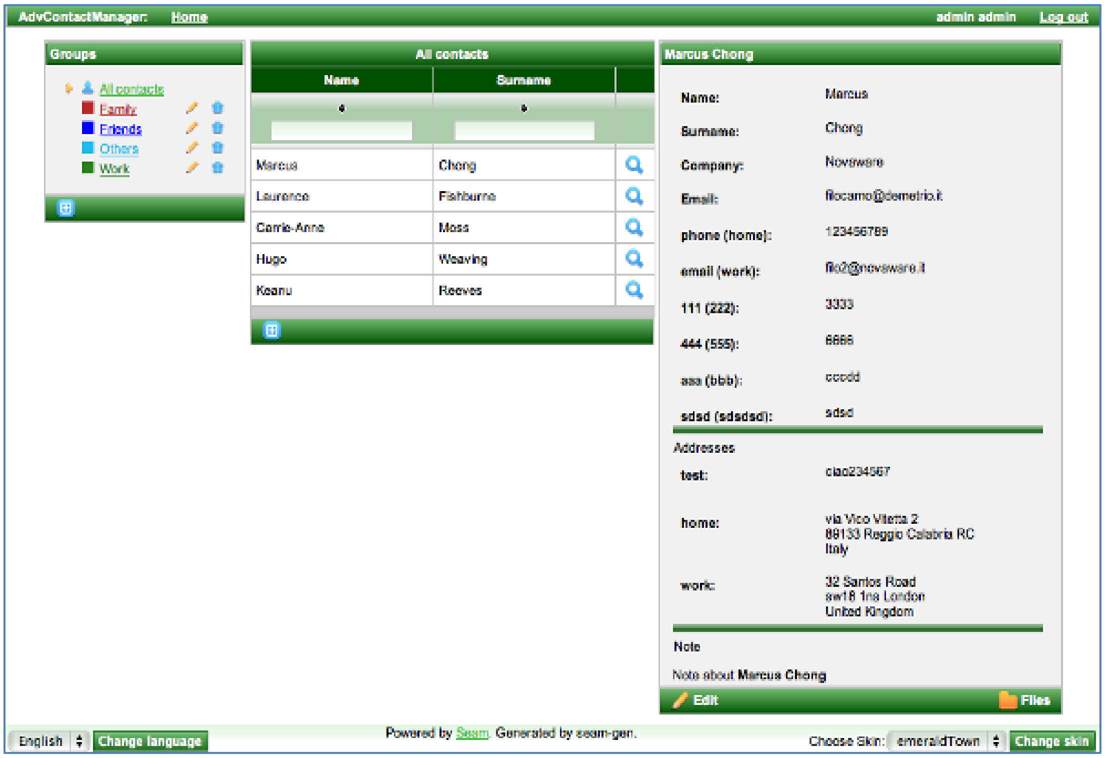
a) Built-In Skins

There are certain standard skins which are built in RichFaces Library, and developers can readily use it just by calling the names. He is a list of all the built-in skins supported:

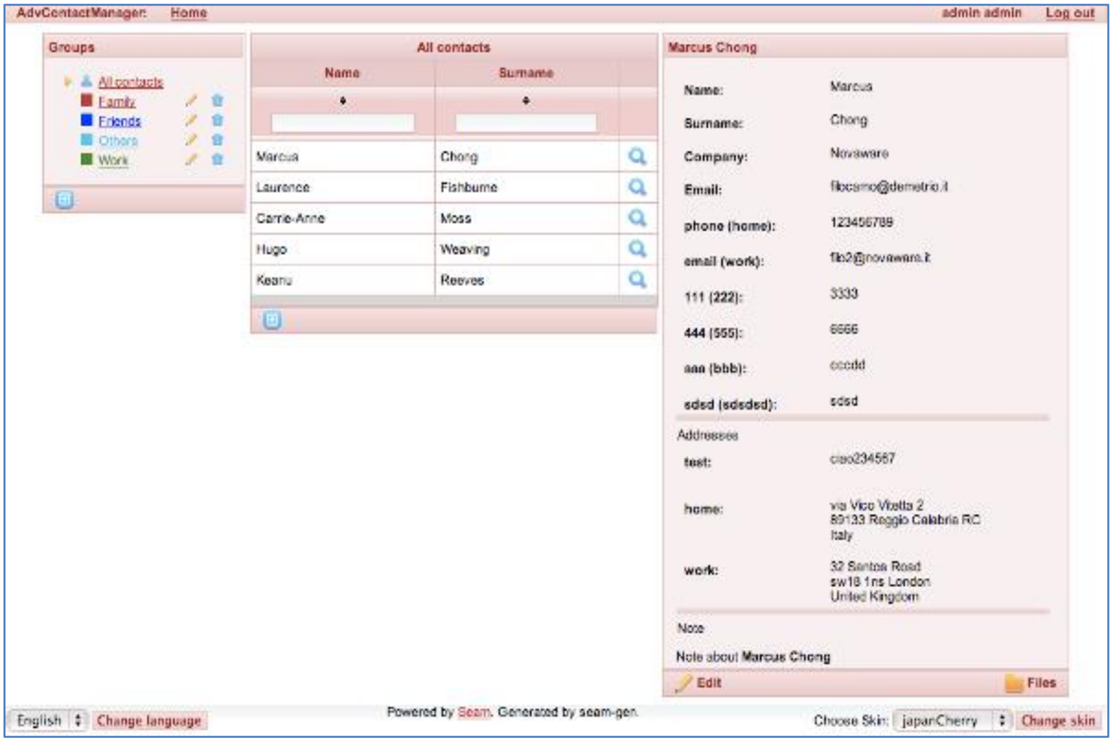
- Plain
- EmeraldTown
- BlueSky
- Wine

- JapanCherry
- Ruby
- Classic
- DeepMarine

Here is a sample screen using **EmeraldTown** built-in skin :



Skin **JapanCherry** would look as follows:



b) Pluggable Skins

The plug 'n' skin skins are packaged in external jar files that must be added into the project in order to be able to use them. Here is a list of few Pluggable ones :

- Laguna
- DarkX
- GlassX

Both Built-In and Pluggable Skins to be used by the application can be set as context-parameter in the web.xml file:

```
<context-param>

<param-name>org.richfaces.SKIN</param-name>

<param-value>emeraldTown</param-value>

</context-param>
```

Also, a basic skin can be edited in RichFaces by editing the skin ".properties" file kept inside the RichFaces Jar "richfaces-impl-3.x.x.jar", kept inside META-INF directory.

Below is the sample ".properties" file of Richfaces Skin :

```
#Colors

headerBackgroundColor=#005000

headerGradientColor=#70BA70

headerTextColor=#FFFFFF

headerWeightFont=bold

generalBackgroundColor=#f1f1f1

generalTextColor=#000000

generalSizeFont=18px

generalFamilyFont=Arial, Verdana, sans-serif

controlTextColor=#000000

controlBackgroundColor=#ffffff

additionalBackgroundColor=#E2F6E2

shadowBackgroundColor=#000000

shadowOpacity=1

panelBorderColor=#C0C0C0

subBorderColor=#ffffff

tabBackgroundColor=#ADCDAD

tabDisabledTextColor=#67AA67

trimColor=#BBECBB

tipBackgroundColor=#FAE6B0

tipBorderColor=#E5973E

selectControlColor=#FF9409
```

```
generalLinkColor=#43BD43

hoverLinkColor=#FF9409

visitedLinkColor=#43BD43

# Fonts

headerSizeFont=18px

headerFamilyFont=Arial, Verdana, sans-serif

tabSizeFont=11

tabFamilyFont=Arial, Verdana, sans-serif

buttonSizeFont=18

buttonFamilyFont=Arial, Verdana, sans-serif

tableBackgroundColor=#FFFFFF

tableFooterBackgroundColor=#cccccc

tableSubfooterBackgroundColor=#f1f1f1

tableBorderColor=#C0C0C0

tableBorderWidth=2px

#Calendar colors

calendarWeekBackgroundColor=#f5f5f5

calendarHolidaysBackgroundColor=#FFEBDA

calendarHolidaysTextColor=#FF7800

calendarCurrentBackgroundColor=#FF7800

calendarCurrentTextColor=#FFEBDA

calendarSpecBackgroundColor=#E2F6E2

calendarSpecTextColor=#000000

warningColor=#FFE6E6

warningBackgroundColor=#FF0000
```

```
editorBackgroundColor=#F1F1F1

editBackgroundColor=#FEFFDA

#Gradients

gradientType=plain
```

2.5.4 Sample Screens Implemented in JSF

Few pages of a sample Bank Website was built using JSF framework, taking help of different layout components available.

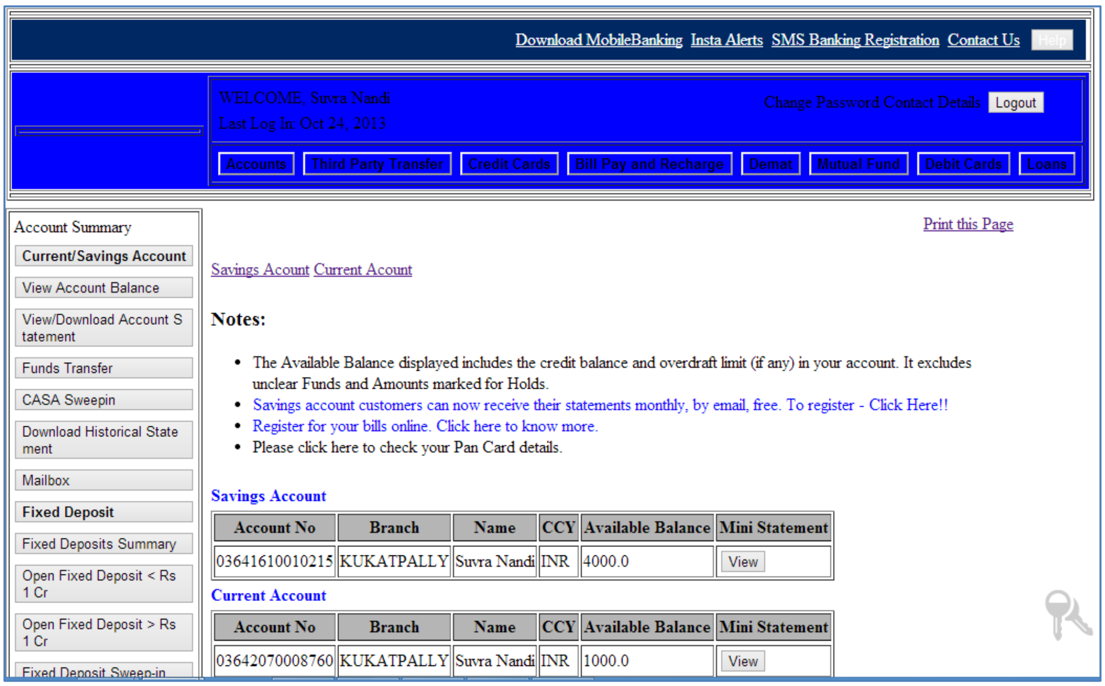


Figure 21: JSF Implementation Screen - Account Summary

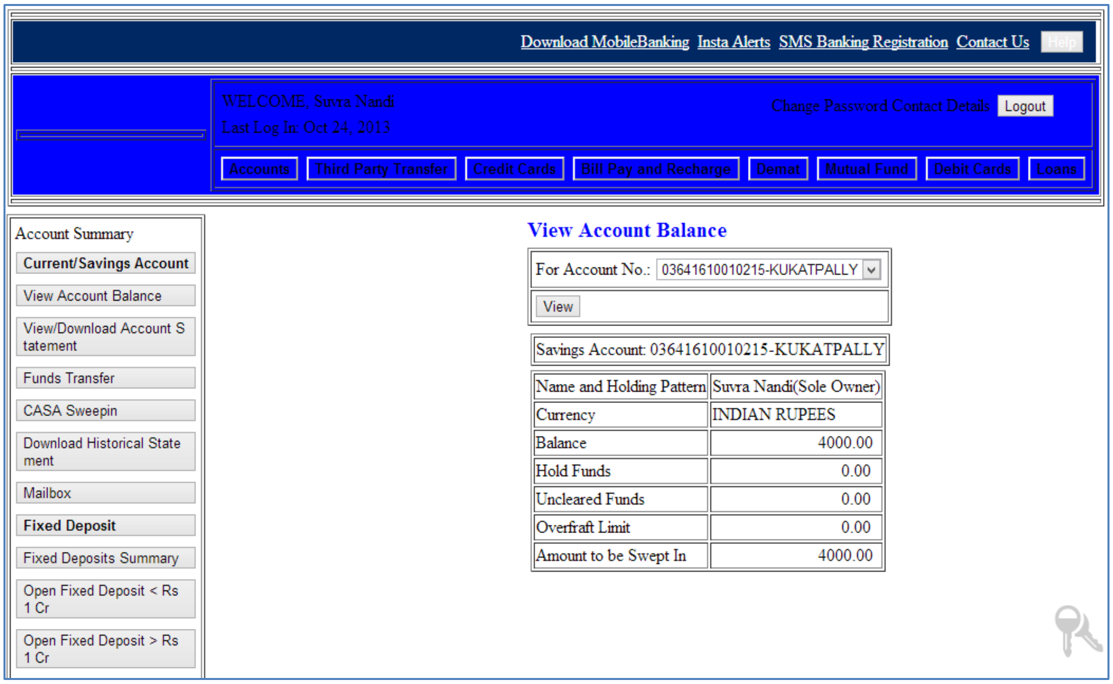


Figure 22: JSF Implementation Screen - View Account Balance - Savings Account

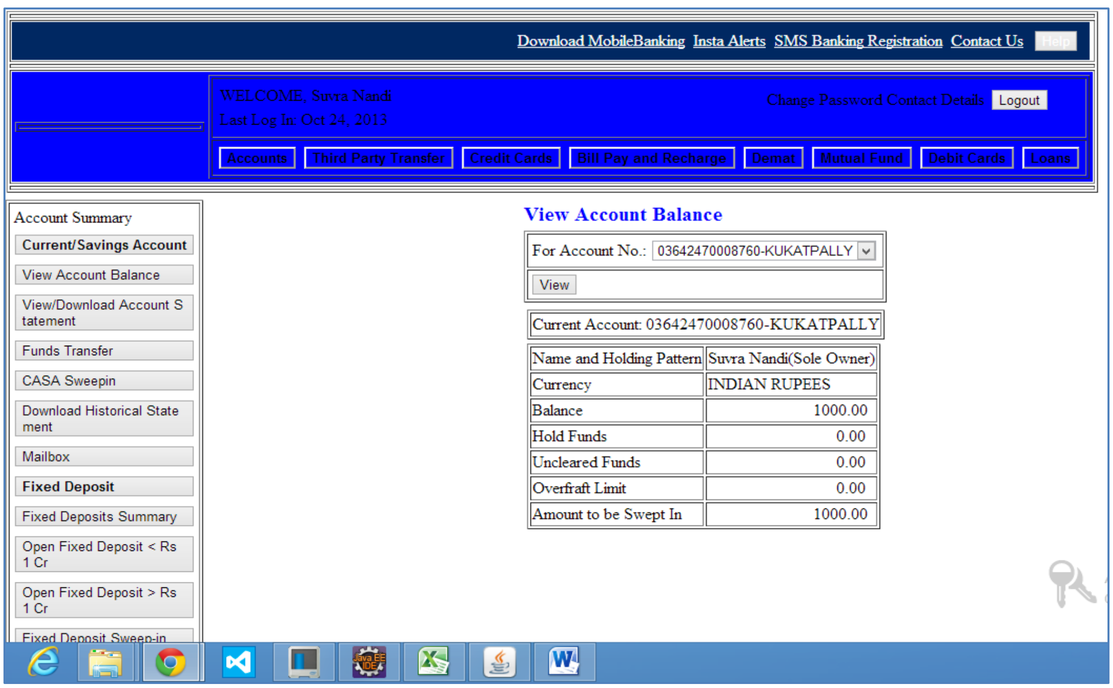


Figure 23: JSF Implementation Screen - View Account Balance - Current Account

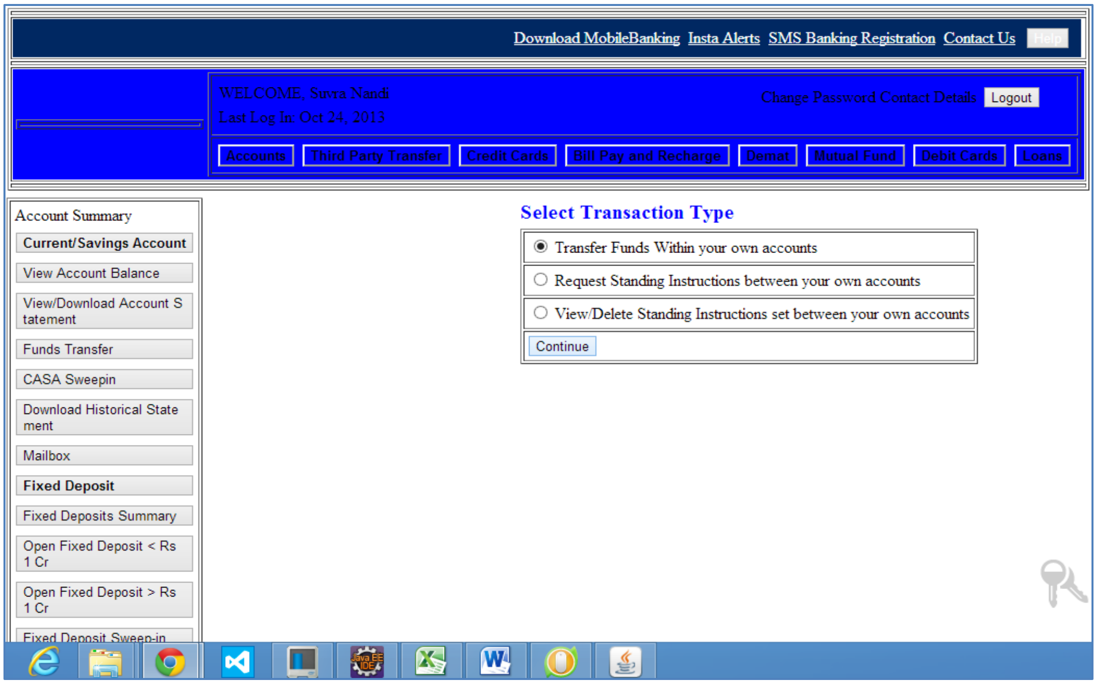


Figure 24: JSF Implementation Screen - Fund Transfer - Transaction Type

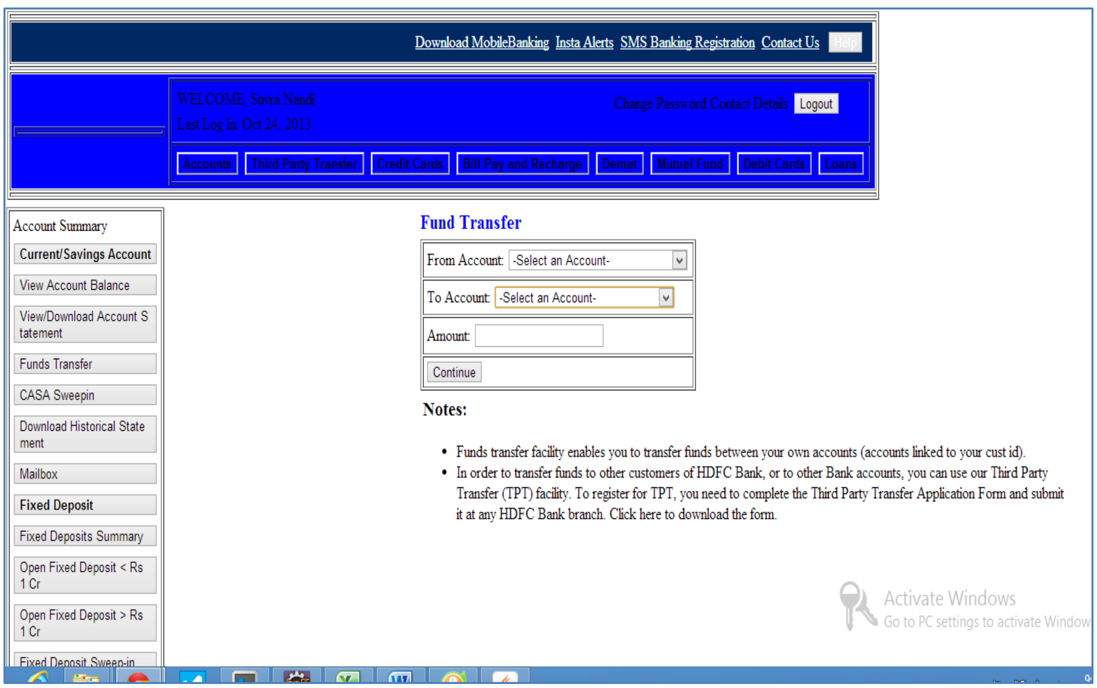


Figure 25: JSF Implementation Screen - Fund Transfer

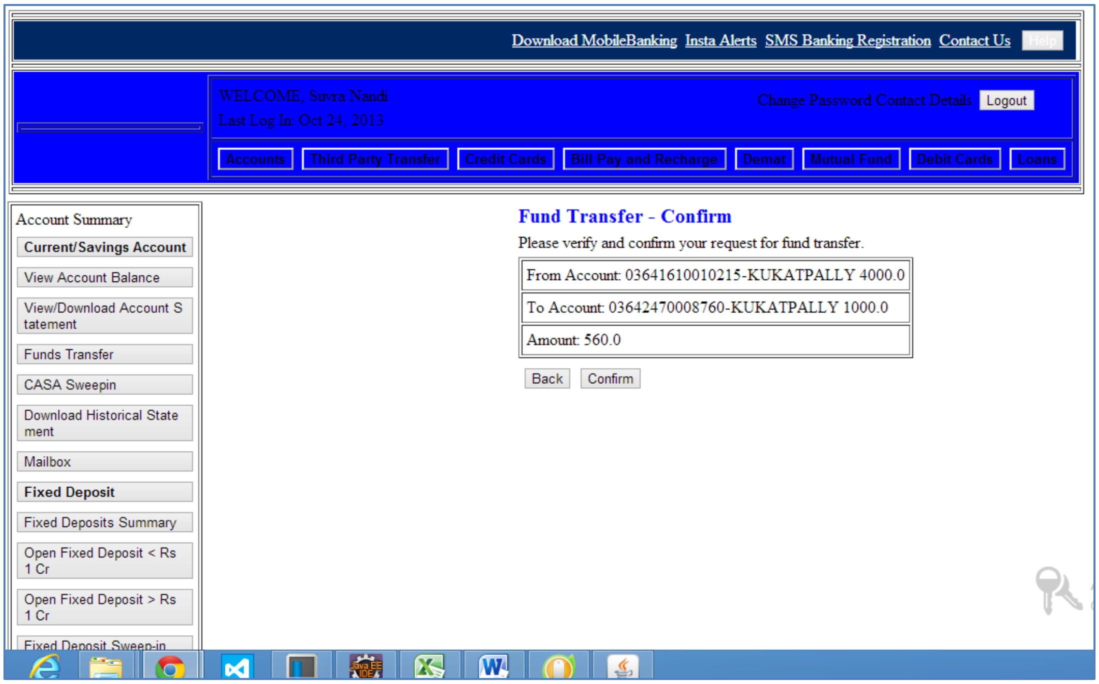


Figure 26: JSF Implementation Screen - Fund Transfer – Confirm

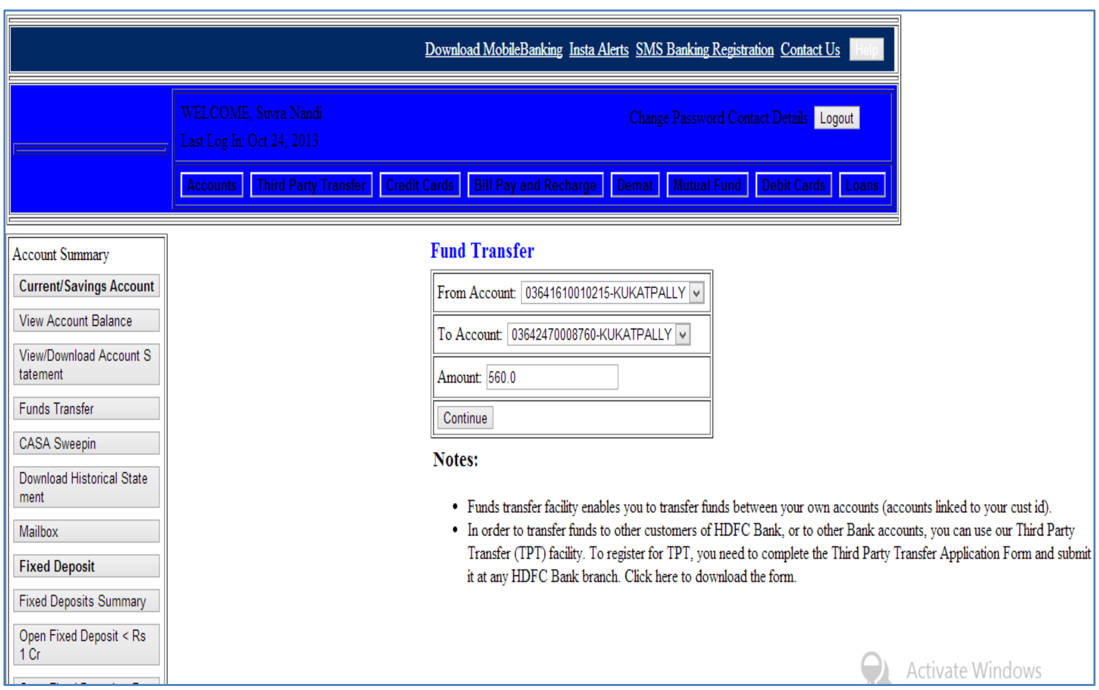


Figure 27: JSF Implementation Screen - Fund Transfer after Back Navigation

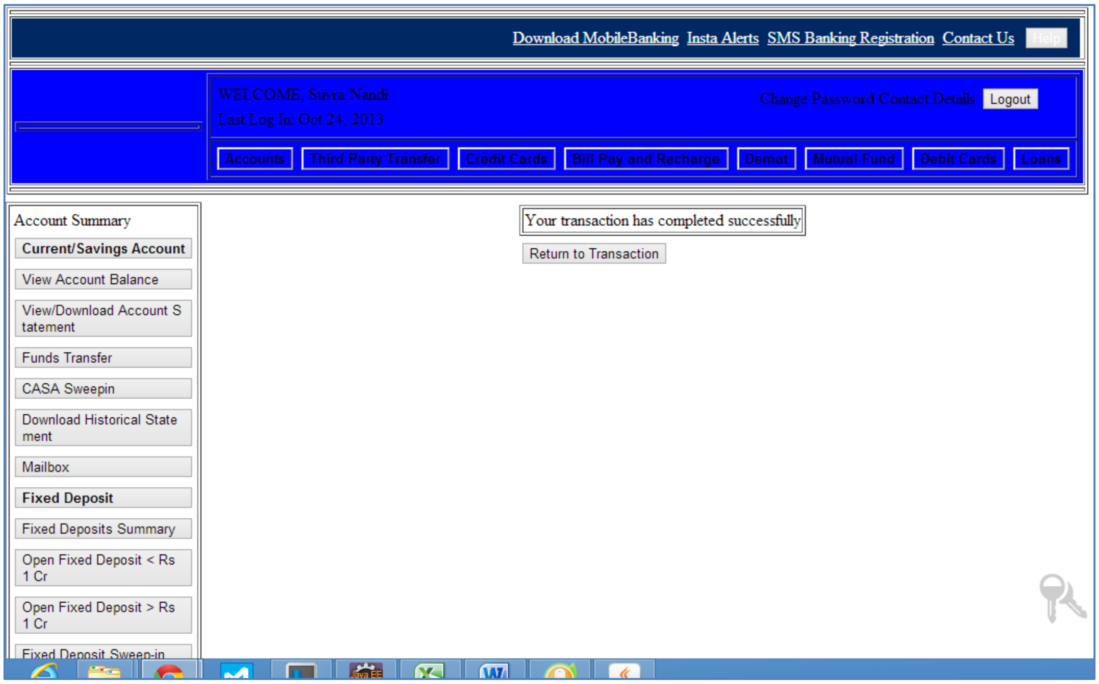


Figure 28: JSF Implementation Screen - Transaction Success Page

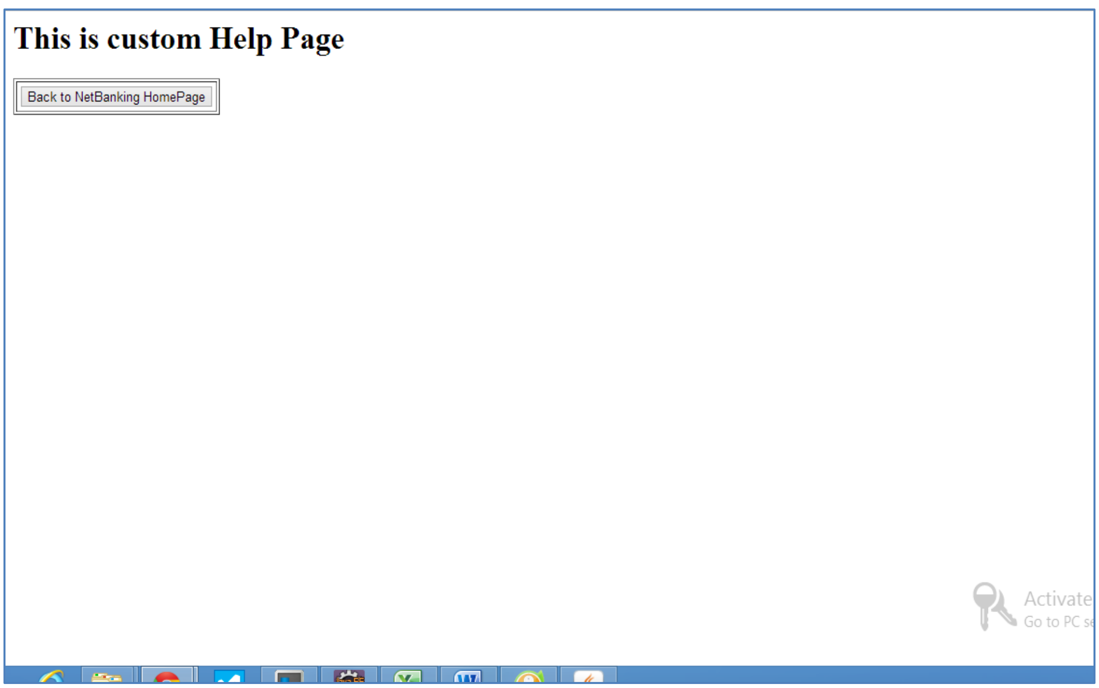


Figure 29: JSF Implementation Screen - Page Navigation Demo from Help Link

2.5.5 Evaluation through Implementation Experience

Java Server Faces (JSF) is a Java specification for building component-based user interfaces for web applications.

Based on a component-driven UI design-model, Java Server Faces uses XML files called view templates or Facelets views. The FacesServlet processes requests, loads the appropriate view template, builds a component tree, processes events, and renders the response (typically in the HTML language) to the client. The state of UI components and other objects of scope interest is saved at the end of each request in a process called stateSaving (note: transient true), and restored upon next creation of that view. Either the client or the server side can save objects and states.

a) Implementation Notes

- Bank Application's sample pages were implemented using JSF Basic Tags <h:panelGrid> and <h:panelGroup>. <h:panelGrid> helped in creating HTML table structures easily.
- JSF Facelets allow for **parameterized templating**. Facelets provide a server-side templating facility that allows composing the actual view from several separate physical pages in a way that maximizes markup code reuse and eliminates redundancy among views. It provides a way to declare a JSF View using standard XHTML syntax
- PrimeFaces and RichFaces component libraries provide elegant set of Layout, Navigation and Skin features
- Basic JSF provides a very good Navigation system, for Page Navigation, depending upon Button Press or Hyperlinks
- Declarative programming leads to some real improvements in how we write code, not just in terms of fewer lines of code, or (potentially) performance, but by writing code at a higher level of abstraction we can focus much more on *what* we want. This way it is better than SWING or SWT
- RichFaces library has good skin support
- Every button or link clicked results in a form post. Form submission for page navigation make complex coding for simple requirement like Cancel button
- There is no skinning support in Basic JSF. CSS can be used to create custom skins

b) When JSF should be used ?

Developers may choose JSF as GUI Development Framework when :

- Declarative Programming is Preferred over Imperative Programming
- Component Based MVC architecture is preferred
- Developer need to replace a repeated group of components by a single component
- Rich User Interface needs to be built, JSF RichFaces can be used

2.6 Study outcome of Windows Store Apps Framework

2.6.1 Layout Support in Windows Store Apps

Windows Store Apps (Reference[4]) use XAML with C++/C#/VB as framework implementation language. The View Definition Language XAML, Layout System supports both Absolute Layout and Dynamic Layout. The details of each layout are covered below:

a) Absolute Layout

In Absolute Layout child elements are arranged in Layout Panels by specifying their exact location in terms of (x, y) coordinates, with respect to their parent element. Absolute Positioning doesn't consider the size of the screen. The XAML framework provides a Canvas control to support absolute positioning, where attached properties Canvas.Left and Canvas.Top need to be set.

b) Dynamic Layout

In a Dynamic Layout, the User Interface appears correctly on various screen resolutions. Developer needs to specify how child elements should be arranged and how they should wrap relative to their parent. To use automatic or proportional sizing, developer must assign special values to Height and Width properties.

Dynamic Layout requires implementation by using StackPanel or Grid elements. StackPanel is the simplest layout element that arranges its child elements into a single line that can be oriented horizontally or vertically. Grid control supports arranging elements in multi-row and multi-column layouts.

The following are the recommended settings for Dynamic Layouts:

- Set the Height and Width of the control or layout to Auto. When these values are used for controls in the Grid layout, the control fills the cell that contains it. Auto sizing is supported for controls and for the Grid and Stack Panel layouts.
- For controls that contain text, remove the Height and Width properties, and set the MinWidth or MinHeight properties. This prevents the text from scaling down to a size that's unreadable.
- To set proportional values for the RowDefinition and ColumnDefinition elements in a Grid layout, use relative Height and Width values. For example, to specify that one column is five times wider than another column, use "*" and "5*" for the Width properties in the ColumnDefinition elements.

Auto and star sizing

Auto sizing is used to allow controls to fit their content, even if the content changes size. Star sizing is used to distribute available space among the rows and columns of a grid by weighted proportions. In XAML, star values are expressed as * (or n*). For example, if you have a grid with four columns, you could set the widths of the columns to the values shown in the following table.

Column_1	*Auto*	*The column will size to fit its content.*
Column_2	*	*After the Auto columns are calculated, the column gets part of the remaining width. Column 2 will be one-half as wide as Column 4.*
Column_3	*Auto*	*The column will size to fit its content.*
Column_4	*2**	*After the Auto columns are calculated, the column gets part of the remaining width. Column 4 will be two times as wide as Column 2.*

Panel controls

The built-in layout panels in the XAML

```
<Rectangle Canvas.Left="200" Canvas.Top="100"
Fill="red" Width="350" Height="350" />
</Canvas>
```

The preceding XAML produces output that is similar to the following illustration.

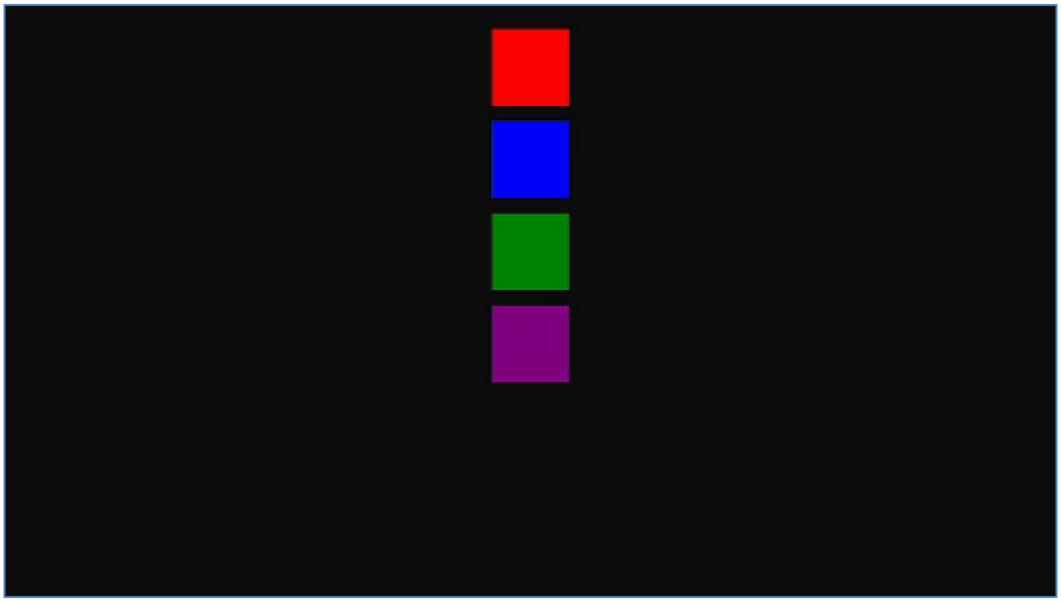
StackPanel

The StackPanel is a simple layout panel that arranges its child elements into a single line that can be oriented horizontally or vertically. Developer can use the Orientation property to specify the direction of the child elements. The default value for the Orientation property is Orientation.Vertical. StackPanel controls are typically used in scenarios where you want to arrange a small subsection of the UI on your page.

The following XAML shows how to create a vertical StackPanel of items.

```
<StackPanel Margin="20">

<Rectangle Fill="Red" Width="100" Height="100" Margin="10" />

<Rectangle Fill="Blue" Width="100" Height="100" Margin="10" />

<Rectangle Fill="Green" Width="100" Height="100" Margin="10" />

<Rectangle Fill="Purple" Width="100" Height="100" Margin="10" />

</StackPanel>
```

The preceding XAML produces output that is similar to the following illustration.



Grid

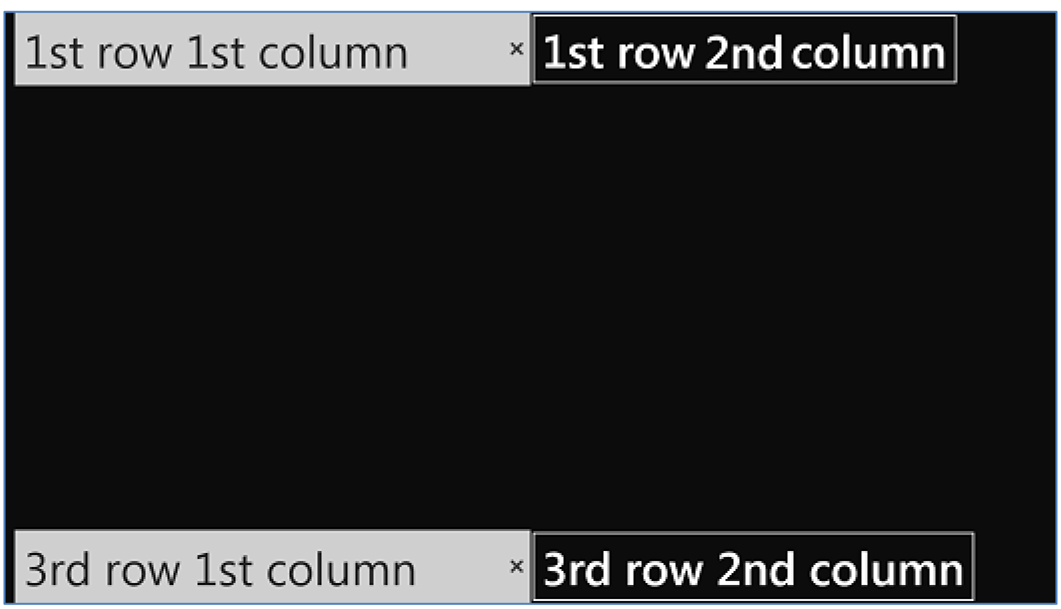
The Grid control supports arranging controls in multi-row and multi-column layouts. Developer can specify a Grid control's row and column definitions by using the RowDefinitions and ColumnDefinitions properties that are declared immediately within the Grid control. Objects can be positioned in specific cells of the Grid by using the Grid.Column and Grid.Row attached properties. Space can be distributed within a column or a row by using Auto or star sizing. Content can span across multiple rows and columns by using the Grid.RowSpan and Grid.ColumnSpan attached properties.

The following XAML shows how to create a Grid with three rows and two columns. The height of the first and third rows is just large enough to contain the text. The height of the second row fills up the rest of the available height. The width of the columns is split equally within the available container width.

```
<Grid Margin="12,0,12,0">

<Grid.RowDefinitions>

<RowDefinition Height="auto" />

<RowDefinition />

<RowDefinition Height="auto" />

</Grid.RowDefinitions>
```

```
<Grid.ColumnDefinitions>

<ColumnDefinition Width="*" />

<ColumnDefinition Width="*" />

</Grid.ColumnDefinitions>

<TextBox Text="1st row 1st column"
FontSize="60" Grid.Column="0"
Grid.Row="0" />

<TextBox Text="3rd row 1st column"
FontSize="60" Grid.Column="0"
Grid.Row="2" />

<Button Content="1st row 2nd column"
FontSize="60" Grid.Column="1"
Grid.Row="0" />

<Button Content="3rd row 2nd column"
FontSize="60" Grid.Column="1"
Grid.Row="2" />

</Grid>
```

The preceding XAML produces output that is similar to the following illustration.



2.6.2 Navigation Support in Windows Store Apps

UI developed using Windows Store App is based on a Frame. The Frame class is primarily responsible for navigation and implements methods such as Navigate, GoBack, and GoForward. Developers use the Navigate method to display content in the Frame.

Class NavigationHelper is an implementation of a page that provides the following important conveniences:

- Event handlers for Navigate, GoBack, and GoForward.
- Mouse and keyboard shortcuts for navigation.
- State management for navigation and process lifetime management.

In addition to providing the implementations described, NavigationHelper also needs to be called from the OnNavigatedTo() and OnNavigatedFrom() event handlers that are implemented on each page. When these events occur, NavigationHelper calls a page-specific implementation of LoadState() and SaveState(). Developers can customize the implementation of these functions on each page. They should be used in place of OnNavigatedTo() and OnNavigatedFrom() respectively.

OnNavigatedTo() is called when the page is about to be displayed in a Frame. When we are navigating to a new page we load state associated with the page. If the page is being restored, the state that was previously saved for the page is restored. LoadState is then called such that each page can react. LoadState has two parameters; the original navigation parameter passed into OnNavigatedTo and the previous page state if it exists.

OnNavigatedFrom() is called when the page is no longer going to be displayed in a Frame. When we are navigating away from a page, we allow the page to save its current state. An empty dictionary is passed into SaveState(). Each page can override SaveState and store objects in the keyed dictionary (string to object). This dictionary is then associated with the page and added to the SessionState that SuspensionManager keeps track of for the given frame.

Navigation Support is provided for these Store App Page Templates :

- Basic Page
- Group Detail Page
- Grouped Items Page
- Item Detail Page

- Items Page
- Split Page
- Hub Page
- Search Results Page

Apart from simple Navigation, Passing information between pages & Caching a Page are also supported in Windows Store App.

2.6.3 Theme Support in Windows Store Apps

Windows Store App supports only two categories of Themes: Light Theme & Dark Theme. Themes can be changed by providing Theme value in RequestedTheme attributes of the Application tag of App.xaml file, as follows:

```
<Application

x:class=”App1.App”

xmlns=”http://schemas.microsoft.com/winfx/ 2006/xaml/presentation”

xmlns:x=”http://schemas.microsoft.com/winfx/ 2006/xaml”

xmlns:local=”using:App1” RequestedTheme=”Light”>
```

Here are common XAML controls under Dark and Light Themes respectively:

2.6.4 Sample Screens Implemented in Windows Store Apps

Few pages of a sample Bank Website was built using WINDOWS STORE APPS framework, taking help of Layout objects.

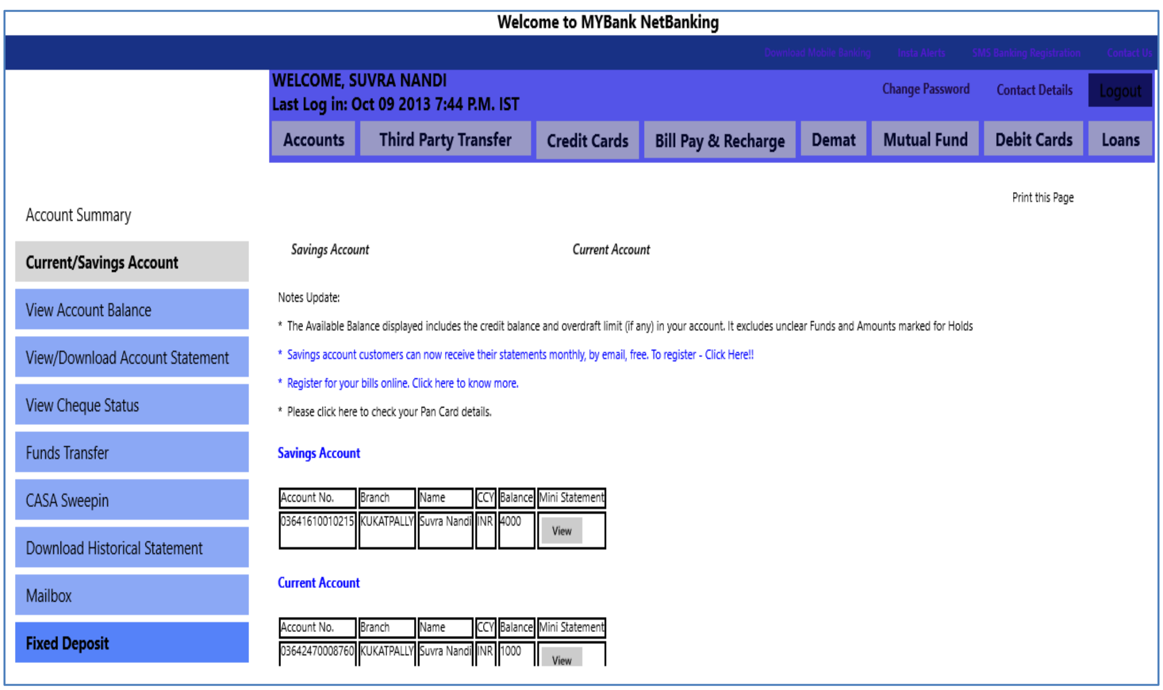


Figure 30: Windows Store Apps Implementation Screen - Current/Savings Account

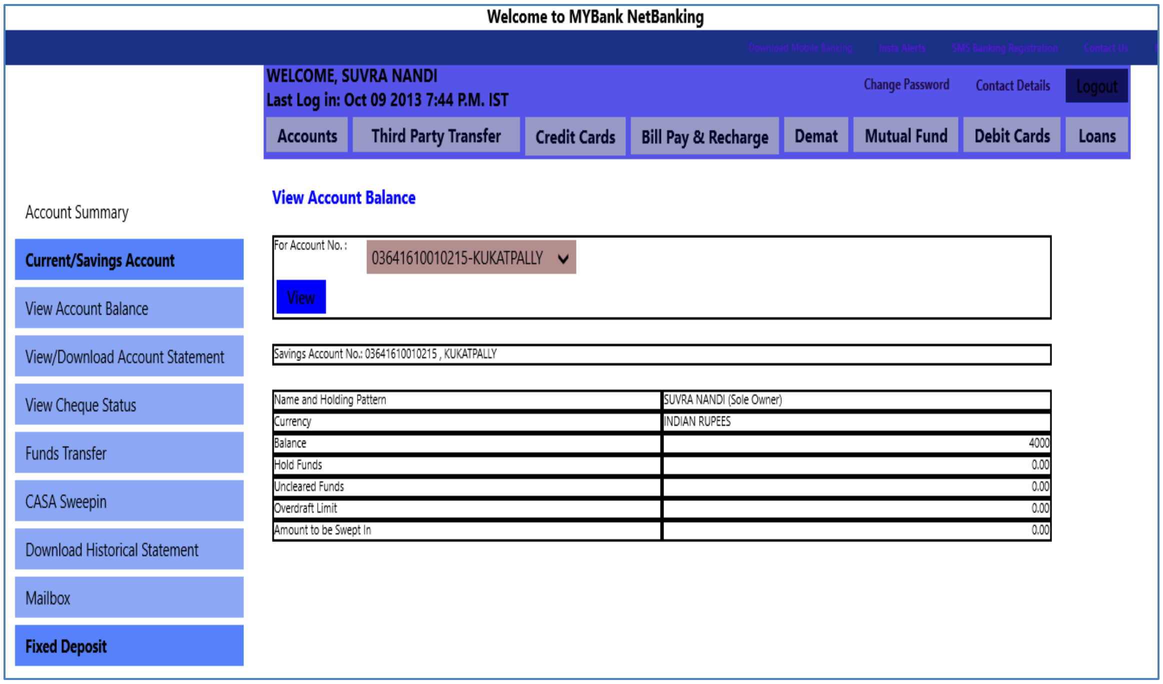


Figure 31: Windows Store Apps Implementation Screen - View Account Balance – Savings

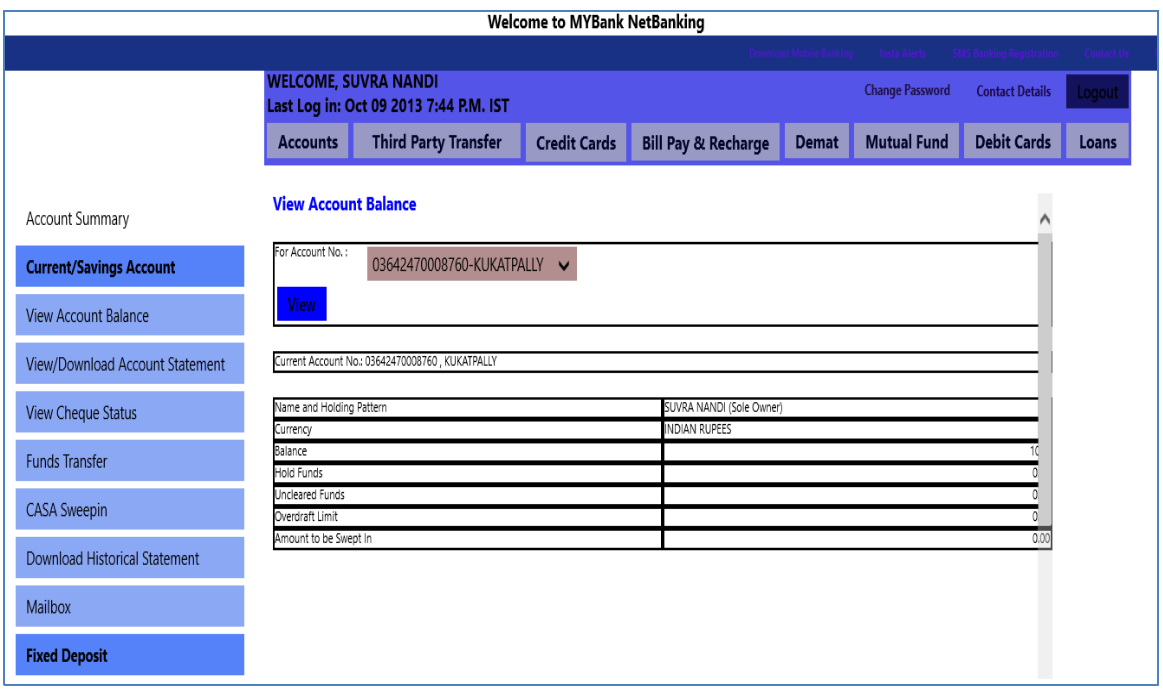


Figure 32: Windows Store Apps Implementation Screen - View Account Balance – Current

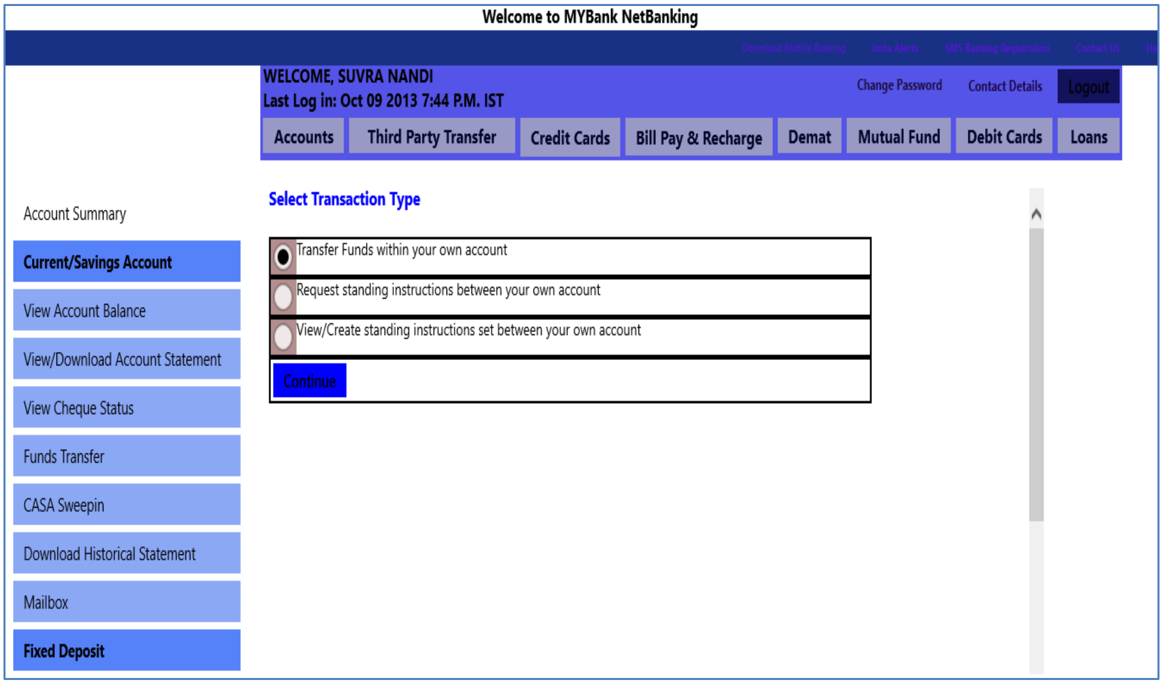


Figure 33: Windows Store Apps Implementation Screen - Funds Transfer - Transaction Type

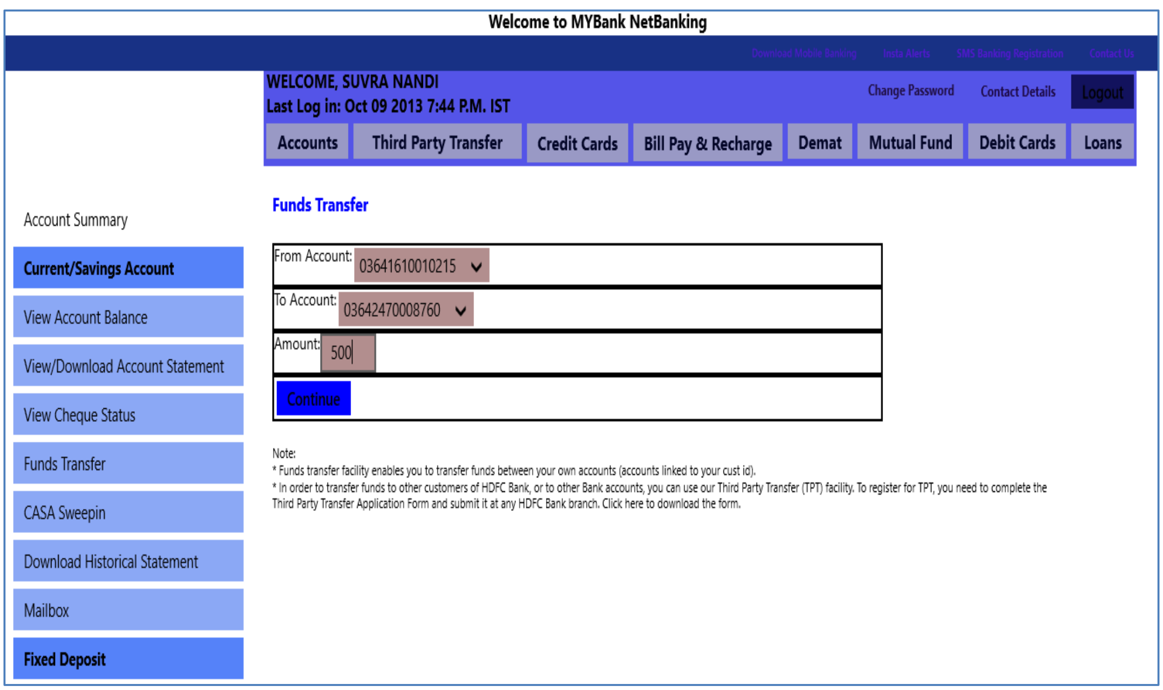


Figure 34: Windows Store Apps Implementation Screen - Funds Transfer

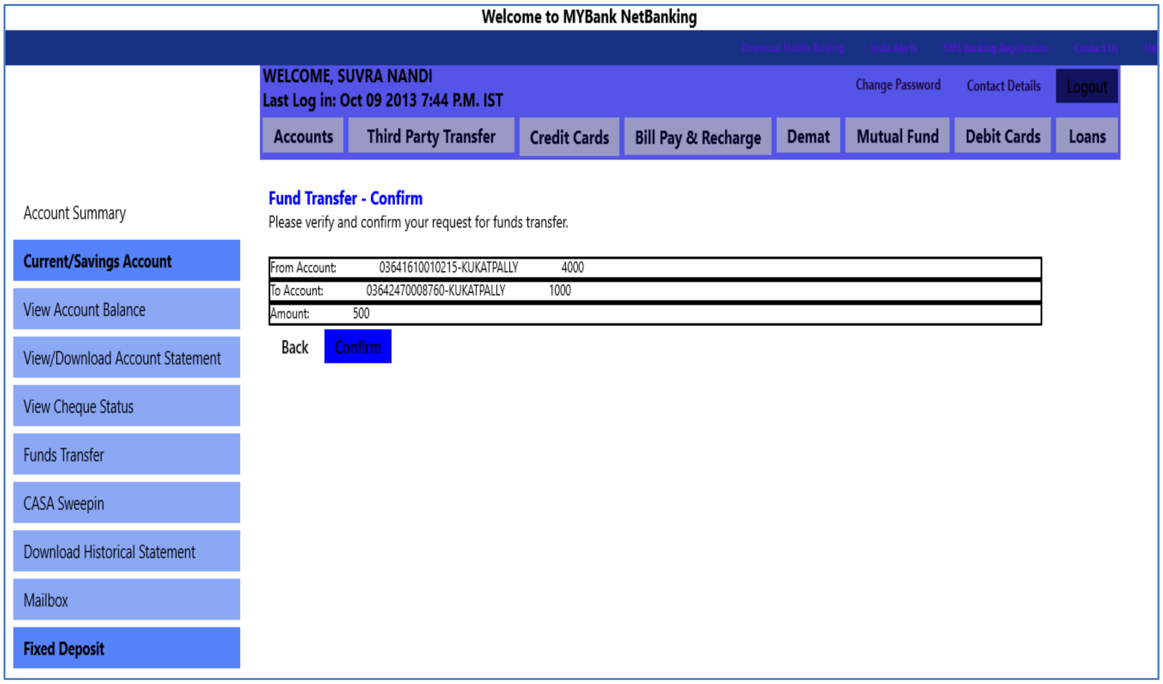


Figure 35: Windows Store Apps Implementation Screen - Funds Transfer – Confirm

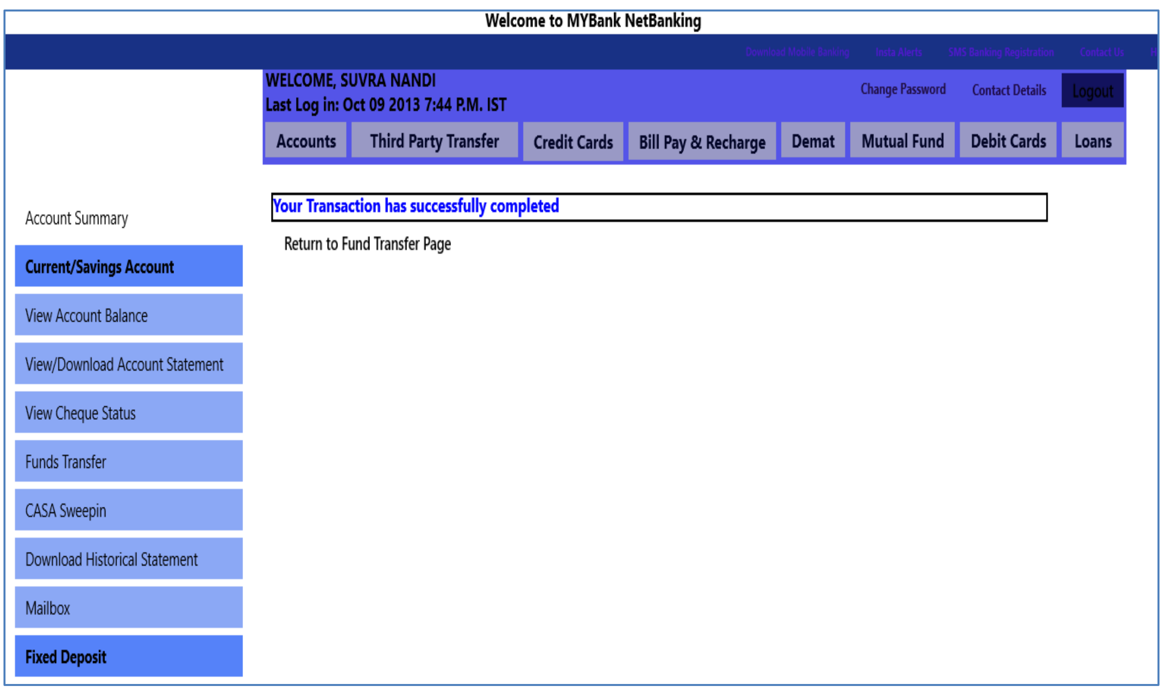


Figure 36: Windows Store Apps Implementation Screen - Funds Transfer Success Page

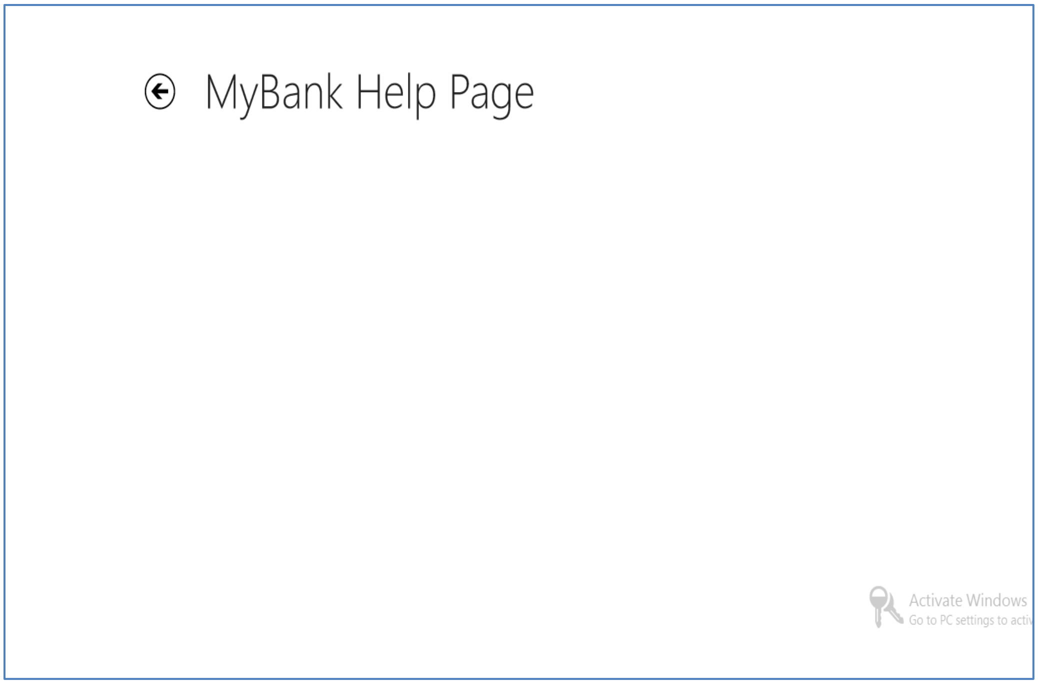


Figure 37: Windows Store Apps Implementation Screen - Blank Help Page with "Back" Link - for showing Page Navigation

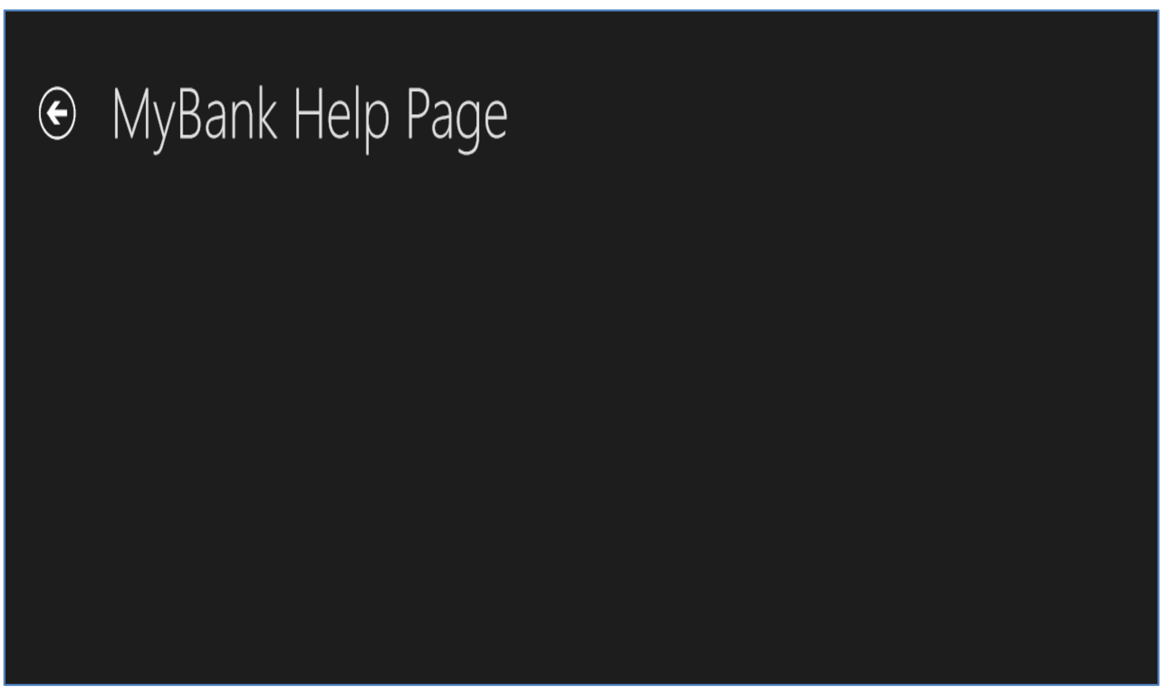


Figure 38: Windows Store Apps Implementation Screen - Blank Help page in Dark theme

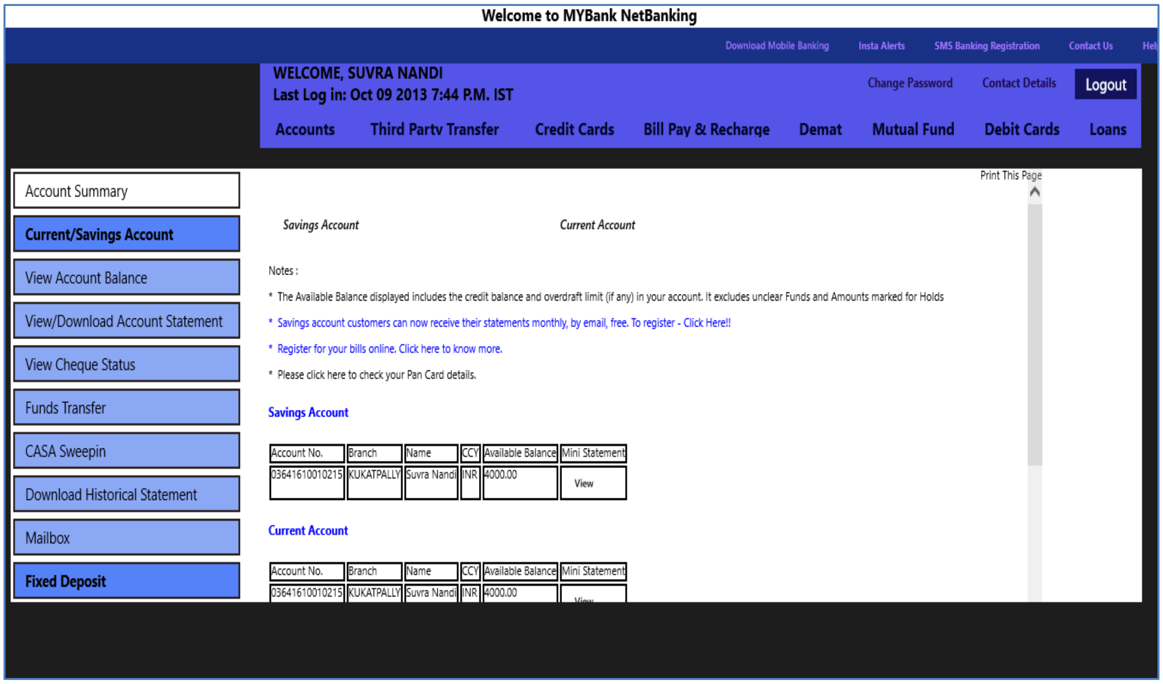


Figure 39: Windows Store Apps Implementation Screen - Account Summary page in Dark Theme

2.6.5 Evaluation through Implementation Experience

Windows Store is a digital distribution platform in Microsoft's Windows 8 and Windows RT operating systems. The platform can be used to provide listings for desktop applications certified to run on Windows 8, but are also the primary distribution platform for a new type of app called "Windows Store apps".

In this implementation XAML with Visual C++ was chosen as framework implementation languages.

a) Implementation Notes

- XAML provides Canvas, StackPanel and Grid objects for laying out components. Canvas was used in absolute layout, while StackPanel and Grid were used in Dynamic Layout
- Laying out an App Page contains : Grid system, Header, Content Area, Vertical Padding, Horizontal Padding
- XAML has very good support for Page Navigation
- XAML provides Declarative way of coding
- Passing parameter between pages in possible

- Viewing external webpage from an application page is also possible
- Two types of inbuilt themes are available in XAML based Store Apps : Light and Dark Theme. Themes need to be set programmatically
- Inbuilt Layout options are less compared to other frameworks
- Custom Skin, Custom Layout cannot be created
- Parsing XAML and creating the corresponding objects in memory can be time consuming for a complex UI

b) When XAML based Windows Store App should be used?

Developer may choose Windows Store Apps as UI design framework when:

- Windows Store Applications need to be built: Platform required for these applications are Windows 8 and MS Visual Studio Express. The platform can be used to provide listings for desktop applications certified to run on Windows 8. Declarative programming is used in this framework.

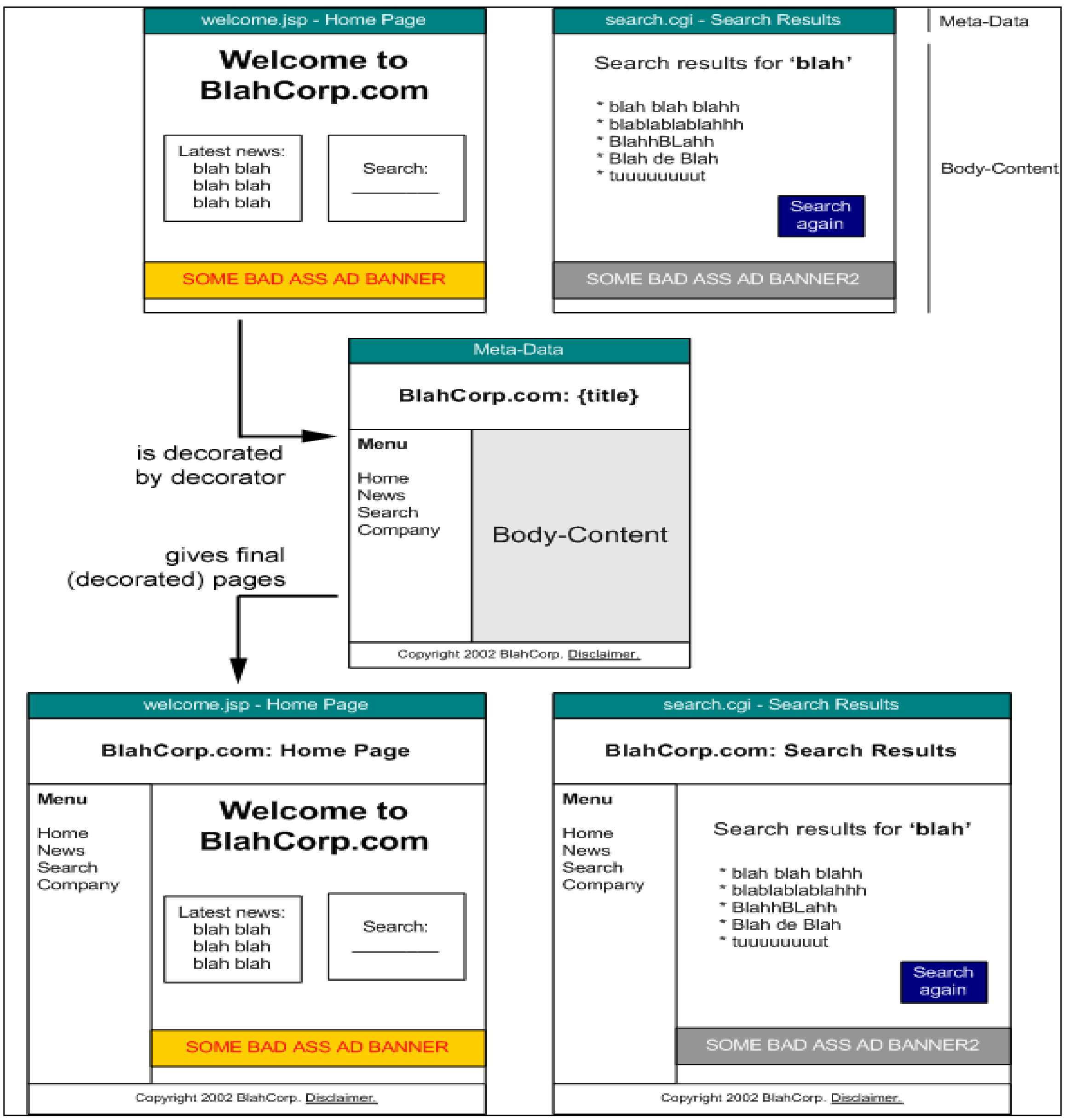
2.7 Study outcome of SiteMesh Framework

2.7.1 Layout Support in SiteMesh

SiteMesh (Reference[5]) provides Decorator Pattern for Laying out pages. Multiple decorators can also be used in a single application. Decorator pattern is described in detail below:

a) Decorator Design Pattern

Below figure demonstrates how Decorator Design Pattern is used:



The **decorator** contains the common layout and style that should be applied to the pages in the web application. It is a template that contains place holders for the content's <title>,<head> and <body> elements.

At the bare minimum, it should contain:

```
<html>

<head>

<title><sitemesh:write property='title'/></title>

<sitemesh:write property='head'/>

</head>

<body>

<sitemesh:write property='body'/>

</body>

</html>
```

The **<sitemesh:write property='...'/>** tag will be rewritten by SiteMesh to include properties extracted from the content. There are more properties that can be extracted from the content and it's possible to define one’s own rules.

The bare minimum decorator isn't very useful. Let's add some style and a bit of common layout.

Below is the sample file **/decorator.html** in a web-app, containing:

```
<html>

<head>

<title>SiteMesh example: <sitemesh:write property='title'/></title>

<style type='text/css'>

/* Some CSS */

body { font-family: arial, sans-serif; background-color: #ffffcc; }
```

```
h1, h2, h3, h4 { text-align: center; background-color: #ccffcc;
border-top: 1px solid #66ff66; }
.mainBody { padding: 10px; border: 1px solid #555555; }
.disclaimer { text-align: center; border-top: 1px solid #cccccc;
margin-top: 40px; color: #666666; font-size: smaller; }
</style>
<sitemesh:write property='head'/>
</head>
<body>
<h1 class='title'>SiteMesh example site: <sitemesh:write
property='title'/></h1>
<div class='mainBody'>
<sitemesh:write property='body'/>
</div>
<div class='disclaimer'>Site disclaimer. This is an example.</div>
</body>
</html>
```

In this example, the decorator is a static .html file, but if developer wants the decorator to be more dynamic, technologies such as JSP, FreeMarker, etc can be used.

Configuration

SiteMesh needs to be configured to know about this decorator and what it should do with it.

The configuration file should be created at **/WEB-INF/sitemesh3.xml**:

```
<sitemesh>

<mapping path="/*" decorator="/decorator.html"/>

</sitemesh>
```

This tells SiteMesh that requests matching the path /* (i.e. all requests) should be decorated with /decorator.html that was just created.

Creating Some Content

Below is /hello.html file created as sample content:

```
<html>

<head>

<title>Hello World</title>

<meta name='description' content='A simple page'>

</head>

<body>

<p>Hello <strong>world</strong>!</p>

</body>

</html>
```

Like the decorator, the content may be static files or dynamically generated by the Servlet engine (e.g. JSP).

The Result

Pointing browser to http://myserver/hello.html will serve the content just created, with the decorator applied. The resulting merged HTML will look like this:

```
<html>

<head>

<title>SiteMesh example: Hello World</title>

<style type='text/css'>

/* Some CSS */

body { font-family: arial, sans-serif; background-color: #ffffcc; }

h1, h2, h3, h4 { text-align: center; background-color: #ccffcc;

border-top: 1px solid #66ff66; }

.mainBody { padding: 10px; border: 1px solid #555555; }

.disclaimer { text-align: center; border-top: 1px solid #cccccc;

margin-top: 40px; color: #666666; font-size: smaller; }

</style>

<meta name='description' content='A simple page'>

</head>

<body>

<h1 class='title'>SiteMesh example site: Hello World</h1>

<div class='mainBody'>

<p>Hello <strong>world</strong>!</p>

</div>

<div class='disclaimer'>Site disclaimer. This is an example.</div>

</body>

</html>
```

Here the <title>, <head> and <body> have been extracted from the content and inserted into the decorator template.

2.7.2 Navigation Support in SiteMesh

Sitemesh framework provides good support for Page Navigation. It follows the decorator pattern. Each page including decorator page is in form of a “jsp” page. Hence, common page elements are placed in a decorator jsp, whereas dynamic parts are kept separately in respective “jsp” pages. Proper “jsp” page needs to be called by a command action, by mentioning the page name in the “Action” attribute of the command widget, such as Button, CommandLink etc.

2.7.3 Theme Support in SiteMesh

Custom skins and themes can be provided to sitemesh components by exclusively providing styling information in CSS <style> tag. CSS classes defined can be used while creating original components. Sample code is provided below:

```
<?xml version="1.0" encoding="UTF-8" ?>

<%@ taglib uri="http://www.opensymphony.com/sitemesh/decorator"
prefix="decorator" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<style>

.loginLink {

color: #002862;

font-size: 11px;

line-height: 13px;

padding-left: 0;

padding-top: 25px;

width: 300px;

}

.buttonBold {

display: block;

width: 160px;
```

```
height: 25px;

background: #c9dfef;

padding: 10px;

text-align: center;

border-radius: 5px;

color: black;

font-weight: bold;

}

…

…

<div id="leftPanel" style="width: 200px; height: 400px; overflow: scroll">

<table width="100%" border="1">

<tbody>

<tr><td>Account Summary</td></tr>

<tr><td><a href="accountSummary.jsp" class="buttonBold">Current/Savings
Account</a></td></tr>

<tr><td><a href="viewAccountBalance.jsp" class="button">View Account
Balance</a></td></tr>

<tr><td><a href="" class="button">View/Download Account
Statement</a></td></tr>

<tr><td><a href="fundXferTranType.jsp" class="button">Fund
Transfer</a></td></tr>

<tr><td><a href="" class="button">CASA Sweepin</a></td></tr>

<tr><td><a href="" class="button">Download Historical
Statement</a></td></tr>

<tr><td><a href="" class="button">Mailbox</a></td></tr>

<tr><td><a href="" class="buttonBold">Fixed Deposit</a></td></tr>
```

```
<tr><td><a href="" class="button">Fixed Deposits Summary</a></td></tr>

<tr><td><a href="" class="button">Open Fixed Deposit &lt Rs 1 Cr</a></td></tr>

<tr><td><a href="" class="button">Open Fixed Deposit &gt  Rs 1
Cr</a></td></tr>

<tr><td><a href="" class="button">Fixed Deposit Sweep-in</a></td></tr>

</tbody>

</table>

</div>
```

2.7.4 Sample Screens Implemented in SiteMesh

Few pages of a sample Bank Website was built using SITEMESH framework, taking help of Layout objects.

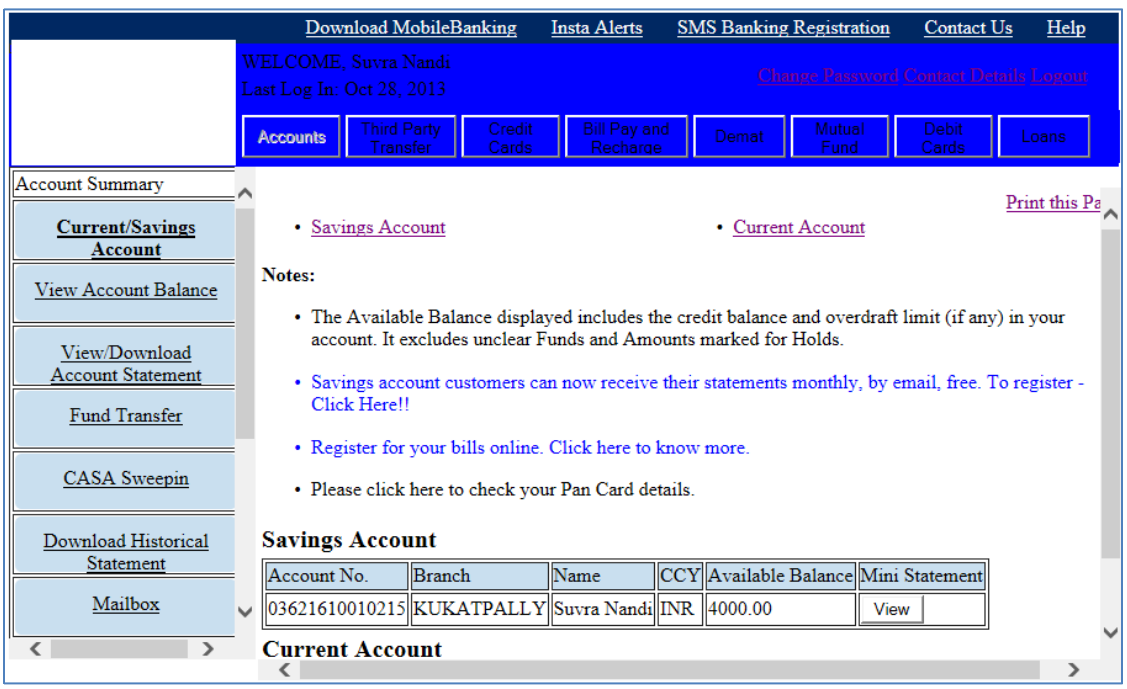


Figure 40: SiteMesh Implementation Screen - Account Summary

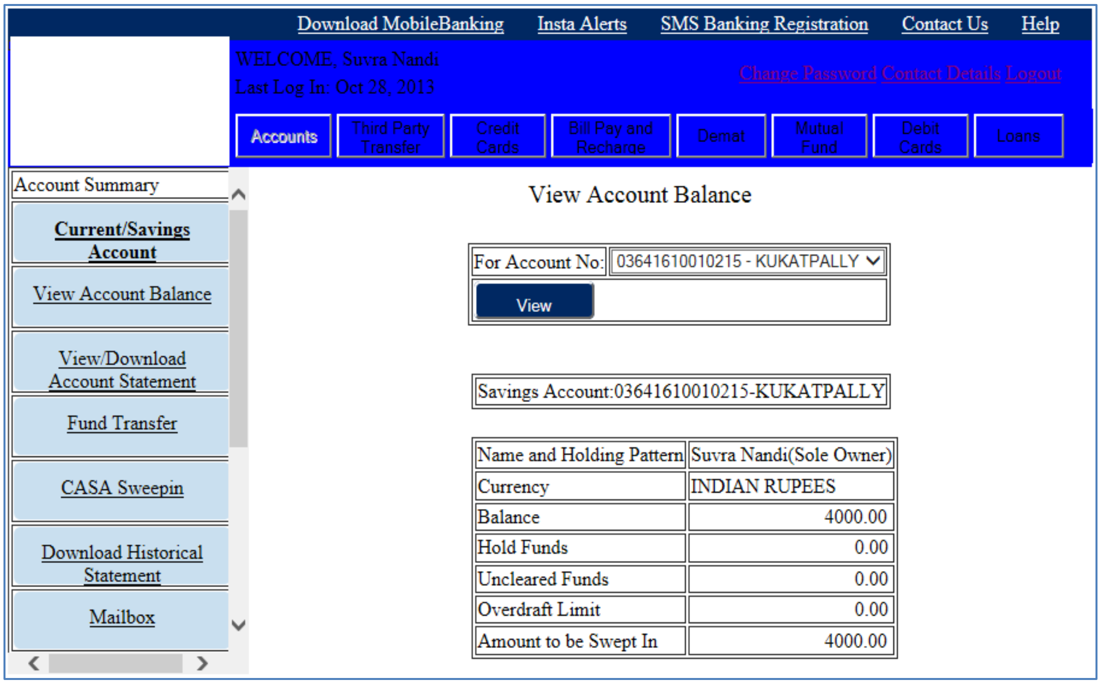


Figure 41: SiteMesh Implementation Screen - View Account Balance – Savings

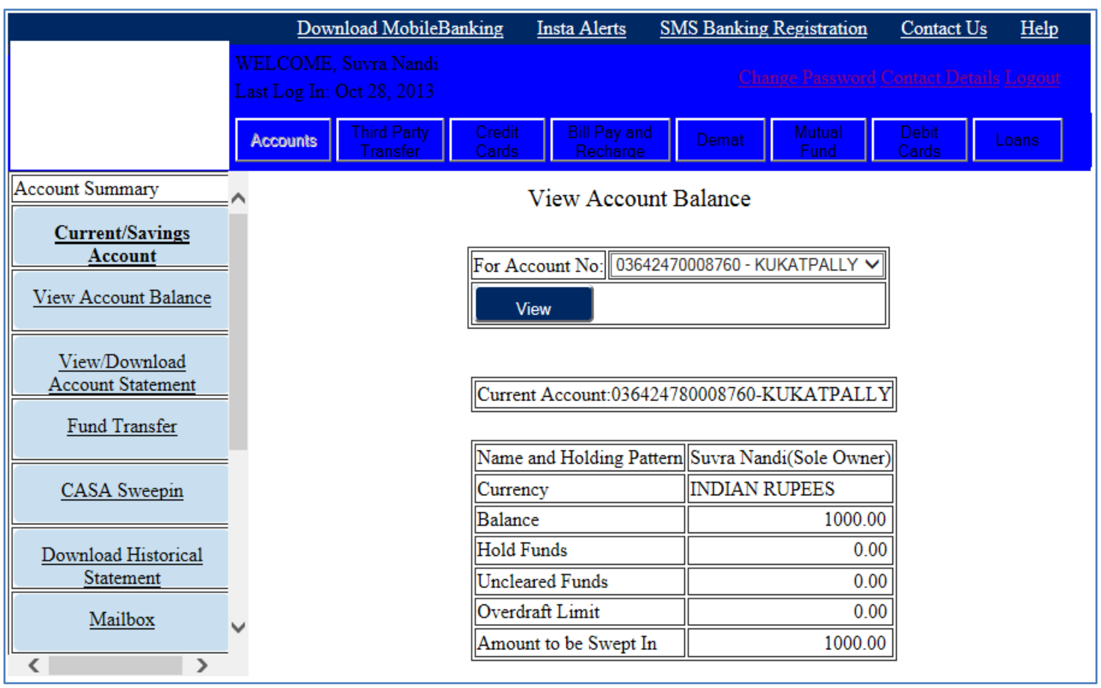


Figure 42: SiteMesh Implementation Screen - View Account Balance – Current

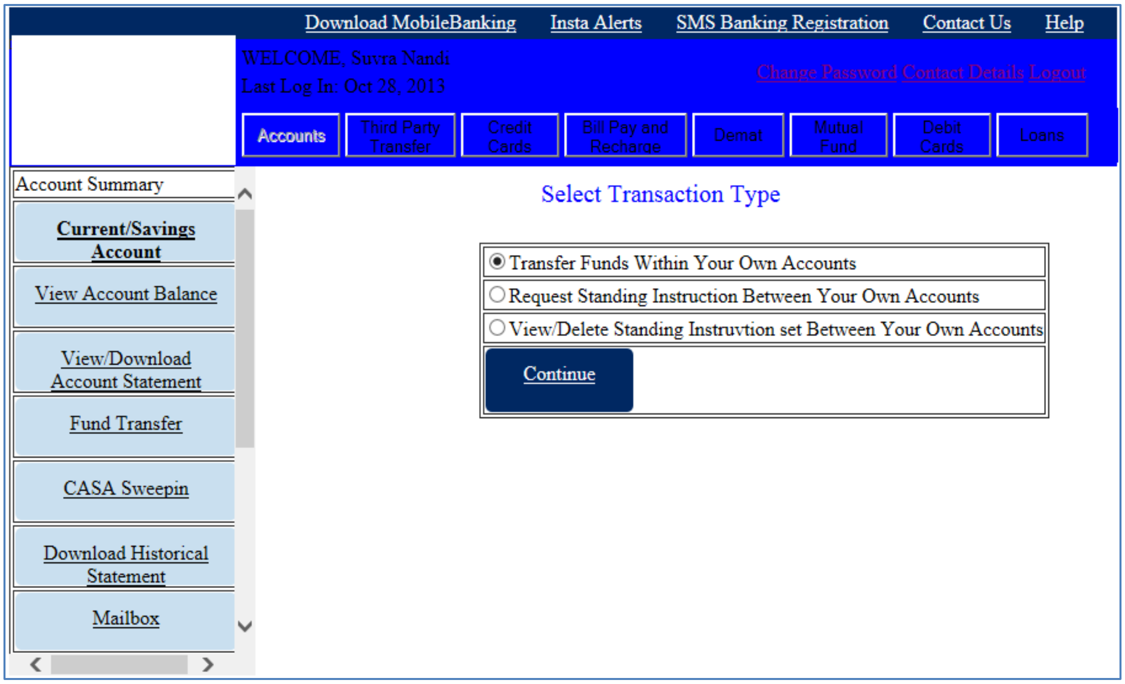


Figure 43: SiteMesh Implementation Screen - Fund Transfer - Tran Type

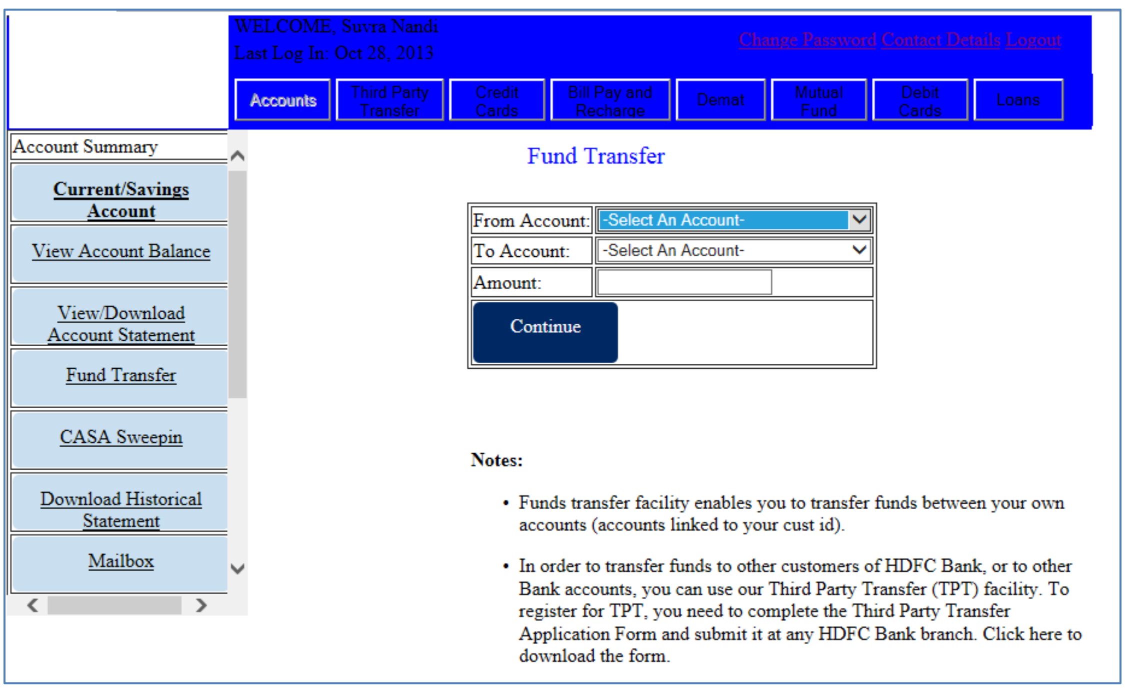


Figure 44: SiteMesh Implementation Screen - Fund Transfer

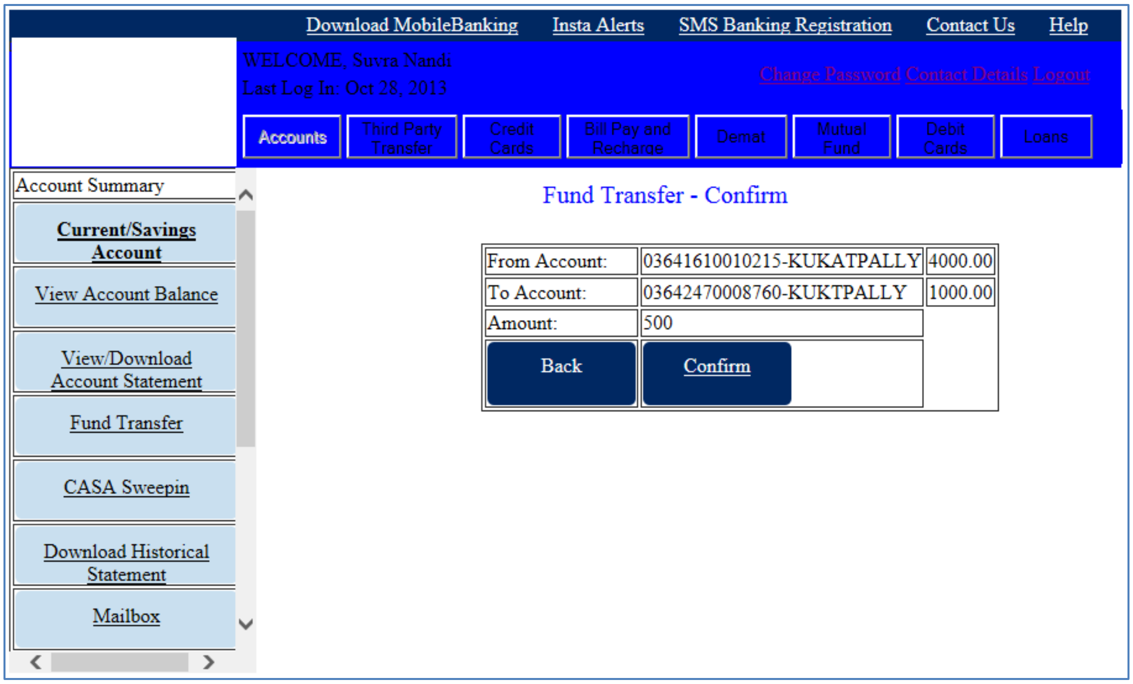


Figure 45: SiteMesh Implementation Screen - Fund Transfer – Confirm

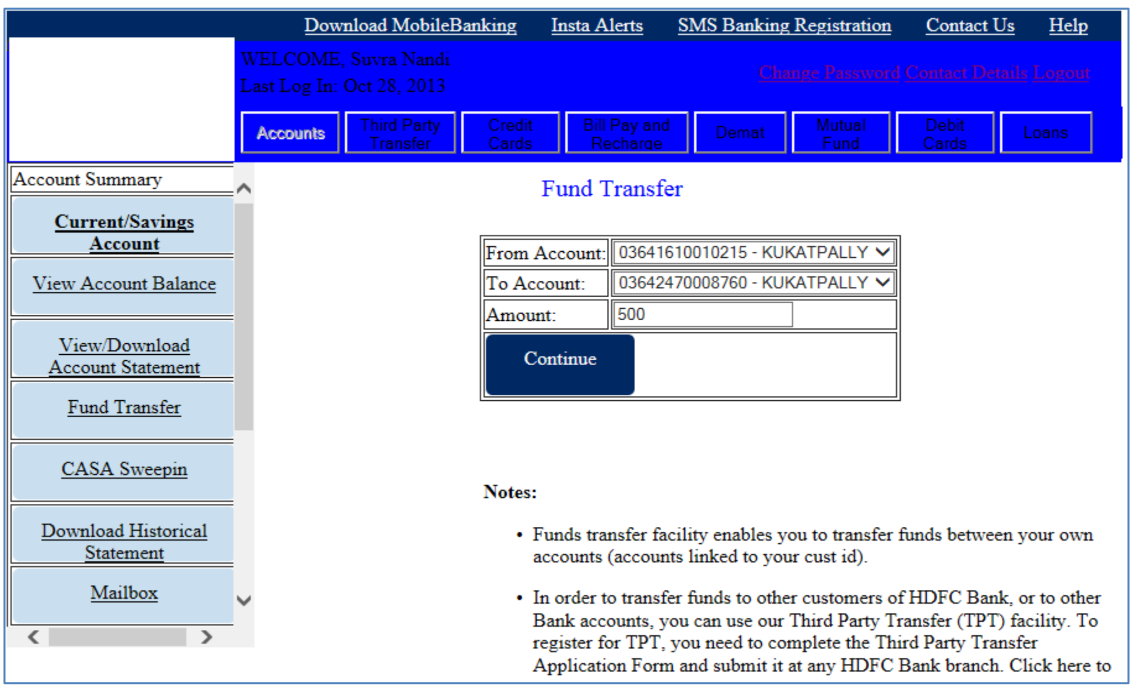


Figure 46: SiteMesh Implementation Screen - Fund Transfer after Back Navigation

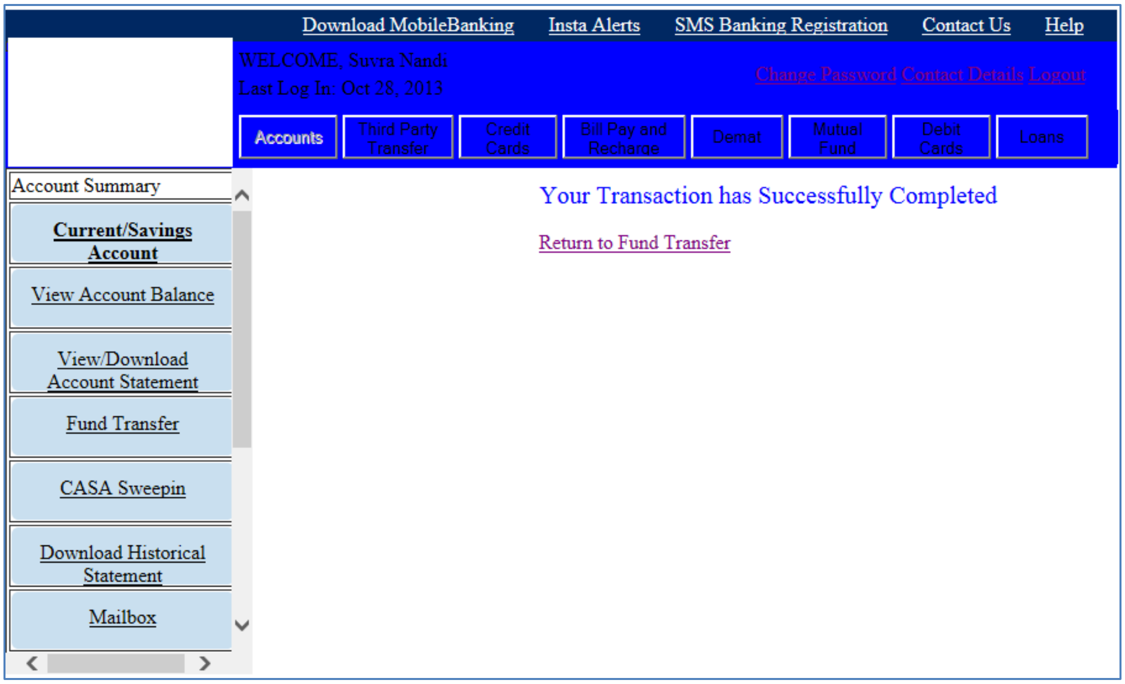


Figure 47: SiteMesh Implementation Screen - Transaction Success Page

Figure 48: SiteMesh Implementation Screen - Blank Help Page on click of "Help" link for Page Navigation Demo

2.7.5 Evaluation through Implementation Experience

SiteMesh is a web-page layout and decoration framework and web- application integration framework to aid in creating sites consisting of pages for which a consistent look/feel, navigation and layout scheme is required.

a) Implementation Notes

- Decorator pattern helped creating templates which were merged with main content. This concept enhanced code reuse
- The filter intercepts requests to Content, runs it through the Content Processor and merges with a Decorator
- Content and Decorator pages can be created using HTML / JSP or other View Technologies
- The Content Processor contains the rules for extracting and transforming the content - it has some simple default rules and can be customized
- SiteMesh provides good page navigation support
- Decorators can be used for creating Custom Skins in CSS, which in turn will be applied to all decorated pages
- Built on Java. The Framework is Simple, Very Flexible and can be integrated with other Decorators like Velocity, FreeMaker etc
- SiteMesh stores the entire content of HTML body into memory before it decorates it. If developer has some very large pages, such as might happen in a reporting application where pagination is not implemented and end up with one large page of rows, severe memory problems may arise

b) When SiteMesh should be used?

Developers should choose SITEMESH as GUI Development Framework when they need:

- Decorator Pattern: Decorator Pattern enhances code reuse and filtering.
- Needs Flexibility on Technology: SiteMesh does not care what technologies are used to generate the content or the decorator. They may be static files, Servlet, JSPs, other filters, MVC frameworks, etc. So long as it's served by the Servlet engine, SiteMesh can work with it.

2.8 Study outcome of QT QML Framework

2.8.1 Layout Support in QT QML

QT-QML (Reference[6]) provides three ways of positioning items on a page :

- Anchor Based Layout
- Positioners
- Manual Positioning

a) Anchor Based Layout

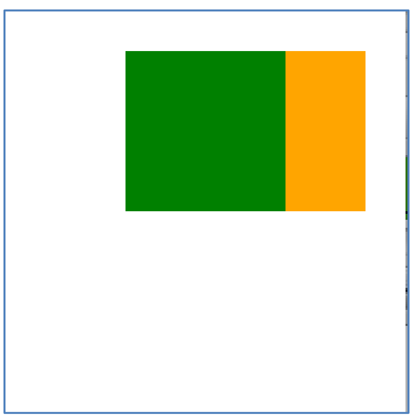
In Anchor Based Layouts, anchor can be applied to an item with its sibling and direct parent only.

The Item type provides the ability to anchor to other Item types. There are six anchor lines for each item: left, right, vertical center, top, bottom and horizontal center. The three vertical anchor lines can be anchored to any of the three vertical anchor lines of another item, and the three horizontal anchor lines can be anchored to the horizontal anchor lines of another item.

Below is an example of Anchor Based layout with its result :

```
import QtQuick 2.0

Item {

width: 200; height: 200

Rectangle {

// Anchored to 20px off the top right corner of the parent

anchors.right: parent.right

anchors.top: parent.top

anchors.margins: 20 // Sets all margins at once

width: 80

height: 80

color: "orange"

}
```

```
Rectangle {

// Anchored to 20px off the top center corner of the parent.

// Notice the different group property syntax for 'anchors'
compared to

// the previous Rectangle. Both are valid.

anchors { horizontalCenter: parent.horizontalCenter; top:
parent.top; topMargin: 20 }

width: 80

height: 80

color: "green"

}

}
```



b) Positioners

For the common case of wanting to position a set of elements in a regular pattern, QtQuick provides some positioner types. Items placed in a positioner are automatically positioned in some way; for example, a Row positions items to be horizontally adjacent (forming a row).

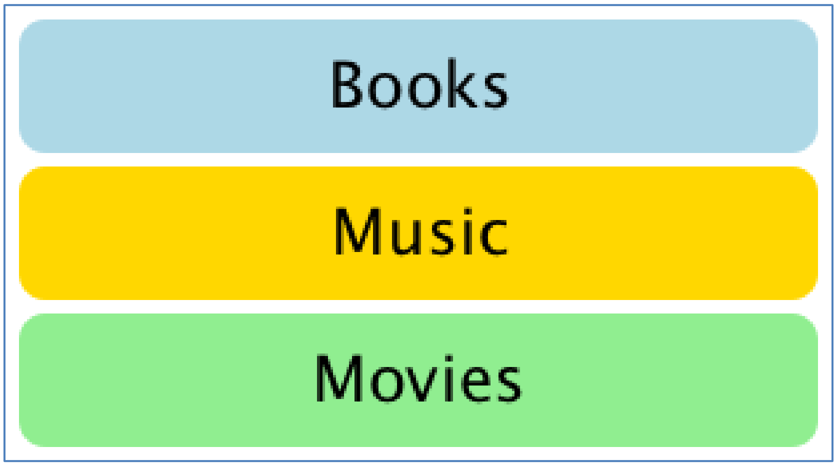
Four Types of Positioners are available in QT QML:

- Column
- Row

- Grid
- Flow

Column

Column items are used to vertically arrange items. The following example uses a Column item to arrange three Rectangle items in an area defined by an outer Item. The spacing property is set to include a small amount of space between the rectangles.



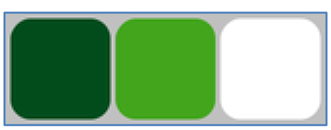

```
import QtQuick 2.0

Item {

width: 310; height: 170

Column {

anchors.horizontalCenter: parent.horizontalCenter

anchors.verticalCenter: parent.verticalCenter

spacing: 5

Rectangle { color: "lightblue"; radius: 10.0

width: 300; height: 50

Text { anchors.centerIn: parent

font.pointSize: 24; text: "Books" } }

Rectangle { color: "gold"; radius: 10.0
```

```
width: 300; height: 50

Text { anchors.centerIn: parent

font.pointSize: 24; text: "Music" } }

Rectangle { color: "lightgreen"; radius: 10.0

width: 300; height: 50

Text { anchors.centerIn: parent

font.pointSize: 24; text: "Movies" } }

}

}
```

Row

Row items are used to horizontally arrange items. The following example uses a Row item to arrange three rounded Rectangle items in an area defined by an outer colored Rectangle. The spacing property is set to include a small amount of space between the rectangles.

We ensure that the parent Rectangle is large enough so that there is some space left around the edges of the horizontally centered Row item.



```
import QtQuick 2.0

Rectangle {

width: 320; height: 110

color: "#c0c0c0"

Row {       anchors.horizontalCenter: parent.horizontalCenter

anchors.verticalCenter: parent.verticalCenter
```

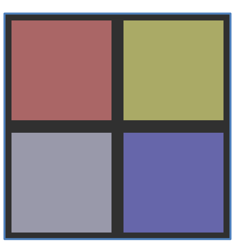
```
spacing: 5

Rectangle { width: 100; height: 100; radius: 20.0

color: "#024c1c" }

Rectangle { width: 100; height: 100; radius: 20.0

color: "#42a51c" }

Rectangle { width: 100; height: 100; radius: 20.0

color: "white" }

}

}
```

Grid

Grid items are used to place items in a grid or table arrangement. The following example uses a Grid item to place four Rectangle items in a 2-by-2 grid. As with the other positioners, the spacing between items can be specified using the spacing property.

There is no difference between horizontal and vertical spacing inserted between items, so any additional space must be added within the items themselves.

Any empty cells in the grid must be created by defining placeholder items at the appropriate places in the Grid definition.



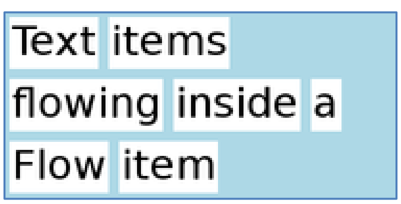
```
import QtQuick 2.0

Rectangle {

width: 112; height: 112

color: "#303030"
```

```
Grid {

anchors.horizontalCenter: parent.horizontalCenter

anchors.verticalCenter: parent.verticalCenter

columns: 2

spacing: 6

Rectangle { color: "#aa6666"; width: 50; height: 50 }

Rectangle { color: "#aaaa66"; width: 50; height: 50 }

Rectangle { color: "#9999aa"; width: 50; height: 50 }

Rectangle { color: "#6666aa"; width: 50; height: 50 }

}

}
```

Flow

Flow items are used to place items like words on a page, with rows or columns of non-overlapping items.

Flow items arrange items in a similar way to Grid items, with items arranged in lines along one axis (the minor axis), and lines of items placed next to each other along another axis (the major axis). The direction of flow, as well as the spacing between items, is controlled by the flow and spacing properties.




```
import QtQuick 2.0

Rectangle {

color: "lightblue"

width: 300; height: 200
```

```
Flow {
anchors.fill: parent
anchors.margins: 4
spacing: 10
Text { text: "Text"; font.pixelSize: 40 }
Text { text: "items"; font.pixelSize: 40 }
Text { text: "flowing"; font.pixelSize: 40 }
Text { text: "inside"; font.pixelSize: 40 }
Text { text: "a"; font.pixelSize: 40 }
Text { text: "Flow"; font.pixelSize: 40 }
Text { text: "item"; font.pixelSize: 40 }
}
}
```

The main differences between the Grid and Flow positioners are that items inside a Flow will wrap when they run out of space on the minor axis, and items on one line may not be aligned with items on another line if the items do not have uniform sizes. As with Grid items, there is no independent control of spacing between items and between lines of items.

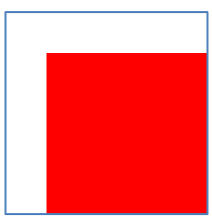
c) Manual Positioning

Items can be placed at specific x,y coordinates on the screen by setting their x,y properties. This will setup their position relative to the top left corner of their parent, according to the visual coordinate system rules.

Combined with using bindings instead of constant values for these properties, relative positioning is also easily accomplished by setting the x and y coordinates to the appropriate bindings.

Below is an example of Manual Positioning with its result :

```
import QtQuick 2.0

Item {

width: 100; height: 100

Rectangle {

// Manually positioned at 20,20

x: 20

y: 20

width: 80

height: 80

color: "red"

}

}
```



2.8.2 Navigation Support in QT QML

QT QML provides support for Page Navigation in two different ways :

- Page Stack Based Navigation
- State Based Navigation

a) Page Stack Based Navigation

QT-QML supports Stack Based Page Navigation model. A Page Stack consists of one or more pages, defined by Page Objects. A Page may be pushed onto the stack to place it on top of the stack, or popped to remove it from the stack.

This stack-based navigation model makes it simple to provide hierarchical navigation within an application. When the application is required to provide a page of content, a new page can be pushed onto the stack; if the user requests to return to the previous page, the current page can be popped to reveal the previous page. Otherwise, another page can be pushed onto the stack to navigate deeper into the user interface hierarchy, and so on.

The QStackedWidget class provides a stack of widgets where only one widget is visible at a time. QStackedWidget can be constructed and populated with a number of child widgets ("pages") like below:

```
QWidget *firstPageWidget = new QWidget;

QWidget *secondPageWidget = new QWidget;

QWidget *thirdPageWidget = new QWidget;

QStackedWidget *stackedWidget = new QStackedWidget;

stackedWidget->addWidget(firstPageWidget);

stackedWidget->addWidget(secondPageWidget);

stackedWidget->addWidget(thirdPageWidget);

QVBoxLayout *layout = new QVBoxLayout;

layout->addWidget(stackedWidget);

setLayout(layout);
```

QStackedWidget provides no intrinsic means for the user to switch page. This is typically done through a QComboBox or a QListWidget that stores the titles of the QStackedWidget's pages. For example:

```
QComboBox *pageComboBox = new QComboBox;

pageComboBox->addItem(tr("Page 1"));

pageComboBox->addItem(tr("Page 2"));

pageComboBox->addItem(tr("Page 3"));

connect(pageComboBox, SIGNAL(activated(int)),

stackedWidget, SLOT(setCurrentIndex(int)));
```

b) State Based Navigation

In QML, states are a set of property configurations defined in a State element. Different configurations could, for example:

- ✓ Show some UI elements and hide others
- ✓ Present different available actions to the user
- ✓ Start, stop, or pause animations
- ✓ Execute some script required in the new state
- ✓ Change a property value for a particular item
- ✓ Show a different view or screen

States can be used for page navigation in below way :

```
states :[

State{

name: "accountSummary"

when: leftPanel.activeButton == "accountSummary"

PropertyChanges { target: centerPanelAccountSummary; visible:true; }

PropertyChanges { target: centerPanelViewAccountBalance;
visible:false; }

PropertyChanges { target: centerPanelFundXfer; visible:false; }

},

State{

name: "viewAccountBalance"

when: leftPanel.activeButton == "viewAccountBalance"

PropertyChanges { target: centerPanelAccountSummary; visible:false;
}

PropertyChanges { target: centerPanelViewAccountBalance;
visible:true; }
```

```
PropertyChanges { target: centerPanelFundXfer; visible:false; }

},

State{

name: "fundXfer"

when: leftPanel.activeButton == "fundXfer"

PropertyChanges { target: centerPanelAccountSummary; visible:false;
}

PropertyChanges { target: centerPanelViewAccountBalance;
visible:false; }

PropertyChanges { target: centerPanelFundXfer; visible:true; }

} ]
```

This code can be placed in the main.qml file, where specified targets “centerPanelAccountSummary”, “centerPanelViewAccountBalance”, “centerPanelFundXfer” are different qml pages that gets called on fulfilling the “when” criteria.

2.8.3 Theme Support in QT QML

The types provided in the QtQuick module are not complete user interface elements on their own. A common use case is to develop a set of custom styled user interface elements out of the types in the QtQuick module. This is easily accomplished by creating one's reusable components.

With the reusable components approach, developers can define their own type with the appearance they want to have in the application and style that type directly. Then the custom type can be used instead of unstyled type.

In the below example MyText.qml is created, with certain style properties in Text. MyText can then be used in main code instead of Text just to reflect the styled text.

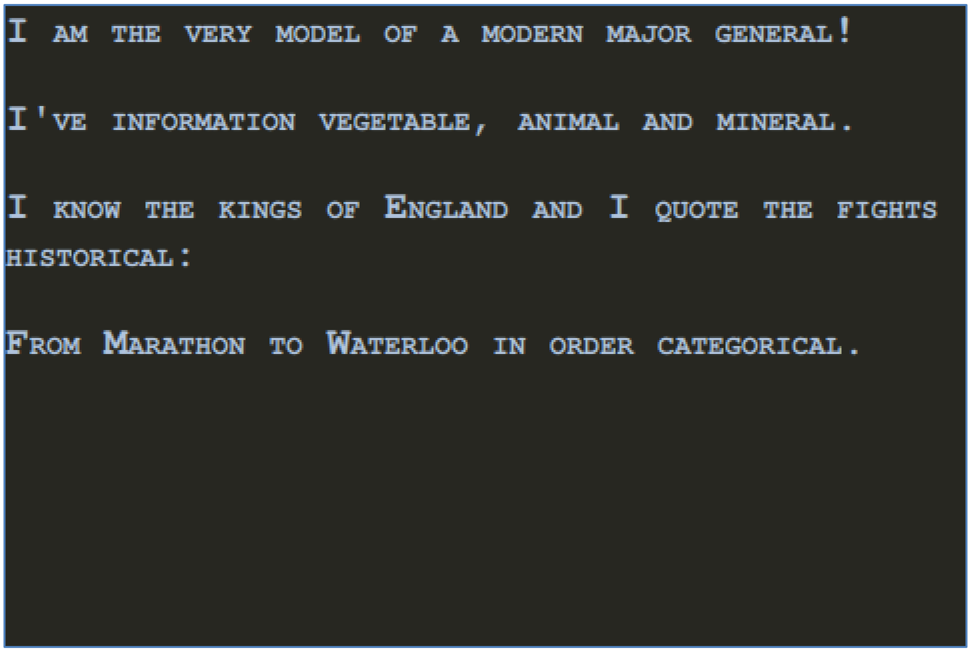
MyText.qml :

```
import QtQuick 2.0

Text {

color: "lightsteelblue"

font { family: 'Courier'; pixelSize: 20; bold: true; capitalization:
Font.SmallCaps }

}
```

Using MyText, instead of Text :

```
Column {

spacing: 20

MyText { text: 'I am the very model of a modern major general!' }

MyText { text: 'I\'ve information vegetable, animal and mineral.' }

MyText {

width: root.width

wrapMode: Text.WordWrap

text: 'I know the kings of England and I quote the fights historical:'

}

MyText { text: 'From Marathon to Waterloo in order categorical.' }
}
```

Below is the output of above code snippet:



2.8.4 Sample Screens Implemented in QT QML

Few pages of a sample Bank Website was built using QT QML framework, taking help of Layout objects.

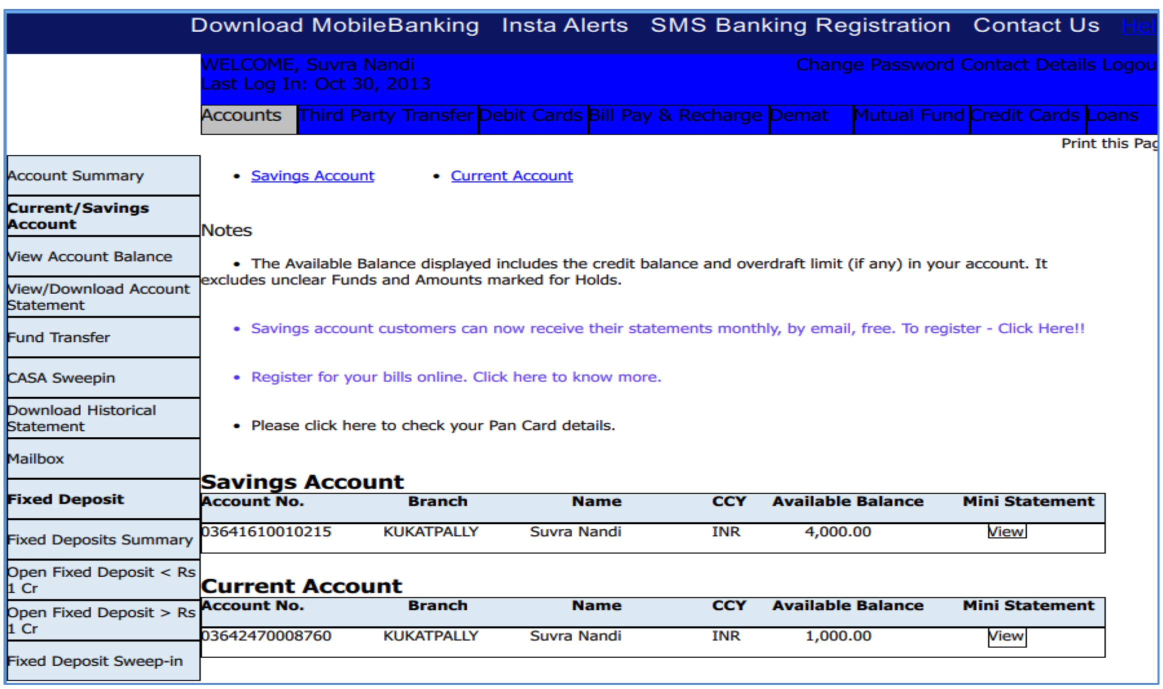


Figure 49: Qt QML Implementation Screen - Account Summary

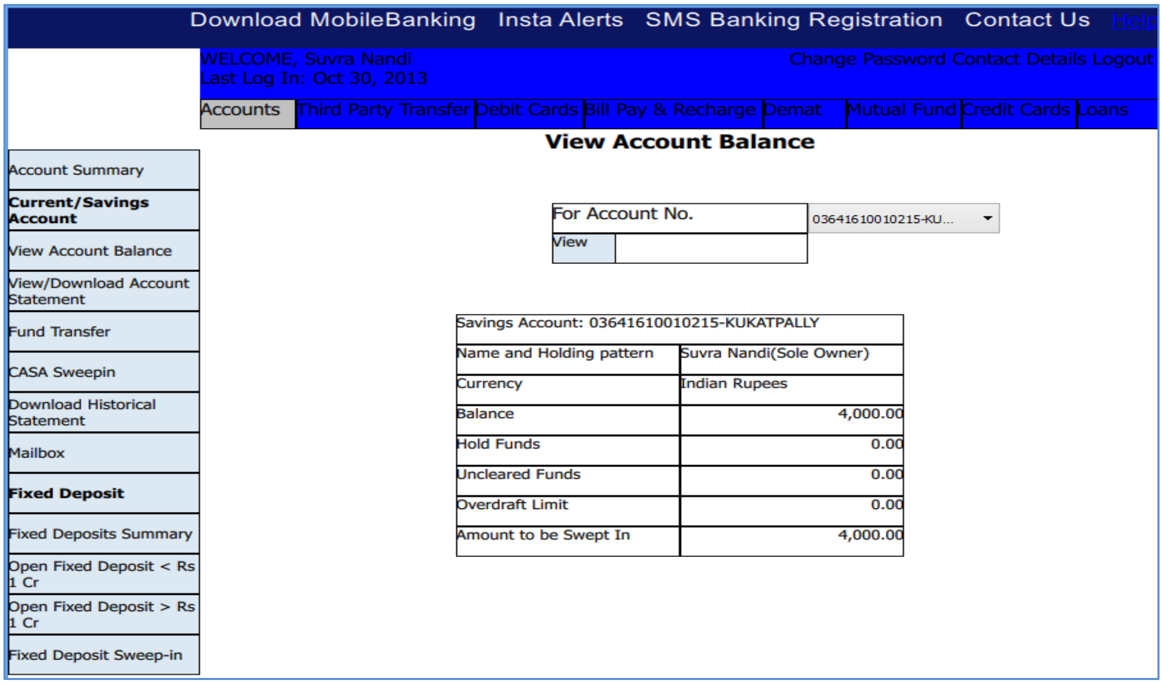


Figure 50: Qt QML Implementation Screen - View Account Balance – Savings

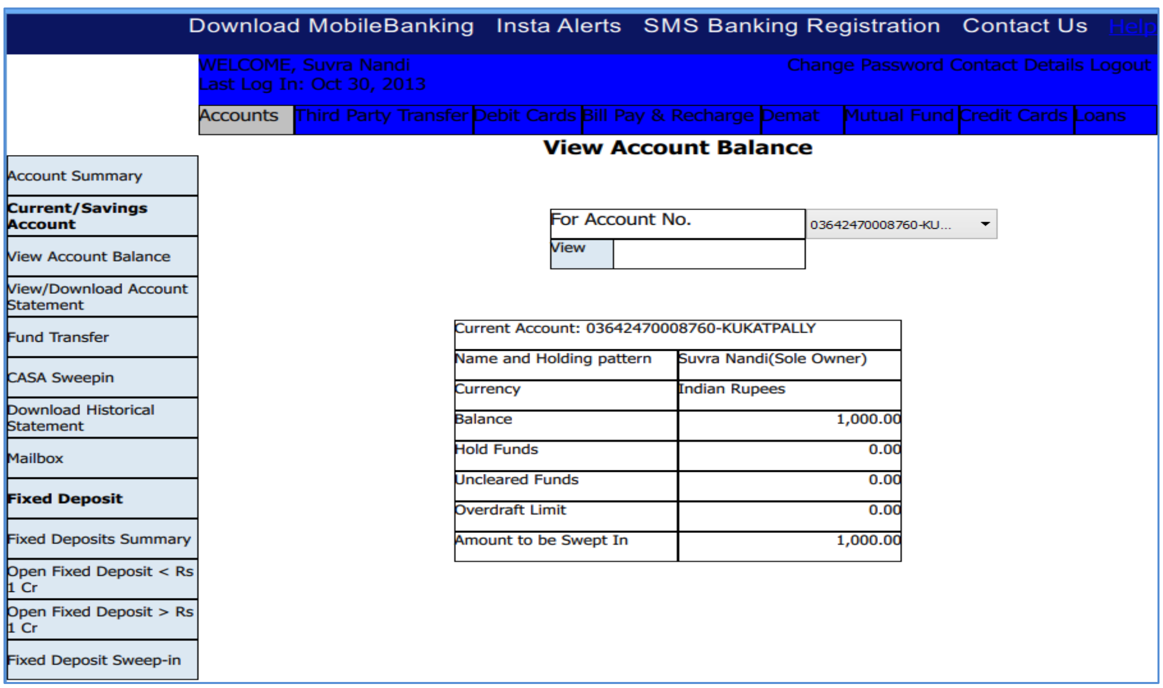


Figure 51: Qt QML Implementation Screen - View Account Balance – Current

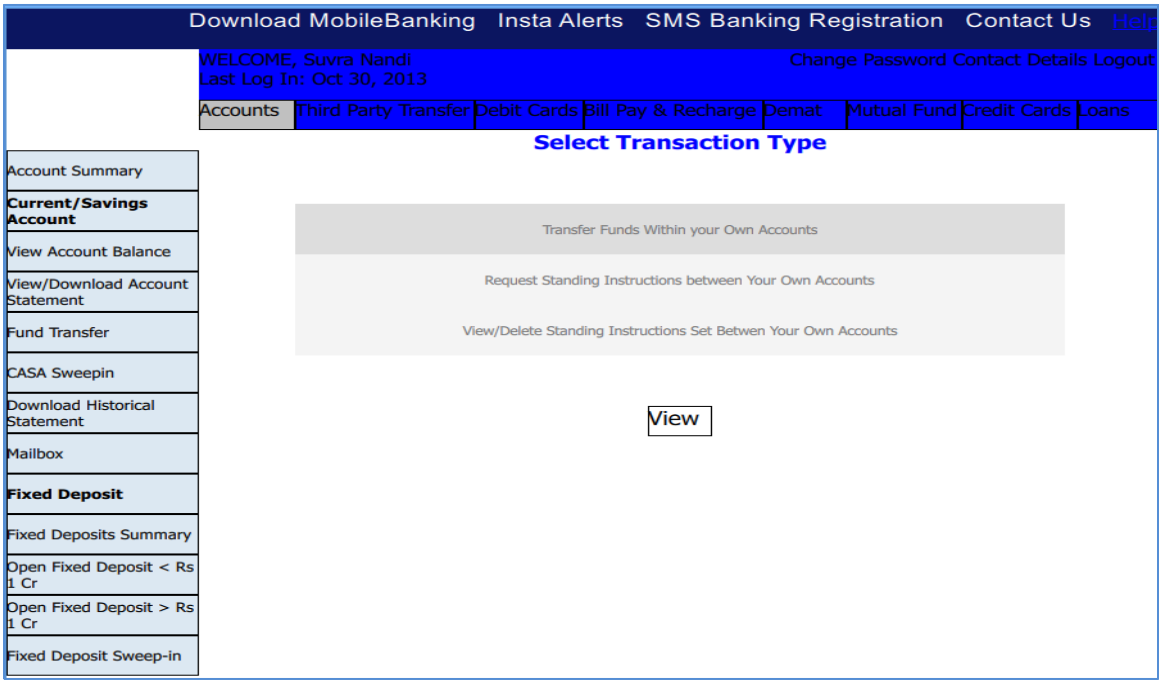


Figure 52: Qt QML Implementation Screen - Fund Transfer - Transaction Type

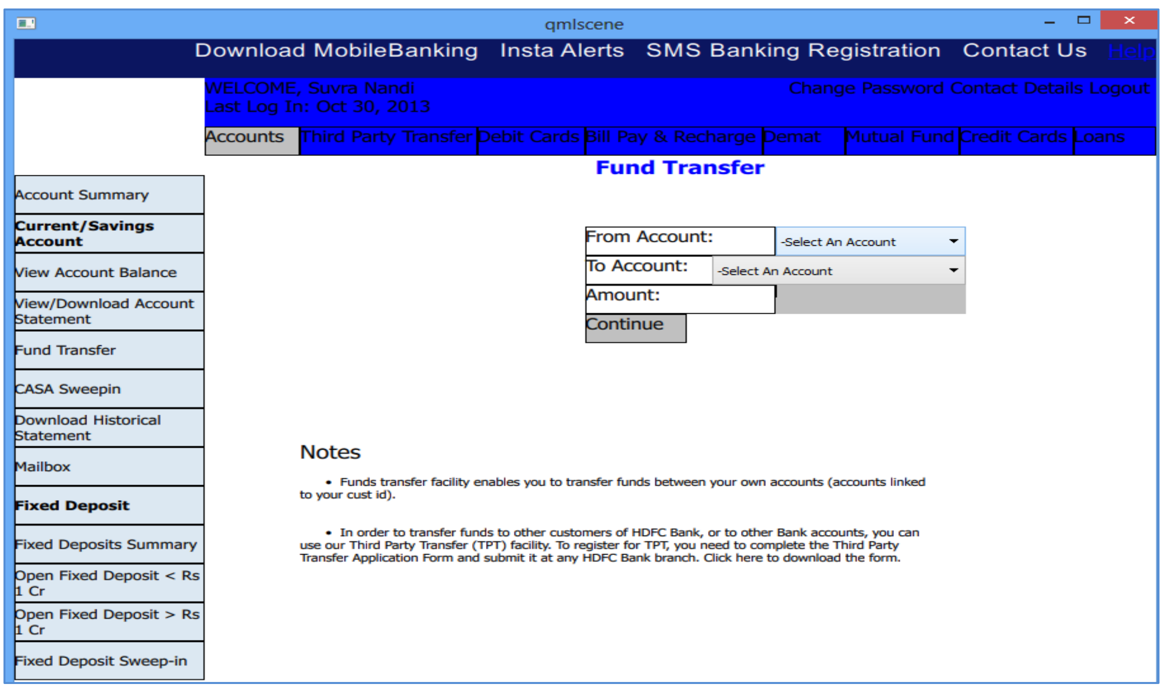


Figure 53: Qt QML Implementation Screen - Fund Transfer

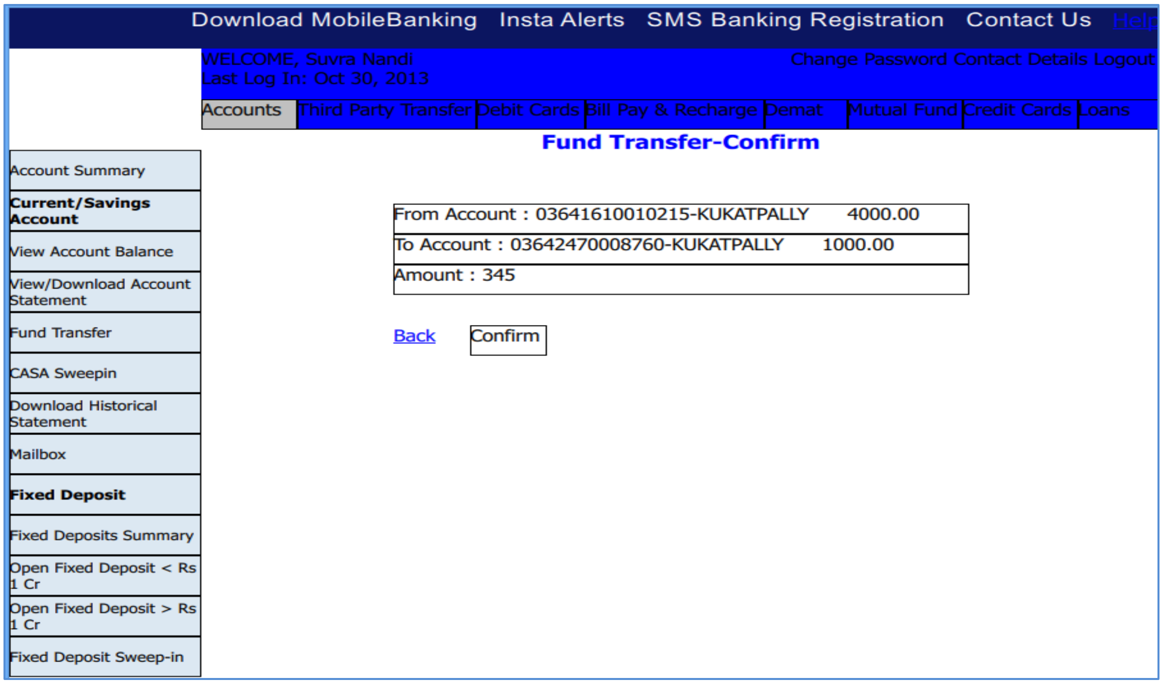


Figure 54: Qt QML Implementation Screen - Fund Transfer Confirm

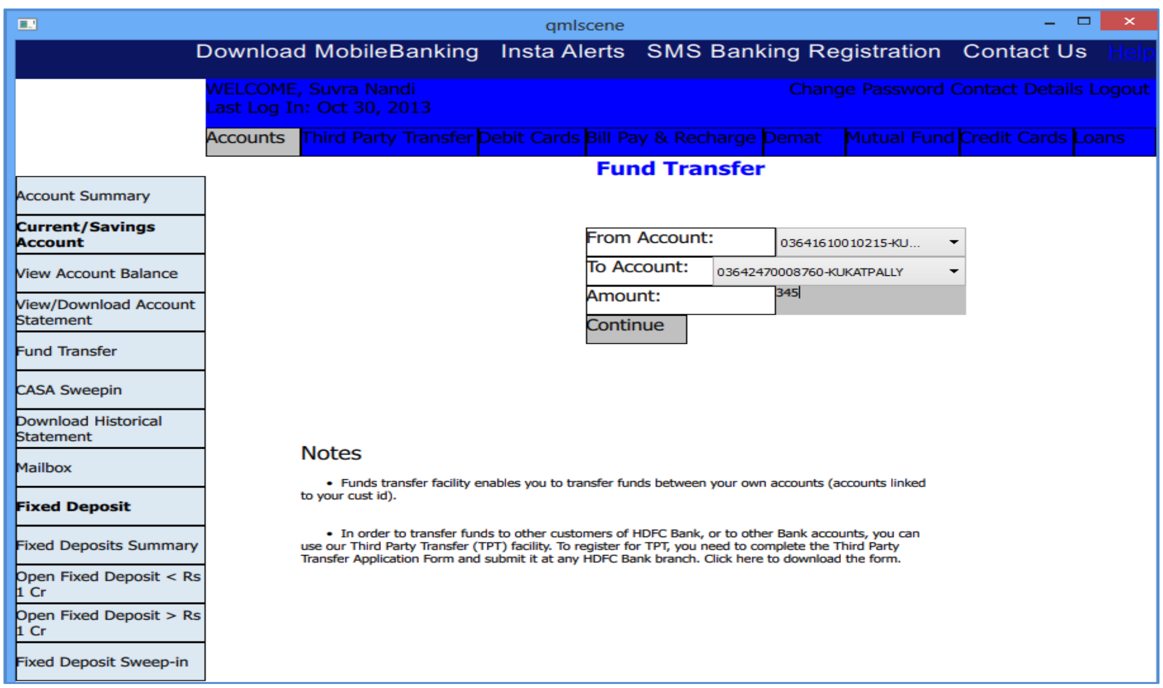


Figure 55: Qt QML Implementation Screen - Fund Transfer - after navigating back through "Back" link

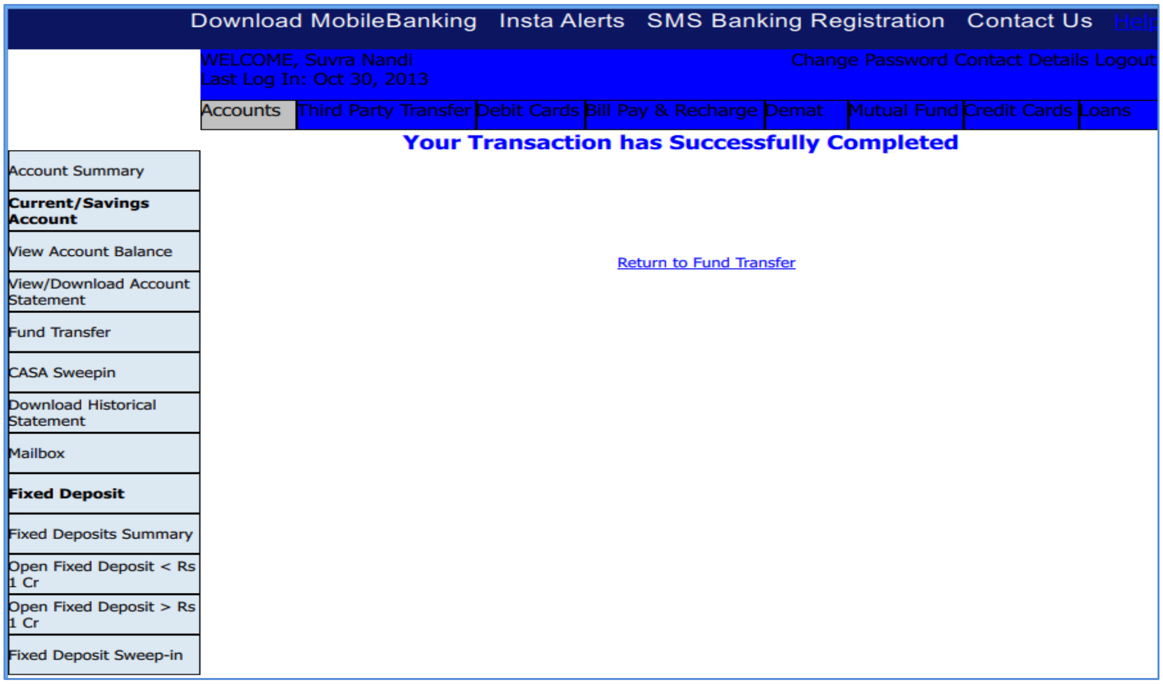


Figure 56: Qt QML Implementation Screen - Transaction Success Page

2.8.5 Evaluation through Implementation Experience

QT is a cross-platform desktop application design framework.

QML (Qt Meta Language or Qt Modeling Language) is a JavaScript-based, declarative language for designing user interface–centric applications. It is part of Qt Quick, the UI creation kit developed by Nokia within the Qt framework.

QML elements can be augmented by standard JavaScript both inline and via included .js files. Elements can also be seamlessly integrated and extended by C++ components using the Qt framework.

QML is the language; the runtime is called Qt Declarative.

a) Implementation Notes

- QML is a declarative language
- Qt Quick provides inbuilt Layout, which provides resizing facility too. Layouts are of type RowLayout, ColumnLayout, GridLayout.
- Anchor based layouts in QML are easy to implement.
- Dynamic and Flexible positioning with Binding expressions are possible.

- Positioning with Binding has the advantage that the values will automatically be updated as the dependencies of the bindings change. For example, the width of one Rectangle might depend on the width of the Rectangle next to it.

- Qt Quick supports for Right-to-Left positioning. This feature supports Right-to-Left language.

- Layout Mirroring is supported : The LayoutMirroring attached property is used to horizontally mirror Item anchors, positioner elements (such as Row and Grid) and views (such as GridView and horizontal ListView). Mirroring is a visual change: left anchors become right anchors and positioner elements like Grid and Row reverse the horizontal layout of child items. Layout Mirroring is provided as a convenience for easily implementing right-to-left support for existing left-to-right Qt Quick applications.

- QT Provides vary good support for Stack based Page Navigation. Methods available for PageStack are clear, push, pop, find, replace. Page is navigated forward via push method and navigated back by pop method.

- Provides support for Custom Styling and Themes

- The styles provided by Qt are not complete on their own. Developers can create custom style as per the need and use that in QML.

- Positioners (like Row) are available for general positioning number of items. It does not support item resize

- QML doesn't provide basic UI components like button, combobox etc. Everything needs to be created by developer by creating Rectangle components. Basic QML components available are : Rectangle, Text, TextEdit, Image. This requires lot of extra coding.

b) When QT Qml should be used?

Developers may choose QT as GUI Design framework when :

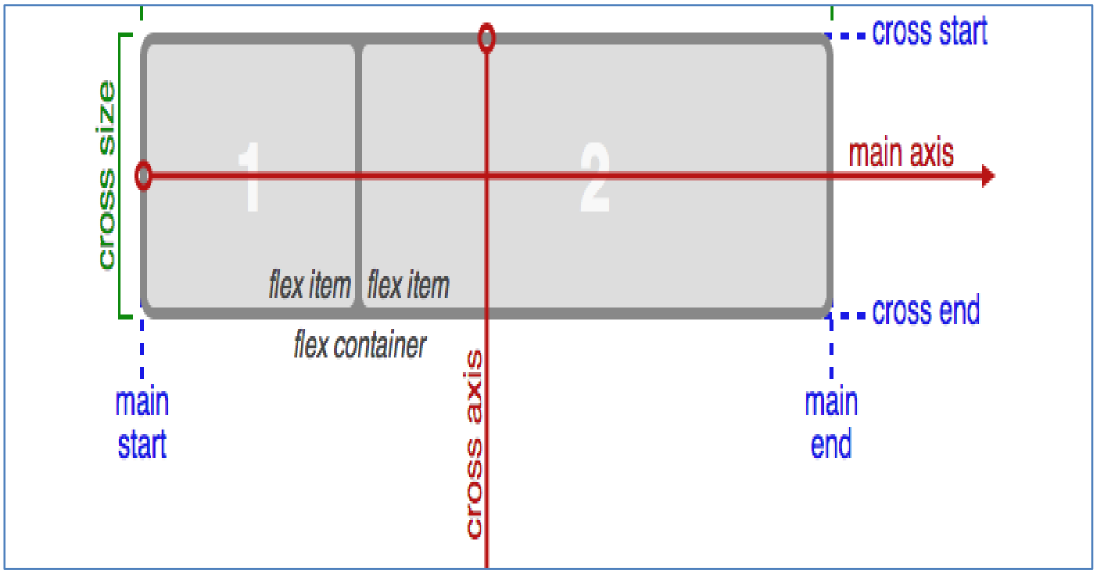
- Rich set of GUI building blocks are to be created: QML elements shipped with Qt are a sophisticated set of building blocks, graphical (e.g., rectangle, image) and behavioral (e.g., state, transition, animation). These elements can be combined to build components ranging in complexity from simple buttons and sliders, to complete internet-enabled programs.
- Cross-Platform development is preferred

- C++ programming is preferred : QT framework can be used with C++ programming also, other than QML

2.9 Study outcome of CSS3 FLEXBOX Framework

2.9.1 Layout Support in CSS3 FLEXBOX

If regular layout is based on both block and inline flow directions, the flex layout is based on "flex-flow directions". Following figure explains the main idea behind the flex layout. (Reference[7])



Basically, items will be laid out following either the main_axis (from main_start to main-end) or the cross axis (from cross-start to cross-end). Components in Flexbox layout can be of two types : Flex Container and Flex Item, on which different properties may be set as described below.

Properties:

- display: flex | inline-flex; (Applies to: parent flex container element)

This defines a flex container; inline or block depending on the given value. Thus, it enables a flex context for all its direct children. Possible values are flex, inline-flex.

- flex-direction (Applies to: parent flex container element)

This establishes the main-axis, thus defining the direction flex items are placed in the flex container. Possible values are row, row-reverse, column, column-reverse.

- flex-wrap (Applies to: parent flex container element)

This defines whether the flex container is single-line or multi-line, and the direction of the cross-axis, which determines the direction new lines are stacked in. Possible values are nowrap, wrap, wrap-reverse.

- flex-flow (Applies to: parent flex container element)

This is a shorthand \`flex-direction\` and \`flex-wrap\` properties, which together define the flex container's main and cross axes. Default is \`row nowrap\`.

- justify-content (Applies to: parent flex container element)

This defines the alignment along the main axis. It helps distribute extra free space leftover when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line. Possible values are flex-start, flex-end, center, space-between, space-around.

- align-items (Applies to: parent flex container element)

This defines the default behavior for how flex items are laid out along the cross axis on the current line. Think of it as the justify-content version for the cross-axis (perpendicular to the main-axis). Possible values are flex-start, flex-end, center, baseline, stretch.

- align-content (Applies to: parent flex container element)

This aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis. Possible values are flex-start, flex-end, center, space-between, space-around, stretch.

Note: this property has no effect when the flexbox has only a single line.

- order (Applies to: child element / flex item)

By default, flex items are laid out in the source order. However, the order property controls the order in which they appear in their container. It takes integer as its value.

- flex-grow (Applies to: child element / flex item)

This defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.

If all items have flex-grow set to 1, every child will set to an equal size inside the container. If you were to give one of the children a value of 2, that child would take up twice as much space as the others.

- flex-shrink (Applies to: child element / flex item)

This defines the ability for a flex item to shrink if necessary. It takes any number, default is 1.

- flex-basis (Applies to: child element / flex item)

This defines the default size of an element before the remaining space is distributed. Default value is auto.

- flex (Applies to: child element / flex item)

This is the shorthand for flex-grow, flex-shrink and flex-basis. The second and third parameters (flex-shrink, flex-basis) are optional. Default is 0 1 auto.

- align-self (Applies to: child element / flex item)

This allows the default alignment or the one specified by align-items to be overridden for individual flex items.

2.9.2 Navigation Support in CSS3 FLEXBOX

CSS3 Flexbox is a layout and styling approach. Hence, it doesn't provide any specific support for Page Navigation. Navigation can be achieved in Flexbox by defining JavaScript Functions and calling them from Button/Link Action attribute. Here different pages can be treated as different "div" s which can be displayed or hidden as required. Below is a sample code snippet of Navigation Implementation in FlexBox:

JS Function for displaying/hiding DIVs

```
function showFundXferTranType() {

document.getElementById("centerPanel").style.display="none";

document.getElementById("accountBalance").style.display="none";

document.getElementById("fundXferTranType").style.display="block";

document.getElementById("fundXfer").style.display="none";

document.getElementById("fundXferConfirm").style.display="none";

document.getElementById("fundXferSuccess").style.display="none";

}
```

JS Function gets called from OnClick attribute of Button

```
…

<tr ><td ><button class="leftPanelButtons">View Cheque
Status</button></td></tr>

<tr ><td ><button class="leftPanelButtons"
onclick="showFundXferTranType()">Funds Transfer</button></td></tr>

<tr ><td ><button class="leftPanelButtons">CASA
Sweepin</button></td></tr>

…
```

2.9.3 Theme Support in CSS3 FLEXBOX

There are no standard built-in themes available in CSS3 Flexbox.

Custom themes can be created in CSS3 Flexbox by using CSS Stylesheets. Styles can be specified in CSS style classes, which need to be included in Flexbox code by <link> attribute:

```
<link rel="stylesheet" type="text/css" href="myBankTest.css" />
```

Below is a code snippet from “.css” file, which needs to be linked from the main code file :

```
…

#mainContainer{

display: -webkit-flex;

-webkit-flex-direction: column;

}

#topNarrowBar{

-webkit-justify-content: flex-end;

display: -webkit-flex;

width: 1010px;

height: 20px;

background: blue;

}

…
```

2.9.4 Sample Screens Implemented in CSS3 FLEXBOX

Few pages of a sample Bank Website was built using CSS3 FLEXBOX layout module, taking help of different inbuilt layout managers.

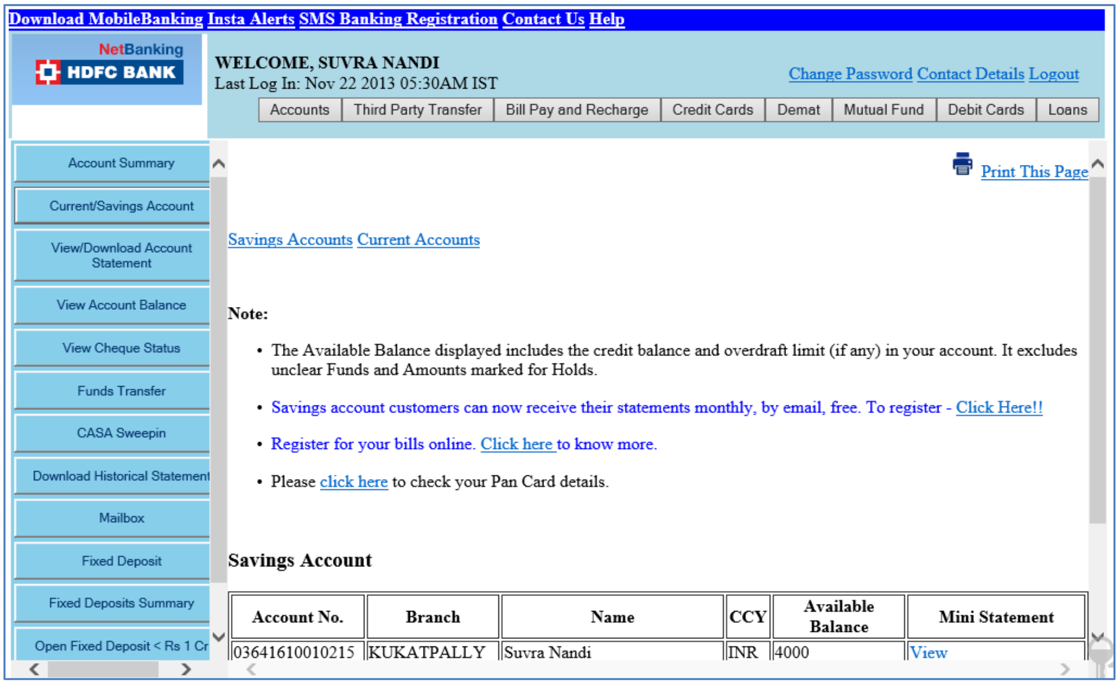


Figure 57: CSS3 FlexBox Implementation Screen - Account Summary

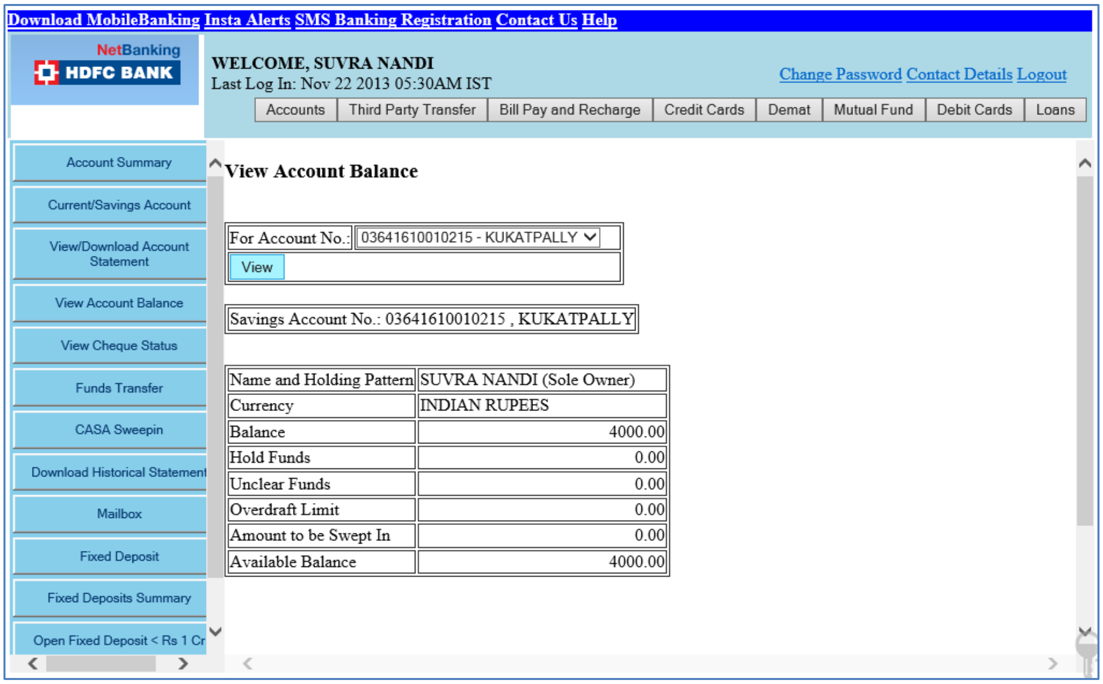


Figure 58: CSS3 FlexBox Implementation Screen - View Account Balance

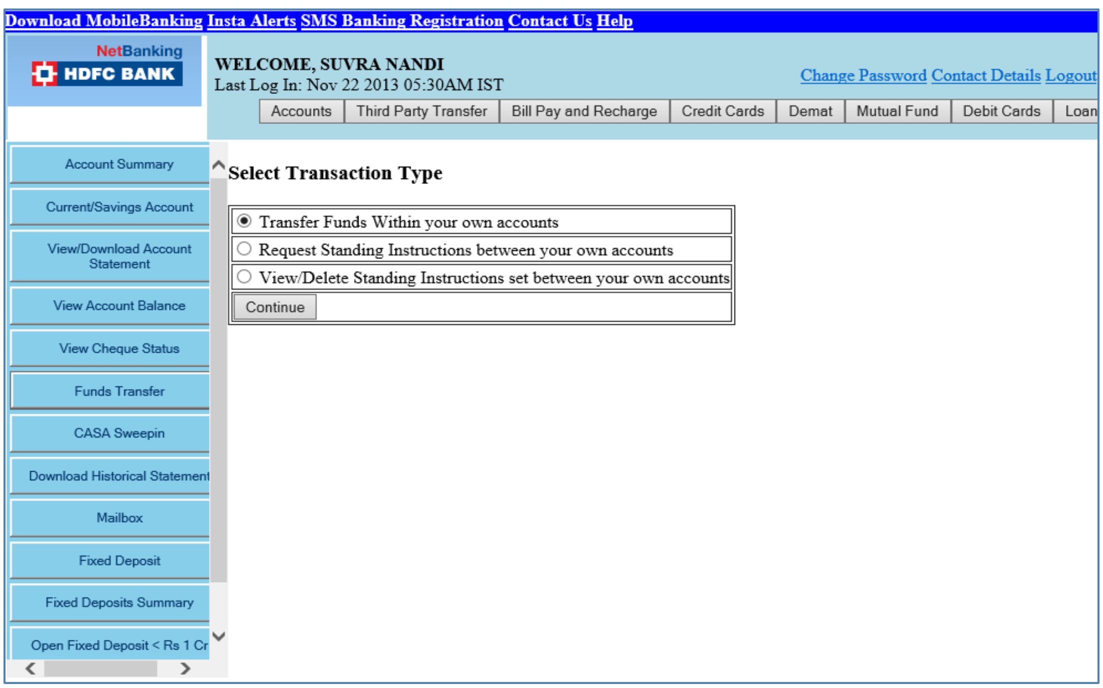


Figure 59: CSS3 FlexBox Implementation Screen - Fund Transfer - Tran Type

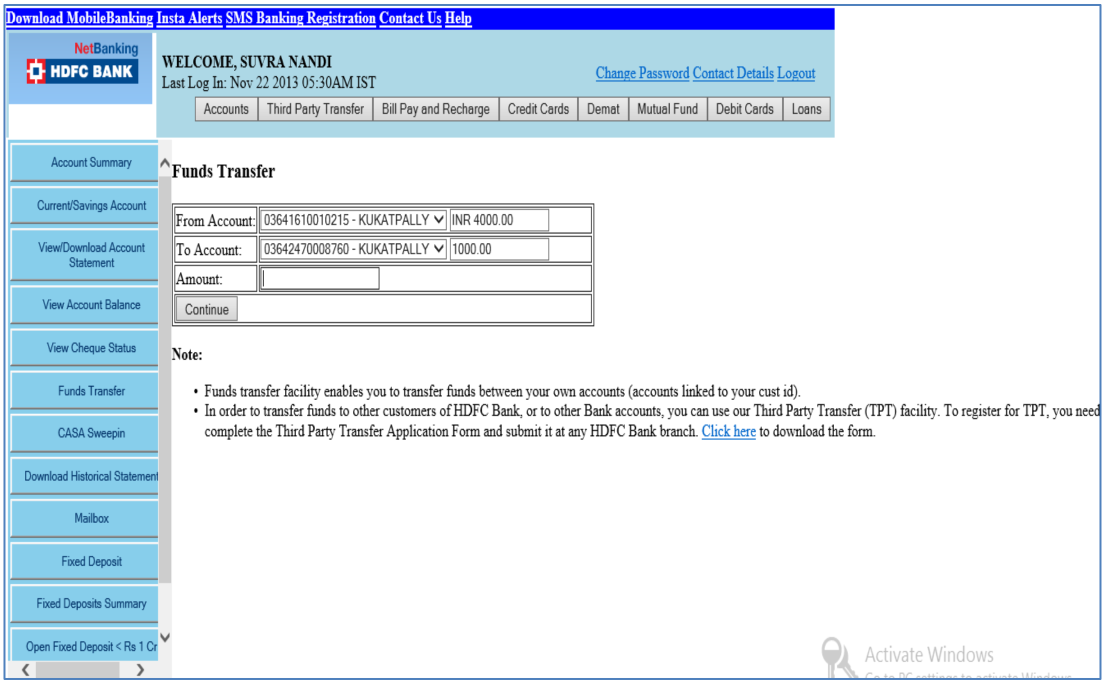


Figure 60: CSS3 FlexBox Implementation Screen - Fund Transfer

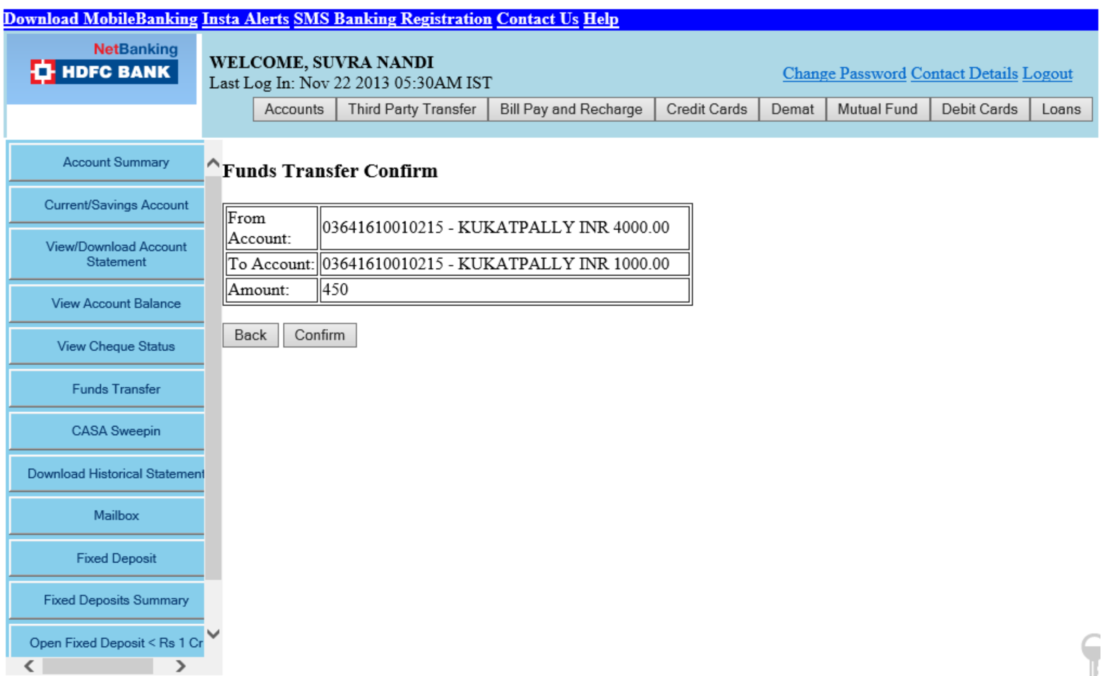


Figure 61: CSS3 FlexBox Implementation Screen - Fund Transfer – Confirm

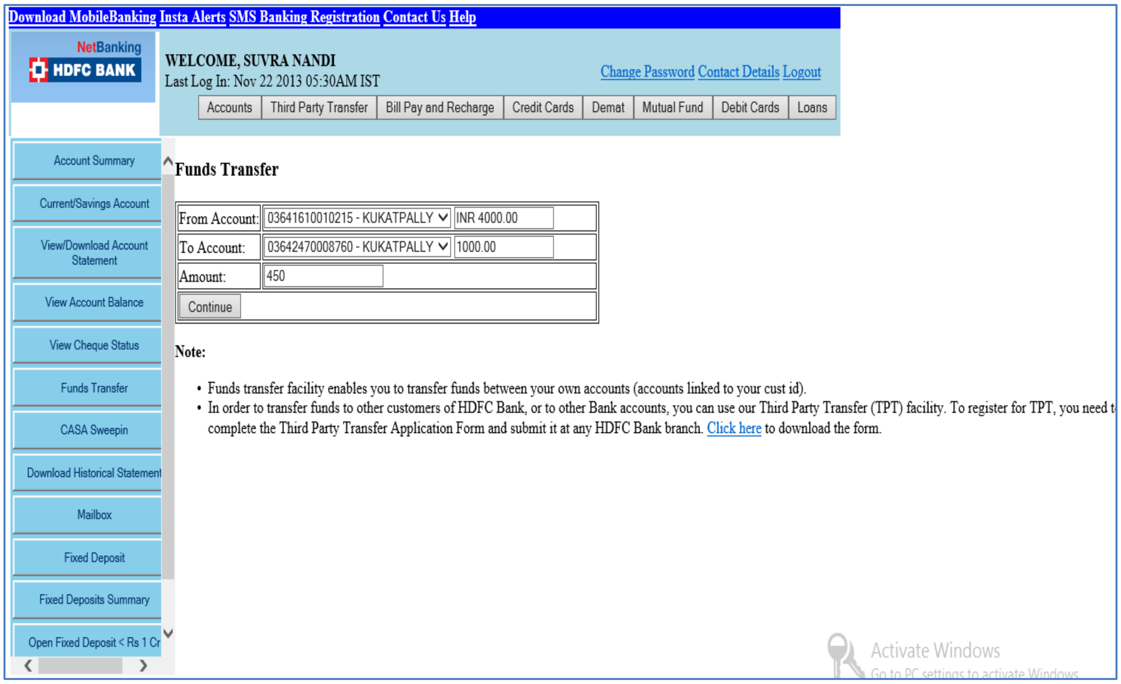


Figure 62: CSS3 FlexBox Implementation Screen - Fund Transfer after Back Navigation

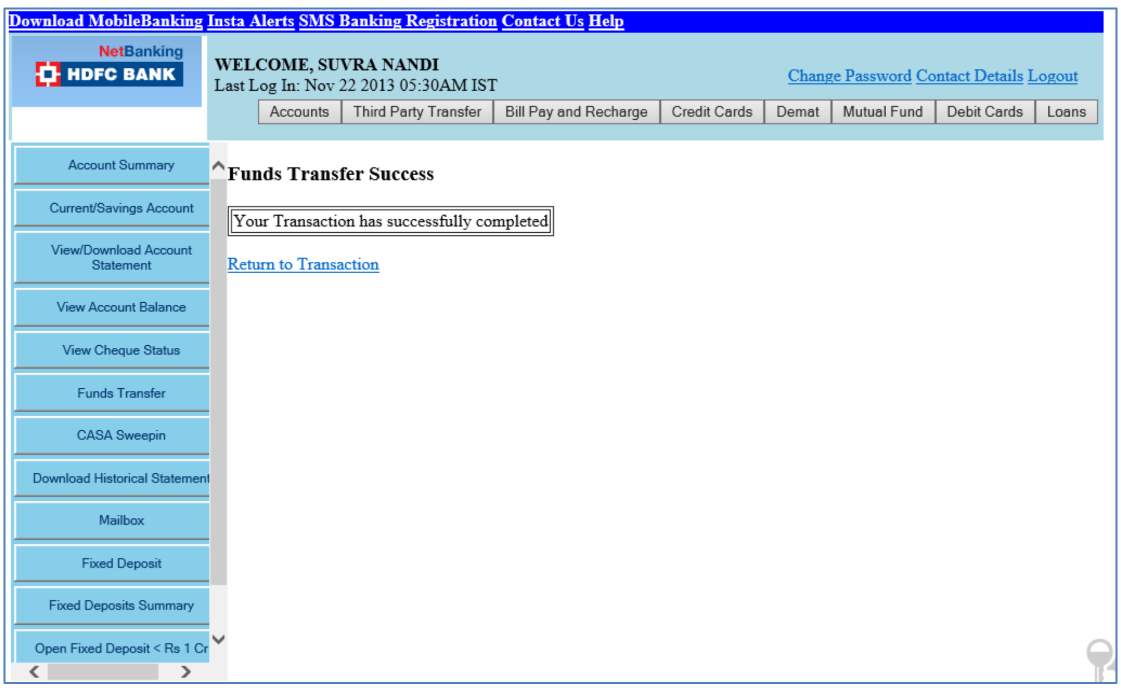


Figure 63: CSS3 FlexBox Implementation Screen - Fund Transfer Success

2.9.5 Evaluation through Implementation Experience

The Flexbox Layout (Flexible Box) module (currently a W3C Candidate Recommendation) aims at providing a more efficient way to lay out, align and distribute space among items in a container, even when their size is unknown and/or dynamic (thus the word "flex").

The main idea behind the flex layout is to give the container the ability to alter its items' width/height (and order) to best fill the available space (mostly to accommodate to all kind of display devices and screen sizes). A flex container expands items to fill available free space, or shrinks them to prevent overflow.

a) Implementation Notes

- Components can be laid out in any direction using FlexBox layout.
- FlexBox layout components can have their visual order reversed or rearranged.
- Components can flex their sizes to respond to available space.
- Components can be aligned with respect to their container or with respect to each other.
- CSS3 FlexBox properties are hard to implement, as they don't always work with all HTML components, e.g. tables.
- Because flexible boxes use a different layout algorithm, some properties do not make sense on a flex container:
  - ✓ column-* properties of the Multicol module have no effect on a flex item.
  - ✓ float and clear have no effect on a flex item. Using float causes the display property of the element to compute to block.
  - ✓ vertical-align has no effect on the alignment of flex items.

b) When CSS3 FLEXBOX should be used?

Developers may choose CSS3 FlexBox layout model when they need:

- Specific positioning of Components in Container: CSS3 Flexbox is best suited for targeting the position of child elements and their spatial relationship within a parent element.

- Source-ordering independence: In the flex layout model, the children of a flex container can be laid out in any direction, and can “flex” their sizes, either growing to fill unused space or shrinking to avoid overflowing the parent.
- Two Dimensional Design is required – here both horizontal and vertical alignment of the children can be easily manipulated. Nesting of these boxes (horizontal inside vertical, or vertical inside horizontal) can be used to build layouts in two dimensions.
- Responsive Layout support is required

3. Framework Comparison

Below is the Framework Comparison Table for various GUI Design Frameworks targeting different attributes as placed along the rows of the table. The attributes are all selected depending upon the usage requirements in most of the web applications now-a-days. Mentioned below are the reasons of choosing the few listed attributes for comparing among existing layouts:

- Few of the most common and demanding layouts are Form Layout, Menu Layout, Multi-Column Layout etc.
- Browser Dependency is one of the pain areas in current frameworks and web page design
- Ease of Deployment, Performance, Page Quality are some important factors which decide the adoption of the particular frameworks in user community
- Liquid or Fluid layout is part of responsive layout design which is gaining popularity with the advent of multi-size devices and their increased usage

Framework /Attribute	**SWING**	**SWT**	**JSF**	**WINDOWS STORE APPS**	**SITEMESH**	**QT with QML**	**CSS3 FLEXBOX**
Support for Form Layout	BoxLayout can be used to create forms. No separate layout component is available for Forms.	FormLayout class is available.	<h:panelGrid> with <h:panelGroup> can be used.	Single and Multi-Column Form Layouts are available.	Forms can be created as HTML components are used. However, it doesn't allow POST on the requests by default.	It is hard to create forms using QT QML. Labels and TextEdits can be used to create forms in a custom way.	Form Can be created with HTML

Framework /Attribute	SWING	SWT	JSF	WINDOWS STORE APPS	SITEMESH	QT with QML	CSS3 FLEXBOX
Support for Menu Layout	BoxLayout can be used to create menus. No separate layout component is available for Menus.	FormLayout can be used to create Menu, SubMenu, CheckMenu, PopupMenu etc.	JSF PrimeFaces and RichFaces libraries provide support for Menu creation. Basic JSF doesn't provide support for this.	Can create Menus with <page:bottomAppBar>, <AppBarButton:Flyout>, <MenuFlyout>, <MenuFlyoutItem> etc	There are issues with Menu Layout in Sitemesh. If a page has menu and layout coming from different pages and combining them into one page, Layout overrides menu.	Like form, menu can be created from basic elements . No built-in component is available for Menu.	Custom Menu Layout can be created using CSS and HTML.
Support for Multi-Column Page Layout	Custom Layout need to be created. It is easy to achieve in SWING. No inbuilt layout components available.	Can be achieved by nesting layout managers. No inbuilt layout components available.	Custom JSF Renderer needs to be built. No inbuilt layout components available.	Windows Store apps using JavaScript in Windows 8, introduces support for the CSS Multi-column Layout Module.	Custom Decorator Layouts can be created and applied to similar pages by applyLayout Tag.	ColumnLayout can be used here.	Can be achieved in CSS3, by changing attribute values like display, max-height for column etc. IT is hard to achieve in FlexBox.

Framework /Attribute	SWING	SWT	JSF	WINDOWS STORE APPS	SITEMESH	QT with QML	CSS3 FLEXBOX
Support for Liquid/Fluid Layout	BorderLayout provides liquid layout by default.	Standards layouts in SWT support Window resizing.	PrimeFaces Layout can respond to expand, collapse, close and resize events of each layout unit with Ajax listeners.	Has good support for Liquid Layouts.	Has support for Liquid Layouts.	QML Positioners does not support item resize. Other layout constructs in Qt QML has support for this.	Has good support for Window resizes. But the contents maintain their position relative to each other
Browser Support	All major java based browsers with Javascript enabled are supported by SWING. JXBrowser library of SWING provides support for IE and Mozilla in Windows OS machines.	SWT Browser Widget is available, which is used to display native HTML components. By default, Browser instances created with style SWT.NONE uses following renderers: IE 5.5 for Windows 2000, WebKit for Mac OSX. Default renderer could be IE, Webkit or Mozilla.	All major browsers are supported in general.	Windows Store Apps are out of browsers. These apps are known as non-browser apps.	All major browsers are supported as it is HTML.	All major browsers are supported. IE 9.0 onwards	IE 11.0, Firefox 25.0 – Partial, Chrome 31.0

Framework /Attribute	SWING	SWT	JSF	WINDOWS STORE APPS	SITEMESH	QT with QML	CSS3 FLEXBOX
Deployment	The easiest deployment method for desktop Java applications is Java Web Start. This involves serving the application files from a web server. Apart from that other options like Building executable jar, Building startup files are also possible.	SWT Deployment is somewhat tedious job. Need SWT Designer to be installed on Eclipse 3.1. Executable JAR file can be created by making proper entries in manifest.mf file , Putting required library files from plugins folder and running Eclipse File->Export->Jar. This approach doesn't always work with all versions on eclipse.	JSF application can be deployed by creating a ".war" file and placing it in Tomcat webapps directory and running it through Tomcat server. Need below installation for JSF application: Eclipse Juno, Maven 2.2.1, Tomcat 6, JDK 1.6.	Visual Studio Express 2013 is enough for building, testing and deploying Windows Store Apps. It includes Windows SDK, Blend for Visual Studio and Project Templates. Only thing is it is not free and needs to be installed on Windows 8.1.	SiteMesh application needs to be deployed on Tomcat server. Need below installations for SiteMesh Application: Eclipse Juno, SiteMesh 2.4.1 jar, Tomcat7, JDK 1.6	To deploy an application that uses QML, the QML runtime must be invoked by the application. This is done by writing a Qt C++ application that loads the QQmlEngine by either: Loading the QML file through a QQuickView instance, or Creating a QQmlEngine instance and loading QML files with QQmlComponent	Easiest Deployment. HTML files can be directly run on Browsers.

Framework /Attribute	SWING	SWT	JSF	WINDOWS STORE APPS	SITEMESH	QT with QML	CSS3 FLEXBOX
Browser Execution Code & Page Quality	As SWING is pure Java, “.class” files are generated by Java compiler for execution. Supports Native and Pluggable Look and Feel. SWING pages look standard.	Same as SWING. As SWT is Java based, class files are generated for execution. Pages designed using SWT have native Look and Feel.	XHTML – (Extensible HyperText Markup Language), CSS, JS, JAVA. Provides standard look and feel.	XAML – (Extensible Application Markup Language) & Visual C++. Pages designed have very smooth and sophisticated Look and Feel.	JSP, CSS & JS are used to build pages. XML is used as Decortaors mapping. Can provide standard Look and Feel. Used mostly for consistent looking pages.	QML – (QT Markup Language or QT Modeling Language) is used for Page building. Page quality is not so good. Pages built have native look and feel. Components are also not so smooth.	HTML , CSS & JS is used for pages. Page Quality is good and components are more flexible compared to other frameworks.

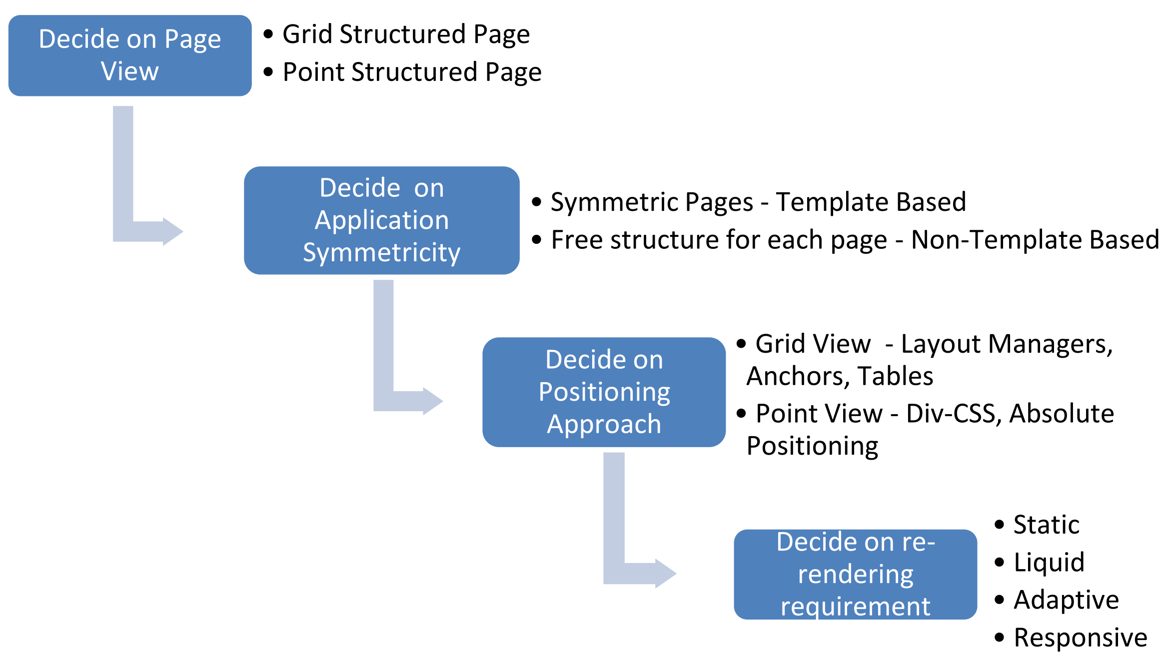
Framework /Attribute	SWING	SWT	JSF	WINDOWS STORE APPS	SITEMESH	QT with QML	CSS3 FLEXBOX
Performance	SWING Startup is a bit slower, as applets take quite some time to get ready. The initial overhead is due to slow code at various places of JDK. Other than initial startup, SWING provides standard performance level.	Same performance behavior as SWING, because SWT too encounters Applet.	In general, JSF provides good performance on Page Load and component rendering. However, using different JSF libraries like Richfaces etc. may reduce performance as few of the component are very costly, compared to plain JSF. Best practices need to be followed while using AJAX and framework libraries to get standard performance.	The XAML Page performance of Windows Store App can be analyzed by a Visual Studio Debug option called "Start Performance Analysis". It marks all the expensive lines in a code. Details of this utility can be viewed at : http://blogs.msdn.com/b/visualstudioalm/archive/2013/02/27/how-to-profile-a-xaml-windows-store-app.aspx	SiteMesh performance is not very good, as it has a Decorator pattern. Decorating using Meta Tags increase Page Load Time for simple pages too. Also, SiteMesh stores the entire content of HTML body into memory before it decorates it. It causes serious performance issues for large pages.	In general, QT QML provides standard performance on Page Load and Page Rendering etc. There are very good documentation on Performance Best Practices by which performance can be significantly increased. There are performance measuring engine too in QT. Performance details for QT QML can be viewed at the below link: http://qt.developpez.com/doc/5.0-snapshot/qtquick2-performance/	CSS3 FlexBox has performance issues on Page Load and component rendering when scrolling though Pages. Complex pages perform worse. Using alternate CSS component performs better.

4. Layout Design Patterns – Step by Step Approach

A design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations. Patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.

Based on the implementation experience and study of several UI design frameworks, we are able to establish a step-by-step approach towards achieving a design pattern in terms of positioning elements on a page. (References[41, 42, 1, 6, 8, 9, 10])

The layout design decision making steps can be depicted by the below flow-diagram:



4.1 Page View Patterns

A page can be perceived by a developer as a set of rows and columns or Grids, while another developer may look at it as set of point. Depending upon this perception differences two page view patterns are identified as follows:

4.1.1 Grid Based Pages

A grid is a set of guidelines, able to be seen in the design process and invisible to the end-user/audience, for aligning and repeating elements on a page. A page layout may or may not stay within those guidelines, depending on how much repetition or variety the design style in the series calls for.

Example

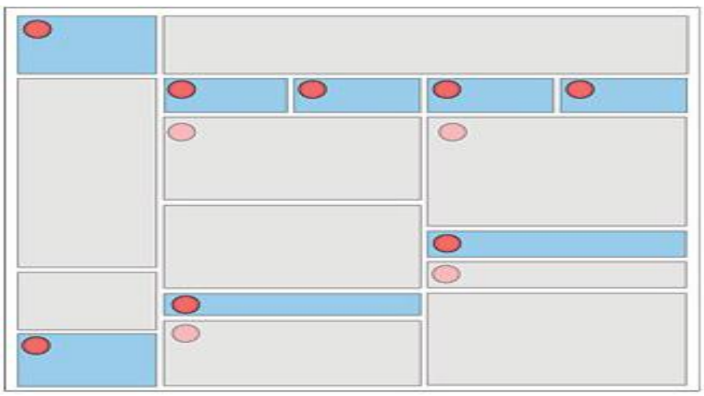


Figure 64: Page View Patterns - Probable Grid Structure - 1

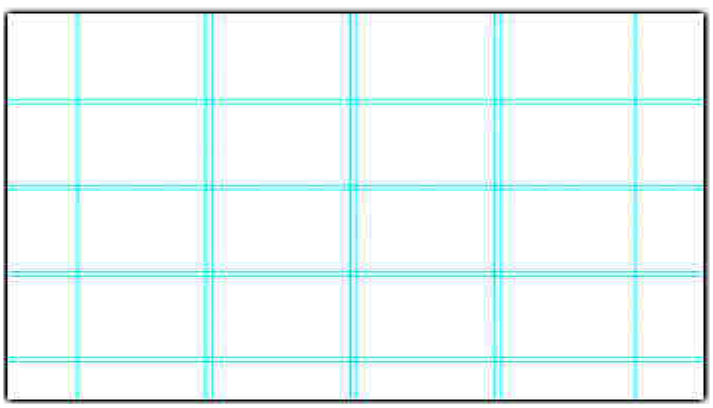


Figure 65: Page View Patterns - Probable Grid Structure – 2

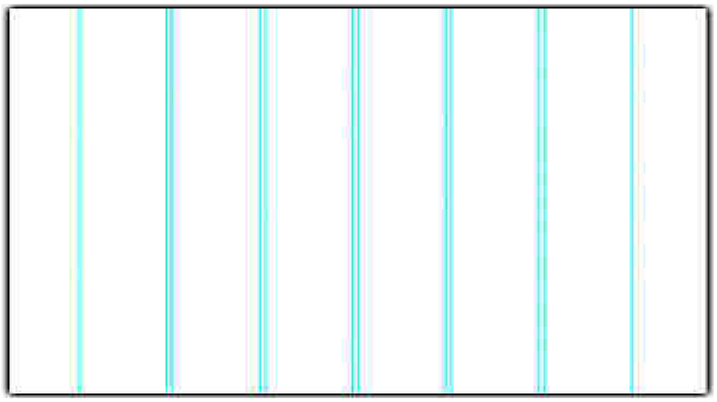


Figure 66: Page View Patterns - Probable Grid Structure – 3

4.1.2 Point Based Pages

The whole page is just considered as set of individual points with unique (x,y) coordinate values for each point. Each location is associated to one point on page.

Figure 67: Page View Patterns - Point based page view - 1

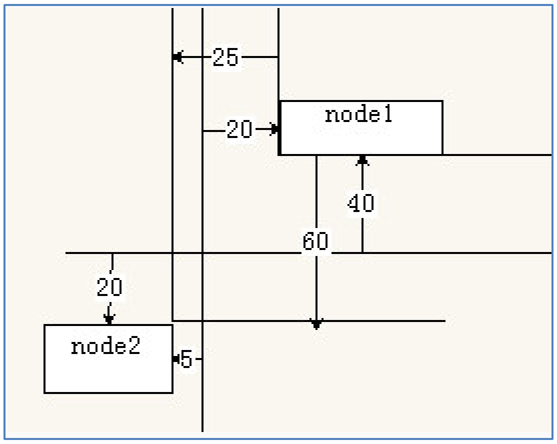


Figure 68: Page View Patterns - Point based page view – 2

4.2 Patterns in Application Symmetricity

This pattern will decide the symmetric nature of all pages in an application. If the symmetric look throughout the application is preferred, Template based layout structure is recommended. If there is no requirement of such symmetricity in the application, then its pages can be non-template based and free to be structured in any way.

4.2.1 Template Based Page Structure

In this approach Frameworks provide provision for creating generalized templates or decorative pages, which in turn can be applied to various pages. This pattern is most useful when any application has a fixed pattern for all the pages. In that case the fixed page elements code can be written only once in the template and reused in all subsequent pages.

Example

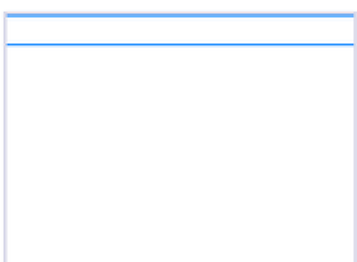


Figure 69: Template based Page Structure - One Column Page Layout

Figure 70: Template based Page Structure - Two Column Page Layout

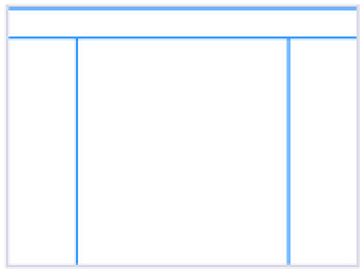


Figure 71: Template based Page Structure - Three Column Page Layout

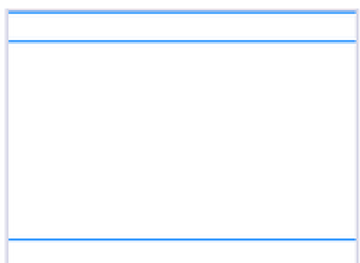


Figure 72: Template based Page Structure - Layout with Footer

Templating normally works with Declarative programming. Works well with window resize.

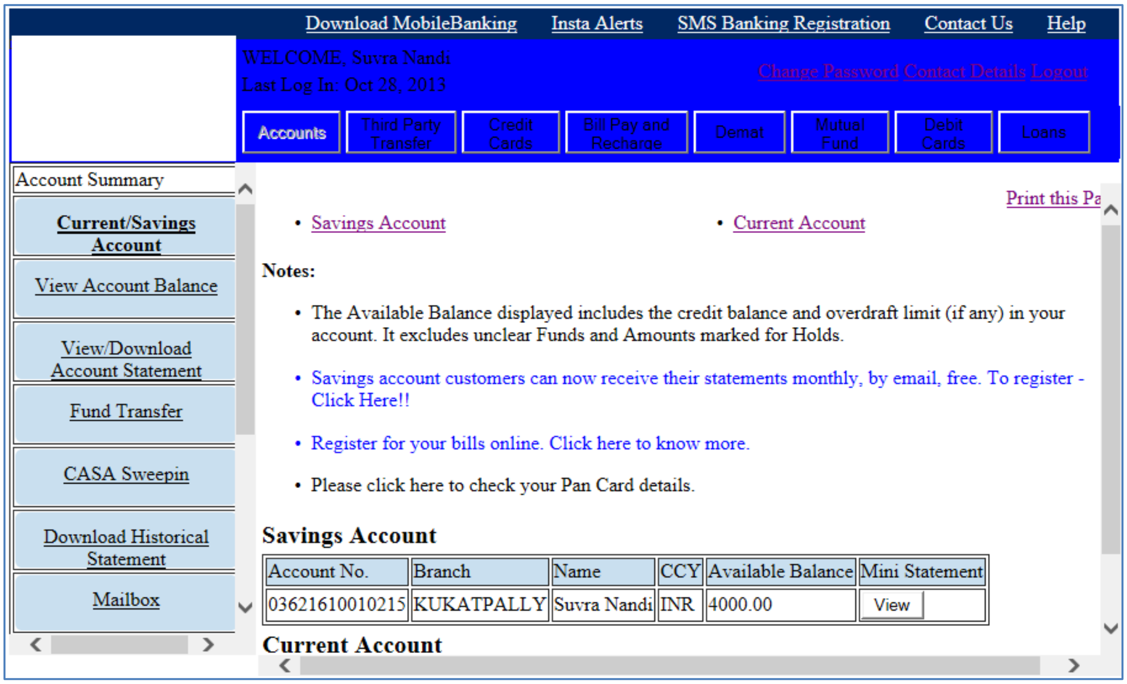
Framework Examples under this category are: JSF Facelets, SiteMesh, and Apache Tapestry.

Example Code Snippet from SiteMesh Decorator:

```
<body>

<div id="leftPanel" style="width: 200px; height: 400px; overflow: scroll">

<table width="100%" border="1">

<tbody>
```

```
<tr><td>Account Summary</td></tr>

<tr><td><a href="accountSummary.jsp" class="buttonBold">Current/Savings
Account</a></td></tr>

<tr><td><a href="viewAccountBalance.jsp" class="button">View Account
Balance</a></td></tr>

<tr><td><a href="" class="button">View/Download Account
Statement</a></td></tr>

<tr><td><a href="fundXferTranType.jsp" class="button">Fund
Transfer</a></td></tr>

<tr><td><a href="" class="button">CASA Sweepin</a></td></tr>

<tr><td><a href="" class="button">Download Historical Statement</a></td></tr>

<tr><td><a href="" class="button">Mailbox</a></td></tr>

<tr><td><a href="" class="buttonBold">Fixed Deposit</a></td></tr>

<tr><td><a href="" class="button">Fixed Deposits Summary</a></td></tr>

<tr><td><a href="" class="button">Open Fixed Deposit &lt Rs 1 Cr</a></td></tr>

<tr><td><a href="" class="button">Open Fixed Deposit &gt  Rs 1 Cr</a></td></tr>

<tr><td><a href="" class="button">Fixed Deposit Sweep-in</a></td></tr>

</tbody>

</table>

</div>

</div>

<decorator:body /> <!--This section will be replaced by original body code->

</body>
```

Below is the screen diagram as result of above code snippet, in SiteMesh. The left and top panel are part of the template, with body replacing <decorator:body/> tag of the code in all screens.



4.3 Patterns in Positioning Approach

Positioning approach varies depending upon different perspectives of page views – the Grid based implementations differ significantly from Point based implementations. Layout Managers, Anchor based positioning, Table based positioning are all targeted to place a component on cells of a grid, which is assumed to have constructed the page. Whereas DIV-CSS based positioning and Absolute positioning are there for point based implementations where the whole page is perceived to be a collection of points. Below are the details of each approach:

4.3.1 Layout Managers or Class Based Layout

In this approach, framework provides several inbuilt classes that need to be instantiated for achieving that particular layout.

This layout approach supports both Imperative and Declarative programming, Window Resize.

Framework Examples under this category are: SWING, Vaadin (imperative), flex (declarative), SWT, Windows Store Apps, JSF Basic Tags, wxWidgets, JQuery.

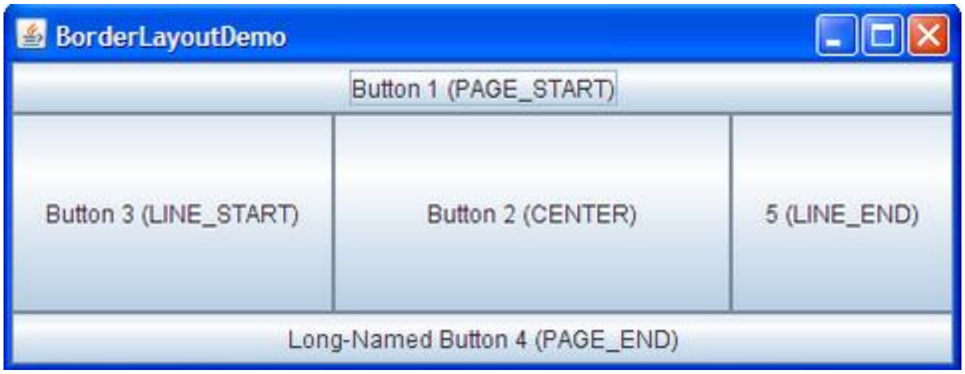
e.g.

- SWING provides BorderLayout, FillLayout, RowLayout, CardLayout, GridLayout, GridBagLayout etc. for different kinds of Layout structures.
- VAADIN provides VerticalLayout, HorizontalLayout, HorizontalSplitPannel, FormLayout, GridLayout, TabSheet etc. for different kinds of layout structure.

Example Code Snippet from SWING:

```
...//Container pane = aFrame.getContentPane()...

JButton button = new JButton("Button 1 (PAGE_START)");

pane.add(button, BorderLayout.PAGE_START);

//Make the center component big, since that's the

//typical usage of BorderLayout.

button = new JButton("Button 2 (CENTER)");

button.setPreferredSize(new Dimension(200, 100));

pane.add(button, BorderLayout.CENTER);

button = new JButton("Button 3 (LINE_START)");

pane.add(button, BorderLayout.LINE_START);

button = new JButton("Long-Named Button 4
(PAGE_END)");

pane.add(button, BorderLayout.PAGE_END);

button = new JButton("5 (LINE_END)");

pane.add(button, BorderLayout.LINE_END);
```

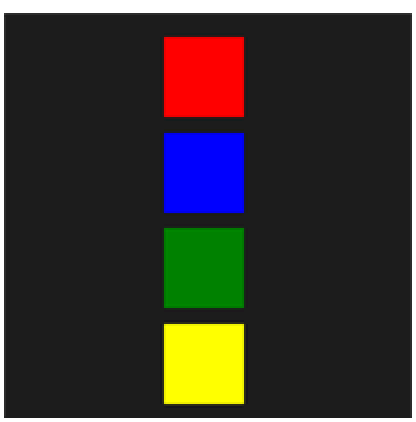
The above code generates below structure:



Example Code Snippet from WINDOWS STORE APP:

```
<StackPanel Margin="20">

<Rectangle Fill="Red" Width="50" Height="50" Margin="5" />

<Rectangle Fill="Blue" Width="50" Height="50" Margin="5" />

<Rectangle Fill="Green" Width="50" Height="50" Margin="5" />

<Rectangle Fill="Purple" Width="50" Height="50" Margin="5" />

</StackPanel>
```

StackPanel is a layout panel that arranges child elements into a single line that can be oriented horizontally or vertically**.** The above code generates below structure:



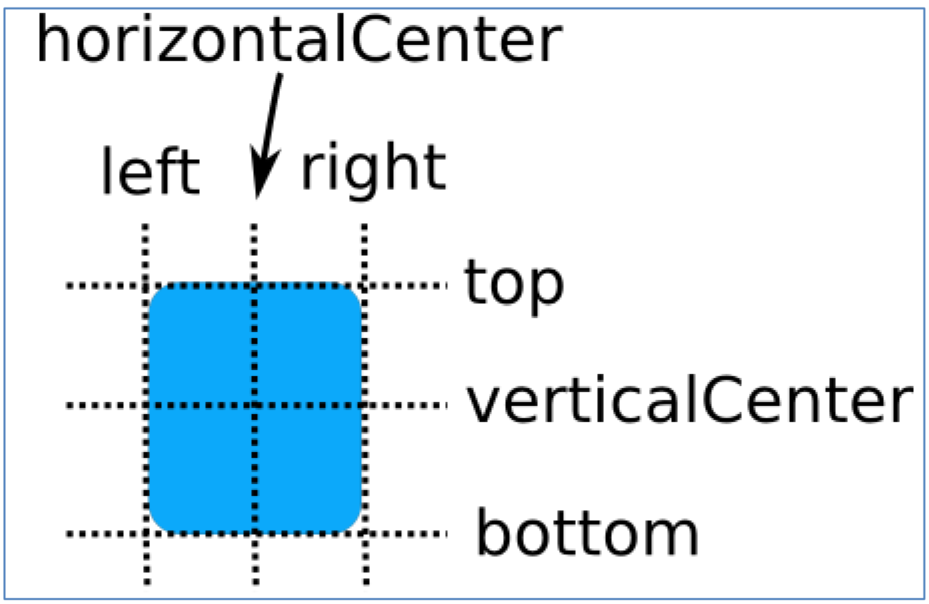
4.3.2 Anchor Based Layout

In this approach, Framework provides anchor based positioning facility for all the widgets. Here any component can be placed in any position, with respect to an existing component's position. In anchor based positioning, any widget is placed with respect to its direct parent and sibling. Anchor based positioning cannot be mixed with Absolute Positioning.

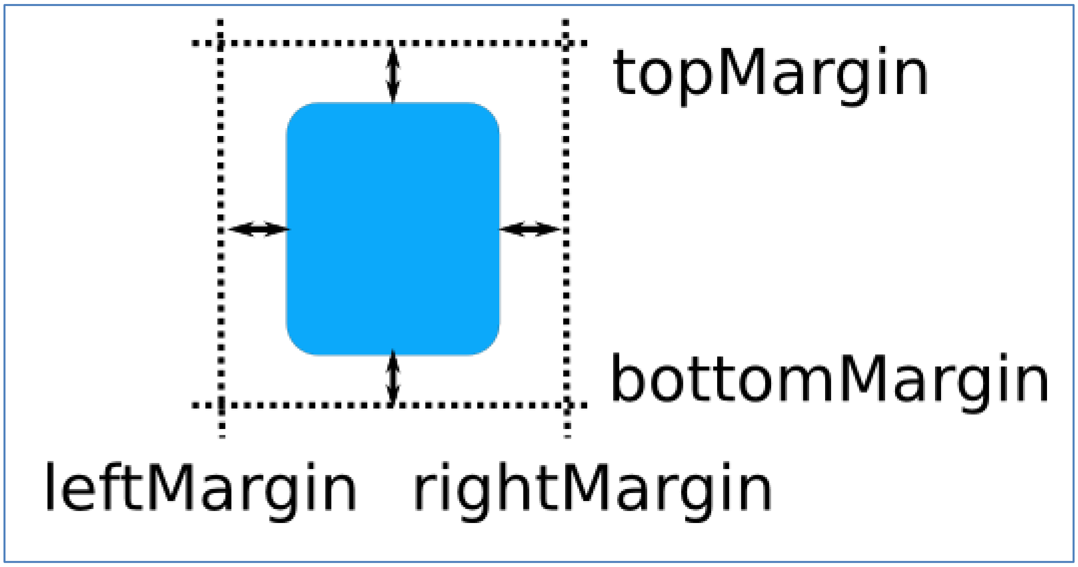
This approach is currently available with QT framework's QT Markup Language, which is a special kind of Declarative coding.

Framework Examples under this category are: QT with QML.

In Anchor based concept each item can be thought of as having a set of 7 invisible "anchor lines": left, horizontalCenter, right, top, verticalCenter, baseline, and bottom, as shown in the fig. below:



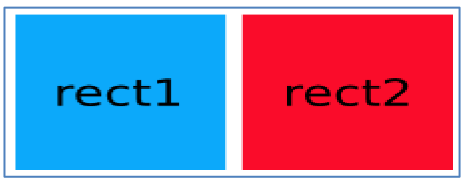
Anchor system also allows to specify margins and offsets. Margins specify the amount of empty space to leave to the outside of an item, while offsets allow to manipulate positioning using the center anchor lines. The visualization of this concept is provided here:



Example Code Snippet1 from QT-QML:

```
Rectangle { id: rect1; ... } Rectangle { id: rect2;
anchors.left: rect1.right; anchors.leftMargin: 5; ... }
```

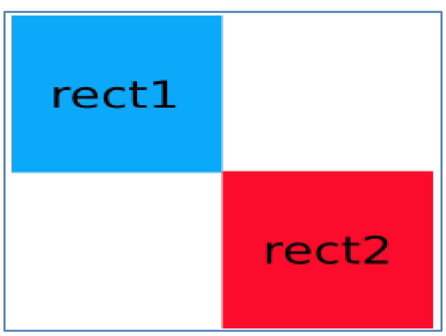
Code Snippet1 produces the below output:



Example Code Snippet2 from QT-QML:

```
Rectangle { id: rect1; ... } Rectangle { id: rect2;
anchors.left: rect1.right; anchors.top: rect1.bottom;
... }
```

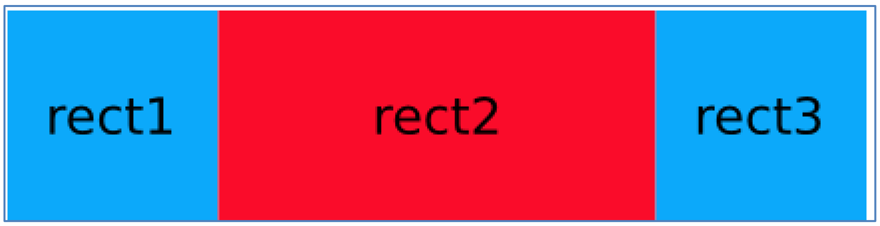
Code Snippet2 produces the below output:



Example Code Snippet3 from QT-QML:

```
Rectangle { id: rect1; x: 0; ... }

 Rectangle { id: rect2; anchors.left: rect1.right;
anchors.right: rect3.left; ... }

 Rectangle { id: rect3; x: 150; ... }
```

Code Snippet3 produces the below output:



4.3.3 Nested Table based Static layout

This approach involves creating table structure in HTML tags, like <table>, <tr>, <td> etc and nesting them as required. All styling attributes go with these tags only, instead of referring separate style sheet. However, this approach involves excess coding which slows down the development and raises maintenance cost. Table structure is very rigid and there is very little room for change without affecting the underlying structure.

This approach is part of HTML coding and hence declarative.

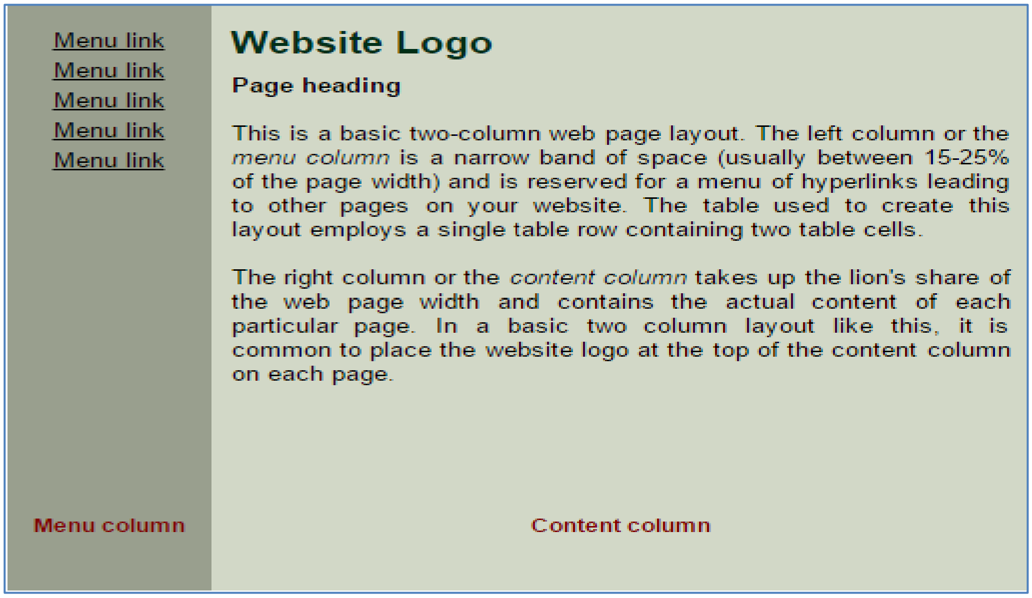
Framework examples under this category are: CSS3 FlexBox, or any HTML based UI design Frameworks.

Example Code Snippet from normal HTML:

```
<html>

<head>

<title>WEB PAGE TITLE GOES HERE</title>

</head>

<body style="margin: 0px; padding: 0px; font-family: 'Trebuchet
MS',verdana;">

<table width="100%" style="height: 100%;" cellpadding="10"
cellspacing="0" border="0"><tr>
```

```
<!-- =========== LEFT COLUMN (MENU) ============= -->

<td width="20%" valign="top" bgcolor="#999f8e">

<a href="#">Menu link</a><br>

<a href="#">Menu link</a><br>

<a href="#">Menu link</a><br>

<a href="#">Menu link</a><br>

<a href="#">Menu link</a>

</td>

<!-- =========== RIGHT COLUMN (CONTENT) ============= -->

<td width="80%" valign="top" bgcolor="#d2d8c7">

<h1>Website Logo</h1>

<h2>Page heading</h2>

This is a basic two-column web page layout. The left column or the
<i>menu column</i> is a narrow band of space (usually between 15-25%
of the page width) and is reserved for a menu of hyperlinks leading to
other pages on your website. The table used to create this layout employs
a single table row containing two table cells.<br>

<br>

The right column or the <i>content column</i> takes up the lion's share of
the web page width and contains the actual content of each particular
page. In a basic two column layout like this, it is common to place the
website logo at the top of the content column on each
page.</td></tr></table>

</body>

</html>
```

Above code snippet produces the below structure:



4.3.4 DIV, CSS & JS based layout

HTML <div> elements can be used to build single column, as well as multi-column layouts. Both static and dynamic layouts can be easily achieved using <div> based approach. It requires less coding compared to table based approach and is easily maintainable too. DIV approach makes use of CSS stylesheets and is able to create absolute, relative & float positioning, as required. JavaScript Library is used to provide greater flexibility and dynamism as required.

DIV, CSS & JS based layout approach works mainly in declarative programming.

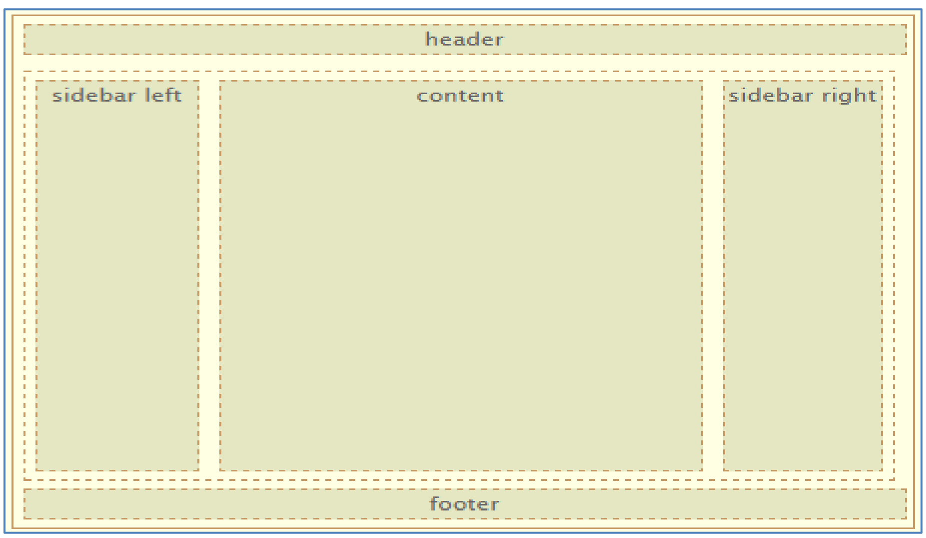
Framework Examples under this category are: JQuery, CSS3 FlexBox , Apache Tapestry. Flexbox often supports Responsive layouts as depicted in next section

Example Code Snippet for DIV-CSS based layout:

```
#wrapper- { width: 450px; margin: 0 auto; }

#header-,

#footer-,

#content-wrapper- { float: left; width: 440px; margin:5px;
display: block; clear: both; }

#sidebar-left-,
```

```
#sidebar-right- { float: left; width: 90px; height: 250px; }

#content- { float: left; width: 250px; height: 250px; }

<div id="wrapper-">

<div id="header-"> header  </div>

<div id="content-wrapper-">

<div id="sidebar-left-"> sidebar left </div>

<div id="content-"> content </div>

<div id="sidebar-right-"> sidebar right </div>

</div>

<div id="footer-"> footer </div>
```

Above code snippet produces below structure:



4.3.5 Absolute Positioning

In this approach, developers need to mention the absolute (x, y) position on screen for all the widgets. This is used when there is no layout available.

All the frameworks may use this pattern for positioning their widgets on page.

This approach is not flexible enough to resize components with window resize.

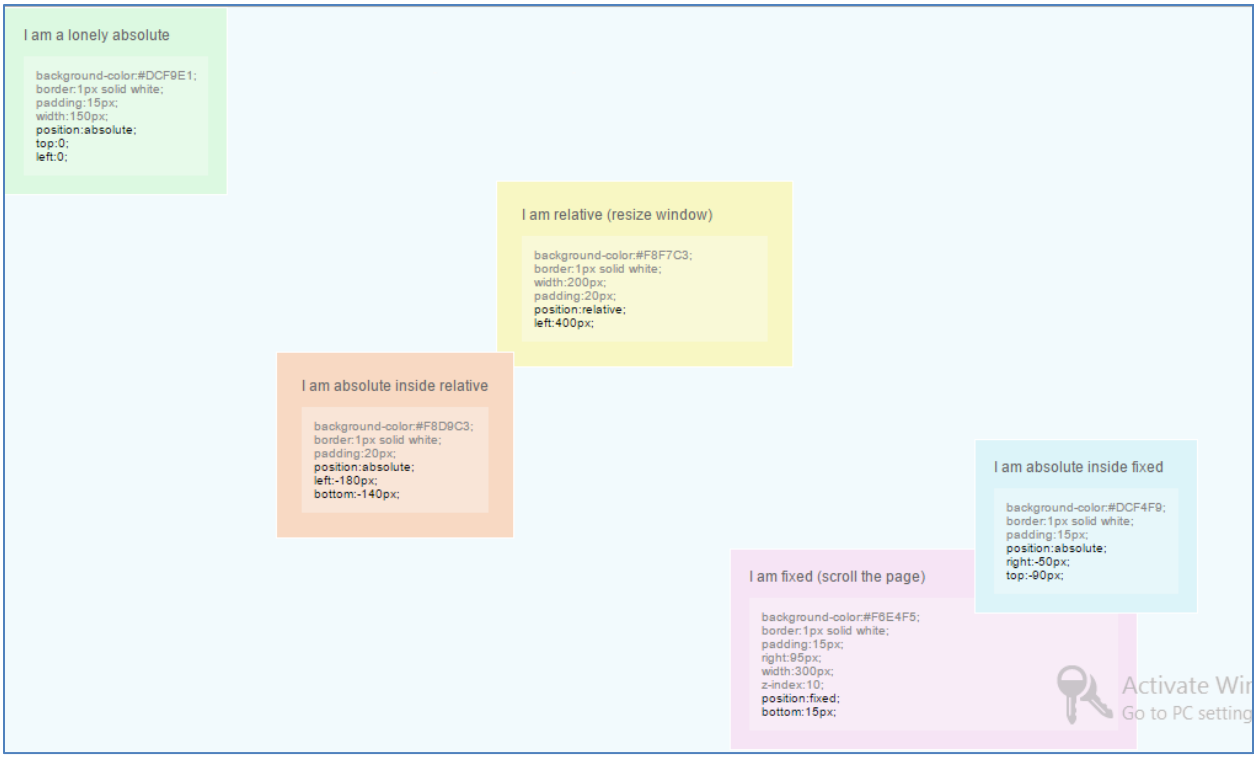
Example Code Snippet from VAADIN:

```
// A 400x250 pixels size layout

AbsoluteLayout layout = new AbsoluteLayout();

layout.setWidth("400px");

layout.setHeight("250px");


// A component with coordinates for its top-left
corner
TextField text = new TextField("Somewhere
someplace");
layout.addComponent(text, "left: 50px; top: 50px;");
```

Example Code Snippet from CSS:

```
p.pos_fixed
{
position:fixed;
top:30px;
right:5px;

}
```

Here is another example of CSS absolte positioning with demo:



4.4 Patterns in Layout Re-rendering Approach

This is about the ability to re-adjust the component positions and / or component size depending upon browser size variation, user interaction with devices etc. This requirement comes in design pattern identification when an application is meant to be used for multiple sized devices.

Static Rendering is for applications which are built to run on a single device. The usage of this pattern is almost eliminated now-a-days.

Liquid Rendering takes percentage of page size to decide on component size, component locations etc. instead of pixels as used in static rendering.

Adaptive approach is predicting the browser width using media queries and based on that component level details are either in pixel, or in percentage.

Responsive design is the most flexible design patterns now–a-days with variety of innovative features included in it. This approach is purely based on User Requirements Usage mode, Device sizes etc. and a matter of continuous evolution.

The available approaches and their best usage are depicted in detail at Appendix 2: Layout Rendering Approach.

5. Understanding the LearnITy Framework

For further experiments and enhancing the layout, navigation and skin UI design one existing framework was needed on top of which the new design could be implemented for evaluation of the enhancements. Apart from providing useful features in terms of UI Design ease of use for a framework was also considered. Considering adverse eligibilities of developers in terms of learning any new programming language for using any UI design framework, declarative programming approach was targeted for enhancement design. This is because declarative programming is actually made for any kind of developers, who are not even well versed with Software Programming Languages. It is often English language based and follows some pre-defined structure in XML, hence easy to use as well.

LearnITy Framework is one declarative programming based framework and inspired by IBM's AUIML concept. It is easy to use and worth enhancing because, it may fulfill all our design criteria covering rich UI features and usage of framework. This framework may have well marketability if it can be enhanced properly to address current UI design pain points in terms of Layout, Navigation and Skin.

5.1 Basic Concepts of LearnITy Framework

(Reference[40]), PPT "LearnITy Framework"

- A high-level framework for developing web (browser) based software using Java
- Not tied to any particular application domain, rather a generic framework
- Inspired by IBM's AUIML (Abstract User Interface Modeling Language) (References[11-14])
- An application is considered to be a collection of interfaces represented using XML
- An interface contains UI element inventory, layout and style information, browser event controller logic, and screen assets such as images
- Server side logic is implemented using POJOs that are called directly from browser event controllers using AJAX with auto marshalling and unmarshalling of data between Java and JavaScript
- The overview of existing Java classes along with their functionalities are captured in the "Feature_Code_Mapping" excel which is being maintained at BOX repository of LearnITy Framework (Reference[40])
- An interface is zipped collection of the base interface XML file and other assets such as images, CSS and JS files, etc.
- An application is loaded in to an application server through a (browser based) CoreAdmin interface that facilitates uploading of interface Zips
- The LF comes with its own database schema, which is distinct from the application schema

- The LF currently supports XHTML rendering and the information uploaded via the interface is used to pre-generate XHTML snippets and stored in the framework's own schema
- The runtime rendering is performed through a front controller *(PortalServlet)* that takes the ID of an interface as an argument *(./interfaceengine.PortalServlet?IID=myInterface)*
- Static elements of the screen are served from the pre-generated XHTML snippets while dynamic elements are created mostly on the browser using DOM manipulation or in some cases (like where DB interaction is required) created within POJOs and rendered in browser using JS function

5.2 Why use LearnITy Framework

- Main design objective of the framework is to provide a higher level abstraction so that web-based applications may be developed more productively
- Minimize html/javascript Code writing UI Design
- Users need to provide structure, layout, content & behavior information of any interface and framework will parse it, convert it into HTML tags for displaying on screen
- Faster development cycle for web based applications
- Learning the framework usage is very smooth for the developers
- None of the existing third party framework provide right balance of complexity and productivity which is mainly targeted in LearnITy Framework

5.3 Main Components of LearnITy Framework

InterfaceEngine is a framework that primarily developed as an in-house framework has a great potential to be adopted as a framework-of-choice for many problem domains requiring maintenance-critical and incredibly customizable software development.

It's a framework for web development using J2EE. This application uses the tomcat web server for development of sample application and uses MySQL as database. An interface collection has the following components -

- manifest.xml
- interfacerole.xml
- Interface Archives

They are compressed into a zip file to form a deployable package that is uploaded from interface engine coreadmin. The interface collection unfolds at the server side and translated to a format that contains information needed to generate a dynamic html page by PoralServlet (the unanimous controller of interface engine pages) when the page is requested. All the information resides at a database repository (presently MySQL).

5.3.1 Manifest.xml

The manifest.xml is an index of interface archives and it must be present inside an interface collection. It helps Interfaceengine core administration to recognize an interface archive as a part of an interface collection. Each interface archive must have an entry in the manifest to get accommodated in the interface repository. The following is an example of manifest.xml.

```
<? Xml version="1.0" encoding="UTF-8"?>

 <manifest id="myApp Interface" title="myApp Interface Collection">

  <interface id="loginPage" title="loginPage"  type="Interface" zip="loginPage.zip" />

  <interface id="myInterface" title="myInterface"  type="Interface" zip="myInterface.zip"
/>

 <!-- and so on -->

</manifest>
```

This xml document encloses all the entries inside <manifest> tag-pair. This tag has two attributes. The id attribute is an identifier of the interface collection and the title is the string that will be displayed in the paginated grid of ‘Interface Management’ screen of coreadmin. Each <interface/> tag points to an interface archive referred by the id of the interface. Value of the title attribute shows up in the Interface Management grid of coreadmin and the type attribute is always ‘Interface’. Zip attribute mention the name of the interface archive zip file.

5.3.2 Interfacerole.xml

This is another xml configuration file which contains information about interface role-mapping. This is required to generate html response according to role on invocation of interfaces. This xml configuration file is also mandatory in an interface collection. Roles are subsets of actions or functions of a system. For example, the employees of an organization have responsibilities according to their position in the organization. In a cricket match bowlers, batsmen, fielders, wicketkeepers, umpires and the match refry all play their respective roles. Software is used to carry on the working of an organization in an automated and networked way. A user of the software is authorized to access only those parts of the software or use only those features of the software which he/she is allowed to.

In this file we need to define the roles & the work of that role. We need to define which interface will be accessible under which role. Also, since a single interface may have multiple interface components ((layout, content, behaviour, etc.) of the same type; for each accessible interface, we need to specify the id of the interface components that are accessible for that particular role.

```
<? Xml version="1.0" encoding="UTF-8"?>

<roles>

<role id="ADMIN">

<interface id="AccountSummaryCenterPanel">

<layout id="AccountSummaryCenterPanelLayout"/>

<content id="AccountSummaryCenterPanelContent"/>

<style id="AccountSummaryCenterPanelStyle"/>

<behaviour id="AccountSummaryCenterPanelBehaviour"/>

</interface>

</role> </roles>
```

In the above example we have defined a role called 'Admin', and the admin will get the access of “AccountSummaryCenterPanel” interface. We also need to mention the “layout id”, “content id” , “styleid”, “and behaviour id”.

5.3.3 Interface.xml

In an “interface.xml” file there are the following six sections.

- <Interface>

Under this section we need to define the id, name & the type of the interface. All other sections come under this part only.

```
E.g.:  <interface id="LearnityPortal" title="LearnityPortal"
type="Interface">
```

- <structure>

The Structure section is used define the complete inventory of UI parts along with their attributes. Under the 'structure' section we need to define the classes (like – label, textlink etc.) that we need to use in the page which we are designing. All class should have one unique id by which these classes will be recognized in other sections.

E.g.: <structure><part id="Main Screen" class="label"></part></structure>

- <layout>

The Layout section is used define the positioning of the UI elements on the screen as well as the hierarchical parent-child relationship among the elements. Under the 'layout' section we need to define the position of all the parts, which we have defined under the 'structure' section.

E.g.: - <layout id="Portallayout"> <part id="MainScreen" height="600px" width="1000px" left="1px" top="1px" position="absolute"></part> </layout>

- <content>

The Content section is used associate various types of static content with the UI elements. Under the 'content' section we need to define what should be the content of the above defined parts

E.g.: - <content id="Portalcontent"> <part id="appname" value="Learnity Portal" value type="inline"></part> </content>

- <style>

The Style section is used define the visual attributes of the UI elements on the screen such as font face, colour, alignment, etc. The syntax for defining styling information is same as the syntax for defining styles in cascading style sheets. External CSS files may also be used

E.g.: - <style id="Portalstyle"> <part id="MainScreen" value="maincss" valuetype="reference"></part> <part id="appname" value="[CDATA [color: #D4D4C5; font: 16px arial;font-weight:bold;]]" valuetype="inline"></part> </style>

In the 1st example we have embedded an existing 'css' file with the class, but in the 2nd example we have defined the style for that class inside the tag only.

- <behaviour>

The Behaviour section is used define the behaviour of the UI elements against various events occurring on the screen. The Behaviour section is used define the behaviour of the UI elements against various events occurring on the screen. For example if we want the 'Submit' button to perform a task when the user clicks it, then we need to define how it will perform the task under this 'behaviour' section.

```
E.g. - <behaviour id="Portalbehaviour"> <part id="root">  <event
name="onload_click ()" type="simple" function="" callback="" javaclass=""
target="" valuetype="jsevent" resourced="accmainjs"></event> </part>
</behaviour>
```

- <resource>

The Resource section is used associate various types of external content with the interface elements

```
E.g.: - <resource> <resourceitem id="logoutimg" href="h_logout.png"
valuetype="image"></resourceitem> </resource>
```

5.4 How to Use LearnITy Framework

Step1: Decide on the names of the UI pages that are going to be created.

Step2: Create “manifest.xml” file, listing all the interface/UI Page names in <interface> tags and associating each interface with its zip file name

Step3: Decide on the roles that will be available in accessing your web application

Step4: Create “interfacerole.xml” file where each role wise available interface, page layout, page structure, page content and page behaviour will be listed by using relevant part ids

Step5: Design all the UI pages using interface.xml file for each page. Each page content in XML, JS Scripts, Images etc. need to be zipped together and zip name must match with the name specified in manifest.xml file

Step6: Configure web.xml to include servlet class names and servlet mapping

Step7: Start Tomcat and login to the Admin page using admin URL

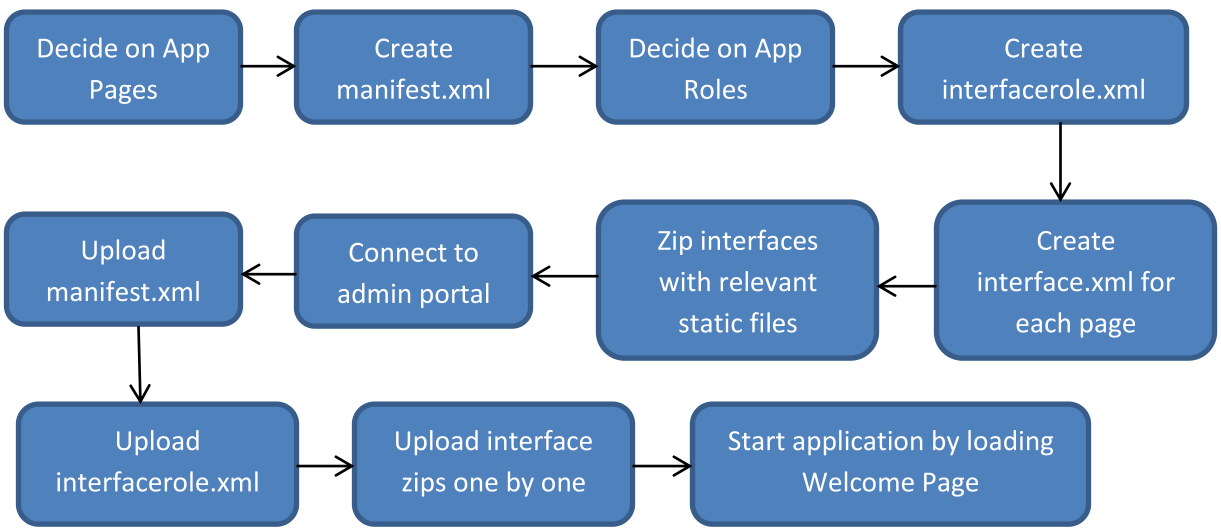
e.g. http://localhost:8080/threatanalysis/coreadmin/, with admin user login

Step8: Upload manifest.xml , interfacerole.xml and interface zip files in the same order. Files will be validated against pre-defined XSD structures and on successful validation only XML files will be parsed for getting stored on Database.

Step9: Login to portal using proper URL and welcome page will be displayed

e.g.,
http://localhost:8080/threatanalysis/servlet/interfaceenginev2.PortalServlet?IID=LoginPage

Below is the flow diagram of all the steps mentioned above:



5.5 Sample Screens Implemented in LearnITy Framework

Configuration xmls of below screen (A/C summary view) is provided at the Live Implementation section.

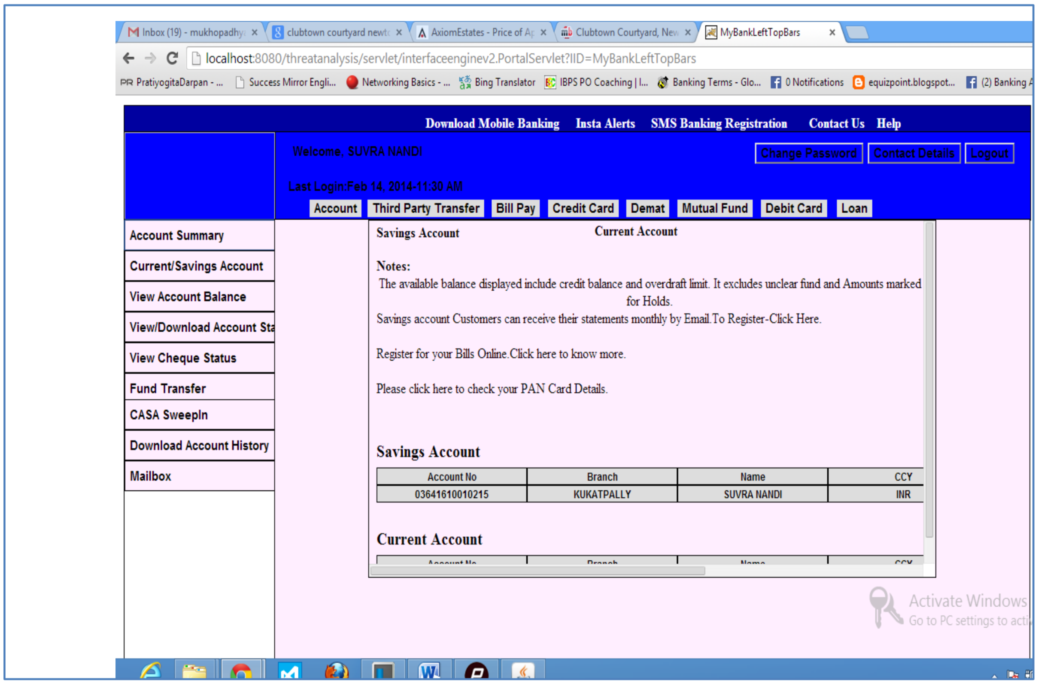


Figure 73: LearnITy Framework Implementation Screen - Account Summary Screen

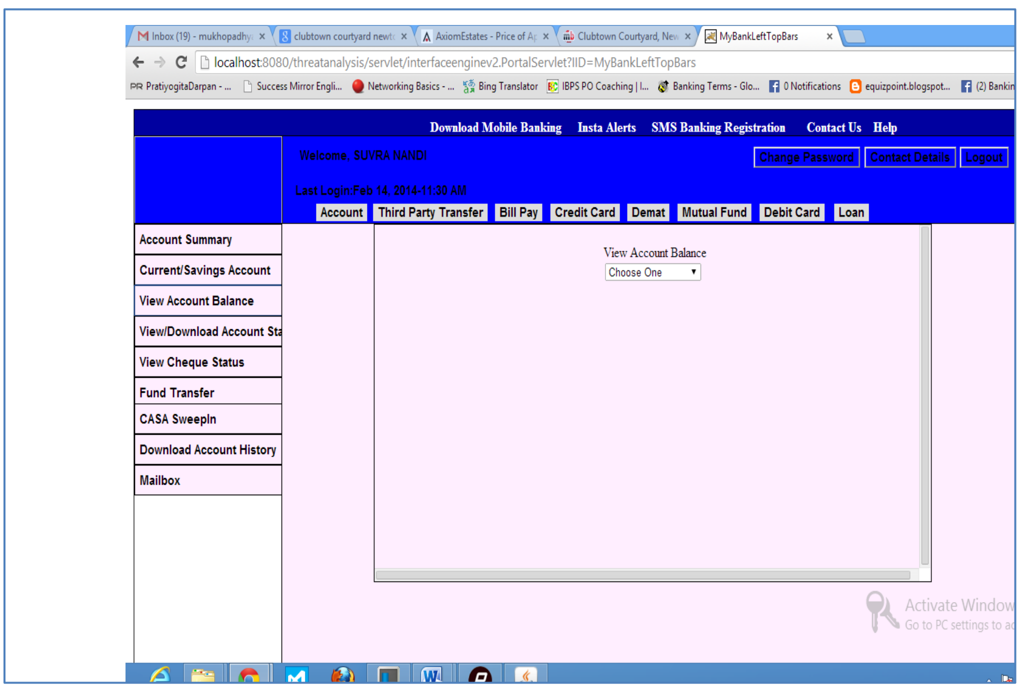


Figure 74: LearnITy Framework Implementation Screen - Select Account

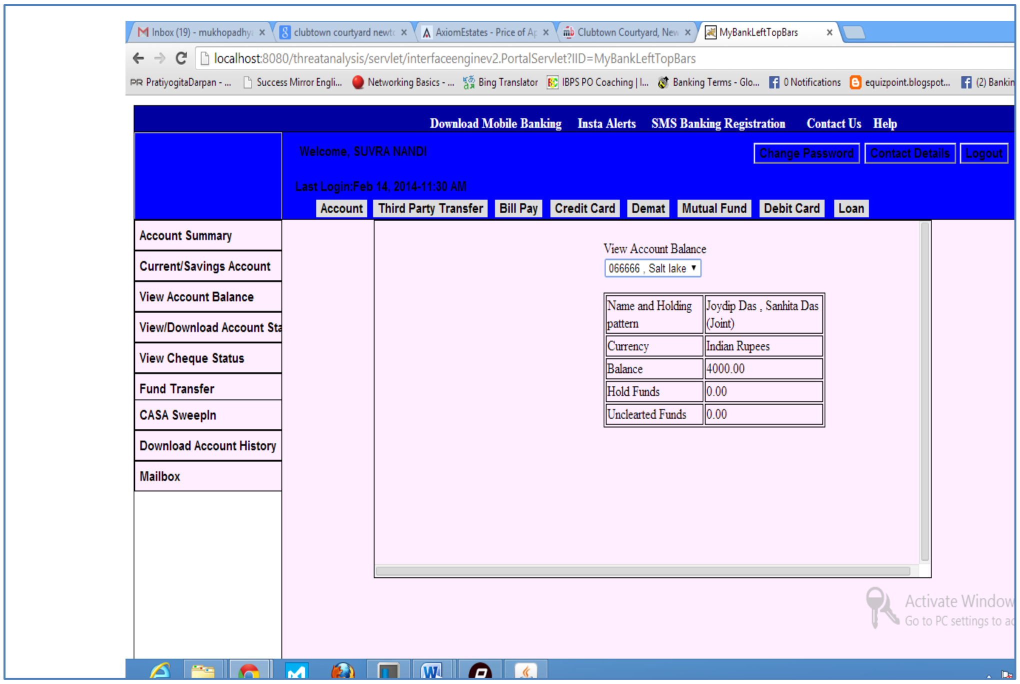


Figure 75: LearnITy Framework Implementation Screen - View Account Balance - Account1

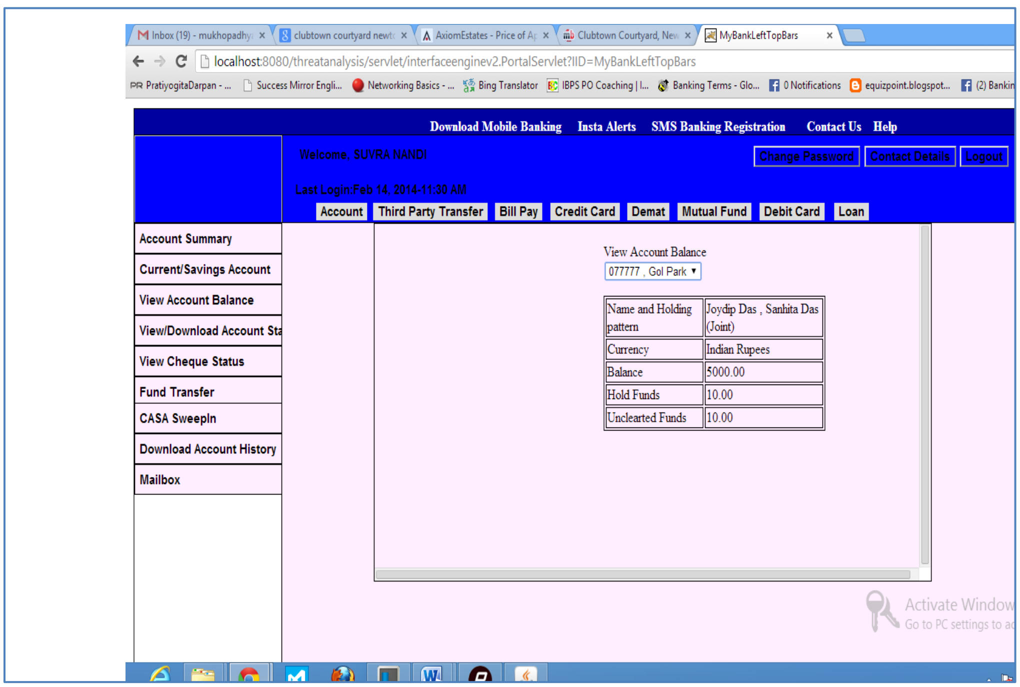


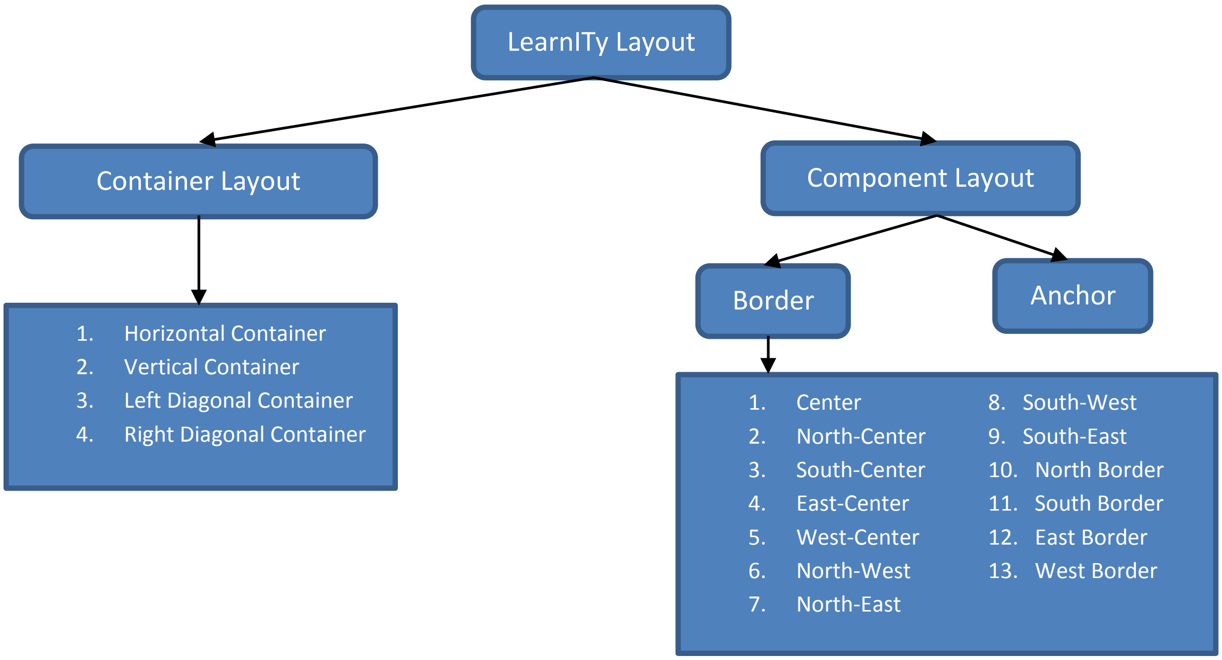
Figure 76: LearnITy Framework Implementation Screen - View Account Balance - Account2

6. Improved Design Implementation in the LearnITy Framework

6.1 LearnITy Framework Layout Design

The enhanced layout design considers the page as grid, with non-template based free flow structure for each page. For implementation of layout on grid structure it reuses the Layout Manager concept for implementing Container layout, Anchor Layout concept is for component positioning. Also part of Layout Manager concept is utilized as Border Layout which again places components along the page borders.

The new Layout design in LearnITy Framework is built in following ways:



6.1.1 Container Layout

At container level developer will be able to provide layout modes such as horizontal, vertical, leftDiagonal, rightDiagonal. This will make child components oriented that way within the container. In this framework new classes "*horizontalContainer*", "*verticalContainer*", "*rightDiagonalContainer*" and "*leftDiagonalContainer*" are introduced.

Sample Screens

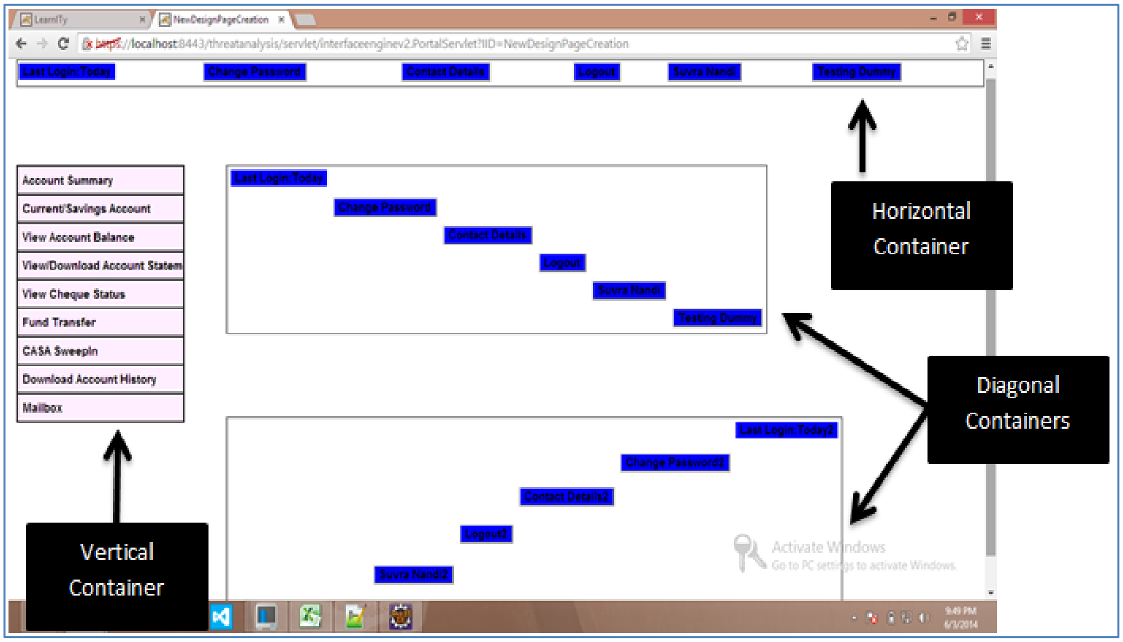


Figure 77: Enhanced Layout Design - HorizontalContainer, VerticalContainer, RightDiagonalContainer, LeftDiagonalContainer

```
<structure>

<part id="main" class="label"/>

<part id="topNarrowPanel" class="horizontalContainer"/>

<part id="lastLogin" class="button" />

<part id="changePasswd" class="button" />

<part id="contactDetails" class="button" />

<part id="logout" class="button" />

<part id="dummyButton1" class="button" />

<part id="dummyButton2" class="button" />

<part id="leftBar" class="verticalContainer"/>

<part id="acctSummary" class="button" />

<part id="currSavAccount" class="button"/>
```

```
<part id="viewAcctBal" class="button" />

<part id="viewDownloadAcctStmt" class="button" />

<part id="viewChkStatus" class="button" />

<part id="fundXfer" class="button" />

<part id="casaSweepIn" class="button" />

<part id="downloadHistStmt" class="button" />

<part id="mailBox" class="button" />

<part id="centerPanel1" class="rightDiagonalContainer"/>

<part id="lastLogin1" class="button" />

<part id="changePasswd1" class="button" />

<part id="contactDetails1" class="button" />

<part id="logout1" class="button" />

<part id="dummyButton11" class="button" />

<part id="dummyButton21" class="button" />

<part id="centerPanel2" class="leftDiagonalContainer"/>

<part id="lastLogin2" class="button" />

<part id="changePasswd2" class="button" />

<part id="contactDetails2" class="button" />

<part id="logout2" class="button" />

<part id="dummyButton12" class="button" />

<part id="dummyButton22" class="button" />

</structure>

<layout id="NewDesignPageCreationLayout">

<part id="main" height="99%" width="99%" left="1px" top="1px"
position="absolute"/>

<part id="topNarrowPanel" height="30px" width="100%" left="10px" top="10px"
```

```
position="absolute" parent_id="main"/>

<part id="lastLogin" height="30px" parent_id="topNarrowPanel" />

<part id="changePasswd" height="30px" parent_id="topNarrowPanel" />

<part id="contactDetails" height="30px"  parent_id="topNarrowPanel" />

<part id="logout" height="30px" parent_id="topNarrowPanel" />

<part id="dummyButton1" height="30px" parent_id="topNarrowPanel" />

<part id="dummyButton2" height="30px" parent_id="topNarrowPanel" />

<part id="leftBar" height="" width="" left="10px" top="140px"
position="absolute" parent_id="main" />

<part id="acctSummary" parent_id="leftBar" />

<part id="currSavAccount" parent_id="leftBar" />

<part id="viewAcctBal" parent_id="leftBar" />

<part id="viewDownloadAcctStmt" parent_id="leftBar" />

<part id="viewChkStatus" parent_id="leftBar" />

<part id="fundXfer" parent_id="leftBar" />

<part id="casaSweepIn" parent_id="leftBar" />

<part id="downloadHistStmt" parent_id="leftBar" />

<part id="mailBox" parent_id="leftBar" />

<part id="centerPanel1" height="" width="" left="300px" top="140px"
position="absolute" parent_id="main"/>

<part id="lastLogin1" height="30px" parent_id="centerPanel1" />

<part id="changePasswd1" height="30px" parent_id="centerPanel1" />

<part id="contactDetails1" height="30px"  parent_id="centerPanel1" />

<part id="logout1" height="30px" parent_id="centerPanel1" />

<part id="dummyButton11" height="30px" parent_id="centerPanel1" />

<part id="dummyButton21" height="30px" parent_id="centerPanel1" />
```

```
<part id="centerPanel2" height="" width="" left="300px" top="450px"
position="absolute" parent_id="main"/>

<part id="lastLogin2" height="30px" parent_id="centerPanel2" />

<part id="changePasswd2" height="30px" parent_id="centerPanel2" />

<part id="contactDetails2" height="30px"  parent_id="centerPanel2" />

<part id="logout2" height="30px" parent_id="centerPanel2" />

<part id="dummyButton12" height="30px" parent_id="centerPanel2" />

<part id="dummyButton22" height="30px" parent_id="centerPanel2" />

</layout>

<content id="NewDesignPageCreationContent">

<part id="main"  value="nadmin" valuetype="reference"/>

<part id="topNarrowPanel"  value="" valuetype="inline"/>

<part id="lastLogin" value="Last Login:Today" valueType="inline" />

<part id="changePasswd" value="Change Password" valueType="inline" />

<part id="contactDetails" value="Contact Details" valueType="inline" />

<part id="logout" value="Logout" valueType="inline" />

<part id="dummyButton1" value="Suvra Nandi" valueType="inline" />

<part id="dummyButton2" value="Testing Dummy" valueType="inline" />

<part id="leftBar" value="" valueType="inline" />

<part id="acctSummary" value="Account Summary" valueType="inline" />

<part id="currSavAccount" value="Current/Savings Account" valueType="inline" />

<part id="viewAcctBal" value="View Account Balance" valueType="inline" />

<part id="viewDownloadAcctStmt" value="View/Download Account Statement"
valueType="inline" />

<part id="viewChkStatus" value="View Cheque Status" valueType="inline" />

<part id="fundXfer" value="Fund Transfer" valueType="inline" />
```

```
<part id="casaSweepIn" value="CASA SweepIn" valueType="inline" />

<part id="downloadHistStmt" value="Download Account History"
valueType="inline" />

<part id="mailBox" value="Mailbox" valueType="inline" />

<part id="centerPanel1"  value="" valuetype="inline"/>

<part id="lastLogin1" value="Last Login:Today" valueType="inline" />

<part id="changePasswd1" value="Change Password" valueType="inline" />

<part id="contactDetails1" value="Contact Details" valueType="inline" />

<part id="logout1" value="Logout" valueType="inline" />

<part id="dummyButton11" value="Suvra Nandi" valueType="inline" />

<part id="dummyButton21" value="Testing Dummy" valueType="inline" />

<part id="centerPanel2"  value="" valuetype="inline"/>

<part id="lastLogin2" value="Last Login:Today2" valueType="inline" />

<part id="changePasswd2" value="Change Password2" valueType="inline" />

<part id="contactDetails2" value="Contact Details2" valueType="inline" />

<part id="logout2" value="Logout2" valueType="inline" />

<part id="dummyButton12" value="Suvra Nandi2" valueType="inline" />

<part id="dummyButton22" value="Testing Dummy2" valueType="inline" />

</content>

<style id="NewDesignPageCreationStyle">

<part id="topNarrowPanel" value="[CDATA[border: 1px solid black;;background-
color:#FFFFFF]]" valuetype="inline"/>

<part id="lastLogin" value="[CDATA[border: 0px solid black;;background-
color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]"
valuetype="inline"/>

<part id="changePasswd" value="[CDATA[border: px solid black;;background-
color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]"
```

```
valuetype="inline"/>

<part id="contactDetails" value="[CDATA[border: px solid black;;background-
color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]"
valuetype="inline"/>

<part id="logout" value="[CDATA[border: px solid black;;background-
color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]"
valuetype="inline"/>

<part id="dummyButton1" value="[CDATA[border: px solid black;;background-
color:#0000EE;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]"
valuetype="inline"/>

<part id="dummyButton2" value="[CDATA[border: px solid black;;background-
color:#0000EE;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]"
valuetype="inline"/>

<part id="leftBar" value="[CDATA[width:230px;border: 1px solid
black;;background-color:#FFFFFF;text-align:center;]]" valuetype="inline"/>

<part id="acctSummary" value="[CDATA[height:35px; width: 230px;border: 1px
solid black;;background-color:#FFEEFF;text-align:left;color:#000000;font-
size:15px;font-weight:bold]]" valuetype="inline"/>

<part id="currSavAccount" value="[CDATA[height:35px; width: 230px;border: 1px
solid black;;background-color:#FFEEFF;text-align:left;color:#000000;font-
size:15px;font-weight:bold]]" valuetype="inline"/>

<part id="viewAcctBal" value="[CDATA[height:35px; width: 230px;border: 1px
solid black;;background-color:#FFEEFF;text-align:left;color:#000000;font-
size:15px;font-weight:bold]]" valuetype="inline"/>

<part id="viewDownloadAcctStmt" value="[CDATA[height:35px; width:
230px;border: 1px solid black;;background-color:#FFEEFF;text-
align:left;color:#000000;font-size:15px;font-weight:bold;word-wrap:true]]"
valuetype="inline"/>

<part id="viewChkStatus" value="[CDATA[height:35px; width: 230px;border: 1px
solid black;;background-color:#FFEEFF;text-align:left;color:#000000;font-
size:15px;font-weight:bold]]" valuetype="inline"/>

<part id="fundXfer" value="[CDATA[height:35px; width: 230px;border: 1px solid
black;;background-color:#FFEEFF;text-align:left;color:#000000;font-size:15px;font-
weight:bold]]" valuetype="inline"/>
```

```
<part id="casaSweepIn" value="[CDATA[height:35px; width: 230px;border: 1px solid black;;background-color:#FFEEFF;text-align:left;color:#000000;font-size:15px;font-weight:bold]]" valuetype="inline"/>

<part id="downloadHistStmt" value="[CDATA[height:35px; width: 230px;border: 1px solid black;;background-color:#FFEEFF;text-align:left;color:#000000;font-size:15px;font-weight:bold]]" valuetype="inline"/>

<part id="mailBox" value="[CDATA[height:35px; width: 230px;border: 1px solid black;;background-color:#FFEEFF;text-align:left;color:#000000;font-size:15px;font-weight:bold]]" valuetype="inline"/>

<part id="centerPanel1" value="[CDATA[border: 1px solid black;;background-color:#FFFFFF]]" valuetype="inline"/>

<part id="lastLogin1" value="[CDATA[border: 0px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="changePasswd1" value="[CDATA[border: px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="contactDetails1" value="[CDATA[border: px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="logout1" value="[CDATA[border: px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="dummyButton11" value="[CDATA[border: px solid black;;background-color:#0000EE;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="dummyButton21" value="[CDATA[border: px solid black;;background-color:#0000EE;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="centerPanel2" value="[CDATA[border: 1px solid black;;background-color:#FFFFFF]]" valuetype="inline"/>

<part id="lastLogin2" value="[CDATA[border: 0px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>
```

```
<part id="changePasswd2" value="[CDATA[border: px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="contactDetails2" value="[CDATA[border: px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="logout2" value="[CDATA[border: px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>
  T
<part id="dummyButton12" value="[CDATA[border: px solid black;;background-color:#0000EE;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="dummyButton22" value="[CDATA[border: px solid black;;background-color:#0000EE;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

</style>
```

XML Code Snippet for Container Layout

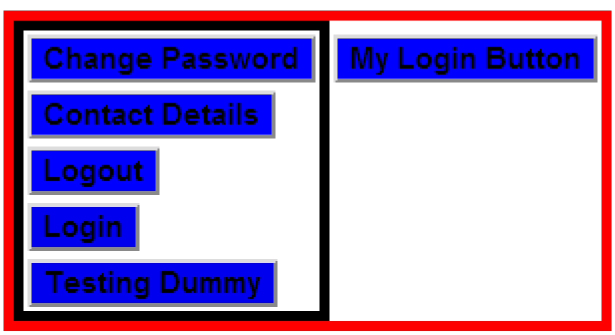


Figure 78: Enhanced Layout Design - Recursive Container - Red Bordered Horizontal Container containing one verticalContainer and one Button

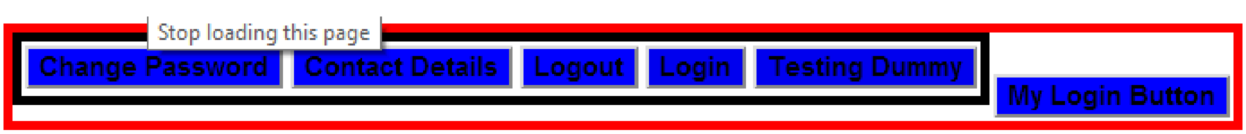


Figure 79: Enhanced Layout Design - Recursive Container: Red Bordered Horizontal Container containing one horizontalContainer and one Button

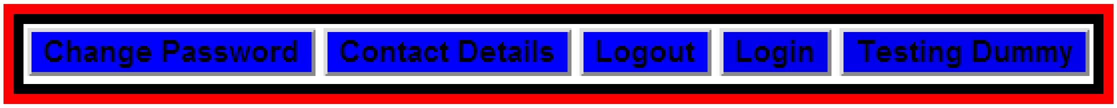


Figure 80: Enhanced Layout Design - Recursive Container - Black Bordered Horizontal Container, inside Red Bordered Horizontal Container

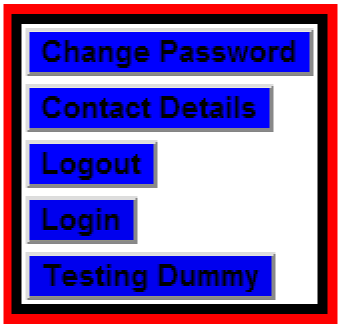


Figure 81: Enhanced Layout Design - Recursive Container - Black Bordered Vertical Container, inside Red Bordered Horizontal Container

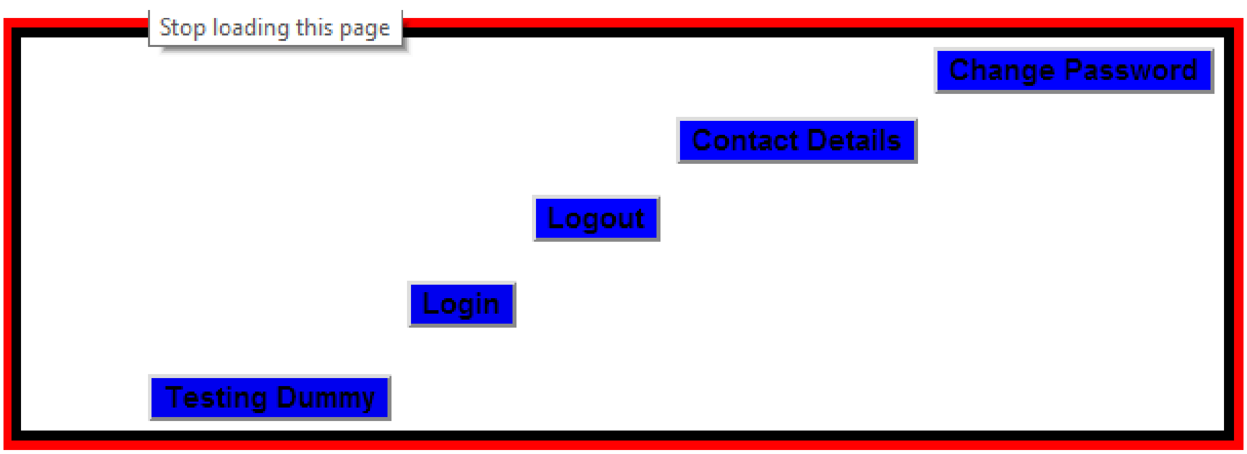


Figure 82: Enhanced Layout Design - Recursive Container - Black Bordered LeftDiagonalContainer, inside Red Bordered Horizontal Container

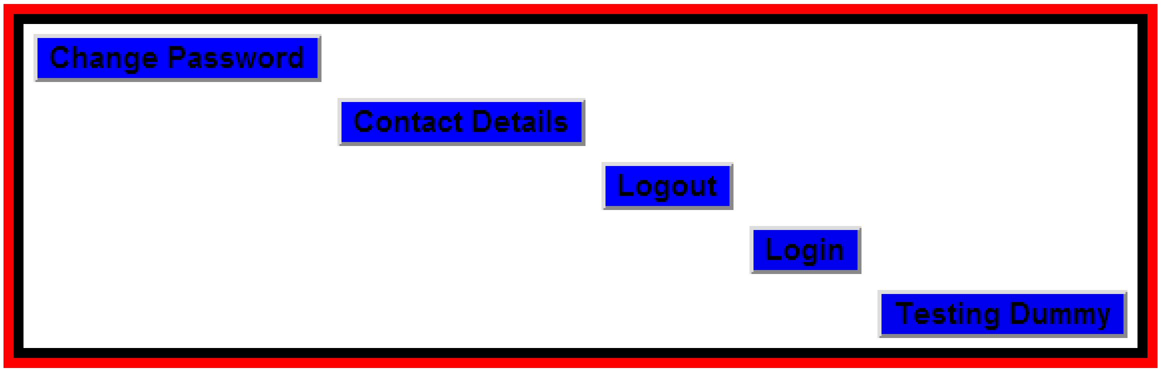


Figure 83: Enhanced Layout Design - Recursive Container - Black Bordered RightDiagonalContainer, inside Red Bordered Horizontal Container

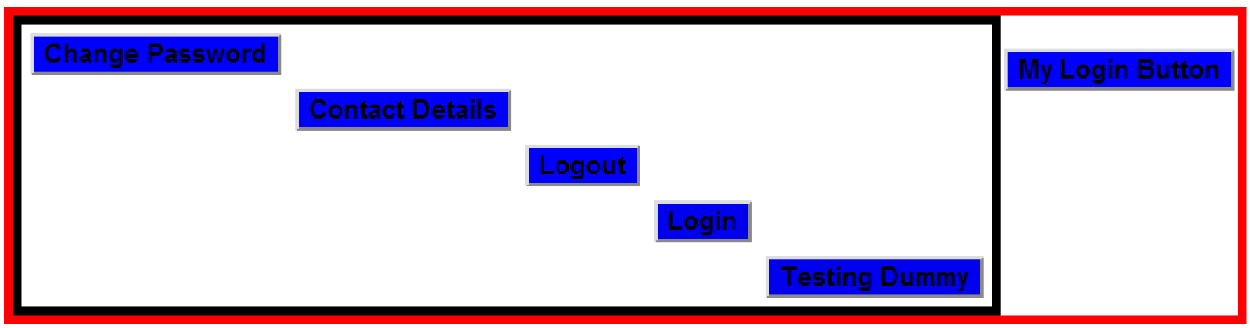


Figure 84: Enhanced Layout Design - Recursive Container - Red Bordered Horizontal Container containing one RightDiagonalContainer and one Button

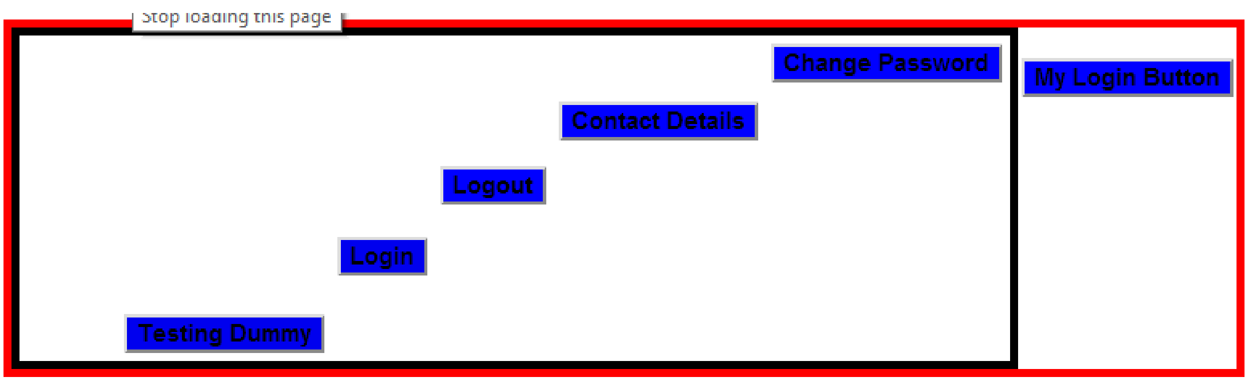


Figure 85: Enhanced Layout Design - Recursive Container - Red Bordered Horizontal Container containing one LeftDiagonalContainer and one Button

6.1.2 Component Layout

In case developer doesn't want to provide any layout mode at container, there will be options to provide layout mode at component levels too. At component level layout mode can be any one of border and anchor.

a) Border Layout

If layout mode is border, position & spacing attributes needs to be filled in. In Border Layout positions can be of 13 types:

Center, North-Center (nc), South-Center(sc), West-Center(wc), East-Center(ec), North-West(nw), North-East(ne), South-West(sw), South-East(se), North Border(north), South Border(south), West Border(west), East Border(east).

Sample Demo with Blocks

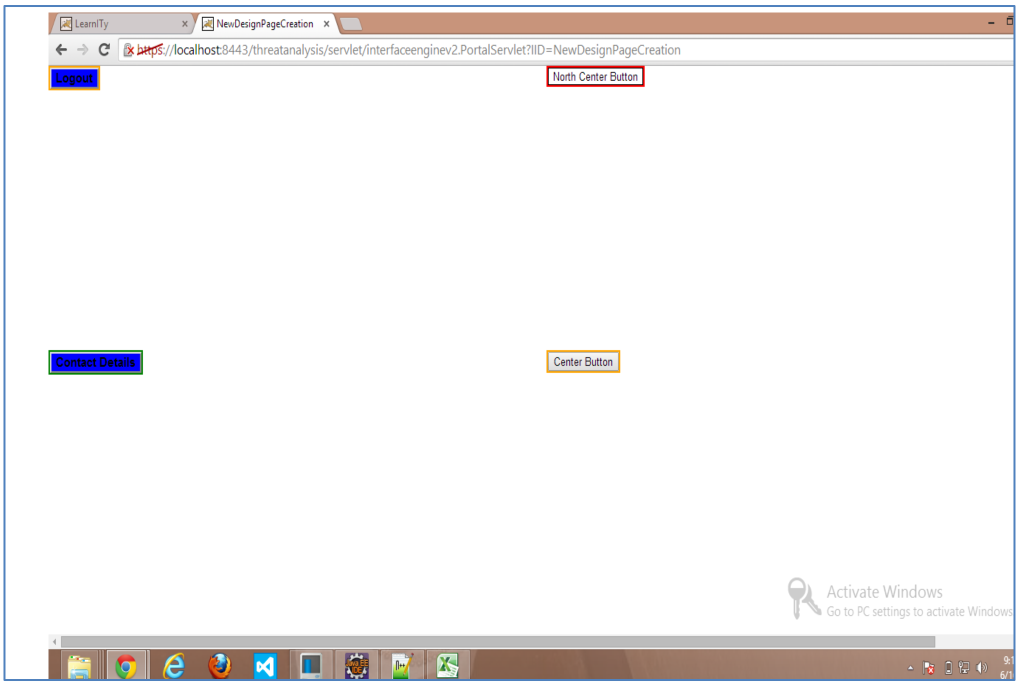


Figure 86: Enhanced Layout Design - Border Layout - Buttons Positioned at North West, North Center, West Center, Center

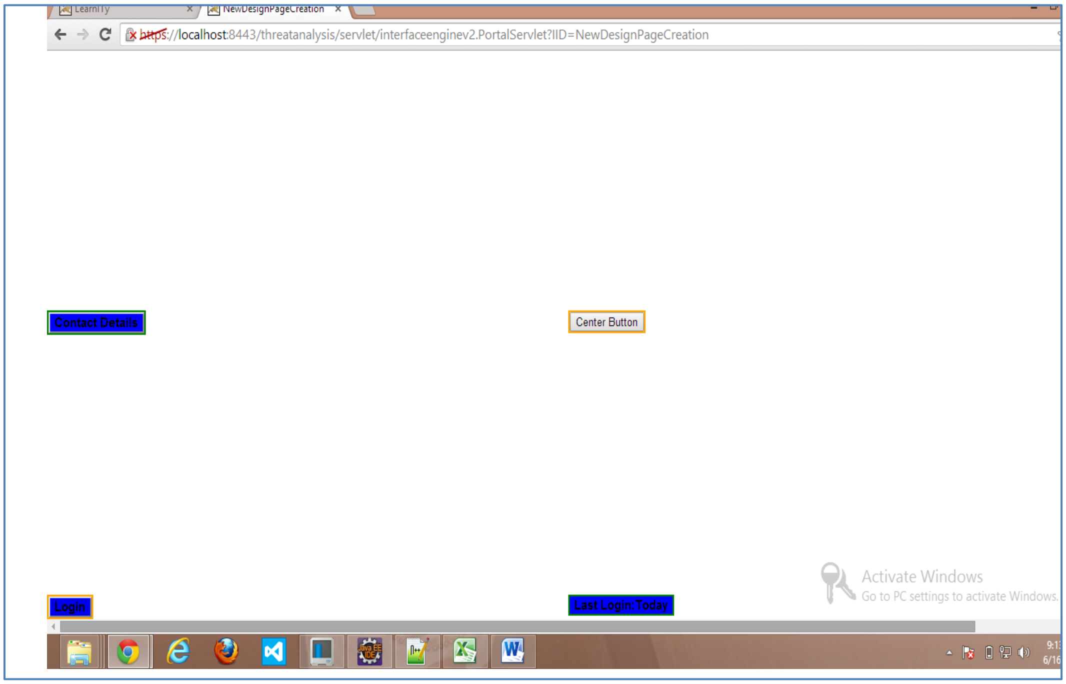


Figure 87: Enhanced Layout Design - Border Layout - Buttons Positioned at South West, South Center, West Center, and Center

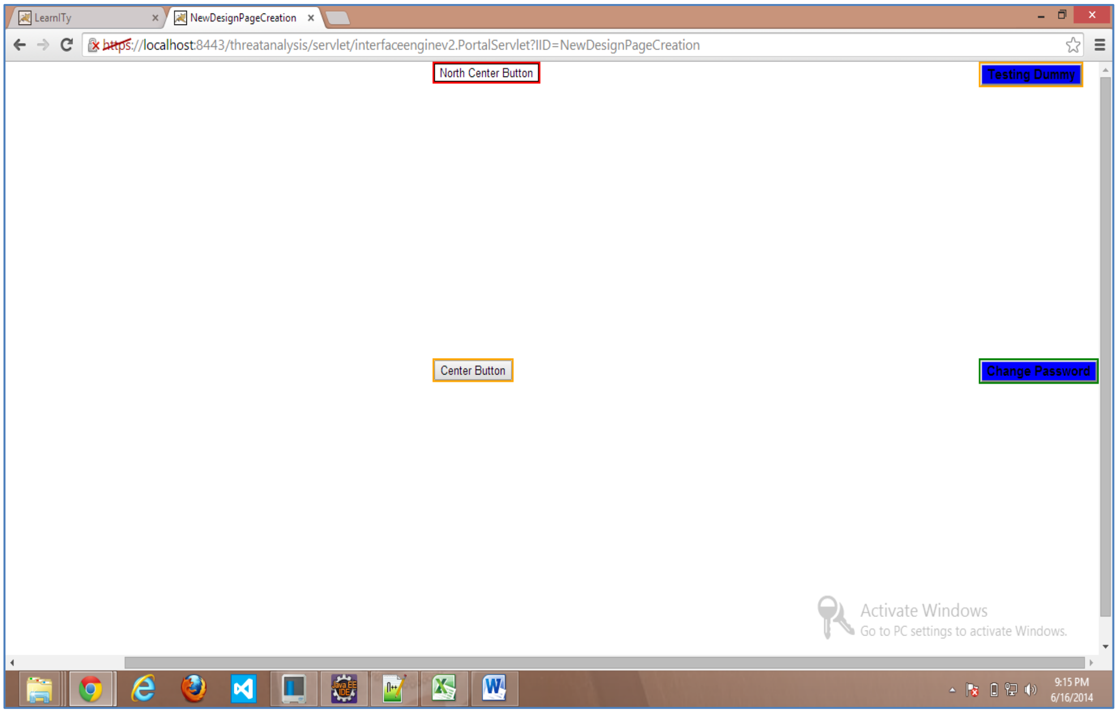


Figure 88: Enhanced Layout Design - Border Layout - Buttons Positioned at North East, North Center, East Center, and Center

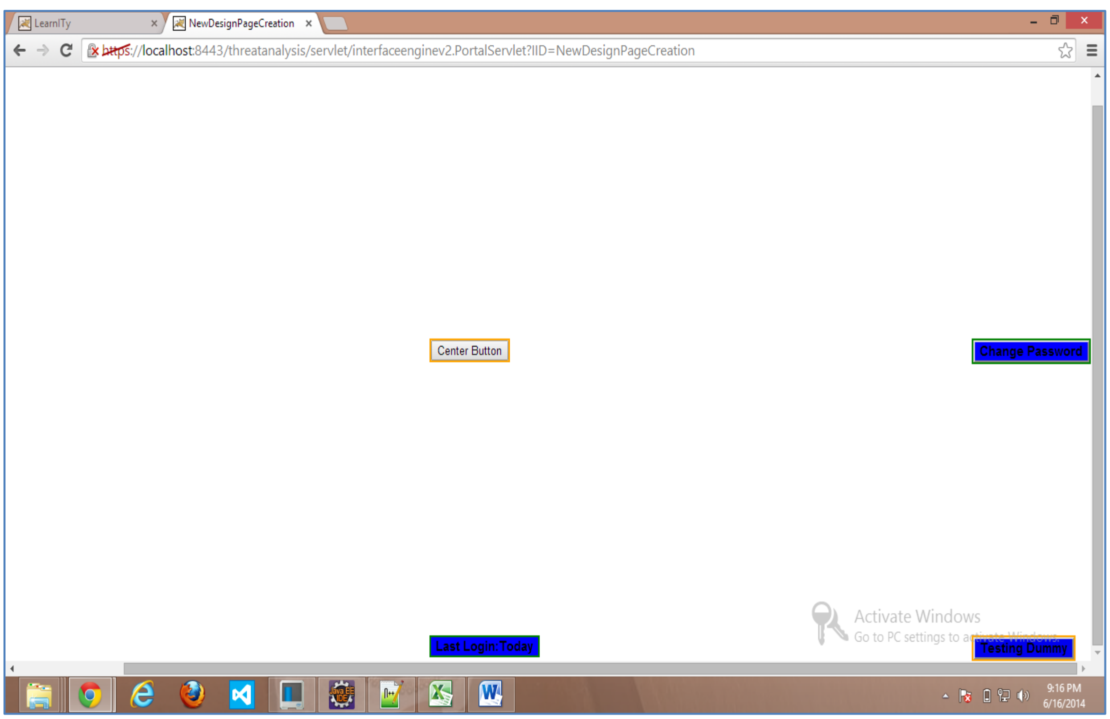


Figure 89: Enhanced Layout Design - Border Layout - Buttons Positioned at South East, South Center, East Center, and Center

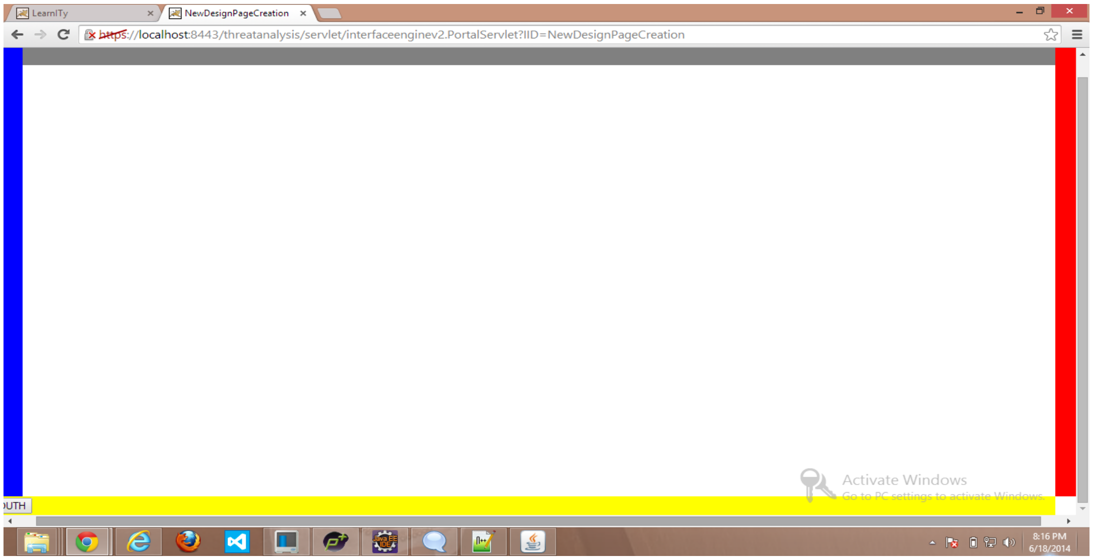


Figure 90: Enhanced Layout Design - Border Layout - North, South, East and West Bars placed in page using Border Layout

```
<interface id="NewDesignPageCreation" title="NewDesignPageCreation"
type="Interface">

<structure>

<!--Containers - Start-->

<part id="main" class="label"/>

<part id="topNarrowPanel" class="button"/>

<part id="lastLogin" class="button" />

<part id="changePasswd" class="button" />

<part id="contactDetails" class="button" />

<part id="logout" class="button" />

<part id="dummyButton1" class="button" />

<part id="dummyButton2" class="button" />

<part id="dummyButton3" class="button" />

<part id="dummyButton4" class="button" />
```

```
<part id="northBar" class="button"/>

<part id="westBar" class="button"/>

<part id="southBar" class="button"/>

<part id="eastBar" class="button"/>

</structure>

<layout id="NewDesignPageCreationLayout">

<part id="main" height="100%" width="100%" left="1px" top="1px"
position="absolute"/>

<part id="topNarrowPanel" height="" width="" left="" top="" position="border"
borderPosition="nc" parent_id="main"/>

<part id="lastLogin" height="" width="" left="" top="" position="border"
borderPosition="sc" parent_id="main" />

<part id="changePasswd" height="" width="" left="" top="" position="border"
borderPosition="ec" parent_id="main" />

<part id="contactDetails" height="" width="" left="" top="" position="border"
borderPosition="wc" parent_id="main" />

<part id="logout" height="" width="" left="" top="" position="border"
borderPosition="nw" parent_id="main"  />

<part id="dummyButton1" height="" width="" left="" top="" position="border"
borderPosition="sw" parent_id="main"  />

<part id="dummyButton2" height="" width="" left="" top="" position="border"
borderPosition="se" parent_id="main"  />

<part id="dummyButton3" height="" width="" left="" top="" position="border"
borderPosition="ne" parent_id="main"  />

<part id="dummyButton4" height="" width="" left="" top="" position="border"
borderPosition="cen" parent_id="main"  />

<part id="northBar" height="50px" width="" left="" top="" position="border"
borderPosition="north" parent_id="main"  />

<part id="westBar" height="" width="50px" left="" top="" position="border"
borderPosition="west" parent_id="main"  />
```

```
<part id="southBar" height="" width="" left="" top="" position="border"
borderPosition="south" parent_id="main"  />

<part id="eastBar" height="" width="" left="" top="" position="border"
borderPosition="east" parent_id="main"  />

</layout>

<content id="NewDesignPageCreationContent">

<part id="main"  value="nadmin" valuetype="reference"/>

<part id="topNarrowPanel"  value="North Center Button" valuetype="inline"/>

<part id="lastLogin" value="Last Login:Today" valueType="inline" />

<part id="changePasswd" value="Change Password" valueType="inline" />

<part id="contactDetails" value="Contact Details" valueType="inline" />

<part id="logout" value="Logout" valueType="inline" />

<part id="dummyButton1" value="Login" valueType="inline" />

<part id="dummyButton2" value="Testing Dummy" valueType="inline" />

<part id="dummyButton3" value="Testing Dummy" valueType="inline" />

<part id="dummyButton4" value="Center Button" valueType="inline" />

<part id="northBar" value="NORTH" valueType="inline" />

<part id="westBar" value="WEST" valueType="inline" />

<part id="southBar" value="SOUTH" valueType="inline" />

<part id="eastBar" value="EAST" valueType="inline" />

</content>

<style id="NewDesignPageCreationStyle">

<part id="topNarrowPanel" value="[CDATA[border: 1px solid black;;background-
color:#FFFFFF]]" valuetype="inline"/>

<part id="lastLogin" value="[CDATA[border: 0px solid black;;background-
color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
```

```
<part id="changePasswd" value="[CDATA[border: px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="contactDetails" value="[CDATA[border: px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="logout" value="[CDATA[border: px solid black;;background-color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="dummyButton1" value="[CDATA[border: px solid black;;background-color:#0000EE;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="dummyButton2" value="[CDATA[border: px solid black;;background-color:#0000EE;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="dummyButton3" value="[CDATA[border: px solid black;;background-color:#0000EE;color:#000000;font-size:15px;font-weight:bold;text-align:center;]]" valuetype="inline"/>

<part id="dummyButton4" value="" valuetype="inline"/>

<part id="northBar" value="" valuetype="inline"/>

<part id="westBar" value="" valuetype="inline"/>

<part id="southBar" value="" valuetype="inline"/>

<part id="eastBar" value="" valuetype="inline"/>

</style>

<behaviour id="NewDesignPageCreationBehaviour">

</behaviour></interface>
```

Table 1 : XML Code Snippet for Border Layouts

b) Anchor Layout

If layout mode is anchor, neighbourPosition & indentation attributes needs to be filled in. neighbourPosition value can be “Left”, “Top”, “Right”, “Down”.

Indentation value can be any combination of “\\n” (new line),” \\t” (new tab) & “\\p “ (new paragraph)

Sample Demo with Blocks

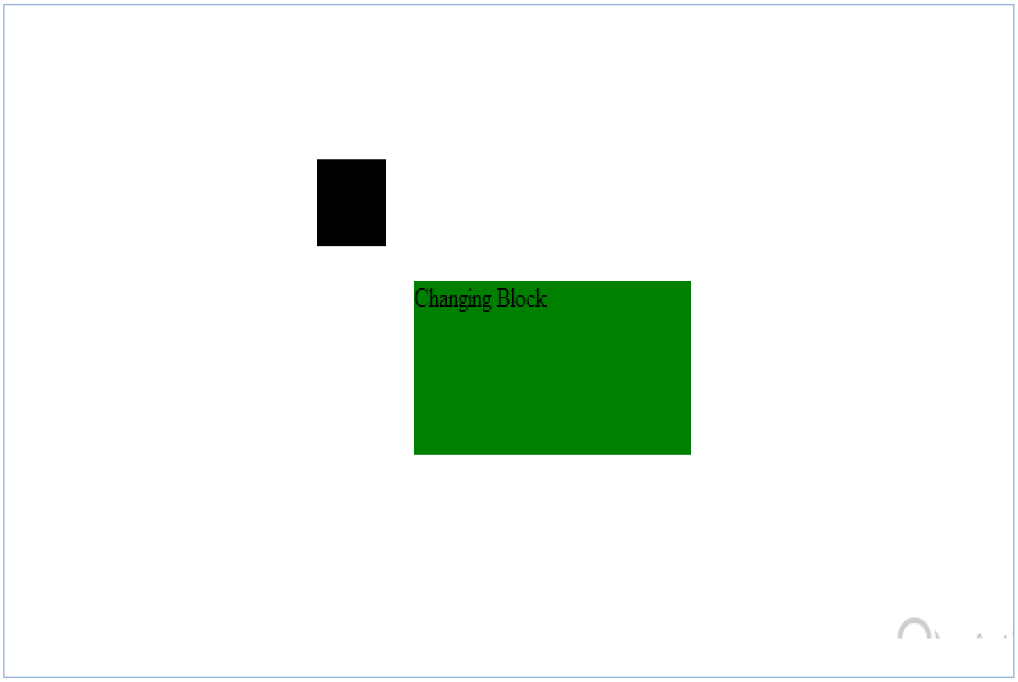


Figure 91: Enhanced Layout Design - Anchor Layout - Placing Block to the Right of Fixed Black Block, with indentation “\\n\\t”

```
<layout id="sampleLayout">

<part id="main" height="100%" width="100%" left="0px" top="2px"
position="absolute"/>

<part id="fixedDiv" height="50px" width="200px" position="border"
borderPosition=”center” parent_id="main"/>

<part id="changingDiv" height="50px" width="200px" position="anchor"
neighbout_id=” fixedDiv” neighbourPosition=”left”  indentation=”\\n\\t”
parent_id="main"/>

</layout>
```

XML Code Snippet for Anchor Based Position1

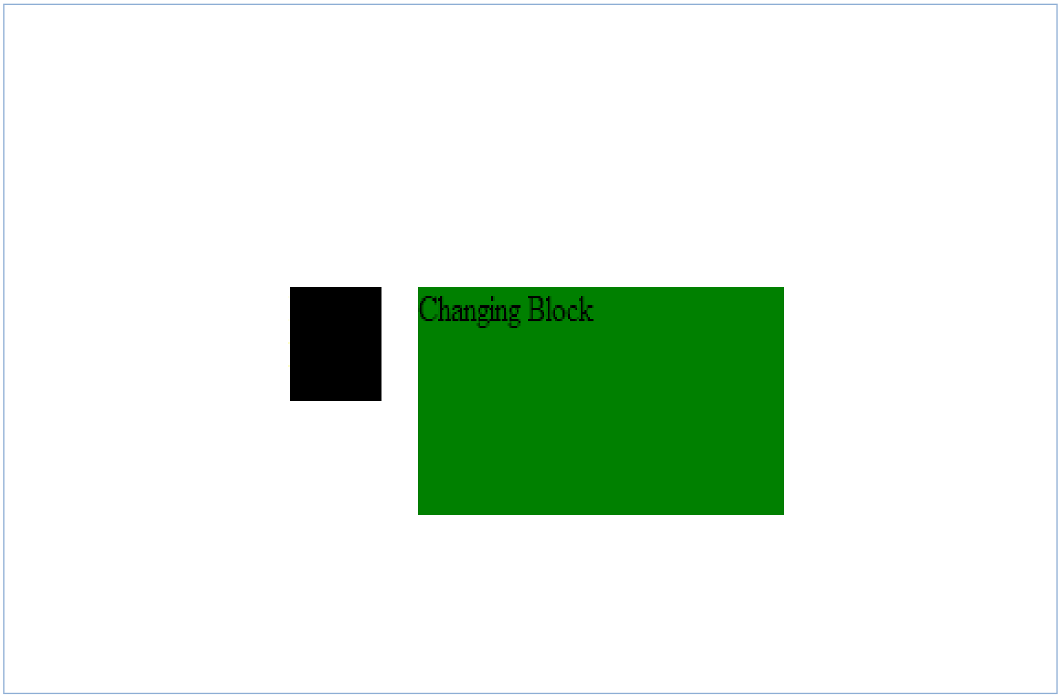


Figure 92: Enhanced Layout Design - Anchor Layout - Placing Block to the Right of Fixed Black Block, with indentation “\\t”

```
<layout id="sampleLayout">

<part id="main" height="100%" width="100%" left="0px" top="2px"
position="absolute"/>

<part id="fixedDiv" height="50px" width="200px" position="border"
borderPosition=”center” parent_id="main"/>

<part id="changingDiv" height="50px" width="200px" position="anchor"
neighbout_id=” fixedDiv” neighbourPosition=”left”  indentation=”\\t”
parent_id="main"/>

</layout>
```

XML Code Snippet for Anchor Based Position2

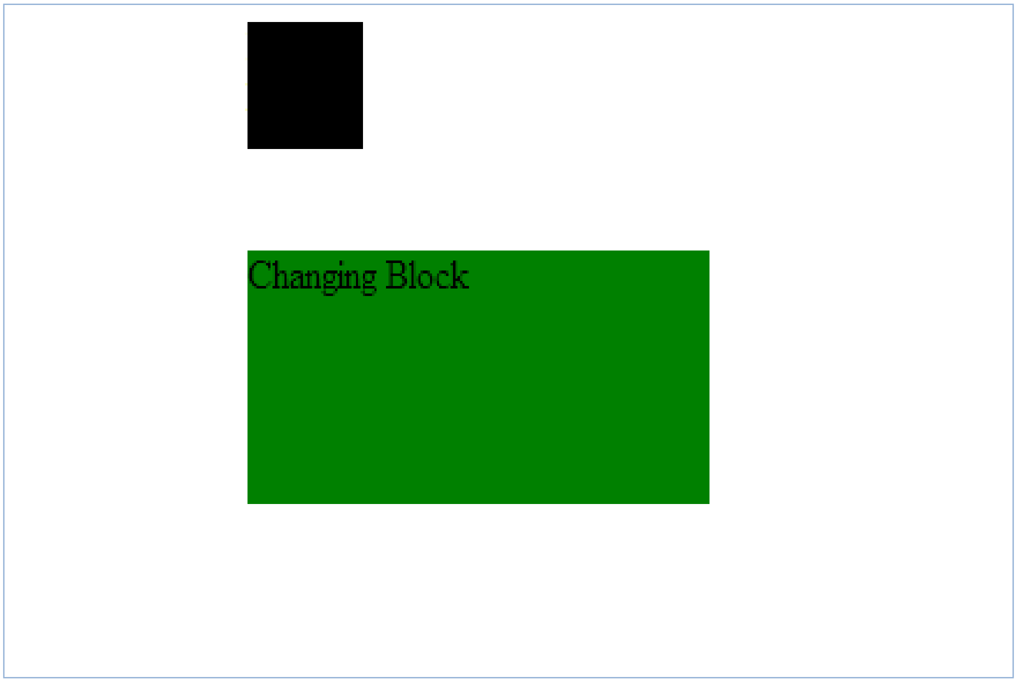


Figure 93: Enhanced Layout Design - Anchor Layout - Placing Block to the Right of Fixed Black Block, with indentation “\\p\\n”

```
<layout id="sampleLayout">

<part id="main" height="100%" width="100%" left="0px" top="2px"
position="absolute"/>

<part id="fixedDiv" height="50px" width="200px" position="border"
borderPosition=”center” parent_id="main"/>

<part id="changingDiv" height="50px" width="200px" position="anchor"
neighbout_id=” fixedDiv” neighbourPosition=”top”  indentation=”\\p\\n”
parent_id="main"/>

</layout>
```

XML Code Snippet for Anchor Based Position3

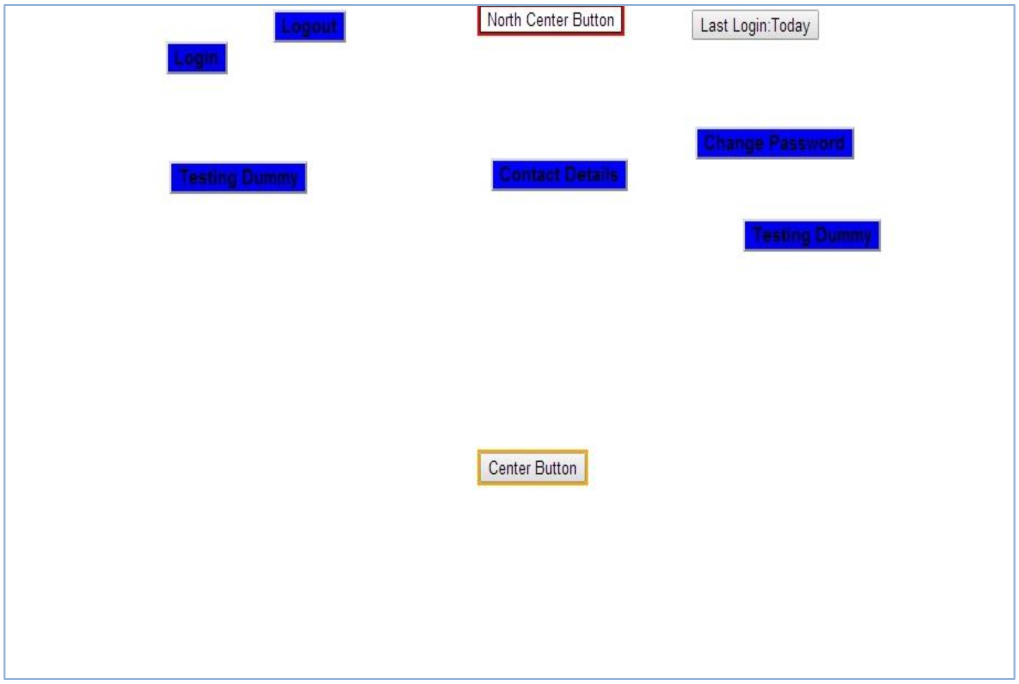


Figure 94: Enhanced Layout Design - Anchor Layout - Demo of different anchor positioned blocks

```
<interface id="NewDesignPageCreation" title="NewDesignPageCreation"
type="Interface">
<structure>
<!--Containers - Start-->
<part id="main" class="label"/>
<part id="topNarrowPanel" class="button"/>
<part id="lastLogin" class="button" />
<part id="changePasswd" class="button" />
<part id="contactDetails" class="button" />
<part id="logout" class="button" />
<part id="dummyButton1" class="button" />
<part id="dummyButton2" class="button" />
<part id="dummyButton3" class="button" />
<part id="dummyButton4" class="button" />
</structure>

<layout id="NewDesignPageCreationLayout">
<part id="main" height="100%" width="100%" left="1px" top="1px"
position="absolute"/>
<part id="topNarrowPanel" height="" width="" left="" top=""
position="border" borderPosition="NORTH-CENTER" parent_id="main"/>
<part id="lastLogin" height="" width="" left="" top="" position="anchor"
neighbour_id="topNarrowPanel" neighbourPosition="LEFT"
```

```
indentation="\\t\\t\\t" parent_id="main"/>
<part id="changePasswd" height="" width="" left="" top=""
position="anchor" neighbour_id="lastLogin" neighbourPosition="TOP"
indentation="\\n\\n\\n" parent_id="main" />
<part id="contactDetails" height="" width="" left="" top=""
position="anchor" neighbour_id="changePasswd"
neighbourPosition="Right" indentation="\\n\\t\\t\\t" parent_id="main" />
<part id="logout" height="" width="" left="" top="" position="anchor"
neighbour_id="topNarrowPanel" neighbourPosition="RIGHT"
indentation="\\t\\t\\t\\t\\t\\t" parent_id="main" />
<part id="dummyButton1" height="" width="" left="" top=""
position="anchor" neighbour_id="logout" neighbourPosition="RIGHT"
indentation="\\n\\t\\t" parent_id="main" />
<part id="dummyButton2" height="" width="" left="" top=""
position="anchor" neighbour_id="dummyButton1"
neighbourPosition="TOP" indentation="\\n\\n\\n" parent_id="main" />
<part id="dummyButton3" height="" width="" left="" top=""
position="anchor" neighbour_id="changePasswd"
neighbourPosition="TOP" indentation="\\n\\n\\t\\t" parent_id="main" />
<part id="dummyButton4" height="" width="" left="" top=""
position="border" borderPosition="CENTER" parent_id="main"  />
</layout>

<content id="NewDesignPageCreationContent">
<part id="main"  value="nadmin" valuetype="reference"/>
<part id="topNarrowPanel"  value="North Center Button"
valuetype="inline"/>
<part id="lastLogin" value="Last Login:Today" valueType="inline" />
<part id="changePasswd" value="Change Password" valueType="inline" />
<part id="contactDetails" value="Contact Details" valueType="inline" />
<part id="logout" value="Logout" valueType="inline" />
<part id="dummyButton1" value="Login" valueType="inline" />
<part id="dummyButton2" value="Testing Dummy" valueType="inline" />
<part id="dummyButton3" value="Testing Dummy" valueType="inline" />
<part id="dummyButton4" value="Center Button" valueType="inline" />
</content>

<style id="NewDesignPageCreationStyle">
<part id="topNarrowPanel" value="[CDATA[border: 1px solid
black;;background-color:#FFFFFF]]" valuetype="inline"/>
<part id="lastLogin" value="" valuetype="inline"/>
<part id="changePasswd" value="[CDATA[border: px solid
black;;background-color:#0000FF;color:#000000;font-size:15px;font-
weight:bold;text-align:center;]]" valuetype="inline"/>
<part id="contactDetails" value="[CDATA[border: px solid
black;;background-color:#0000FF;color:#000000;font-size:15px;font-
weight:bold;text-align:center;]]" valuetype="inline"/>
<part id="logout" value="[CDATA[border: px solid black;;background-
```

```
color:#0000FF;color:#000000;font-size:15px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="dummyButton1" value="[CDATA[border: px solid
black;;background-color:#0000EE;color:#000000;font-size:15px;font-
weight:bold;text-align:center;]]" valuetype="inline"/>
<part id="dummyButton2" value="[CDATA[border: px solid
black;;background-color:#0000EE;color:#000000;font-size:15px;font-
weight:bold;text-align:center;]]" valuetype="inline"/>
<part id="dummyButton3" value="[CDATA[border: px solid
black;;background-color:#0000EE;color:#000000;font-size:15px;font-
weight:bold;text-align:center;]]" valuetype="inline"/>
<part id="dummyButton4" value="" valuetype="inline"/>
</style>
<behaviour id="NewDesignPageCreationBehaviour">

</behaviour></interface>
```

XML Code Snippet for Multiple Anchor Based Positions

6.2 LearnITy Framework XML Validation Feature

LearnITy Framework encounters three main XML files at configuration level, named *manifest.xml*, *interfacerole.xml* and *interface.xml* (for each page). These three files need to be created by developers who intend to use this framework for web page design. In its current format there is no scope for validating XML files, prepared by developers.

XSD Schema validation method is introduced in the new design of the framework. Each of the above mentioned XML files will be validated during uploading, against pre-defined XSD schema files. If the validation fails, uploading stops there and user is provided with proper error message on screen. On successful validation only backend processing logic proceeds with HTML generation part and storing the components into database.

Below screenshots show the sample status pages of a successful and unsuccessful validation:

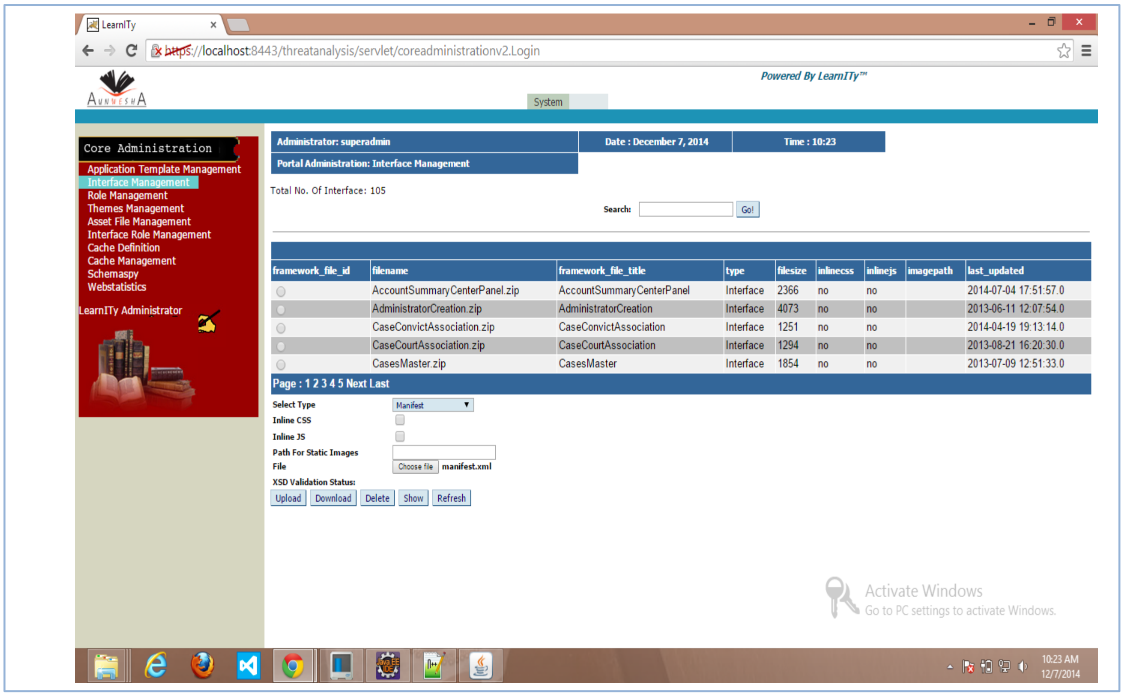


Figure 95: XSD Validation - Uploading Manifest.xml

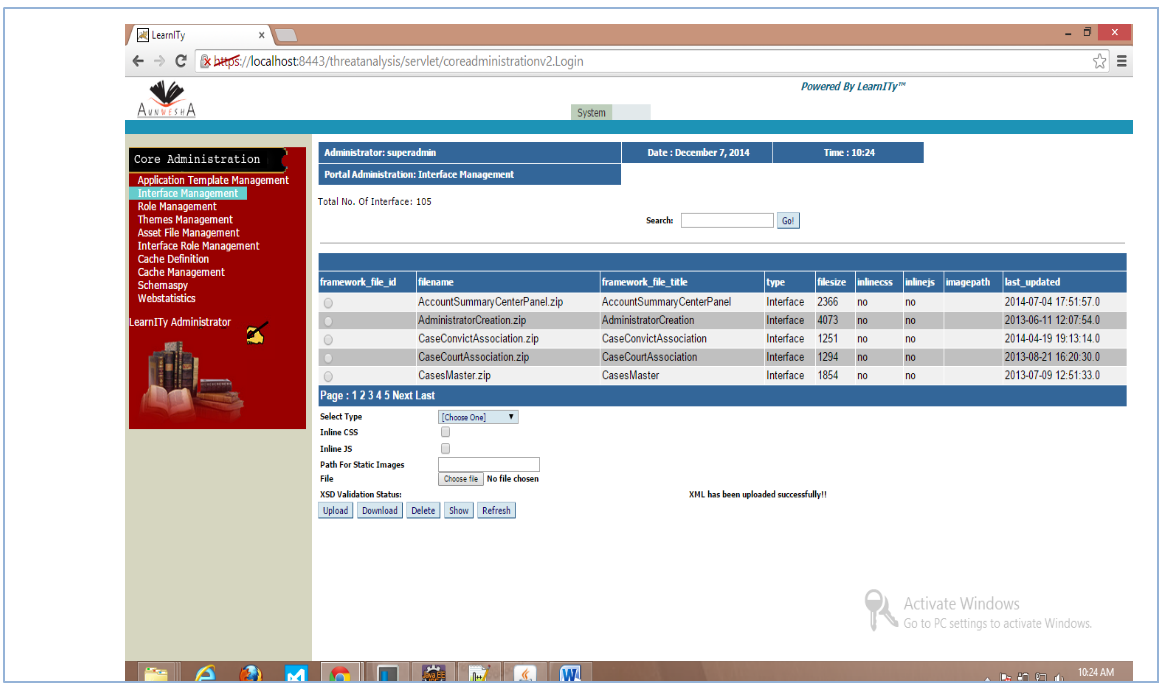


Figure 96: XSD Validation - Manifest.xml uploaded successfully

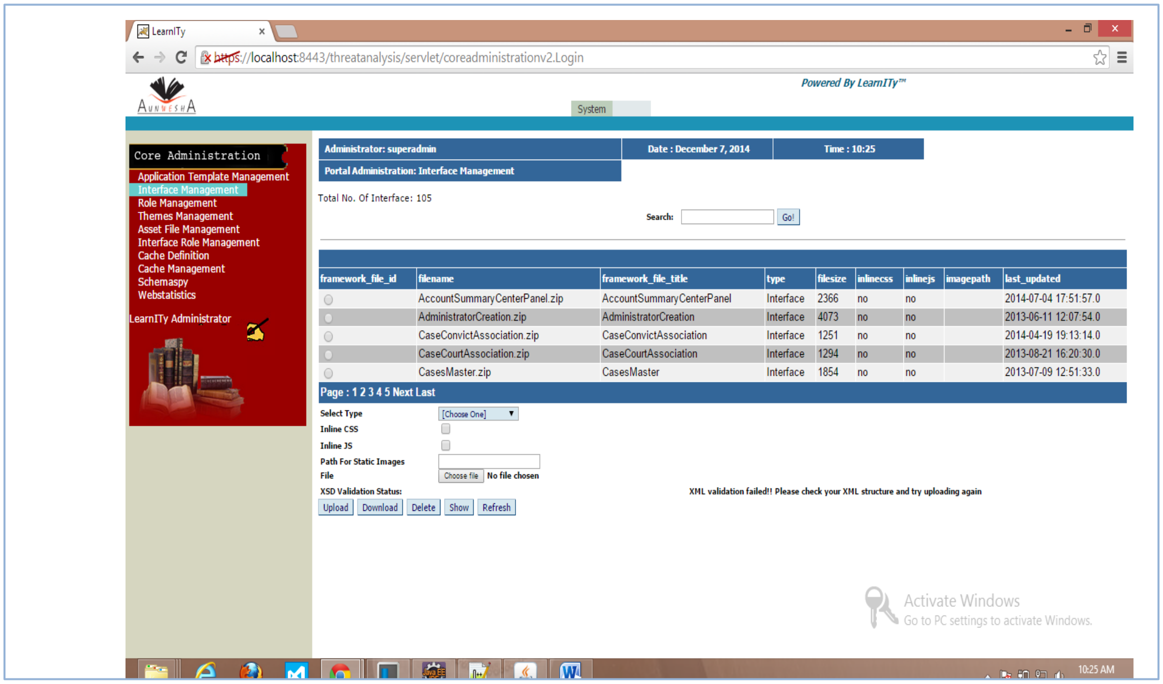


Figure 97: XSD Validation - XML validation through XSD failed for Manifest.xml

6.3 Comparison with Baseline Version and Advantages of Enhanced Design in LearnITy Framework

6.3.1 Baseline Version of LearnITy Framework

Below are some features available in the baseline version of design in the target framework:

- The interface xml code is structured as combination of six sub-sections : Structure, Layout, Content, Style, Behaviour and Resource – represented as six different XML tags
- Structure taking care of all the components that exist and parent-child relationship of the components – it creates the unique ids for each component
- Layout taking care of positioning the elements on the page with support of only absolute layout
- Layout tag containing attributes as follows:
    - ✓ *<Id>* – part id which is unique for each component
    - ✓ *<height>* - height of the component to be placed
    - ✓ *<width>* - width of the component to be placed
    - ✓ *<left>* - x-position of the laid out component

  - ✓ *<top>* - y-position of the laid out component
  - ✓ *<position>* - value "absolute" for denoting absolute positioning
  - ✓ *<scrolling>* - provision of providing scrolls to iFrame components
- Content taking care of the value or content of any components that will be displayed
- Style taking care of the themes or look-and-feel of existing components, having below attributes:
  - ✓ *<id>* - part id of the component which is to be styled
  - ✓ *<value>* - CSS tags for inline components
  - ✓ *<valuetype>* - "inline" or "reference" based on how styling is provided. Reference valuetype allows css files to be stored separately
- Only CSS Style sheet based theme is currently supported in baseline version of the framework
- Behaviour taking care of various events as mentioned below:
  - ✓ On double click row
  - ✓ Grid complete
  - ✓ Load complete
  - ✓ On click
  - ✓ On change
  - ✓ Invoke URL
  - ✓ On load, On click
  - ✓ On key press
  - ✓ Body on load
- Navigation is currently supported by behaviour tag only, on pre-defined event execution
- Resource taking care of all static resources like image, JavaScript, CSS stylesheet etc.

6.3.2 Improved Design in LearnITy Framework

With the implementation of enhanced design below features can be achieved:

- The improved design is drafted for Layout, Navigation and Style enhancement, and till now implemented partially for Layout and Style
- Enhanced design for Skin or Style is categorized as follows:
  - ✓ Part id based skinning – style information are to be provided at individual part level - this feature is currently present in the baseline version and subject to improvement

    Below is the sample code snippet:

```
<interface id="home"… >
<structure>
<part id="myButton" class="button" …></part>
</structure>
…
<style>
<part id="myButton" type="radio"  disabled="false"
buttonColor="#006633" buttonStyle="circle"
contentFont="23;10;true;false;false;#FF9999;#000055;AA" 3DView="no"
shadow="no"></part>
</style>
</interface>
```

- ✓ Component / Class based skinning – style or skinning are provided at class level, the skinning attribute set will be applicable to all instances of the component in that interface

  Below is the sample code snippet:

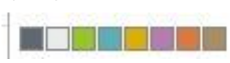
```
<theme id="indiafirst">
<themeselement class="button" type="reference" cssclasses="c2
c3"></themeselement>
<themeselement class="inputtext" type="reference" cssclasses="c1"
></themeselement>

<cssfile>
<file name="themes1.css"></file>
<file name="themes2.css"></file>
</cssfile>
</theme>
```

- ✓ Theme based skinning – few standard themes will be provided by the framework and developers will also be given the flexibility of creating custom themes as per their own choice

  Below is the sample code snippet:

```
<?xml version="1.0"?>
<theme id="transtutortheme">

<appTheme         color="pageBackColor;headingColor;quoteColor;nor
malLabelColor;frameBackColor;chartBarColor1;chartBarColor2;hyperli
nkColor"
font="headingFont;bodyFont;headingSize;bodySize"
rectEdge="RoundedCorner / SharpCorner / DiagonalRounded"
circleEdge="Filled / MarginFilled"
```

```
borderWeight="--weight in pixel--" >

</appTheme>
</theme>
```

- Few standard themes with their attributes are finalized at design stage as follows:
  - ✓ Office Theme
    - Color: 
    - Font : “Callibri; Cambria; 12; 10”
    - RectEdge: “sharpCorner”
    - CircularEdge: “filled”
    - BorderWeight : “2”
  - ✓ Afternoon Theme
    - Color: 
    - Font : “Cambria;Callibri; 14; 12”
    - RectEdge: “diagonal_rounded”
    - CircularEdge: “filled”
    - BorderWeight : “1”
  - ✓ Grayscale Theme
    - Color: 
    - Font : “Lucida Sans;Book Antiqua; 14; 12”
    - RectEdge: “diagonal_rounded”
    - CircularEdge: “margin_filled”
    - BorderWeight : “2”
  - ✓ Composite Theme
    - Color: 
    - Font : “Calibri;Calibri; 13; 10”
    - RectEdge: “roundCorner”
    - CircularEdge: “margin_filled”
    - BorderWeight : “2”
- Component based skinning to supersede Theme based skinning
- Part id based skinning to supersede Component based skinning
- Eight color attributes are: pageBackColor, headingColor, quoteColor, normalLabelColor, frameBackColor, chartBarColor1, chartBarColor2, hyperlinkColor
- Four font attributes are: headingFont, bodyFont, headingSize, bodySize
- RectEdge attribute’s probable values are: “roundedCorner”, “sharpCorner”, “diagonalRounded”

- CircleEdge attribute's probable values are: "filled" and "marginFilled"
- pageBackColor value will be applicable to below areas:
  - ✓ page background
  - ✓ form background
  - ✓ grid row
  - ✓ column heading
- frameBackColor value will be applicable to below areas:
  - ✓ image background
  - ✓ button background
  - ✓ list background
  - ✓ grid background
  - ✓ chart background
  - ✓ iframe background
- normalLabelColor value will be applicable to below areas:
  - ✓ label
  - ✓ button content
  - ✓ combobox content
  - ✓ list content
  - ✓ grid content
  - ✓ iframe content
  - ✓ tree content
  - ✓ any border color
  - ✓ page title
- Already implemented style enhancements on LearnITy Framework are detailed in this report - (Reference [16])
- As part of enhanced navigation design, rule based navigation implementation is planned where navigation rules need to be provided in configuration file and this will be applicable for the whole application being developed.

  The pseudo code for rule based config file is provided below:

```
<navigation>  <!--optional section to be added to an interface.xml file-->

 <navigation_items  no_of_navigation_items =””>

   <navigation_item navigation_item_id=””>
<from-page fromInterfaceID =" " ></from-page>

     <navigation_cases no_of_navigation_cases = “”>

       <navigation-case navigation_case_id =””  requestedObjectType =””
```

```
httpRequestType = "" toPageId="" behavior="" componentId="" >

            <navigation_parameters  no_of_navigation_parameters ="">
<param param_id =""  param-name =""  param-type ="" param-value =""  param-part-
id ="" value_required_at ="" to_field_name ="">

             </param>
</navigation_parameters  >
</navigation_case>
</navigation_cases>
</navigation_item>
 </navigation_items>
</navigation>
```

However, this part is yet to be implemented on LearnITy Framework and considered as future scope of work.

- Layout enhancement design is classified into Container Layout and Component Layout, as mentioned in detail within the previous section
- Container layout concept helps arranging components in stated orientation with predefined gaps in between, without any extra coding at component level
- Container layouts are targeted to build flexible layout feature as provided by already existing layout managers, with below mentioned advantages:
  - ✓ Correctly positioning components that are independent of fonts, screen resolutions, and platform differences
  - ✓ Intelligent component placement of containers that are dynamically resized at runtime
  - ✓ Ease of translation – if a string increases in length after translation, the associated components stay properly aligned
- Nested layout support is achieved at container level, by using any Container layout
  e.g., a horizontalContainer may contain no. of components which are again vertical containers or diagonal containers, with nested level containers having ability to contain either basic level components or some other containers further.
- For creating Container layout feature new components or part classes are introduced altogether, which will serve as different kinds of containers only
- The pseudo code for Container Layout is provided below:

```
…
<structure>
<part id = "horizontalContainerId" class="horizontaContainer"/>
```

```
<part id="hButton1" class="button" />
<part id="hButton2" class="button" />
…
</structure>

<layout>
…
<part id=" horizontalContainerId" … parent_id="main"/>
<part id="hButton1" … parent_id=" horizontalContainerId"/>
<part id="hButton2" … parent_id=" horizontalContainerId"/>
…
</layout>
…
```

- With Component level Border layout usage, any component is free to be placed at any of the thirteen border positions: North Border, South Border, East Border, West Border, North-East, North-West, South-East, South-West, North-Center, South-Center, West-Center, East-Center & Center
- With Component level Anchor layout usage, any component can be placed at any location of the page by using already placed component's reference.
- Anchor layout provides provision for any kind of alignment by new-line ("\\n"), new-tab ("\\t") and new-paragraph ("\\p") indentation features
- Component layout's Border positioning is influenced by Swing framework's Border Layout which simplified the sample screen implementation to a large extent
- Component layout's Anchor positioning is influenced by QT QML framework's Anchor layout which made it possible to place components at any location on the page
- In QT QML anchor based positioning there was no scope of changing the indentation of the newly placed components, which led to introducing "indentation" attribute whole value can be any count and combination of new line (\\n), new tab (\\t) , new paragraph (\\p)
- The pseudo code for Component's Border Layout is provided below:

```
…
<structure>
<part id = "northBorder" class="label"/>
<part id="westCenter" class="image" />
<part id="northEast" class="button" />
…
</structure>

<layout>
```

```
…
<part id=" northBorder" … position="border" borderPosition="north"/>
<part id=" westCenter" … position="border" borderPosition="wc"/>
<part id=" northEast" … position="border" borderPosition="ne"/>
…
</layout>
…
```

- The pseudo code for Component's Anchor Layout is provided below:

```
…
<structure>
<part id = "center" class="image"/>
<part id="button1" class="button" />
<part id="button2" class="button" />
…
</structure>

<layout>
…
<part id=" center" … position="border" borderPosition="center"/>
<part id=" button1" … position="anchor" neighbour_id=" center"
neighbourPosition="right" indentation="\\t\\t\\t"/>

<part id=" button2" … position="anchor" neighbour_id=" button1"
neighbourPosition="top" indentation="\\n\\t\\t"/>
…
</layout>
…
```

- Multi-Column layout is supported by use of repetitive horizontalContainer and verticalContainer layout classes
- Positioning approach in LearnITy Framework is Adaptive and works with browser resizing.

  I.e. Browser size is determined beforehand by using JavaScript and according to that components are placed with the browser size reference. Sometimes component sizes are set in pixels, sometimes they are percentage size of browser size

- Developers need to put very minimal parameters for implementing any kind of layouts, while the framework still having support for Absolute Positioning as well

- Importing strong features of various frameworks layout, navigation and skin design into LearnITy Framework has the advantage because, this framework is declarative approach based and has a very easy learning curve for any kind of developers – knowledge of programming language is not needed as creation of pages is purely XML based
- Inclusion on XML file validation based on pre-defined XSD structure would help this framework in maintaining the consistency across all page designs, while making it easier for developers to know about the rules and constraints in writing pages

6.4 Implementation Notes

This section covers actual implementation perspectives of baseline version as well as improved design of the LearnITy Framework

6.4.1 Existing Structure of the Backend Code

- At the backend the Java classes come with the below folder structure:
  - ✓ coreadministrationv2
    - dbconnection
      - DataBaseLayer.java
    - sysmgmt
      - ThemeManagement.java
      - RoleManagement.java
      - ResourceInterface.java
      - InterfaceRoleManagement.java
      - InterfaceManagement.java
      - FolderZiper.java
      - DownloadThemes.java
      - DownloadInterface.java
      - DownloadResource.java
      - DownloadDefaultXML.java
      - DownloadAsset.java
      - CacheManagement.java
      - CacheDefinition.java
      - AssetFileManagement.java
      - ApplicationTemplateManagement.java
    - utility
      - TableExtension.java
      - TRExtension.java
    - Login.java
    - Logout .java

        - Welcome.java
- interfaceenginev2
    - Xmlcreator.java
    - ValidatorFunction.java
    - ThemeCss.java
    - TestHtml.java
    - Snippetxml.java
    - SelectDataProviderServlet.java
    - ResourceVideo.java
    - ResourceProps.java
    - ResourcePdf.java
    - ResourceJS.java
    - ResourceImage.java
    - ResourceHtml.java
    - ResourceCss.java
    - ResourceAnimation.java
    - PortalServlet.java
    - PortalEngine.java
    - Part.java
    - NewDataBaseLayer.java
    - jsonWriter.java
    - InterfaceCachePojo.java
    - GsonChildjs.java
    - FileUploaderPojo.java
    - FileDownloaderPojo.java
    - EncryptedFileDownloaderPojo.java
    - DisplayEngine.java
    - DBGridQueryEditorServletForMulti.java
    - DBGridQueryEditorServlet.java
    - CustomEditAction.java
- comv2
    - JSPGrid
        - JSPHeader.java
        - JSPGridPro2.java
        - JSPGridPro1.java
        - JSPGridPro.java
        - JSPGrid2.java
        - JSPGrid.java
        - JSPCol2.java
        - JSPCol.java

- JSPCell2.java
- JSPCell.java

- param
  - CoreAdminInitHostInfo.java
  - DataBaseLayer.java

- Feature for Login to CoreAdmin site is being handled by Coreadministrationv2/Login.Java code
- Interface management related activities are handled by Coreadministrationv2/sysmgmt/InterfaceManagement.java code. The detailed methods for this code are provided below
  - ✓ function "addLayout_onclick()" is called when user selects a file and clicks on "Upload" button. This method forms the URL which is invoking the required page to be displayed
  - ✓ uploading "manifest.xml" file is taken care by method uploadManifest()
  - ✓ uploading "interfacerole.xml" file is taken care by method uploadRoleXML()
  - ✓ uploading "interface.xml" file is taken care by method uploadInterface()
  - ✓ each of the above methods parses the uploaded XML and stores the attribute values in respective schema tables by using Coreadministrationv2/dbconnection/DatabaseLayer.java methods
- In DatabaseLayer.java code separate methods are available which take care of inserting data into individual tables
- Interfaceenginev2/DisplayEngine.java code is used for actual rendering of page when browser hits the associated URL.
  - ✓ doPost() method is calling another method named createParentLayout() with all associated ids like interface id, layout id, style id, behavior id etc. for a page
  - ✓ createParentLayout() method is calling Interfaceenginev2/NewDataBaseLayer.java getParentLayout() method with interface id and layout id
  - ✓ getParentLayout() method of Interfaceenginev2/NewDataBaseLayer.java is searching "layout" table in schema structure and creating a vector with retrieved parameters; It returns the vector to calling method createParentLayout()
  - ✓ createParentLayout() method is again calling its own class's getChildNode() method by passing all relevant ids, which are gained from returned vector

- ✓ getChildNode() is actually responsible for retrieving all the nested elements from parent id tagging and it further calls method createLayout()
- ✓ createLayout() method is responsible for generating html tags for each and every component and placing it accordingly

6.4.2 Modified Structure of the Backend Code

The improved design has led to change in four areas namely

- Layout table schema change
- coreadministrationv2\sysmgmt\ InterfaceManagement.java
- coreadministrationv2\dbconnection\DatabaseLayer.java
- interfaceenginev2\DisplayEngine.java
- interfaceenginev2\NewDatabaseLayer.java

a) Change in Layout Schema

In its original database layout table had these columns : id, height, width, left, top, position, parent_id, scrolling. Left and Top margins were to be provided for position value "absolute". Scrolling parameter is there for iframe scroll bars.

At database schema level table "layout" is modified to include below columns

- position: changed to accept values "border", "anchor" newly
- borderPosition: new columns to take any of 13 border location values
- neighbour_id: new column for position "anchor" only, it takes reference component's id
- neighbourPosition: new column for position "anchor" only, it takes any of 4 values top, left, bottom, right
- indentation: new column for position "anchor" only, it takes any combination of newline, new tab and new space characters

b) Change in Source File - **InterfaceManagement.java**

Changes in coreadministrationv2\dbconnection\InterfaceManagement.java are in methods uploadInterfaceFragment() and uploadInterface()

These methods include additional layout parameters in the parser field, to be inserted into database

c) Change in Source File - **DatabaseLayer.java**

Changes in coreadministrationv2\dbconnection\DatabaseLayer.java

Method insertLayout() is changed to include all parameters which are newly added to layout schema

d) Change in Source File - DatabaseLayer.java

Below changes are implemented:

- New classes HorizontalContainer, VerticalContainer, LeftDiagonalContainer, RightDiagonalContainer are introduced
- Container level elements need to be placed on main page using absolute positioning as depicted in the interface code

```
<structure>
<part id="main" class="verticalContainer"/>
<part id="topPanel" class="horizontalContainer"/>

<part id="logo" class="image"/>
<part id="rightPanel" class="verticalContainer"/>
...
</structure>
<layout id="NewDesignPageCreationLayout">
<part id="main" height="100%" width="100%" left="1px"
top="1px" position="absolute"/>
<part id="topPanel" height="80px" width="1000px" left=""
top="" position="border" borderPosition="NORTH-CENTER"
parent_id="main"/>

<part id="logo" height="80px" width="500px"
parent_id="topPanel"/>
<part id="rightPanel" height="100px" width="500px"
parent_id="topPanel"/>
...
</layout>
```

- Container implementation backend pseudo code is like this:

```
If component class = “horizontalContainer”
1.Create a “div” with specified height, width, x, y positioning
2.Style the div as table-row
3.Set border spacing of table as common spacing in container
4.Include child components as table cell

If component class = “verticalContainer”
1.Create a “div” with specified height, width, x, y positioning
2.Include child components –they will automatically be added
vertically
```

If component class = "leftDiagonalContainer"
1.Create a "div" with specified height, width, x, y positioning
2.Create a table and include that table inside above div
3.For each child maintain child count
4.Set Child 1 at cell (1,1); Set Child 2 at cell (2,2)..like this all children need to be set

If component class = "rightDiagonalContainer"
1.Create a "div" with specified height, width, x, y positioning
2.Create a table and include that table inside above div
3.For each child maintain child count, have largest child count as well
4.Set Child 1 at cell (1,n); Set Child 2 at cell (2,n-1)..like this all children need to be set, n being largest child count

- Component Layouts for Border positioning are implemented for thirteen different border positions. This calculation is linear and based on differential calculation approach

If position is "border" and borderPosition is "East" – component to be placed along full east border
1.Get page width by javaScript
2.If component width is provided by developer, left margin width will be page width – component width
3. If component width is not provided by developer, left margin width will be page width
4.Left Margin will be calculated as percentage of actual page width, before positioning the component

If position is "border" and borderPosition is "South" – component to be placed along full south border
1.Get page height by javaScript
2.If component height is provided by developer, top margin height will be page height – component height
3. If component height is not provided by developer, top margin height will be page height
4.Top Margin will be calculated as percentage of actual page height, before positioning the component

If position is "border" and borderPosition is "West" – component to be placed along full west border
1.No calculation of width or height is required, as left and top margin of component will be 0, 0 by default
2. Component's height, width will be as provided by developer

If position is "border" and borderPosition is "North" – component to be placed along full north border
1.No calculation of width or height is required, as left and top margin of component will be 0, 0 by default
2. Component's height, width will be as provided by developer

If position is "border" and borderPosition is "Center" – component to be placed at center of the page
1.Get page width, page height by javaScript
2.Derive mid page width, mid page height by dividing the actual width and height by two respectively
3.If component width is provided by developer, left margin width will be mid page width – mid component width
4.If component width is not provided by developer, left margin width will be mid page width only
5.If component height is provided by developer, top margin height will be mid page height – mid component height
6.If component height is not provided by developer, top margin height will be mid page height only
7. Finally left margin and top margin are to be converted as percentages of page width and page height only, for positioning component

If position is "border" and borderPosition is "North-East"
1.Get page width by javaScript
2.If component width is provided by developer, left margin width will be page width – component width
3. If component width is not provided by developer, left margin width will be page width
4.Left Margin will be calculated as percentage of actual page width, top margin as zero(0) before positioning the component

If position is "border" and borderPosition is "South-East"
1.Get page width & page height by javaScript
2.If component width is provided by developer, left margin width will be page width – component width
3. If component width is not provided by developer, left margin width will be page width
4.If component height is provided by developer, top margin height will be page height – component height
5. If component height is not provided by developer, top margin height will be page height
6.Left Margin will be calculated as percentage of actual page width, top margin o be calculated as percentage of actual page height before positioning the component

If position is "border" and borderPosition is "South-West"
1.Get page height by javaScript
2.If component height is provided by developer, top margin height will be page height – component height

3. If component height is not provided by developer, top margin height will be page height
4.Top Margin will be calculated as percentage of actual page height, left margin as zero(0) before positioning the component

If position is "border" and borderPosition is "North-West"
1.No calculation of width or height is required, as left and top margin of component will be 0, 0 by default
2. Component's height, width will be as provided by developer

If position is "border" and borderPosition is "West-Center"
1.Get page height by javaScript, mid page height is derived from that
2.If component height is provided by developer, top margin height will be mid page height – mid component height
3. If component height is not provided by developer, top margin height will be mid page height
4.Top Margin will be calculated as percentage of actual page height, left margin as zero(0) before positioning the component

If position is "border" and borderPosition is "East-Center"
1.Get page height, page width by javaScript, mid page height & mid page width are derived from that respectively
2.If component height is provided by developer, top margin height will be mid page height – component mid height
3. If component height is not provided by developer, top margin height will be mid page height
4.If component width is provided by developer, left margin width will be page width – component width
5. If component width is not provided by developer, left margin width will be page width
6.Top Margin will be calculated as percentage of actual page height, left margin as percentage of actual page width before positioning the component

If position is "border" and borderPosition is "South-Center"
1.Get page height, page width by javaScript, mid page height & mid page width are derived from that respectively
2.If component height is provided by developer, top margin height will be page height – component height
3. If component height is not provided by developer, top margin height will be page height
4.If component width is provided by developer, left margin width will be mid page width – mid component width
5. If component width is not provided by developer, left margin width will be mid page width
6.Top Margin will be calculated as percentage of actual page height, left margin as percentage of actual page width before positioning the component

If position is "border" and borderPosition is "North-Center" *1.Get page width by javaScript, mid page width is derived from that* *2.If component width is provided by developer, left margin width will be mid page width – mid component width* *3. If component width is not provided by developer, left margin width will be mid page width* *4.Left Margin will be calculated as percentage of actual page width, top margin as zero(0) before positioning the component*

- Component Layouts for Anchor positioning are implemented for four different anchor positions. This calculation is linear and based on differential calculation approach

If position is "anchor" *1.Get page width and page height from javaSctipt* *2. Derive neighbour's offsetwidth, offsetheight (after considering left, top margin distance of the neighbour from page border)* *3.if new component is to be placed at right of neighbour, neighbour's offsetwidth is summed up with horizontal indentation , offsetheight of neighbour is summed up with vertical indentation and that creates new location for the component* *4. For new components positioning at the left of neighbour, width needs to be deducted from offsetwidth; offsetHeight is depending upon vertical indentation* *5. For down position of neighbour, offsetHeight needs to be summed up with component height; offsetWidth is depending upon horizontal indentation* *6. For top position of neighbour, offsetHeight is deducted from component height; offsetWidth is depending upon horizontal indentation* *7. All margin calculations are in percentage of page dimension*

e) Change in Source File - **NewDatabaseLayer.***java*

The changes in this file are due to the change in query parameters in retrieving data from database tables, due to schema change

The methods which are changed in this code are : getlayoutinformationchild(), getParentPartClass()

7. Future Research Work

The motive of this research work is to provide an abstraction layer to user where user can provide different rule/configuration for different feature incorporation in the GUI framework without writing actual html code. The future work for improvements may concentrate more on user driven rule based more flexible approach for html code generation of different component with respect to skin , layout and navigation. Below we outlined some features.

7.1 Implementation of Theme Based Styling

Theme based styling is a very convenient way to decorate the application pages with wide variety of styles as available on Microsoft Office tools. Application wide html would be generated using theme based configuration xml as per user choice.

7.2 Implementation of Rule Based Navigation Support

Rule based navigation with parameter passing from one page to other page, data retention from page to page and availability of parameter values at client side and server side approach provides the most convenient way of navigating back and forth of an application.

7.3 Implementation of Responsive Layout with Component Resizing

Apart of providing provision for positioning elements anywhere on the page by Container layouts and Border & Anchor based Component layout, there are further scope of improvements in terms of Responsive Layout considering device screen sizes and user interaction with devices. It poses a more user friendly way of laying out components on a screen and adjusting as per the requirement. (References[15, 31, 32])

8. References

[1]

Java SWING Tutorial *< http://www.javabeginner.com/java-swing/java-swing-tutorial>*

Layout Managers | JFormDesigner – Java/Swing GUI Designer *<http://www.formdev.com/jformdesigner/doc/layouts/>*

[2]

Java SWT Tutorial *<http://zetcode.com/gui/javaswt/>*

SWT Tutorial *<http://www.vogella.com/tutorials/SWT/article.html>*

[3]

JSF Tutorial *< http://www.tutorialspoint.com/jsf/>*

JSF Page Navigation *<http://www.tutorialspoint.com/jsf/jsf_page_navigation.htm>*

JSF and PrimeFaces Tutorial with Eclipse *<http://www.coreservlets.com/JSF-Tutorial/>*

Java Beginners Tutorialrichfaces Archives – Java Beginners Tutorial *<http://javabeginnerstutorial.com/tag/richfaces/>*

PrimeFaces *<http://www.primefaces.org/gettingStarted>*

[4]

Get Started – Windows App Development *< https://dev.windows.com/en-us/getstarted>*

[5]

SiteMesh Tutorial *<http://www.hacktrix.com/sitemesh-tutorial>*

Start Using SiteMesh in 10 Minutes – SiteMesh 2 – Confluence *<http://wiki.sitemesh.org/wiki/display/sitemesh/Start+Using+SiteMesh+in+10+Minutes>*

[6]

QML Tutorial | Qt 4.8 *<http://doc.qt.io/qt-4.8/qml-tutorial.html>*

Building your first QML App | Ubuntu developer portal *<https://developer.ubuntu.com/en/apps/qml/tutorials/building-your-first-qml-app/>*

Anchor Layout – Kivy 1.9.1 dev documentation *<http://kivy.org/docs/api-kivy.uix.anchorlayout.html>*

[7]

A Complete Guide to Flexbox – CSS Tricks <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

CSS Flex Property <http://www.w3schools.com/cssref/css3_pr_flex.asp>

Using CSS flexible boxes – Web developer guide | MDN <https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Flexible_boxes>

[8]

Which Page Layout? Static, Liquid, Adaptive, or Responsive <http://blog.teamtreehouse.com/which-page-layout>

[9]

Responsive Web Design Basics – Web Fundamentals <https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/?hl=en>

[10]

Responsive Web Design Patterns|This is Responsive <http://bradfrost.github.io/this-is-responsive/patterns.html>

[11]

User interface markup language – Wikipedia, the free encyclopedia <http://en.wikipedia.org/wiki/User_interface_markup_language>

[12]

Cover Pages: XML Markup Languages for User Interface Definition <http://xml.coverpages.org/userInterfaceXML.html>

[13]

Building Multi-Platform User Interfaces with UIML - Mir Farooq Ali, Manuel A. Pérez-Quiñones, Eric Shel <http://arxiv.org/ftp/cs/papers/0111/0111024.pdf>

[14]

UIML.Net: an Open UIML Rendered for the .Net Framework – Springer <http://link.springer.com/chapter/10.1007%2F1-4020-3304-4_21>

[15]

A New Layout Method for Graphical User Interfaces - Adriano Scoditti
<http://iihm.imag.fr/publs/2010/intuilayout.pdf>

iOS 5 iPhone Rotation, View Resizing and Layout Handling – Techotopia
<http://www.techotopia.com/index.php/IOS_5_iPhone_Rotation,_View_Resizing_and_Layout_Handling>

Responsive Web Design: What It Is and How To Use It – Smashing Magazine
<http://www.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/>

[16]

Analysis of Popular Web Frameworks for Features Supported in Skin, Layout and Navigation and Design of Improved Skin Support in an Existing Framework – Joydip Das, Information Technology, Jadavpur University
<http://dspace.jdvu.ac.in/handle/123456789/17686/browse?type=author&order=ASC&rpp=20&value=Das%2C+Joydip>

[17]

List of widget toolkits –Wikipedia, the free encyclopedia
<http://en.wikipedia.org/wiki/List_of_widget_toolkits>

[18]

Laying out your UI (XAML) (Windows) <http://msdn.microsoft.com/en-us/library/windows/apps/hh465330.aspx>

[19]

Extensible Application Markup Language – Wikipedia, the free encyclopedia
<http://en.wikipedia.org/wiki/Extensible_Application_Markup_Language>

[20]

iOS Developer Library – Auto Layout Guide
<https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/AutolayoutPG/Introduction/Introduction.html>

[21]

Overview of Motif 2.0
<http://www.opengroup.org/tech/desktop/motif/xjournal.htm#RTFToC9>

[22]

Java view technologies and frameworks – Wikipedia, the free encyclopedia <http://en.wikipedia.org/wiki/Java_view_technologies_and_frameworks>

[23]

jQuery UI <http://jqueryui.com/>

[24]

Layouts – Architect 2 – Sencha Docs <http://docs-origin.sencha.com/architect/2/#!/guide/views_layouts>

[25]

Silverlight Layout System <http://msdn.microsoft.com/en-us/library/cc645025%28v=vs.95%29.aspx>

[26]

Frameworks, Methodologies, and Tools for Developing Rich Internet Applications – Google Books <https://books.google.co.in/books?id=nx2XBQAAQBAJ&pg=PR14&lpg=PR14&dq=GUI+Design+Frameworks+Classification&source=bl&ots=QUpWkN-AA_&sig=i22-6rpZf0i3q2IsreKfidh6Q-4&hl=en&sa=X&ei=X4kVVYuwE87n8AXr3AI&ved=0CDEQ6AEwAw#v=onepage&q=GUI%20Design%20Frameworks%20Classification&f=true>

[27]

The Box Model – Mozilla|MDN <https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL/Tutorial/The_Box_Model>

[28]

DockPanel Class (System.Windows.Controls) <http://msdn2.microsoft.com/en-us/library/system.windows.controls.dockpanel.aspx>

[29]

MiG Layout Java Layout Manager for Swing and SWT <http://www.miglayout.com/>

[30]

Declarative programming – Wikipedia, the free encyclopedia
<http://en.wikipedia.org/wiki/Declarative_programming>

[31]

ALM – The Auckland Layout Model *<http://aucklandlayout.sourceforge.net/>*

[32]

Speeding up SOR Solvers for Constraint-based GUIs with a Warm-Start Strategy - Noreen Jamil, Johannes Muller, Christof Lutteroth and Gerald Weber
<http://arxiv.org/pdf/1401.1752.pdf>

[33]

Designing A Winning Navigation Menu: Ideas and Inspirations – Hongkiat
<http://www.hongkiat.com/blog/navigation-design-ideas-inspiration/>

User Interface UI Kit | Gumby Framework *<http://gumbyframework.com/docs/ui-kit/#!/navigation>*

[34]

A Flexible Approach to Responsive Navigation – Tuts+ Web Design Tutorial
<http://webdesign.tutsplus.com/tutorials/a-flexible-approach-to-responsive-navigation--webdesign-8397>

[35]

Responsive Navigation Patterns | Brad Frost *<http://bradfrost.com/blog/web/responsive-nav-patterns/>*

[36]

The Pearsonified Skin Will Rock Your Web Design World
<http://diythemes.com/thesis/pearsonified-skin/>

[37]

Skin design | Website design | ecommerce Los Angeles *<http://www.myecommerce.biz/Skin-Design.html>*

[38]

Standard Website Responsive Skin Designs Burleigh Heads, Gold Coast *<https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0CDYQFjAE&url=http%3A%2F%2Fwww.bluerockwebsites.com%2Fstandard-website-skin-designs&ei=v3AZVfD0K4jHuATVwIDYDg&usg=AFQjCNG1FKBQSkG3arf0pOLgBoq5ELzGmg&sig2=fJ_g86ccJt_3a-DGbVwIUQ>*

[39]

CSS Architectures: Scalable and Modular Approaches *<http://www.sitepoint.com/css-architectures-scalable-and-modular-approaches/>*

4 ways to create CSS that's modular and scalable *<http://www.creativebloq.com/css3/create-modular-and-scalable-css-9134351>*

[40]

LearnITy GUI Design Framework Artifacts *<https://app.box.com/s/3yc8q8qose91h6ch5fuocx0wdcp54gic>*

[41]

Designing With Grid-Based Approach – Smashing Magazine *<http://www.smashingmagazine.com/2007/04/14/designing-with-grid-based-approach/>*

Page Layout Design *<http://www.serif.com/appresources/ppx5/tutorials/en-gb/tutorials/design_grid.htm>*

[42]

Free CSS Layouts And Templates – Smashing Magazine *< http://www.smashingmagazine.com/2007/01/12/free-css-layouts-and-templates/>*

Layout Gala – a collection of 40 CSS layouts based on the same markup and ready for download! *<http://blog.html.it/layoutgala/>*

Appendix 1: Live Implementation of the LearnITy Framework

This section contains the full configuration files & interface code of sample implementation using LearnITy Framework. The code can be reused by the developers of this framework for quick learning of the framework usage. As an example sample banking screen and efiling home pages were implemented using enhanced design of the framework. The screenshots are also attached for reference. Apart from that three XSD files are also provided. These are used for implementing XML validation feature for manifest.xml, interfacerole.xml and interface.xml files before uploading them through Core Admin upload utility of the framework.

Manifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest id="MyBank Interface" title="MyBank Interface Collection Framework">

<interface title="LoginPage" id="LoginPage" zip="LoginPage.zip" type="Interface"/>

<interface title="NewDesignPageCreation" id="NewDesignPageCreation"
zip="NewDesignPageCreation.zip" type="Interface"/>

<interface title="AccountSummaryCenterPanel" id="AccountSummaryCenterPanel"
zip="AccountSummaryCenterPanel.zip" type="Interface"/>
</manifest>
```

Interfacerole.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<roles>
<role id="DEFAULT">

<interface id="LoginPage">
<layout id="Loginlayout"/>
<content id="Logincontent"/>
<style id="Loginstyle"/>
<behaviour id="Loginbehaviour"/>
</interface>

<interface id="AccountSummaryCenterPanel">
<layout id="AccountSummaryCenterPanelLayout"/>
<content id="AccountSummaryCenterPanelContent"/>
<style id="AccountSummaryCenterPanelStyle"/>
<behaviour id="AccountSummaryCenterPanelBehaviour"/>
</interface>

</role>

<role id="Common_User">
```

```
<interface id="NewDesignPageCreation">
<layout id="NewDesignPageCreationLayout"/>
<content id="NewDesignPageCreationContent"/>
<style id="NewDesignPageCreationStyle"/>
<behaviour id="NewDesignPageCreationBehaviour"/>
</interface>

<interface id="AccountSummaryCenterPanel">
<layout id="AccountSummaryCenterPanelLayout"/>
<content id="AccountSummaryCenterPanelContent"/>
<style id="AccountSummaryCenterPanelStyle"/>
<behaviour id="AccountSummaryCenterPanelBehaviour"/>
</interface>

</role>

</roles>
```

Interface.xml for NewDesignPageCreation

```
<interface id="NewDesignPageCreation" title="MyBankLeftTopBars" type="Interface">
<structure>
<part id="main" class="verticalContainer"/>
<part id="topPanel" class="horizontalContainer" />

<part id="logo" class="image"/>
<part id="rightPanel" class="verticalContainer"/>

<part id="rightTop" class="horizontalContainer"/>
<part id="rightMiddle" class="horizontalContainer" spacing="10px"/>
<part id="rightBottom" class="horizontalContainer" />

<part id="languageLogo" class="image"/>
<part id="sizeLogo" class="image"/>
<part id="skipToMainLink" class="textlink"/>
<part id="aboutUs" class="button"/>
<part id="feedback" class="button"/>
<part id="contactUs" class="button"/>
<part id="help" class="button"/>

<part id="space" class="label"/>
<part id="searchBox" class="inputtext"/>
<part id="searchButton" class="button"/>

<part id="panel2left" class="verticalContainer"/>
<part id="panel2middle" class="verticalContainer"/>
<part id="panel2right" class="verticalContainer"/>
```

```
<part id="iAm" class="button"/>
<part id="leftPanel1" class="button"/>
<part id="leftPanel2" class="button"/>
<part id="leftPanel3" class="button"/>
<part id="leftPanel4" class="button"/>
<part id="leftPanel5" class="button"/>

<part id="middleTop" class="verticalContainer"/>
<part id="middleMiddle" class="horizontalContainer"/>
<part id="middleBottom" class="horizontalContainer"/>
<part id="middleBottomButton" class="button"/>

<part id="middleTop1" class="horizontalContainer" />
<part id="middleTop2" class="horizontalContainer" spacing="5px"/>

<part id="eFileYourReturn" class="button"/>
<part id="eFileButton" class="button"/>
<part id="smallText" class="button"/>
<part id="learn" class="button"/>

<part id="image1" class="image"/>
<part id="image2" class="image"/>
<part id="image3" class="image"/>

<part id="newText" class="button"/>
<part id="newButton" class="button"/>
<part id="line1" class="image"/>
<part id="regText" class="button"/>
<part id="regButton" class="button"/>
<part id="line2" class="image"/>
<part id="custText" class="button"/>
<part id="custButton" class="button"/>
<part id="line3" class="image"/>

<part id="splitLine1" class="horizontalContainer"/>

<part id="services" class="verticalContainer"/>

<part id="servicesHead" class="button"/>
<part id="servicesLine" class="horizontalContainer"/>
<part id="services1" class="button"/>
<part id="services2" class="button"/>
<part id="services3" class="button"/>
<part id="services4" class="button"/>
<part id="services5" class="button"/>
<part id="services6" class="button"/>
<part id="services7" class="button"/>
```

```
<part id="services8" class="button"/>
<part id="services9" class="button"/>
<part id="services10" class="button"/>

<part id="news" class="verticalContainer"/>
<part id="newsHead" class="button"/>
<part id="newsLine" class="horizontalContainer"/>
<part id="news1" class="button"/>
<part id="news2" class="button"/>
<part id="news3" class="button"/>
<part id="news4" class="button"/>
<part id="dashedLine" class="image"/>

<part id="download" class="verticalContainer"/>
<part id="downloadHead" class="button"/>
<part id="downloadLine" class="horizontalContainer"/>
<part id="download1" class="button"/>
<part id="download2" class="horizontalContainer"/>
<part id="download3" class="horizontalContainer"/>
<part id="download4" class="horizontalContainer"/>
<part id="download5" class="horizontalContainer"/>
<part id="download6" class="button"/>
<part id="download7" class="button"/>
<part id="download8" class="button"/>
<part id="download9" class="button"/>
<part id="download10" class="button"/>
<part id="download11" class="button"/>

<part id="sahaj" class="button"/>
<part id="sugam" class="button"/>
<part id="itr2" class="button"/>
<part id="itr3" class="button"/>
<part id="itr4" class="button"/>
<part id="itr5" class="button"/>
<part id="itr6" class="button"/>
<part id="itr7" class="button"/>
</structure>
<layout id="NewDesignPageCreationLayout">
<part id="main" height="100%" width="100%" position="border"
borderPosition="NORTH"/>
<part id="topPanel" height="80px" width="1000px" left="" top="" position="border"
borderPosition="NORTH-CENTER" parent_id="main"/>

<part id="logo" height="80px" width="500px" parent_id="topPanel"/>
<part id="rightPanel" height="100px" width="500px" parent_id="topPanel"/>

<part id="rightTop" height="15%" width="100%" parent_id="rightPanel" />
<part id="rightMiddle" height="15%" width="100%" parent_id="rightPanel" />
```

```
<part id="rightBottom" height="30%" width="100%" parent_id="rightPanel" />

<part id="languageLogo" height="" width="" parent_id="rightTop" />
<part id="sizeLogo" height="" width="" parent_id="rightTop" />
<part id="skipToMainLink" height="30%" width="40%" parent_id="rightTop" />
<part id="aboutUs" height="60%" width="" parent_id="rightTop" />
<part id="feedback" height="60%" width="" parent_id="rightTop" />
<part id="contactUs" height="60%" width="" parent_id="rightTop" />
<part id="help" height="60%" width="" parent_id="rightTop" />

<part id="space" height="60%" width="40%" parent_id="rightMiddle" />
<part id="searchBox" height="60%" width="30%" parent_id="rightMiddle" />
<part id="searchButton" height="60%" width="30%" parent_id="rightMiddle" />

<part id="panel2left" height="250px" width="250px" position="anchor"
neighbour_id="logo" neighbourPosition="top" indentation="\\n" parent_id="main" />
<part id="panel2middle" height="250px" width="453px" position="anchor"
neighbour_id="panel2left" neighbourPosition="left" indentation="\\t" parent_id="main"
/>
<part id="panel2right" height="250px" width="250px" position="anchor"
neighbour_id="panel2middle" neighbourPosition="left" indentation="\\t"
parent_id="main" />

<part id="iAm" height="" width="" parent_id="panel2left"/>
<part id="leftPanel1" height="" width="" parent_id="panel2left"/>
<part id="leftPanel2" height="" width="" parent_id="panel2left"/>
<part id="leftPanel3" height="" width="" parent_id="panel2left"/>
<part id="leftPanel4" height="" width="" parent_id="panel2left"/>
<part id="leftPanel5" height="" width="" parent_id="panel2left"/>

<part id="middleTop" height="115px" width="453px" parent_id="panel2middle"/>
<part id="middleMiddle" height="115px" width="453px" parent_id="panel2middle"/>
<part id="middleBottom" height="15px" width="453px" parent_id="panel2middle"/>
<part id="middleBottomButton" parent_id="middleBottom"/>

<part id="middleTop1" parent_id="middleTop"/>
<part id="middleTop2" parent_id="middleTop"/>

<part id="eFileYourReturn" parent_id="middleTop1"/>
<part id="eFileButton" parent_id="middleTop1"/>
<part id="smallText" parent_id="middleTop2"/>
<part id="learn" parent_id="middleTop2"/>

<part id="image1" parent_id="middleMiddle"/>
<part id="image2" parent_id="middleMiddle"/>
<part id="image3" parent_id="middleMiddle"/>

<part id="newText" parent_id="panel2right"/>
```

```
<part id="newButton" parent_id="panel2right"/>
<part id="line1" parent_id="panel2right"/>
<part id="regText" parent_id="panel2right"/>
<part id="regButton" parent_id="panel2right"/>
<part id="line2" parent_id="panel2right"/>
<part id="custText" parent_id="panel2right"/>
<part id="custButton" parent_id="panel2right"/>
<part id="line3" parent_id="panel2right"/>

<part id="splitLine1" height="5px" width="1000px" position="anchor"
neighbour_id="panel2left" neighbourPosition="top" indentation="" parent_id="main" />

<part id="services" height="500px" width="250px" position="anchor"
neighbour_id="splitLine1" neighbourPosition="top" indentation="\\n" parent_id="main"
/>

<part id="servicesHead" parent_id="services" />
<part id="servicesLine" height="5px" width="250px" parent_id="services" />
<part id="services1" parent_id="services" />
<part id="services2" parent_id="services" />
<part id="services3" parent_id="services" />
<part id="services4" parent_id="services" />
<part id="services5" parent_id="services" />
<part id="services6" parent_id="services" />
<part id="services7" parent_id="services" />
<part id="services8" parent_id="services" />
<part id="services9" parent_id="services" />
<part id="services10" parent_id="services" />

<part id="news" height="500px" width="453px" position="anchor"
neighbour_id="services" neighbourPosition="left" indentation="\\t" parent_id="main" />

<part id="newsHead" parent_id="news" />
<part id="newsLine" height="5px" width="453px" parent_id="news" />
<part id="news1" parent_id="news" />
<part id="dashedLine" parent_id="news" />
<part id="news2" parent_id="news" />
<part id="dashedLine" parent_id="news" />
<part id="news3" parent_id="news" />
<part id="dashedLine" parent_id="news" />
<part id="news4" parent_id="news" />
<part id="dashedLine" parent_id="news" />

<part id="download" height="500px" width="250px" position="anchor"
neighbour_id="news" neighbourPosition="left" indentation="\\t" parent_id="main" />
<part id="downloadHead" parent_id="download" />
<part id="downloadLine" height="5px" width="250px" parent_id="download" />
<part id="download1" parent_id="download" />
```

```
<part id="download2" parent_id="download" />
<part id="download3" parent_id="download" />
<part id="download4" parent_id="download" />
<part id="download5" parent_id="download" />
<part id="download6" parent_id="download" />
<part id="download7" parent_id="download" />
<part id="download8" parent_id="download" />
<part id="download9" parent_id="download" />
<part id="download10" parent_id="download" />
<part id="download11" parent_id="download" />

<part id="sahaj" parent_id="download2" />
<part id="sugam" parent_id="download2" />
<part id="itr2" parent_id="download3" />
<part id="itr3" parent_id="download3" />
<part id="itr4" parent_id="download4" />
<part id="itr5" parent_id="download4" />
<part id="itr6" parent_id="download5" />
<part id="itr7" parent_id="download5" />

</layout>
<content id="NewDesignPageCreationContent">
<part id="main"  value="nadmin" valuetype="reference"/>
<part id="logo"  value="logo" valuetype="reference"/>
<part id="languageLogo"  value="languageLogo" valuetype="reference"/>
<part id="sizeLogo"  value="sizeLogo" valuetype="reference"/>
<part id="skipToMainLink"  value="Skip to Main Link" valuetype="inline"/>                -
<part id="aboutUs"  value="About Us" valuetype="inline"/>
<part id="feedback"  value="Feedback" valuetype="inline"/>
<part id="contactUs"  value="Contact Us" valuetype="inline"/>
<part id="help"  value="Help" valuetype="inline"/>

<part id="searchButton"  value="Search" valuetype="inline"/>

<part id="iAm" value="I Am..." valuetype="inline"/>
<part id="leftPanel1" value="TAXPAYER" valuetype="inline"/>
<part id="leftPanel2" value="PROFESSIONLS FOR TAX AUDIT" valuetype="inline"/>
<part id="leftPanel3" value="E RETURN INTERMEDIARY" valuetype="inline"/>
<part id="leftPanel4" value="BULK PAN VERIFICATION USER" valuetype="inline"/>
<part id="leftPanel5" value="TAX DEDUCTOR AND COLLECTOR" valuetype="inline"/>

<part id="middleBottomButton" value="There are Mobile Applications which are not
supported by Income Tax Department..........." valuetype="inline"/>

<part id="eFileYourReturn" value="e-File Your Tax Return" valuetype="inline"/>
<part id="eFileButton" value="e-File>>" valuetype="inline"/>
<part id="smallText" value="Its Fast Easy and Secure..." valuetype="inline"/>
<part id="learn" value="Learn How To e-File" valuetype="inline"/>
```

```
<part id="image1"  value="image1" valuetype="reference"/>
<part id="image2"  value="image2" valuetype="reference"/>
<part id="image3"  value="image3" valuetype="reference"/>

<part id="newText" value="New to eFiling?" valuetype="inline"/>
<part id="newButton" value="Register Yourself" valuetype="inline"/>
<part id="line1" value="line1" valuetype="reference"/>
<part id="regText" value="Registered User?" valuetype="inline"/>
<part id="regButton" value="Login Here" valuetype="inline"/>
<part id="line2" value="line2" valuetype="reference"/>
<part id="custText" value="Need Assistance?" valuetype="inline"/>
<part id="custButton" value="Customer Care" valuetype="inline"/>
<part id="line3" value="line3" valuetype="reference"/>

<part id="servicesHead" value="Services" valuetype="inline"/>
<part id="services1" value="Submit Returns/Forms" valuetype="inline"/>
<part id="services2" value="View Form 26AS(Tax Credit)" valuetype="inline"/>
<part id="services3" value="Outstanding Tax Demand" valuetype="inline"/>
<part id="services4" value="CPC Refund Status" valuetype="inline"/>
<part id="services5" value="Rectification Status" valuetype="inline"/>
<part id="services6" value="ITR-V Receipt Status" valuetype="inline"/>
<part id="services7" value="Know Your Judisdictional A.O." valuetype="inline"/>
<part id="services8" value="Know Your PAN" valuetype="inline"/>
<part id="services9" value="Know Your TAN" valuetype="inline"/>
<part id="services10" value="Apply Online(PAN/TAN)" valuetype="inline"/>

<part id="newsHead" value="News and Updates" valuetype="inline"/>
<part id="news1" value="29/7/2014\nPlease check for email from Department for PIN in
your Inbox as well as the Spam or Junk folder(mail
id:DONOTREPLY@incometaxindiaefiling.gov.in)" valuetype="inline"/>
<part id="news2" value="29/7/2014\nPlease check for email from Department for PIN in
your Inbox as well as the Spam or Junk folder(mail
id:DONOTREPLY@incometaxindiaefiling.gov.in)" valuetype="inline"/>
<part id="news3" value="29/7/2014\nPlease check for email from Department for PIN in
your Inbox as well as the Spam or Junk folder(mail
id:DONOTREPLY@incometaxindiaefiling.gov.in)" valuetype="inline"/>
<part id="dashedLine"  value="dashedLine" valuetype="reference"/>
<part id="news4" value="29/7/2014\nPlease check for email from Department for PIN in
your Inbox as well as the Spam or Junk folder(mail
id:DONOTREPLY@incometaxindiaefiling.gov.in)" valuetype="inline"/>
<part id="dashedLine"  value="dashedLine" valuetype="reference"/>

<part id="downloadHead" value="Downloads" valuetype="inline"/>
<part id="download1" value="AY 2014-15" valuetype="inline"/>
<part id="sahaj" value="ITR-1(SAHAJ)" valuetype="inline"/>
<part id="sugam" value="ITR-4S(SUGAM)" valuetype="inline"/>
<part id="itr2" value="ITR-2" valuetype="inline"/>
```

```
<part id="itr3" value="ITR-3" valuetype="inline"/>
<part id="itr4" value="ITR-4" valuetype="inline"/>
<part id="itr5" value="ITR-5" valuetype="inline"/>
<part id="itr6" value="ITR-6" valuetype="inline"/>
<part id="itr7" value="ITR-7" valuetype="inline" />
<part id="download6" value="Forms (Other than ITR)" />
<part id="download7" value="Previous Year ITRs" />
<part id="download8" value="Form BB(Return of Net Wealth)" />
<part id="download9" value="Schema Downloads" />
<part id="download10" value="Quick e-File ITR-1 and ITR-4S Online" />
<part id="download11" value="e-Filing Statistics" />

</content>
<style id="NewDesignPageCreationStyle">
<part id="topPanel" value="[CDATA[border: 1px solid black;;background-
color:#FFEEFF;]]" valuetype="inline"/>
<part id="logo" value="[CDATA[border: 2px solid black;;background-color:#FFEEFF;]]"
valuetype="inline"/>
<part id="languageLogo" value="[CDATA[border: 2px solid black;;background-
color:#FFEEFF;]]" valuetype="inline"/>
<part id="sizeLogo" value="[CDATA[border: 2px solid black;;background-color:#FFEEFF;]]"
valuetype="inline"/>
<part id="skipToMainLink" value="[CDATA[color:#000000;]]" valuetype="inline"/>
<part id="aboutUs" value="[border: 2px solid black;;background-color:#0000A0;font-
weight:bold;color:#FFFFFF;]]" valuetype="inline"/>
<part id="feedback" value="[border: 2px solid black;;background-color:#0000A0;font-
weight:bold;color:#FFFFFF;]]" valuetype="inline"/>
<part id="contactUs" value="[border: 2px solid black;;background-color:#0000A0;font-
weight:bold;color:#FFFFFF;]]" valuetype="inline"/>
<part id="help" value="[border: 2px solid black;;background-color:#0000A0;font-
weight:bold;color:#FFFFFF;]]" valuetype="inline"/>

<part id="searchButton" value="[border: 2px solid black;;background-color:#800000;font-
weight:bold;color:#FFFFFF;]]" valuetype="inline"/>

<part id="panel2left" value="[CDATA[border: none;;background-color:#0000A0;]]"
valuetype="inline"/>
<part id="panel2middle" value="[CDATA[border: none;;background-color:#008000;]]"
valuetype="inline"/>
<part id="panel2right" value="[CDATA[border: none;;background-color:#0000A0;]]"
valuetype="inline"/>

<part id="iAm" value="[CDATA[border: none;;background-
color:#0000A0;color:#FFFFFF;font-size:20px;]]" valuetype="inline"/>
<part id="leftPanel1" value="[CDATA[border: none;;background-
color:#0000A0;color:#FFFFFF;font-size:15px;]]" valuetype="inline"/>
<part id="leftPanel2" value="[CDATA[border: none;;background-
color:#0000A0;color:#FFFFFF;font-size:15px;]]" valuetype="inline"/>
```

```
<part id="leftPanel3" value="[CDATA[border: none;;background-color:#0000A0;color:#FFFFFF;font-size:15px;]]" valuetype="inline"/>
<part id="leftPanel4" value="[CDATA[border: none;;background-color:#0000A0;color:#FFFFFF;font-size:15px;]]" valuetype="inline"/>
<part id="leftPanel5" value="[CDATA[border: none;;background-color:#0000A0;color:#FFFFFF;font-size:15px;width:200px;white-space:normal;text-align:left]]" valuetype="inline"/>

<part id="middleTop" value="[CDATA[border: none;;background-color:#008000;]]" valuetype="inline"/>
<part id="middleMiddle" value="[CDATA[border: none;;background-color:#FFFFFF;]]" valuetype="inline"/>
<part id="middleBottom" value="[CDATA[border: none;;background-color:#98AFC7;]]" valuetype="inline"/>
<part id="middleBottomButton" value="[CDATA[border: 1px solid black;;background-color:#98AFC7;width:453px;]]" valuetype="inline"/>

<part id="eFileYourReturn"  value="[CDATA[border: none;;background-color:#008000;color:#FFFFFF;font-size:30px;]]" valuetype="inline"/>
<part id="eFileButton"  value="[CDATA[border: 1px solid black;;background-color:#FFF380;color:#000000;]]" valuetype="inline"/>
<part id="smallText"  value="[CDATA[border: none;;background-color:#008000;color:#FFFFFF;font-size:18px;]]" valuetype="inline"/>
<part id="learn"  value="[CDATA[border: 1px solid black;;background-color:#B6B6B4;color:#000000;]]" valuetype="inline"/>

<part id="image1" value="[CDATA[border: 2px solid black;;background-color:#98AFC7;]]" valuetype="inline"/>
<part id="image2" value="[CDATA[border: 2px solid black;;background-color:#98AFC7;]]" valuetype="inline"/>
<part id="image3" value="[CDATA[border: 2px solid black;;background-color:#98AFC7;]]" valuetype="inline"/>

<part id="newText" value="[CDATA[border: none;;background-color:#0000A0;color:#FFFFFF;font-size:20px;]]" valuetype="inline"/>
<part id="newButton" value="[CDATA[border: 1px solid black;;background-color:#008000;color:#000000;]]" valuetype="inline"/>

<part id="regText" value="[CDATA[border: none;;background-color:#0000A0;color:#FFFFFF;font-size:20px;]]" valuetype="inline"/>
<part id="regButton" value="[CDATA[border: 1px solid black;;background-color:#FFF380;color:#000000;]]" valuetype="inline"/>

<part id="custText" value="[CDATA[border: none;;background-color:#0000A0;color:#FFFFFF;font-size:20px;]]" valuetype="inline"/>
<part id="custButton" value="[CDATA[border: 1px solid black;;background-color:#FFF380;color:#000000;]]" valuetype="inline"/>
```

```
<part id="splitLine1" value="[CDATA[border: 1px solid black;;background-color:#000000;]]" valuetype="inline"/>

<part id="services" value="[CDATA[border: none;;background-color:#FFFFFF;]]" valuetype="inline"/>

<part id="servicesHead" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:30px;]]" valuetype="inline"/>
<part id="servicesLine" value="[CDATA[border: none;;background-color:#000000;]]" valuetype="inline"/>
<part id="services1" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="services2" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="services3" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="services4" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="services5" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="services6" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="services7" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="services8" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="services9" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="services10" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>

<part id="newsHead" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:30px;]]" valuetype="inline"/>
<part id="newsLine" value="[CDATA[border: none;;background-color:#000000;]]" valuetype="inline"/>
<part id="news1" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;width:450px;white-space:normal;text-align:left;]]" valuetype="inline"/>
<part id="news2" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;width:450px;white-space:normal;text-align:left;]]" valuetype="inline"/>
<part id="news3" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;width:450px;white-space:normal;text-align:left;]]" valuetype="inline"/>
<part id="news4" value="[CDATA[border: none;;background-color:#FFFFFF;color:#000000;font-size:15px;width:450px;white-space:normal;text-align:left;]]" valuetype="inline"/>
```

```
<part id="downloadHead" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:30px;]]" valuetype="inline"/>
<part id="downloadLine" value="[CDATA[border: none;;background-color:#000000;]]"
valuetype="inline"/>
<part id="download1" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="sahaj" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="sugam" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="itr2" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="itr3" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="itr4" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="itr5" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="itr6" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="itr7" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="download6" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="download7" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="download8" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="download9" value="[CDATA[border: none;;background-
color:#FFFFFF;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="download10" value="[CDATA[border: 1px solid green;;background-
color:#FFEEEE;color:#000000;font-size:15px;]]" valuetype="inline"/>
<part id="download11" value="[CDATA[border: 1px solid green;;background-
color:#FFEEEE;color:#000000;font-size:15px;]]" valuetype="inline"/>

</style>
<behaviour id="NewDesignPageCreationBehaviour">

</behaviour>
<resource>
<resourceitem id="logo" href="logo.jpg" valuetype="image" />
<resourceitem id="languageLogo" href="languageLogo.jpg" valuetype="image" />
<resourceitem id="sizeLogo" href="sizeLogo.jpg" valuetype="image" />
<resourceitem id="image1" href="image1.jpg" valuetype="image" />
<resourceitem id="image2" href="image2.jpg" valuetype="image" />
<resourceitem id="image3" href="image3.jpg" valuetype="image" />
<resourceitem id="line1" href="line.jpg" valuetype="image" />
<resourceitem id="line2" href="line.jpg" valuetype="image" />
```

```
<resourceitem id="line3" href="line.jpg" valuetype="image" />
<resourceitem id="dashedLine" href="dashedLine.jpg" valuetype="image" />
</resource>
</interface>
```

Interface.xml for AccountSummaryCenterPanel

```
<interface id="AccountSummaryCenterPanel" title="AccountSummaryCenterPanel"
type="Interface">
<structure>
<part id="main" class="verticalContainer"/>
<part id="topLinkSection" class="horizontalContainer" />
<part id="savings" class="textlink"/>
<part id="gap" class="label"/>
<part id="current" class="textlink"/>

<part id="noteHead" class="label" />
<part id="note1" class="label" />
<part id="note2" class="label" />
<part id="note3" class="label" />
<part id="note4" class="label" />

<part id="savingsHead" class="label" />
<part id="savingsTableRow1" class="horizontalContainer" />
<part id="savingsTableRow2" class="horizontalContainer" />

<part id="acctNoHead" class="button" />
<part id="acctNoVal" class="button" />
<part id="branchHead" class="button" />
<part id="branchVal" class="button" />
<part id="nameHead" class="button" />
<part id="nameVal" class="button" />
<part id="ccyHead" class="button" />
<part id="ccyVal" class="button" />
<part id="balHead" class="button" />
<part id="balVal" class="button" />
<part id="miniHead" class="button" />
<part id="miniVal" class="button" />

<part id="currentHead" class="label" />
<part id="currentTableRow1" class="horizontalContainer" />
<part id="currentTableRow2" class="horizontalContainer" />

<part id="acctNoHeadCurr" class="button" />
<part id="acctNoValCurr" class="button" />
<part id="branchHeadCurr" class="button" />
<part id="branchValCurr" class="button" />
```

```
<part id="nameHeadCurr" class="button" />
<part id="nameValCurr" class="button" />
<part id="ccyHeadCurr" class="button" />
<part id="ccyValCurr" class="button" />
<part id="balHeadCurr" class="button" />
<part id="balValCurr" class="button" />
<part id="miniHeadCurr" class="button" />
<part id="miniValCurr" class="button" />

<!-- Page Navigation test - Start-->
<part id="navigationPanel" class="horizontalContainer" />
<part id="pageNavTestText" class="inputtext" />
<part id="pageNavTestButton" class="button" />
<!-- Page Navigation test - End-->
</structure>
<layout id="AccountSummaryCenterPanelLayout">
<part id="main" height="100%" width="100%" left="0px" top="0px" position="absolute"/>
<part id="topLinkSection" parent_id="main" />
<part id="savings" parent_id="topLinkSection"/>
<part id="gap" width="100px" parent_id="topLinkSection" />
<part id="current" parent_id="topLinkSection"/>

<part id="noteHead" parent_id="main"/>
<part id="note1" parent_id="main"/>
<part id="note2" parent_id="main"/>
<part id="note3" parent_id="main"/>
<part id="note4" parent_id="main"/>

<!-- Page Navigation test - Start-->
<part id="navigationPanel" parent_id="main"/>
<part id="pageNavTestText" parent_id="navigationPanel"/>
<part id="pageNavTestButton"  parent_id="navigationPanel"/>
<!-- Page Navigation test - End-->

<part id="savingsHead" parent_id="main"/>
<part id="savingsTableRow1" parent_id="main"/>
<part id="savingsTableRow2" parent_id="main"/>
<part id="acctNoHead" parent_id="savingsTableRow1"/>
<part id="branchHead" parent_id="savingsTableRow1"/>
<part id="nameHead" parent_id="savingsTableRow1"/>
<part id="ccyHead" parent_id="savingsTableRow1"/>
<part id="balHead" parent_id="savingsTableRow1"/>
<part id="miniHead" parent_id="savingsTableRow1"/>

<part id="acctNoVal" parent_id="savingsTableRow2"/>
<part id="branchVal" parent_id="savingsTableRow2"/>
<part id="nameVal" parent_id="savingsTableRow2"/>
<part id="ccyVal" parent_id="savingsTableRow2"/>
```

```
<part id="balVal" parent_id="savingsTableRow2"/>
<part id="miniVal" parent_id="savingsTableRow2"/>

<part id="currentHead" parent_id="main"/>
<part id="currentTableRow1" parent_id="main"/>
<part id="currentTableRow2" parent_id="main"/>

<part id="acctNoHeadCurr" parent_id="currentTableRow1"/>
<part id="acctNoValCurr" parent_id="currentTableRow2"/>
<part id="branchHeadCurr" parent_id="currentTableRow1"/>
<part id="branchValCurr" parent_id="currentTableRow2"/>
<part id="nameHeadCurr" parent_id="currentTableRow1"/>
<part id="nameValCurr"  parent_id="currentTableRow2"/>
<part id="ccyHeadCurr" parent_id="currentTableRow1"/>
<part id="ccyValCurr" parent_id="currentTableRow2"/>
<part id="balHeadCurr" parent_id="currentTableRow1"/>
<part id="balValCurr" parent_id="currentTableRow2"/>
<part id="miniHeadCurr" parent_id="currentTableRow1"/>
<part id="miniValCurr" parent_id="currentTableRow2"/>

</layout>
<content id="AccountSummaryCenterPanelContent">
<part id="main"  value="nadmin" valuetype="reference"/>
<part id="savings"  value="Savings Account" valuetype="inline"/>
<part id="current"  value="Current Account" valuetype="inline"/>
<part id="noteHead"  value="Notes:" valuetype="inline"/>
<part id="note1"  value="1.The available balance displayed include credit balance and overdraft
limit. It excludes unclear fund and Amounts marked for Holds." valuetype="inline"/>
<part id="note2"  value="2.Savings account Customers can receive their statements monthly by
Email.To Register-Click Here." valuetype="inline"/>
<part id="note3"  value="3.Register for your Bills Online.Click here to know more."
valuetype="inline"/>
<part id="note4"  value="4.Please click here to check your PAN Card Details."
valuetype="inline"/>
<part id="savingsHead"  value="Savings Account" valuetype="inline"/>

<part id="savingsTableRow1"  value="" valuetype="inline"/>
<part id="savingsTableRow2"  value="" valuetype="inline"/>

<part id="acctNoHead" value="Account No" valuetype="inline"/>

<part id="branchHead" value="Branch" valuetype="inline"/>

<part id="nameHead" value="Name" valuetype="inline"/>

<part id="ccyHead" value="CCY" valuetype="inline"/>

<part id="balHead" value="Available Balance" valuetype="inline"/>
```

```
<part id="miniHead" value="Mini Statement" valuetype="inline"/>


<part id="acctNoVal" value="03641610010215" valuetype="inline"/>
<part id="branchVal" value="KUKATPALLY" valuetype="inline"/>
<part id="nameVal" value="SUVRA NANDI" valuetype="inline"/>
<part id="ccyVal" value="INR" valuetype="inline"/>
<part id="balVal" value="6000.00" valuetype="inline"/>
<part id="miniVal" value="View" valuetype="inline"/>
<part id="currentHead" value="Current Account" valuetype="inline"/>

<part id="acctNoHeadCurr" value="Account No" valuetype="inline"/>
<part id="acctNoValCurr" value="03642470008760" valuetype="inline"/>
<part id="branchHeadCurr" value="Branch" valuetype="inline"/>
<part id="branchValCurr" value="KUKATPALLY" valuetype="inline"/>
<part id="nameHeadCurr" value="Name" valuetype="inline"/>
<part id="nameValCurr" value="SUVRA NANDI" valuetype="inline"/>
<part id="ccyHeadCurr" value="CCY" valuetype="inline"/>
<part id="ccyValCurr" value="INR" valuetype="inline"/>
<part id="balHeadCurr" value="Available Balance" valuetype="inline"/>
<part id="balValCurr" value="4000.00" valuetype="inline"/>
<part id="miniHeadCurr" value="Mini Statement" valuetype="inline"/>
<part id="miniValCurr" value="View" valuetype="inline"/>

<!-- Page Navigation test - Start-->
<!--<part id="pageNavTestText" value="" valuetype="inline"/>-->
<part id="pageNavTestButton" value="Next" valuetype="inline"/>
<!-- Page Navigation test - End-->


</content>
<style id="AccountSummaryCenterPanelStyle">
<part id="main" value="[CDATA[border: 1px solid black;;background-color:#FFFFFF;]]"
valuetype="reference"/>
<part id="savings" value="[CDATA[border: none;;background-color:#FFEEFF;color:#000000;font-
weight:bold;text-align:right;]]" valuetype="inline"/>
<part id="current" value="[CDATA[border: none;;background-color:#FFEEFF;color:#000000;font-
weight:bold;text-align:right;]]" valuetype="inline"/>
<part id="noteHead" value="[CDATA[border: none;;background-
color:#FFEEFF;color:#000000;font-weight:bold;text-align:right;]]" valuetype="inline"/>
<part id="note1" value="[CDATA[border: none;;background-color:#FFEEFF;color:#000000;text-
align:center;]]" valuetype="inline"/>
<part id="note2" value="[CDATA[border: none;;background-color:#FFEEFF;color:#000000;]]"
valuetype="inline"/>
<part id="note3" value="[CDATA[border: none;;background-color:#FFEEFF;color:#000000;]]"
valuetype="inline"/>
<part id="note4" value="[CDATA[border: none;;background-color:#FFEEFF;color:#000000;]]"
```

```
valuetype="inline"/>
<part id="savingsHead" value="[CDATA[border: none;;background-
color:#FFEEFF;color:#000000;font-size:20px;font-weight:bold;]]" valuetype="inline"/>

<part id="acctNoHead" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>

<part id="branchHead" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>

<part id="nameHead" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>

<part id="ccyHead" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="balHead" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>

<part id="miniHead" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>

<part id="acctNoVal" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="branchVal" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="nameVal" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="ccyVal" value="[CDATA[height:20px;width:200px;border: 1px solid black;;background-
color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-align:center;]]"
valuetype="inline"/>
<part id="balVal" value="[CDATA[height:20px;width:200px;border: 1px solid black;;background-
color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-align:center;]]"
valuetype="inline"/>
<part id="miniVal" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>

<part id="currentHead" value="[CDATA[border: none;;background-
color:#FFEEFF;color:#000000;font-size:20px;font-weight:bold;]]" valuetype="inline"/>
```

```
<part id="acctNoHeadCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="acctNoValCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="branchHeadCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="branchValCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="nameHeadCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="nameValCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="ccyHeadCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="ccyValCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="balHeadCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="balValCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="miniHeadCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
<part id="miniValCurr" value="[CDATA[height:20px;width:200px;border: 1px solid
black;;background-color:#DDDDDD;color:#000000;font-size:12px;font-weight:bold;text-
align:center;]]" valuetype="inline"/>
</style>
<behaviour id="AccountSummaryCenterPanelBehaviour">
<!-- Page Navigation test - Start-->
<part id="root">
<event name="" type="simple" function="" callback="" javaclass="" target="" valuetype=""
resourceid="PageNavigationJs"></event>
</part>
<!--<part id="pageNavTestButton">
<event name="onclick" type="simple" function="alert('Hi!!')" callback="" javaclass="" target=""
valuetype="inline" resourceid=""></event>
</part>-->
```

```
<!-- Page Navigation test - End-->
</behaviour>
<resource>
<resourceitem id="PageNavigationJs" href="PageNavigation.js" valuetype="js"></resourceitem>
</resource>
</interface>
```

Below is the Bank Screen implemented using above configuration :

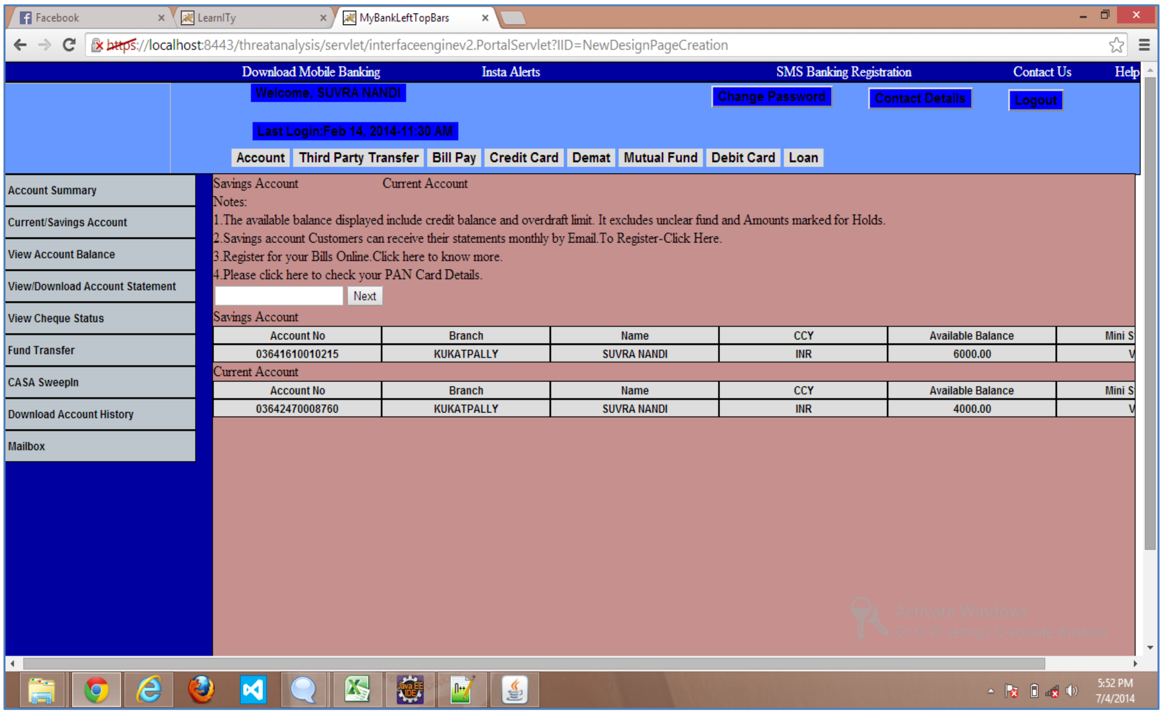


Figure 98: Bank screen implemented using enhanced layout design of LearnITy Framework

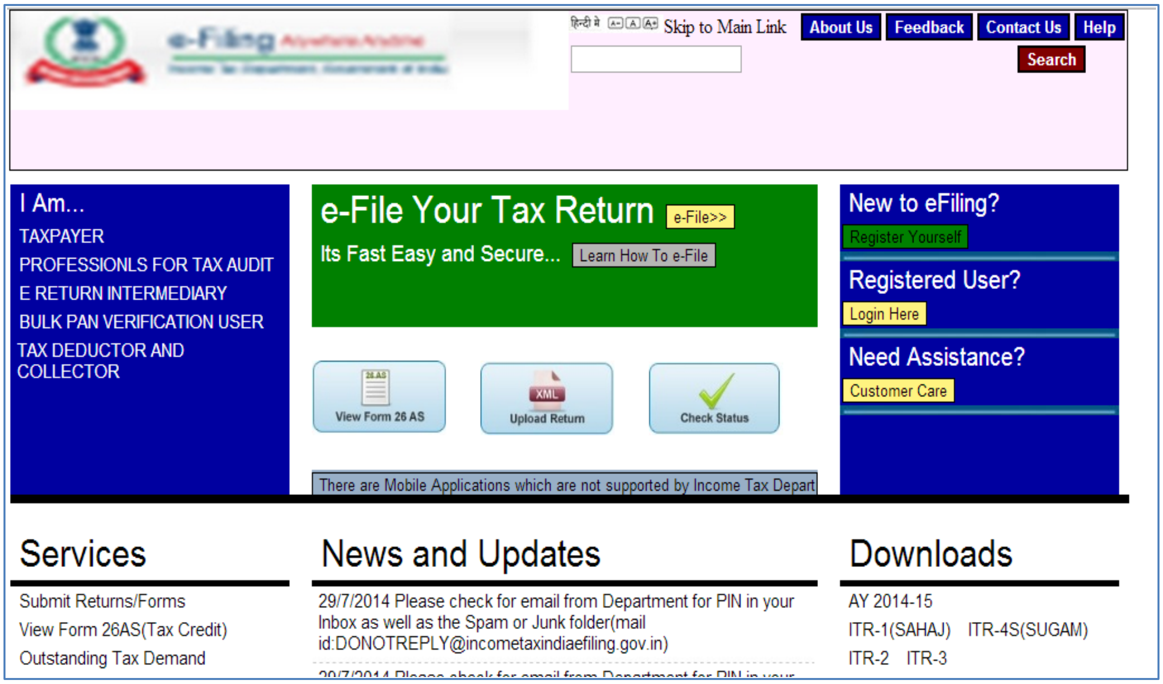


Figure 99: eFiling screen implemented using enhanced layout design of LearnITy Framework

Manifest.xsd structure

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified">
<xs:element name="manifest">
<xs:complexType>
<xs:sequence>
<xs:element name="interface" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:attribute type="xs:string" name="id" use="required"/>
<xs:attribute type="xs:string" name="title" use="required"/>
<xs:attribute type="xs:string" name="type" use="required"/>
<xs:attribute name="zip" use="required" >
<xs:simpleType >
<xs:restriction base="xs:string">
<xs:pattern value="([A-Za-z0-9])*(.zip)" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="id"/>
<xs:attribute type="xs:string" name="title"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

Interfacerole.xsd structure

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified">
<xs:element name="roles">
<xs:complexType>
<xs:sequence>
<xs:element name="role" maxOccurs="unbounded" minOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="interface" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="layout">
<xs:complexType>

<xs:attribute type="xs:string" name="id" use="required"/>

</xs:complexType>
</xs:element>
<xs:element name="content">
<xs:complexType>

<xs:attribute type="xs:string" name="id" use="required"/>

</xs:complexType>
</xs:element>
<xs:element name="style">
<xs:complexType>

<xs:attribute type="xs:string" name="id" use="required"/>

</xs:complexType>
</xs:element>
<xs:element name="behaviour">
<xs:complexType>

<xs:attribute type="xs:string" name="id" use="required"/>

</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="id1" use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="id" use="required">
<xs:simpleType>
```

```
<xs:restriction base="xs:string">
<xs:enumeration value="DEFAULT"/>
<xs:enumeration value="Common_User"/>
<xs:enumeration value="Superadmin"/>
<xs:enumeration value="Audi"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Interface.xsd structure

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified">
<xs:element name="interface">
<xs:complexType>
<xs:sequence>
<xs:element name="structure" maxOccurs="1" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="part" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="column" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="edit" maxOccurs="1" minOccurs="0">
<xs:complexType>
<!--<xs:sequence>
<xs:element name=" " maxOccurs="unbounded" minOccurs="0"/>
</xs:sequence>-->
<xs:attribute type="xs:string" name="type" use="optional"/>
<xs:attribute type="xs:string" name="size" use="optional"/>
<xs:attribute type="xs:string" name="rows" use="optional"/>
<xs:attribute type="xs:string" name="cols" use="optional"/>
<xs:attribute name="editdomaintype" use="optional">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="" />
<xs:enumeration value="query" />
<xs:enumeration value="fixed" />
</xs:restriction>
```

```
</xs:simpleType>
</xs:attribute>
<xs:attribute type="xs:string" name="value" use="optional"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="head" use="optional" />
<xs:attribute type="xs:string" name="name" use="optional" />
<xs:attribute type="xs:string" name="index" use="optional" />
<xs:attribute type="xs:string" name="width" use="optional" />
<xs:attribute type="xs:string" name="editable" use="optional" />
<xs:attribute type="xs:string" name="hidden" use="optional" />
<xs:attribute type="xs:string" name="edithidden" use="optional" />
<xs:attribute type="xs:string" name="key" use="optional" />
<xs:attribute name="required" use="optional" >
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="true" />
<xs:enumeration value="false" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="custom" use="optional" >
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="true" />
<xs:enumeration value="false" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute type="xs:string" name="custom_func" use="optional" />
<xs:attribute type="xs:string" name="minval" use="optional" />
<xs:attribute type="xs:string" name="maxval" use="optional" />
</xs:complexType>
</xs:element>
<xs:element name="loadquery" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:attribute type="xs:string" name="value" use="required" />
</xs:complexType>
</xs:element>
<xs:element name="delete" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="validations" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:sequence>
```

```
<xs:element name="validation" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="validationquery" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:attribute type="xs:string" name="id" use="optional" />
<xs:attribute type="xs:string" name="sql" use="optional" />
<xs:attribute type="xs:string" name="parameter" use="optional" />
<xs:attribute type="xs:string" name="message" use="optional" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="type" use="optional" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="queries" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="query" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:attribute type="xs:string" name="id" use="optional" />
<xs:attribute type="xs:string" name="sql" use="optional" />
<xs:attribute type="xs:string" name="parameter" use="optional" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="add" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="validations" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="validation" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="validationquery" maxOccurs="unbounded"
```

```
minOccurs="0">
<xs:complexType>
<xs:attribute type="xs:string" name="id" use="optional" />
<xs:attribute type="xs:string" name="sql" use="optional" />
<xs:attribute type="xs:string" name="parameter" use="optional" />
<xs:attribute type="xs:string" name="message" use="optional" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="type" use="optional" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="queries" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="query" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:attribute type="xs:string" name="id" use="optional" />
<xs:attribute type="xs:string" name="sql" use="optional" />
<xs:attribute type="xs:string" name="parameter" use="optional" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="modify" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="validations" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="validation" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="validationquery" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:attribute type="xs:string" name="id" use="optional" />
<xs:attribute type="xs:string" name="sql" use="optional" />
<xs:attribute type="xs:string" name="parameter" use="optional" />
```

```
<xs:attribute type="xs:string" name="message" use="optional" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="type" use="optional" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="queries" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="query" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:attribute type="xs:string" name="id" use="optional" />
<xs:attribute type="xs:string" name="sql" use="optional" />
<xs:attribute type="xs:string" name="parameter" use="optional" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<!--<xs:simpleContent>
<xs:extension base="xs:string">-->
<xs:attribute type="xs:string" name="id" use="required"/>
<xs:attribute name="class" use="required">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="label"/>
<xs:enumeration value="image"/>
<xs:enumeration value="textlink"/>
<xs:enumeration value="list"/>
<xs:enumeration value="listitem"/>
<xs:enumeration value="inputtext"/>
<xs:enumeration value="textarea"/>
<xs:enumeration value="combo"/>
<xs:enumeration value="button"/>
<xs:enumeration value="form"/>
<xs:enumeration value="iframe"/>
<xs:enumeration value="ajaxcomponent"/>
<xs:enumeration value="applet"/>
<xs:enumeration value="DBgrid"/>
<xs:enumeration value="Conditionalgrid"/>
```

```
<xs:enumeration value="tree"/>
<xs:enumeration value="statictree"/>
<xs:enumeration value="dynamictree"/>
<xs:enumeration value="dbform"/>
<xs:enumeration value="reportwindow"/>
<xs:enumeration value="report"/>
<xs:enumeration value="chart"/>
<xs:enumeration value="flashcomponent"/>
<xs:enumeration value="inputfile"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute type="xs:string" name="size" use="optional"/> <!--For
inputtext, textarea-->
<xs:attribute type="xs:string" name="type" use="optional"/> <!--For
inputtext, textarea-->
<xs:attribute type="xs:string" name="tabindex" use="optional"/> <!--
For inputtext, textarea-->
<xs:attribute type="xs:string" name="name" use="optional"/> <!--For
inputtext, textarea-->
<xs:attribute type="xs:string" name="maxLength" use="optional"/> <!--
For inputtext, textarea-->
<xs:attribute type="xs:string" name="style" use="optional"/> <!--For
inputtext, textarea-->
<xs:attribute type="xs:string" name="rows" use="optional"/> <!--For
textarea -->
<xs:attribute type="xs:string" name="cols" use="optional"/> <!--For
textarea -->
<xs:attribute type="xs:string" name="jscontrol" use="optional"/> <!-- -->
<xs:attribute type="xs:string" name="loadurl" use="optional"/> <!-- For
DBgrid, Conditionalgrid-->
<xs:attribute type="xs:string" name="editurl" use="optional"/> <!-- For
DBgrid, Conditionalgrid-->
<xs:attribute type="xs:string" name="caption" use="optional"/> <!-- For
DBgrid, Conditionalgrid-->
<xs:attribute type="xs:string" name="sortname" use="optional"/> <!--
For DBgrid, Conditionalgrid-->
<xs:attribute name="sortorder" use="optional"> <!-- For DBgrid,
Conditionalgrid-->
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="asc" />
<xs:enumeration value="desc" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute type="xs:string" name="gridnavbar" use="optional"/> <!--
For DBgrid, Conditionalgrid-->
```

```
<xs:attribute type="xs:string" name="treedataremotefunction"
use="optional"/> <!-- For tree-->
<xs:attribute type="xs:string" name="onselectremotefunction"
use="optional"/> <!-- For tree-->
<xs:attribute type="xs:string" name="autocollapse" use="optional"/>
<!-- For tree-->
<xs:attribute type="xs:string" name="initialiseonload" use="optional"/>
<!-- For tree-->
<xs:attribute type="xs:string" name="lazynode" use="optional"/> <!--
For statictree-->
<xs:attribute type="xs:string" name="tooltip" use="optional"/> <!-- For
statictree-->
<xs:attribute type="xs:string" name="parentquery" use="optional"/> <!-
- For dynamictree-->
<xs:attribute type="xs:string" name="childquery" use="optional"/> <!--
For dynamictree-->
<xs:attribute type="xs:string" name="parameter" use="optional"/> <!--
For dynamictree-->
<xs:attribute type="xs:string" name="archieve" use="optional"/> <!--
For applet-->
<xs:attribute type="xs:string" name="codebase" use="optional"/> <!--
For applet-->
<xs:attribute type="xs:string" name="mayscript" use="optional"/> <!--
For applet-->
<xs:attribute type="xs:string" name="width" use="optional"/> <!-- For
iframe-->
<xs:attribute type="xs:string" name="height" use="optional"/> <!-- For
iframe-->
<xs:attribute type="xs:string" name="frameborder" use="optional"/> <!-
- For iframe-->
<xs:attribute type="xs:string" name="allowtransparency"
use="optional"/> <!-- For iframe-->
<xs:attribute type="xs:string" name="scrolling" use="optional"/> <!--
For iframe-->
<xs:attribute type="xs:string" name="background-color"
use="optional"/> <!-- For iframe-->
<!--</xs:extension>
</xs:simpleContent>-->
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="layout" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="part" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
```

```
<!--<xs:simpleContent>
<xs:extension base="xs:string">-->
<xs:attribute type="xs:string" name="id" use="required"/>
<xs:attribute type="xs:string" name="height" use="optional"/>
<xs:attribute type="xs:string" name="width" use="optional"/>
<xs:attribute type="xs:string" name="left" use="optional"/>
<xs:attribute type="xs:string" name="top" use="optional"/>
<xs:attribute type="xs:string" name="position" use="optional"/>
<xs:attribute type="xs:string" name="parent_id" use="optional"/>
<xs:attribute type="xs:string" name="scrolling" use="optional"/><!-- For
iframe-->
<!--</xs:extension>
</xs:simpleContent>-->
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="id"/>
</xs:complexType>
</xs:element>
<xs:element name="content" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="part" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="chartstructure" minOccurs="0"
maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="matric_title" type="xs:string" />
<xs:element name="description" type="xs:string" />
<xs:element name="X_axis_query" type="xs:string" />
<xs:element name="series1_query" type="xs:string" />
<xs:element name="series2_query" type="xs:string" />
<xs:element name="series3_query" type="xs:string" />
<xs:element name="legenddata1" type="xs:string" />
<xs:element name="legenddata2" type="xs:string" />
<xs:element name="legenddata3" type="xs:string" />
<xs:element name="chart_type" type="xs:string" />
<xs:element name="subtype" type="xs:string" />
<xs:element name="width" type="xs:string" />
<xs:element name="height" type="xs:string" />
<xs:element name="yaxis_label" type="xs:string" />
<xs:element name="color" type="xs:string" />
<xs:element name="transpose" type="xs:string" />
<xs:element name="stacked" type="xs:string" />
<xs:element name="chart_dimension" type="xs:string" />
</xs:sequence>
```

```
</xs:complexType>
</xs:element>
<xs:element name="combooption" minOccurs="0"
maxOccurs="unbounded">
<xs:complexType>
<!--<xs:simpleContent>
<xs:extension base="xs:string">-->
<xs:attribute type="xs:string" name="name" use="optional"/>
<xs:attribute type="xs:string" name="value" use="optional"/>
<!--</xs:extension>
</xs:simpleContent>-->
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="id" use="required"/>
<xs:attribute type="xs:string" name="value" use="optional"/>
<xs:attribute name="valuetype" use="optional">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="inline" />
<xs:enumeration value="reference" />
<xs:enumeration value="fixed" /> <!-- has combooption taag-->
<xs:enumeration value="query" />
<xs:enumeration value="" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="valueType" use="optional">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="inline" />
<xs:enumeration value="reference" />
<xs:enumeration value="fixed" /> <!-- has combooption taag-->
<xs:enumeration value="query" />
<xs:enumeration value="" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="id"/>
</xs:complexType>
</xs:element>
<xs:element name="style" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="part" maxOccurs="unbounded" minOccurs="0">
```

```
<xs:complexType>
<!--<xs:simpleContent>
<xs:extension base="xs:string">-->
<xs:attribute type="xs:string" name="id" use="required"/>
<xs:attribute type="xs:string" name="value" use="optional"/>
<xs:attribute name="valuetype" use="optional">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="inline" />
<xs:enumeration value="reference" />
<xs:enumeration value="" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<!--</xs:extension>
</xs:simpleContent>-->
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="id"/>
</xs:complexType>
</xs:element>
<xs:element name="behaviour" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="part" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="event">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="name" use="optional">
<xs:simpleType>
<xs:restriction base="xs:string" >
<xs:enumeration value="ondblClickRow" />
<xs:enumeration value="gridComplete" />
<xs:enumeration value="loadComplete" />
<xs:enumeration value="onclick" />
<xs:enumeration value="onchange" />
<xs:enumeration value="invokeurl" />
<xs:enumeration value="onload_onclick();init()" />
<xs:enumeration value="onload_click()" />
<xs:enumeration value="onkeypress" />
<xs:enumeration value="body_onload()" />
<xs:enumeration value="" />
</xs:restriction>
```

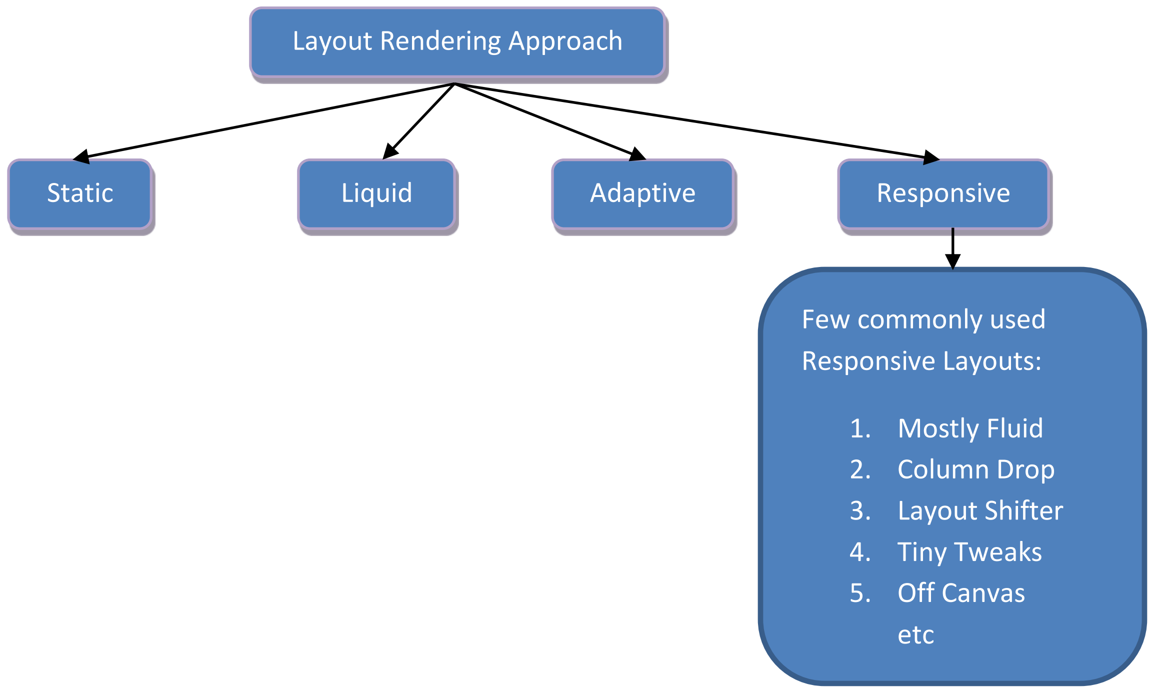
```
</xs:simpleType>
</xs:attribute>
<xs:attribute type="xs:string" name="type" use="optional"/>
<xs:attribute type="xs:string" name="function" use="optional"/>
<xs:attribute type="xs:string" name="callback" use="optional"/>
<xs:attribute type="xs:string" name="javaclass" use="optional"/>
<xs:attribute type="xs:string" name="target" use="optional"/>
<xs:attribute type="xs:string" name="valuetype" use="optional"/>
<xs:attribute type="xs:string" name="resourceid" use="optional"/>
<xs:attribute type="xs:string" name="package" use="optional"/>
<xs:attribute type="xs:string" name="pacakge" use="optional"/>

</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="id" use="optional"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="id"/>
</xs:complexType>
</xs:element>
<xs:element name="resource" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element name="resourceitem" maxOccurs="unbounded"
minOccurs="0">
<xs:complexType>
<!--<xs:simpleContent>
<xs:extension base="xs:string">-->
<xs:attribute type="xs:string" name="id" use="required"/>
<xs:attribute type="xs:string" name="href" use="optional"/>
<xs:attribute name="valuetype" use="optional">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="image" />
<xs:enumeration value="js" />
<xs:enumeration value="css" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<!--</xs:extension>
</xs:simpleContent>-->
</xs:complexType>
</xs:element>
```

```
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="id"/>
<xs:attribute type="xs:string" name="title"/>
<xs:attribute type="xs:string" name="type"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

Appendix 2: Layout Rendering Approach

Positioning flexibility refers to the ability of the layed out components on a page to adjust their size and positions with change in page size or screen size. The flexibility can be applied to any positioning approach discussed so far. Based on the usage of differently sized devices in opening web pages, the layout patterns in terms of flexibility can broadly be classified as follows:

(References[8-10])



Static Layout

- Uses Present Page size
- Does not change based on browser width
- Device wise rendering of static layout is unpredictable
- Mostly not used now-a-days, as separate layout required for separate device sizes

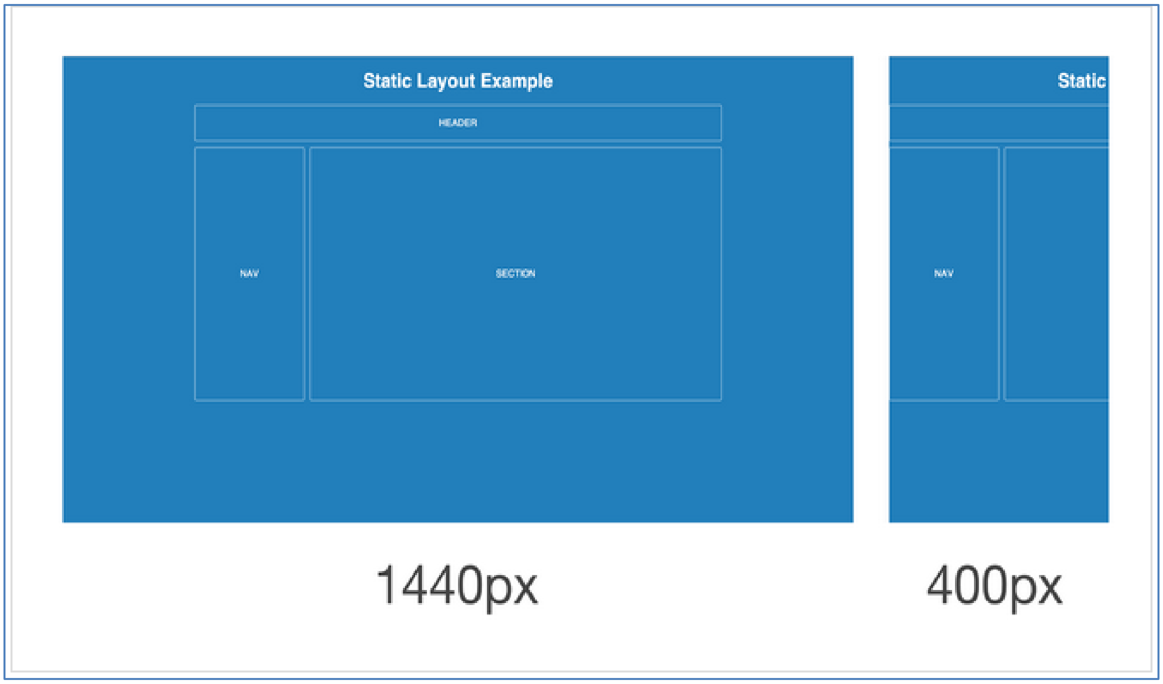


Figure 100: Layout Rendering Approach - Static Page Layout

Liquid Layout

- Also known as Fluid Width layout
- Uses relative units instead of fixed units, e.g. percentage
- Fills width of the page irrespective of browser width
- Drawbacks with very large or very small bowser width

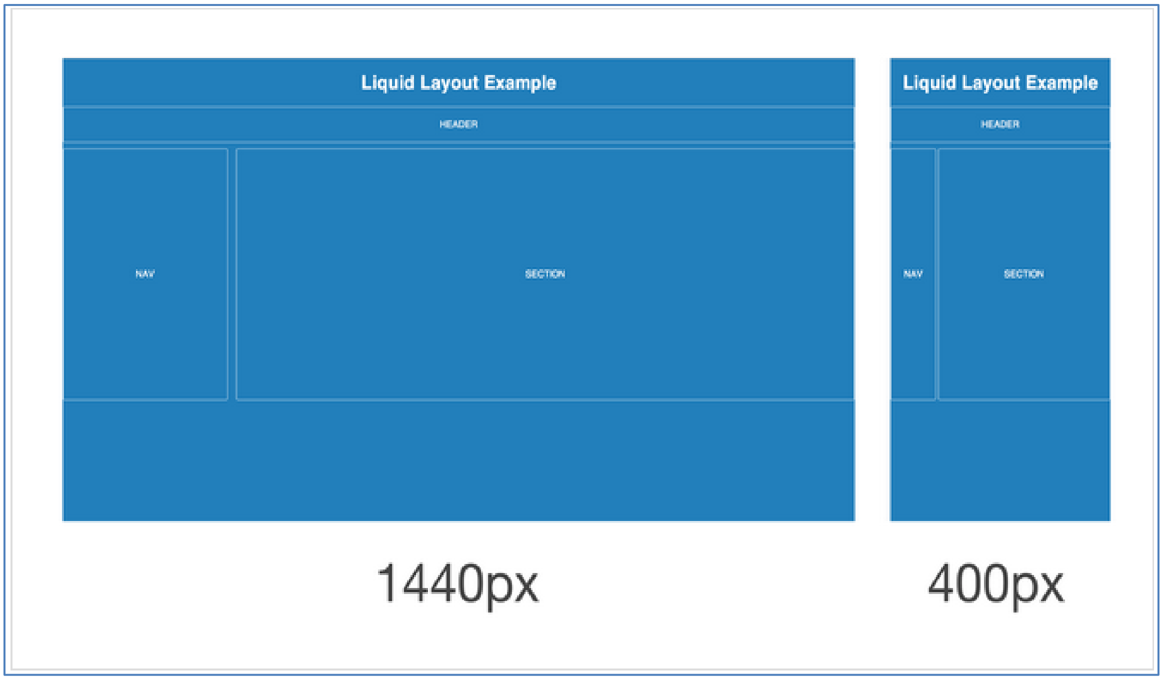


Figure 101: Layout Rendering Approach - Liquid Page Layout

Adaptive Layout

- Uses CSS media queries to detect the width of the browser and alter the layout accordingly
- Adaptive layout uses fixed unit pixels as in Static layout, but there are multiple fixed widths defined
- Media Query is an expression of logic, and when used in combination with other media queries they form an algorithm.

  e.g., an adaptive page layout might say something like this

  "If the browser is 500 pixels wide, set the main content container to be 400 pixels wide. If the browser is 1000 pixels wide, then set the main content container to be 960 pixels wide"
- Adaptive layouts are good stop gap solution if a legacy static layout needs to be converted to support mobile devices
- Less Development time than a responsive layout

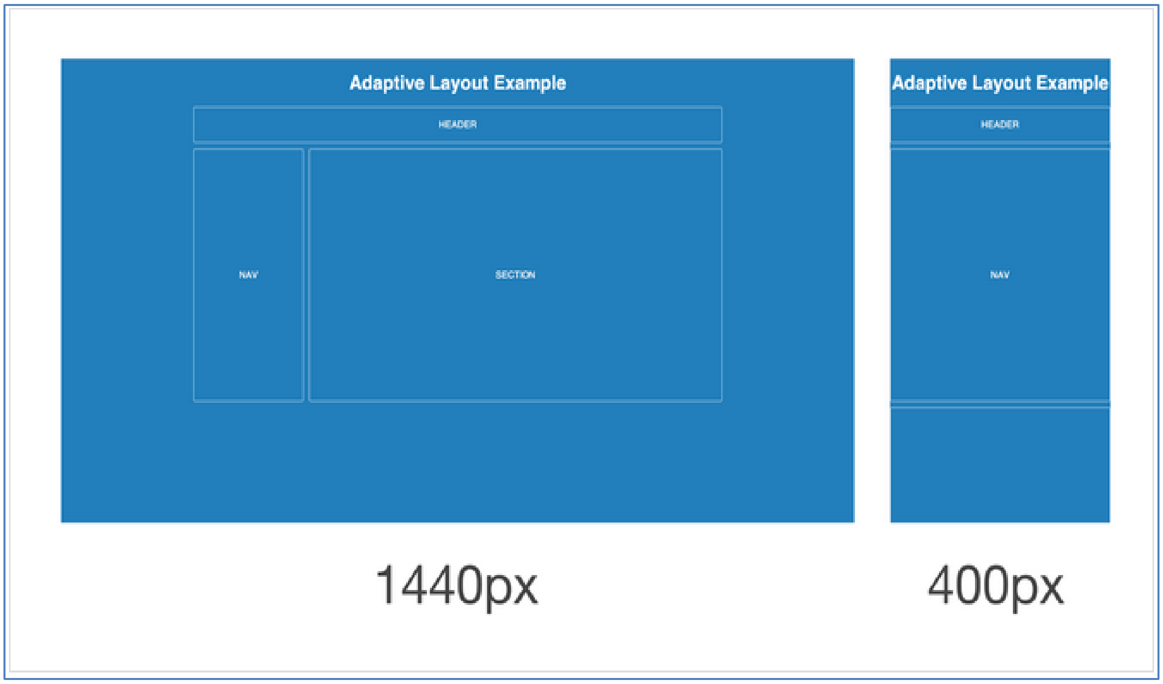


Figure 102: Layout Rendering Approach - Adaptive Page Layout

Responsive Layout

- Responds to the needs of the users and the devices they're using
- The layout changes based on the size and capabilities of the device. For example, on a phone, users would see content shown in a single column view; a tablet might show the same content in two columns
- Combination of Liquid Layout and Adaptive Layout
- As the browser increases or decreases in width, a responsive layout will flex just like a liquid layout
- However, if the browser goes beyond certain widths defined by media query breakpoints, then the layout will change more dramatically to accommodate a wide or narrow width
- Typically responsive designs are built using a mobile-first approach

Below are the examples of few commonly available Responsive layouts:

Mostly Fluid

The mostly fluid pattern consists primarily of a fluid grid. On large or medium screens, it usually remains the same size, simply adjusting the margins on wider screens. On smaller screens, the fluid grid causes the main content to reflow, while columns are stacked vertically.

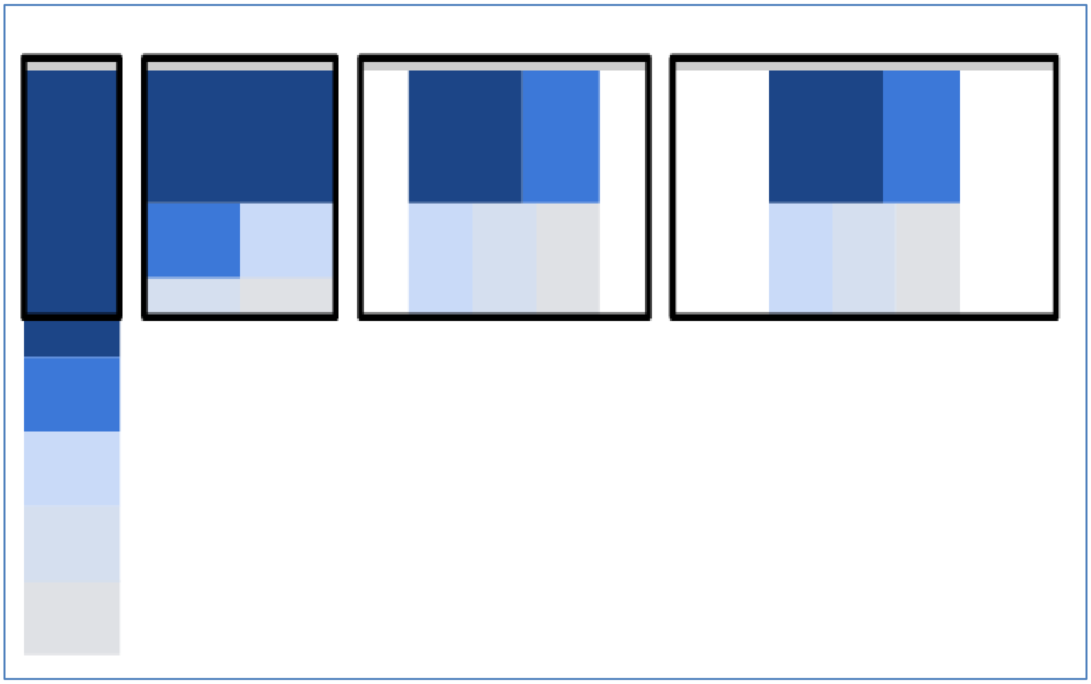


Figure 103: Layout Rendering Approach - Responsive Design - Mostly Fluid Layout

Column Drop

For full-width multi-column layouts, column drop simply stacks the columns vertically as the window width becomes too narrow for the content.

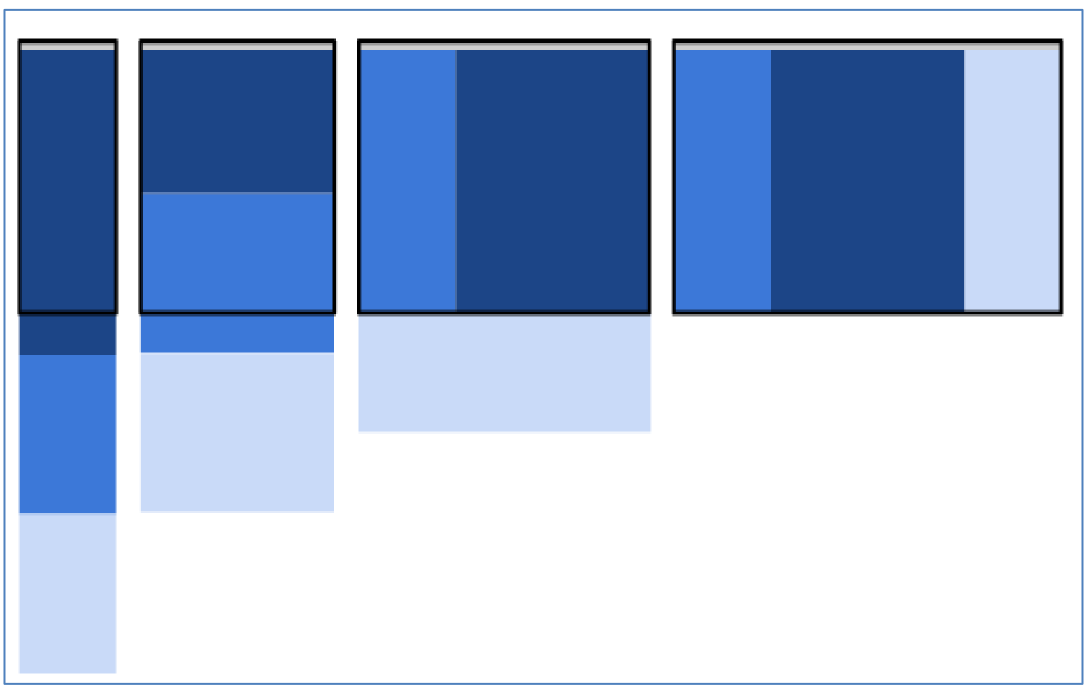


Figure 104: Layout Rendering Approach - Responsive Design - Column Drop Layout

Layout Shifter

The layout shifter pattern is the most responsive pattern, with multiple breakpoints across several screen widths. Key to this layout is the way content moves about, instead of reflowing and dropping below other columns. Due to the significant differences between each major breakpoint, it is more complex to maintain and likely involves changes within elements, not just overall content layout.

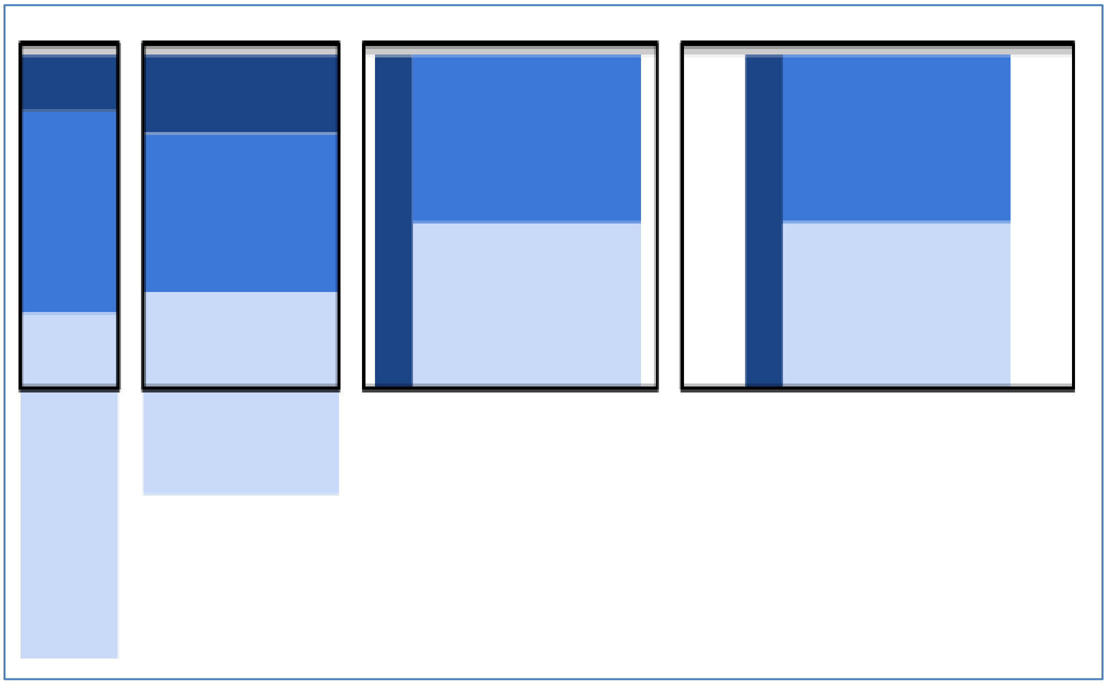


Figure 105: Layout Rendering Approach - Responsive Design - Layout Shifter

Tiny Tweaks

Tiny tweaks simply make small changes to the layout, such as adjusting font size, resizing images or moving content around in very minor ways. It works well on single column layouts such as one page linear websites, text heavy articles.

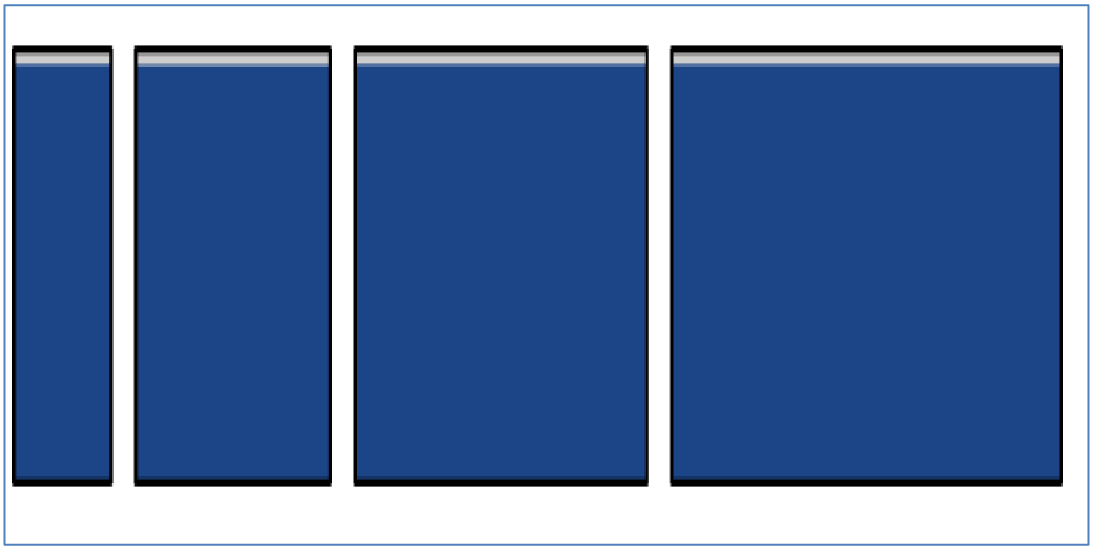


Figure 106: Layout Rendering Approach - Responsive Design - Tiny Tweaks Layout

Off Canvas

Rather than stacking content vertically, the off canvas pattern places less frequently used content, perhaps navigation or app menus off screen, only showing it when the screen size is large enough, and on smaller screens, content is only a click away.

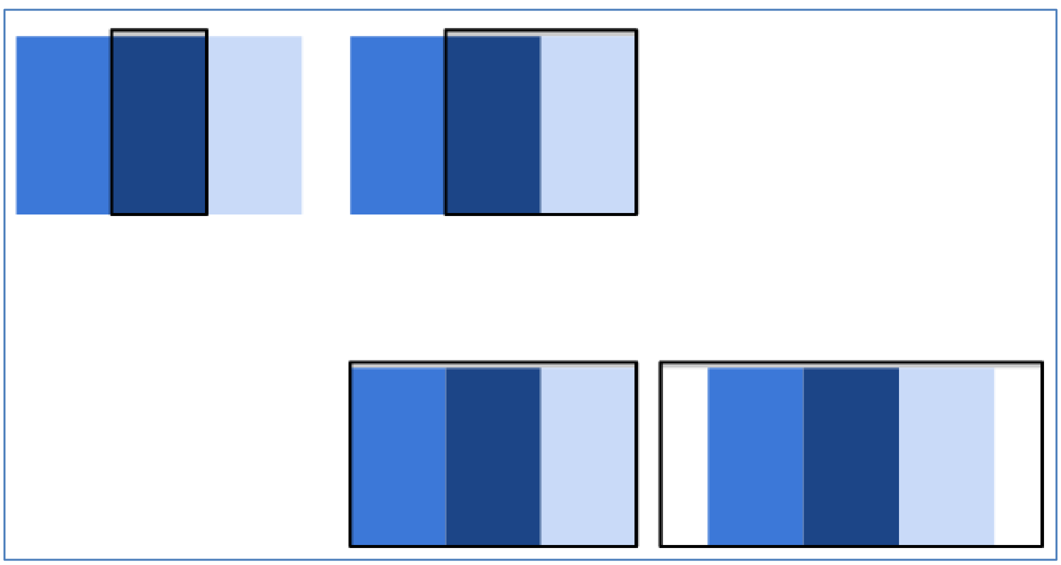


Figure 107: Layout Rendering Approach - Responsive Design - Off Canvas