General Approach

Background

The primary feature of a stock market is public companies looking out to sell some of their shares to the public, investors looking to buy those shares in anticipation of profit, and stock exchanges that bring companies and investors together.

Problem/Challenge

As a company or an investor it's somewhat difficult to get stock exchange details in one place and especially when one is always travelling.

Target Audience

- Business Companies
- Business individuals (Executives, Managers)
- Market Individuals (Brokers)

Solution Plan

Project Scope

The project will analyze the requirements from an implementation perspective and detail the solution to provide visibility into – scope for delivering the identified features. The scope identified as:

Analysis of SXViewer application

- Functional Requirements (features)
- Analyzing source (Yahoo Finance) in reading stock market data
- Benefits and usage of International stock market data to audience
- Benefit and usage of charts (linear graph) to the audience

Technical Design of SXViewer application

- Android Graphical User Interface to display data to audience
- Application connectivity to Yahoo Finance to read stock market data

Development of SXViewer application features

- Real-time stock quotes
- Stock market watch activity
- Drill down stock market details
- Interactive stock charts (linear graph)
- International Exchanges updates

SXViewer application deployment

- Android mobile device

The proposed solution is planned to divide in two major areas:

1. **Graphical User Interface:**
   The Graphical User Interface for SXViewer will be developed using Google Android SDK (Software Development Kit) and C# language. The SDK will help developing screens to display stock related data and charts (linear graph) to the audience/customer.

2. **Backend Service:**
   In order to read stock data; Yahoo Finance Service has been planned to use. As per the initial study and a little research it has been revealed that Yahoo Finance Service is a FREE service which helps reading international stock market data.

In addition Yahoo Finance Service provides data in a format which is not user friendly. Since we are developing a Graphical User Interface to display stock market related data for customers, we have to design a middle layer (Web Service) which should make the data easily understandable by the Graphical User Interface.

ANALYSIS

Functional Requirements

We will be building primarily four Screens and one Web Service:

1. Splash Screen
2. Market Watch
3. Stock Script Detail View
4. Add Stock
5. Get Stock Quote
6. Show Stock Chart

User Requirement Specification

Splash Screen

This will display the initial screen with the SXViewer application name

Market Watch (Main Screen):

This screen will display all the requested Stock Exchange Quotes for group of selected companies.

Stock Detail View (Detail Screen):

This screen will display Stock Details of individual companies.

Add Stock (User Input Screen):

This screen will allow user to add the company of his choice to market watch screen.

Get Stock (Input/Output Screen):

This screen will also allow user to get the stock quote of company, but not be added in the market watch screen. This will show the stock details for keeping check on other un-interested stocks. If user’s interest start building on a particular stock, they can simply add it in main screen using add stock screen.

Show Stock Chart (Line Graph):

This is especially for the manager who relies on Graphical Data instead of reading lines.

Software Required:

- MS Visual Studio
- Xamarin (Mono for Android)
- Google Android SDK
- SQLite
- MS Project
- MS Word
- MS Visio

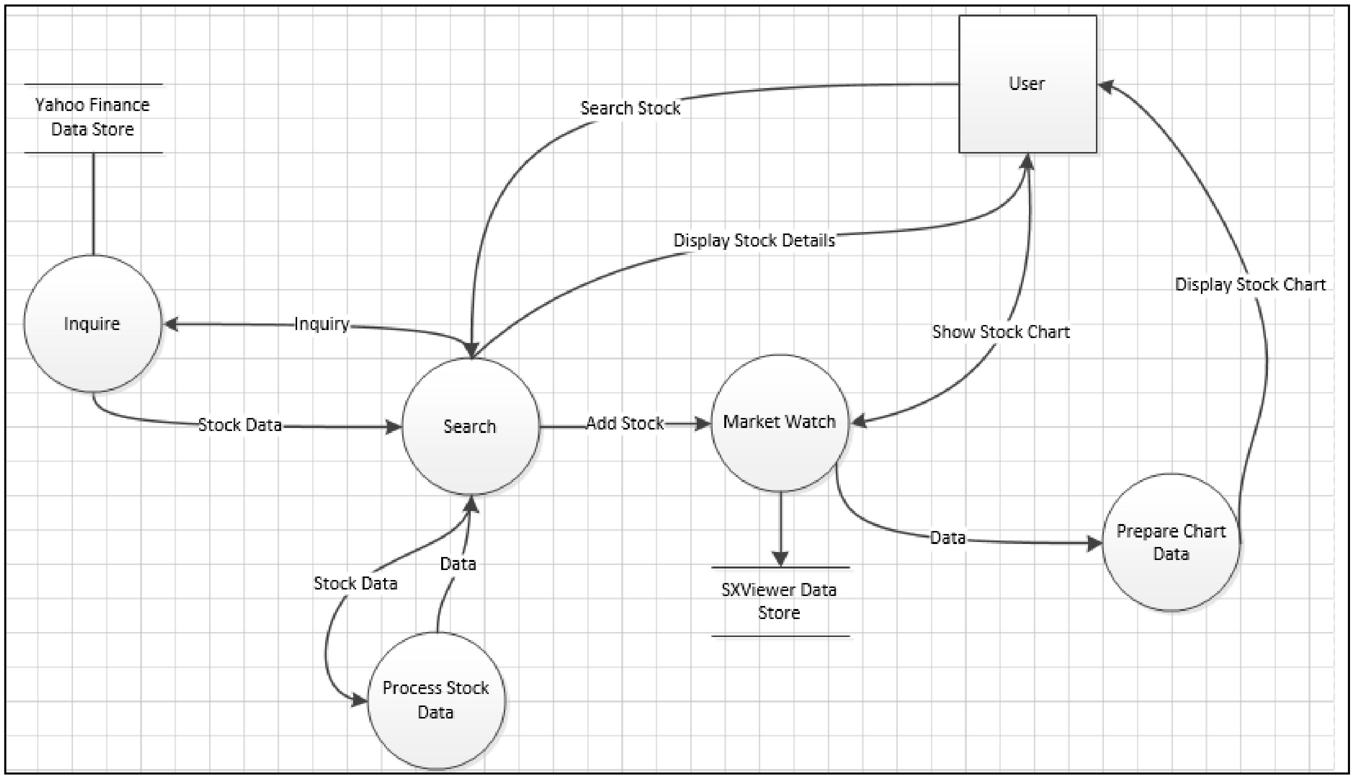
Software Requirements:

- Windows Operating System (typically Windows 7 or Windows 8)
- Android Emulator (this comes with Android SDK – Software Development Kit)
- Internet Connection (this will be used to connect to Yahoo Finance API)

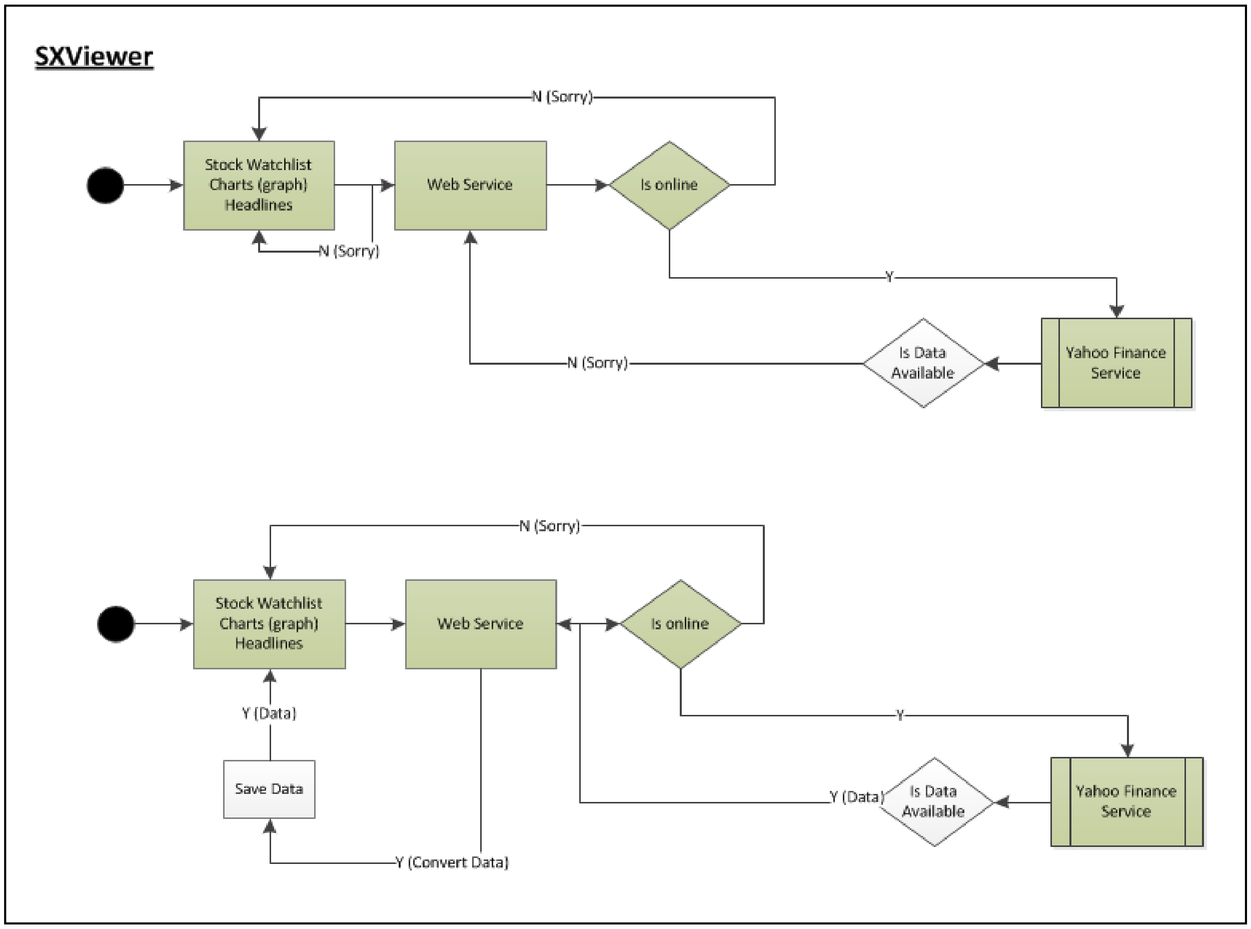
Hardware Requirements:

Requirement	Required
Processor	Recommended: 1 gigahertz (GHz) processor[1]
RAM	Minimum: 1GB Recommended: 2GB
Available Hard Disk Space	• 1 GB of available space required on system drive • 2 GB of available space required on installation drive for the required software
Operating System	• Windows 7 or Windows 8
CD-ROM Drive or DVD-ROM Drive	Not Required
Video	Minimum: 800 X 600, 256 colours Recommended: 1024 X 768, High Colour 32-bit
Mouse	Microsoft mouse or compatible pointing device
Keyboard	Compatible Keyboard device

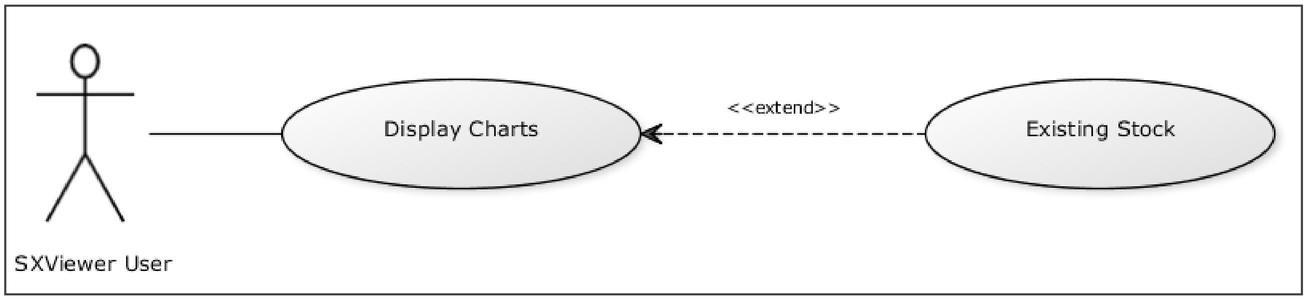
General Functionality of SXViewer

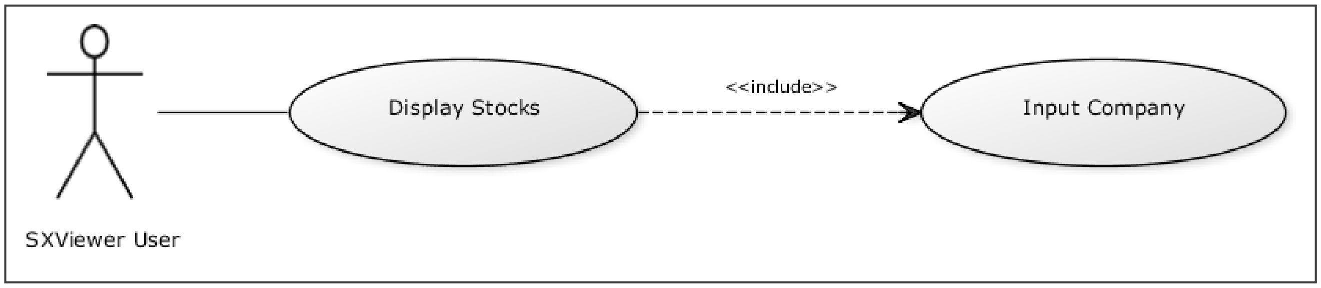


DFD or Data Flow Diagram

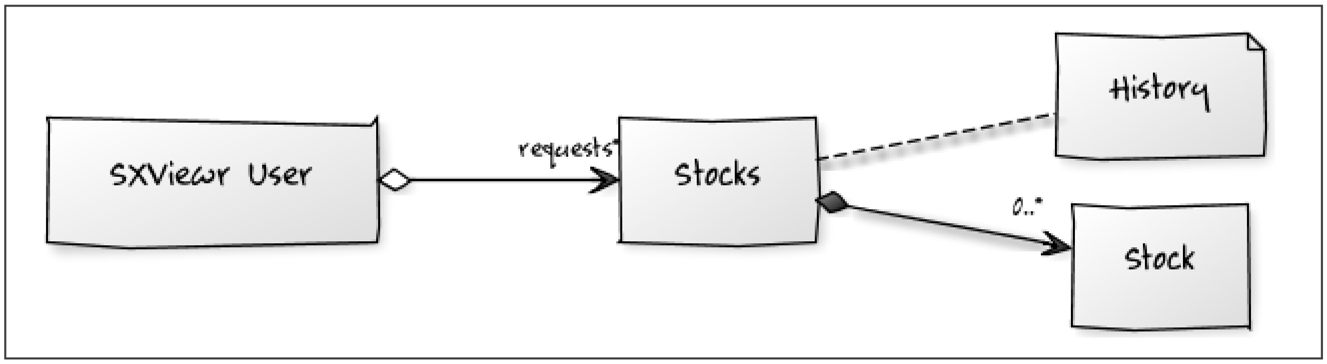


Use Case Diagram

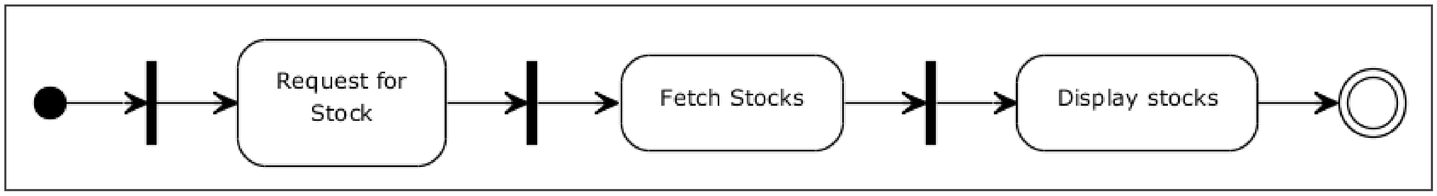


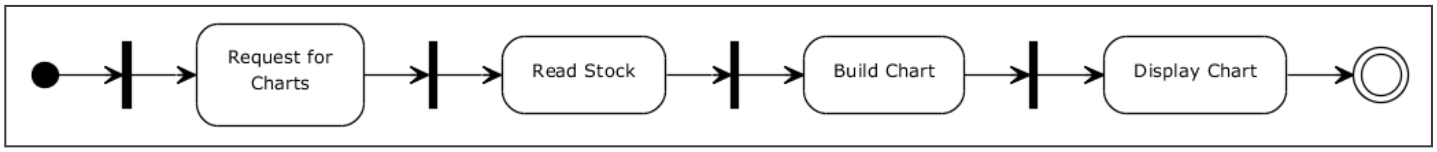


Class Diagram

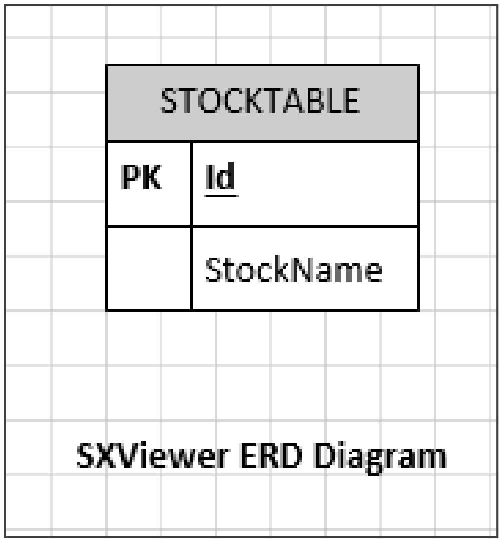


Activity Diagram





Entity Relationship Diagram:



Functional Dependency:

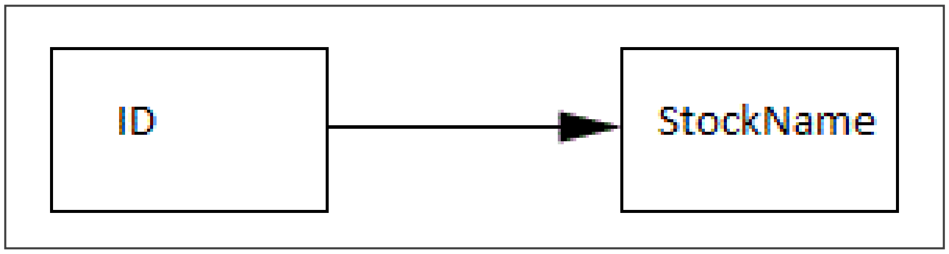
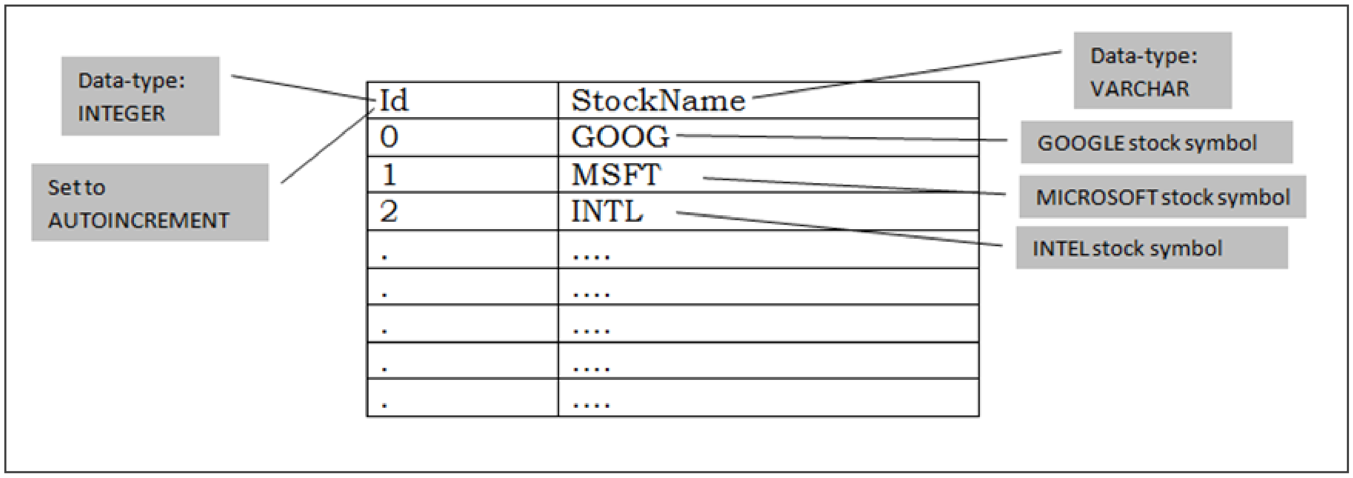


Table Structure:

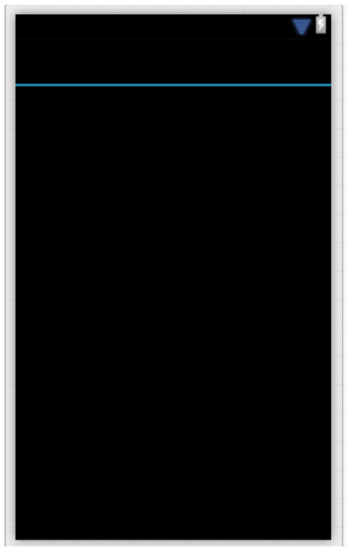


Id	StockName
0	GOOG
1	MSFT
2	INTL
.	....
.	....
.	....
.	....
.	....

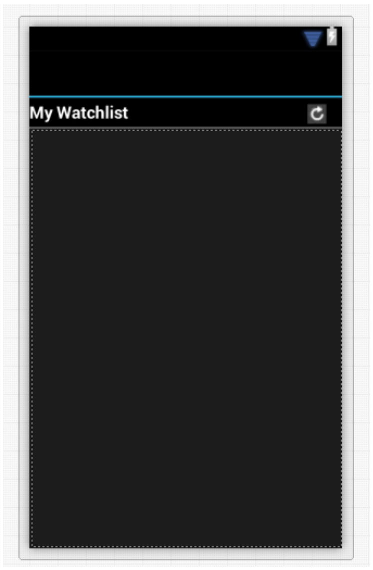
Layout of Screens

We have prepared the Layout of screens in **Microsoft Visual Studio.**

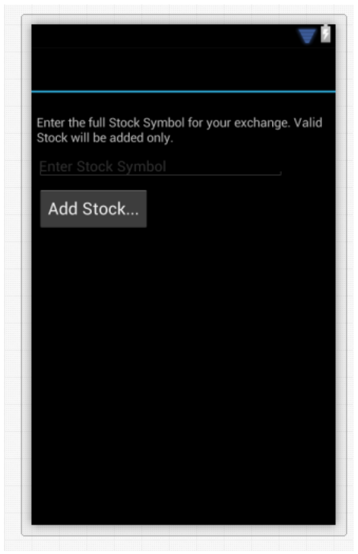
SPLASH SCREEN (Build at run-time)



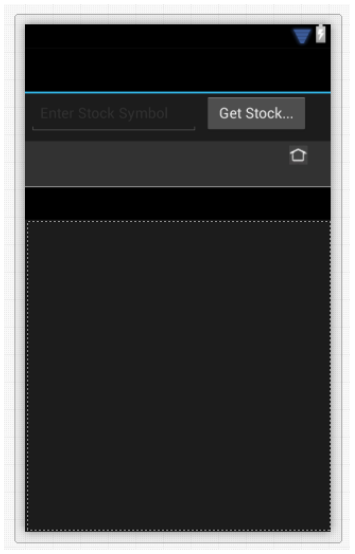
MARKET WATCH SCREEN:



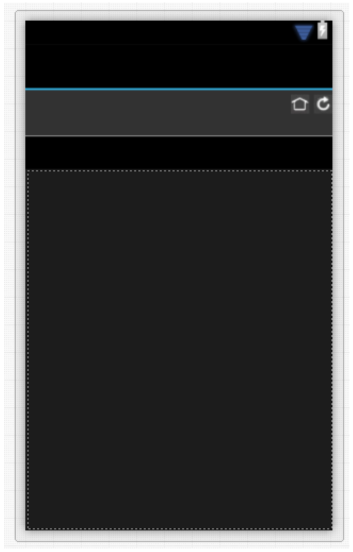
ADD STOCK SCREEN:


Enter the full Stock Symbol for your exchange. Valid Stock will be added only.

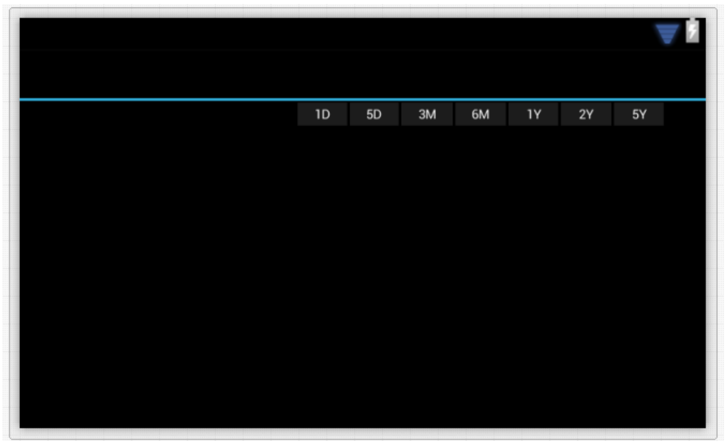
GET STOCK SCREEN:



STOCK DETAILS SCREEN:

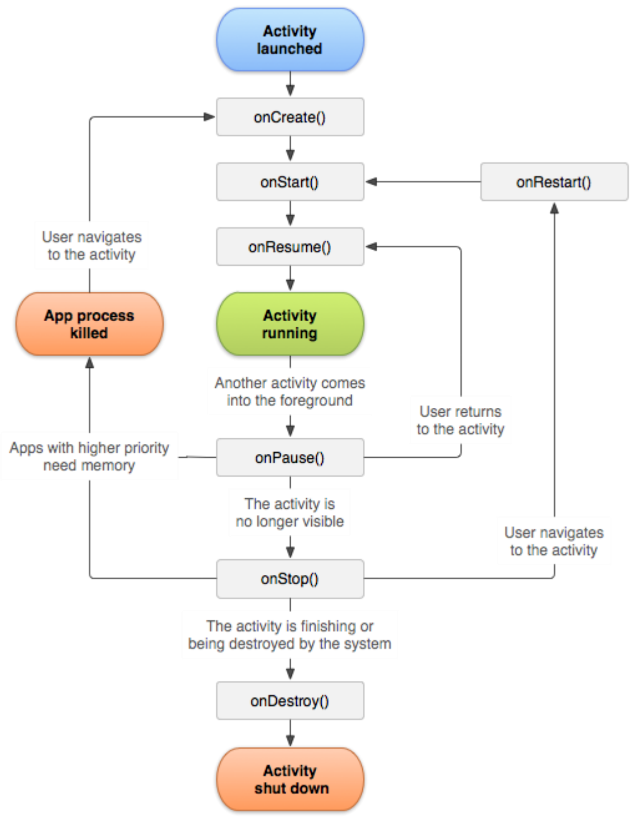


SHOW CHART SCREEN:

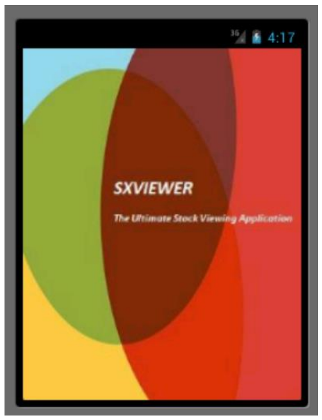


IMPLEMENTATION

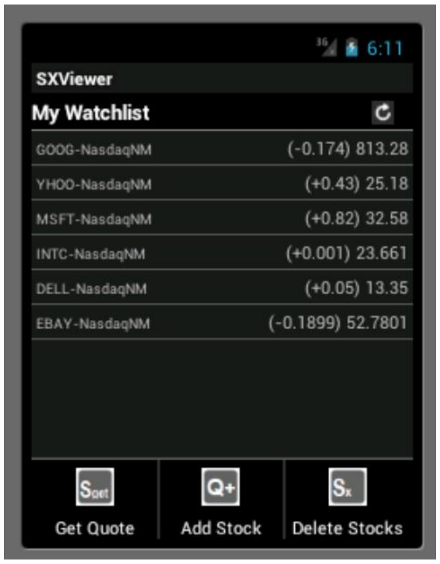
Activity Life Cycle:


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Splash Screen



Market Watch Screen:

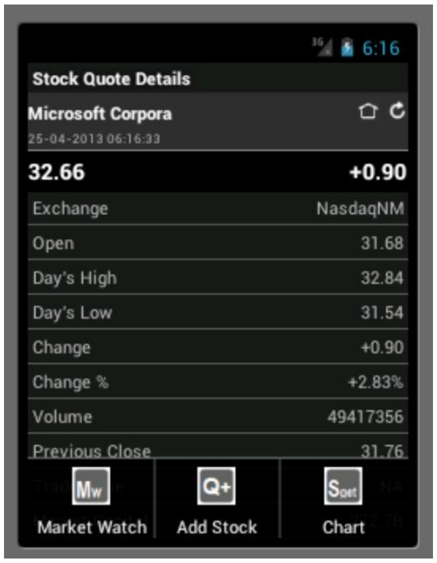


SXViewer

My Watchlist

GOOG-NasdaqNM	(-0.174) 813.28
YHOO-NasdaqNM	(+0.43) 25.18
MSFT-NasdaqNM	(+0.82) 32.58
INTC-NasdaqNM	(+0.001) 23.661
DELL-NasdaqNM	(+0.05) 13.35
EBAY-NasdaqNM	(-0.1899) 52.7801

Stock Details Screen



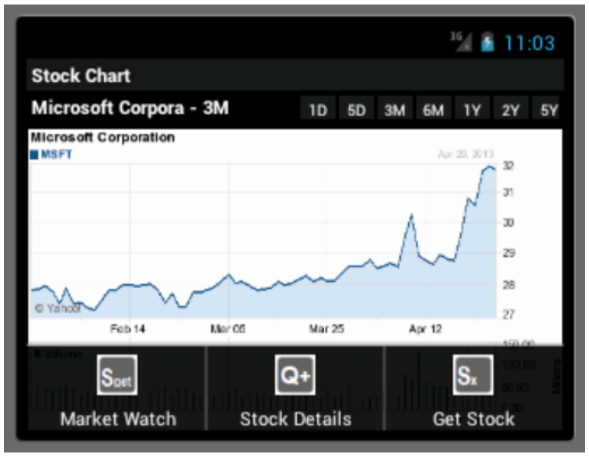
Stock Quote Details

Microsoft Corpora

25-04-2013 06:16:33

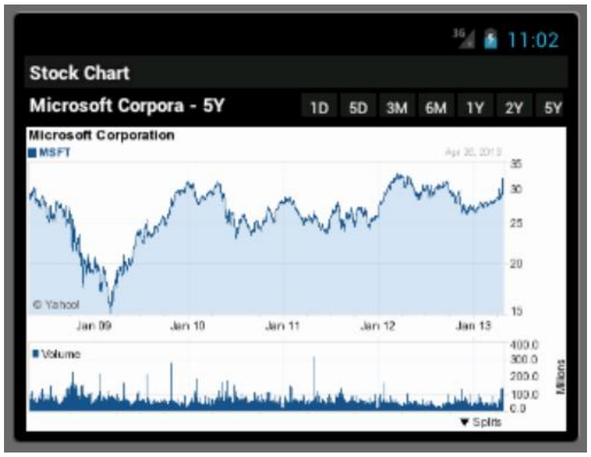
32.66	**+0.90**
Exchange	NasdaqNM
Open	31.68
Day's High	32.84
Day's Low	31.54
Change	+0.90
Change %	+2.83%
Volume	49417356
Previous Close	31.76

Show Chart Screen:



Stock Chart

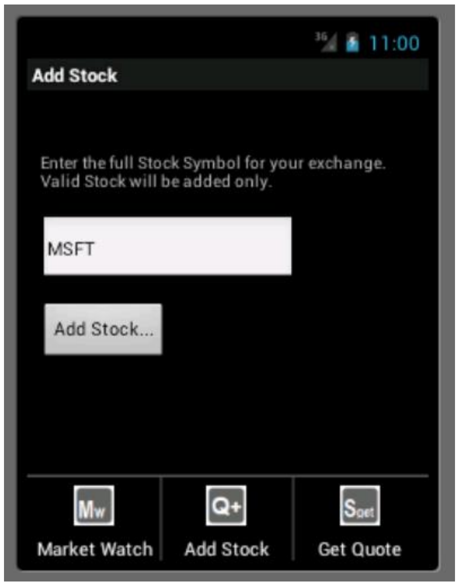
Microsoft Corpora - 3M



Stock Chart

Microsoft Corpora - 5Y

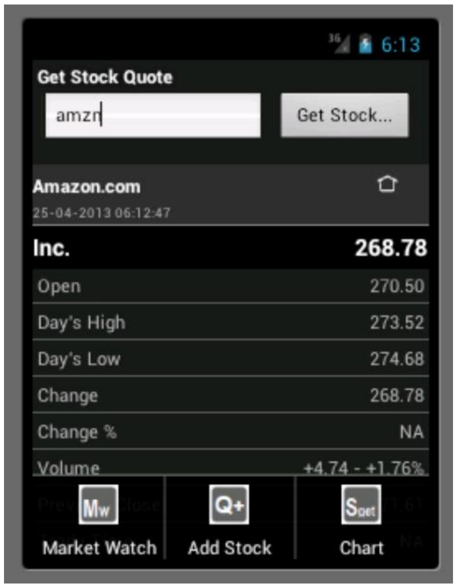
Add Stock Screen



Enter the full Stock Symbol for your exchange.
Valid Stock will be added only.

MSFT

Get Stock Screen

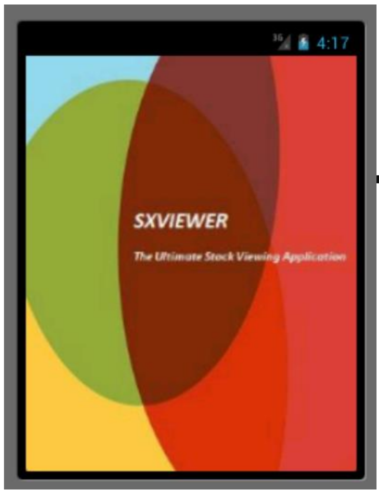


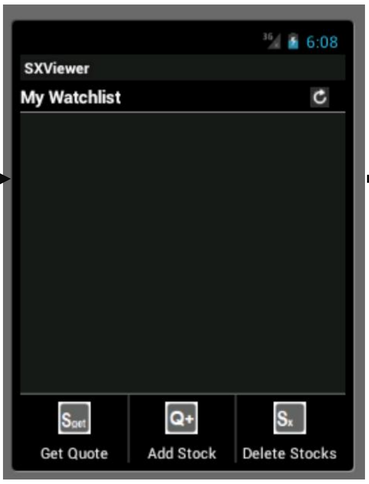
amzn

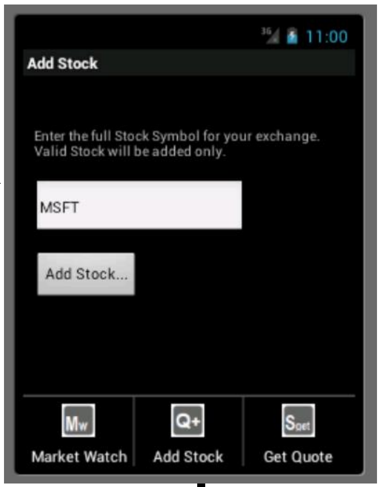
Amazon.com
25-04-2013 06:12:47

Inc.	268.78
Open	270.50
Day's High	273.52
Day's Low	274.68
Change	268.78
Change %	NA
Volume	+4.74 - +1.76%

Below is the one possible cycle of movements between screens:





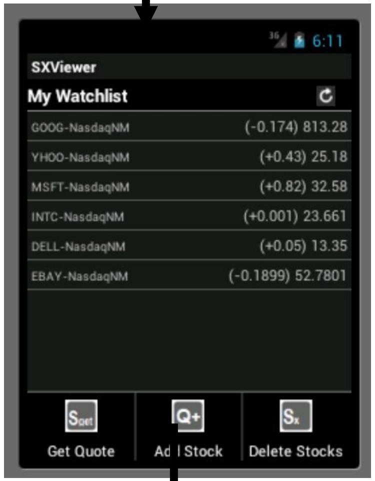

11:00

Add Stock

Enter the full Stock Symbol for your exchange.
Valid Stock will be added only.

MSFT

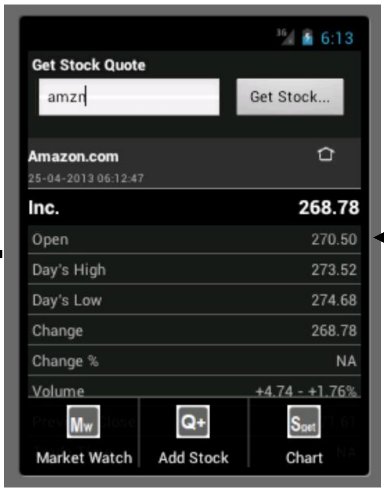
Add Stock...


6:11

SXViewer

My Watchlist

GOOG-NasdaqNM	(-0.174) 813.28
YHOO-NasdaqNM	(+0.43) 25.18
MSFT-NasdaqNM	(+0.82) 32.58
INTC-NasdaqNM	(+0.001) 23.661
DELL-NasdaqNM	(+0.05) 13.35
EBAY-NasdaqNM	(-0.1899) 52.7801


6:13

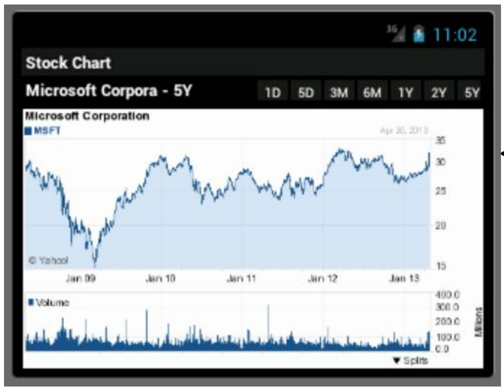
Get Stock Quote

amzn

Get Stock...

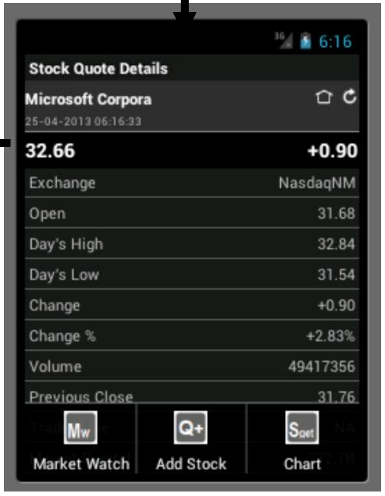
Amazon.com
25-04-2013 06:12:47

Inc.	268.78
Open	270.50
Day's High	273.52
Day's Low	274.68
Change	268.78
Change %	NA
Volume	+4.74 - +1.76%


11:02

Stock Chart

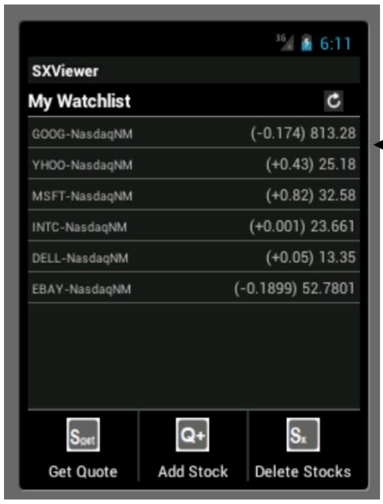
Microsoft Corpora - 5Y


6:16

Stock Quote Details

Microsoft Corpora
25-04-2013 06:16:33

32.66	+0.90
Exchange	NasdaqNM
Open	31.68
Day's High	32.84
Day's Low	31.54
Change	+0.90
Change %	+2.83%
Volume	49417356
Previous Close	31.76


6:11

SXViewer

My Watchlist

GOOG-NasdaqNM	(-0.174) 813.28
YHOO-NasdaqNM	(+0.43) 25.18
MSFT-NasdaqNM	(+0.82) 32.58
INTC-NasdaqNM	(+0.001) 23.661
DELL-NasdaqNM	(+0.05) 13.35
EBAY-NasdaqNM	(-0.1899) 52.7801

Future work

In future we planned to update our software SXViewer by adding some more useful features that will make the software more efficient. Some of these features are:

Notification Feature: This feature will notify users about any change in the stock prices of their favorite stocks (saved in market watch screen). This feature will keep the users up-to-date.

Stock Details sharing feature: More buttons will be added in the Stock Details screen, which allow users to share the interested stock details on the web (e.g. Facebook/Twitter).

Daily ticker screen: One more screen will be added, that will implement the most popular **Daily ticker** service of Yahoo Finance. Daily ticker contains the insight of popular business stories (investing, economy, politics and company earnings).

Highlighting Feature: One more feature will be added that will highlight the favorite stock in the market watch, if its price falls

APPENDIX

Splash Screen Activity Code (C#)

```
/*This is the first screen. This displays the SXViewer application name  and take the control
to the Market Watch Screen(Main Screen)*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using SXViewer;

namespace SXViewer
{
    [Activity(MainLauncher = true, Label = "SXViewer", Theme = "@style/Theme.Splash",
NoHistory = true)]
    public class MainActivity : Activity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            StartActivity(typeof(MarketWatchActivity));
        }
    }
}
```

Market Watch Screen Activity Code (C#)

```
/*Market Watch screen displays all the requested Stock Exchange Quotes for group of selected
companies.This is the main screen of the SXViewer; it will be initially empty (without
stocks). Stocks will be added using Add Stock Screen.*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.Graphics;

using System.Xml;
using System.Threading;
using Android.Database.Sqlite;
using SXViewer.Core.BusinessLayer;
using SXViewer.stockwebservice;

namespace SXViewer
{
    [Activity(MainLauncher = false, Label = "SXViewer",
        ScreenOrientation = Android.Content.PM.ScreenOrientation.Portrait,
        Icon = "@drawable/icon_activity")]
    public class MarketWatchActivity : Activity
    {
        private bool IsLoading = false;
        private TableLayout tablelayout = null;

        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);


            View titleView = Window.FindViewById(Android.Resource.Id.Title);
            //update the default title
            if (titleView != null)
            {
                IViewParent parent = titleView.Parent;
                if (parent != null && (parent is View))
                {
                    View parentView = (View)parent;
                    parentView.SetBackgroundColor(Color.Rgb(28, 28, 28));
                    parentView.SetMinimumHeight(32);
                    parentView.SetMinimumHeight(32);
                }
            }

            // Set our view from the "main" layout resource
            SetContentView(Resource.Layout.MarketWatch);

            tablelayout = FindViewById<TableLayout>(Resource.Id.deatlWatchLayout);
            ProgressDialog progress = ProgressDialog.Show(this, "", "Loading Market Watch...",
                                                          true);
            new Thread(new ThreadStart(() =>
            {
                this.RunOnUiThread(() =>
                {
                    doLoadMarketWatch();
                    progress.Dismiss();
                });
            })).Start();
            IsLoading = false;

            // Refresh button code
            ImageView btnRefresh = FindViewById<ImageView>(Resource.Id.watchRefresh);
            btnRefresh.Click += (sender, e) =>
            {
                if (!IsLoading)
                {
                    ProgressDialog progress1 = ProgressDialog.Show(this, "", "Loading
Quote...",
                                                                  true);
                    new Thread(new ThreadStart(() =>
                    {
                        this.RunOnUiThread(() =>
                        {
                            doLoadMarketWatch();
                            progress1.Dismiss();
                        });
                    })).Start();
                    IsLoading = false;
                }
            };


        }

     // This method loads the Market Watch screen (also updates the stock quotes, saved
previously)
        private bool doLoadMarketWatch()
        {
            bool ret = false;
            string strStocks = string.Empty; //create a comma delimited stock name from
database

            IEnumerable<Stock> stockdata = StockManager.GetStocks();
            foreach (Stock rec in stockdata)
            {
                strStocks += rec.StockName + ",";
            }
            if (strStocks.Length > 0) strStocks.TrimEnd(',');
            if (strStocks != string.Empty)
            {
                PopulateDataToControls(strStocks);
            }
            else
            {
                //clear all views from table layout
                tablelayout.RemoveAllViews();
                tablelayout.RefreshDrawableState();
            }
            return ret;
        }

        // This method gets data from the webservice
        private void PopulateDataToControls(string _stocks)
        {
            //we have now the stocks delimited by comma
           //this string will be passed to Webservice as a parameter to fetch the stock block
in xml

            string strXML = string.Empty;
            stockwebservice.StockWebservice quoteObject = null;
            try
            {
                quoteObject = new stockwebservice.StockWebservice();
                strXML = quoteObject.GetStockQuote(_stocks);
            }
            finally
            {
                quoteObject.Dispose();
                quoteObject = null;
            }

            //if error occurred while connecting to web service
            if (strXML.Substring(0, 5) == "error")
            {
                var t = Toast.MakeText(this, "Error connecting to web service. Please check
your
                                       Internet connection...", ToastLength.Short);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();
                return;
            }
            if (strXML.ToLower() == "exception")
            {
                var t = Toast.MakeText(this, "Service not available now. Please try after
                                       sometime...", ToastLength.Short);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();
                return;
            }

            //load the xml to XmlDocument
            XmlDocument doc = new XmlDocument();
            doc.LoadXml(strXML);

            tablelayout.RemoveAllViews();
            tablelayout.RefreshDrawableState();

            XmlNodeList xnList = doc.SelectNodes("/stock/symbol");
            foreach (XmlNode xn in xnList)
            {
                if (xn != null)
                {
                    TableRow demoTableRow = new TableRow(this);
                    TextView tv_l = new TextView(this);
                    TextView tv_r = new TextView(this);

                    tv_l.SetPadding(3, 3, 3, 3);
                    tv_r.SetPadding(3, 3, 3, 3);
                    tv_r.Gravity = GravityFlags.Right;
                    tv_l.SetTextSize(Android.Util.ComplexUnitType.Px, 21);
                    tv_l.Text = xn["code"].InnerText.Trim() + "-" +
xn["exchange"].InnerText.Trim();
                    tv_r.Text = "(" + xn["change"].InnerText.Trim() + ") " +
                                  xn["last"].InnerText.Trim();

                    demoTableRow.Clickable = true;
                    demoTableRow.Click += (sender, e) =>
                    {
                        doRowClick(sender);
                    };

                    demoTableRow.AddView(tv_l);
                    demoTableRow.AddView(tv_r);

                    tablelayout.AddView(demoTableRow);
                    View vLineRow = new View(this);
                    vLineRow.SetMinimumHeight(1);
                    vLineRow.SetBackgroundColor(Color.Rgb(88, 88, 88));
                    tablelayout.AddView(vLineRow);
                }
            }
        }

        // This method shows the stock details of chosen(clicked) stock
        private void doRowClick(object Sender)
        {
            TableRow tr = Sender as TableRow;
            if (tr != null)
            {
                TextView v = tr.GetChildAt(0) as TextView;
                string _script = v.Text.Trim();
                int _index = _script.IndexOf('-');
                if (_index > 0) _script = _script.Substring(0, _index);

                if (_script != string.Empty)
                {
                    var t = Toast.MakeText(this, "Loading " + _script, ToastLength.Short);
                    t.SetGravity(GravityFlags.Center, 0, 0);
                    t.Show();


                    var intent = new Intent();
                    intent.SetClass(this, typeof(StockDetailsActivity));
                    intent.PutExtra("Script", _script);
                    StartActivity(intent);
                }
            }
        }

        // This method is overridden to create a menu with three menu options
        public override bool OnCreateOptionsMenu(IMenu menu)
        {
            menu.Add("Get Quote").SetIcon(Resource.Drawable.ic_stock_get); ;
            menu.Add("Add Stock").SetIcon(Resource.Drawable.ic_quote_add);
            menu.Add("Delete Stocks").SetIcon(Resource.Drawable.ic_stock_delete);
            return true;
        }

        // This method is overridden for adding desired functionality in the menu options
        public override bool OnOptionsItemSelected(IMenuItem item)
        {
            switch (item.TitleFormatted.ToString())
            {
                case "Delete Stocks":
                    doDeleteStocks(); break;
                case "Add Stock":
                    doOpenAddStock(); break;
                case "Get Quote":
                    doOpenGetStock(); break;
            }
            return base.OnOptionsItemSelected(item);
        }

        // This method deleted all the stocks saved in Market Watch
        protected void doDeleteStocks()
        {
            StockManager.DeleteAllStocks();
            if (!IsLoading)
            {
                ProgressDialog progress1 = ProgressDialog.Show(this, "", "Deleting Stock...",
true);
                new Thread(new ThreadStart(() =>
                {
                    this.RunOnUiThread(() =>
                    {
                        doLoadMarketWatch();
                        progress1.Dismiss();
                    });
                })).Start();
                IsLoading = false;
            };
        }

        // This method takes user to the Add Stock Screen
        protected void doOpenAddStock()
        {
            StartActivity(typeof(AddStockActivity));
        }

        // This method takes user to the Get Stock Screen
        protected void doOpenGetStock()
        {
            StartActivity(typeof(GetStockActivity));
        }
    }
}
```

Add Stock Screen Activity Code (C#)

```
/*Add Stock Screen adds valid Stock to SQLite Database. Stocks that are added will be shown in
summary in Market Watch and detail in StockDetails Screen.*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.Graphics;
using System.Xml;

using SXViewer.Core.BusinessLayer;
using SXViewer.stockwebservice;

namespace SXViewer
{
    [Activity(Label = "Add Stock",
        ScreenOrientation = Android.Content.PM.ScreenOrientation.Portrait)]
    public class AddStockActivity : Activity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            View titleView = Window.FindViewById(Android.Resource.Id.Title);

            //update the default title
            if (titleView != null)
            {
                IViewParent parent = titleView.Parent;
                if (parent != null && (parent is View))
                {
                    View parentView = (View)parent;
                    parentView.SetBackgroundColor(Color.Rgb(28, 28, 28));
                    parentView.SetMinimumHeight(32);
                    parentView.SetMinimumHeight(32);
                }
            }

            // Set our view from the "main" layout resource
            SetContentView(Resource.Layout.AddStock);

            //EditText textStock = FindViewById<EditText>(Resource.Id.textStock);

            /** BEGIN: This block of code will provide a list of stocks while typing */
            string[] STOCKS = Resources.GetStringArray(Resource.Array.stockcodelist);
            AutoCompleteTextView textView =

FindViewById<AutoCompleteTextView>(Resource.Id.autocomplete_stock);
            var adapter = new ArrayAdapter<String>(this, Resource.Layout.list_item, STOCKS);
            textView.Adapter = adapter;
            /** END*/
            //Add Stock Button Code
            Button btnAddStock = FindViewById<Button>(Resource.Id.buttonAddStock);
            btnAddStock.Click += (sender, e) =>
            {
                if (textView.Text == string.Empty)
                {
                    //message if no text is entered in script
                    var t = Toast.MakeText(this, "Please enter Stock Script",
ToastLength.Short);
                    t.SetGravity(GravityFlags.Center, 0, 0);
                    t.Show();

                    textView.Focusable = true;
                    return;
                }

                //check script from webservice to see, if it is a valid script
                if (doCheckStockScript(textView.Text.Trim()))
                {
                    //if valid, add script to sqlite database
                    bool ret = AddStockScript(textView.Text.Trim());
                    if (!ret)
                    {
                        textView.Focusable = true;
                        return;
                    }
                    else
                    {
                        textView.Text = "";
                        textView.Focusable = true;

                        StartActivity(typeof(MarketWatchActivity));
                    }
                }
                else
                {
                    //message if no text entered is not a valid script
                    var t = Toast.MakeText(this, "You have entered invalid Stock Script.
Please
                                          enter a valid Stock Script", ToastLength.Short);
                    t.SetGravity(GravityFlags.Center, 0, 0);
                    t.Show();

                    textView.Focusable = true;
                    return;
                }
            };
        }

//This method checks the stock script, also handles errors that might occur while connecting to
webservice
        private bool doCheckStockScript(string _stockscript)
        {
            //connecting to webservice and get the xml
            string strXML = string.Empty;
            stockwebservice.StockWebservice quoteObject = null;
            try
            {
                quoteObject = new stockwebservice.StockWebservice();
                strXML = quoteObject.StockExists(_stockscript);
            }
            finally
            {
                quoteObject.Dispose();
                quoteObject = null;
            }

            //if error occurred while connecting to web service
            if (strXML.Substring(0, 5) == "error")
            {
                var t = Toast.MakeText(this, "Error connecting to web service. Please check
your
                                Internet connection...", ToastLength.Short);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();
                return false;
            }
            if (strXML.ToLower() == "exception")
            {
                var t = Toast.MakeText(this, "Service not available now. Please try after
                              sometime...", ToastLength.Short);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();
                return false;
            }
            XmlDocument doc = new XmlDocument();
            doc.LoadXml(strXML);

            XmlNodeList xnList = doc.SelectNodes("/stock/symbol");
            foreach (XmlNode xn in xnList)
            {
                if (xn["exchange"].InnerText.Trim().ToUpper() == "NA")
                    return false;
            }

            return true;
        }

        //This method adds valid stocks to database
        private bool AddStockScript(string _stockscript)
        {
            bool Ok = false;
            StockManager.Message = string.Empty;
            Ok = StockManager.SaveStock(_stockscript);
            if (StockManager.Message != string.Empty)
            {
                var t = Toast.MakeText(this, StockManager.Message, ToastLength.Short);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();
            }

            return Ok;
        }

        // This method is overridden to create a menu with three menu options
        public override bool OnCreateOptionsMenu(IMenu menu)
        {
            menu.Add("Market Watch").SetIcon(Resource.Drawable.ic_market_watch);
            menu.Add("Add Stock").SetIcon(Resource.Drawable.ic_quote_add);
            menu.Add("Get Quote").SetIcon(Resource.Drawable.ic_stock_get); ;
            return true;
        }

        // This method is overridden for adding desired functionality in the menu options
        public override bool OnOptionsItemSelected(IMenuItem item)
        {
            switch (item.TitleFormatted.ToString())
            {
                case "Market Watch":
                    doOpenMarketWatch(); break;
                case "Add Stock":
                    doOpenAddStock(); break;
                case "Get Quote":
                    doOpenGetStock(); break;
            }
            return base.OnOptionsItemSelected(item);
        }


        void MenuItemClicked(string item)
        {
            Console.WriteLine(item + " option menuitem clicked");
            var t = Toast.MakeText(this, "Options Menu '" + item + "' clicked",
ToastLength.Short);
            t.SetGravity(GravityFlags.Center, 0, 0);
            t.Show();
        }

        // This method takes user to the Market Watch Screen
        protected void doOpenMarketWatch()
        {
            StartActivity(typeof(MarketWatchActivity));
        }

        // This method takes user to the Add Stock Screen again
        protected void doOpenAddStock()
        {
        }

        // This method takes user to the Get Stock Screen
        protected void doOpenGetStock()
        {
            StartActivity(typeof(GetStockActivity));
        }
    }
}
```

Get Stock Screen Activity Code (C#)

```
/*Get Stock Screen is a customized search of a Stock. The user will enter the script and press
the "Get Stock" button and stock Quote details are displayed on screen. */

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.Graphics;

using System.Threading;
using System.Xml;

using SXViewer.Core.BusinessLayer;

namespace SXViewer
{
    [Activity(Label = "Get Stock Quote",
        ScreenOrientation = Android.Content.PM.ScreenOrientation.Portrait)]
    public class GetStockActivity : Activity
    {
        private bool IsLoading = false;
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            View titleView = Window.FindViewById(Android.Resource.Id.Title);
            //update the default title
            if (titleView != null)
            {
                IViewParent parent = titleView.Parent;
                if (parent != null && (parent is View))
                {
                    View parentView = (View)parent;
                    parentView.SetBackgroundColor(Color.Rgb(28, 28, 28));
                    parentView.SetMinimumHeight(32);
                    parentView.SetMinimumHeight(32);
                }
            }

            // Set our view from the "main" layout resource
            SetContentView(Resource.Layout.GetStock);

            //Home Button Code
            ImageView btnHome = FindViewById<ImageView>(Resource.Id.buttonHomeget);
            btnHome.Click += (sender, e) =>
            {
                StartActivity(typeof(MarketWatchActivity));
                return;
            };

            EditText textStock = FindViewById<EditText>(Resource.Id.textGetStock);
            //Get Stock Button Code
            Button btnGetStock = FindViewById<Button>(Resource.Id.buttonGetStock);
            btnGetStock.Click += (sender, e) =>
            {
                if (textStock.Text == string.Empty)
                {
                    //message if no text is entered in script
                    var t = Toast.MakeText(this, "Please enter Stock Script", 
ToastLength.Short);
                    t.SetGravity(GravityFlags.Center, 0, 0);
                    t.Show();

                    textStock.Focusable = true;
                    return;
                }

                //check script from webservice to see, if it is a valid script
                if (doCheckStockScript(textStock.Text.Trim()))
                {
                    ProgressDialog progressMain = ProgressDialog.Show(this, "", "Loading 
Quote...",
                                                                      true);
                    new Thread(new ThreadStart(() =>
                    {
                        //Thread.Sleep(4 * 1000);
                        this.RunOnUiThread(() =>
                        {
                            doLoadDetails(textStock.Text.Trim());
                            progressMain.Dismiss();
                        });
                    })).Start();
                }
                else
                {
                    //message if no text entered is not a valid script
                    var t = Toast.MakeText(this, "You have entered invalid Stock Script. 
Please 
                                          enter a valid Stock Script", ToastLength.Short);
                    t.SetGravity(GravityFlags.Center, 0, 0);
                    t.Show();

                    textStock.Focusable = true;
                    return;
                }
            };
        }

//This method checks the stock script, also handles errors that might occur while connecting to 
webservice
        private bool doCheckStockScript(string _stockscript)
        {
            //connecting to webservice and get the xml
            string strXML = string.Empty;
            stockwebservice.StockWebservice quoteObject = null;
            try
            {
                quoteObject = new stockwebservice.StockWebservice();
                strXML = quoteObject.StockExists(_stockscript);
            }
            finally
            {
                quoteObject.Dispose();
                quoteObject = null;
            }

            //if error occurred while connecting to web service
            if (strXML.Substring(0, 5) == "error")
            {
                var t = Toast.MakeText(this, "Error connecting to web service. Please check 
your
                                       Internet connection...", ToastLength.Short);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();
                return false;
            }
            if (strXML.ToLower() == "exception")
            {
                var t = Toast.MakeText(this, "Service not available now. Please try after 
                                       sometime...", ToastLength.Short);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();
                return false;
            }

            XmlDocument doc = new XmlDocument();
            doc.LoadXml(strXML);

            XmlNodeList xnList = doc.SelectNodes("/stock/symbol");
            foreach (XmlNode xn in xnList)
            {
                if (xn["exchange"].InnerText.Trim().ToUpper() == "NA")
                    return false;
            }
            return true;
        }

        //This method loads all the stock details in a tabular form
        protected void doLoadDetails(string _script)
        {
            try
            {
                if (IsLoading) return;
                IsLoading = true;

                TextView symbolCaption;
                TextView priceCaption;
                TextView changeCaption;
                TextView datetimeCaption;

                String[,] data = new String[12, 2];

                //connecting to webservice and get the xml
                string strXML = string.Empty;
                stockwebservice.StockWebservice quoteObject = null;
                try
                {
                    quoteObject = new stockwebservice.StockWebservice();
                    strXML = quoteObject.GetStockQuote(_script);
                }
                finally
                {
                    quoteObject.Dispose();
                    quoteObject = null;
                }

                //if error occurred while connecting to web service
                if (strXML.Substring(0, 5) == "error")
                {
                    var t = Toast.MakeText(this, "Error connecting to web service. Please
check your
                                           Internet connection...", ToastLength.Short);
                    t.SetGravity(GravityFlags.Center, 0, 0);
                    t.Show();
                    return;
                }
                if (strXML.ToLower() == "exception")
                {
                    var t = Toast.MakeText(this, "Service not available now. Please try after
                                           sometime...", ToastLength.Short);
                    t.SetGravity(GravityFlags.Center, 0, 0);
                    t.Show();
                    return;
                }

                LinearLayout stockdetailsLinearLayout =

FindViewById<LinearLayout>(Resource.Id.linearLayoutGetstockdetails);
                symbolCaption = FindViewById<TextView>(Resource.Id.symbolcaptiongetget);
                symbolCaption.SetTextAppearance(this, Resource.Style.boldText);

                priceCaption = FindViewById<TextView>(Resource.Id.pricecaptionget);
                priceCaption.SetTextAppearance(this, Resource.Style.boldText19);
                changeCaption = FindViewById<TextView>(Resource.Id.changecaptionget);
                changeCaption.SetTextAppearance(this, Resource.Style.boldText19);


                datetimeCaption = FindViewById<TextView>(Resource.Id.datetimecaptionget);
                datetimeCaption.SetTextAppearance(this, Resource.Style.smallText);

                //load the xml to XmlDocument
                XmlDocument doc = new XmlDocument();
                doc.LoadXml(strXML);

                XmlNodeList xnList = doc.SelectNodes("/stock/symbol");
                foreach (XmlNode xn in xnList)
                {
                    data[0, 0] = "Symbol";
                    data[0, 1] = xn["code"].InnerText.Trim();

                    data[1, 0] = "Name";
                    data[1, 1] = xn["company"].InnerText.Trim();
                    symbolCaption.Text = xn["company"].InnerText.Trim();
                    datetimeCaption.Text = DateTime.Now.ToString("dd-MM-yyyy hh:mm:ss" +
"\n");

                    priceCaption.Text = xn["last"].InnerText.Trim() + " " +
                                        xn["currency"].InnerText.Trim();
                    changeCaption.Text = xn["change"].InnerText.Trim();

                    View vLinePrice = new View(this);
                    vLinePrice.SetMinimumHeight(2);
                    vLinePrice.SetBackgroundColor(Color.Rgb(164, 164, 164));
                    stockdetailsLinearLayout.AddView(vLinePrice);

                    data[2, 0] = "Exchange";
                    data[2, 1] = xn["exchange"].InnerText.Trim();

                    data[3, 0] = "Open";
                    data[3, 1] = xn["open"].InnerText.Trim();

                    data[4, 0] = "Day's High";
                    data[4, 1] = xn["high"].InnerText.Trim();

                    data[5, 0] = "Day's Low";
                    data[5, 1] = xn["low"].InnerText.Trim();

                    data[6, 0] = "Change";
                    data[6, 1] = xn["change"].InnerText.Trim();

                    data[7, 0] = "Change %";
                    data[7, 1] = xn["changepercent"].InnerText.Trim();

                    data[8, 0] = "Volume";
                    data[8, 1] = xn["volume"].InnerText.Trim();

                    data[9, 0] = "Previous Close";
                    data[9, 1] = xn["previousclose"].InnerText.Trim();

                    data[10, 0] = "Trade Time";
                    data[10, 1] = xn["tradetime"].InnerText.Trim();

                    data[11, 0] = "Market Capital";

                    decimal marketcapital = 0;
                    if (decimal.TryParse(xn["marketcapital"].InnerText.Trim(), out
marketcapital))
                        data[11, 1] = string.Format("{0:#,0}", marketcapital);
                    else
                        data[11, 1] = xn["marketcapital"].InnerText.Trim();


                    TableLayout tableLayout =
FindViewById<TableLayout>(Resource.Id.deatlLayoutget);
                    tableLayout.RemoveAllViews();
                    tableLayout.RefreshDrawableState();

                    for (int i = 2; i < 12; i++)
                    {
      // add all information to your tablerow in such a manner that you want to display on
screen.
                        TableRow demoTableRow = new TableRow(this);
                        TextView tv_l = new TextView(this);
                        TextView tv_r = new TextView(this);

                        tv_l.SetPadding(3, 3, 3, 3);
                        tv_r.SetPadding(3, 3, 3, 3);
                        tv_r.Gravity = GravityFlags.Right;

                        tv_l.Text = data[i, 0];
                        tv_r.Text = data[i, 1];

                        demoTableRow.AddView(tv_l);
                        demoTableRow.AddView(tv_r);
                        if (i == 0)
                        {
                            tv_l.SetTextAppearance(this, Resource.Style.boldText);
                        }

                        tableLayout.AddView(demoTableRow);
                        View vLineRow = new View(this);
                        vLineRow.SetMinimumHeight(1);
                        vLineRow.SetBackgroundColor(Color.Rgb(88, 88, 88));
                        tableLayout.AddView(vLineRow);
                    }
                }
            }
            finally
            {
                IsLoading = false;
            }
        }

        // This method is overridden to create a menu with three menu options
        public override bool OnCreateOptionsMenu(IMenu menu)
        {
            menu.Add("Market Watch").SetIcon(Resource.Drawable.ic_market_watch);
            menu.Add("Add Stock").SetIcon(Resource.Drawable.ic_quote_add);
            menu.Add("Chart").SetIcon(Resource.Drawable.ic_stock_get);
            return true;
        }

        // This method is overridden for adding desired functionality in the menu options
        public override bool OnOptionsItemSelected(IMenuItem item)
        {
            switch (item.TitleFormatted.ToString())
            {
                case "Market Watch":
                    doOpenMarketWatch(); break;
                case "Add Stock":
                    doAddStockStock(); break;
                case "Chart":
                    doOpenChart(); break;
            }
            return base.OnOptionsItemSelected(item);
        }

        void MenuItemClicked(string item)
        {
            Console.WriteLine(item + " option menuitem clicked");
            var t = Toast.MakeText(this, "Options Menu '" + item + "' clicked",
ToastLength.Short);
            t.SetGravity(GravityFlags.Center, 0, 0);
            t.Show();
        }

        // This method takes user to the Market Watch Screen
        protected void doOpenMarketWatch()
        {
            StartActivity(typeof(MarketWatchActivity));
        }

        // This method takes user to the Add Stock Screen
        protected void doAddStockStock()
        {
            StartActivity(typeof(AddStockActivity));
        }

        // This method takes user to the Show Chart Screen
        protected void doOpenChart()
        {
            EditText textStock = FindViewById<EditText>(Resource.Id.textGetStock);
            if (textStock.Text.Trim() == string.Empty)
            {
                //message if no text is entered in script
                var t = Toast.MakeText(this, "Please enter Stock Script", ToastLength.Short);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();

                textStock.Focusable = true;
                return;
            }

            //get the _script from the previous activity
            var intent = new Intent();
            intent.SetClass(this, typeof(ShowChartActivity));

            //sending the script value to next activity (stock chart)
            intent.PutExtra("Script", textStock.Text.Trim());
            StartActivity(intent);
        }
    }
}
```

Show Chart Screen Activity Code (C#)

```
/*Show chart screen allows users to view charts for different Parameters, helps users to
understand the Stock Movement Trend.*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using Android.Graphics;
using System.Threading;
using System.Net;
using System.IO;
using SXViewer.stockwebservice;
using SXViewer.Core.BusinessLayer;

namespace SXViewer
{
    [Activity(Label = "Stock Chart",
        ScreenOrientation = Android.Content.PM.ScreenOrientation.Landscape)]

    public class ShowChartActivity : Activity
    {
        private bool IsLoading = false;
        private string _script = string.Empty;
        private string _scriptname=string.Empty;
        private ImageView imgChart = null;
        private TextView stockchartcaption = null;

        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            View titleView = Window.FindViewById(Android.Resource.Id.Title);
            //update the default title
            if (titleView != null)
            {
                IViewParent parent = titleView.Parent;
                if (parent != null && (parent is View))
                {
                    View parentView = (View)parent;
                    parentView.SetBackgroundColor(Color.Rgb(28, 28, 28));
                    parentView.SetMinimumHeight(32);
                    parentView.SetMinimumHeight(32);
                }
            }

            // Set our view from the "main" layout resource
            SetContentView(Resource.Layout.ShowChart);

            _script = Intent.GetStringExtra("Script");
            if (_script == string.Empty)
            {
                var t = Toast.MakeText(this, "Invalid Script...", ToastLength.Long);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();

            //if script is blank for some reason, return back to Market Watch
                StartActivity(typeof(MarketWatchActivity));
            }


            //get the script name from previous screen
            _scriptname = Intent.GetStringExtra("ScriptName");

            //set the scriptname in text view id:chartcaption defined in ChartLayout
            stockchartcaption = FindViewById<TextView>(Resource.Id.stockchartcaption);
            stockchartcaption.Text = _scriptname;

            //get all the button controls and attach click event handler
            Button btn1D = FindViewById<Button>(Resource.Id.button1D);
            Button btn5D = FindViewById<Button>(Resource.Id.button5D);
            Button btn3M = FindViewById<Button>(Resource.Id.button3M);
            Button btn6M = FindViewById<Button>(Resource.Id.button6M);
            Button btn1Y = FindViewById<Button>(Resource.Id.button1Y);
            Button btn2Y = FindViewById<Button>(Resource.Id.button2Y);
            Button btn5Y = FindViewById<Button>(Resource.Id.button5Y);

            btn1D.Click += (sender, e) => { doLoadChart(sender, "1D"); };
            btn5D.Click += (sender, e) => { doLoadChart(sender, "5D"); };
            btn3M.Click += (sender, e) => { doLoadChart(sender, "3M"); };
            btn6M.Click += (sender, e) => { doLoadChart(sender, "6M"); };
            btn1Y.Click += (sender, e) => { doLoadChart(sender, "1Y"); };
            btn2Y.Click += (sender, e) => { doLoadChart(sender, "2Y"); };
            btn5Y.Click += (sender, e) => { doLoadChart(sender, "5Y"); };

            doLoadChart(_script, "1D");//default Day 1 - Chart
            IsLoading = false;
        }

        //This method loads the chart of a stock
        private void doLoadChart(object _sender, string _option)
        {
            string str_script = string.Empty;
            Button btn = _sender as Button;
            if (btn != null)
            {
                str_script = btn.Text.Trim();
                if (str_script == string.Empty)
                {
                    var t = Toast.MakeText(this, "Invalid script, cannot load Chart",
                                           ToastLength.Short);
                    t.SetGravity(GravityFlags.Center, 0, 0);
                    t.Show();
                    StartActivity(typeof(StockDetailsActivity));
                    return;
                }
            }

            ProgressDialog progress = ProgressDialog.Show(this, "", "Loading Chart...", true);
            new Thread(new ThreadStart(() =>
            {
                this.RunOnUiThread(() =>
                {
                    if (!IsLoading)
                    {
                        doLoadChart(_script, _option);
                        IsLoading = false;
                    }
                    progress.Dismiss();
                });
            })).Start();
            IsLoading = false;
        }

//This method loads the chart of a stock with specific parameters (1 Day, 5 Days and 3 Months
etc..)
        private void doLoadChart(string _stock, string _type)
        {
            //get the image control
            imgChart = FindViewById<ImageView>(Resource.Id.imageChart);

            byte[] image_data;
            stockwebservice.StockWebservice quoteObject = null;
            try
            {
                quoteObject = new stockwebservice.StockWebservice();
                image_data = quoteObject.GetStckChart(_stock, _type);
            }
            finally
            {
                quoteObject.Dispose();
                quoteObject = null;
            }
            //convert byte array to image
            Bitmap bitmapChart = BitmapFactory.DecodeByteArray(image_data, 0,
image_data.Length);
            imgChart.SetImageBitmap(bitmapChart);
            imgChart.SetBackgroundResource(Resource.Color.transparent);

            //set the option selected (1 day, 5 days, 3 months etc)
            stockchartcaption = FindViewById<TextView>(Resource.Id.stockchartcaption);
            stockchartcaption.Text = _scriptname + " - " + _type;

            IsLoading = false;
        }

        // This method is overridden to create a menu with three menu options
        public override bool OnCreateOptionsMenu(IMenu menu)
        {
            menu.Add("Market Watch").SetIcon(Resource.Drawable.ic_stock_get); ;
            menu.Add("Stock Details").SetIcon(Resource.Drawable.ic_quote_add);
            menu.Add("Get Stock").SetIcon(Resource.Drawable.ic_stock_delete);
            return true;
        }

        // This method is overridden for adding desired functionality in the menu options
        public override bool OnOptionsItemSelected(IMenuItem item)
        {
            switch (item.TitleFormatted.ToString())
            {
                case "Market Watch":
                    doMarketWatch(); break;
                case "Stock Details":
                    doOpenStockDetails(); break;
                case "Get Stock":
                    doOpenGetStock(); break;
            }
            return base.OnOptionsItemSelected(item);
        }

        // This method takes user to the Market Watch Screen
        protected void doMarketWatch()
        {
            StartActivity(typeof(MarketWatchActivity));
        }

        // This method takes user to the Stock Details Screen
        protected void doOpenStockDetails()
        {
            StartActivity(typeof(StockDetailsActivity));
        }

        // This method takes user to the Get Stock Screen
        protected void doOpenGetStock()
        {
            StartActivity(typeof(GetStockActivity));
        }
    }
}
```

Stock Details Screen Activity Code (C#)

```
/* Stock Details screen displays the Stock Details of individual companies. It allows users to
check the complete details of stock. */

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.Graphics;

using System.Threading;
using System.Xml;
using SXViewer.stockwebservice;

namespace SXViewer
{
    [Activity(Label = "Stock Quote Details",
        ScreenOrientation = Android.Content.PM.ScreenOrientation.Portrait,
        Icon = "@drawable/icon_activity")]
    public class StockDetailsActivity : Activity
    {
        private bool IsLoading = false;
        private string _script = string.Empty;
        private string _scriptname = string.Empty;

        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            View titleView = Window.FindViewById(Android.Resource.Id.Title);

            //update the default title
            if (titleView != null)
            {
                IViewParent parent = titleView.Parent;
                if (parent != null && (parent is View))
                {
                    View parentView = (View)parent;
                    parentView.SetBackgroundColor(Color.Rgb(28, 28, 28));
                    parentView.SetMinimumHeight(32);
                    parentView.SetMinimumHeight(32);
                }
            }


            // Set our view from the "main" layout resource
            SetContentView(Resource.Layout.StockDetails);

            _script = Intent.GetStringExtra("Script");
            if (_script == string.Empty)
            {
                var t = Toast.MakeText(this, "Invalid Script...", ToastLength.Long);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();
                return;
            }

            //Refresh Button Code
            ImageView btnRefresh = FindViewById<ImageView>(Resource.Id.buttonRefresh);
            btnRefresh.Click += (sender, e) =>
            {
                if (!IsLoading)
                {
                    ProgressDialog progress = ProgressDialog.Show(this, "", "Loading
Quote...",
                                                                  true);
                    new Thread(new ThreadStart(() =>
                    {
                        this.RunOnUiThread(() =>
                        {
                            doLoadDetails();
                            progress.Dismiss();
                        });
                    })).Start();
                    IsLoading = false;
                }
            };

            //Home Button Code
            ImageView btnHome = FindViewById<ImageView>(Resource.Id.buttonHome);
            btnHome.Click += (sender, e) =>
            {
                StartActivity(typeof(MarketWatchActivity));
                return;
            };

            ProgressDialog progressMain = ProgressDialog.Show(this, "", "Loading Quote...",
true);
            new Thread(new ThreadStart(() =>
            {
                this.RunOnUiThread(() =>
                {
                    doLoadDetails();
                    progressMain.Dismiss();
                });
            })).Start();
        }

        //This method loads the details of a selected stock in a tabular form.
        protected void doLoadDetails()
        {
            try
            {
                if (IsLoading) return;
                IsLoading = true;

                TextView symbolCaption;
                TextView priceCaption;
                TextView changeCaption;
                TextView datetimeCaption;

                String[,] data = new String[12, 2];
                //connecting to webservice and get the xml
                string strXML = string.Empty;
                stockwebservice.StockWebservice quoteObject = null;
                try
                {
                    quoteObject = new stockwebservice.StockWebservice();
                    strXML = quoteObject.GetStockQuote(_script);
                }
                finally
                {
                    quoteObject.Dispose();
                    quoteObject = null;
                }

                //if error occurred while connecting to web service
                if (strXML.Substring(0, 5) == "error")
                {
                    var t = Toast.MakeText(this, "Error connecting to web service. Please 
check your 
                                      Internet connection...", ToastLength.Short);
                    t.SetGravity(GravityFlags.Center, 0, 0);
                    t.Show();
                    return;
                }
                if (strXML.ToLower() == "exception")
                {
                    var t = Toast.MakeText(this, "Service not available now. Please try after 
                                      sometime...", ToastLength.Short);
                    t.SetGravity(GravityFlags.Center, 0, 0);
                    t.Show();
                    return;
                }

                //find the controls in layout from resource
                LinearLayout stockdetailsLinearLayout = 
                           FindViewById<LinearLayout>(Resource.Id.linearLayoutstockdetails);
                symbolCaption = FindViewById<TextView>(Resource.Id.symbolcaption);
                symbolCaption.SetTextAppearance(this, Resource.Style.boldText);
                priceCaption = FindViewById<TextView>(Resource.Id.pricecaption);
                priceCaption.SetTextAppearance(this, Resource.Style.boldText19);
                changeCaption = FindViewById<TextView>(Resource.Id.changecaption);
                changeCaption.SetTextAppearance(this, Resource.Style.boldText19);
                datetimeCaption = FindViewById<TextView>(Resource.Id.datetimecaption);
                datetimeCaption.SetTextAppearance(this, Resource.Style.smallText);

                //load the xml to XmlDocument
                XmlDocument doc = new XmlDocument();
                doc.LoadXml(strXML);

                XmlNodeList xnList = doc.SelectNodes("/stock/symbol");
                foreach (XmlNode xn in xnList)
                {
                    data[0, 0] = "Symbol";
                    data[0, 1] = xn["code"].InnerText.Trim();

                    data[1, 0] = "Name";
                    data[1, 1] = xn["company"].InnerText.Trim();

             //store the company nae in _scriptname to be used in chart Activity passed through 
intent
                    _scriptname = data[1, 1].Trim();
                    symbolCaption.Text = xn["company"].InnerText.Trim();
                    datetimeCaption.Text = DateTime.Now.ToString("dd-MM-yyyy hh:mm:ss" +
"\n");

                    priceCaption.Text = xn["last"].InnerText.Trim() + " " +
                                        xn["currency"].InnerText.Trim();
                    changeCaption.Text = xn["change"].InnerText.Trim();

                    View vLinePrice = new View(this);
                    vLinePrice.SetMinimumHeight(2);
                    vLinePrice.SetBackgroundColor(Color.Rgb(164, 164, 164));
                    stockdetailsLinearLayout.AddView(vLinePrice);

                    data[2, 0] = "Exchange";
                    data[2, 1] = xn["exchange"].InnerText.Trim();

                    data[3, 0] = "Open";
                    data[3, 1] = xn["open"].InnerText.Trim();

                    data[4, 0] = "Day's High";
                    data[4, 1] = xn["high"].InnerText.Trim();

                    data[5, 0] = "Day's Low";
                    data[5, 1] = xn["low"].InnerText.Trim();

                    data[6, 0] = "Change";
                    data[6, 1] = xn["change"].InnerText.Trim();

                    data[7, 0] = "Change %";
                    data[7, 1] = xn["changepercent"].InnerText.Trim();

                    data[8, 0] = "Volume";
                    data[8, 1] = xn["volume"].InnerText.Trim();

                    data[9, 0] = "Previous Close";
                    data[9, 1] = xn["previousclose"].InnerText.Trim();

                    data[10, 0] = "Trade Time";
                    data[10, 1] = xn["tradetime"].InnerText.Trim();

                    data[11, 0] = "Market Capital";

                    decimal marketcapital = 0;
                    if (decimal.TryParse(xn["marketcapital"].InnerText.Trim(), out
marketcapital))
                        data[11, 1] = string.Format("{0:#,0}", marketcapital);
                    else
                        data[11, 1] = xn["marketcapital"].InnerText.Trim();


                    TableLayout tableLayout =
FindViewById<TableLayout>(Resource.Id.deatlLayout);
                    tableLayout.RemoveAllViews();
                    tableLayout.RefreshDrawableState();

                    for (int i = 2; i < 12; i++)
                    {
       // add all information to your tablerow in such a manner that you want to display on
screen.
                        TableRow demoTableRow = new TableRow(this);
```

```
                TextView tv_l = new TextView(this);
                TextView tv_r = new TextView(this);

                tv_l.SetPadding(3, 3, 3, 3);
                tv_r.SetPadding(3, 3, 3, 3);
                tv_r.Gravity = GravityFlags.Right;

                tv_l.Text = data[i, 0];
                tv_r.Text = data[i, 1];

                demoTableRow.AddView(tv_l);
                demoTableRow.AddView(tv_r);

                if (i == 0)
                {
                    tv_l.SetTextAppearance(this, Resource.Style.boldText);
                }


                tableLayout.AddView(demoTableRow);
                View vLineRow = new View(this);
                vLineRow.SetMinimumHeight(1);
                vLineRow.SetBackgroundColor(Color.Rgb(88, 88, 88));
                tableLayout.AddView(vLineRow);
            }
        }
    }
    finally
    {
        IsLoading = false;
    }
}

// This method is overridden to create a menu with three menu options
public override bool OnCreateOptionsMenu(IMenu menu)
{
    menu.Add("Market Watch").SetIcon(Resource.Drawable.ic_market_watch);
    menu.Add("Add Stock").SetIcon(Resource.Drawable.ic_quote_add);
    menu.Add("Chart").SetIcon(Resource.Drawable.ic_stock_get); ;
    return true;
}

// This method is overridden for adding desired functionality in the menu options
public override bool OnOptionsItemSelected(IMenuItem item)
{
    switch (item.TitleFormatted.ToString())
    {
        case "Market Watch":
            doOpenMarketWatch(); break;
        case "Add Stock":
            doOpenAddStock(); break;
        case "Chart":
            doOpenChart(); break;
    }
    return base.OnOptionsItemSelected(item);
}

void MenuItemClicked(string item)
{
    Console.WriteLine(item + " option menuitem clicked");
```

```
            var t = Toast.MakeText(this, "Options Menu '" + item + "' clicked",
ToastLength.Short);
            t.SetGravity(GravityFlags.Center, 0, 0);
            t.Show();
        }

        // This method takes user to the Market Watch Screen
        protected void doOpenMarketWatch()
        {
            StartActivity(typeof(MarketWatchActivity));
        }

        // This method takes user to the Add Stock Screen
        protected void doOpenAddStock()
        {
            StartActivity(typeof(AddStockActivity));
        }

        // This method takes user to the Show Chart Screen
        protected void doOpenChart()
        {
            //get the _script from the previous activity
            if (_script != string.Empty)
            {
                var intent = new Intent();
                intent.SetClass(this, typeof(ShowChartActivity));

                //sending the script and scriptname value to next activity (stock chart)
                intent.PutExtra("Script", _script);
                intent.PutExtra("ScriptName", _scriptname);
                StartActivity(intent);
            }
            else
            {
                var t = Toast.MakeText(this, "Chart cannot be shown for this script",
                                       ToastLength.Short);
                t.SetGravity(GravityFlags.Center, 0, 0);
                t.Show();
            }
        }
    }
}
```

Stock Activity Class

```
/*Stock Class is used for creating a Data Repository. This class has two constructors and also
getters and setters methods.*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace SXViewer.Core.BusinessLayer
{
    public class Stock : Java.Lang.Object
    {
        public long Id { get; set; }
        public string StockName { get; set; }

        public Stock()
        {
            Id = -1;
            StockName = string.Empty;
        }

        public Stock(long id, string stockName)
        {
            Id = id;
            StockName = stockName;
        }
        public override string ToString()
        {
            return StockName.ToString();
        }
    }

}
```

Stock Manager Activity Class

```
/*Stock Manager class is used for all Database Operations. This class has four methods, which
have the same method signatures as methods in database. Also this class has one constructor.*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using SXViewer.Core.BusinessLayer;
using SXViewer.Core.DataLayer;

namespace SXViewer.Core.BusinessLayer
{
    public static class StockManager
    {
        public static string Message { get; set; }
        static StockManager()
        {
        }

//This method call the method in database to check whether stock exists in database, and returns True or
False
        public static bool IsStockExists(string _stockname)
        {
            return StockDatabase.IsStockExists(_stockname);
        }

//This method calls the method in database to display stocks from Stock Table
        public static IEnumerable<Stock> GetStocks()
        {
            return StockDatabase.GetStocks();
        }

//This method calls the method in database to insert stock in the stock table
        public static bool SaveStock(string _stockname)
        {
            return StockDatabase.SaveStock(_stockname);
        }

//This method calls the method in database to delete all the stocks from the stock table
        public static bool DeleteAllStocks()
        {
            return StockDatabase.DeleteAllStocks();
        }

    }
}
```

Stock Database

```
/*Stock Database is used to store saved stocks in Market Watch Screen. This database has one
single Table with subsequent methods to Fetch, Add and Delete stocks that we are used in
Market Watch - Favorite List. */

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using Mono.Data.Sqlite;
using System.IO;
using SXViewer.Core.BusinessLayer;

namespace SXViewer.Core.DataLayer
{
    public class StockDatabase
    {
        private static string db_file = "stockdata.db3";
        private string stockName = string.Empty;
        private static string sMessage;

        public string StockName
        {
            get { return stockName; }
            set { stockName = value; }
        }

        //This method creates database or returns database
        private static SqliteConnection GetConnection()
        {
            var dbPath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Personal),
                                     db_file);
            bool exists = File.Exists(dbPath);

            if (!exists)
                SqliteConnection.CreateFile(dbPath);

            var conn = new SqliteConnection("Data Source=" + dbPath);

            if (!exists)
                CreateDatabase(conn);

            return conn;
        }

        //This method creates Stock table
        private static void CreateDatabase(SqliteConnection connection)
        {
            var sql = "CREATE TABLE STOCKTABLE (Id INTEGER PRIMARY KEY AUTOINCREMENT,
StockName
                       VARCHAR);";
            connection.Open();

            using (var cmd = connection.CreateCommand())
            {
                cmd.CommandText = sql;
                cmd.ExecuteNonQuery();
            }
            connection.Close();
        }

        //This method displays stocks from Stock Table
        public static IEnumerable<Stock> GetStocks()
        {
            try
            {
                var sql = "SELECT * FROM STOCKTABLE ORDER BY ID;";

                using (var conn = GetConnection())
                {
                    conn.Open();

                    using (var cmd = conn.CreateCommand())
                    {
                        cmd.CommandText = sql;

                        using (var reader = cmd.ExecuteReader())
                        {
                            while (reader.Read())
                                yield return new Stock(reader.GetInt32(0),
reader.GetString(1));
                        }
                    }
                }
            }
            finally
            {
                StockManager.Message = sMessage;
            }
        }

        //This method checks whether stock exists in the database, and returns True or False
        public static bool IsStockExists(string _stockname)
        {
            bool Ok = false;
            var sql = string.Format("SELECT * FROM STOCKTABLE WHERE STOCKNAME='{0}';",
_stockname);

            try
            {
                using (var conn = GetConnection())
                {
                    conn.Open();

                    using (var cmd = conn.CreateCommand())
                    {
                        cmd.CommandText = sql;

                        using (var reader = cmd.ExecuteReader())
                        {
                            while (reader.Read())
```

```
                                Ok = true;
                            }
                        }
                    }
                }
                finally
                {
                    StockManager.Message = sMessage;
                }
                return Ok;
            }

            //This method inserts stock in the stock table
            public static bool SaveStock(string _stockname)
            {
                try
                {
                    bool Ok = IsStockExists(_stockname.Trim().ToUpper());
                    if (Ok)
                    {
                        sMessage = string.Format("Stock Script '{0}' is already added.",
        _stockname);
                        return false;
                    }
                    using (var conn = GetConnection())
                    {
                        conn.Open();
                        using (var cmd = conn.CreateCommand())
                        {
                            try
                            {
                                // Do an insert
                                cmd.CommandText = "INSERT INTO STOCKTABLE (StockName) VALUES
                                                  (@StockName);";
                                cmd.Parameters.AddWithValue("@StockName", _stockname.ToUpper());
                                cmd.ExecuteNonQuery();

                                sMessage = string.Format("Stock Script '{0}' is added
        successfully.",
                                                       _stockname.ToUpper());
                                return true;
                            }
                            catch (SqliteException ex)
                            {
                                sMessage = ex.Message;
                                return false;
                            }
                        }
                    }
                }
                finally
                {
                    StockManager.Message = sMessage;
                }
            }
            //This method deletes all the stocks from the stock table
            public static bool DeleteAllStocks()
            {
                try
                {
                    using (var conn = GetConnection())
                    {
                        conn.Open();

                        using (var cmd = conn.CreateCommand())
                        {
                            try
                            {
                                // Do an insert
                                cmd.CommandText = "DELETE FROM STOCKTABLE;";
                                cmd.ExecuteNonQuery();

                                sMessage = "All Stocks are deleted successfully...\nTo view the
    stocks
                                            in Market Watch, you need to add your custom stock";
                                return true;
                            }
                            catch (SqliteException ex)
                            {
                                sMessage = ex.Message;
                                return false;
                            }
                        }
                    }
                }
                finally
                {
                    StockManager.Message = sMessage;
                }
            }
        }
    }
```

Stock Quote Webservice

```
/* Although great care has gone into developing this finance web service using finance yahoo
API, This webservice takes data from Yahoo Finance Webservice. This webserive has three very
powerful methods; StockExists, GetStockQuote and GetStckChart */

using System;
using System.Web;
using System.Web.Services;
using System.Net;
using System.Text;
using System.IO;
using System.Xml;


[WebService(Namespace = "http://v900u039rux.maximumasp.com", Description = "Get Stock Details
- All Exchange", Name = "Stock Webservice")]
public class StockQuoteService : System.Web.Services.WebService
{
    public StockQuoteService()
    {
        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

/*This web method sends multiple stock scripts in a comma separated string and pass it as a
parameter. This will return the "Exchange" of the Stock. If a stock script is Invalid, the XML
will return "NA" (Not Applicable).*/

    [WebMethod]
    [System.Web.Script.Services.ScriptMethod(ResponseFormat =
                                     System.Web.Script.Services.ResponseFormat.Xml)]
    public string StockExists(string symbol)
    {
        if (symbol == "") return "";
        string[] arrStocks = symbol.Split(',');

        string uri = "http://download.finance.yahoo.com/d/quotes.csv?s=";

        //build the querystring value for "s"
        string sParam = string.Empty;
        foreach (string sStock in arrStocks)
        {
            if (sStock.Trim() != string.Empty)
            {
                sParam += sStock.Trim() + "+";
            }
        }
        if (sParam.Length > 0) sParam = sParam.TrimEnd('+');

        //build the querystring value for "f"
        string sOptions = string.Empty;

        //Symbol - s
        //Stock Exchange - x
        sOptions = "&f=sx";


        uri = uri + sParam + sOptions;
        string _ret = string.Empty;
        HttpWebRequest stockHttpWebRequest = null;
        StreamReader responseStream = null;

        try
        {
            stockHttpWebRequest = (HttpWebRequest)WebRequest.Create(uri);
            stockHttpWebRequest.MaximumAutomaticRedirections = 1;
            stockHttpWebRequest.AllowAutoRedirect = true;

            HttpWebResponse webresp = (HttpWebResponse)stockHttpWebRequest.GetResponse();

            // get the  response from the server.
            responseStream = new StreamReader(webresp.GetResponseStream(), Encoding.ASCII);

            //creating xml
            StringBuilder _xml = new StringBuilder();
            _xml.Append("<stock>");

            string strLine = responseStream.ReadLine();
            while (strLine != null)
            {
                string data = strLine.Replace("\"", "");
                string[] arrData = data.ToString().Split(',');

                _xml.Append("<symbol>");
                _xml.Append(string.Format("<code>{0}</code>", arrData[0]));

                if (arrData[1] == "N/A")
                    _xml.Append("<exchange>NA</exchange>");
                else
                    _xml.Append(string.Format("<exchange>{0}</exchange>", arrData[1]));
                _xml.Append("</symbol>");
                strLine = responseStream.ReadLine();
            }
            _xml.Append("</stock>");
            _ret = _xml.ToString();
            _xml = null;
        }
        catch (Exception)
        {
            return "exception";
        }
        finally
        {
            if (responseStream != null) { responseStream.Close(); responseStream.Dispose(); }
            stockHttpWebRequest = null;
        }
        return _ret;
    }


/*This web method sends multiple stocks in a comma separated string and pass it as a
parameter.  This will return the XML of all the Stocks*/

    [WebMethod]
    [System.Web.Script.Services.ScriptMethod(ResponseFormat =
System.Web.Script.Services.ResponseFormat.Xml)]
    public string GetStockQuote(string symbol)
    {
        if (symbol == "") return "";
        string[] arrStocks = symbol.Split(',');
```

```
string uri = "http://download.finance.yahoo.com/d/quotes.csv?s=";

//build the querystring value for "s"
string sParam = string.Empty;
foreach (string sStock in arrStocks)
{
    if (sStock.Trim() != string.Empty)
    {
        sParam += sStock.Trim() + "+";
    }
}
if (sParam.Length > 0) sParam = sParam.TrimEnd('+');

//build the querystring value for "f"
string sOptions = string.Empty;

//Symbol - s
//Stock Exchange - x
//Name of the company - n
//Last Price- l1
//day's high - h
//day's low - g
//open - o
//previous close - p
//currency -
//change & change percent - c
//Last Volume - v
//Market Capitalization - j1

sOptions = "&f=sxnl1hgopcvj1";

uri = uri + sParam + sOptions;
string _ret = string.Empty;
HttpWebRequest stockHttpWebRequest = null;
StreamReader responseStream = null;

try
{
    stockHttpWebRequest = (HttpWebRequest)WebRequest.Create(uri);
    stockHttpWebRequest.MaximumAutomaticRedirections = 1;
    stockHttpWebRequest.AllowAutoRedirect = true;

    HttpWebResponse webresp = (HttpWebResponse)stockHttpWebRequest.GetResponse();

    // get the  response from the server.
    responseStream = new StreamReader(webresp.GetResponseStream(), Encoding.ASCII);

    //creating xml
    StringBuilder _xml = new StringBuilder();
    _xml.Append("<stock>");

    string strLine = responseStream.ReadLine();
    while (strLine != null)
    {
        string data = strLine.Replace("\"", "");
        string[] arrData = data.ToString().Split(',');

        _xml.Append("<symbol>");
        _xml.Append(string.Format("<code>{0}</code>", arrData[0]));
```

```
                if (arrData[1] == "N/A")
                {
                    _xml.Append("<exchange>NA</exchange>");
                }
                else
                {
                    _xml.Append(string.Format("<exchange>{0}</exchange>", arrData[1]));
                    _xml.Append(string.Format("<company>{0}</company>", arrData[2]));
                    _xml.Append(string.Format("<currency>{0}</currency>", ""));
                    _xml.Append(string.Format("<last>{0}</last>", arrData[3]));
                    _xml.Append(string.Format("<high>{0}</high>", arrData[4]));
                    _xml.Append(string.Format("<low>{0}</low>", arrData[5]));
                    _xml.Append(string.Format("<open>{0}</open>", arrData[6]));
                    _xml.Append(string.Format("<previousclose>{0}</previousclose>",
arrData[7]));

                    string sChange = arrData[8];
                    if (sChange.Length > 0)
                    {
                        string[] arrChange = sChange.Split(' ');
                        if (arrChange.Length >= 2)
                        {
                            _xml.Append(string.Format("<change>{0}</change>", arrChange[0]));
                            _xml.Append(string.Format("<changepercent>{0}</changepercent>",
arrChange[2]));
                        }
                        else
                        {
                            _xml.Append(string.Format("<change>{0}</change>", sChange));
                            _xml.Append(string.Format("<changepercent>{0}</changepercent>",
"NA"));
                        }
                    }
                    _xml.Append(string.Format("<volume>{0}</volume>", arrData[9]));
                    _xml.Append(string.Format("<marketcapital>{0}</marketcapital>",
arrData[10]));
                    _xml.Append(string.Format("<tradetime>{0}</tradetime>", "NA"));
                }
                _xml.Append("</symbol>");


                strLine = responseStream.ReadLine();
            }
            _xml.Append("</stock>");
            _ret = _xml.ToString();
            _xml = null;
        }
        catch (Exception)
        {
            return "exception";
        }
        finally
        {
            if (responseStream != null) { responseStream.Close(); responseStream.Dispose(); }
            stockHttpWebRequest = null;
        }
        return _ret;
    }
```

```
/*This web method takes stock symbol and chart parameter, and shows the stock chart according
to the chart parameter.*/

    [WebMethod]
    [System.Web.Script.Services.ScriptMethod(ResponseFormat =
System.Web.Script.Services.ResponseFormat.Xml)]
    public byte[] GetStckChart(string symbol, string chartparam)
    {
        if (symbol == "") return null;
        string uri = "http://ichart.finance.yahoo.com";
        string _ret = string.Empty;

        //get a new gui for version
        string str_guid_ver = Guid.NewGuid().ToString();

        switch (chartparam)
        {
            case "1D":
                uri += string.Format("/b?s={0}", symbol);
                break;
            case "5D":
                uri += string.Format("/w?s={0}", symbol);
                break;
            case "3M":
                uri += string.Format("/c/3m/{0}?version={1}", symbol, str_guid_ver);
                break;
            case "6M":
                uri += string.Format("/c/6m/{0}?version={1}", symbol, str_guid_ver);
                break;
            case "1Y":
                uri += string.Format("/c/1y/{0}?version={1}", symbol, str_guid_ver);
                break;
            case "2Y":
                uri += string.Format("/c/2y/{0}?version={1}", symbol, str_guid_ver);
                break;
            case "5Y":
                uri += string.Format("/c/5y/{0}?version={1}", symbol, str_guid_ver);
                break;
            default:
                uri = "NA";
                break;
        }

        if (uri == "NA")
        {
            return null;
        }
        HttpWebRequest myHttpWebRequest = null;
        MemoryStream memoryStream = null;
        try
        {
            myHttpWebRequest = (HttpWebRequest)WebRequest.Create(uri);
            myHttpWebRequest.MaximumAutomaticRedirections = 1;
            myHttpWebRequest.AllowAutoRedirect = true;

            memoryStream = new MemoryStream(0x10000);

            using (Stream responseStream = myHttpWebRequest.GetResponse().GetResponseStream())
            {
                byte[] buffer = new byte[0x1000];
                int bytes;
                while ((bytes = responseStream.Read(buffer, 0, buffer.Length)) > 0)
                {
                    memoryStream.Write(buffer, 0, bytes);
                }
            }
            byte[] response = memoryStream.ToArray();
            return response;
        }
        catch (Exception)
        {
            return null;
        }
        finally
        {
            memoryStream.Close();
            memoryStream.Dispose();
            myHttpWebRequest = null;
        }
    }
}
```

Splash Screen AXML Code (Auto-generated code)

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/linearLayout1"
    android:minWidth="25px"
    android:minHeight="25px">
    <Style
        name="Theme.Splash"
        parent="android:Theme">
        <item
            name="android:windowBackground" />
        <item
            name="android:windowNoTitle" />
    </Style>
</LinearLayout>
```

Market Watch Screen AXML Code (Auto-generated code)

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/deep_gray"
    android:id="@+id/linearLayoutstockdetails">
    <RelativeLayout
        android:layout_width="fill_parent"
        android:background="#000000"
        android:layout_height="30dp"
        android:paddingTop="3dip">
        <TextView
            android:id="@+id/watchcaption"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_centerVertical="true"
            android:textColor="@android:color/white"
            android:background="#000000"
            android:layout_marginLeft="0px"
            android:textSize="17dip"
            android:textStyle="bold"
            android:text="My Watchlist"
            android:paddingTop="2px" />
        <ImageView
            android:id="@+id/watchRefresh"
            android:layout_width="50dip"
            android:layout_height="50dip"
            android:layout_alignParentRight="true"
            android:paddingRight="1dip"
            android:scaleType="centerInside"
            android:clickable="true"
            android:src="@drawable/refresh_icon" />
    </RelativeLayout>
    <View
        android:id="@+id/separatorwatch"
        android:background="#848484"
        android:layout_width="fill_parent"
        android:layout_height="1dip"
        android:layout_centerVertical="true"
        android:layout_alignParentTop="true" />
    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TableLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:stretchColumns="1"
            android:id="@+id/deatlWatchLayout" />
    </ScrollView>
</LinearLayout>
```

Add Stock Screen AXML Code (Auto-generated code)

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/linearLayout1">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="50.0dp"
        android:text="Enter the full Stock Symbol for your exchange. Valid Stock will be added only."
        android:layout_marginTop="36px"
        android:layout_marginBottom="5.3dp"
        android:layout_marginLeft="8px"
        android:textSize="15sp"
        android:layout_marginRight="8px" />
    <AutoCompleteTextView
        android:layout_width="fill_parent"
        android:layout_height="30.7dp"
        android:id="@+id/autocomplete_stock"
        android:textColor="@android:color/black"
        android:hint="Enter Stock Symbol"
        android:layout_marginLeft="5.3dp"
        android:layout_marginBottom="7.3dp"
        android:layout_marginRight="80px"
        android:padding="4px"
        android:textColorHint="#323232"
        android:textSize="15sp"
        android:layout_marginTop="0.0dp" />
    <Button
        android:id="@+id/buttonAddStock"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add Stock..."
        android:layout_marginLeft="8px" />
</LinearLayout>
```

Get Stock Screen AXML Code (Auto-generated code)

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:icon="@drawable/refresh_icon"
    android:background="@color/deep_gray"
    android:id="@+id/linearLayoutGetstockdetails">
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <EditText
            android:layout_width="178dp"
            android:layout_height="42dp"
            android:id="@+id/textGetStock"
            android:textColor="@android:color/black"
            android:layout_marginLeft="6px"
            android:layout_marginBottom="12px"
            android:textSize="15sp"
            android:hint="Enter Stock Symbol"
            android:textColorHint="#323232" />
        <Button
            android:id="@+id/buttonGetStock"
            android:layout_width="110dp"
            android:layout_height="42dp"
            android:text="Get Stock..."
            android:layout_marginLeft="8px"
            android:textSize="15sp" />
    </LinearLayout>
    <RelativeLayout
        android:layout_width="fill_parent"
        android:background="@color/medium_gray"
        android:layout_height="30dp"
        android:paddingTop="3dip">
        <TextView
            android:id="@+id/symbolcaptiongetget"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_centerVertical="true"
            android:layout_alignParentLeft="true"
            android:textColor="@android:color/white"
            android:background="@color/medium_gray"
            android:layout_marginLeft="0px"
            android:layout_marginRight="8dp"
            android:paddingTop="2px" />
        <ImageView
            android:id="@+id/buttonHomeget"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_margin="24dip"
            android:layout_alignParentRight="true"
            android:layout_centerVertical="true"
            android:clickable="true"
            android:src="@drawable/home_icon"
            android:layout_marginLeft="2dp"
            android:layout_marginRight="10.0dp" />
    </RelativeLayout>
    <TextView
        android:id="@+id/datetimecaptionget"
        android:layout_width="fill_parent"
        android:layout_height="17dp"
        android:textColor="@android:color/white"
        android:background="@color/medium_gray"
        android:layout_marginLeft="0px"
        android:paddingBottom="4px" />
    <View
        android:id="@+id/separatorget"
        android:background="#848484"
        android:layout_width="fill_parent"
        android:layout_height="1dip"
        android:layout_centerVertical="true"
        android:layout_alignParentTop="true" />
    <RelativeLayout
        android:layout_width="fill_parent"
        android:background="#000000"
        android:layout_height="34dp"
        android:paddingTop="3dip">
        <TextView
            android:id="@+id/pricecaptionget"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_centerVertical="true"
            android:textColor="@android:color/white"
            android:background="#000000"
            android:layout_marginLeft="0px"
            android:paddingTop="1px"
            android:paddingBottom="1px" />
        <TextView
            android:id="@+id/changecaptionget"
            android:layout_width="wrap_content"
            android:layout_alignParentRight="true"
            android:paddingRight="1dip"
            android:scaleType="centerInside"
            android:layout_height="wrap_content"
            android:textColor="@android:color/white"
            android:background="#000000"
            android:layout_marginLeft="0px"
            android:paddingTop="1px"
            android:paddingBottom="1px" />
    </RelativeLayout>
    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TableLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:stretchColumns="1"
            android:id="@+id/deatlLayoutget" />
    </ScrollView>
</LinearLayout>
```

Show Chart Screen AXML Code (Auto-generated code)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:minWidth="25px"
    android:minHeight="25px">
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="28dp"
        android:id="@+id/linearLayoutstockdetails"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp">
        <TextView
            android:id="@+id/stockchartcaption"
            android:layout_width="206dp"
            android:layout_height="fill_parent"
            android:layout_alignParentLeft="true"
            android:layout_centerVertical="true"
            android:textColor="@android:color/white"
            android:background="#000000"
            android:layout_marginLeft="0px"
            android:textSize="14dp"
            android:textStyle="bold"
            android:paddingTop="2dp"
             android:layout_gravity="center_horizontal" />
        <Button
            android:text="1D"
            android:layout_width="38.7dp"
            android:layout_height="18dp"
            android:textSize="10dp"
            android:id="@+id/button1D"
            android:textColor="#ffffff"
            android:padding="3dp"
            android:background="#1C1C1C"
            android:layout_marginTop="1.4dp"
            android:layout_marginBottom="4dp"
            android:layout_marginRight="2dp"
            android:layout_marginLeft="6dp" />
        <Button
            android:text="5D"
            android:layout_width="38.0dp"
            android:layout_height="18dp"
            android:textSize="10dp"
            android:textColor="#ffffff"
            android:padding="3dp"
            android:background="#1C1C1C"
            android:layout_marginTop="1.3dp"
            android:layout_marginBottom="4dp"
            android:layout_marginRight="2dp"
            android:id="@+id/button5D" />
        <Button
            android:text="3M"
            android:layout_width="40.0dp"
            android:layout_height="18dp"
            android:textSize="10dp"
            android:textColor="#ffffff"
            android:padding="3dp"
            android:background="#1C1C1C"
            android:layout_marginTop="1.4dp"
            android:layout_marginBottom="4dp"
            android:layout_marginRight="2dp"
            android:id="@+id/button3M" />
        <Button
            android:text="6M"
            android:layout_width="39.3dp"
            android:layout_height="18dp"
            android:textSize="10dp"
            android:textColor="#ffffff"
            android:padding="3dp"
            android:background="#1C1C1C"
            android:layout_marginTop="1.3dp"
            android:layout_marginBottom="4dp"
            android:layout_marginRight="2dp"
            android:id="@+id/button6M" />
        <Button
            android:text="1Y"
            android:layout_width="38.7dp"
            android:layout_height="18dp"
            android:textSize="10dp"
            android:textColor="#ffffff"
            android:padding="3dp"
            android:background="#1C1C1C"
            android:layout_marginTop="1.3dp"
            android:layout_marginBottom="4dp"
            android:layout_marginRight="2dp"
            android:id="@+id/button1Y" />
        <Button
            android:text="2Y"
            android:layout_width="39.3dp"
            android:layout_height="18dp"
            android:textSize="10dp"
            android:textColor="#ffffff"
            android:padding="3dp"
            android:background="#1C1C1C"
            android:layout_marginTop="1.3dp"
            android:layout_marginBottom="4dp"
            android:layout_marginRight="2dp"
            android:id="@+id/button2Y" />
        <Button
            android:text="5Y"
            android:layout_width="38.7dp"
            android:layout_height="18dp"
            android:textSize="10dp"
            android:textColor="#ffffff"
            android:padding="3dp"
            android:background="#1C1C1C"
            android:layout_marginTop="1.3dp"
            android:layout_marginBottom="4dp"
            android:layout_marginRight="2.6dp"
            android:id="@+id/button5Y" />
    </LinearLayout>
    <ImageView
        android:id="@+id/imageChart"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
```

```xml
        android:clickable="true"
        android:layout_marginTop="2dp" />
</LinearLayout>
```

Stock Details Screen AXML Code (Auto-generated code)

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:icon="@drawable/refresh_icon"
    android:background="@color/deep_gray"
    android:id="@+id/linearLayoutstockdetails">
    <RelativeLayout
        android:layout_width="fill_parent"
        android:background="@color/medium_gray"
        android:layout_height="30dp"
        android:paddingTop="3dip">
        <TextView
            android:id="@+id/symbolcaption"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_centerVertical="true"
            android:layout_alignParentLeft="true"
            android:textColor="@android:color/white"
            android:background="@color/medium_gray"
            android:layout_marginLeft="0px"
            android:paddingTop="2px" />
        <ImageView
            android:id="@+id/buttonRefresh"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_centerVertical="true"
            android:layout_alignParentRight="true"
            android:lines="1"
            android:clickable="true"
            android:src="@drawable/refresh_icon" />
        <ImageView
            android:id="@+id/buttonHome"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_margin="24dip"
            android:layout_alignParentRight="true"
            android:layout_centerVertical="true"
            android:clickable="true"
            android:src="@drawable/home_icon"
            android:layout_marginLeft="3.3dp" />
    </RelativeLayout>
    <TextView
        android:id="@+id/datetimecaption"
        android:layout_width="fill_parent"
        android:layout_height="17dp"
        android:textColor="@android:color/white"
        android:background="@color/medium_gray"
        android:layout_marginLeft="0px"
        android:paddingBottom="4px" />
    <View
        android:id="@+id/separator"
        android:background="#848484"
        android:layout_width="fill_parent"
        android:layout_height="1dip"
        android:layout_centerVertical="true"
        android:layout_alignParentTop="true" />
    <RelativeLayout
        android:layout_width="fill_parent"
        android:background="#000000"
        android:layout_height="34dp"
        android:paddingTop="3dip">
        <TextView
            android:id="@+id/pricecaption"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_centerVertical="true"
            android:textColor="@android:color/white"
            android:background="#000000"
            android:layout_marginLeft="0px"
            android:paddingTop="1px"
            android:paddingBottom="1px" />
        <TextView
            android:id="@+id/changecaption"
            android:layout_width="wrap_content"
            android:layout_alignParentRight="true"
            android:paddingRight="1dip"
            android:scaleType="centerInside"
            android:layout_height="wrap_content"
            android:textColor="@android:color/white"
            android:background="#000000"
            android:layout_marginLeft="0px"
            android:paddingTop="1px"
            android:paddingBottom="1px" />
    </RelativeLayout>
    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TableLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:stretchColumns="1"
            android:id="@+id/deatlLayout" />
    </ScrollView>
</LinearLayout>
```

Some companies' stock symbols are mentioned in the below table:

Stock Exchange	Stock Symbol	Company Name
NASDAQ	**ABAX**	Abaxis
NASDAQ	**DELL**	Dell Inc.
NASDAQ	**EBAY**	Ebay Inc.
NASDAQ	**GOOG**	Google Inc.
NASDAQ	**GOLD**	Randgold Resources
NASDAQ	**INTC**	Intel Corp.
NASDAQ	**KRFT**	Kraft Foods
NASDAQ	**FB**	Facebook Inc
NYSE	**SNE**	Sony Corporation
PCX	**USO**	United States Oil
NYSE	**NKE**	Nike Inc.
HKSE	**0592.HK**	Bossini International
NYSE	**HOG**	Harley-Davidson, Inc.
OTC Markets	**TOSBF**	Toshiba Corp.
NYSE	**JOF**	Japan Smaller Cap. Inc.
NYSE	**NYT**	The New York Times Company
NASDAQ	**ASIA**	AsiaInfo-Linkage, Inc.
NYSE	**ARSD**	Arabian American Company
OTC Markets	**NHSH**	NHS Health Solutions, Inc.
OTC Markets	**SCHSQ**	School Specialty Inc.
HKSE	**^HSCE**	HANG SENG CHINA ENTERPRISES
Taiwan	**8215.TW**	BENQ MATERIALS COR
OTC Markets	**ICNB**	Iconic Brands, Inc.
EUX	**BMWF.EX**	BMW
NYSE	**REV**	Revlon, Inc.
OTC Markets	**CRRFY**	Carrefour SA
VTX	**SREN.VX**	SWISS RE N
NSE	**SIEMENS.NS**	Siemens Limited
ASX	**MLC.AX**	MOTHERCARE FPO
NYSE	**NOK**	Nokia Corporation
Taiwan	**2498.TW**	HTC Corporation
LSE	**DOM.L**	Domino's Pizza Group plc
NYSE	**AVP**	Avon Products Inc.
LSE	**BRBY.L**	Burberry Group plc
NASDAQ	**ADBE**	Adobe Systems Inc.
NasdaqCM	**PZZI**	Pizza Inn Holdings, Inc.
NYSE	**GES**	Guess' Inc.

OTC Markets	**BOSSY**	Hugo Boss AG
NASDAQ	**AMZN**	Amazon.com Inc.
NYSE	**FDX**	FedEx Corporation
NYSE	**KF**	The Korea Fund Inc.
NYSE	**RHT**	Red Hat, Inc.
OTC Markets	**HTHIY**	Hitachi Ltd.
NYSE	**XRX**	Xerox Corporation
NYSE	**MAC**	The Macerich Company
NASDAQ	**AHGP**	Alliance Holdings GP