

TABLE OF CONTENTS

INTRODUCTION AND OBJECTIVE.....	6
INTRODUCTION	6
OBJECTIVE.....	7
SYSTEM ANALYSIS	8
IDENTIFICATION OF NEED.....	8
PRELIMINARY INVESTIGATION.....	8
FEASIBILITY STUDY	9
PROJECT PLANNING	9
TRACKING GANTT	9
PROJECT SCHEDULING	9
PERT CHART (NETWORK DIAGRAM)	9
GANTT CHART	10
SOFTWARE REQUIREMENT AND SPECIFICATION	10
FUNCTIONAL REQUIREMENTS	10
non FUNCTIONAL REQUIREMENTS.....	16
SOFTWARE ENGINEERING PARADIGM APPLIED	16
DATA MODELS	18
DATA FLOW DIAGRAM (DFD)	18
CONTEXT DIAGRAM	18
Sequence diagrams.....	22
Activity Diagrams	23
ENTITY RELATIONSHIP MODEL	24
SYSTEM DESIGN	25
MODULARISATION DETAILS.....	25
SHOPPING mall ui	25
SHOPPING Mall data	26

SHOPPING Mall db	26
DATA INTEGRITY AND CONSTRAINTS	26
Entity integrity	26
Referential Integrity	26
Domain Integrity	26
User Defined Integrity	27
DATABASE AND TABLE DESIGN	27
USER INTERFACE DESIGN	28
Desktop Application User Interface	28
Main window	28
Login window	29
After login	29
Add Products	30
View Products Details	30
ADD SHOPS	31
VIEW SHOPS DETAILS	31
FLOOR DETAILS VIEW	32
ADD CONTACT US	32
CONTACT US DETAILS VIEW	33
INSERT SEARCH	33
VIEW SEARCH VALUE	34
WITHOUT VALUE ERROR MESSAGE	34
Mobile App Interface	34
TEST CASES	37
UNIT TEST CASES	37
System Test Cases	51
CODING	57
COMPLETE PROJECT CODING	57
Desktop application Coding:	57

Mobile Application Coding	151
COMMENTS AND DESCRIPTION OF CODING SEGMENTS	156
Standardization of the coding.....	157
Code Efficiency	159
Error handling	159
Validation checks	160
TESTING	163
TESTING TECHNIQUES AND TESTING STRATEGIES USED.....	163
Unit Testing:.....	163
Smoke Testing:.....	163
Functional Testing:	164
Regression Testing:	164
Database & Data Integrity Testing.....	165
User Interface Testing:	165
Performance Profiling:.....	165
Load Testing:.....	166
Stress Testing:.....	167
Volume Testing:	167
Security & Access Control Testing:	167
Failover & Recovery Testing:	167
Configuration Testing:	168
Installation/Deploy & Back out Testing:	168
Post Production Testing:.....	168
Data Migration Testing:	169
TESTING PLAN USED	169
Creation of Test Plan	169
Testing Scope and Objectives	170

Testing Methodology	170
Features and Functions to Test	171
Risk Factors	171
Testing Schedule	171
TESTING REPORTS	171
Test reports for Unit Test Cases and System Test Cases	171
Test reports for Unit Test Cases	172
Test reports for System Test Cases	174
DEBUGGING AND CODE IMPROVEMENT	176
Create a Sample with the Debug Class	177
Using the Trace Class	178
Verify That It Works	179
Complete Code Listing	180
Troubleshoot	181
SYSTEM SECURITY MEASURES	182
DATABASE / DATA SECURITY	182
CREATION OF USER PROFILES AND ACCESS RIGHTS	182
COST ESTIMATION	182
COST ESTIMATION MODEL	182
Estimation of development effort	183
Estimation of development time	183
FUTURE SCOPE AND FURTHER ENHANCEMENT	185
BIBLIOGRAPHY	185
Websites	185
Books	187
APPENDICES	187
IDE:	187

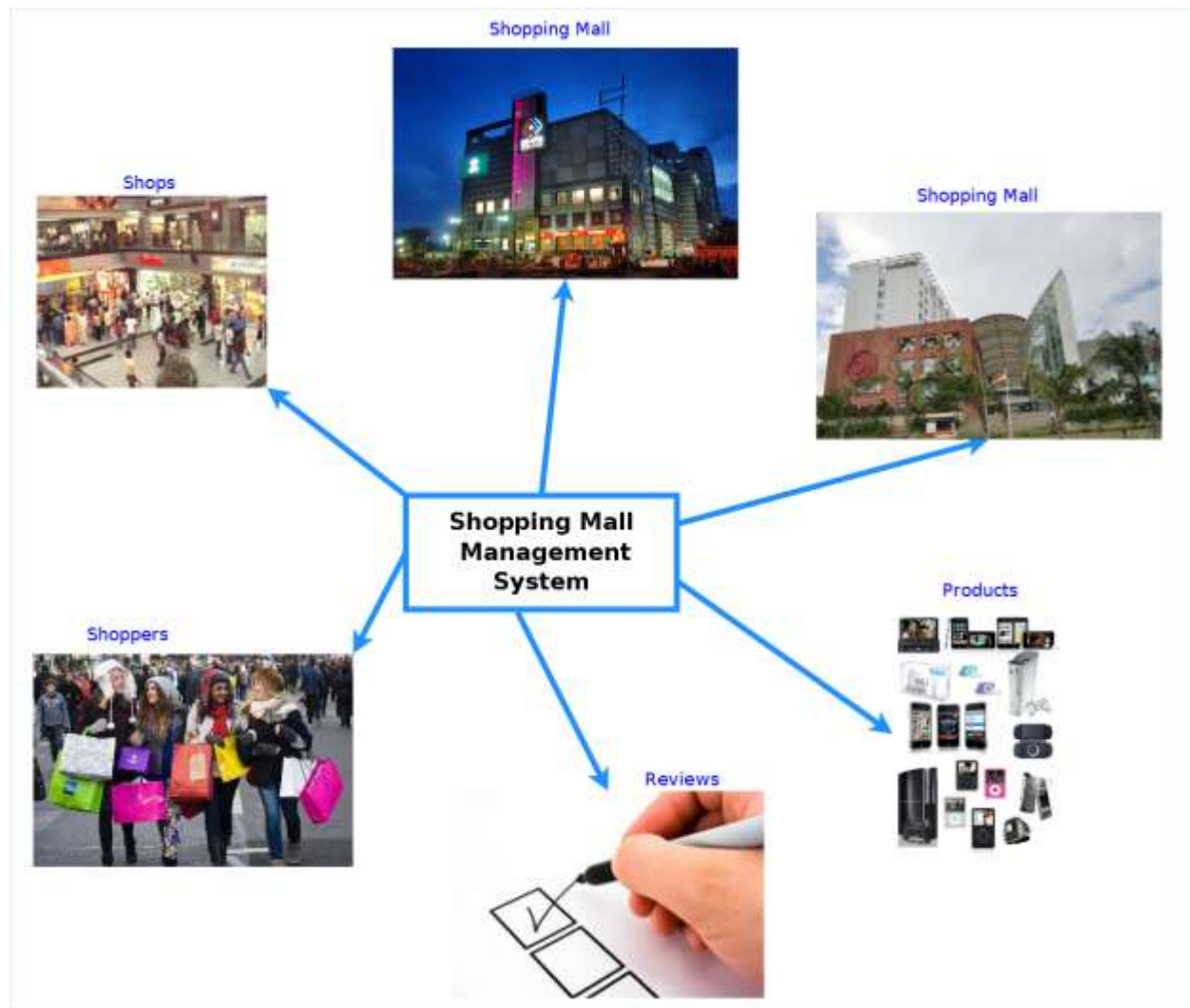
Visual Studio 2010	187
Front End	190
WPF (Windows Presentation Framework)	190
Extensible application Markup Language (XaML).....	193
Programming Framework.....	194
.NET 4.5.....	194
Database/backend:.....	201
MySQL.....	201
ide for Database	204
MySQL workbench.....	204
Programming Language	206
C# - c sharp	206
Mobile App Development.....	208
Nokia SDK 2.0 for Java — for Series 40 apps	208
Nokia Web - Tools.....	209
Other technologies	212
GitHub – repository management tool.....	212
Dia for Diagram Drawing & Modeling.....	213
Cacoo: online drawing tool.....	214
API.....	215
Google Spreadsheet Interface:	216
GLOSSARY	216

INTRODUCTION AND OBJECTIVE

INTRODUCTION

Rapid advancement of both Information Technology & Civil Engineering has taken the urban life style of our country to another level. Huge flats are replacing small huts and large malls and multiplexes are eliminating small, medium and even large stationary shops. Even the common people are now going for shopping to the big huge shopping malls. Most of us often find it confusing to understand what or which things are available in these places as these places contain uncountable shops selling uncountable products. So, obviously it is the modern technology again that could eliminate these sorts of problems.

We could easily use latest softwares and technologies available in the world to provide a proper guideline to the visitors. We could develop something that would easily let a visitor find whatever he is looking for in the mall, from a single place. May be an application containing all the information about the shops and products available in the shopping mall could to the trick.

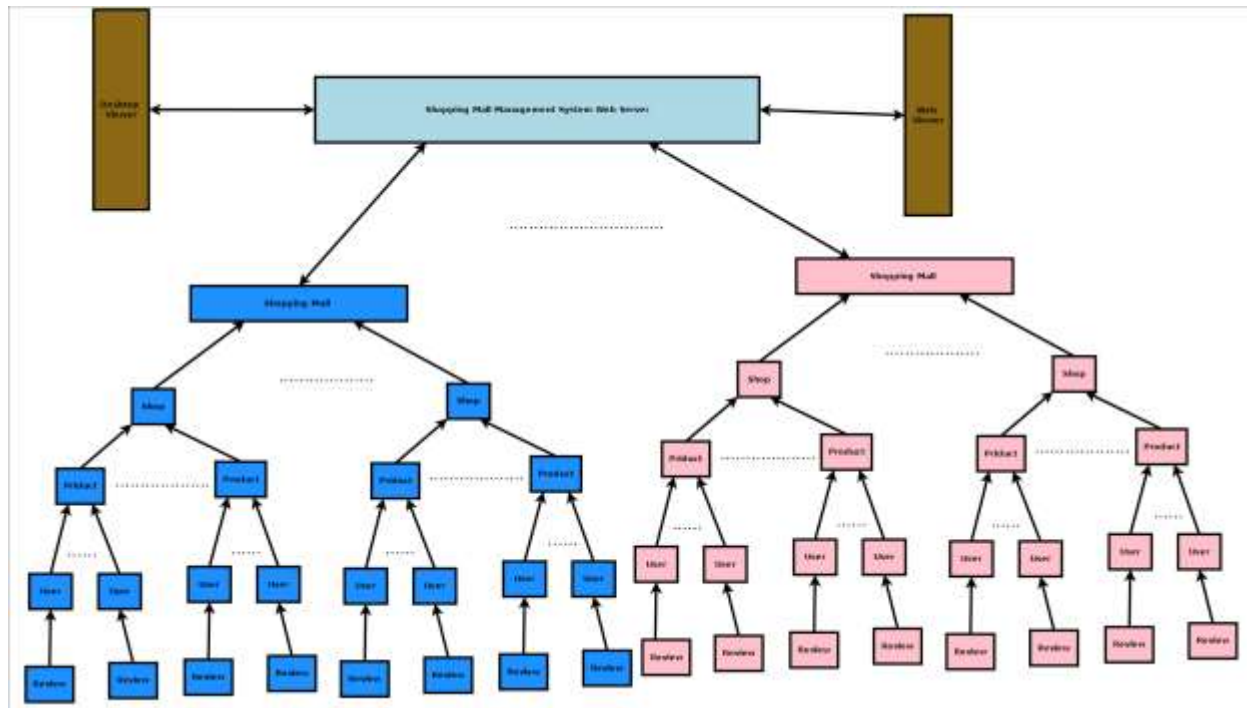


OBJECTIVE

Shopping Management System is versatile and complete end-to-end shopping management software. Following are the objectives of this application:

- Provide complete details of the shopping mall.
- Provide complete details of the shops of the shopping mall & its locations in the shopping mall.
- Details of the products available in the shopping mall.
- User review and rating of shops, products and prices.
- Search for a shop or a product.
- Search for the location(s) where a particular product is available.
- Website provides all the shopping mall related information.
- Via website gather customer review and rating from everywhere.
- Provide overall rating and review/experience for other customers.

- Windows based Desktop application which will provide all the information about a shopping mall. This application can be placed in a Kiosk while entering the Shopping mall so that the users can enquire about desired shop and product.



SYSTEM ANALYSIS

IDENTIFICATION OF NEED

I have been visiting various shopping malls for a few years. It is really exciting that when we go to a shopping mall for buying something, we get lots of options and we could choose the best one out of them. But, more the options more confusions. That is why after going to a large shopping mall I struggle to know whether my favorite brands or items are available or not. Or we often miss some good shops as we do not know much about them.

So I personally felt that it is really necessary to have a guide in a shopping mall for the visitors so that they could know everything available (or not available) in the mall. I really wished there was a way to know how good those shops which are never visited by me.

PRELIMINARY INVESTIGATION

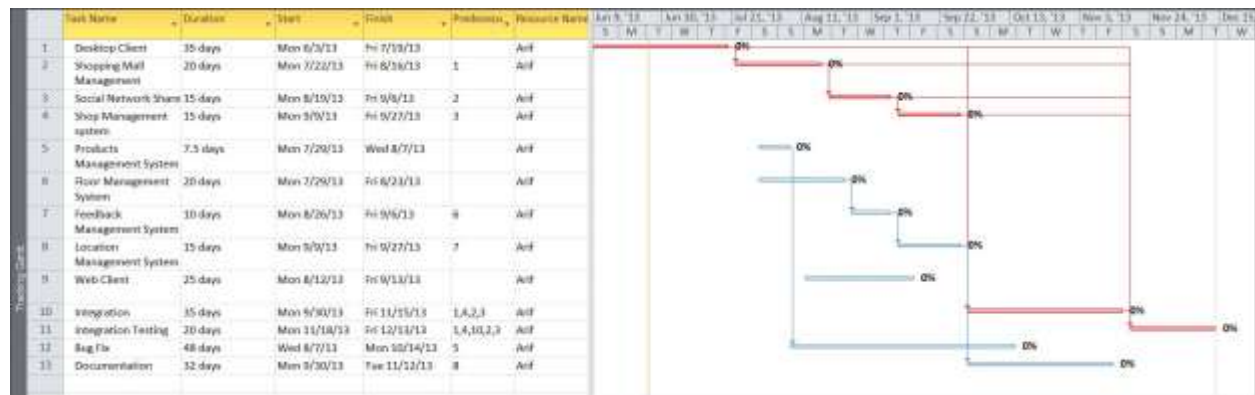
I discussed these with my class mates, friends and some of my seniors who are IT professionals as well. As expected, they face the same problem also. I queried about their expectations and ask for their thought on this topic. I understood that it is really necessary to have an automated guide of the shopping mall that could tell us anything and everything about those shops, their products, foods and even what other visitors and customers are thinking about them.

FEASIBILITY STUDY

We all know that the numbers of shopping mall are growing as fast as Information Technology. We need a proper guide while visiting a shopping mall so that we could get exactly what we wanted and in a cheapest possible price. No human can memorise all these things and remember them when asked. Developing an application is very easy these days so people would love to use technology ease their tasks. There is need and there is solution so undoubtedly this software is going to be appreciated by the market.

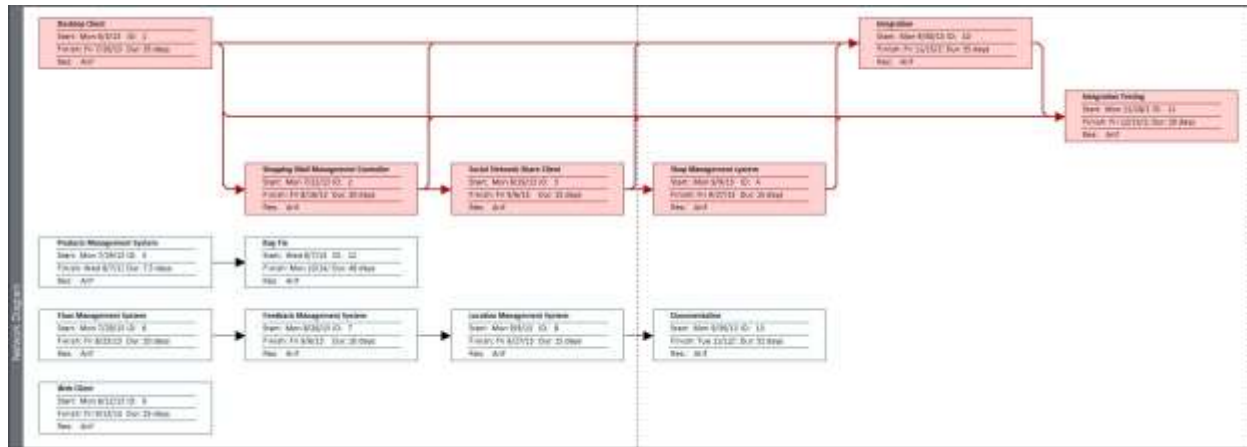
PROJECT PLANNING

TRACKING GANTT

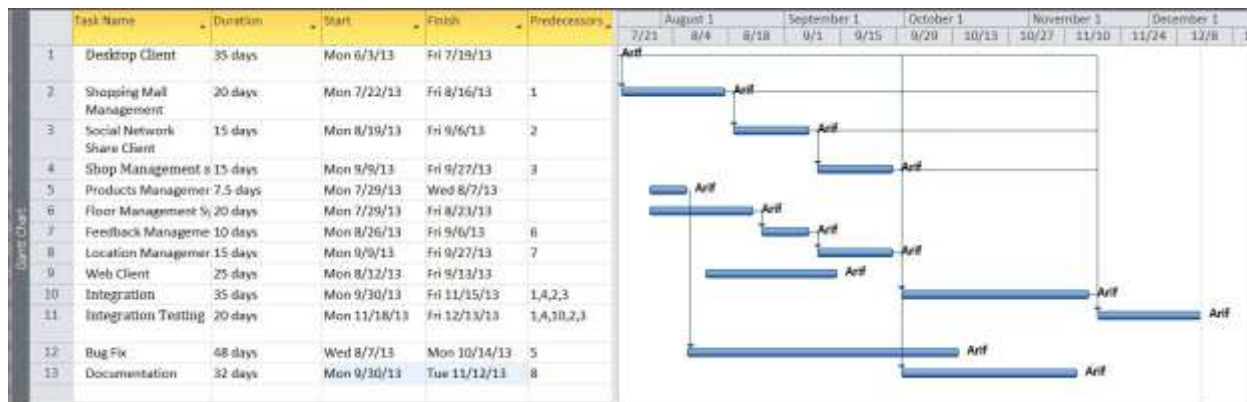


PROJECT SCHEDULING

PERT CHART (NETWORK DIAGRAM)



GANTT CHART



SOFTWARE REQUIREMENT AND SPECIFICATION

FUNCTIONAL REQUIREMENTS

ENTER NEW SHOP INFORMATION

INTRODUCTION

A person with proper authority (a person from the shopping mall admin) can enter details of a shop that is located in the shopping mall.

INPUT

Relevant shop data like name, category, description and floor are written in the text fields.

PROCESSING

Details provided by the admin are stored in MySQL database along with an auto generated id.

OUTPUT

Visitors can see shop details from the application.

ENTER NEW PRODUCT INFORMATION

INTRODUCTION

A person with proper authority (a person from the shopping mall admin) can enter details of a product that is available in a shop.

INPUT

Relevant product data like product name, brand, category, description are written in enter product details window.

PROCESSING

Details provided by the admin are stored in MySQL database along with a auto generated id.

OUTPUT

Visitors can see shop details from the shop details window.

SEND FEEDBACK ABOUT EXPERIENCE IN SHOPPING MALL

INTRODUCTION

Customer can send their feedback about their experience in shopping mall

INPUT

Customers write down experience in the feedback window and click on send button.

PROCESSING

The data provided by the customers are stored in the corresponding table at the database.

OUTPUT

The details can be seen by other customers in that particular product details.

CHANGING PASSWORD AND USERNAME

INTRODUCTION

Admin will be able to change existing username and password

INPUT

New username and password along with the old password are provided.

PROCESSING

Old username and password will be replaced by user provided new username and password after authenticating.

OUTPUT

Password and Username can be changed according to the higher authority requirement whenever they want to change for better security of the System.

SHOW SHOP DETAILS

INTRODUCTION

Every shop details are available for customers.

INPUT

Customers click on show shop details button.

PROCESSING

All details of a shop along with rating and review are fetched from database.

OUTPUT

All the details are shown to the customer in a list view.

SHOW PRODUCT DETAILS

INTRODUCTION

Details of each product are available and customer can also view new product that are available in market

INPUT

A customer clicks on product details window.

PROCESSING

All the details of products are fetched from database along with user rating and reviews.

OUTPUT

Customer can see product details

SHOW CONTACT US DETAILS

INTRODUCTION

Every customer details are stored in database at the time of complain.

INPUT

Date, name, address, mobile no, email id, type, complain/feedback of customer at complain area.

PROCESSING

The data provided by the customers are stored in the corresponding table at the database.

OUTPUT

Each customer's feedback or complain are viewed by other customer.

SORTING OF INFORMATION

INTRODUCTION

A user should be able to sort the information provided in the list view.

INPUT

A user clicks on the top of a particular column.

PROCESSING

The application sorts the items according to the data type.

OUTPUT

The user can see all the data sorted alphabetically or numerically according to the data type.

SEARCHING OF INFORMATION

INTRODUCTION

A user should be able to search a particular item inside the list view

INPUT

A user writes some product/shop name in the provided text fields and press search.

PROCESSING

Application searches the information inside the database's particular table.

OUTPUT

All the available result that matches the name of the text provided in the text field are shown in the list view.

WEB SYNC

INTRODUCTION

Web sync helps to share local data from local computer to web server.

INPUT

Admin click on web sync button.

PROCESSING

All details of shopping mall, shop, product along with rating and review are synced to web from local database.

OUTPUT

All the details are shown to the customer in the web site.

SEND FEEDBACK ABOUT IN WEBSITE

INTRODUCTION

Customer can send their feedback about their experience in shopping mall via website

INPUT

Customers write down experience in the feedback window and click on submit button.

PROCESSING

The data provided by the customers are stored in the web server.

OUTPUT

The details can be seen by other customers globally in website.

GIVE RATING

INTRODUCTION

Customer can put their rating related to shopping mall, shop, product etc.

INPUT

Customers put rating via web site.

PROCESSING

The rating data provided by the user are stored in the corresponding table at the database.

OUTPUT

The rating can be seen by other customers via web site.

NON FUNCTIONAL REQUIREMENTS

- ❖ The software must have a guideline for a shopping mall that help user to shop from it.
- ❖ User can give feedback about their experience by which the administrative department of the shopping mall can overcome their drawbacks
- ❖ The software should be password protected to secure confidential data.
- ❖ The application must be fast and flexible so that it reduces the time of shopping.
- ❖ There must be a backup and restore feature so that the valuable data stays secure forever.

SOFTWARE ENGINEERING PARADIGM APPLIED

We have followed agile version of Model Driven Development (MDD). As the name implies, AMDD is the agile version of Model Driven Development (MDD). MDD is an approach to software development where extensive models are created before source code is written. A primary example of MDD is the Object Management Group (OMG)'s Model Driven Architecture (MDA) standard. With MDD a serial approach to development is often taken, MDD is quite popular with traditionalists, although as the RUP/EUP shows it is possible to take an iterative approach with MDD. The difference with AMDD is that instead of creating extensive models before writing source code you instead create agile models which are just barely good enough that drive your overall development efforts. AMDD is a critical strategy for scaling agile software development beyond the small, co-located team approach that we saw during the first stage of agile adoption.

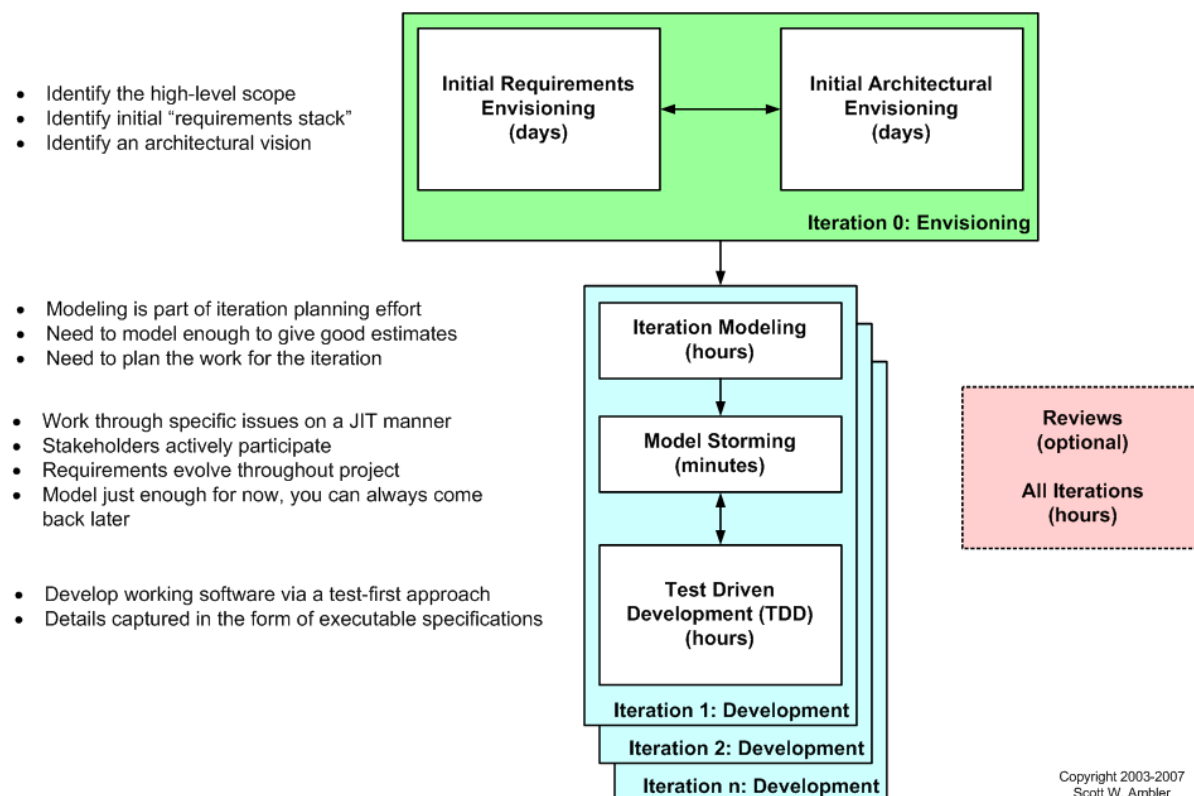


Figure 1 The AMDD lifecycle: Modeling activities throughout the lifecycle of a project

Above Figure depicts a high-level lifecycle for AMDD for the release of a system. First, let's start with how to read the diagram. Each box represents a development activity. The envisioning includes two main sub-activities, initial requirements envisioning and initial architecture envisioning. These are done during iteration 0, iteration being another term for cycle or sprint. "Iteration 0" is a common term for the first iteration before you start into development iterations, which are iterations one and beyond (for that release). The other activities – iteration modeling, model storming, reviews, and implementation – potentially occur during any iteration, including iteration 0. The time indicated in each box represents the length of an average session: perhaps you'll model for a few minutes then code for several hours. I'll discuss timing issues in more detail below..

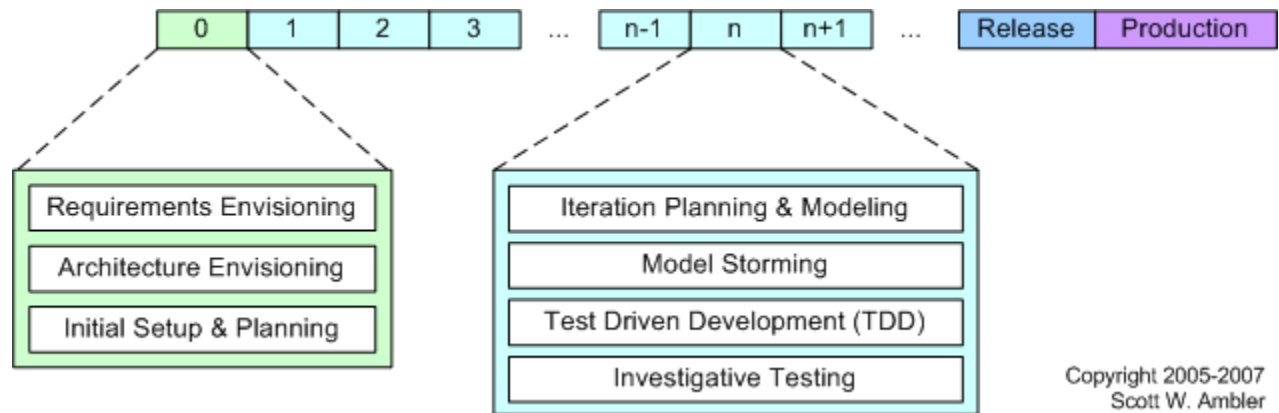


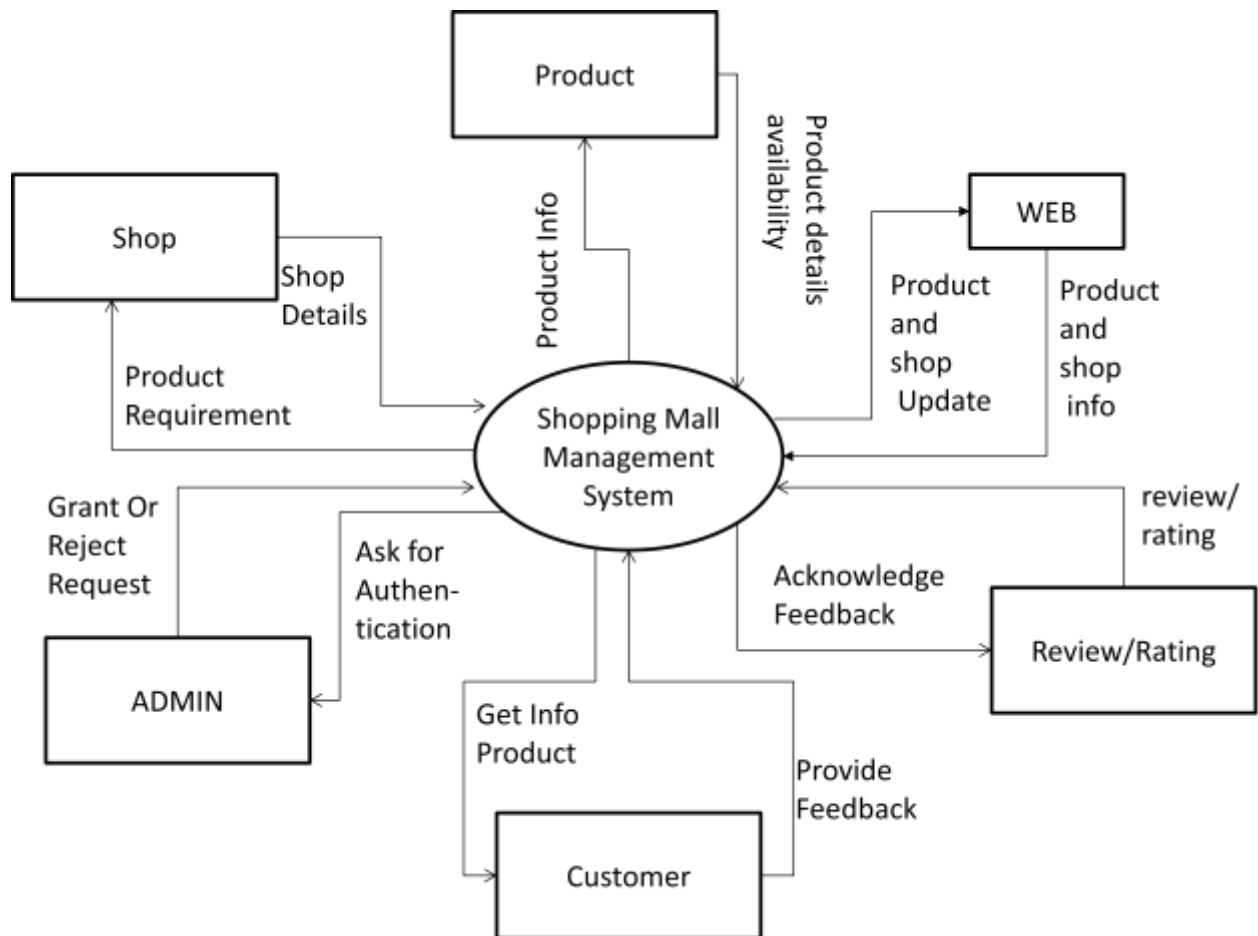
Figure 2AMDD Through the Agile Development Lifecycle.

Above Figure depicts how the AMDD activities fit into the various iterations of the agile software development lifecycle. It's simply another way to show that an agile project begins with some initial modelling and that modelling still occurs in each construction's iteration.

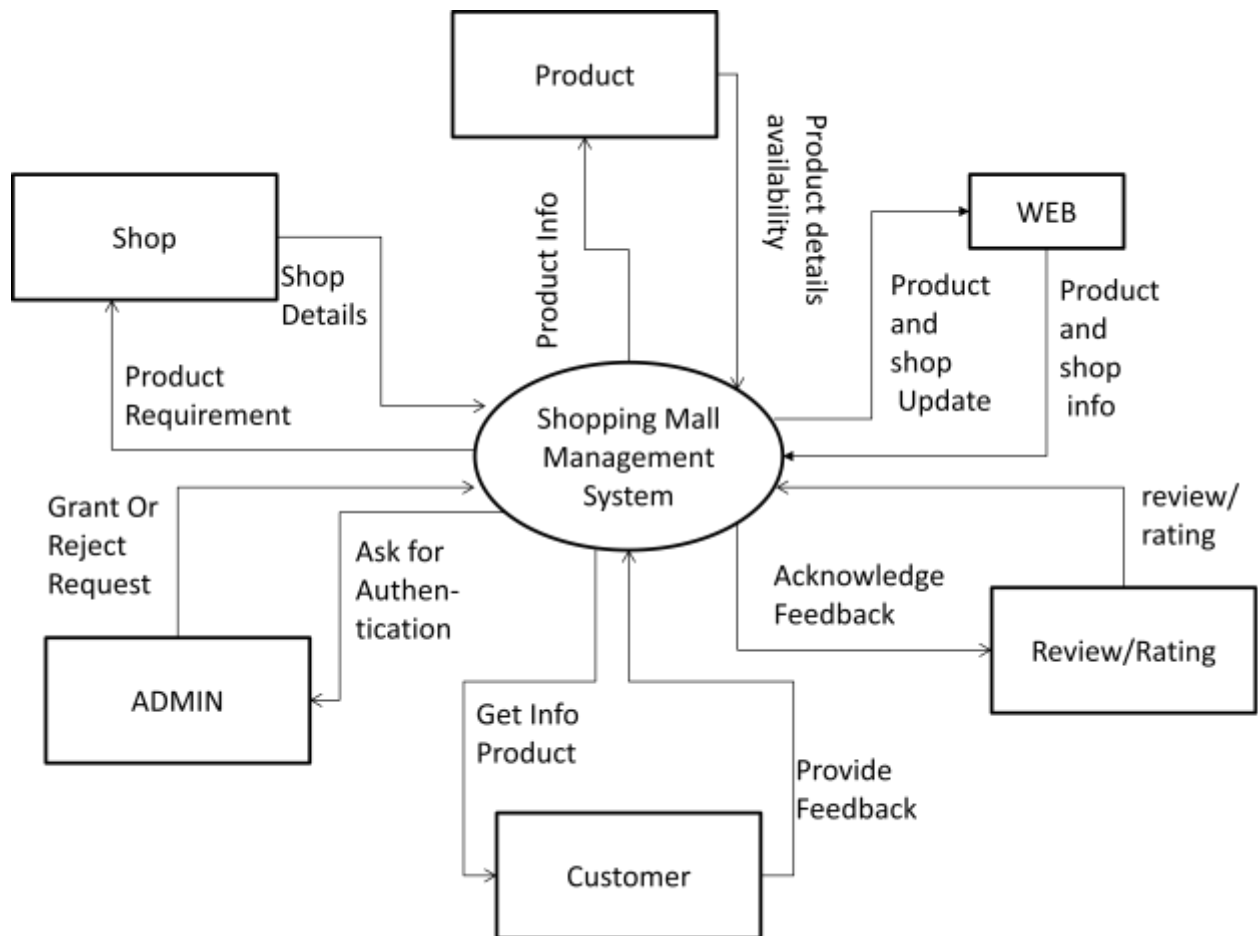
DATA MODELS

DATA FLOW DIAGRAM (DFD)

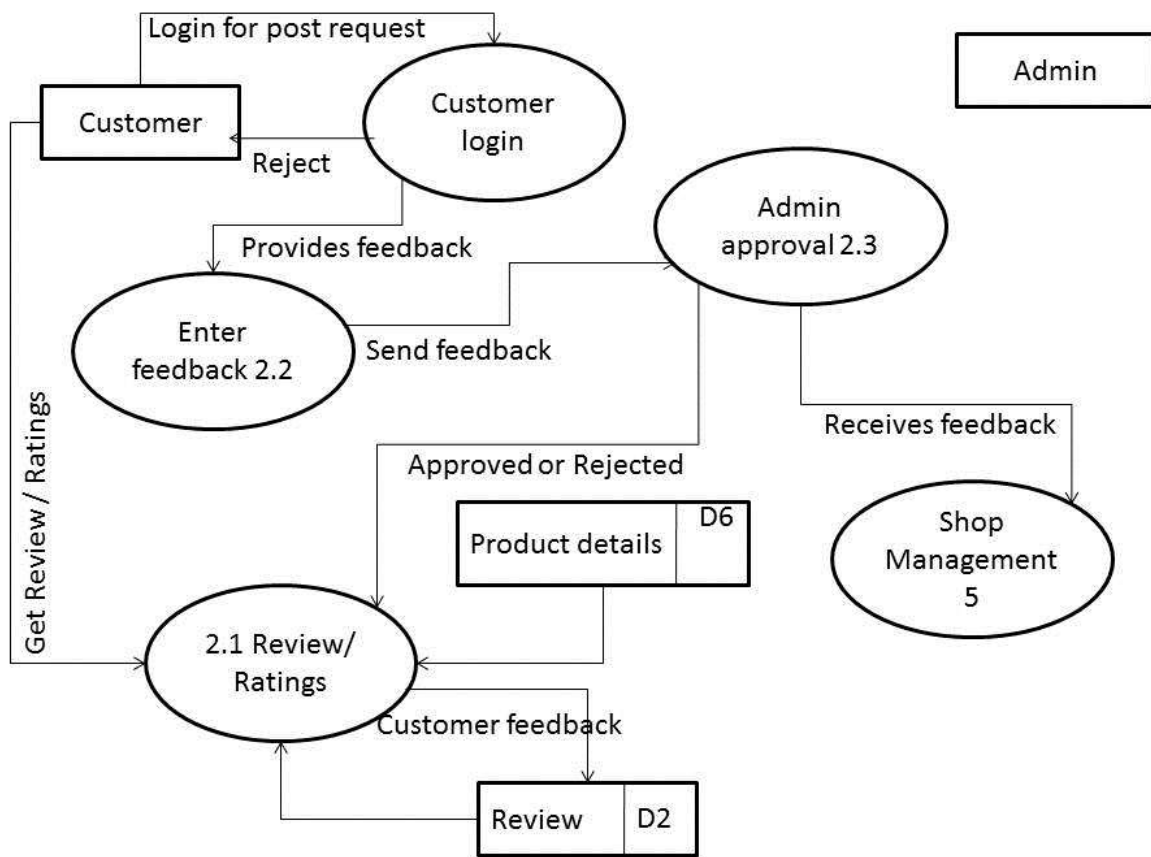
CONTEXT DIAGRAM



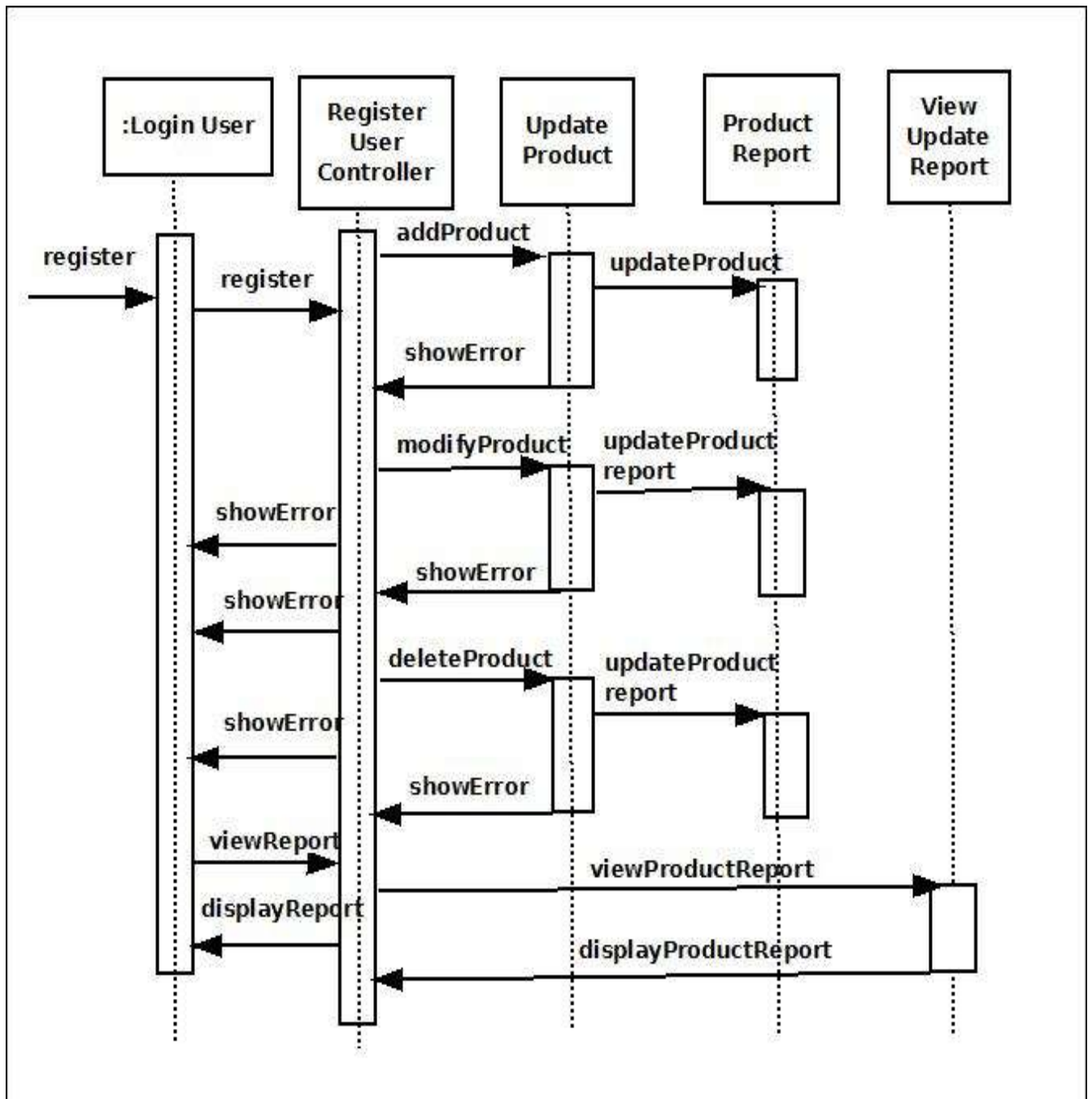
LEVEL 0 DFD



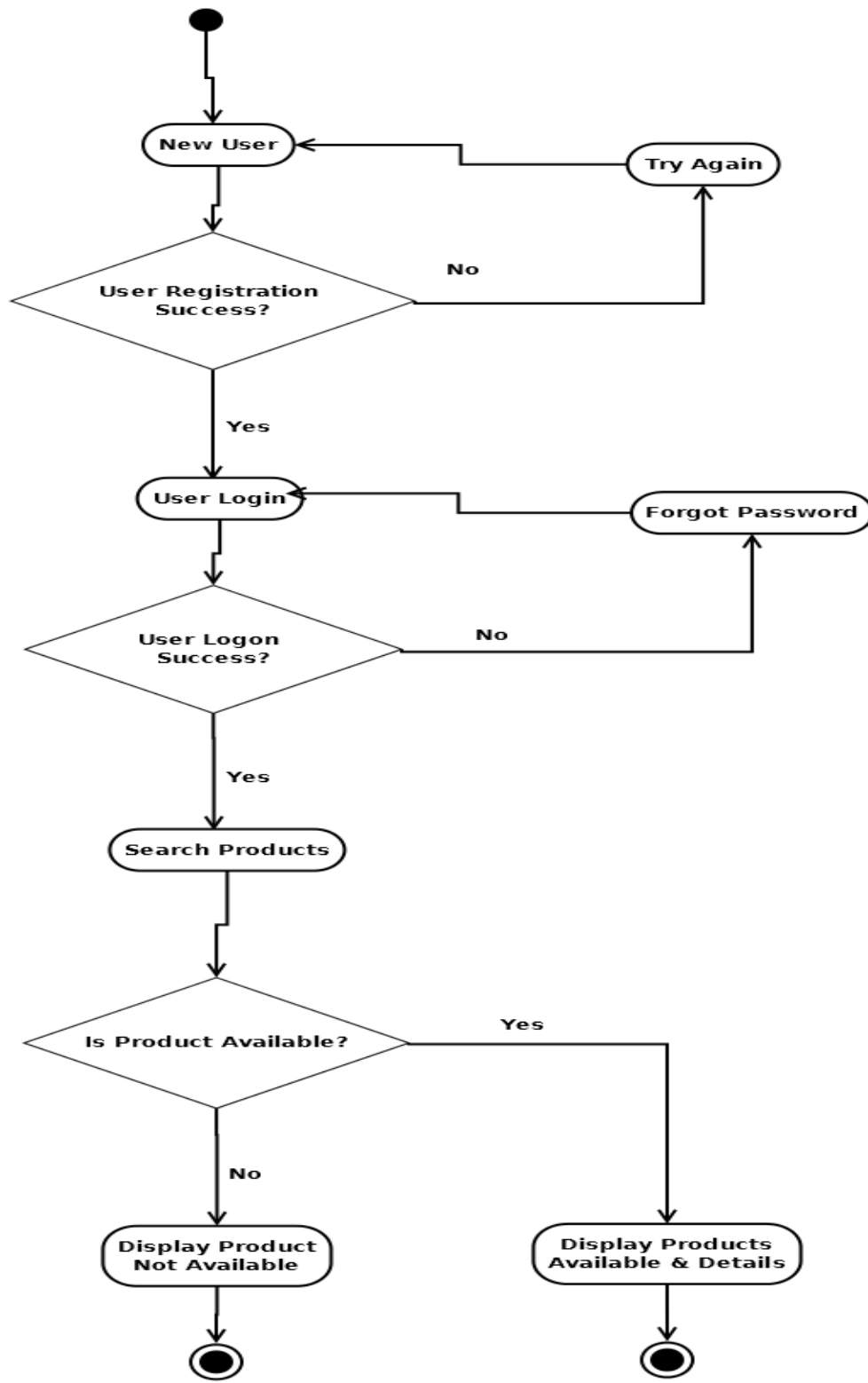
LEVEL 1 DFD



SEQUENCE DIAGRAMS



ACTIVITY DIAGRAMS



ENTITY RELATIONSHIP MODEL

We will design a RDBMS for Shopping Mall Management System. The entities and their attributes are listed below. Attributes in Bold letter is the primary key.

Entities	Attributes
Admin	Id , username, password, adminType
Shops	Id , name, tag, type, availableinfloor, rating, description
Products	Id , name, brand, type, description, image
feedback	Id , item, feedDate, name, email, rate, feedback
ContactDetails	Id , name, email

Relationship between Entities:

Shopping Management System has Shops 1 : N

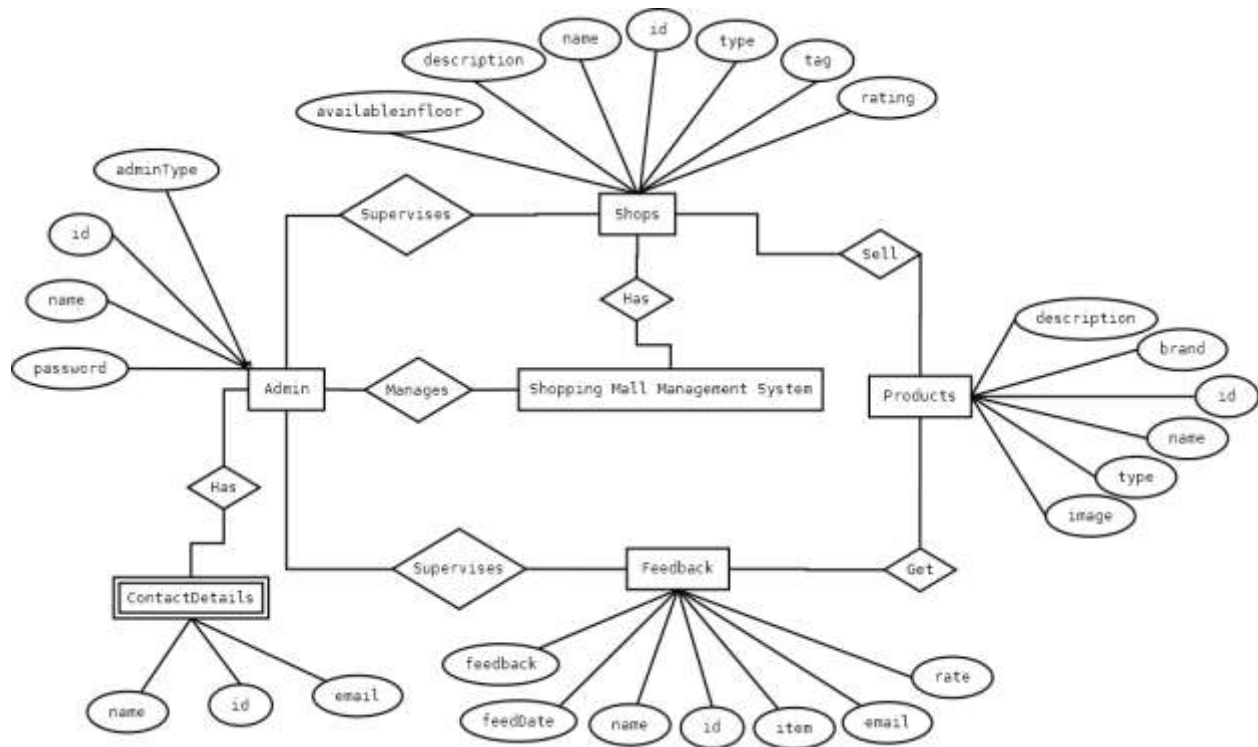
Shopping Management System has admin 1 : 1

Shops sell products 1:N

Admin manages shops 1:N

Products get feedbacks N:N

Admin supervises feedbacks 1:N



SYSTEM DESIGN

MODULARISATION DETAILS

Shoppin Mall Management System is divided three main modules such as:

1. ShoppingMallUI
2. ShoppingMallData
3. ShoppingMallDb

SHOPPING MALL UI

This module consists on all the Graphical User Interface related codes. We first added this module and started designing the UI according to the requirement. All the codes are done using WPF and XAML. This module is able to connect with the ShoppingMallData module to send the data to ShoppingMallDb.

The cs pages in this module control the logical codes of inputs and outputs.

SHOPPING MALL DATA

This module does one and one task only, it carries the data. All the data or inputs taken from user are sent to this module and it further sends the data to the ShoppingMallDb. On the other hand, the outputs sent by the ShoppingMallDb are carried by this module and those are sent to the UI to show them to the user. This module consists on c sharp logics

SHOPPING MALL DB

In this module, we write database queries to store and fetch data from database. It takes inputs and stores in database and sends those data as output to user according to their query.

Shopping Management System will have a unified database for storing all the information. It can be a networked database or a database situated in the server machine. In our project, we are using local database.

DATA INTEGRITY AND CONSTRAINTS

We have used Integrity constraints in SMMS to ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity. There are many types of integrity constraints in **SMMS** that play a role in referential integrity.

Codd initially defined two sets of constraints but, in his second version of the relational model, he came up with four integrity constraints:

ENTITY INTEGRITY

In **SMMS** we used various type of primary key and consciously we set the primary key property as not null. The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation. Having null value for the primary key implies that we cannot identify some tuples. This also specifies that there may not be any duplicate entries in primary key column key row.

REFERENTIAL INTEGRITY

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation. It is a rule that maintains consistency among the rows of the two relations.

DOMAIN INTEGRITY

SMMS has various type of data field with set by default value of Null because if the value is not provided by the user, the value will be set as null. The domain integrity states that every element from a relation should respect the type and restrictions of its corresponding attribute. A type can have a variable length which needs to be respected.

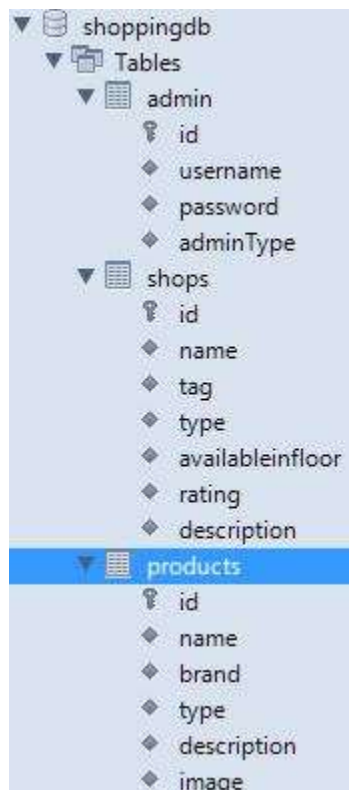
Restrictions could be the range of values that the element can have, the default value if none is provided, and if the element can be NULL.

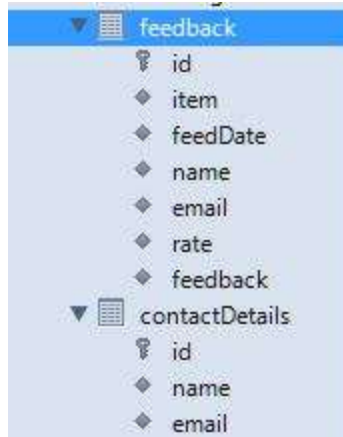
USER DEFINED INTEGRITY

A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behaviour of the business.

DATABASE AND TABLE DESIGN

The database used for this software is called **shoppingDB**. A screenshot from the MySQL workbench is given below. It shows the tables and its columns. The first row is the primary key.





USER INTERFACE DESIGN

DESKTOP APPLICATION USER INTERFACE

FOR WINDOWS(C#.NET)



Main window

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Admin Zone

User Name: rozinanktr

Password: ••••••

Reset Login

Login window

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Admin Zone Logout Admin

Manage Products Manage Shops Manage FeedBack Manage Others

Name: [] Available in : []

Brand: [] Delete Add

Type: Garments

Description: []

Browse Reset Submit

Id	Name	Brand	Type	Description	Available In

Search Refresh Edit Delete

After login

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Admin Zone Logout Admin

Manage Products Manage Shops Manage FeedBack Manage Others

Name Lumia 620 Available in :

Brand Nokia

Type Mobile

Description Good configure.

Add Products

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Admin Zone Logout Admin

Manage Products Manage Shops Manage FeedBack Manage Others

Name Lumia 620 Available in :

Brand Nokia

Type Mobile

Description Good configure

Id	Name	Brand	Type	Description	Available In
41408.524	Lumia 620	Nokia	Mobile	Good configure.	
41408.532	Laptop	Dell	Computer	HDD- 500gb, RAM-4gb, Processor-i5, Pspe	
41408.534	LED TV	Sony Bravia	Electronics	Good and Clear picture, sound quality is so g	
41408.536	Web Camera	Coolex	Electronics	Its picture and vedio quality are HD.	
41408.545	Garments Things	Killer, Levis.	Garments	Branded garments things avilable here.	Nokia Care, West Estern , Saha Texti

View Products Details

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Admin Zone Logout Admin

Manage Products Manage Shops Manage FeedBack Manage Others

Name: Saha Textile
 Tag: Your family shop
 Type: Garments
 Available In: First Floor
 Rate (Out of 10): 7
 Description: All type of garments things available here.

Available Product :

Delete Add

Browse

Reset Submit

ADD SHOPS

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Admin Zone Logout Admin

Manage Products Manage Shops Manage FeedBack Manage Others

Name: Saha Textile
 Tag: Your family shop
 Type: Garments
 Available In: First Floor
 Rate (Out of 10): 7
 Description: All type of garments things available here.

Available Product :

Delete Add

Browse

Reset Submit

Id	Name	Tag	Type	Floor	Rate	Description
41408.529	Saha Textile	Your family shop.	Garments	First Floor	7	All type of garments things available here.
41408.537	Nokia Care	Mobile Shop	Mobile	First Floor	2	All types mobile store here
41408.540	West Estern	Electronics Shop	Electronics	Ground Floor	5	Tv, Washing Machine, Sound System.
41408.541	Raj Computer	Computer Shop	Computer	Second Floor	4	Dekstop, Laptop, All type computer's parts a
41408.543	Nandi Gift Center	Cosmetics And Gift Shop	Others	Second Floor	6	Brand Cosmetics things and Gift things avail

Search Refresh Edit Delete

VIEW SHOPS DETAILS

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Floor

Ground Floor First Floor Second Floor

Go Refresh

Shop List


West Eastern

Moon World

Good collection.

Available Products :

Date	Name	Email	Rating	Feedback
5/14/2013	Soumen Paul	soumen@gmail.c	6	Nokia Lumia 620 mobile so good for gaming
5/14/2013	Amit Paul	amit@ymail.com	8	Dell laptop is a good for use.
5/14/2013	Anirban Nandi	anirban@gmail.c	7	LG's electronics things are good.
5/14/2013	Santu Sengupta	santu@hotmail.c	3	Nokia Lumia 620's don't external sd card.



FLOOR DETAILS VIEW

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Contact Us

Today 5/14/2013

Name Rozina Akhter

Address Kolkata

Mobile 9836123456

Email rozina@gmail.com

Type Customer

Complain/Feedback All thing are good.

Reset Submit

Date	Name	Address	Contact No.	Email	Type	Feedback
------	------	---------	-------------	-------	------	----------

ADD CONTACT US

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Contract Us

Today 5/14/2013

Enter your Name Address Mobile Email Type Complain/Feedback

Feed Back

Reset Submit

Date	Name	Address	Contact No.	Email	Type	Feedback
5/14/2013	Soumen Paul	Barasat	9933123456	soumen@gmail.com	Customer	Nokia Care is solve all nokia mobiles pro
5/14/2013	Amit Paul	Barasat	9830123456	amit@ymail.com	Customer	Raj Computer not good for byeing laptop
5/14/2013	Anirban Nandy	Barasat	9883123456	anirban@hotmail.com	Customer	Saha Textile is good for all types garmen
5/14/2013	Santu Sengupta	Bongaon	9831123456	santu@ymail.com	Customer	West Estern Electronics shop is not good
5/14/2013	Rozina Akhter	Kolkata	9836123456	rozina@gmail.com	Customer	All thing are good.

CONTACT US DETAILS VIEW

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Admin Zone Logout Admin

Manage Products Manage Shops Manage FeedBack Manage Others

Name Brand Type Description

Available in : Delete Add

Browse Reset Submit

Id	Name	Brand	Type	Description	Available In
41408.524	Lumia 620	Nokia	Mobile	Good configure.	
41408.532	Laptop	Dell	Computer	HDD- 500gb, RAM-4gb, Processor-i5, Pspes	
41408.534	LED TV	Sony Bravia	Electronics	Good and Clear picture, sound quality is so g	
41408.536	Web Camera	Coolux	Electronics	Its picture and vedio quality are HD.	
41408.545	Garments Things	Killer, Levis.	Garments	Branded garments things available here.	Nokia Care, West Estern , Saha Texti

lumia 620 Search Refresh Edit Delete

INSERT SEARCH

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Admin Zone Logout Admin

Manage Products Manage Shops Manage FeedBack Manage Others

Name Available in :
 Brand Delete Add
 Type Garments
 Description

Browse Reset Submit

Id	Name	Brand	Type	Description	Available In
41408.524	Lumia 620	Nokia	Mobile	Good configure.	

lunia 620 Search Refresh Edit Delete

VIEW SEARCH VALUE

New Mail

Welcome to the Shoppingmall

Products Shops Floor Contact Us Admin Zone

Shops Moon World

Go Refresh

Shop List

- Saba Textile
- Nokia Care
- West Eastern
- Raj Computer
- Nandi Gift Center
- Kmart

Good collection.

Available Products :

Lumia 620, Laptop, Garments Things.

Please Insert Info Properly

Date	Name	Email	Feedback
5/14/2013	Soumen Paul	soumen@	lumia 620 mobile so good for gaming
5/14/2013	Amit Paul	amit@ymail.com	8 Dell laptop is a good for use.
5/14/2013	Anirban Nandi	anirban@gmail.c	7 LG's electronics things are good.
5/14/2013	Santu Sengupta	santu@hotmail.c	3 Nokia Lumia 620's don't external sd card.

Your Name

Email

Shop Rating (Out of 10)

Share your experience with all

Reset Submit

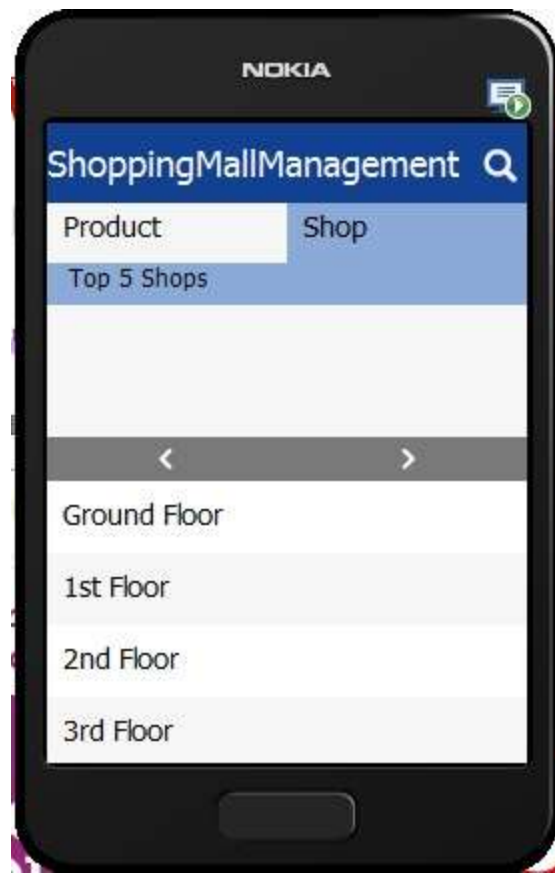
WITHOUT VALUE ERROR MESSAGE

MOBILE APP INTERFACE

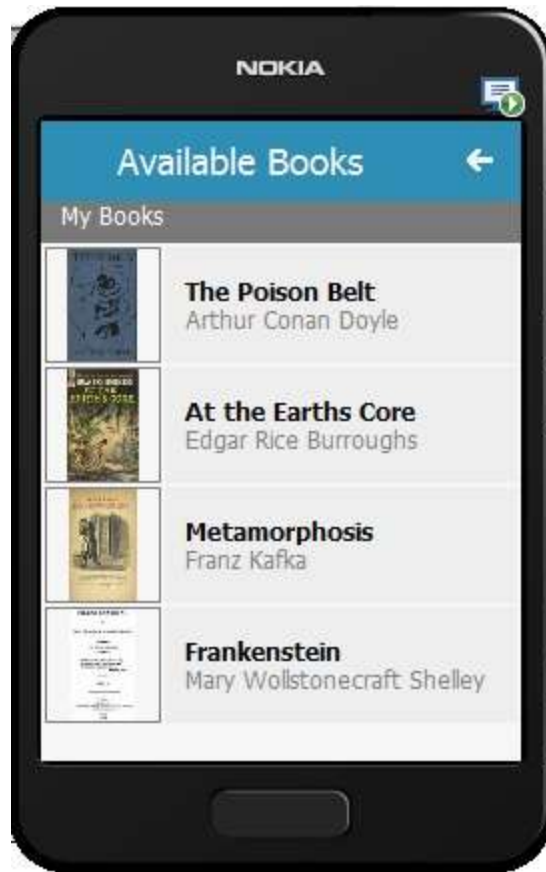
PRODUCTS WINDOW



SHOPS WINDOW:



BOOKS (SELECTED PRODUCT) WINDOW



TEST CASES

UNIT TEST CASES

Test Case Id	Type	Github ID	Subject	Test Name	Test Description	Step Name	Description	Expected Result
SMMS-001	Manual	04ad2993baf3b9b74d8f0d97d29de653f92bdd3f	E:\DEVELOPERS_ZONE\GitHub\ShoppingMallManagementSystem	Check Successful Login for Admin Zone	The purpose of this test is to verify that the User	Step 1	Insert wrong User Name and Password. And Click on Login Button.	SMMS will display error message. And Failed to Login.

			\code		Name and Password of Admin is valid.			
SMMS -002						Step 2	Insert Wrong User Name and valid Password. And Click on Login Button.	SMMS will display error message. And Failed to Login.
SMMS -003						Step 3	Insert Valid User Name and Wrong Password. And Click on Login Button.	SMMS will display error message. And Failed to Login.
SMMS -004						Step 4	Insert Nothing in User Name and Password fields. And Click on Login Button.	SMMS will display error message. And Failed to Login.
SMMS -005						Step 5	Insert Nothing in User Name and insert Valid Password fields. And Click on Login Button.	SMMS will display error message. And Failed to Login.
SMMS -006						Step 6	Insert Nothing in Password and insert Valid User Name fields. And Click on Login Button.	SMMS will display error message. And Failed to Login.
SMMS -007						Step 7	Insert Nothing in User Name and insert invalid	SMMS will display error message. And

							Password fields. And Click on Login Button.	Failed to Login.
SMMS -008						Step 8	Insert Nothing in Password and insert invalid User Name fields. And Click on Login Button.	SMMS will display error message. And Failed to Login.
SMMS -009						Step 9	Insert valid User Name and Password. And Click on Login Button.	Successfully login to Admin Zone.
SMMS -010	M a n u a l	bd0bd3e2d5 fae2e5f58a9 3527d806cb 179dcd828	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check Successful Product's insertion in SMMS Admin Zone.	The purpose of this test is to check insertion process of product.	Step 1	Insert Nothing in Product's details fields like in Name, Type, Brand etc. and click on Submit Button.	Insertion failed. Display error message.
SMMS -011						Step 2	Keep empty one field among Product's details fields like in Name, Type, Brand, Description and click on Submit Button.	Insertion failed. Display error message.
SMMS -012						Step 3	Insert proper Product's details fields like in Name, Type, Brand, Description and click on Submit Button.	Successful Insertion. And Product details will display in List view.

SMMS -013						Step 4	Insert Product's details fields like in Name, Type, Brand, Description and click Reset button.	Successful to clear Product's details fields.
SMMS -014	M a n u a l	9f6457fbecf a275217a7e d1a23f12dcf a6e33342	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check Successful Product's deletion in SMMS Admin Zone.	The purpose of this test is to check delete process of product.	Step 1	Click on Delete button without selecting Product's details.	Display error message.
SMMS -015						Step 2	Select Product's details and Click on Delete button.	Deleted successfully.
SMMS -016	M a n u a l	c5804ae447 a24ef03121 4c085501a5 4882e2c4cb	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check Successful Product's edition in SMMS Admin Zone.	The purpose of this test is to check edition process of product.	Step 1	Click on Edit button without selecting Product's details.	Display error message.
SMMS -017						Step 2	Select a Product's details and Click on Edit button.	Asking to edit and click on Update to successfully update Product's details.
SMMS -018	M a n u a	926c344b67 d699991bfa 9c954a7471 ec1bc34bad	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall	Check Successful Product's search in SMMS	The purpose of this test is to check	Step 1	Write search key word in search product fields and click on Search Button.	List view will display search result. If not found List view will

	1		Managem entSystem \code	Admin Zone.	search process of product.			display nothing.
SMMS -019						Step 2	Keep empty search product fields and click on Search Button.	List view will display All Product's details.
SMMS -020	M a n u a l	bd0bd3e2d5 fae2e5f58a9 3527d806cb 179dcd828	E:\DEVE LOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check Successful Shop's insertion in SMMS Admin Zone.	The purpose of this test is to check insertion process of Shop.	Step 1	Insert Nothing in Shop's details fields like in Name, Tag, Type, Available in etc. and click on Submit Button.	Insertion failed. Display error message.
SMMS -021						Step 2	Keep empty one field among Shop's details fields like in Name, Tag, Type, and Available in Description and click on Submit Button.	Insertion failed. Display error message.
SMMS -022						Step 3	Insert proper Shop's details fields like in Name, Tag, Type, and Available in etc. click on Submit Button.	Successful Insertion. And Shop details will display in List view.
SMMS -023						Step 4	Insert proper Shop's details fields like in Name, Tag, Type, and Available in etc.	Successful to clear Shop's details fields.

							click on Reset button.	
SMMS-024	Manual	9f6457fbecfa275217a7ed1a23f12dcfa6e33342	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check Successful Shop's deletion in SMMS Admin Zone.	The purpose of this test is to check delete process of Shop.	Step 1	Click on Delete button without selecting Shop's details.	Display error message.
SMMS-025						Step 2	Select Shop's details and Click on Delete button.	Deleted successfully.
SMMS-026	Manual	c5804ae447a24ef031214c085501a54882e2c4cb	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check Successful Shop's edition in SMMS Admin Zone.	The purpose of this test is to check edition process of Shop.	Step 1	Click on Edit button without selecting Shop's details.	Display error message.
SMMS-027						Step 2	Select a Shop's details and Click on Edit button.	Asking to edit and click on Update to successfully update Shop's details.
SMMS-028	Manual	926c344b67d699991bf9c954a7471ec1bc34bad	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check Successful Shop's search in SMMS Admin Zone.	The purpose of this test is to check search process of Shop.	Step 1	Write search key word in search Shop fields and click on Search Button.	List view will display search result. If not found List view will display nothing.
SMMS-029						Step 2	Keep empty search Shop fields and click on Search	List view will display All Shop's details.

							Button.	
SMMS -030	M a n u a l	9f6457fbecf a275217a7e d1a23f12dcf a6e33342	E:\DEVE LOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check Successful Feedback deletion in SMMS Admin Zone.	The purpose of this test is to check delete process of Feedback .	Step 1	Click on Delete button without selecting Feedback 's details.	Display error message.
SMMS -031						Step 2	Select Feedback's details and Click on Delete button.	Deleted successfully.
SMMS -032	M a n u a l	04ad2993ba f3b9b74d8f 0d97d29de6 53f92bdd3f	E:\DEVE LOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check Logout of Admin Zone.	It is to check that Logout works properly for Admin Zone.	Step 1	Click on logout button to logout from Admin Zone section.	Successfully Logout from Admin Zone and will ask for login.
SMMS -033	M a n u a l	114b6803a3 d7461287aa 649d526c29 2a3819b0a2	E:\DEVE LOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check Product details.	It is to check that the product details displayin g all informati on about the product.	Step 1	Click on Product on SMMS .	Successfully Display all information area about Products.

SMMS -034						Step 2	Click on particular product from the product list.	All Details will display in Product area with Users feedbacks rating etc.
SMMS -035	M a n u a l	07fcafc2276 7661397ce1 98c00f9076 ea3b6c578	E:\DEVE LOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check Successful Product's search in SMMS .	The purpose of this test is to check search process of product.	Step 1	Write search key word in search product fields and click on Search Button.	List view will display search result. If not found List view will display nothing.
SMMS -036							Keep empty search product fields and click on Search Button.	List view will display All Product's details.
SMMS -037	M a n u a l	bd0bd3e2d5 fae2e5f58a9 3527d806cb 179dcd828	E:\DEVE LOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check Successful Product feedback submission in SMMS .	The purpose of this test is to check Product feedback submissi on process.	Step 1	Insert Nothing in Product Feedback fields like in Name, Email etc. and click on Submit Button.	Feedback failed. Display error message.
SMMS -038						Step 2	Keep empty one field among Product feedback fields like in Name, Email etc. and click on Submit Button.	Feedback failed. Display error message.

SMMS -039						Step 3	Insert proper Product details fields like in Name, Email etc. and click on Submit Button.	Successful Insertion Feedback. And Feedback will display in List view.
SMMS -040						Step 4	Insert proper Product details fields like in Name, Email etc. and click on Reset button.	Successful to clear Product's Feedback details fields.
SMMS -041	M a n u a l	114b6803a3d7461287aa649d526c292a3819b0a2	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check Shop details.	It is to check that the Shop details displaying all information about the Shop .	Step 1	Click on Shop on SMMS .	Successfully Display all information area about Shop s.
SMMS -042						Step 2	Click on particular Shop from the Shop list.	All Details will display in Shop area with Users feedbacks, rating etc.
SMMS -043	M a n u a l	f10d08847c7de769ef7c32f2280e0ef63750d816	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check Successful Shop's search in SMMS .	The purpose of this test is to check search process of Shop.	Step 1	Write search key word in search Shop fields and click on Search Button.	List view will display search result. If not found List view will display nothing.

SMMS -044							Keep empty search Shop fields and click on Search Button.	List view will display All Shop's details.
SMMS -045	M a n u a l	bd0bd3e2d5 fae2e5f58a9 3527d806cb 179dcd828	E:\DEVELOPERS_ ZONE\Git Hub\ShoppingMall ManagementSystem \code	Check Successful Shop feedback submission in SMMS .	The purpose of this test is to check Shop feedback submission process.	Step 1	Insert Nothing in Shop Feedback fields like in Name, Email etc. and click on Submit Button.	Feedback failed. Display error message.
SMMS -046						Step 2	Keep empty one field among Shop feedback fields like in Name, Email etc. and click on Submit Button.	Feedback failed. Display error message.
SMMS -047						Step 3	Insert proper Shop details fields like in Name, Email etc. and click on Submit Button.	Successful Insertion Feedback. And Feedback will display in List view.
SMMS -048						Step 4	Insert proper Shop details fields like in Name, Email etc. and click on Reset button.	Successful to clear Shop's Feedback details fields.
SMMS -049	M a n u a l	e29fa8c3bb 471b39c821 e762f95bce ecf555b9e9	E:\DEVELOPERS_ ZONE\Git Hub\ShoppingMall ManagementSystem	Check Floor wise shop details.	It is to check that the Shop details displaying all	Step 1	Click on Floor in SMMS .	Successfully Display all Shop details information Floor wise.

			\code		informati on in floor wise.			
SMMS -050	M a n u a l		E:\DEVE LOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check contact us page insertion	In this step it is checked whether data could be inserted in contact us page or not	Step 1	Open the application and click on contact us page	The contact us page gets opened along with the field to provide input on the top side and the listview at the bottom
SMMS -051						Step 2	Provide some data on the text boxes and press submit	The data should get stored in the database and the data should be visible in the list view as well
SMMS -052						Step 3	Provide data in all text fields and then press the reset button	All the data gets cleared from the text boxes
SMMS -053						Step 4	Provide some data in the text boxes but leave some empty	No data should get inserted; instead, an error message should be displayed about the empty fields.
SMMS -054						Step 5	Provide some wrong data in the text boxes, like, alphabetic value in the mobile number	An error message should be shown explaining that user

							field	cannot do such thing.
SMMS -055						Step 6	Open the contact details and see the output of inserted data	All the inserted data must be visible in the listview.
SMMS -056						Step 7	Delete or update a data from the listview	Ahe corresponding data gets updated or deleted.
SMMS -057	M a n u a l	90330b9232 8d862892fc 7743653908 1cc2b7f70d	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check everything about Password Manager information .	It is to check that Password Manager works properly.	Step 1	Click on Password tab.	Password is asking for login password to access all information.
SMMS -058						Step 2	Insert wrong User Name and Password. And Click on Login Button.	Password Manager of SMMS will display error message. And Failed to Login.
SMMS -059						Step 3	Insert correct User Name and Password. And Click on Login Button.	Password Manager login successfully. And display all Password related information.

SMMS -060						Step 4	Click on Add Info Expender and enter incorrect data or empty the mandatory field.	Failed to store Info information in database and display the error message.
						Step 5	Click on Add Info Expender and enter correct data in the mandatory field.	Successfully Added Info Information. And display in View area.
						Step 6	Click on delete button without selecting details.	Display error message.
						Step 7	Select an information and Click on Delete button.	Deleted successfully.
						Step 8	Click on Edit button without selecting details.	Display error message.
						Step 9	Select an information and Click on Edit button.	Expender Opened and asking to edit and click on Save to successfully update data.

						Step 10	Click on logout button to logout from password section.	Successfully Logout from Password.
	M a n u a l	90330b92328d862892fc77436539081cc2b7f70d	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check everything about Task information .	It is to check that Task works properly.	Step 1	Click on Task tab under Extra Tab.	Display all Task related information.
						Step 2	Click on Add Task Expender and enter incorrect data or empty the mandatory field.	Failed to store Task Info information in database and display the error message.
						Step 3	Click on Add Task Expender and enter correct data in the mandatory field.	Successfully Added Task Info Information. And display in View area.
SMMS-061						Step 4	Click on delete button without selecting details.	Display error message.
SMMS-062						Step 5	Select an information and Click on Delete button.	Deleted successfully.
SMMS-063						Step 6	Click on Edit button without selecting details.	Display error message.

SMMS -064						Step 7	Select an information and Click on Edit button.	Expenders Opened and asking to edit and click on Save to successfully update data.
SMMS -065	M a n u a l	e22bd0e470 f145f3db33 6ed9e28d47 4d8f4637d7	E:\DEVELOPERS_ ZONE\Git Hub\ShoppingMall ManagementSystem \code	Search event by Date in SMMS .	The purpose of this test is to verify that the Search option is working.	Step 1	Select a date and Click on Go To Date Button.	Display all events of the particular date .
SMMS -066								

SYSTEM TEST CASES

Test Case Id	T y p e	Github ID	Subject	Test Name	Test Descripti on	Step Nam e	Description	Expected Result
SMMS -067	M a n u a l	f3563be0a9 c431104f52 839039e860 43cf640cf1	E:\DEVELOPERS_ ZONE\Git Hub\ShoppingMall ManagementSystem \code	Check Log in.	It is to check that Login works properly.	Step 1	Click on Login button after inserting invalid User id and password from SMMS .	Login failed to SMMS . And can't able to use the feature.
SMMS -068						Step 2	Click on Login button after inserting valid User id and	Successfully Login to SMMS . And can able to

							password from SMMS .	use the feature.
SMMS -069	M a n u a l	d01197ee4c d3bee92458 74b5937ba7 40019fd131	E:\DEVE LOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check Successful Registratio n for New SMMS User.	The purpose of this test is to verify that a new user could be created in the admin zone or not.	Step 1	Click on Admin Zone button and provide right password and user name.	The admin zone gets opened.
SMMS -070						Step 2	Click on submit button after inserting invalid information in the username and password fields.	Shows error and does not log in.
SMMS -071						Step 3	Click on submit button after inserting valid username and password.	Admin gets access in the admin zone with all the options like manage products, shops etc.
SMMS -072						Step 4	Press the log out button	The admin gets the page closed and he is asked for the username and password again.
SMMS -073	M a n u a	f0657bbdf4 7e26ec481f a172b0fa76f 9becb2681	E:\DEVE LOPERS_ ZONE\Git Hub\Shop pingMall	Check Successful product manage in SMMS .	The purpose of this test is to check	Step 1	Click on manage products button and provide data in the text boxes.	Successfully Inserted product.

	1		Managem entSystem \code		product managem ent process.			
SMMS -074						Step 2	See the list view for valid output	All the products with details inserted in the products management are visible in the list view.
SMMS -075						Step 3	Check the assignment of shop names on a product	Products are displayed in the listview along with the shop names where the product is available.
SMMS -076						Step 4	Check the search product option	Should show the name of the product in the list view by which the searching is done.
SMMS -077	M a n u a l	47fb570f63f fec837a49e 235f629c49 cc55a70f0	E:\DEVE LOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check Successful shops managem ent in SMMS .	The purpose of this test is to check shops managem ent process.	Step 1	Click on manage shops button and provide data in the text boxes.	Successfully Inserted shops.
SMMS -078						Step 2	See the list view for valid output.	All the shops with details inserted in the shops management are visible in the list view.

SMMS -079						Step 3	Check the assignment of product names on a shop.	Shops are displayed in the listview along with the product names.
SMMS -080						Step 4	Check the search shop option	Should show the name of the shop in the list view by which the searching is done.
SMMS -081	M a n u a l	b8a65899fa 408a4d99a8 ee8cabbe2a c0b54226b0	E:\DEVELOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check Successful feedback manageme nt in SMMS .	The purpose of this test is to check feedback managem ent process.	Step 1	Check whether all the feedbacks are visible in the listview or not	All the valid feedback data should be visible in the list view.
SMMS -082						Step 2	Check whether all the feedbacks visible could be deleted or not	Selected feedback should get deleted permanently.
SMMS -083	M a n u a l	6c53de9297 011864bd6c 07bdcce834 15a76e8bd4	E:\DEVELOPERS_ ZONE\Git Hub\Shop pingMall Managem entSystem \code	Check everything about contact us.	It is to check that contact us option working properly.	Step 1	Insertion and viewing of contact us.	Contact us listview displays all saved data. Contact could be created successfully.
SMMS -084						Step 2	Select Contact from list view area .And click on Delete it.	Successfully Deleted Contact from list view area.

SMMS -085						Step 3	Select Contact from list view area. And click on Edit button.	Contact ready in Edit field and after editing click on Save button to update the Contact. And Successfully Display edited Contact in list view area.
SMMS -086	M a n u a l	90330b92328d862892fc77436539081cc2b7f70d	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check everything about floor details.	It is to check whether all the functions in floor management system are working or not.	Step 1	Click on floor and see whether each and every tab like first floor, second floor are visible or not.	Should show related data when a user clicks on first floor or second floor or third floor.
SMMS -087						Step 2	Check whether all the products available in the floor are visible or not	In each floor, we could see the names of the products available in it.
SMMS -089						Step 3	Check whether all the shops available in the floor are visible or not	In each floor, we could see the names of the shops available in it.
SMMS -090	M a n u a l	90330b92328d862892fc77436539081cc2b7f70d	E:\DEVELOPERS_ZONE\Git Hub\ShoppingMall ManagementSystem\code	Check everything about shops details.	It is to check that shops details are working properly or not.	Step 1	Click on the shops tab and see all the details.	Displays all saved data. And created shop names.

SMMS -091						Step 2	Whether shop details along with feedback are visible while clicking on a particular shop name or not.	All the data of the selected shop are visible.
SMMS -092						Step 3	Check whether a feedback on a shop could be provided or not.	After entering data on a particular shop, the feedback should be visible in the list view.
SMMS -093	M a n u a l	f3563be0a9 c431104f52 839039e860 43cf640cf1	E:\DEVELOPERS_ ZONE\Git Hub\ShoppingMall Managem entSystem \code	Check everything about products details.	It is to check that products details are working properly or not.	Step 1	Click on the products tab and see all the details.	Displays all saved data. And created product names.
SMMS -094						Step 2	Whether product details along with feedback are visible while clicking on a particular shop name or not.	All the data of the selected product are visible.
SMMS -095						Step 3	Check whether a feedback on a product could be provided or not.	After entering data on a particular product, the feedback should be visible in the list view.

CODING

COMPLETE PROJECT CODING

DESKTOP APPLICATION CODING:

C#.NET

MainWindow.xaml

```
<Window x:Class="ShoppingMall.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="New Mall" WindowState="Maximized" FontFamily="Times New Roman"
        FontSize="20">

    <Window.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="/ControlStyle;component/Commonstyle.xaml" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Window.Resources>

    <DockPanel LastChildFill="True">

        <UniformGrid DockPanel.Dock="Top" Columns="1">
```

```

        <Label Name="lable1" VerticalContentAlignment="Center"
HorizontalContentAlignment="Center" FontFamily="Times New Roman" FontSize="70"
Foreground="#FF0966DF"

        Content="Welcome to the Shoppingmall" FontWeight="Bold">

        <Label.Triggers>

            <EventTrigger RoutedEvent="FrameworkElement.Loaded">

                <BeginStoryboard>

                    <Storyboard>

                        <DoubleAnimation AutoReverse="True" From="2" To="0"
Storyboard.TargetName="lable1" Storyboard.TargetProperty="Opacity"
RepeatBehavior="Forever">

                            </DoubleAnimation>

                        </Storyboard>

                    </BeginStoryboard>

                </EventTrigger>

            </Label.Triggers>

        </Label>

    </UniformGrid>

    <UniformGrid DockPanel.Dock="Top" Rows="1" >

        </UniformGrid>

        <UniformGrid DockPanel.Dock="Top" Margin="2" Height="45">

            <TextBlock FontFamily="Sylfaen" FontWeight="Bold" FontStyle="Normal" >

                <Button Name="product" Style="{StaticResource ControlBtnStyle}"
Click="product_Click" Height="40">Products</Button>

                <Button Name="shop" Style="{StaticResource ControlBtnStyle}"
Click="shop_Click" Height="40">Shops</Button>

                <Button Name="floor" Style="{StaticResource ControlBtnStyle}" Height="40"

```

```

Click="floor_Click">Floor</Button>

        <Button Name="contactusBtn" Style="{StaticResource ControlBtnStyle}"
Click="contactusBtn_Click" Height="40">Contact Us</Button>

        <Button Name="adminZoneobjHL" Style="{StaticResource ControlBtnStyle}"
Click="adminZoneobjHL_Click" Height="40">Admin Zone</Button>

    </TextBlock>

</UniformGrid>

<DockPanel DockPanel.Dock="Bottom" Name="infodocP" LastChildFill="True">

    <UniformGrid DockPanel.Dock="Top">

        <UniformGrid.Background>

            <ImageBrush ImageSource="/ShoppingMallUI;component/Images/shopping-
femmes-hd-fond-d-cran-182156.jpg" />

        </UniformGrid.Background>

    </UniformGrid>

</DockPanel>

</DockPanel>
</Window>

```

MainWindow.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;

```

```
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace ShoppingMall
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();

            private void product_Click(object sender, RoutedEventArgs e)
            {
                ShoppingMall.Products productobj = new ShoppingMall.Products();
                infodocP.Children.Clear();
                infodocP.Children.Add(productobj);

                product.IsEnabled = false;
                shop.IsEnabled = true;
                floor.IsEnabled = true;
                contactusBtn.IsEnabled = true;
                adminZoneobjHL.IsEnabled = true;
            }
        }
    }
}
```

```

        //var bc = new BrushConverter();

        //product.Background = (Brush)bc.ConvertFrom("#FFDFEBF2");

        //product.Foreground = (Brush)bc.ConvertFrom("#FF0966DF");

    }

    private void shop_Click(object sender, RoutedEventArgs e)
    {
        ShoppingMall.Shops shopobj = new ShoppingMall.Shops();
        infodocP.Children.Clear();
        infodocP.Children.Add(shopobj);

        product.IsEnabled = true;
        shop.IsEnabled = false;
        floor.IsEnabled = true;
        contactusBtn.IsEnabled = true;
        adminZoneobjHL.IsEnabled = true;
    }

    private void contactusBtn_Click(object sender, RoutedEventArgs e)
    {
        ShoppingMall.ContactUs ContactUsobj = new ShoppingMall.ContactUs();
        infodocP.Children.Clear();
        infodocP.Children.Add(ContactUsobj);

        product.IsEnabled = true;
        shop.IsEnabled = true;
        floor.IsEnabled = true;
        contactusBtn.IsEnabled = false;
        adminZoneobjHL.IsEnabled = true;
    }

```

```
}

private void adminZoneobjHL_Click(object sender, RoutedEventArgs e)
{
    ShoppingMall.AdminZone AdminZoneobj = new ShoppingMall.AdminZone();
    infodocP.Children.Clear();
    infodocP.Children.Add(AdminZoneobj);

    product.IsEnabled = true;
    shop.IsEnabled = true;
    floor.IsEnabled = true;
    contactusBtn.IsEnabled = true;
    adminZoneobjHL.IsEnabled = false;
}

private void clearshopfeedbackFields()
{
}

private void floor_Click(object sender, RoutedEventArgs e)
{
    ShoppingMall.Floor Floorobj = new ShoppingMall.Floor();
    infodocP.Children.Clear();
    infodocP.Children.Add(Floorobj);

    product.IsEnabled = true;
    shop.IsEnabled = true;
    floor.IsEnabled = false;
    contactusBtn.IsEnabled = true;
    adminZoneobjHL.IsEnabled = true;
}
```

```

    }
}
}

```

AdminZone.xml

```

<UserControl x:Class="ShoppingMall.AdminZone"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    DataContext="{Binding RelativeSource={RelativeSource Self}}"
    mc:Ignorable="d"

    d:DesignHeight="541" d:DesignWidth="851" Style="{DynamicResource
UserCntrlStyle}">

    <UserControl.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="/ControlStyle;component/Commonstyle.xaml" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </UserControl.Resources>

    <DockPanel LastChildFill="True" >

        <DockPanel LastChildFill="True" DockPanel.Dock="Top">

            <Button DockPanel.Dock="Right" Name="adminlogoutBtn" Style="{StaticResource
ControlBtnStyle}" Visibility="Collapsed" Click="adminlogoutBtn_Click">Logout
Admin</Button>

            <Label DockPanel.Dock="left" Background="#FF0966DF" Foreground="#FFFCFDFF"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center">Admin Zone</Label>

```

```

</DockPanel>

<UniformGrid Name="loginUG" Background="White" DockPanel.Dock="Top" Columns="4">
    <Label></Label>
    <Label Style="{StaticResource LblStyle}" >User Name</Label>
    <TextBox Name="adminUserNameTB" Style="{StaticResource
commonTBstyle}"></TextBox>
    <Label></Label>
    <Label></Label>
    <Label Style="{StaticResource LblStyle}" >Password</Label>
    <PasswordBox Name="adminUserPassPb" ></PasswordBox>
    <Label></Label>

    <Label></Label>
    <Button Name="adminresetBtn" Style="{StaticResource ControlBtnStyle}"
Click="adminresetBtn_Click">Reset</Button>
    <Button Name="adminloginBtn" Style="{StaticResource ControlBtnStyle}"
Click="adminloginBtn_Click">Login</Button>
    <Label Name="loginErrorlbl" Style="{StaticResource LblStyle}"></Label>

</UniformGrid>

<UniformGrid Name="logoutUG" DockPanel.Dock="Top" Columns="4">
    <Label></Label>
    <Label></Label>
    <Label></Label>

</UniformGrid>

<UniformGrid DockPanel.Dock="Bottom" Name="manageadminUG" Visibility="Collapsed">

```



```

<TabControl >

    <TabItem Header="Manage Products" Style="{StaticResource TItemStyle}">

        <DockPanel LastChildFill="True">

            <UniformGrid Name="operationProductAdmitnUG"
DockPanel.Dock="Bottom" Rows="1">

                <TextBox Name="prdctAdminSearchTB"></TextBox>

                <Button Name="searchAdminProductBtn" Style="{StaticResource
ControlBtnStyle}" Click="searchAdminProductBtn_Click">Search</Button>

                <Label></Label>

                <Button Name="refreshAdminProductBtn" Style="{StaticResource
ControlBtnStyle}" Click="refreshAdminProductBtn_Click">Refresh</Button>

                <Button Name="editProductBtn" Style="{StaticResource
ControlBtnStyle}" Click="editProductBtn_Click">Edit</Button>

                <Button Name="deleteProduct" Style="{StaticResource
ControlBtnStyle}" Click="deleteProduct_Click">Delete</Button>

            </UniformGrid>

            <UniformGrid Name="manageUG" DockPanel.Dock="Top" >

                <DockPanel LastChildFill="True">

                    <Grid>

                        <Image Name="prdctimgPhoto" Stretch="Uniform"
Height="142" Margin="14,6,14,53" Width="203"></Image>

                        <TextBox Name="prdctimagelinkTB"
Margin="14,154,100,8" Width="117" Height="39"></TextBox>

                        <Button Name="prdctbrowseBtn" Style="{StaticResource
ControlBtnStyle}" Click="prdctbrowseBtn_Click" Margin="137,154,6,8" Height="39"
Width="88">Browse</Button>

                    </Grid>

                    <DockPanel DockPanel.Dock="Right" LastChildFill="True"
Margin="6">

                        <UniformGrid DockPanel.Dock="Top" Columns="2">

                            <Label >Available in :</Label>

```

```

                                <ComboBox Name="avaiableShopCB" IsEditable="True"
Style="{StaticResource comboboxStyle}" ItemsSource="{Binding shopsCollection}"
DisplayMemberPath="name" SelectedValuePath="id" MaxWidth="120" ></ComboBox>

                                <Button Name="deleteAvailShopItemBtn"
Style="{StaticResource ControlBtnStyle}"
Click="deleteAvailShopItemBtn_Click">Delete</Button>

                                <Button Name="addShopToListview"
Style="{StaticResource ControlBtnStyle}" Click="addShopToListview_Click">Add</Button>

                                </UniformGrid>

                                <ListView Name="availableShopLView"
DockPanel.Dock="Bottom" HorizontalAlignment="Stretch" MaxHeight="120"
ScrollViewer.VerticalScrollBarVisibility="Visible">

                                </ListView>

                                </DockPanel>

                                <UniformGrid DockPanel.Dock="Left" Columns="2">

                                <Label Style="{StaticResource LblStyle}"
>Name</Label>

                                <TextBox Name="nameTB"></TextBox>

                                <Label Style="{StaticResource LblStyle}"
>Brand</Label>

                                <TextBox Name="BrandTB"></TextBox>

                                <Label Style="{StaticResource LblStyle}"
>Type</Label>

                                <ComboBox IsEditable="True" Name="ProducttypeCB"
SelectedIndex="1">

                                <ComboBoxItem>Electronics</ComboBoxItem>

                                <ComboBoxItem>Garments</ComboBoxItem>

                                <ComboBoxItem>Computer</ComboBoxItem>

                                <ComboBoxItem>Mobile</ComboBoxItem>

                                <ComboBoxItem>Others</ComboBoxItem>

                                </ComboBox>

                                <Label Style="{StaticResource LblStyle}"
>Description</Label>

```

```

        <TextBox Name="productdescriptionTB"></TextBox>

        <Button Name="resetBtn" Style="{StaticResource
ControlBtnStyle}" Click="resetBtn_Click">Reset</Button>

        <Button Name="submitproductkBtn"
Style="{StaticResource ControlBtnStyle}" Click="submitproductkBtn_Click">Submit</Button>

        <Button Name="editProductkBtn" Style="{StaticResource
ControlBtnStyle}" Visibility="Collapsed" Click="editProductkBtn_Click">Update</Button>

    </UniformGrid>

</DockPanel>

</UniformGrid>

<UniformGrid DockPanel.Dock="Bottom">

    <ListView Name="productsView" HorizontalAlignment="Stretch"
ScrollViewer.VerticalScrollBarVisibility="Visible" ItemsSource="{Binding
productsCollection}" Loaded="Window_Loaded">

        <ListView.View>

            <GridView>

                <GridViewColumn Width="100" Header="Id"
DisplayMemberBinding="{Binding id}" />

                <GridViewColumn Width="200" Header="Name"
DisplayMemberBinding="{Binding name}" />

                <GridViewColumn Width="250" Header="Brand"
DisplayMemberBinding="{Binding brand}" />

                <GridViewColumn Width="110" Header="Type"
DisplayMemberBinding="{Binding type}" />

                <GridViewColumn Width="370" Header="Description"
DisplayMemberBinding="{Binding description}" />

                <GridViewColumn Width="370" Header="Available In"
DisplayMemberBinding="{Binding availableinshop}" />

            </GridView>

        </ListView.View>

    </ListView>

```

```

        </UniformGrid>

    </DockPanel>

</TabItem>

<TabItem Header="Manage Shops" Style="{StaticResource TItemStyle}">

    <DockPanel LastChildFill="True">

        <UniformGrid Name="oprptionShopUG" DockPanel.Dock="Bottom"
Rows="1">

            <TextBox Name="searchAdminShopmangTB"></TextBox>

            <Button Name="searchAdminShopmangBtn" Style="{StaticResource
ControlBtnStyle}" Click="searchAdminShopmangBtn_Click">Search</Button>

            <Label></Label>

            <Button Name="refreshAdminShopBtn" Style="{StaticResource
ControlBtnStyle}" Click="refreshAdminShopBtn_Click">Refresh</Button>

            <Button Name="editShopBtn" Style="{StaticResource
ControlBtnStyle}" Click="editShopBtn_Click">Edit</Button>

            <Button Name="deleteShop" Click="deleteShop_Click"
Style="{StaticResource ControlBtnStyle}">Delete</Button>

        </UniformGrid>

        <DockPanel DockPanel.Dock="Top">

            <Grid>

                <Image Name="shopimgPhoto" Stretch="Uniform"
Height="142" Margin="14,6,14,53" Width="203"></Image>

                <TextBox Name="shopimagelinkTB" Margin="14,154,100,8"
Width="117" Height="39"></TextBox>

                <Button Name="shopbrowseBtn" Style="{StaticResource
ControlBtnStyle}" Click="shopbrowseBtn_Click" Margin="137,154,6,8" Height="39"
Width="88">Browse</Button>

            </Grid>

            <DockPanel DockPanel.Dock="Right" LastChildFill="True"
Margin="6">

                <UniformGrid DockPanel.Dock="Top" Columns="2">

                    <Label >Available Product :</Label>

                    <ComboBox Name="avaiaibleProductCB" IsEditable="True"
Style="{StaticResource comboboxStyle}" ItemsSource="{Binding productsCollection}"
DisplayMemberPath="name" SelectedValuePath="id" MaxWidth="150"></ComboBox>

                    <Button Name="deleteAvailProductItemBtn"

```

```

Style="{StaticResource ControlBtnStyle}"
Click="deleteAvailProductItemBtn_Click">Delete</Button>

        <Button Name="addProductToListview"
Style="{StaticResource ControlBtnStyle}" Click="addProductToListview_Click">Add</Button>

        </UniformGrid>

        <ListView DockPanel.Dock="Bottom" Name="availproductView"
HorizontalAlignment="Stretch" MaxHeight="150"
ScrollViewer.VerticalScrollBarVisibility="Visible">

        </ListView>

</DockPanel>

<UniformGrid Name="manageShopUG" DockPanel.Dock="Top"
Columns="2">

        <Label Style="{StaticResource LblStyle}" >Name</Label>

        <TextBox Name="shopnameTB"></TextBox>

        <Label Style="{StaticResource LblStyle}" >Tag</Label>

        <TextBox Name="shopTagTB"></TextBox>

        <Label Style="{StaticResource LblStyle}" >Type</Label>

        <ComboBox IsEditable="True" Name="shopTypeCB"
SelectedIndex="1">

                <ComboBoxItem>Electronics</ComboBoxItem>

                <ComboBoxItem>Garments</ComboBoxItem>

                <ComboBoxItem>Computer</ComboBoxItem>

                <ComboBoxItem>Mobile</ComboBoxItem>

                <ComboBoxItem>Others</ComboBoxItem>

        </ComboBox>

        <Label Style="{StaticResource LblStyle}" >Available
In</Label>

        <ComboBox IsEditable="True" Name="availableShopFloorCB"
SelectedIndex="1">

                <ComboBoxItem>Ground Floor</ComboBoxItem>

                <ComboBoxItem>First Floor</ComboBoxItem>

                <ComboBoxItem>Second Floor</ComboBoxItem>

        </ComboBox>

        <Label Style="{StaticResource LblStyle}" >Rate (Out of
10)</Label>

```

```

        <TextBox Name="shopRateTB"></TextBox>

        <Label Style="{StaticResource LblStyle}"
>Description</Label>

        <TextBox Name="shopDescriptionTB"></TextBox>

        <Button Name="resetShopmangBtn" Style="{StaticResource
ControlBtnStyle}" Click="resetShopmangBtn_Click">Reset</Button>

        <Button Name="submitkShopmangBtn" Style="{StaticResource
ControlBtnStyle}" Click="submitkShopmangBtn_Click">Submit</Button>

        <Button Name="UpdateShopmangBtn" Style="{StaticResource
ControlBtnStyle}" Visibility="Collapsed" Click="UpdateShopmangBtn_Click">Update</Button>


    </UniformGrid>

</DockPanel>


<UniformGrid DockPanel.Dock="Bottom">

    <ListView Name="shopsView" HorizontalAlignment="Stretch"
ScrollViewer.VerticalScrollBarVisibility="Visible" ItemsSource="{Binding
shopsCollection}" Loaded="Window_Loaded">


        <ListView.View>

            <GridView>

                <GridViewColumn Width="100" Header="Id"
DisplayMemberBinding="{Binding id}" />

                <GridViewColumn Width="200" Header="Name"
DisplayMemberBinding="{Binding name}" />

                <GridViewColumn Width="250" Header="Tag"
DisplayMemberBinding="{Binding tag}" />

                <GridViewColumn Width="110" Header="Type"
DisplayMemberBinding="{Binding type}" />

                <GridViewColumn Width="110" Header="Floor"
DisplayMemberBinding="{Binding availableinfloor}" />

                <GridViewColumn Width="370" Header="Rate"
DisplayMemberBinding="{Binding rating}" />

```

```

                <GridViewColumn Width="370" Header="Description"
DisplayMemberBinding="{Binding description}" />

                <GridViewColumn Width="370" Header="Available In"
DisplayMemberBinding="{Binding availableProduct}" />

            </GridView>

            </ListView.View>

        </ListView>

    </UniformGrid>

</DockPanel>

</TabItem>

<TabItem Header="Manage FeedBack" Style="{StaticResource TItemStyle}">

    <DockPanel LastChildFill="True">

        <UniformGrid Rows="1" DockPanel.Dock="Bottom" >

            <Label></Label>

            <Label></Label>

            <Label></Label>

            <Label></Label>

            <Button Name="refreshFeedbackAdminBtn" Style="{StaticResource
ControlBtnStyle}" Click="refreshFeedbackAdminBtn_Click">Refresh</Button>

            <Button Name="deleteContactusBtn" DockPanel.Dock="Bottom"
Style="{StaticResource ControlBtnStyle}" Click="deleteContactusBtn_Click">Delete</Button>

        </UniformGrid>

        <UniformGrid DockPanel.Dock="Top">

            <ListView Name="contactusView" HorizontalAlignment="Stretch"
ScrollViewer.VerticalScrollBarVisibility="Visible" ItemsSource="{Binding
contactusCollection}" Loaded="Window_Loaded" >

                <ListView.View>

                    <GridView>

                        <GridViewColumn Width="100" Header="Id"

```

```

DisplayMemberBinding="{Binding id}" />
                <GridViewColumn Width="100" Header="Date"
DisplayMemberBinding="{Binding feedDate}" />
                <GridViewColumn Width="200" Header="Name"
DisplayMemberBinding="{Binding name}" />
                <GridViewColumn Width="250" Header="Address"
DisplayMemberBinding="{Binding address}" />
                <GridViewColumn Width="150" Header="Contact No."
DisplayMemberBinding="{Binding mobilenno}" />
                <GridViewColumn Width="200" Header="Email"
DisplayMemberBinding="{Binding email}" />
                <GridViewColumn Width="110" Header="Type"
DisplayMemberBinding="{Binding type}" />
                <GridViewColumn Width="370" Header="Contactus"
DisplayMemberBinding="{Binding contactus}" />

                </GridView>
            </ListView.View>
        </ListView>
    </UniformGrid>
</DockPanel>
</TabItem>

    <TabItem Header="Manage Others" Style="{StaticResource TItemStyle}">

        </TabItem>
    </TabControl>
</UniformGrid>
</DockPanel>
</UserControl>

```


AdminZone.xmal.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using ShoppingMallData;
using ShoppingMallDb;
using System.Collections.ObjectModel;

namespace ShoppingMall
{
    /// <summary>
    /// Interaction logic for AdminZone.xaml
    /// </summary>
    public partial class AdminZone : UserControl
    {
        public AdminZone()
        {
            InitializeComponent();
        }
    }
}
```

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    fetchProductData();
    fetchShopData();
    fetchFeedBackData();
}

private void adminloginBtn_Click(object sender, RoutedEventArgs e)
{
    if (adminUserNameTB.Text.Equals("1") && adminUserPassPb.Password.Equals("1"))
    {
        manageadminUG.Visibility = Visibility.Visible;
        loginUG.Visibility = Visibility.Collapsed;
        adminlogoutBtn.Visibility = Visibility.Visible;
        clearSecurityFields();
    }

    else
    {
        adminUserPassPb.Password = "";
        MessageBox.Show("Please Insert Info Properly");
    }
}

private void adminlogoutBtn_Click(object sender, RoutedEventArgs e)
{
    manageadminUG.Visibility = Visibility.Collapsed;
    loginUG.Visibility = Visibility.Visible;
}
```

```

        adminlogoutBtn.Visibility = Visibility.Collapsed;
        clearSecurityFields();
    }

    private void clearSecurityFields()
    {
        adminUserNameTB.Text = adminUserPassPb.Password = "";
    }

    private void adminresetBtn_Click(object sender, RoutedEventArgs e)
    {
        clearSecurityFields();
    }

    private void clearProductFields()
    {
        nameTB.Text = BrandTB.Text = productdescriptionTB.Text = "";
        ProducttypeCB.SelectedIndex = 1;
    }

    private void resetBtn_Click(object sender, RoutedEventArgs e)
    {
        clearProductFields();
    }

    #region Insert Product
    private void submitproductkBtn_Click(object sender, RoutedEventArgs e)
    {
        string availableinshop = "";

        for (int i = 0; i < availableShopLView.Items.Count; i++)
        {
            availableinshop += availableShopLView.Items[i].ToString() + ", ";
        }
    }

```

```

    }

    if (!nameTB.Text.Equals("") && !BrandTB.Text.Equals("") &&
        !ProducttypeCB.Text.Equals("") && !productdescriptionTB.Text.Equals(""))
    {
        ShoppingMallData.ProductInfo newProduct = new
        ShoppingMallData.ProductInfo();

        newProduct.id = GenerateId();

        newProduct.name = nameTB.Text;
        newProduct.brand = BrandTB.Text;
        newProduct.type = ProducttypeCB.Text;
        newProduct.description = productdescriptionTB.Text;
        newProduct.availableinshop = availableinshop;
        //newProduct.image = shopimgPhoto.BitmapImage;

        ShoppingMallDb.DbInteraction.DoEnterProduct(newProduct);
        clearProductFields();
        fetchProductData();
        //takepic();
    }
    else
    {
        MessageBox.Show("Please Insert Info Properly");
    }
}

```

```

    }

    private string GenerateId()
    {
        return DateTime.Now.ToOADate().ToString();
    }

    //private void takepic()
    //{

        //    MySql.Data.MySqlClient.MySqlConnection msqlConnection = new
        //    MySql.Data.MySqlClient.MySqlConnection("server=localhost; user
        //    id=root;password=technicise;database=shoppingdb;persist security info=false");

        //    if (msqlConnection.State != System.Data.ConnectionState.Open)

        //        msqlConnection.Open();

        //    MySql.Data.MySqlClient.MySqlCommand msqlcommand = new
        //    MySql.Data.MySqlClient.MySqlCommand();

        //    msqlcommand.Connection = msqlConnection;

        //    msqlcommand.CommandText = "insert into product(image)" + "values(@image)";

```

```

//      msqlcommand.Parameters.AddWithValue("@image", prdctimgPhoto);

//      msqlcommand.ExecuteNonQuery();


//      msqlConnection.Close();


//      MessageBox.Show("Info Added");


//}
#endregion

#region Get Product

    ObservableCollection<ProductInfo> _productsCollection = new
ObservableCollection<ProductInfo>();


    public ObservableCollection<ProductInfo> productsCollection
    {
        get
        {
            return _productsCollection;
        }
    }

```

```

private void fetchProductData()
{
    List<ProductInfo> products = DbInteraction.GetAllProductList();

    _productsCollection.Clear();

    foreach (ProductInfo product in products)
    {
        _productsCollection.Add(product);
    }
}

#endregion

#region Insert Shop
private void clearShopFields()
{
    shopnameTB.Text = shopTagTB.Text = shopRateTB.Text = shopDescriptionTB.Text =
"";

    shopTypeCB.SelectedIndex = 1;
}

private void resetShopmangBtn_Click(object sender, RoutedEventArgs e)
{
    clearShopFields();
}

private void submitkShopmangBtn_Click(object sender, RoutedEventArgs e)
{
    string availableProduct = "";

```

```

        for (int i = 0; i < availproductView.Items.Count; i++)
        {
            availableProduct += availproductView.Items[i].ToString() + ", ";
        }

        if (!shopnameTB.Text.Equals("") && !shopTagTB.Text.Equals("") &&
!shopTypeCB.Text.Equals("") && !shopRateTB.Text.Equals("") &&
!shopDescriptionTB.Text.Equals(""))
        {

            ShoppingMallData.ShopInfo newShop = new ShoppingMallData.ShopInfo();

            newShop.id = GenerateId();

            newShop.name = shopnameTB.Text;
            newShop.tag = shopTagTB.Text;
            newShop.type = shopTypeCB.Text;
            newShop.availableinfloor = availableShopFloorCB.Text;
            newShop.rating = shopRateTB.Text;
            newShop.description = shopDescriptionTB.Text;
            newShop.availableProduct = availableProduct;

            ShoppingMallDb.DbInteraction.DoEnterShop(newShop);
            clearShopFields();
            fetchShopData();
        }

        else
        {
            MessageBox.Show("Please Insert Info Properly");
        }

```



```

    }

    #endregion

    #region Get Shop

    ObservableCollection<ShopInfo> _shopsCollection = new
    ObservableCollection<ShopInfo>();

    public ObservableCollection<ShopInfo> shopsCollection
    {
        get
        {
            return _shopsCollection;
        }
    }

    private void fetchShopData()
    {
        List<ShopInfo> shops = DbInteraction.GetAllShopList();

        _shopsCollection.Clear();

        foreach (ShopInfo shop in shops)
        {
            _shopsCollection.Add(shop);
        }
    }

    #endregion

    #region Get FeedBack

```

```

        ObservableCollection<ContactusInfo> _contactusCollection = new
ObservableCollection<ContactusInfo>();

    public ObservableCollection<ContactusInfo> contactusCollection
    {
        get
        {
            return _contactusCollection;
        }
    }

    private void fetchFeedBackData()
    {
        List<ContactusInfo> contactuss = DbInteraction.GetAllContactusList();

        _contactusCollection.Clear();

        foreach (ContactusInfo contactus in contactuss)
        {
            _contactusCollection.Add(contactus);
        }
    }
#endregion

#region Delete Product

    private ProductInfo GetSelectedProductItemforDel()
    {

```

```

        ProductInfo productToDelete = null;

        if (productsView.SelectedIndex == -1)
        {
            MessageBox.Show("Please Select an Item");

        }
        else
        {
            ProductInfo i = (ProductInfo)productsView.SelectedItem;

            productToDelete = _productsCollection.Where(item =>
item.id.Equals(i.id)).First();

        }

        return productToDelete;
    }

    private void deleteProduct_Click(object sender, RoutedEventArgs e)
    {
        ProductInfo productToDelete = GetSelectedProductItemforDel();
        if (productToDelete != null)
        {
            productsCollection.Remove(productToDelete);
            ShoppingMallDb.DbInteraction.DeleteProduct(productToDelete.id);
            fetchProductData();
        }
    }
}

#endregion

```

```

#region Delete Shop

private ShopInfo GetSelectedShopItemforDel()
{

    ShopInfo shopToDelete = null;

    if (shopsView.SelectedIndex == -1)
        MessageBox.Show("Please Select an Item");
    else
    {
        ShopInfo i = (ShopInfo)shopsView.SelectedItem;

        shopToDelete = _shopsCollection.Where(item =>
item.id.Equals(i.id)).First();
    }

    return shopToDelete;
}

private void deleteShop_Click(object sender, RoutedEventArgs e)
{
    ShopInfo shopToDelete = GetSelectedShopItemforDel();
    if (shopToDelete != null)
    {
        shopsCollection.Remove(shopToDelete);
        ShoppingMallDb.DbInteraction.DeleteShop(shopToDelete.id);
        fetchShopData();
    }
}

#endregion

```

```

#region Delete FeedBack

private ContactusInfo GetSelectedContactusItem()
{
    ContactusInfo contactusToDelete = null;
    if (contactusView.SelectedIndex == -1)
        MessageBox.Show("please select an item");
    else
    {
        ContactusInfo i = (ContactusInfo)contactusView.SelectedItem;
        contactusToDelete = _contactusCollection.Where(item =>
item.id.Equals(i.id)).First();
    }
    return contactusToDelete;
}

private void deleteContactusBtn_Click(object sender, RoutedEventArgs e)
{
    ContactusInfo contactusToDelete = GetSelectedContactusItem();
    if (contactusToDelete != null)
    {
        contactusCollection.Remove(contactusToDelete);
        ShoppingMallDb.DbInteraction.DeleteContactus(contactusToDelete.id);
    }
}

#endregion

#region Edit Product

private ProductInfo GetSelectedProductItemForEdit()
{

```

```

        ProductInfo productToDelete = null;

        if (productsView.SelectedIndex == -1)
        {
            MessageBox.Show("Please Select an Item");
        }
        else
        {
            editProductkBtn.Visibility = Visibility.Visible;
            submitproductkBtn.Visibility = Visibility.Collapsed;
            operationProductAdmitnUG.IsEnabled = false;

            ProductInfo i = (ProductInfo)productsView.SelectedItem;

            productToDelete = _productsCollection.Where(item =>
item.id.Equals(i.id)).First();
        }

        return productToDelete;
    }

    private void editProductBtn_Click(object sender, RoutedEventArgs e)
    {
        ProductInfo productToEdit = GetSelectedProductItemForEdit();

        if (productToEdit != null)
        {
            nameTB.Text = productToEdit.name;

```

```

        BrandTB.Text = productToEdit.brand;
        ProducttypeCB.Text = productToEdit.type;
        productdescriptionTB.Text = productToEdit.description;
    }

}

private void editProductkBtn_Click(object sender, RoutedEventArgs e)
{
    if (!nameTB.Text.Equals("") && !BrandTB.Text.Equals("") &&
        !ProducttypeCB.Text.Equals("") && !productdescriptionTB.Text.Equals(""))
    {
        ProductInfo productToEdit = GetSelectedProductItemForEdit();

        productToEdit.name = nameTB.Text;
        productToEdit.brand = BrandTB.Text;
        productToEdit.type = ProducttypeCB.Text;
        productToEdit.description = productdescriptionTB.Text;

        ShoppingMallDb.DbInteraction.EditProduct(productToEdit);

        clearProductFields();
        editProductkBtn.Visibility = Visibility.Collapsed;
        submitproductkBtn.Visibility = Visibility.Visible;
        fetchProductData();

        operationProductAdmitnUG.IsEnabled = true;
    }
    else

```

```

        {
            MessageBox.Show("Please Insert Info Properly");
        }
    }
#endregion

#region Edit Shop

private ShopInfo GetSelectedShopItemforEdit()
{
    ShopInfo shopToDelete = null;

    if (shopsView.SelectedIndex == -1)
        MessageBox.Show("Please Select an Item");
    else
    {
        UpdateShopmangBtn.Visibility = Visibility.Visible;
        submitkShopmangBtn.Visibility = Visibility.Collapsed;
        oprtionShopUG.IsEnabled = false;

        ShopInfo i = (ShopInfo)shopsView.SelectedItem;

        shopToDelete = _shopsCollection.Where(item =>
item.id.Equals(i.id)).First();
    }

    return shopToDelete;
}

```



```

private void editShopBtn_Click(object sender, RoutedEventArgs e)
{
    ShopInfo shopToEdit = GetSelectedShopItemforEdit();

    if (shopToEdit != null)
    {
        shopnameTB.Text = shopToEdit.name;
        shopTagTB.Text = shopToEdit.tag;
        shopTypeCB.Text = shopToEdit.type;
        availableShopFloorCB.Text = shopToEdit.availableinfloor;
        shopRateTB.Text = shopToEdit.rating;
        shopDescriptionTB.Text = shopToEdit.description;
    }
}

private void UpdateShopmangBtn_Click(object sender, RoutedEventArgs e)
{
    if (!shopnameTB.Text.Equals("") && !shopTagTB.Text.Equals("") &&
    !shopTypeCB.Text.Equals("") && !availableShopFloorCB.Text.Equals("") &&
    !shopRateTB.Text.Equals("") && !shopDescriptionTB.Text.Equals(""))
    {
        ShopInfo shopToEdit = GetSelectedShopItemforEdit();

        shopToEdit.name = shopnameTB.Text;
        shopToEdit.tag = shopTagTB.Text;
        shopToEdit.type = shopTypeCB.Text;
        shopToEdit.availableinfloor = availableShopFloorCB.Text;
        shopToEdit.rating = shopRateTB.Text;
        shopToEdit.description = shopDescriptionTB.Text;
    }
}

```

```

        ShoppingMallDb.DbInteraction.EditShop(shopToEdit);

        clearShopFields();
        UpdateShopmangBtn.Visibility = Visibility.Collapsed;
        submitkShopmangBtn.Visibility = Visibility.Visible;
        fetchShopData();

        oprtionShopUG.IsEnabled = true;
    }
    else
    {
        MessageBox.Show("Please Insert Info Properly");
    }
}
#endregion

private void prdctbrowseBtn_Click(object sender, RoutedEventArgs e)
{
    var fd = new Microsoft.Win32.OpenFileDialog();

    fd.Filter = "All image formats (*.jpg; *.jpeg; *.bmp; *.png; *.gif)|*.jpg;*.jpeg;*.bmp;*.png;*.gif";

    var ret = fd.ShowDialog();

    if (ret.GetValueOrDefault())
    {
        prdctimagelinkTB.Text = fd.FileName;

        try
        {

```

```

        BitmapImage bmp = new BitmapImage(new Uri(fd.FileName,
UriKind.Absolute));

        prdctimgPhoto.Source = bmp;
    }
    catch (Exception)
    {
        MessageBox.Show("Invalid image file.", "Browse", MessageBoxButton.OK,
MessageBoxImage.Exclamation);
    }
}

private void shopbrowseBtn_Click(object sender, RoutedEventArgs e)
{
    var fd = new Microsoft.Win32.OpenFileDialog();

    fd.Filter = "All image formats (*.jpg; *.jpeg; *.bmp; *.png;
*.gif)|*.jpg;*.jpeg;*.bmp;*.png;*.gif";

    var ret = fd.ShowDialog();

    if (ret.GetValueOrDefault())
    {
        shopimagelinkTB.Text = fd.FileName;

        try
        {
            BitmapImage bmp = new BitmapImage(new Uri(fd.FileName,
UriKind.Absolute));

            shopimgPhoto.Source = bmp;
        }
        catch (Exception)
        {
            MessageBox.Show("Invalid image file.", "Browse", MessageBoxButton.OK,
MessageBoxImage.Exclamation);
        }
    }
}

```

```

        }

    }

}

#region Search Admin Product

private void searchAdminProductBtn_Click(object sender, RoutedEventArgs e)
{
    if (prdctAdminSearchTB.Text == "")
        fetchProductData();
    else
    {
        ProductInfo prodctInfo = new ProductInfo();
        prodctInfo.name = prdctAdminSearchTB.Text;

        List<ProductInfo> products = DbInteraction.searchProductList(prodctInfo);

        _productsCollection.Clear();

        foreach (ProductInfo product in products)
        {
            _productsCollection.Add(product);
        }
    }
}

#endregion

private void refreshAdminProductBtn_Click(object sender, RoutedEventArgs e)
{
    fetchProductData();
}

```

```
}

#region Search Admin Shop

private void searchAdminShopmangBtn_Click(object sender, RoutedEventArgs e)
{
    if (searchAdminShopmangTB.Text == "")
        fetchShopData();
    else
    {
        ShopInfo shopInfo = new ShopInfo();
        shopInfo.name = searchAdminShopmangTB.Text;

        List<ShopInfo> shops = DbInteraction.searchShopList(shopInfo);

        _shopsCollection.Clear();

        foreach (ShopInfo shop in shops)
        {
            _shopsCollection.Add(shop);
        }
    }
}

#endregion

private void refreshAdminShopBtn_Click(object sender, RoutedEventArgs e)
{
    fetchShopData();
}
```

```

        private void refreshFeedbackAdminBtn_Click(object sender, RoutedEventArgs e)
        {
            fetchFeedBackData();
        }

        private void deleteAvailShopItemBtn_Click(object sender, RoutedEventArgs e)
        {
            availableShopLView.Items.RemoveAt(availableShopLView.Items.IndexOf(availableShopLView.SelectedItem));
        }

        private void addShopToListview_Click(object sender, RoutedEventArgs e)
        {
            availableShopLView.Items.Add(availableShopCB.Text);
        }

        private void addProductToListview_Click(object sender, RoutedEventArgs e)
        {
            availproductView.Items.Add(availableProductCB.Text);
        }

        private void deleteAvailProductItemBtn_Click(object sender, RoutedEventArgs e)
        {
            availproductView.Items.RemoveAt(availproductView.Items.IndexOf(availproductView.SelectedItem));
        }
    }

```

```
}
```

App.xmal

```
<Application x:Class="ShoppingMall.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    StartupUri="MainWindow.xaml">
    <Application.Resources>

    </Application.Resources>
</Application>
```

App.xmal.cs

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Windows;

namespace ShoppingMall
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>

    public partial class App : Application
```

```
{  
  }  
}
```

ContactUs.xmal

```
<UserControl x:Class="ShoppingMall.ContactUs"  
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  
    xmlns:sys="clr-namespace:System;assembly=mscorlib"  
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  
    DataContext="{Binding RelativeSource={RelativeSource Self}}"  
    mc:Ignorable="d"  
    d:DesignHeight="453" d:DesignWidth="785" Style="{DynamicResource  
UserCntrlStyle}">  
  
    <UserControl.Resources>  
        <ResourceDictionary>  
            <ResourceDictionary.MergedDictionaries>  
                <ResourceDictionary Source="/ControlStyle;component/Commonstyle.xaml"/>  
            </ResourceDictionary.MergedDictionaries>  
        </ResourceDictionary>  
    </UserControl.Resources>  
  
    <DockPanel LastChildFill="True" >  
        <Label DockPanel.Dock="Top" Background="#FF0966DF" Foreground="#FFFCFDFF"  
VerticalContentAlignment="Center" HorizontalContentAlignment="Center">Contract Us</Label>  
  
        <UniformGrid Name="upinfo" Background="White" DockPanel.Dock="Top" Columns="4">  
            <Label></Label>
```



```

        <Label Style="{StaticResource LblStyle}" >Today</Label>

        <DatePicker Name="feedDateDp" IsEnabled="False"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
SelectedDate="{x:Static sys:DateTime.Now}" ></DatePicker>

        <Label></Label>

        <Label></Label>

        <Label Style="{StaticResource LblStyle}" >Name</Label>

        <TextBox Name="nameTB"></TextBox>

        <Label></Label>

        <Label></Label>

        <Label Style="{StaticResource LblStyle}" >Address</Label>

        <TextBox Name="addressTB"></TextBox>

        <Label></Label>

        <Label Style="{StaticResource LblStyle}" >Enter your</Label>

        <Label Style="{StaticResource LblStyle}" >Mobile</Label>

        <TextBox Name="mobilenTB"></TextBox>

        <Label></Label>

        <Label Style="{StaticResource LblStyle}" >Feed Back</Label>

        <Label Style="{StaticResource LblStyle}" >Email</Label>

        <TextBox Name="emailTB"></TextBox>

        <Label></Label>

        <Label></Label>

        <Label Style="{StaticResource LblStyle}" >Type</Label>

        <ComboBox IsEditable="True" Name="typeCB" SelectedIndex="1">
            <ComboBoxItem>Guest</ComboBoxItem>
            <ComboBoxItem>Customer</ComboBoxItem>
            <ComboBoxItem>Others</ComboBoxItem>
        </ComboBox>

        <Label></Label>

        <Label></Label>

        <Label Style="{StaticResource LblStyle}" >Complain/Feedback</Label>

```

```

        <TextBox Name="contactusTB"></TextBox>

        <Label></Label>

        <Label></Label>

        <Button Name="resetFdbckBtn" Style="{StaticResource ControlBtnStyle}"
Click="resetFdbckBtn_Click">Reset</Button>

        <Button Name="submitFdBckBtn" Style="{StaticResource ControlBtnStyle}"
Click="submitFdBckBtn_Click">Submit</Button>

        <Label></Label>

    </UniformGrid>

    <UniformGrid DockPanel.Dock="Bottom">

        <ListView Name="contactusView" HorizontalAlignment="Stretch"
ScrollViewer.VerticalScrollBarVisibility="Visible" ItemsSource="{Binding
contactusCollection}" Loaded="Window_Loaded" >

            <ListView.View>

                <GridView>

                    <!--<GridViewColumn Width="100" Header="Id"
DisplayMemberBinding="{Binding id}" />-->

                    <GridViewColumn Width="100" Header="Date"
DisplayMemberBinding="{Binding feedDate}" />

                    <GridViewColumn Width="200" Header="Name"
DisplayMemberBinding="{Binding name}" />

                    <GridViewColumn Width="250" Header="Address"
DisplayMemberBinding="{Binding address}" />

                    <GridViewColumn Width="150" Header="Contact No."
DisplayMemberBinding="{Binding mobileno}" />

                    <GridViewColumn Width="200" Header="Email"
DisplayMemberBinding="{Binding email}" />

                    <GridViewColumn Width="110" Header="Type"
DisplayMemberBinding="{Binding type}" />

                    <GridViewColumn Width="370" Header="Feedback"
DisplayMemberBinding="{Binding feedback}" />

```

```
        </GridView>
    </ListView.View>
</ListView>
</UniformGrid>
</DockPanel>
</UserControl>
```

ContactUs.xmal.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using ShoppingMallData;
using ShoppingMallDb;
using System.Collections.ObjectModel;

namespace ShoppingMall
```

```

{
    /// <summary>
    /// Interaction logic for ContactUs.xaml
    /// </summary>

    public partial class ContactUs : UserControl
    {
        public ContactUs()
        {
            InitializeComponent();
        }

        #region insert FeedBack

        private void submitFdBckBtn_Click(object sender, RoutedEventArgs e)
        {
            if (!nameTB.Text.Equals("") && !addressTB.Text.Equals("") &&
!mobilennoTB.Text.Equals("") && !emailTB.Text.Equals("") && !typeCB.Text.Equals("") &&
!contactusTB.Text.Equals(""))
            {

                ShoppingMallData.ContactusInfo newContactus = new
ShoppingMallData.ContactusInfo();

                newContactus.id = GenerateId();

                newContactus.feedDate = feedDateDp.SelectedDate.Value;
                newContactus.name = nameTB.Text;
                newContactus.address = addressTB.Text;
                newContactus.mobilenno = mobilennoTB.Text;
                newContactus.email = emailTB.Text;
                newContactus.type = typeCB.Text;
                newContactus.feedback = contactusTB.Text;
            }
        }
    }
}

```

```

        ShoppingMallDb.DbInteraction.DoEnterContactus(newContactus);
        clearFeedBackFields();
        fetchFeedBackData();

    }
    else
    {
        MessageBox.Show("Please Insert Info Properly");
    }
}

private string GenerateId()
{
    return DateTime.Now.ToOADate().ToString();
}

private void resetFdbckBtn_Click(object sender, RoutedEventArgs e)
{
    clearFeedBackFields();
}

private void clearFeedBackFields()
{
    nameTB.Text = addressTB.Text = mobilenotTB.Text = emailTB.Text = typeCB.Text =
contactusTB.Text = "";
}

#endregion

ObservableCollection<ContactusInfo> _contactusCollection = new
ObservableCollection<ContactusInfo>();

```

```
public ObservableCollection<ContactusInfo> contactusCollection
{
    get
    {
        return _contactusCollection;
    }
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    fetchFeedBackData();
}

private void fetchFeedBackData()
{
    List<ContactusInfo> contactuss = DbInteraction.GetAllContactusList();

    _contactusCollection.Clear();

    foreach (ContactusInfo contactus in contactuss)
    {
        _contactusCollection.Add(contactus);
    }
}
}
```

Feedback.xmal

```
<UserControl x:Class="ShoppingMall.Feedback"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="300" Style="{DynamicResource
UserCntrlStyle}">
    <UserControl.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="/ControlStyle;component/Commonstyle.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </UserControl.Resources>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"></RowDefinition>
            <RowDefinition Height="*"></RowDefinition>
            <RowDefinition Height="Auto"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"></ColumnDefinition>
            <ColumnDefinition Width="Auto"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <TextBlock Text="Name:" Grid.Row="0" Margin="5"></TextBlock>
        <TextBlock Text="Date:" Grid.Row="0" Grid.Column="1" Margin="5"></TextBlock>
        <TextBlock Text="Feedback:" Grid.Row="1" Margin="5"
ScrollViewer.VerticalScrollBarVisibility="Visible"></TextBlock>
```

```
        <TextBlock Text="Rating:" Grid.Row="2" Margin="5"></TextBlock>

    </Grid>
</UserControl>
```

Feedback.xml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace ShoppingMall
{
    /// <summary>
    /// Interaction logic for Feedback.xaml
    /// </summary>
    public partial class Feedback : UserControl
    {
        public Feedback()
```



```

    {
        InitializeComponent();
    }
}

```

Floor.xaml

```

<UserControl x:Class="ShoppingMall.Floor"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:sys="clr-namespace:System;assembly=mscorlib"
    DataContext="{Binding RelativeSource={RelativeSource Self}}"
    xmlns:my="clr-namespace:ShoppingMall"
    mc:Ignorable="d" Style="{DynamicResource UserCntrlStyle}">

    <UserControl.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="/ControlStyle;component/Commonstyle.xaml" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </UserControl.Resources>

    <DockPanel>

        <Label DockPanel.Dock="Top" Background="#FF0966DF" Foreground="#FFFCFDFF"
            VerticalContentAlignment="Center" HorizontalContentAlignment="Center">Floor</Label>
    </DockPanel>

```

```

<UniformGrid>

    <TabControl >

        <TabItem Header="Ground Floor" Style="{StaticResource TItemStyle}">

            <DockPanel LastChildFill="True">

                <DockPanel DockPanel.Dock="Left" LastChildFill="True">

                    <UniformGrid DockPanel.Dock="Top">

                        <DockPanel LastChildFill="True">

                            <Button Name="refreshgroundShopBtn"
DockPanel.Dock="Right" Style="{StaticResource ControlBtnStyle}"
Click="refreshgroundShopBtn_Click">Refresh</Button>

                            <Button Name="gogroundShopBtn" DockPanel.Dock="Right"
Style="{StaticResource ControlBtnStyle}" >Go</Button>

                            <ComboBox DockPanel.Dock="Left"
Name="groundShopNameSrchCB" Style="{StaticResource comboboxStyle}" ItemsSource="{Binding
shopsCollection}" DisplayMemberPath="name" SelectedValuePath="id"></ComboBox>

                        </DockPanel>

                    </UniformGrid>

                    <UniformGrid DockPanel.Dock="Bottom" Columns="1">

                        <ScrollViewer Width="250">

                            <ListView HorizontalAlignment="Stretch"
Background="#FF0966DF" Foreground="#FFFCFDF" Name="shopDetailsList"
ItemsSource="{Binding shopsCollection}" Loaded="Window_Loaded"
SelectionChanged="ListView_GroundFloorSelectionChanged">

                                <ListView.View>

                                    <GridView>

                                        <GridViewColumn Width="250" Header="Shop
List" DisplayMemberBinding="{Binding name}" />

                                    </GridView>

                                </ListView.View>

                            </ListView>

                        </ScrollViewer>

                    </UniformGrid>

```

```

        </DockPanel>

        <UniformGrid Name="grndflrupinfo" Background="White">
            <DockPanel LastChildFill="True">
                <Label DockPanel.Dock="Top" Background="#FF0966DF"
Foreground="#FFFCFDF" VerticalContentAlignment="Center"
HorizontalContentAlignment="Center">Ground Floor</Label>

                <DockPanel DockPanel.Dock="Right" >
                    <Image DockPanel.Dock="top"
Source="/ShoppingMallUI;component/Images/Shopping-Black-Friday1.jpg" Width="234"
Height="138" />

                </DockPanel>

                <DockPanel LastChildFill="True">
                    <TextBox Name="grndflrshopNameTb"
DockPanel.Dock="Top" HorizontalContentAlignment="Center" IsEnabled="False">Shop
Name</TextBox>

                    <TextBlock Name="grndflrshopdetailsTBlock"
DockPanel.Dock="Top" Style="{StaticResource TextBlockStyle}"
ScrollViewer.VerticalScrollBarVisibility="Visible">Details</TextBlock>

                    <Label DockPanel.Dock="Top" >Available Products
: </Label>

                    <TextBlock Name="availableGroundProductsTBlock"
DockPanel.Dock="Top" Style="{StaticResource TextBlockStyle}"
MouseDown="availableProducts_MouseDown">Available Products</TextBlock>

                    <UniformGrid DockPanel.Dock="Bottom" Columns="1">

                        <ListView Name="grndsfloorhopFeedbackView"
HorizontalAlignment="Stretch" ScrollViewer.VerticalScrollBarVisibility="Visible"
ItemsSource="{Binding shopFeedbackCollection}" Loaded="Window_Loaded" >

                            <ListView.View>
                                <GridView>
                                    <!--<GridViewColumn Width="100"
Header="Id" DisplayMemberBinding="{Binding id}" />-->

                                    <GridViewColumn Width="100"
Header="Date" DisplayMemberBinding="{Binding feedDate}" />
                                </GridView>
                            </ListView.View>
                        </UniformGrid>
                    </DockPanel>
                </UniformGrid>
            </DockPanel>
        </DockPanel>
    </Grid>
</Page>

```

```

                                <GridViewColumn Width="150"
Header="Name" DisplayMemberBinding="{Binding name}" />
                                <GridViewColumn Width="150"
Header="Email" DisplayMemberBinding="{Binding email}" />
                                <GridViewColumn Width="70"
Header="Rating" DisplayMemberBinding="{Binding rate}" />
                                <GridViewColumn Width="370"
Header="Feedback" DisplayMemberBinding="{Binding feedback}" />

                                </GridView>
                                </ListView.View>
                                </ListView>
                                </UniformGrid>
                                </DockPanel>
                                </DockPanel>
                                </UniformGrid>

                                </DockPanel>

                                </TabItem>
                                <TabItem Header="First Floor" Style="{StaticResource TItemStyle}">
                                    <DockPanel LastChildFill="True">

                                        <DockPanel DockPanel.Dock="Left" LastChildFill="True">

                                            <UniformGrid DockPanel.Dock="Top">

                                                <DockPanel LastChildFill="True">

                                                    <Button Name="refreshfirstflrShopBtn"
DockPanel.Dock="Right" Style="{StaticResource ControlBtnStyle}"
Click="refreshfirstflrShopBtn_Click">Refresh</Button>

                                                    <Button Name="gofirstflrShopBtn"
DockPanel.Dock="Right" Style="{StaticResource ControlBtnStyle}" >Go</Button>

                                                    <ComboBox DockPanel.Dock="Left"
Name="firstflrNameSrchCB" Style="{StaticResource comboboxStyle}" ItemsSource="{Binding
shopsCollection}" DisplayMemberPath="name" SelectedValuePath="id"></ComboBox>

```

```

        </DockPanel>

    </UniformGrid>

    <UniformGrid DockPanel.Dock="Bottom" Columns="1">

        <ScrollViewer Width="250">

            <ListView HorizontalAlignment="Stretch"
Background="#FF0966DF" Foreground="#FFFCFDF" Name="firstflrDetailsList"
ItemsSource="{Binding frstfloorshopsCollection}"
SelectionChanged="ListView_FirstFloorSelectionChanged">

                <ListView.View>

                    <GridView>

                        <GridViewColumn Width="250" Header="Shop
List" DisplayMemberBinding="{Binding name}" />

                    </GridView>

                </ListView.View>

            </ListView>

        </ScrollViewer>

    </UniformGrid>

</DockPanel>

<UniformGrid Name="firstflrupinfo" Background="White">

    <DockPanel LastChildFill="True">

        <Label DockPanel.Dock="Top" Background="#FF0966DF"
Foreground="#FFFCFDF" VerticalContentAlignment="Center"
HorizontalContentAlignment="Center">First Floor</Label>

        <DockPanel DockPanel.Dock="Right" >

            <Image DockPanel.Dock="top"
Source="/ShoppingMallUI;component/Images/Shopping-Black-Friday1.jpg" Width="234"
Height="138" />

        </DockPanel>

        <DockPanel LastChildFill="True">

            <TextBox Name="firstflrshopNameTb"
DockPanel.Dock="Top" HorizontalContentAlignment="Center" IsEnabled="False">Shop
Name</TextBox>

            <TextBlock Name="firstflrshopdetailsTBlock"

```

```

DockPanel.Dock="Top" Style="{StaticResource TextBlockStyle}"
ScrollViewer.VerticalScrollBarVisibility="Visible">Details</TextBlock>

<Label DockPanel.Dock="Top" >Available Products
:</Label>

<TextBlock Name="availableFirstProductsTBlock"
DockPanel.Dock="Top" Style="{StaticResource TextBlockStyle}"
MouseDown="availableProducts_MouseDown">Available Products</TextBlock>

<UniformGrid DockPanel.Dock="Bottom" Columns="1">

<ListView Name="firstfloorshopFeedbackView"
HorizontalAlignment="Stretch" ScrollViewer.VerticalScrollBarVisibility="Visible"
ItemsSource="{Binding shopFeedbackCollection}" Loaded="Window_Loaded" >

<ListView.View>

<GridView>

<!--<GridViewColumn Width="100"
Header="Id" DisplayMemberBinding="{Binding id}" />-->

<GridViewColumn Width="100"
Header="Date" DisplayMemberBinding="{Binding feedDate}" />

<GridViewColumn Width="150"
Header="Name" DisplayMemberBinding="{Binding name}" />

<GridViewColumn Width="150"
Header="Email" DisplayMemberBinding="{Binding email}" />

<GridViewColumn Width="70"
Header="Rating" DisplayMemberBinding="{Binding rate}" />

<GridViewColumn Width="370"
Header="Feedback" DisplayMemberBinding="{Binding feedback}" />

</GridView>

</ListView.View>

</ListView>

</UniformGrid>

</DockPanel>

</DockPanel>

</UniformGrid>

```

```

        </DockPanel>

    </TabItem>

    <TabItem Header="Second Floor" Style="{StaticResource TItemStyle}">

        <DockPanel LastChildFill="True">

            <DockPanel DockPanel.Dock="Left" LastChildFill="True">

                <UniformGrid DockPanel.Dock="Top">

                    <DockPanel LastChildFill="True">

                        <Button Name="refreshsecondflrShopBtn"
DockPanel.Dock="Right" Style="{StaticResource ControlBtnStyle}"
Click="refreshsecondflrShopBtn_Click">Refresh</Button>

                        <Button Name="gosecondflrShopBtn"
DockPanel.Dock="Right" Style="{StaticResource ControlBtnStyle}" >Go</Button>

                        <ComboBox DockPanel.Dock="Left"
Name="secondflrNameSrchCB" Style="{StaticResource comboboxStyle}" ItemsSource="{Binding
shopsCollection}" DisplayMemberPath="name" SelectedValuePath="id"></ComboBox>

                    </DockPanel>

                </UniformGrid>

                <UniformGrid DockPanel.Dock="Bottom" Columns="1">

                    <ScrollViewer Width="250">

                        <ListView HorizontalAlignment="Stretch"
Background="#FF0966DF" Foreground="#FFFCFDFF" Name="secondflrDetailsList"
ItemsSource="{Binding secondfloorshopsCollection}"
SelectionChanged="ListView_SecondFloorSelectionChanged">

                            <ListView.View>

                                <GridView >

                                    <GridViewColumn Width="250" Header="Shop
List" DisplayMemberBinding="{Binding name}" />

                                </GridView>

                            </ListView.View>

                        </ListView>

                    </ScrollViewer>

                </UniformGrid>

            </DockPanel>

```

```

        <UniformGrid Name="secondflrupinfo" Background="White">
            <DockPanel LastChildFill="True">
                <Label DockPanel.Dock="Top" Background="#FF0966DF"
Foreground="#FFFCFDF" VerticalContentAlignment="Center"
HorizontalContentAlignment="Center">Second Floor</Label>

                <DockPanel DockPanel.Dock="Right" >

                    <Image DockPanel.Dock="top"
Source="/ShoppingMallUI;component/Images/Shopping-Black-Friday1.jpg" Width="234"
Height="138" />

                </DockPanel>

                <DockPanel LastChildFill="True">

                    <TextBox Name="secondflrshopNameTb"
DockPanel.Dock="Top" HorizontalContentAlignment="Center" IsEnabled="False">Shop
Name</TextBox>

                    <TextBlock Name="secondflrshopdetailsTBlock"
DockPanel.Dock="Top" Style="{StaticResource TextBlockStyle}"
ScrollViewer.VerticalScrollBarVisibility="Visible">Details</TextBlock>

                    <Label DockPanel.Dock="Top" >Available Products
:</Label>

                    <TextBlock Name="availableSecondProductsTBlock"
DockPanel.Dock="Top" Style="{StaticResource TextBlockStyle}"
MouseDown="availableProducts_MouseDown">Available Products</TextBlock>

                    <UniformGrid DockPanel.Dock="Bottom" Columns="1">

                        <ListView Name="secondfloorshopFeedbackView"
HorizontalAlignment="Stretch" ScrollViewer.VerticalScrollBarVisibility="Visible"
ItemsSource="{Binding shopFeedbackCollection}" Loaded="Window_Loaded" >

                            <ListView.View>

                                <GridView>

                                    <!--<GridViewColumn Width="100"
Header="Id" DisplayMemberBinding="{Binding id}" />-->

                                    <GridViewColumn Width="100"
Header="Date" DisplayMemberBinding="{Binding feedDate}" />

                                    <GridViewColumn Width="150"
Header="Name" DisplayMemberBinding="{Binding name}" />

```



```

                                <GridViewColumn Width="150"
Header="Email" DisplayMemberBinding="{Binding email}" />
                                <GridViewColumn Width="70"
Header="Rating" DisplayMemberBinding="{Binding rate}" />
                                <GridViewColumn Width="370"
Header="Feedback" DisplayMemberBinding="{Binding feedback}" />

                                </GridView>
                                </ListView.View>
                                </ListView>
                                </UniformGrid>
                                </DockPanel>
                                </DockPanel>
                                </UniformGrid>

                                </DockPanel>
                                </TabItem>

                                </TabControl>
                                </UniformGrid>

                                </DockPanel>
</UserControl>

```

Floor.xml.cs

```

using System;
using System.Collections.Generic;

```

```
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Collections.ObjectModel;
using ShoppingMallData;
using ShoppingMallDb;
namespace ShoppingMall
{
    /// <summary>
    /// Interaction logic for Floor.xaml
    /// </summary>
    public partial class Floor : UserControl
    {
        public Floor()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            GetSelectedGroundFloorItem();
            //GetSelectedFirstFloorItem();
        }
    }
}
```

```
        //GetSelectedSecondFloorItem();
    }

    #region Get Groundfloor Shop

    ObservableCollection<ShopInfo> _shopsCollection = new
    ObservableCollection<ShopInfo>();

    public ObservableCollection<ShopInfo> shopsCollection
    {
        get
        {
            return _shopsCollection;
        }
    }

    ObservableCollection<FeedbackInfo> _shopFeedbackCollection = new
    ObservableCollection<FeedbackInfo>();

    public ObservableCollection<FeedbackInfo> shopFeedbackCollection
    {
        get
        {
            return _shopFeedbackCollection;
        }
    }

    private void GetSelectedGroundFloorItem()
    {
```

```

        ShopInfo shopInfo = new ShopInfo();
        shopInfo.name = "Ground Floor";

        List<ShopInfo> shops = DbInteraction.getGroundfloorShopList(shopInfo);

        _shopsCollection.Clear();

        foreach (ShopInfo shop in shops)
        {
            _shopsCollection.Add(shop);
        }
    }
#endregion

#region Get First Floor Shop
    ObservableCollection<ShopInfo> _frstfloorshopsCollection = new
ObservableCollection<ShopInfo>();

    public ObservableCollection<ShopInfo> frstfloorshopsCollection
    {
        get
        {
            return _frstfloorshopsCollection;
        }
    }

    private void GetSelectedFirstFloorItem()
    {

```

```

        ShopInfo shopInfo = new ShopInfo();
        shopInfo.name = "First Floor";

        List<ShopInfo> shops = DbInteraction.getFirstFloorShopList(shopInfo);

        _frstfloorshopsCollection.Clear();

        foreach (ShopInfo shop in shops)
        {
            _frstfloorshopsCollection.Add(shop);
        }
    }
#endregion

#region Get Second Floor Shop
    ObservableCollection<ShopInfo> _secondfloorshopsCollection = new
ObservableCollection<ShopInfo>();

    public ObservableCollection<ShopInfo> secondfloorshopsCollection
    {
        get
        {
            return _secondfloorshopsCollection;
        }
    }

    private void GetSelectedSecondFloorItem()
    {

```

```
ShopInfo shopInfo = new ShopInfo();
shopInfo.name = "Second Floor";

List<ShopInfo> shops = DbInteraction.getSecondFloorShopList(shopInfo);

_secondfloorshopsCollection.Clear();

foreach (ShopInfo shop in shops)
{
    _secondfloorshopsCollection.Add(shop);
}
}
#endregion

private void refreshgroundShopBtn_Click(object sender, RoutedEventArgs e)
{
    GetSelectedGroundFloorItem();
}

private void refreshfirstflrShopBtn_Click(object sender, RoutedEventArgs e)
{
    GetSelectedFirstFloorItem();
}

private void refreshsecondflrShopBtn_Click(object sender, RoutedEventArgs e)
{
    GetSelectedSecondFloorItem();
}
```

```

        private void ListView_GroundFloorSelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            if (shopDetailsList.SelectedIndex != -1)
            {
                ShopInfo shopInfoObj =
_shopsCollection.ElementAt(shopDetailsList.SelectedIndex);

                List<ShopInfo> shops = DbInteraction.GetSelectedShopList(shopInfoObj);

                grndflrshopNameTb.Text = shopInfoObj.name;
                grndflrshopdetailsTBlock.Text = shopInfoObj.description;
                availableGroundProductsTBlock.Text = shopInfoObj.availableProduct;

                GetSelectedshopfeedbackGrndflrItem();
            }
            else
            {
                MessageBox.Show("SelectedIndex equals -1");
            }
        }

        private void ListView_FirstFloorSelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            if (firstflrDetailsList.SelectedIndex != -1)
            {
                ShopInfo shopInfoObj =
_shopsCollection.ElementAt(firstflrDetailsList.SelectedIndex);

                List<ShopInfo> shops = DbInteraction.GetSelectedShopList(shopInfoObj);

                firstflrshopNameTb.Text = shopInfoObj.name;
                firstflrshopdetailsTBlock.Text = shopInfoObj.description;
                availableFirstProductsTBlock.Text = shopInfoObj.availableProduct;

                GetSelectedshopfeedbackfrstflrItem();
            }
        }

```

```

    }

    else

        MessageBox.Show("SelectedIndex equals -1");

    }

    private void ListView_SecondFloorSelectionChanged(object sender,
    SelectionChangedEventArgs e)
    {
        if (secondflrDetailsList.SelectedIndex != -1)
        {
            ShopInfo shopInfoObj =
            _shopsCollection.ElementAt(secondflrDetailsList.SelectedIndex);

            List<ShopInfo> shops = DbInteraction.GetSelectedShopList(shopInfoObj);

            secondflrshopNameTb.Text = shopInfoObj.name;
            secondflrshopdetailsTBlock.Text = shopInfoObj.description;
            availableSecondProductsTBlock.Text = shopInfoObj.availableProduct;

            GetSelectedshopfeedbackSecondflrItem();
        }
        else

            MessageBox.Show("SelectedIndex equals -1");
    }

    private void GetSelectedshopfeedbackGrndflrItem()
    {
        FeedbackInfo shopInfo = new FeedbackInfo();

        shopInfo.name = grndflrshopNameTb.Text;

        List<FeedbackInfo> shops = DbInteraction.getshopFeedbackList(shopInfo);

        _shopFeedbackCollection.Clear();
    }

```



```

        foreach (FeedbackInfo shop in shops)
        {
            _shopFeedbackCollection.Add(shop);
        }
    }

    private void GetSelectedshopfeedbackfirstflrItem()
    {
        FeedbackInfo shopInfo = new FeedbackInfo();
        shopInfo.name = firstflrshopNameTb.Text;

        List<FeedbackInfo> shops = DbInteraction.getshopFeedbackList(shopInfo);

        _shopFeedbackCollection.Clear();

        foreach (FeedbackInfo shop in shops)
        {
            _shopFeedbackCollection.Add(shop);
        }
    }

    private void GetSelectedshopfeedbackSecondflrItem()
    {
        FeedbackInfo shopInfo = new FeedbackInfo();
        shopInfo.name = secondflrshopNameTb.Text;

        List<FeedbackInfo> shops = DbInteraction.getshopFeedbackList(shopInfo);
    }

```

```

        _shopFeedbackCollection.Clear();

        foreach (FeedbackInfo shop in shops)
        {
            _shopFeedbackCollection.Add(shop);
        }
    }

    private void availableProducts_MouseDown(object sender, RoutedEventArgs e)
    {
        MessageBox.Show("Go To Product");
    }
}

```

Products.xaml

```

<UserControl x:Class="ShoppingMall.Products"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:sys="clr-namespace:System;assembly=mscorlib"
    DataContext="{Binding RelativeSource={RelativeSource Self}}"
    xmlns:toolkit="http://schemas.microsoft.com/wpf/2008/toolkit"
    xmlns:my="clr-namespace:ShoppingMall"

```

```

        mc:Ignorable="d"

        d:DesignHeight="586" d:DesignWidth="923" FontFamily="Times New Roman"
        FontSize="20" Foreground="#FF318931" Loaded="UserControl_Loaded" >

        <UserControl.Resources>

            <ResourceDictionary>

                <ResourceDictionary.MergedDictionaries>

                    <ResourceDictionary Source="/ControlStyle;component/Commonstyle.xaml" />

                </ResourceDictionary.MergedDictionaries>

            </ResourceDictionary>

        </UserControl.Resources>

        <DockPanel LastChildFill="True" >

            <Label DockPanel.Dock="Top" Background="#FF0966DF" Foreground="#FFFCFDFF"
            VerticalContentAlignment="Center" HorizontalContentAlignment="Center">Products</Label>

            <DockPanel DockPanel.Dock="Left" LastChildFill="True">

                <UniformGrid DockPanel.Dock="Top">

                    <DockPanel LastChildFill="True">

                        <Button Name="refreshProductBtn" DockPanel.Dock="Right"
                        Style="{StaticResource ControlBtnStyle}" Click="refreshProductBtn_Click">Refresh</Button>

                        <Button Name="goProductBtn" DockPanel.Dock="Right"
                        Style="{StaticResource ControlBtnStyle}" Click="goProductBtn_Click">Go</Button>

                        <ComboBox DockPanel.Dock="Left" Name="productNameSrchCB"
                        Style="{StaticResource comboboxStyle}" ItemsSource="{Binding productsCollection}"
                        DisplayMemberPath="name" SelectedValuePath="id"></ComboBox>

                    </DockPanel>

                </UniformGrid>

                <UniformGrid DockPanel.Dock="Bottom" Columns="1">

                    <ScrollViewer Width="250">

                        <ListView Name="productDetailsList" ItemsSource="{Binding
                        productsCollection}" HorizontalAlignment="Stretch" Background="#FF0966DF"
                        Foreground="#FFFCFDFF" SelectionChanged="productDetailsList_SelectionChanged">

                            <ListView.View>

                                <GridView>

                                    <GridViewColumn Width="250" Header="Product Item"
                                    DisplayMemberBinding="{Binding name}" />

```

```

        </GridView>

    </ListView.View>

</ListView>

</ScrollView>

</UniformGrid>

</DockPanel>

<UniformGrid Name="upinfo" Background="White">

    <DockPanel LastChildFill="True">

        <DockPanel DockPanel.Dock="Right" >

            <Image DockPanel.Dock="top"
Source="/ShoppingMallUI;component/Images/Shopping-Black-Friday1.jpg" Width="234"
Height="138" />

            <UniformGrid DockPanel.Dock="Bottom" Columns="1">

                <DatePicker Name="feedDateDp" Visibility="Collapsed"
SelectedDate="{x:Static sys:DateTime.Now}" ></DatePicker>

                <Label Style="{StaticResource LblStyle}"
VerticalContentAlignment="Bottom">Your Name</Label>

                <TextBox Name="nameTb"></TextBox>

                <Label Style="{StaticResource LblStyle}"
VerticalContentAlignment="Bottom">Email</Label>

                <TextBox Name="mailTb"></TextBox>

                <Label Style="{StaticResource LblStyle}"
VerticalContentAlignment="Bottom">Product Rating (Out of 10)</Label>

                <TextBox Name="ratingTb"></TextBox>

                <Label Style="{StaticResource LblStyle}"
VerticalContentAlignment="Bottom">Share your experience with all</Label>

                <TextBox Name="feedbackTb"></TextBox>

                <Label></Label>

            </UniformGrid Rows="1">

                <Button Name="clearProductfeedbackBtn" Style="{StaticResource
ControlBtnStyle}" Click="clearProductfeedbackBtn_Click">Reset</Button>

                <Button Name="prdfdbckbtn" Style="{StaticResource
ControlBtnStyle}" Click="prdfdbckbtn_Click">Submit</Button>

```

```

        </UniformGrid>

    </UniformGrid>

</DockPanel>

<DockPanel LastChildFill="True">

    <TextBox Name="productNameTb" DockPanel.Dock="Top"
HorizontalContentAlignment="Center" IsEnabled="False">Product Name</TextBox>

    <TextBlock Name="productDetailsTBlock" DockPanel.Dock="Top"
Style="{StaticResource TextBlockStyle}">Contactus</TextBlock>

    <Label DockPanel.Dock="Top" >Available In Following Shops : </Label>

    <TextBlock Name="availableInShopTBlock" DockPanel.Dock="Top"
Style="{StaticResource TextBlockStyle}" MouseDown="availableShops_MouseDown">Available In
Shop</TextBlock>

    <UniformGrid DockPanel.Dock="Bottom" Columns="1">

        <ListView Name="productFeedbackView"
HorizontalAlignment="Stretch" ScrollViewer.VerticalScrollBarVisibility="Visible"
ItemsSource="{Binding productFeedbackCollection}" Loaded="Window_Loaded" >

            <ListView.View>

                <GridView>

                    <!--<GridViewColumn Width="100" Header="Id"
DisplayMemberBinding="{Binding id}" />-->

                    <GridViewColumn Width="100" Header="Date"
DisplayMemberBinding="{Binding feedDate}" />

                    <GridViewColumn Width="150" Header="Name"
DisplayMemberBinding="{Binding name}" />

                    <GridViewColumn Width="150" Header="Email"
DisplayMemberBinding="{Binding email}" />

                    <GridViewColumn Width="70" Header="Rating"
DisplayMemberBinding="{Binding rate}" />

                    <GridViewColumn Width="370" Header="Feedback"
DisplayMemberBinding="{Binding feedback}" />

                </GridView>

            </ListView.View>

        </ListView>

```

```
        </UniformGrid>

        </DockPanel>

    </DockPanel>

    </UniformGrid>

</DockPanel>

</UserControl>
```

Products.xmal.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
//using Microsoft.Windows.Controls;
using ShoppingMallData;
using System.Collections.ObjectModel;
using ShoppingMallDb;
```

```
namespace ShoppingMall
{
    /// <summary>
    /// Interaction logic for Products.xaml
    /// </summary>
    public partial class Products : UserControl
    {
        ObservableCollection<ProductInfo> _productsCollection = new
        ObservableCollection<ProductInfo>();

        public ObservableCollection<ProductInfo> productsCollection
        {
            get
            {
                return _productsCollection;
            }
        }

        public Products()
        {
            InitializeComponent();

            //private void babiesproduct_MouseEnter(object sender, MouseEventArgs e)
            //{
            //    ShoppingMall.BabiesProduct bobj = new BabiesProduct();
            //    upinfo.Children.Clear();
            //    upinfo.Children.Add(bobj);
            //}
```

```

    //}

    private void Slider_ValueChanged(
object sender,
RoutedPropertyChangedEventArgs<double> e)
    {
        var slider = sender as Slider;
        var tick = slider.Ticks
            .Where(xx => Math.Abs(e.NewValue - xx) < slider.LargeChange);
        if (tick.Any())
        {
            var newValue = tick.First();
            if (e.NewValue != newValue)
            {
                //DispatcherInvoke(() => slider.Value = newValue);
            }
        }
    }

    private void fetchProductData()
    {
        List<ProductInfo> products = DbInteraction.GetAllProductList();

        _productsCollection.Clear();

        foreach (ProductInfo product in products)
        {
            _productsCollection.Add(product);
        }
    }
}

```



```

private void UserControl_Loaded(object sender, RoutedEventArgs e)
{

    fetchProductData();

}

private void productDetailsList_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{

    if (productDetailsList.SelectedIndex != -1)
    {

        ProductInfo productInfoObj =
_productsCollection.ElementAt(productDetailsList.SelectedIndex);

        List<ProductInfo> products =
DbInteraction.GetSelectedProductList(productInfoObj);

        productNameTb.Text = productInfoObj.name;
        productDetailsTBlock.Text = productInfoObj.description;
        availableInShopTBlock.Text = productInfoObj.availableinshop;

        GetSelectedProductfeedbackItem();

    }
    else
        MessageBox.Show("SelectedIndex equals -1");

}

private void prdfdbckbtn_Click(object sender, RoutedEventArgs e)
{

    if (!nameTb.Text.Equals("") && !mailTb.Text.Equals("") &&
!ratingTb.Text.Equals("") && !feedbackTb.Text.Equals(""))
    {

```

```

        ShoppingMallData.FeedbackInfo newFeedback = new
ShoppingMallData.FeedbackInfo();

        newFeedback.id = GenerateId();

        newFeedback.item = productNameTb.Text;
        newFeedback.feedDate = feedDateDp.SelectedDate.Value;
        newFeedback.name = nameTb.Text;
        newFeedback.email = mailTb.Text;
        newFeedback.rate = ratingTb.Text;
        newFeedback.feedback = feedbackTb.Text;

        ShoppingMallDb.DbInteraction.DoEnterFeedback(newFeedback);

        //fetchFeedBackData();
        clearProductfeedbackFields();
        GetSelectedProductfeedbackItem();
    }
    else
    {
        MessageBox.Show("Please Insert Info Properly");
    }
}

private string GenerateId()
{
    return DateTime.Now.ToOADate().ToString();
}

```

```

private void clearProductfeedbackFields()
{
    nameTb.Text = mailTb.Text = ratingTb.Text = feedbackTb.Text = "";
}

private void clearProductfeedbackBtn_Click(object sender, RoutedEventArgs e)
{
    clearProductfeedbackFields();
}

private void goProductBtn_Click(object sender, RoutedEventArgs e)
{
    if (productNameSrchCB.Text == "")
        fetchProductData();
    else
    {
        ProductInfo prodctInfo = new ProductInfo();
        prodctInfo.name = productNameSrchCB.Text;

        List<ProductInfo> products = DbInteraction.searchProductList(prodctInfo);

        _productsCollection.Clear();

        foreach (ProductInfo product in products)
        {
            _productsCollection.Add(product);
        }
    }
}

```

```

private void refreshProductBtn_Click(object sender, RoutedEventArgs e)
{
    fetchProductData();
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    //fetchshopFeedbackData();
}

#region Display only selected product Feedback

    ObservableCollection<FeedbackInfo> _productFeedbackCollection = new
ObservableCollection<FeedbackInfo>();

public ObservableCollection<FeedbackInfo> productFeedbackCollection
{
    get
    {
        return _productFeedbackCollection;
    }
}

private void GetSelectedProductfeedbackItem()
{
    FeedbackInfo productInfo = new FeedbackInfo();

    productInfo.name = productNameTb.Text;

```

```
List<FeedbackInfo> products =  
DbInteraction.getProductFeedbackList(productInfo);
```

```
_productFeedbackCollection.Clear();
```

```
foreach (FeedbackInfo product in products)  
{  
    _productFeedbackCollection.Add(product);  
}
```

```
}
```

```
#endregion
```

```
private void availableShops_MouseDown(object sender, RoutedEventArgs e)
```

```
{
```

```
    MessageBox.Show("Go To shop");
```

```
}
```

```
}
```

```
}
```

Shops.xml

```

<UserControl x:Class="ShoppingMall.Shops"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:sys="clr-namespace:System;assembly=mscorlib"
    xmlns:my="clr-namespace:ShoppingMall"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    DataContext="{Binding RelativeSource={RelativeSource Self}}"
    mc:Ignorable="d"

    d:DesignHeight="548" d:DesignWidth="882" FontFamily="Times New Roman"
    FontSize="20" Foreground="#FF318931" Loaded="UserControl_Loaded">

    <UserControl.Resources>

        <ResourceDictionary>

            <ResourceDictionary.MergedDictionaries>

                <ResourceDictionary Source="/ControlStyle;component/Commonstyle.xaml" />

            </ResourceDictionary.MergedDictionaries>

        </ResourceDictionary>

    </UserControl.Resources>

    <DockPanel LastChildFill="True">

        <Label DockPanel.Dock="Top" Background="#FF0966DF" Foreground="#FFFCFDFF"
        VerticalContentAlignment="Center" HorizontalContentAlignment="Center">Shops</Label>

        <DockPanel DockPanel.Dock="Left" LastChildFill="True">

            <UniformGrid DockPanel.Dock="Top">

                <DockPanel LastChildFill="True">

                    <Button Name="refreshShopBtn" DockPanel.Dock="Right"
                    Style="{StaticResource ControlBtnStyle}" Click="refreshShopBtn_Click">Refresh</Button>

                    <Button Name="goShopBtn" DockPanel.Dock="Right"
                    Style="{StaticResource ControlBtnStyle}" Click="goShopBtn_Click">Go</Button>

                    <ComboBox DockPanel.Dock="Left" Name="shopNameSrchCB"
                    Style="{StaticResource comboboxStyle}" ItemsSource="{Binding shopsCollection}"
                    DisplayMemberPath="name" SelectedValuePath="id"></ComboBox>

                </DockPanel>

```

```

        </UniformGrid>

        <UniformGrid DockPanel.Dock="Bottom" Columns="1">

            <ScrollViewer Width="250">

                <ListView HorizontalAlignment="Stretch" Background="#FF0966DF"
Foreground="#FFFCFDFF" Name="shopDetailsList" ItemsSource="{Binding shopsCollection}"
Loaded="ListView_Loaded" SelectionChanged="ListView_SelectionChanged">

                    <ListView.View>

                        <GridView>

                            <GridViewColumn Width="250" Header="Shop List"
DisplayMemberBinding="{Binding name}" />

                        </GridView>

                    </ListView.View>

                </ListView>

            </ScrollViewer>

        </UniformGrid>
    </DockPanel>

    <UniformGrid Name="upinfo" Background="White">

        <DockPanel LastChildFill="True">

            <DockPanel DockPanel.Dock="Right" >

                <Image DockPanel.Dock="top"
Source="/ShoppingMallUI;component/Images/Shopping-Black-Friday1.jpg" Width="234"
Height="138" />

                <UniformGrid DockPanel.Dock="Bottom" Columns="1">

                    <DatePicker Name="feedDateDp" Visibility="Collapsed"
SelectedDate="{x:Static sys:DateTime.Now}" ></DatePicker>

                    <Label Style="{StaticResource LblStyle}"
VerticalContentAlignment="Bottom">Your Name</Label>

                    <TextBox Name="nameTb"></TextBox>

                    <Label Style="{StaticResource LblStyle}"
VerticalContentAlignment="Bottom">Email</Label>

                    <TextBox Name="mailTb"></TextBox>

```

```

        <Label Style="{StaticResource LblStyle}"
VerticalContentAlignment="Bottom">Shop Rating (Out of 10)</Label>

        <TextBox Name="ratingTb"></TextBox>

        <Label Style="{StaticResource LblStyle}"
VerticalContentAlignment="Bottom">Share your experience with all</Label>

        <TextBox Name="feedbackTb"></TextBox>

        <Label></Label>

        <UniformGrid Rows="1">

            <Button Name="resetshopFeedback" Style="{StaticResource
ControlBtnStyle}" Click="resetshopFeedback_Click">Reset</Button>

            <Button Name="submitshopBtn" Style="{StaticResource
ControlBtnStyle}" Click="submitshopBtn_Click">Submit</Button>

        </UniformGrid>

    </UniformGrid>

</DockPanel>

    <DockPanel LastChildFill="True">

        <TextBox Name="shopNameTb" DockPanel.Dock="Top"
HorizontalContentAlignment="Center" IsEnabled="False">Shop Name</TextBox>

        <TextBlock Name="shopdetailsTBlock" DockPanel.Dock="Top"
Style="{StaticResource TextBlockStyle}"
ScrollViewer.VerticalScrollBarVisibility="Visible">Details</TextBlock>

        <Label DockPanel.Dock="Top" >Available Products :</Label>

        <TextBlock Name="availableProductsTBlock" DockPanel.Dock="Top"
Style="{StaticResource TextBlockStyle}" MouseDown="availableProducts_MouseDown">Available
Products</TextBlock>

        <UniformGrid DockPanel.Dock="Bottom" Columns="1">

            <ListView Name="shopFeedbackView"
HorizontalContentAlignment="Stretch" ScrollViewer.VerticalScrollBarVisibility="Visible"
ItemsSource="{Binding shopFeedbackCollection}" Loaded="Window_Loaded" >

                <ListView.View>

                    <GridView>

                        <!--<GridViewColumn Width="100" Header="Id"
DisplayMemberBinding="{Binding id}" />-->

                        <GridViewColumn Width="100" Header="Date"

```



```

DisplayMemberBinding="{Binding feedDate}" />
                                <GridViewColumn Width="150" Header="Name"
DisplayMemberBinding="{Binding name}" />
                                <GridViewColumn Width="150" Header="Email"
DisplayMemberBinding="{Binding email}" />
                                <GridViewColumn Width="70" Header="Rating"
DisplayMemberBinding="{Binding rate}" />
                                <GridViewColumn Width="370" Header="Feedback"
DisplayMemberBinding="{Binding feedback}" />

                                </GridView>
                                </ListView.View>
                                </ListView>
                                </UniformGrid>

                                </DockPanel>
                                </DockPanel>
                                </UniformGrid>

                                </DockPanel>

</UserControl>

```

Shops.xmlal.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;

```

```
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Collections.ObjectModel;
using ShoppingMallData;
using ShoppingMallDb;

namespace ShoppingMall
{
    /// <summary>
    /// Interaction logic for Shops.xaml
    /// </summary>
    public partial class Shops : UserControl
    {
        ObservableCollection<ShopInfo> _shopsCollection = new
        ObservableCollection<ShopInfo>();

        public ObservableCollection<ShopInfo> shopsCollection
        {
            get
            {
                return _shopsCollection;
            }
        }

        public Shops()
        {

```

```

        InitializeComponent();
    }

    private void ListView_Loaded(object sender, RoutedEventArgs e)
    {

    }

    private void fetchShopData()
    {
        List<ShopInfo> shops = DbInteraction.GetAllShopList();

        _shopsCollection.Clear();

        foreach (ShopInfo shop in shops)
        {
            _shopsCollection.Add(shop);
        }
    }

    private void ListView_SelectionChanged(object sender, SelectionChangedEventArgs
e)
    {
        if (shopDetailsList.SelectedIndex != -1)
        {
            ShopInfo shopInfoObj =
            _shopsCollection.ElementAt(shopDetailsList.SelectedIndex);

            List<ShopInfo> shops = DbInteraction.GetSelectedShopList(shopInfoObj);

            shopNameTb.Text = shopInfoObj.name;
        }
    }

```

```

        shopdetailsTBlock.Text = shopInfoObj.description;
        availableProductsTBlock.Text = shopInfoObj.availableProduct;

        GetSelectedshopfeedbackItem();
    }
    else
        MessageBox.Show("SelectedIndex equals -1");

}

private void submitshopBtn_Click(object sender, RoutedEventArgs e)
{

    if (!nameTb.Text.Equals("") && !mailTb.Text.Equals("") &&
!ratingTb.Text.Equals("") && !feedbackTb.Text.Equals(""))
    {
        ShoppingMallData.FeedbackInfo newFeedback = new
ShoppingMallData.FeedbackInfo();

        newFeedback.id = GenerateId();

        newFeedback.item = shopNameTb.Text;
        newFeedback.feedDate = feedDateDp.SelectedDate.Value;
        newFeedback.name = nameTb.Text;
        newFeedback.email = mailTb.Text;
        newFeedback.rate = ratingTb.Text;
        newFeedback.feedback = feedbackTb.Text;

        ShoppingMallDb.DbInteraction.DoEnterFeedback(newFeedback);
    }
}

```

```

        clearshopfeedbackFields();
        //fetchFeedBackData();

        GetSelectedshopfeedbackItem();
    }
    else
    {
        MessageBox.Show("Please Insert Info Properly");
    }
}

private string GenerateId()
{
    return DateTime.Now.ToOADate().ToString();
}

private void clearshopfeedbackFields()
{
    nameTb.Text = mailTb.Text = ratingTb.Text = feedbackTb.Text = "";
}

private void resetshopFeedback_Click(object sender, RoutedEventArgs e)
{
    clearshopfeedbackFields();
}

private void refreshShopBtn_Click(object sender, RoutedEventArgs e)
{
    fetchShopData();
}

private void goShopBtn_Click(object sender, RoutedEventArgs e)

```

```

{
    //shopDetailsList.UnselectAll();
    if (shopNameSrchCB.Text == "")
        fetchShopData();
    else
    {
        ShopInfo shopInfo = new ShopInfo();
        shopInfo.name = shopNameSrchCB.Text;

        List<ShopInfo> shops = DbInteraction.searchShopList(shopInfo);

        _shopsCollection.Clear();

        foreach (ShopInfo shop in shops)
        {
            _shopsCollection.Add(shop);
        }
    }
}

private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
    fetchShopData();
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{

```

```
}
```

```
#region Display only selected Shop Feedback
```

```
    ObservableCollection<FeedbackInfo> _shopFeedbackCollection = new  
ObservableCollection<FeedbackInfo>();
```

```
public ObservableCollection<FeedbackInfo> shopFeedbackCollection  
{  
    get  
    {  
        return _shopFeedbackCollection;  
    }  
}
```

```
private void GetSelectedshopfeedbackItem()
```

```
{
```

```
    FeedbackInfo shopInfo = new FeedbackInfo();
```

```
    shopInfo.name = shopNameTb.Text;
```

```
    List<FeedbackInfo> shops = DbInteraction.getshopFeedbackList(shopInfo);
```

```
    _shopFeedbackCollection.Clear();
```

```
    foreach (FeedbackInfo shop in shops)
```

```

        {
            _shopFeedbackCollection.Add(shop);
        }
    }
    #endregion

    private void availableProducts_MouseDown(object sender, RoutedEventArgs e)
    {
        MessageBox.Show("Go To Product");
    }

}
}

```

AssemblyInfo.cs

```

using System.Reflection;
using System.Resources;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Windows;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.

```



```

[assembly: AssemblyTitle("ShoppingMall")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("ShoppingMall")]
[assembly: AssemblyCopyright("Copyright © 2013")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

//In order to begin building localizable applications, set
//<UICulture>CultureYouAreCodingWith</UICulture> in your .csproj file
//inside a <PropertyGroup>.  For example, if you are using US english
//in your source files, set the <UICulture> to en-US.  Then uncomment
//the NeutralResourceLanguage attribute below.  Update the "en-US" in
//the line below to match the UICulture setting in the project file.

//[assembly: NeutralResourcesLanguage("en-US",
UltimateResourceFallbackLocation.Satellite)]


[assembly: ThemeInfo(
    ResourceDictionaryLocation.None, //where theme specific resource dictionaries are
    located
    //(used if a resource is not found in the page,
    // or application resource dictionaries)
    ResourceDictionaryLocation.SourceAssembly //where the generic resource dictionary is
    located

```

```

        //(used if a resource is not found in the page,
        // app, or any theme specific resource dictionaries)
    )]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]

```

Resources.Designer.cs

```

//-----
// <auto-generated>
//      This code was generated by a tool.
//      Runtime Version:4.0.30319.1
//
//      Changes to this file may cause incorrect behavior and will be lost if
//      the code is regenerated.
// </auto-generated>

```

```
//-----

namespace ShoppingMall.Properties
{

    /// <summary>
    ///   A strongly-typed resource class, for looking up localized strings, etc.
    /// </summary>

    // This class was auto-generated by the StronglyTypedResourceBuilder
    // class via a tool like ResGen or Visual Studio.
    // To add or remove a member, edit your .ResX file then rerun ResGen
    // with the /str option, or rebuild your VS project.

    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "4.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
    internal class Resources
    {

        private static global::System.Resources.ResourceManager resourceMan;

        private static global::System.Globalization.CultureInfo resourceCulture;

        [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]
        internal Resources()
        {
        }
    }
}
```

```

    /// <summary>

    ///     Returns the cached ResourceManager instance used by this class.

    /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]

    internal static global::System.Resources.ResourceManager ResourceManager
    {
        get
        {
            if ((resourceMan == null))
            {
                global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("ShoppingMall.Properties.Resources",
typeof(Resources).Assembly);

                resourceMan = temp;
            }
            return resourceMan;
        }
    }

    /// <summary>

    ///     Overrides the current thread's CurrentUICulture property for all
    ///     resource lookups using this strongly typed resource class.

    /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]

    internal static global::System.Globalization.CultureInfo Culture
    {
        get
        {
            return resourceCulture;
        }
    }

```

```

        }
        set
        {
            resourceCulture = value;
        }
    }
}

```

Setting.cs

```

namespace ShoppingMall.Properties {

    // This class allows you to handle specific events on the settings class:
    //  The SettingChanging event is raised before a setting's value is changed.
    //  The PropertyChanged event is raised after a setting's value is changed.
    //  The SettingsLoaded event is raised after the setting values are loaded.
    //  The SettingsSaving event is raised before the setting values are saved.
    internal sealed partial class Settings {

        public Settings() {
            // // To add event handlers for saving and changing settings, uncomment the
lines below:

            //
            // this.SettingChanging += this.SettingChangingEventHandler;
            //
            // this.SettingsSaving += this.SettingsSavingEventHandler;
            //
        }
    }
}

```

```

    }

    private void SettingChangingEventHandler(object sender,
System.Configuration.SettingChangingEventArgs e) {

        // Add code to handle the SettingChangingEvent event here.

    }

    private void SettingsSavingEventHandler(object sender,
System.ComponentModel.CancelEventArgs e) {

        // Add code to handle the SettingsSaving event here.

    }

}

```

Setting.Designer.cs

```

//-----
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.1
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//-----

namespace ShoppingMall.Properties
{

```

```
[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator", "10.0.0.0")]

internal sealed partial class Settings :
global::System.Configuration.ApplicationSettingsBase
{

    private static Settings defaultInstance =
((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new
Settings())));

    public static Settings Default
    {
        get
        {
            return defaultInstance;
        }
    }
}
```

Index.xml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>SMMSmob</title>
  <link href="s40-theme/css/s40-theme.css" rel="stylesheet" type="text/css" />
  <script language="javascript" type="text/javascript" src="s40-theme/js/screensize.js"></script>
  <script type="text/javascript">
    function refreshPageContent() {
      // Add code for refreshing the page here...
    }
  </script>
  <link rel="stylesheet" type="text/css" href="s40-theme/css/s40-theme_labeled_text_field.css" />
  <link rel="stylesheet" type="text/css" href="s40-theme/css/s40-theme_button.css" />
  <link rel="stylesheet" type="text/css" href="s40-theme/css/s40-theme_text_field_with_button.css" />
  <link rel="stylesheet" type="text/css" href="s40-theme/css/s40-theme_category_list.css" />
  <link rel="stylesheet" type="text/css" href="s40-theme/css/s40-theme_category_list_item.css" />
  <link rel="stylesheet" type="text/css" href="s40-theme/css/s40-theme_list.css" />
</head>
<body>
  <div class="ui-page">
    <!-- header -->
    <div class="ui-header">
      <div class="ui-title inline"><h5>Shopping Management System</h5></div><div class="refresh-icon inline">
        <a onclick="refreshPageContent();"><img alt="icon" src="s40-
```



```
theme/images/refresh_40x40.png"/></a></div>
```

```
</div>
```

```
<div class="ui-content">
```

```
    <div class="ui-tab-control inline">
```

```
        <div class="ui-tab ui-tab-selected inline" id="tab_1"
onclick="mwl.setGroupTarget('#tab_control_content', '#tab_1_content', 'ui-show', 'ui-
hide');mwl.switchClass('#tab_1', 'ui-tab-not-selected', 'ui-tab-selected'); mwl.switchClass('#tab_2', 'ui-tab-
selected', 'ui-tab-not-selected');mwl.switchClass('#tab_3', 'ui-tab-selected', 'ui-tab-not-selected');">Admission
```

```
    </div><div class="ui-tab ui-tab-not-selected inline" id="tab_2"
onclick="mwl.setGroupTarget('#tab_control_content', '#tab_2_content', 'ui-show', 'ui-
hide');mwl.switchClass('#tab_1', 'ui-tab-selected', 'ui-tab-not-selected');mwl.switchClass('#tab_2', 'ui-tab-not-
selected', 'ui-tab-selected');mwl.switchClass('#tab_3', 'ui-tab-selected', 'ui-tab-not-selected');">Student
```

```
    </div><div class="ui-tab ui-tab-not-selected inline" id="tab_3"
onclick="mwl.setGroupTarget('#tab_control_content', '#tab_3_content', 'ui-show', 'ui-
hide');mwl.switchClass('#tab_1', 'ui-tab-selected', 'ui-tab-not-selected');mwl.switchClass('#tab_2', 'ui-tab-selected',
'ui-tab-not-selected');mwl.switchClass('#tab_3', 'ui-tab-not-selected', 'ui-tab-selected');">Course</div>
```

```
</div>
```

```
    <div class="ui-tab-content-row" id="tab_control_content">
```

```
        <div id="tab_1_content" class="ui-show">
```

```
            <!-- label-with-text-field -->
```

```
            <div class="ui-fieldset_S40labeledtextfield">
```

```
                <div class="ui-legend_S40labeledtextfield">First Name:</div>
```

```
                <input type="text" name="name" class="ui-text-input_S40labeledtextfield"/>
```

```
                <div class="ui-legend_S40labeledtextfield">Last Name:</div>
```

```
                <input type="text" name="name" class="ui-text-input_S40labeledtextfield"/>
```

```
                <div class="ui-legend_S40labeledtextfield">Address:</div>
```

```
                <input type="text" name="name" class="ui-text-input_S40labeledtextfield"/>
```

```
                <div class="ui-legend_S40labeledtextfield">Std:</div>
```

```
                <input type="text" name="name" class="ui-text-input_S40labeledtextfield"/>
```

```
            <br /><br /><button name="button1" class="ui-button_S40button">Submit</button>
```

```
        </div>
```

```
    </div>
```

```
    <div id="tab_2_content" class="ui-hide">
```

```

        <!-- label-with-text-field -->

        <div class="ui-fieldset_S40labeledtextfield">

            <div class="ui-legend_S40labeledtextfield">User Name:</div>

            <input type="text" name="name" class="ui-text-input_S40labeledtextfield"/>

            <div class="ui-legend_S40labeledtextfield">Password</div>

            <input type="text" name="name" class="ui-text-input_S40labeledtextfield"/>

            <button name="submit" class="ui-button_S40button">Enter</button>

        </div>

        <!-- category-list -->

        <div class="ui-category-list_S40categorylist">

            <!-- list-item-1 -->

            <div class="ui-category-list-item_S40categorylist">

                <div class="ui-category-list-item-title_S40categorylist ui-
open_S40categorylist" id="category_title_1_1" onclick="mwl.toggleClass('#category_items_1_1', 'ui-
hide_S40categorylist');mwl.toggleClass('#category_items_1_1', 'ui-
show_S40categorylist');mwl.toggleClass('#category_title_1_1', 'ui-
open_S40categorylist');mwl.toggleClass('#category_title_1_1', 'ui-close_S40categorylist');">Student Details</div>

                <div class="ui-category-list-item-body_S40categorylist ui-
show_S40categorylist" id="category_items_1_1">

                    <!-- list-items -->

                    <div class="ui-list_S40list">

                        <div class="ui-list-item_S40list">Name:Joysankar Sengupta</div>

                        <div class="ui-list-item_S40list">Class:X</div>

                        <div class="ui-list-item_S40list">Roll No:32</div>

                        <div class="ui-list-item_S40list">Attendance:72%</div>

                    </div>

                    </div>

                </div>

            <!-- list-item-2 -->

            <div class="ui-category-list-item_S40categorylist">

                <div class="ui-category-list-item-title_S40categorylist ui-close_S40categorylist"

```

```

id="category_title_2_1" onclick="mwl.toggleClass('#category_items_2_1', 'ui-
show_S40categorylist');mwl.toggleClass('#category_items_2_1', 'ui-
hide_S40categorylist');mwl.toggleClass('#category_title_2_1', 'ui-
close_S40categorylist');mwl.toggleClass('#category_title_2_1', 'ui-open_S40categorylist');">Result</div>

    <div class="ui-category-list-item-body_S40categorylist ui-hide_S40categorylist"
id="category_items_2_1">

        <!-- list-items -->

<div class="ui-list_S40list">

    <div class="ui-list-item_S40list">Total Marks Obtained:756</div>

    <div class="ui-list-item_S40list">Division:1st</div>

    <div class="ui-list-item_S40list">Status:Promoted</div>

</div>

    </div>

</div>

</div>

    </div>

    <div id="tab_3_content" class="ui-hide">

        <!-- category-list-item -->

        <div class="ui-category-list-item_S40categorylistitem">

            <div class="ui-category-list-item-title_S40categorylistitem ui-open_S40categorylistitem" id="category_title_3_2"
onclick="mwl.toggleClass('#category_items_3_2', 'ui-
hide_S40categorylistitem');mwl.toggleClass('#category_items_3_2', 'ui-
show_S40categorylistitem');mwl.toggleClass('#category_title_3_2', 'ui-
open_S40categorylistitem');mwl.toggleClass('#category_title_3_2', 'ui-close_S40categorylistitem');">Regular
Course</div>

            <div class="ui-category-list-item-body_S40categorylistitem ui-show_S40categorylistitem"
id="category_items_3_2">

                <!-- list-items -->

<div class="ui-list_S40list">

    <div class="ui-list-item_S40list">Science(Class XI-XII)</div>

    <div class="ui-list-item_S40list">Arts(Class XI-XII)</div>

    <div class="ui-list-item_S40list">Class V-X</div>

</div>

    </div>

```

```

        </div>

        <!-- category-list-item -->
<div class="ui-category-list-item_S40categorylistitem">

    <div class="ui-category-list-item-title_S40categorylistitem ui-open_S40categorylistitem" id="category_title_3_3"
onclick="mwl.toggleClass('#category_items_3_3', 'ui-
hide_S40categorylistitem');mwl.toggleClass('#category_items_3_3', 'ui-
show_S40categorylistitem');mwl.toggleClass('#category_title_3_3', 'ui-
open_S40categorylistitem');mwl.toggleClass('#category_title_3_3', 'ui-
close_S40categorylistitem');">Vocational</div>

    <div class="ui-category-list-item-body_S40categorylistitem ui-show_S40categorylistitem"
id="category_items_3_3">

        <!-- list-items -->
<div class="ui-list_S40list">

    <div class="ui-list-item_S40list">Basic Computer</div>

    <div class="ui-list-item_S40list">Drawing</div>

    <div class="ui-list-item_S40list">Others</div>
</div>

</div>

</div>

        </div>

        </div>

        </div>

    </div>

</body>
</html>

```

COMMENTS AND DESCRIPTION OF CODING SEGMENTS

CODE COMMENTING

- All comments have been written in the same language, be grammatically correct, and contain appropriate punctuation.
- Used `//` or `///` but never `/* ... */`
- Did not “flowerbox” comment blocks.

Example:

```
// *****
// Comment block
// *****
```

- Always Used inline-comments to explain assumptions, known issues, and algorithm insights.
- Never used inline-comments to explain obvious code. Well written code is self documenting.
- Only used comments for bad code to say “fix this code” – otherwise remove, or rewrite the code!
- Included comments using Task-List keyword flags to allow comment-filtering.

Example:

```
// TODO: Place Database Code Here
// UNDONE: Removed P\Invoke Call due to errors
// HACK: Temporary fix until able to refactor
```

- Always applied C# comment-blocks (`///`) to public, protected, and internal declarations.
- Only used C# comment-blocks for documenting the API.
- Always included `<summary>` comments. Include `<param>`, `<return>`, and `<exception>` comment sections where applicable.
- Included `<see cref="" />` and `<seealso cref="" />` where possible.
- Always added CDATA tags to comments containing code and other embedded markup in order to avoid encoding issues.

Example:

```
/// <example>
/// Add the following key to the “appSettings” section of your config:
/// <code><![CDATA[
/// <configuration>
/// <appSettings>
/// <add key=”mySetting” value=”myValue” />
/// </appSettings>
/// </configuration>
/// ]]></code> >
/// </example>
```

STANDARDIZATION OF THE CODING

Coding style causes the most inconsistency and controversy between developers. Each developer has a preference, and

rarely are two the same. However, consistent layout, format, and organization are key to creating maintainable code.

The following sections describe the preferred way to implement C# source code in order to create readable, clear, and

consistent code that is easy to understand and maintain.

FORMATTING

- Never declared more than 1 namespace per file.
- Avoided putting multiple classes in a single file.
- Always placed curly braces ({ and }) on a new line.
- Always used curly braces ({ and }) in conditional statements.
- Always used a Tab & Indention size of 4.
- Declared each variable independently – not in the same statement.
- Placed namespace “using” statements together at the top of file. Group .NET namespaces above custom namespaces.
- Grouped internal class implementation by type in the following order:
 - a) Member variables.
 - b) Constructors & Finalizers.
 - c) Nested Enums, Structs, and Classes.
 - d) Properties
 - e) Methods
- Sequence declarations within type groups based upon access modifiers modifier and visibility:
 - a) Public
 - b) Protected
 - c) Internal
 - d) Private
- Segregate interface Implementation by using #region statements.
- Append folder-name to namespace for source files within sub-folders.
- Recursively indent all code blocks contained within namespaces.
- Use white space (CR/LF, Tabs, etc) liberally to separate and organize code.
- Only declare related attribute declarations on a single line, otherwise stack each attribute as a separate declaration.

Example:

```
// Bad!
```

```
[Attribute1, Attribute2, Attribute3]
```

```
public class MyClass
```

```
{...}
```

```
// Good!
```

[Attribute1, RelatedAttribute2]

[Attribute3]

[Attribute4]

public class MyClass

{...}

- Place Assembly scope attribute declarations on a separate line.
- Place Type scope attribute declarations on a separate line.
- Place Method scope attribute declarations on a separate line.
- Place Member scope attribute declarations on a separate line.
- Place Parameter attribute declarations inline with the parameter.
- If in doubt, always err on the side of clarity and consistency.

CODE EFFICIENCY

We started working on the project keeping in mind that we must develop it in a way that it not only provides a very easy to use GUI but also provide a fast and flexible service to the users. We know that a particular work can be done in more than one ways. We have tried all the options and then chose the one which provides the fastest and most secure performance. First of all, we have used the latest technologies of Microsoft like visual studio 2010 as IDE and WPF as GUI to keep our application's performance few steps ahead. We have studies all the rules of software development life cycle and applied them to keep our application flexible. We have given special attention to the storage related codes. We have avoided all the unneSMMSsary database codes and kept them as short as possible without harming our purpose so that insertion, updating, deletion and fetching of data take place flexibly. You can see the result as a user; our application does all the works very smoothly.

ERROR HANDLING

The C# language's exception handling features help us to deal with any unexpected or exceptional situations that occur when a program is running. Exception handling uses the **try**, **catch**, and **finally** keywords to try actions that may not succeed, to handle failures when you decide that it is reasonable to do so, and to clean up resourSMMS afterward. Exceptions can be generated by the common language runtime (CLR), by the .NET Framework or any third-party libraries, or by application code. Exceptions are created by using the **throw** keyword.

In many cases, an exception may be thrown not by a method that your code has called directly, but by another method further down in the call stack. When this happens, the CLR will unwind the stack, looking for a method

with a **catch** block for the specific exception type, and it will execute the first such **catch** block that it finds. If it finds no appropriate **catch** block anywhere in the call stack, it will terminate the process and display a message to the user.

Exceptions Overview

Exceptions have the following properties:

Exceptions are types that all ultimately derive from **System.Exception**.

Use a **try** block around the statements that might throw exceptions.

Once an exception occurs in the **try** block, the flow of control jumps to the first associated exception handler that is present anywhere in the call stack. In C#, the **catch** keyword is used to define an exception handler.

If no exception handler for a given exception is present, the program stops executing with an error message.

Do not catch an exception unless you can handle it and leave the application in a known state. If you catch **System.Exception**, rethrow it using the **throw** keyword at the end of the **catch** block.

If a **catch** block defines an exception variable, you can use it to obtain more information about the type of exception that occurred.

Exceptions can be explicitly generated by a program by using the **throw** keyword.

Exception objects contain detailed information about the error, such as the state of the call stack and a text description of the error.

Code in a **finally** block is executed even if an exception is thrown. Use a **finally** block to release resources, for example to close any streams or files that were opened in the **try** block.

Managed exceptions in the .NET Framework are implemented on top of the Win32 structured exception handling mechanism.

VALIDATION CHECKS

We have performed following data validation checks on available data:

ALLOWED CHARACTER CHECKS

Checks that ascertain that only expected characters are present in a field. For example a numeric field may only allow the digits 0-9, the decimal point and perhaps a minus sign or commas. A text field such as a personal name might disallow characters such as < and >, as they could be evidence of a markup-based security attack. An e-mail

address might require exactly one @ sign and various other structural details. Regular expressions are effective ways of implementing such checks. (See also data type checks below)

BATCH TOTALS

Checks for missing records. Numerical fields may be added together for all records in a batch. The batch total is entered and the computer checks that the total is correct, e.g., add the 'Total Cost' field of a number of transactions together.

CARDINALITY CHECK

Checks that record has a valid number of related records. For example if Contact record classified as a Customer it must have at least one associated Order (Cardinality > 0). If order does not exist for a "customer" record then it must be either changed to "seed" or the order must be created. This type of rule can be complicated by additional conditions. For example if contact record in Payroll database is marked as "former employee", then this record must not have any associated salary payments after the date on which employee left organization (Cardinality = 0).

CHECK DIGITS

Used for numerical data. An extra digit is added to a number which is calculated from the digits. The computer checks this calculation when data are entered. For example the last digit of an ISBN for a book is a check digit calculated modulus 10.[3]

CONSISTENCY CHECKS

Checks fields to ensure data in these fields corresponds, e.g., If Title = "Mr.", then Gender = "M".

CONTROL TOTALS

This is a total done on one or more numeric fields which appears in every record. This is a meaningful total, e.g., add the total payment for a number of Customers.

CROSS-SYSTEM CONSISTENCY CHECKS

Compares data in different systems to ensure it is consistent, e.g., The address for the customer with the same id is the same in both systems. The data may be represented differently in different systems and may need to be transformed to a common format to be compared, e.g., one system may store customer name in a single Name field as 'Doe, John Q', while another in three different fields: First_Name (John), Last_Name (Doe) and Middle_Name (Quality); to compare the two, the validation engine would have to transform data from the second system to match the data from the first, for example, using SQL: Last_Name || ', ' || First_Name || substr(Middle_Name, 1, 1) would convert the data from the second system to look like the data from the first 'Doe, John Q'

DATA TYPE CHECKS

Checks the data type of the input and give an error message if the input data does not match with the chosen data type, e.g., In an input box accepting numeric data, if the letter 'O' was typed instead of the number zero, an error message would appear.

FILE EXISTENCE CHECK

Checks that a file with a specified name exists. This check is essential for programs that use file handling.

FORMAT OR PICTURE CHECK

Checks that the data is in a specified format (template), e.g., dates have to be in the format DD/MM/YYYY.

Regular expressions should be considered for this type of validation.

HASH TOTALS

This is just a batch total done on one or more numeric fields which appears in every record. This is a meaningless total, e.g., add the Telephone Numbers together for a number of Customers.

LIMIT CHECK

Unlike range checks, data are checked for one limit only, upper OR lower, e.g., data should not be greater than 2 (≤ 2).

LOGIC CHECK

Checks that an input does not yield a logical error, e.g., an input value should not be 0 when there will be a number that divides it somewhere in a program.

PRESENCE CHECK

Checks that important data are actually present and have not been missed out, e.g., customers may be required to have their telephone numbers listed.

RANGE CHECK

Checks that the data lie within a specified range of values, e.g., the month of a person's date of birth should lie between 1 and 12.

REFERENTIAL INTEGRITY

In modern Relational database values in two tables can be linked through foreign key and primary key. If values in the primary key field are not constrained by database internal mechanism,[4] then they should be validated.

Validation of the foreign key field checks that referencing table must always refer to a valid row in the referenced table.[5]

SPELLING AND GRAMMAR CHECK

Looks for spelling and grammatical errors.

UNIQUENESS CHECK

Checks that each value is unique. This can be applied to several fields (i.e. Address, First Name, Last Name).

TABLE LOOK UP CHECK

A table look up check takes the entered data item and compares it to a valid list of entries that are stored in a database table.

TESTING

TESTING TECHNIQUES AND TESTING STRATEGIES USED

SMMS application will be tested using following strategies to ensure that the application succeeds to complete all the functional and non functional requirements:

UNIT TESTING:

Unit testing will take place within the construction phase of the project. After application module has been built to meet design specifications, each component (screen, view, interface, conversion program, etc.) will be tested individually to help confirm that it functions properly as an individual unit. Basic performance testing will also be completed during unit test to resolve obvious issues with performance prior to the System Testing.

The resource responsible for development will conduct testing of their module using the unit test conditions defined by the developer based on detailed design documents. The final step of unit test will be a review by the team lead to obtain their signoff on the component test checklist.

SMOKE TESTING:

Test Objective:	Verifies the major functionality at high level in order to determine if further testing is possible.
Technique:	<ul style="list-style-type: none">After initial deployment to the test environment validate all critical components of the application prior to proceeding with testing.
Completion Criteria:	<ul style="list-style-type: none">Navigation through the application at high level is possible, testing can continue.

FUNCTIONAL TESTING:

Function testing focuses on any requirements for test that can be traced directly to use cases or business functions and business rules. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box techniques; that are verifying the application and its internal processes by interacting with the application via the Graphical User Interface (GUI) and analyzing the output or results. Identified below is an outline of the function testing recommended for SMMS:

Test Objective:	Ensure proper target-of-test functionality, including business process validation.
Technique:	<p>Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following:</p> <ul style="list-style-type: none">• The expected results occur when valid data is used.• The appropriate error or warning messages are displayed when invalid data is used.• Business rules are properly applied.• Black Box end to end testing of configured processes. Manual validation of required and optional fields.
Completion Criteria:	<ul style="list-style-type: none">• All planned tests have been executed.• All defects that have been identified have been resolved• All resolutions have been implemented.

REGRESSION TESTING:

Regression testing focuses on software functionality that may have been previously working however through subsequent changes may have been inadvertently impacted. The goals of these tests are to verify that the broader impact of changes has been verified. Identified below is an outline of the regression testing recommended for each application(s)/module(s) of SMMS.

Test Objective:	Ensure that previously passed test cases continue to pass as the new system development is deployed and that surrounding systems that may be impacted by a change are still functioning as expected.
Technique:	<ul style="list-style-type: none">• Execute previous passed testing suites to ensure the following:• The expected results occur when valid data is used.• The appropriate error or warning messages are displayed when invalid data is used.• Each business rule is properly applied.
Completion Criteria:	<ul style="list-style-type: none">• All planned regression tests have been executed.• All identified defects have been resolved.

DATABASE & DATA INTEGRITY TESTING

The databases and the database processes should be tested as a subsystem within the SMMS Application. These subsystems should be tested with the target-of-test's User Interface as the interface to the database.

Test Objective:	Ensure that data is stored correctly, audits can be performed, access is controlled
Technique:	<ul style="list-style-type: none">• SQL queries will be executed in the DB to verify the data content and correctness.
Completion Criteria:	<ul style="list-style-type: none">• All planned tests have been executed.• All defects that have been identified have been resolved• All resolutions have been implemented.

USER INTERFACE TESTING:

User Interface (UI) testing verifies a user's interaction with the software. The goal of UI testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards. Most of this testing will have been done during functional testing. The areas of focus will be on design, layout and navigation of the screens.

Test Objective:	UI testing will verify the screens and the layouts and navigation
Technique:	<ul style="list-style-type: none">• Verify the design and layout of the screen.• Identify the integration links.• Test the functioning of the links – that the proper page is displayed and correct messages, pop-ups are shown when they need to be displayed etc• Validation of general navigation
Completion Criteria:	<ul style="list-style-type: none">• All navigation test cases have been executed.• All screens have been verified as per design and layouts• All defects that have been identified have been resolved.

PERFORMANCE PROFILING:

Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated. The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune performance behaviours as a function of conditions such as workload or hardware configurations

Test Objective:	The purpose of performance profiling is to ensure the performance of the SMMS application is up to the desired level.
Technique:	<ul style="list-style-type: none">• Use a subset of Test Procedures developed for Function and Business Cycle Testing.• Modify data files to increase the number of transactions or the scripts to increase the number of iterations each transaction occurs.• This will be done by using Load Runner or Quick Test Professional (QTP).
Completion Criteria:	<ul style="list-style-type: none">• Single Transaction or single user: Successful completion of the test scripts without any failures and within the expected or required time allocation per transaction.• Results are recorded and a performance baseline is created for the major logical functions within the scenarios listed above.• All performance defects are reviewed and triaged to an acceptable resolution.

LOAD TESTING:

Load testing is a performance test which subjects the target-of-test to varying workloads to measure and evaluate the performance behaviours and ability of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly at the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time sensitive issues.

Test Objective:	The purpose of load testing is to verify performance behaviour time for designated transactions or business cases under varying workload conditions.
Technique:	<ul style="list-style-type: none">• Use a subset of Test Procedures developed for Function and Business Cycle Testing.• Scripts will be executed to simulate the peak load for 1 hour and report will be generated and analysed.• This will be done using Load Runner.

Completion Criteria:	<ul style="list-style-type: none"> • Multiple transactions or multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation. • Results are recorded and a performance baseline is created for the major business functions within the scenarios listed above. • All load testing defects are reviewed and triaged to an acceptable resolution.
----------------------	--

STRESS TESTING:

Stress testing is a type of performance test implemented and executed to find errors due to low resources or competition for resources. Low memory or disk space may reveal defects in the target-of-test that aren't apparent under normal conditions. Other defects might result from competition for shared resources like database locks or network bandwidth. Stress testing can also be used to identify the peak workload the target-of-test can handle, which is often beyond the production workload.

VOLUME TESTING:

Volume Testing subjects the target-of-test to large amounts of data to determine if limits are reached that cause the software to fail. Volume Testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period. For example, if the target-of-test is processing a set of database records to generate a report, a Volume Test would use a large test database and check that the software behaved normally and produced the correct report.

SECURITY & ACCESS CONTROL TESTING:

Security and Access Control Testing focus on following key areas of security:

- Application-level security, including access to the Data or Business Functions

Application-level security ensures the authentication and authorization of a user. Authentication ensures that the user is a valid user of the system and authorization ensures that the user has the proper privileges to perform specific tasks on desired resources of the system. Testing will be conducted to validate the rules by taking into considerations the various roles applicable for the system.

FAILOVER & RECOVERY TESTING:

Failover and Recovery Testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with undue loss of data or data integrity.

Failover testing ensures that, for those systems that must be kept running, when a failover condition occurs, the alternate or backup systems properly “take over” for the failed system without loss of data or transactions.

Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions, or simulated conditions, to cause a failure, such as device Input/ Output (I/O) failures or invalid database pointers and keys. Recovery processes are invoked and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved.

CONFIGURATION TESTING:

Configuration testing verifies the operation of the target-of-test on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections and database servers vary. Client workstations may have different software loaded—for example, applications, drivers, and so on—and at any one time, many different combinations may be active using different resources.

INSTALLATION/DEPLOY & BACK OUT TESTING:

Installation testing has two purposes. The first is to ensure that the software can be installed under different conditions—such as a new installation, an upgrade and a complete or custom installation—under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, and so on. The second purpose is to verify that, once installed; the software operates correctly and can be backed out successfully. This usually means running a number of the tests that were developed for Function testing before and after the back out.

POST PRODUCTION TESTING:

The purpose of Post production testing is to verify that, once deployed, the software operates correctly. This usually means running a number of the tests that were developed for Function Testing ensuring that no data is changed/ modified in production. Typically, the business SME's assist with Post production testing.

DATA MIGRATION TESTING:

This is the process of testing to verify whether or not the data migration (or conversion) has been successfully completed. The testing process will be carried out by running SQL scripts on both the source and destination databases.

The fields which are present in the new data Model in the Destination DB(s) will be migrated from the existing systems source DB(s) to Destination DB(s).

Test Objective:	The objective of this test is to verify that data migration is successful from source DB(s) to destination DB(s).
Technique:	<ul style="list-style-type: none">• The Team is notified before the data migration.• Team runs queries on the source DB and fetches the data.• Data Migration is done.• Mapped data needs to be determined.• Team runs the queries on the Destination DB and fetches the data.• Cross verification of the data is done to see that data fetched from the old database is same as the data fetched from the new database.• Verification of the table structure.• Verification of record counts.• Verification of the data formatting.
Completion Criteria:	<ul style="list-style-type: none">• Data fetched from the Source DB(s) and Destination DB(s) matches.• The record count in the Source and the Destination databases should be equal.• No data are truncated.• Data formatting is proper (if required at any instance).• All defects that have been identified have been resolved.

TESTING PLAN USED

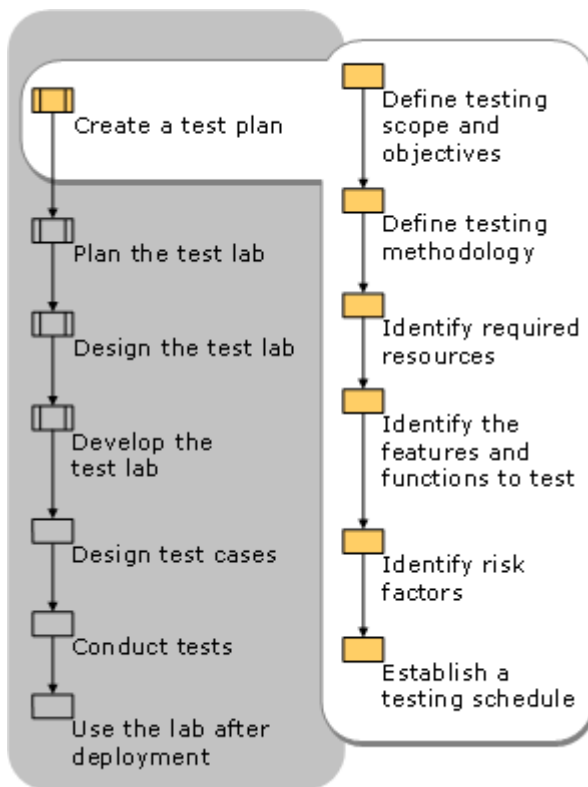
CREATION OF TEST PLAN

Early in the deployment planning phase, the testing team creates a test plan. The *test plan* defines the objectives and scope of the testing effort, and identifies the methodology that our team will use to conduct tests. It also identifies the hardware, software, and tools required for testing and the features and functions that will be tested. A well-rounded test plan notes any risk factors that jeopardize testing and includes a testing schedule.

If our testing team is divided into technology subteams, each subteam should develop a test plan for that team's specific technology area. For example, the networking team would write a test plan for testing networking features. All members of each subteam should review and approve its team's test plan before it is integrated into the general test plan.

Figure 2.3 illustrates the tasks we performed to create the test plan.

Figure 2.3 Creating a Test Plan



TESTING SCOPE AND OBJECTIVES

In the scope and objectives section of the test plan, the testing team described specifically what we want our testing to accomplish. Also, we needed to define the scope of our testing by identifying what we will test and what we will not. We might limit our testing of client computer hardware to the minimum supported configurations or to the standard configurations.

TESTING METHODOLOGY

The general methodology that our team used for testing to testing schema changes might be to configure an isolated domain in the test lab where schema changes can be applied without affecting other lab tests. This section of the test plan addressed the following:

- The domain architecture used for testing
- The tools and techniques used to conduct the tests or to measure results

- Automated techniques we plan to use during testing

FEATURES AND FUNCTIONS TO TEST

Included tests that verify or address:

- The functionality of each feature and service that you will implement.
- Interoperability with existing components and systems in the production environment, both during the phase-in period, when there is a mix of old functionality and new Windows Server 2003 functionality, and after the Windows Server 2003 environment has been rolled out.
- Hardware and driver compatibility for every type of client computer that will be running Windows XP Professional.
- Application compatibility for every application that will run on Windows XP Professional.
- Application compatibility for every server application that will run on Windows Server 2003.
- Optimization of configurations, such as those for standardized desktops on client computers.

RISK FACTORS

We described the risk factors that could prevent the successful completion of all required tests. We found that the test lab is behind schedule, or that required hardware or software is unavailable, or that testers are working on other projects or need additional training. After we have identified the risk factors, decide what we will do to avoid or mitigate each risk.

TESTING SCHEDULE

We drafted a preliminary schedule that includes each test listed in the test plan. The schedule can help us coordinate test lab use among sub teams. Assign a team member, ideally the test lab manager, if our team has one, to maintain and update the lab schedule. Having an up-to-date schedule is critical when unscheduled lab requests are submitted.

TESTING REPORTS

TEST REPORTS FOR UNIT TEST CASES AND SYSTEM TEST CASES

TEST REPORTS FOR UNIT TEST CASES

Test Case Id	Comment	Status
SMMS-001	NA	PASS
SMMS-002	NA	PASS
SMMS-003	NA	PASS
SMMS-004	NA	PASS
SMMS-005	NA	PASS
SMMS-006	NA	PASS
SMMS-007	NA	PASS
SMMS-008	NA	PASS
SMMS-009	NA	PASS
SMMS-010	NA	PASS
SMMS-011	NA	PASS
SMMS-012	NA	PASS
SMMS-013	NA	PASS
SMMS-014	NA	PASS
SMMS-015	NA	PASS
SMMS-016	NA	PASS
SMMS-017	NA	PASS
SMMS-018	NA	PASS
SMMS-019	NA	PASS
SMMS-020	NA	PASS
SMMS-021	NA	PASS

SMMS-022	NA	PASS
SMMS-023	NA	PASS
SMMS-024	NA	PASS
SMMS-025	NA	PASS
SMMS-026	NA	PASS
SMMS-027	NA	PASS
SMMS-028	NA	PASS
SMMS-029	NA	PASS
SMMS-030	NA	PASS
SMMS-031	NA	PASS
SMMS-032	NA	PASS
SMMS-033	NA	PASS
SMMS-034	NA	PASS
SMMS-035	NA	PASS
SMMS-036	NA	PASS
SMMS-037	NA	PASS
SMMS-038	NA	PASS
SMMS-039	NA	PASS
SMMS-040	NA	PASS
SMMS-041	NA	PASS
SMMS-042	NA	PASS
SMMS-043	NA	PASS
SMMS-044	NA	PASS
SMMS-045	NA	PASS
SMMS-046	NA	PASS
SMMS-047	NA	PASS

SMMS-048	NA	PASS
SMMS-049	NA	PASS
SMMS-050	NA	PASS
SMMS-051	NA	PASS
SMMS-052	NA	PASS
SMMS-053	NA	PASS
SMMS-054	NA	PASS
SMMS-055	NA	PASS
SMMS-056	NA	PASS
SMMS-057	NA	PASS
SMMS-058	NA	PASS
SMMS-059	NA	PASS
SMMS-060	NA	PASS
SMMS-061	NA	PASS
SMMS-062	NA	PASS
SMMS-063	NA	PASS
SMMS-064	NA	PASS
SMMS-065	NA	PASS
SMMS-066	NA	PASS

TEST REPORTS FOR SYSTEM TEST CASES

Test Case Id	Comment	Status
SMMS-067	NA	PASS

SMMS-068	NA	PASS
SMMS-069	NA	PASS
SMMS-070	NA	PASS
SMMS-071	NA	PASS
SMMS-072	NA	PASS
SMMS-073	NA	PASS
SMMS-074	NA	PASS
SMMS-075	NA	PASS
SMMS-076	NA	PASS
SMMS-077	NA	PASS
SMMS-078	NA	PASS
SMMS-079	NA	PASS
SMMS-080	NA	PASS
SMMS-081	NA	PASS
SMMS-082	NA	PASS
SMMS-083	NA	PASS
SMMS-084	NA	PASS
SMMS-085	NA	PASS
SMMS-086	NA	PASS
SMMS-087	NA	PASS
SMMS-089	NA	PASS
SMMS-090	NA	PASS
SMMS-091	NA	PASS
SMMS-092	NA	PASS
SMMS-093	NA	PASS
SMMS-094	NA	PASS

SMMS-095	NA	PASS
----------	----	------

DEBUGGING AND CODE IMPROVEMENT

The steps in the bellow section demonstrate how to create a console application that uses the **Debug** class to provide information about the program execution.

When the program is run, we can use methods of the **Debug** class to produce messages that help we to monitor the program execution sequence, to detect malfunctions, or to provide performance measurement information. By default, the messages that the **Debug** class produSMMS appear in the Output window of the Visual Studio Integrated Development Environment (IDE).

The sample code uses the **WriteLine** method to produce a message that is followed by a line terminator. When we use this method to produce a message, each message appears on a separate line in the Output window.

When we use the **Assert** method of the **Debug** class, the Output window displays a message only if a specified condition evaluates to false. The message also appears in a modal dialog box to the user. The dialog box includes the message, the project name, and the **Debug.Assert** statement number. The dialog box also includes the following three command buttons:

- **Abort:** The application stops running.
- **Retry:** The application enters debug mode.
- **Ignore:** The application proceeds.

The user must click one of these buttons before the application can continue.

We can also direct output from the **Debug** class to destinations other than the Output window. The **Debug** class has a collection named **Listeners** that includes **Listener** objects.

Each **Listener** object monitors **Debug** output and directs the output to a specified target.

Each **Listener** in the **Listener** collection receives any output that the **Debug** class generates. Use the **TextWriterTraceListener** class to define **Listener** objects. We can specify the target for a **TextWriterTraceListener** class through its constructor.

Some possible output targets include the following:

- The Console window by using the **System.Console.Out** property.

- A text (.txt) file by using the **System.IO.File.CreateText("FileName.txt")** statement.

After we create a **TextWriterTraceListener** object, we must add the object to the **Debug.Listeners** collection to receive Debug output.

CREATE A SAMPLE WITH THE DEBUG CLASS

1. Start Visual Studio or Visual C# Express Edition.
2. Create a new Visual C# Console Application project named **conInfo**. Class1 is created in Visual Studio .NET. Program.cs is created in Visual Studio 2005.
3. Add the following namespace at top in Class1 or Program.cs.

```
using System.Diagnostics;
```
4. To initialize variables to contain information about a product, add the following declaration statements to **Main** method:
5.

```
string sProdName = "Widget";
```
6.

```
int iUnitQty = 100;  
double dUnitCost = 1.03;
```
7. Specify the message that the class produSMMS as the first input parameter of the **WriteLine** method. Press the CTRL+ALT+O key combination to make sure that the Output window is visible.

```
Debug.WriteLine("Debug Information-Product Starting ");
```
8. For readability, use the **Indent** method to indent subsequent messages in the Output window:

```
Debug.Indent();
```
9. To display the content of selected variables, use the **WriteLine** method as follows:
10.

```
Debug.WriteLine("The product name is " + sProdName);
```
11.

```
Debug.WriteLine("The available units on hand are" + iUnitQty.ToString());  
Debug.WriteLine("The per unit cost is " + dUnitCost.ToString());
```
12. We can also use the **WriteLine** method to display the namespace and the class name for an existent object. For example, the following code displays the **System.Xml.XmlDocument** namespace in the Output window:
13.

```
System.Xml.XmlDocument oxml = new System.Xml.XmlDocument();  
Debug.WriteLine(oxml);
```
14. To organize the output, we can include a category as an optional, second input parameter of the **WriteLine** method. If we specify a category, the format of the Output window message is "category: message." For example, the first line of the following code displays "Field: The product name is Widget" in the Output window:
15.

```
Debug.WriteLine("The product name is " + sProdName,"Field");
```
16.

```
Debug.WriteLine("The units on hand are" + iUnitQty,"Field");
```
17.

```
Debug.WriteLine("The per unit cost is" + dUnitCost.ToString(),"Field");  
Debug.WriteLine("Total Cost is " + (iUnitQty * dUnitCost),"Calc");
```
18. The Output window can display messages only if a designated condition evaluates to true by using the **WriteLineIf** method of the **Debug** class. The condition to be evaluated is the first input parameter of the **WriteLineIf** method. The second parameter of **WriteLineIf** is the message that appears only if the condition in the first parameter evaluates to true.
19.

```
Debug.WriteLineIf(iUnitQty > 50, "This message WILL appear");
```
20.

```
Debug.WriteLineIf(iUnitQty < 50, "This message will NOT appear");
```

21. Use the **Assert** method of the **Debug** class so that the Output window displays the message only if a specified condition evaluates to false:
22. `Debug.Assert(dUnitCost > 1, "Message will NOT appear");`
23. `Debug.Assert(dUnitCost < 1, "Message will appear since dUnitcost < 1 is false");`
24. Create the **TextWriterTraceListener** objects for the Console window (tr1) and for a text file named Output.txt (tr2), and then add each object to the **Debug Listeners** collection:
25. `TextWriterTraceListener tr1 = new TextWriterTraceListener(System.Console.Out);`
26. `Debug.Listeners.Add(tr1);`
- 27.
28. `TextWriterTraceListener tr2 = new
 TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));
 Debug.Listeners.Add(tr2);`
29. For readability, use the **Unindent** method to remove the indentation for subsequent messages that the **Debug** class generates. When we use the **Indent** and the **Unindent** methods together, the reader can distinguish the output as group.
30. `Debug.Unindent();
 Debug.WriteLine("Debug Information-Product Ending");`
31. To make sure that each **Listener** object receives all its output, call the **Flush** method for the **Debug** class buffers:
`Debug.Flush();`

USING THE TRACE CLASS

We can also use the **Trace** class to produce messages that monitor the execution of an application.

The **Trace** and **Debug** classes share most of the same methods to produce output, including the following:

- **WriteLine**
- **WriteLineIf**
- **Indent**
- **Unindent**
- **Assert**
- **Flush**

We can use the **Trace** and the **Debug** classes separately or together in the same application. In a Debug Solution Configuration project, both **Trace** and **Debug** output are active. The project generates output from both of these classes to all **Listener** objects. However, a Release Solution Configuration project only generates output from a **Trace** class. The Release Solution Configuration project ignores any **Debug** class method invocations.

```
Trace.WriteLine("Trace Information-Product Starting ");  
Trace.Indent();  
  
Trace.WriteLine("The product name is "+sProdName);  
Trace.WriteLine("The product name is"+sProdName,"Field" );  
Trace.WriteLineIf(iUnitQty > 50, "This message WILL appear");  
Trace.Assert(dUnitCost > 1, "Message will NOT appear");  
  
Trace.Unindent();  
Trace.WriteLine("Trace Information-Product Ending");
```

```
Trace.Flush();  
  
Console.ReadLine();
```

VERIFY THAT IT WORKS

1. Make sure that **Debug** is the current solution configuration.
2. If the **Solution Explorer** window is not visible, press the CTRL+ALT+L key combination to display this window.
3. Right-click **conInfo**, and then click **Properties**.
4. In the left pane of the conInfo property page, under the **Configuration** folder, make sure that the arrow points to **Debugging**.

Note In Visual C# 2005 and in Visual C# 2005 Express Edition, click **Debug** in the **conInfo** page.

5. Above the **Configuration** folder, in the **Configuration** drop-down list box, click **Active (Debug)** or **Debug**, and then click **OK**. In Visual C# 2005 and in Visual C# 2005 Express Edition, click **Active (Debug)** or **Debug** in the **Configuration** drop-down list box in the **Debug** page, and then click **Save** on the **File** menu.
6. Press CTRL+ALT+O to display the Output window.
7. Press the F5 key to run the code. When the **Assertion Failed** dialog box appears, click **Ignore**.
8. In the Console window, press ENTER. The program should finish, and the Output window should display the output that resembles the following

```
9.      Debug Information-Product Starting  
10.     The product name is Widget  
11.     The available units on hand are100  
12.     The per unit cost is 1.03  
13.     System.Xml.XmlDocument  
14.     Field: The product name is Widget  
15.     Field: The units on hand are100  
16.     Field: The per unit cost is1.03  
17.     Calc: Total Cost is  103  
18.     This message WILL appear  
19.     ---- DEBUG ASSERTION FAILED ----  
20. ---- Assert Short Message ----  
21. Message will appear since dUnitcost  < 1 is false  
22. ---- Assert Long Message ----  
23.  
24.  
25.     at Class1.Main(String[] args)  <%Path%>\class1.cs(34)  
26.  
27.     The product name is Widget  
28.     The available units on hand are100  
29.     The per unit cost is 1.03  
30. Debug Information-Product Ending  
31. Trace Information-Product Starting  
32.     The product name is Widget  
33.     Field: The product name isWidget  
34.     This message WILL appear  
35. Trace Information-Product Ending  
36.
```

37. The Console window and the Output.txt file should display the following output:

```
38. The product name is Widget
39.     The available units on hand are 100
40.     The per unit cost is 1.03
41. Debug Information-Product Ending
42. Trace Information-Product Starting
43.     The product name is Widget
44.     Field: The product name is Widget
45.     This message WILL appear
46. Trace Information-Product Ending
```

Note The Output.txt file is located in the same directory as the conInfo executable (conInfo.exe). Typically, this is the \bin folder where the project source is stored. By default, this is C:\Documents and Settings\User login\My Documents\Visual Studio Projects\conInfo\bin. In Visual C# 2005 and in Visual C# 2005 Express Edition, the Output.txt file is located in the following folder:

C:\Documents and Settings\User login\My Documents\Visual Studio
2005\Projects\conInfo\conInfo\bin\Debug

COMPLETE CODE LISTING

```
using System;
using System.Diagnostics;

class Class1
{
    [STAThread]
    static void Main(string[] args)
    {
        string sProdName = "Widget";
        int iUnitQty = 100;
        double dUnitCost = 1.03;
        Debug.WriteLine("Debug Information-Product Starting ");
        Debug.Indent();
        Debug.WriteLine("The product name is "+sProdName);
        Debug.WriteLine("The available units on hand are"+iUnitQty.ToString());
        Debug.WriteLine("The per unit cost is "+ dUnitCost.ToString());

        System.Xml.XmlDocument oxml = new System.Xml.XmlDocument();
        Debug.WriteLine(oxml);

        Debug.WriteLine("The product name is "+sProdName,"Field");
        Debug.WriteLine("The units on hand are"+iUnitQty,"Field");
        Debug.WriteLine("The per unit cost is"+dUnitCost.ToString(),"Field");
        Debug.WriteLine("Total Cost is  "+(iUnitQty * dUnitCost),"Calc");

        Debug.WriteLineIf(iUnitQty > 50, "This message WILL appear");
    }
}
```

```

        Debug.WriteLineIf(iUnitQty < 50, "This message will NOT appear");

        Debug.Assert(dUnitCost > 1, "Message will NOT appear");
        Debug.Assert(dUnitCost < 1, "Message will appear since dUnitcost  < 1 is
false");

        TextWriterTraceListener tr1 = new
        TextWriterTraceListener(System.Console.Out);
        Debug.Listeners.Add(tr1);

        TextWriterTraceListener tr2 = new
        TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));
        Debug.Listeners.Add(tr2);

        Debug.WriteLine("The product name is "+sProdName);
        Debug.WriteLine("The available units on hand are"+iUnitQty);
        Debug.WriteLine("The per unit cost is "+dUnitCost);
        Debug.Unindent();
        Debug.WriteLine("Debug Information-Product Ending");
        Debug.Flush();

        Trace.WriteLine("Trace Information-Product Starting ");
        Trace.Indent();

        Trace.WriteLine("The product name is "+sProdName);
        Trace.WriteLine("The product name is"+sProdName,"Field" );
        Trace.WriteLineIf(iUnitQty > 50, "This message WILL appear");
        Trace.Assert(dUnitCost > 1, "Message will NOT appear");

        Trace.Unindent();
        Trace.WriteLine("Trace Information-Product Ending");

        Trace.Flush();

        Console.ReadLine();
    }
}

```

TROUBLESHOOT

- If the solution configuration type is **Release**, the **Debug** class output is ignored.
- After we create a **TextWriterTraceListener** class for a particular target, **TextWriterTraceListener** receives output from the **Trace** and the **Debug** classes. This occurs regardless of whether we use the **Add** method of the **Trace** or the **Debug** class to add **TextWriterTraceListener** to the **Listeners** class.

- If we add a **Listeners** object for the same target in the **Trace** and the **Debug** classes, each line of output is duplicated, regardless of whether **Debug** or **Trace** generates the output.

```

•      TextWriterTraceListener myWriter = new
      TextWriterTraceListener(System.Console.Out);
•      Debug.Listeners.Add(myWriter);
•
•      TextWriterTraceListener myCreator = new
      TextWriterTraceListener(System.Console.Out);
      Trace.Listeners.Add(myCreator);

```

SYSTEM SECURITY MEASURES

- Shopping Mall Management System is password protected software. It will be developed such a way that the admin will have complete control on the application's data.
- Admin can create account with various permission levels, like admin, data enterer etc. so that the users can see relevant data only, for example, a data enterer can only see the feedback provided by user, on the other hand, and admin can see and delete them as well.
- A customer can only see the information and provide feedback.

DATABASE / DATA SECURITY

- The data of the SMMS will be stored in the database with an encrypted format so even if someone hacks the database somehow still he can make no real harm.
- The software will provide a backup and restore feature in case of loss of data.

CREATION OF USER PROFILES AND ACCESS RIGHTS

- ❖ The software asks for a predefined user-type, username and password to use its feature.
- ❖ All the data are not available for all types of user, for example, only an admin can use all the fields of the application. On the other hand, a clerk can only enter data and see data from some selected fields. A librarian can use library related data only.

COST ESTIMATION

COST ESTIMATION MODEL

We used the basic COCOMO model, which gives an approximate estimate of our **SMMS** project parameters. The basic COCOMO estimation model is given by the following expressions:

$$\text{Effort} = a_1 * (\text{KLOC})^{a_2} \text{ PM}$$

$$\text{Tdev} = b_1 * (\text{Effort})^{b_2} \text{ months}$$

Where

KLOC is the estimated size of the software product expressed in Kilo Lines of Code a_1 , a_2 , b_1 , b_2 are constants for each category of software products.

Tdev is the estimated time to develop the software, expressed in months.

Effort is the total effort required to develop the software product, expressed in person-month (PM).

Our project is semidetached type, because the development team consists of a mixture of experienced and inexperienced staff like my guide and me. Team members may have limited experience on related system but may be unfamiliar with aspects of the system being developed.

ESTIMATION OF DEVELOPMENT EFFORT

For our Semi-detached class software product **SMMS**, the formula for estimating the effort based on the code size is shown below:

$$\text{Semi-detached SMMS: Tdev} = 3.0 * (\text{KLOC})^{1.12} \text{ PM}$$

ESTIMATION OF DEVELOPMENT TIME

For our Semi-detached class software product **SMMS**, the formula for estimating the development time based on the effort is given below:

$$\text{Semi-detached SMMS: Tdev} = 2.5 * (\text{Effort})^{0.35} \text{ months}$$

Assume that the size of a Semi-detached SMMS product has been estimated to be 4,000 lines of source code.
Assume that the average salary of software engineer(me) is Rs. 15,000 per month.

Assume that the size of our

The basic COCOMO estimation formula for SMMS semidetached software:

$$\text{Our Effort} = 3.0 * (4)^{1.12} \text{ PM}$$

$$= 14 \text{ PM}$$

Normal Development time = $2.5 * (14)^{0.35}$ months

= 6 months

Cost required to develop the product = Rs. 6 * 15,000

= Rs. 90,000

	Task Name	Work	Duration	Start	Finish
1	☐ SMS Desktop Client	40 hrs	5 days	Mon 10/1/12	Fri 10/5/12
	Susmita	40 hrs		Mon 10/1/12	Fri 10/5/12
2	☐ School Management Controller	80 hrs	10 days	Mon 10/8/12	Fri 10/19/12
	Susmita	80 hrs		Mon 10/8/12	Fri 10/19/12
3	☐ SMS Web Client	40 hrs	5 days	Mon 10/22/12	Fri 10/26/12
	Susmita	40 hrs		Mon 10/22/12	Fri 10/26/12
4	☐ Faculty Management system	40 hrs	5 days	Mon 10/29/12	Fri 11/2/12
	Susmita	40 hrs		Mon 10/29/12	Fri 11/2/12
5	☐ Admission Management System	60 hrs	7.5 days	Mon 11/5/12	Wed 11/14/12
	Susmita	60 hrs		Mon 11/5/12	Wed 11/14/12
6	☐ Course Management System	48 hrs	6 days	Thu 11/15/12	Thu 11/22/12
	Susmita	48 hrs		Thu 11/15/12	Thu 11/22/12
7	☐ Library Management System	40 hrs	5 days	Fri 11/23/12	Thu 11/29/12
	Susmita	40 hrs		Fri 11/23/12	Thu 11/29/12
8	☐ Accounts Management System	40 hrs	5 days	Fri 11/30/12	Thu 12/6/12
	Susmita	40 hrs		Fri 11/30/12	Thu 12/6/12
9	☐ Attendance Management System	32 hrs	4 days	Thu 12/6/12	Tue 12/11/12
	Susmita	32 hrs		Thu 12/6/12	Tue 12/11/12
10	☐ Student Management system	48 hrs	6 days	Tue 12/11/12	Tue 12/18/12
	Susmita	48 hrs		Tue 12/11/12	Tue 12/18/12
11	☐ SMS Database	72 hrs	9 days	Tue 12/18/12	Fri 12/28/12
	Susmita	72 hrs		Tue 12/18/12	Fri 12/28/12
12	☐ Integration	80 hrs	10 days	Mon 12/31/12	Fri 1/11/13
	Susmita	80 hrs		Mon 12/31/12	Fri 1/11/13
13	☐ Integration Testing	120 hrs	15 days	Mon 1/14/13	Fri 2/1/13
	Susmita	120 hrs		Mon 1/14/13	Fri 2/1/13
14	☐ Bug Fix	160 hrs	20 days	Fri 2/1/13	Thu 2/28/13
	Susmita	160 hrs		Fri 2/1/13	Thu 2/28/13
15	☐ Documentation	40 hrs	5 days	Thu 2/28/13	Wed 3/6/13
	Susmita	40 hrs		Thu 2/28/13	Wed 3/6/13

- Yearly donation report can be generated
- Salary slips can be created

FUTURE SCOPE AND FURTHER ENHANCEMENT

- To make the app more user friendly, the application could be made run from various parts of the shopping mal, so that multiple user can use it at a time.
- Mobile application of this app could be developed for android, windows phone, iOS etc. so that a user could know about a shopping mall from home and visit it according to their need.
- To make it even larger, we could develop a web app, so that the database of information would be available on the internet as well.

BIBLIOGRAPHY

WEBSITES

- <http://en.wikipedia.org>
- http://en.wikipedia.org/wiki/Windows_Presentation_Foundation
- <http://msdn.microsoft.com/en-us/>
- <http://www.c-sharpcorner.com/beginners/>
- <http://www.microsoft.com/en-us/default.aspx>
- <http://www.codeplex.com/>
- <http://searchitchannel.techtarget.com/>
- <http://stackoverflow.com/>
- <http://www.codeguru.com/>
- <http://www.roseindia.net/>
- <http://www.csharpcourse.com/>
- <http://www.w3shoppings.com>
- <http://blogs.technicise.com/>
- <http://connect.technicise.com/>
- <http://www.mysql.com/>
- <http://www.mysql.com/support/>
- <http://dev.mysql.com/>
- <http://dev.mysql.com/support/>
- <http://www.homeandlearn.co.uk/csharp/csharp.html>

- <http://www.wpftutorial.net/Home.html>
- <http://netbeans.org/>
- <https://www.facebook.com/>
- <https://developers.facebook.com/>
- <https://developers.facebook.com/apps>
- <https://www.facebook.com/creatormyapp>
- <https://www.facebook.com/help/>
- <https://twitter.com/#>
- <https://dev.twitter.com/>
- <https://dev.twitter.com/discussions/3477>
- <https://support.twitter.com/>
- <https://developers.google.com/>
- <https://maps.google.co.in/>
- <http://support.google.com/maps/?hl=en&rd=1>
- <http://www.youtube.com/?gl=IN>
- <http://blendinsider.com/>
- <https://github.com/>
- <https://github.com/anirban-nandy>
- <https://github.com/anirban-nandy/DailyNoteBook>
- <http://learn.github.com/p/intro.html>
- <http://www.vogella.com/articles/Git/article.html>
- <http://try.github.com/levels/1/challenges/1>
- <https://enterprise.github.com/support>
- <https://support.enterprise.github.com/home>

BOOKS

- **Programming Java** - E. R. Balaguruswamy.
- Fundamentals of software engineering by Rajib Mall.
- Pro C# 2010 and the .NET 4.0 Platform by Andrew Troselen.
- C# Programming by Rob Miles.

APPENDICES

IDE:

VISUAL STUDIO 2010



Microsoft Visual Studio is a powerful IDE that ensures quality code throughout the entire application lifecycle, from design to deployment. Whether we are developing applications for SharePoint, the web, Windows, Windows Phone, and beyond, Visual Studio is the ultimate all-in-one solution. Visual Studio includes a code editor supporting IntelliSense as well as code

refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

STANDOUT FEATURES

- User interface built on Windows Presentation Foundation (WPF)
- Improved Start page
- Improved code editor
- Improved IntelliSense
- Call Hierarchy Viewer

WHAT PROBLEMS DOES IT SOLVE?

The newly designed user experience is refreshing for an application showing its age. The user interface is built on WPF and no longer relies on the limited MDI interface in previous versions; this allows for better multi-monitor support with fly-out windows. The first thing you might notice when opening Visual Studio 2010 is the new Start page. As an xaml file, this page is completely customizable and includes the ability to remove and pin project files in the Recent Projects section.

The code editor has a number of enhancements. You can scale the font by holding down [Ctrl] while scrolling the mouse wheel. In previous versions of Visual Studio, users had to set the font size through a dialog and exit to see if the changes were correct.

In Visual Studio 2010, Box Selection is enhanced to allow for zero-length boxes and improved pasting.

The feature that will see the most use (by accident if not design) is Highlight References. By selecting any symbol, such as a variable or a property, all references to the symbol are highlighted. The symbols can then be navigated by holding down [Ctrl][Shift] and pressing the up/down keys.

IntelliSense has been improved to allow for acronyms based on Pascal casing. For example, typing *String.INOE* and then a non-alphanumeric character will convert the call to *String.IsNullOrEmpty*. This still doesn't prevent IntelliSense from interfering when you're writing code that doesn't exist, as you would with a unit test.

The Suggestion Completion mode allows you to type freely without IntelliSense changing the text you typed. You can toggle between Standard and Suggestion Completion modes by pressing [Ctrl][Alt]space.

IntelliSense for JavaScript has seen the most improvement, as it is now able to determine the correct structure of a variable even after the structure is changed.

In the past, I would use .NET Reflector or another tool to analyze a user's call hierarchy; now that functionality is built-in. Right-click the user and choose View Call Hierarchy, and calls to and from the user will be available for browsing.

FRONT END

WPF (WINDOWS PRESENTATION FRAMEWORK)



Windows Presentation Foundation (WPF) is a next-generation presentation system for building Windows client applications with visually stunning user experiences. With WPF, you can create a wide range of both standalone and browser-hosted applications.

Windows Presentation Foundation (WPF) provides developers with a unified programming model for building rich Windows smart client user experiences that incorporate UI, media, and documents. Windows Presentation Foundation (WPF) is a next-generation presentation system for building Windows client applications with visually stunning user experiences. With WPF, you can create a wide range of both standalone and browser-hosted applications. The core of WPF is a resolution-independent and vector-based rendering engine that is built to take advantage of modern graphics hardware. WPF extends the core with a comprehensive set of application-development features that include Extensible Application Markup Language (XAML), controls, data binding, layout, 2-D and 3-D graphics, animation, styles, templates, documents, media, text, and typography. WPF is included in the Microsoft .NET Framework, so you can build applications that incorporate other elements of the .NET Framework class library.

The core of WPF is a resolution-independent and vector-based rendering engine that is built to take advantage of modern graphics hardware. WPF extends the core with a comprehensive set of application-development features that include Extensible Application Markup Language (XAML), controls, data binding, layout, 2-D and 3-D graphics, animation, styles, templates, documents, media, text, and typography. WPF is included in the Microsoft .NET Framework, so you can build applications that incorporate other elements of the .NET Framework class library.

PROGRAMMING WITH WPF

WPF exists as a subset of .NET Framework types that are for the most part located in the `System.Windows` namespace. If you have previously built applications with .NET Framework using managed technologies like ASP.NET and Windows Forms, the fundamental WPF programming experience should be familiar; you instantiate classes, set properties, call methods, and handle events, all using your favorite .NET Framework programming language, such as C# or Visual Basic.

MARKUP & CODE-BEHIND

WPF offers additional programming enhancements for Windows client application development. One obvious enhancement is the ability to develop an application using both *markup* and *code-behind*, an experience that ASP.NET developers should be familiar with. You generally use Extensible Application Markup Language (XAML) markup to implement the appearance of an application while using managed programming languages (code-behind) to implement its behavior.

SECURITY

Because XBAPs are hosted in a browser, security is important. In particular, a partial-trust security sandbox is used by XBAPs to enforce restrictions that are less than or equal to the restrictions imposed on HTML-based applications. Furthermore, each HTML feature that is safe to run from XBAPs in partial trust has been tested using a comprehensive security process.

CONTROLS

The user experiences that are delivered by the application model are constructed controls. In WPF, "control" is an umbrella term that applies to a category of WPF classes that are hosted in either a window or a page, have a user interface (UI), and implement some behavior.

WPF CONTROLS BY FUNCTION

The built-in WPF controls are listed here.

- **Buttons:** Button and RepeatButton.
- **Data Display:** DataGrid, ListView, and TreeView.
- **Date Display and Selection:** Calendar and DatePicker.
- **Dialog Boxes:** OpenFileDialog, PrintDialog, and SaveFileDialog.
- **Digital Ink:** InkCanvas and InkPresenter.
- **Documents:** DocumentViewer, FlowDocumentPageViewer, FlowDocumentReader, FlowDocumentScrollViewer, and StickyNoteControl.
- **Input:** TextBox, RichTextBox, and PasswordBox.
- **Layout:** Border, BulletDecorator, Canvas, DockPanel, Expander, Grid, GridView, GridSplitter, GroupBox, Panel, ResizeGrip, Separator, ScrollBar, ScrollViewer, StackPanel, Thumb, Viewbox, VirtualizingStackPanel, Window, and WrapPanel.
- **Media:** Image, MediaElement, and SoundPlayerAction.
- **Menus:** ContextMenu, Menu, and ToolBar.
- **Navigation:** Frame, Hyperlink, Page, NavigationWindow, and TabControl.
- **Selection:** CheckBox, ComboBox, ListBox, RadioButton, and Slider.
- **User Information:** AccessText, Label, Popup, ProgressBar, StatusBar, TextBlock, and ToolTip.

LAYOUT

When you create a UI, you arrange your controls by location and size to form a layout. A key requirement of any layout is to adapt to changes in window size and display settings. Rather than forcing you to write the code to adapt a layout in these circumstances, WPF provides a first-class, extensible layout system for you.

The cornerstone of the layout system is relative positioning, which increases the ability to adapt to changing window and display conditions. In addition, the layout system manages the negotiation between controls to determine the layout. The negotiation is a two-step process: first, a control tells its parent what location and size it requires; second, the parent tells the control what space it can have.

The layout system is exposed to child controls through base WPF classes. For common layouts such as grids, stacking, and docking, WPF includes several layout controls:

- **Canvas :** Child controls provide their own layout.
- **DockPanel :** Child controls are aligned to the edges of the panel.
- **Grid :** Child controls are positioned by rows and columns.
- **StackPanel :** Child controls are stacked either vertically or horizontally.
- **VirtualizingStackPanel :** Child controls are virtualized and arranged on a single line that is either horizontally or vertically oriented.
- **WrapPanel :** Child controls are positioned in left-to-right order and wrapped to the next line when there are more controls on the current line than space allows.

GRAPHICS

WPF introduces an extensive, scalable, and flexible set of graphics features that have the following benefits:

- **Resolution-independent and device-independent graphics.** The basic unit of measurement in the WPF graphics system is the device independent pixel, which is 1/96th of an inch, regardless of actual screen resolution, and provides the foundation for resolution-independent and device-independent rendering. Each device-independent pixel automatically scales to match the dots-per-inch (dpi) setting of the system it renders on.
- **Improved precision.** The WPF coordinate system is measured with double-precision floating-point numbers rather than single-precision. Transformations and opacity values are also expressed as double-precision. WPF also supports a wide color gamut (scRGB) and provides integrated support for managing inputs from different color spaces.
- **Advanced graphics and animation support.** WPF simplifies graphics programming by managing animation scenes for you; there is no need to worry about scene processing, rendering loops, and bilinear interpolation. Additionally, WPF provides hit-testing support and full alpha-compositing support.
- **Hardware acceleration.** The WPF graphics system takes advantage of graphics hardware to minimize CPU usage.

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)



XAML stands for Extensible Application Markup Language. It's a simple language based on XML to create and initialize .NET objects with hierarchical relations. Although it was originally invented for WPF, it can be used to create any kind of object trees.

Today XAML is used to create user interfaces in WPF, Silverlight, declare workflows in WF and for electronic paper in the XPS standard.

All classes in WPF have parameterless constructors and make excessive use of properties. That is done to make it perfectly fit for XML languages like XAML.

All you can do in XAML can also be done in code. XAML is just another way to create and initialize objects. You can use WPF without using XAML. It's up to you if you want to declare it in XAML or write it in code. Declaring your UI in XAML has some advantages:

- XAML code is short and clear to read
- Separation of designer code and logic

- Graphical design tools like Expression Blend require XAML as source.
- The separation of XAML and UI logic allows it to clearly separate the roles of designer and developer.

PROGRAMMING FRAMEWORK

.NET 4.5



The .NET Framework is a development platform for building apps for Windows, Windows Phone, Windows Server, and Windows Azure. It consists of the common language runtime (CLR) and the .NET Framework class library, which includes classes, interfaces, and value types that support an extensive range of technologies. The .NET Framework provides a managed execution environment, simplified development and deployment, and integration with a variety of programming languages, including Visual Basic and Visual C#.

.NET FRAMEWORK CLASS LIBRARIES

The .NET Framework class library is a library of classes, interfaces, and value types that provide access to system functionality. It is the foundation on which .NET Framework applications, components, and controls are built. The namespaces and namespace categories in the class library are listed in the following table and documented in detail in this reference. The namespaces and categories are listed by usage, with the most frequently used namespaces appearing first.

Namespace	Description
System	The System namespace contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.
System.Activities	The System.Activities namespaces contain all the classes necessary to create and work with activities in Window Workflow Foundation.

System.AddIn	The System.AddIn namespaces contain types used to identify, register, activate, and control add-ins, and to allow add-ins to communicate with a host application.
System.CodeDom	The System.CodeDom namespaces contain classes that represent the elements of a source code document and that support the generation and compilation of source code in supported programming languages.
System.Collections	The System.Collections namespaces contain types that define various standard, specialized, and generic collection objects.
System.ComponentModel	The System.ComponentModel namespaces contain types that implement the run-time and design-time behavior of components and controls. Child namespaces support the Managed Extensibility Framework (MEF), provide attribute classes that define metadata for ASP.NET Dynamic Data controls, and contain types that let you define the design-time behavior of components and their user interfaces.
System.Configuration	The System.Configuration namespaces contain types for handling configuration data, such as data in machine or application configuration files. Child namespaces contain types that are used to configure an assembly, to write custom installers for components, and to support a pluggable model for adding functionality to, or removing functionality from, both client and server applications.
System.Data	The System.Data namespaces contain classes for accessing and managing data from diverse sources. The top-level namespace and a number of the child namespaces together form the ADO.NET architecture and ADO.NET data providers. For example, providers are available for SQL Server, Oracle, ODBC, and OleDb. Other child namespaces contain classes used by the ADO.NET Entity Data Model (EDM) and by WCF Data Services.
System.Deployment	The System.Deployment namespaces contain types that support deployment of ClickOnce applications.
System.Device.Location	The System.Device.Location namespace allows application developers to easily access the computer's location by using a single API. Location information may come from multiple providers, such as GPS, Wi-Fi triangulation, and cell phone tower triangulation. The System.Device.Location classes provide a single API to encapsulate the multiple location providers on a computer and support seamless prioritization and transitioning between them. As a result, application

	<p>developers who use this API do not need to tailor applications to specific hardware configurations.</p>
System.Diagnostics	<p>The System.Diagnostics namespaces contain types that enable you to interact with system processes, event logs, and performance counters. Child namespaces contain types to interact with code analysis tools, to support contracts, to extend design-time support for application monitoring and instrumentation, to log event data using the Event Tracing for Windows (ETW) tracing subsystem, to read to and write from event logs and collect performance data, and to read and write debug symbol information.</p>
System.DirectoryServices	<p>The System.DirectoryServices namespaces contain types that provide access to Active Directory from managed code.</p>
System.Drawing	<p>The System.Drawing parent namespace contains types that support basic GDI+ graphics functionality. Child namespaces support advanced two-dimensional and vector graphics functionality, advanced imaging functionality, and print-related and typographical services. A child namespace also contains types that extend design-time user-interface logic and drawing.</p>
System.Dynamic	<p>The System.Dynamic namespace provides classes and interfaces that support Dynamic Language Runtime.</p>
System.EnterpriseServices	<p>The System.EnterpriseServices namespaces contain types that define the COM+ services architecture, which provides an infrastructure for enterprise applications. A child namespace supports Compensating Resource Manager (CRM), a COM+ service that enables non-transactional objects to be included in Microsoft Distributed Transaction Coordinator (DTC) transactions. Child namespaces are described briefly in the following table and documented in detail in this reference.</p>
System.Globalization	<p>The System.Globalization namespace contains classes that define culture-related information, including language, country/region, calendars in use, format patterns for dates, currency, and numbers, and sort order for strings. These classes are useful for writing globalized (internationalized) applications. Classes such as StringInfo and TextInfo provide advanced globalization functionalities, including surrogate support and text element processing.</p>
System.IdentityModel	<p>The System.IdentityModel namespaces contain types that are used to provide authentication and authorization for .NET applications.</p>

System.IO	The System.IO namespaces contain types that support input and output, including the ability to read and write data to streams either synchronously or asynchronously, to compress data in streams, to create and use isolated stores, to map files to an application's logical address space, to store multiple data objects in a single container, to communicate using anonymous or named pipes, to implement custom logging, and to handle the flow of data to and from serial ports.
System.Linq	The System.Linq namespaces contain types that support queries that use Language-Integrated Query (LINQ). This includes types that represent queries as objects in expression trees.
System.Management	The System.Management namespaces contain types that provide access to management information and management events about the system, devices, and applications instrumented to the Windows Management Instrumentation (WMI) infrastructure. These namespaces also contain types necessary for instrumenting applications so that they expose their management information and events through WMI to potential customers.
System.Media	The System.Media namespace contains classes for playing sound files and accessing sounds provided by the system.
System.Messaging	The System.Messaging namespaces contain types that enable you to connect to, monitor, and administer message queues on the network and to send, receive, or peek messages. A child namespace contains classes that can be used to extend design-time support for messaging classes.
System.Net	The System.Net namespaces contain classes that provide a simple programming interface for a number of network protocols, programmatically access and update configuration settings for the System.Net namespaces, define cache policies for web resources, compose and send e-mail, represent Multipurpose Internet Mail Exchange (MIME) headers, access network traffic data and network address information, and access peer-to-peer networking functionality. Additional child namespaces provide a managed implementation of the Windows Sockets (Winsock) interface and provide access to network streams for secure communications between hosts.
System.Numerics	The System.Numerics namespace contains numeric types that complement the numeric primitives, such as Byte , Double , and Int32 , that are defined by the .NET Framework.

System.Printing	The System.Printing namespaces contain types that support printing, that provide access to the properties of print system objects and enable rapid copying of their property settings to another object of the same type, and that support the interconversion of managed System.PrintTicket objects and unmanaged GDI DEVMODE structures.
System.Reflection	The System.Reflection namespaces contain types that provide a managed view of loaded types, methods, and fields, and that can dynamically create and invoke types. A child namespace contains types that enable a compiler or other tool to emit metadata and Microsoft intermediate language (MSIL).
System.Resources	The System.Resources namespaces contain types that enable developers to create, store, and manage an application's culture-specific resources.
System.Runtime	The System.Runtime namespaces contain types that support an application's interaction with the common language runtime, and types that enable features such as application data caching, advanced exception handling, application activation within application domains, COM interop, distributed applications, serialization and deserialization, and versioning. Additional namespaces enable compiler writers to specify attributes that affect the run-time behavior of the common language runtime, define a contract for reliability between a set of code and other code that takes a dependency on it, and implement a persistence provider for Windows Communication Foundation (WCF).
System.Security	The System.Security namespaces contain classes that represent the .NET Framework security system and permissions. Child namespaces provide types that control access to and audit securable objects, allow authentication, provide cryptographic services, control access to operations and resources based on policy, and support rights management of application-created content.
System.ServiceModel	The System.ServiceModel namespaces contain the types necessary to build Windows Communication Foundation (WCF) service and client applications.
System.ServiceProcess	The System.ServiceProcess namespaces contain types that enable you to implement, install, and control Windows service applications and extend design-time support for Windows service applications.
System.Speech	The System.Speech namespaces contain types that support speech recognition.

System.Text	The System.Text namespaces contain types for character encoding and string manipulation. A child namespace enables you to process text using regular expressions.
System.Threading	The System.Threading namespaces contain types that enable multithreaded programming. A child namespace provides types that simplify the work of writing concurrent and asynchronous code.
System.Timers	The System.Timers namespace provides the Timer component, which allows you to raise an event on a specified interval.
System.Transactions	The System.Transactions namespaces contain types that support transactions with multiple, distributed participants, multiple phase notifications, and durable enlistments. A child namespace contains types that describe the configuration options used by the System.Transactions types.
System.Web	The System.Web namespaces contain types that enable browser/server communication. Child namespaces include types that support ASP.NET forms authentication, application services, data caching on the server, ASP.NET application configuration, dynamic data, HTTP handlers, JSON serialization, incorporating AJAX functionality into ASP.NET, ASP.NET security, and web services.
System.Windows	The System.Windows namespaces contain types used in Windows Presentation Foundation (WPF) applications, including animation clients, user interface controls, data binding, and type conversion. System.Windows.Forms and its child namespaces are used for developing Windows Forms applications.
System.Workflow	The System.Workflow namespaces contain types used to develop applications that use Windows Workflow Foundation. These types provide design time and run-time support for rules and activities, to configure, control, host, and debug the workflow runtime engine.
System.Xaml	The System.Xaml namespaces contain types that support parsing and processing the Extensible Application Markup Language (XAML).
System.Xml	The System.Xml namespaces contain types for processing XML. Child namespaces support serialization of XML documents or streams, XSD schemas, XQuery 1.0 and XPath 2.0, and LINQ to XML, which is an in-memory XML

	programming interface that enables easy modification of XML documents.
Accessibility	The Accessibility and all of its exposed members are part of a managed wrapper for the Component Object Model (COM) accessibility interface.
Microsoft.Activities	The Microsoft.Activities namespaces contain types that support MSBuild and debugger extensions for Windows Workflow Foundation applications.
Microsoft.AspNet.Snapin	The Microsoft.AspNet.Snapin namespace defines the types necessary for the ASP.NET management console application to interact with Microsoft Management Console (MMC). For more information, see "MMC Programmer's Guide" in the MSDN Library .
Microsoft.Build	The Microsoft.Build namespaces contain types that provide programmatic access to, and control of, the MSBuild engine.
Microsoft.CSharp	The Microsoft.CSharp namespaces contain types that support compilation and code generation of source code written in the C# language, and types that support interoperation between the dynamic language runtime (DLR) and C#.
Microsoft.Data.Entity.Build.Tasks	The Microsoft.Data.Entity.Build.Tasks namespace contains two MSBuild tasks that are used by the ADO.NET Entity Data Model Designer (Entity Designer).
Microsoft.JScript	The Microsoft.JScript namespaces contain classes that support compilation and code generation using the JScript language.
Microsoft.SqlServer.Server	The Microsoft.SqlServer.Server namespace contains classes, interfaces, and enumerations that are specific to the integration of the Microsoft .NET Framework common language runtime (CLR) into Microsoft SQL Server, and the SQL Server database engine process execution environment.
Microsoft.VisualBasic	The Microsoft.VisualBasic namespaces contain classes that support compilation and code generation using the Visual Basic language. Child namespaces contain types that provide services to the Visual Basic compiler and types that include support for the Visual Basic application model, the My namespace, lambda expressions, and code conversion.

Microsoft.VisualBasic	The Microsoft.VisualBasic namespaces contain types that support the Visual C++ compiler and types that implement the STL/CLR Library and the generic interface to the STL/CLR Library.
Microsoft.Win32	The Microsoft.Win32 namespaces provide types that handle events raised by the operating system, that manipulate the system registry, and that represent file and operating system handles.
Microsoft.Windows	The Microsoft.Windows namespaces contain types that support themes and preview in Windows Presentation Framework (WPF) applications.
UIAutomationClientsideProviders	Contains a single type that maps client automation providers.
XamlGeneratedNamespace	Contains compiler-generated types that are not intended to be used directly from your code.

DATABASE/BACKEND:

MYSQL



MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout its history.

The MySQL Community Edition includes:

- Pluggable Storage Engine Architecture

- Multiple Storage Engines: InnoDB , MyISAM, NDB (MySQL Cluster),Memory ,Merge , Archive, CSV
- MySQL Replication to improve application performance and scalability
- MySQL Partitioning to improve performance and management of large database applications
- Stored Procedures to improve developer productivity

DETAILED FEATURES OF MYSQL

The following list shows the most important properties of MySQL. This section is directed to the reader who already has some knowledge of relational databases. We will use some terminology from the relational database world without defining our terms exactly. On the other hand, the explanations should make it possible for database novices to understand to some extent what we are talking about.

Relational Database System: Like almost all other database systems on the market, MySQL is a relational database system.

Client/Server Architecture: MySQL is a client/server system. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc. The clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

Almost all of the familiar large database systems (Oracle, Microsoft SQL Server, etc.) are client/server systems. These are in contrast to the file-server systems, which include Microsoft Access, dBase and FoxPro. The decisive drawback to file-server systems is that when run over a network, they become extremely inefficient as the number of users grows.

SQL compatibility: MySQL supports as its database language -- as its name suggests -- SQL (Structured Query Language). SQL is a standardized language for querying and updating data and for the administration of a database. There are several SQL dialects (about as many as there are database systems). MySQL adheres to the current SQL standard (at the moment SQL:2003), although with significant restrictions and a large number of extensions.

Through the configuration setting `sql-mode` you can make the MySQL server behave for the most part compatibly with various database systems. Among these are IBM DB/2 and Oracle. (The setting `sql-mode` changes some of the syntax conventions, and performs no miracles.

SubSELECTs: Since version 4.1, MySQL is capable of processing a query in the form `SELECT * FROM table1 WHERE x IN (SELECT y FROM table2)` (There are also numerous syntax variants for subSELECTs.)

Views: Put simply, views relate to an SQL query that is viewed as a distinct database object and makes possible a particular view of the database. MySQL has supported views since version 5.0.

Stored procedures: Here we are dealing with SQL code that is stored in the database system.

Stored procedures (SPs for short) are generally used to simplify certain steps, such as inserting or deleting a data record. For client programmers this has the advantage that they do not have to process the tables directly, but can rely on SPs. Like views, SPs help in the administration of large database projects. SPs can also increase efficiency. MySQL has supported SPs since version 5.0.

Triggers: Triggers are SQL commands that are automatically executed by the server in certain database operations (INSERT, UPDATE, and DELETE). MySQL has supported triggers in a limited form from version 5.0, and additional functionality is promised for version 5.1.

Unicode: MySQL has supported all conceivable character sets since version 4.1, including Latin-1, Latin-2, and Unicode (either in the variant UTF8 or UCS2).

User interface: There are a number of convenient user interfaces for administering a MySQL server.

Full-text search: Full-text search simplifies and accelerates the search for words that are located within a text field. If you employ MySQL for storing text (such as in an Internet discussion group), you can use full-text search to implement simply an efficient search function.

Replication: Replication allows the contents of a database to be copied (replicated) onto a number of computers. In practice, this is done for two reasons: to increase protection against system failure (so that if one computer goes down, another can be put into service) and to improve the speed of database queries.

Transactions: In the context of a database system, a transaction means the execution of several database operations as a block. The database system ensures that either all of the operations are correctly executed or none of them. This holds even if in the middle of a transaction there is a power failure, the computer crashes, or some other disaster occurs. Thus, for example, it cannot occur that a sum of money is withdrawn from account A but fails to be deposited in account B due to some type of system error.

Transactions also give programmers the possibility of interrupting a series of already executed commands (a sort of revocation). In many situations this leads to a considerable simplification

of the programming process. In spite of popular opinion, MySQL has supported transactions for a long time. One should note here that MySQL can store tables in a variety of formats. The default table format is called MyISAM, and this format does not support transactions. But there are a number of additional formats that do support transactions. The most popular of these is InnoDB, which will be described extensively in this book.

Foreign key constraints: These are rules that ensure that there are no cross references in linked tables that lead to nowhere. MySQL supports foreign key constraints for InnoDB tables.

GIS functions: Since version 4.1, MySQL has supported the storing and processing of two-dimensional geographical data. Thus MySQL is well suited for GIS (geographic information systems) applications.

Programming languages: There are quite a number of APIs (application programming interfaces) and libraries for the development of MySQL applications. For client programming you can use, among others, the languages C, C++, Java, Perl, PHP, Python, and Tcl.

ODBC: MySQL supports the ODBC interface Connector/ODBC. This allows MySQL to be addressed by all the usual programming languages that run under Microsoft Windows (Delphi, Visual Basic, etc.). The ODBC interface can also be implemented under Unix, though that is seldom necessary.

Windows programmers who have migrated to Microsoft's new .NET platform can, if they wish, use the ODBC provider or the .NET interface Connector/.NET.

Platform independence: It is not only client applications that run under a variety of operating systems; MySQL itself (that is, the server) can be executed under a number of operating systems. The most important are Apple Macintosh OS X, Linux, Microsoft Windows, and the countless Unix variants, such as AIX, BSDI, FreeBSD, HP-UX, OpenBSD, Net BSD, SGI Iris, and Sun Solaris.

Speed: MySQL is considered a very fast database program. This speed has been backed up by a large number of benchmark.

IDE FOR DATABASE

MYSQL WORKBENCH



MySQL Workbench is a visual database design tool that integrates SQL development, administration, database design, creation and maintenance into a single integrated development environment for the MySQL database system. It is the successor to DBDesigner 4 from fabFORCE.net, and replaces the previous package of software, MySQL GUI Tools Bundle. [MySQL Workbench](#) enables a DBA, developer, or data architect to visually design, generate, and manage all types of databases including Web, OLTP, and data warehouse databases. It includes everything a data modeler needs for creating complex ER models, and also delivers key features for performing difficult change management and documentation tasks that normally require much time and effort. MySQL Workbench is available on Windows, Linux and Mac OS.

BENEFITS

- Simplifies database design and maintenance
- Automates time-consuming and error-prone tasks
- Enables data architects to visualize requirements, communicate with stakeholders, and resolve design issues before a major investment of time and resources is made
- Enables model-driven database design—the most efficient methodology for creating valid and well-performing databases—while providing the flexibility to respond to evolving business requirements
- Provides capabilities to forward-engineer physical database designs and reverse-engineer existing databases
- Allows you to import SQL scripts to build models and export models to DDL scripts that can be run at a later time
- Enables you to compare two live databases or a model and a live database, visually see the differences, and perform a synchronization between a model and a live database or vice versa
- Simplifies the documentation of database designs, providing a point-and-click process that delivers documentation in HTML or plain-text format

TOOLS

The three main tools of MySQL Workbench are:

- SQL Development
- Data Modelling
- Server Administration



C# is a type-safe, object-oriented language that is simple yet powerful, allowing programmers to build a breadth of applications. C# is a multi-paradigm programming language encompassing imperative, declarative, functional, generic, object-oriented(class-based), and component-oriented programming disciplines. It was developed by Microsoft within the .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270). C# is one of the programming languages designed for the Common Language Infrastructure.

C# is intended to be a simple, modern, general-purpose, object-oriented programming language.

C# was developed to bring rapid development to C++ without sacrificing the power and control of C and C++. C# provides various characteristics, which are:

Simple:

C# eliminates the use of tedious operators such as -->, and pointers. C# treats inter and Boolean as two different data types, which enable the compiler

to recognize the use of = in place of == with if statement.

Consistent:-

C# supports only one integer type and there is no limitation of range.

Modern:-

C# contains various features necessary to develop web applications. Following are the features of C#:

It provides automatic garbage collection.

It provides robust security model.

It provides decimal data type for financial application.

It provides modern approach for debugging.

It provides a rich intrinsic model for error handling.

Object Oriented:-

C# supports all the features of object oriented language such as encapsulation, inheritance and polymorphism. It treats everything as an object and there are no global functions, variables and constants in C#.

Type Safe:-

C# provides various type safe measures, which are:

Dynamically allocated objects and arrays are initialised to zero.

Produces an error message while using an uninitialised variable.

Checks the range of an array and warns when the access goes out of bound.

Unsafe casts are not allowed.

Enforces overflow checking in arithmetic operations.

Versionable:-

C# supports versioning that enables the existing applications to run on different versions with the help of new and override command.

Compatible:

C# contains the .NET specifications and therefore, allows inter operation with other .NET languages.

Flexible:-

C# does not support pointers but you may use pointers to manipulate the data of certain classes and methods by declaring them unsafe.

Inter-operability:

C# enables a program to call out any native API. It also allows the use of COM objects written in different languages.

MOBILE APP DEVELOPMENT

Quickly and efficiently create and test Java™ applications with the Nokia SDKs for Java and the Series 40 platform SDKs.

The Nokia SDKs for Java provide the development tools for phones containing a Java Runtime for Series 40. Each release of the Java Runtime has a corresponding Nokia SDK.

For earlier platform releases Series 40 platform SDKs offer versions to support specific editions and feature packs.

Within both families of tools, each SDK includes Java APIs, an emulator, documentation, code examples, and emulator based debugging tools. The SDKs can be used with either the NetBeans or Eclipse IDEs to create, compile, and package applications and content. Applications can be tested using the emulator.

NOKIA SDK 2.0 FOR JAVA — FOR SERIES 40 APPS

Create apps for Series 40 phones with the Java Runtime 2.0.0, including the full-touch UI equipped Nokia Asha 305, Nokia Asha 306, and Nokia Asha 311 using the Nokia SDK 2.0 for Java. Then test your apps in an emulator based on the Nokia Asha 305. In addition to the features of the Nokia SDK 1.1 for Java, the 2.0 SDK offers:

LWUIT FOR SERIES 40 ARRIVES AT 1.0



LWUIT for Series 40 has graduated beta to a full productised release. With a number of new APIs — such as `PopUpChoiceGroup`, `ContextMenu`, and `NokiaListCellRenderer` — the 1.0 release includes significant improvements in performance, particularly in lists, themes loading, and the `HTMLComponent`. Compatibility with the native full-touch UI has been fine-tuned and many bugs fixed, particularly in command handling and text input. A LWUIT Developer's Library has also been released, providing full technical and design guides. There are many new examples too.

This release is delivered to the Nokia SDK 2.0 for Java through the SDK Manager, while a download for the Nokia SDK 1.1 for Java is available from LWUIT for Series 40 project.

THE NOKIA SDK 2.0 FOR JAVA ADDS NEW FEATURES



The Nokia SDK 2.0 for Java delivers everything you need to develop apps for the exciting new full-touch UI equipped Nokia Asha 305, Nokia Asha 306, and Nokia Asha 311 phones. And now the SDK has graduated from beta.

The SDK delivers the updated Nokia UI API for advanced touch interaction, the Mobile Sensor API (JSR-256) to take advantage of the orientation sensors on the latest Series 40 phones, and the in-app purchase APIs. Now LWUIT is included as a plug-in, so you can create slick UIs, faster. In addition, the emulator gains features for the simulation of multipoint-touch gestures, such as pinch-to-zoom, and PC keyboard input. Based on the Nokia Asha 305, the emulator also provides improved sensor and location support so you can test more of your app on a PC.

The Nokia IDE for Java ME (Eclipse) has also been enhanced with improved searching in the Device SDK Manager and a tool that lets you pull code examples directly into the IDE. Building on the power of the Eclipse platform for

Java development, the Nokia IDE makes delivering your Series 40 Java apps easier with features such as an editor for Nokia specific JAD attributes.

TESTING YOUR JAVA APPS FOR SERIES 40 USING REMOTE DEVICE ACCESS



Testing your Java apps on several Series 40 phones is easy and cheap with remote Device Access. Remote Device Access offers a range of Series 40 phones that you access over the internet free-of-charge. So when you need to test your app you can simply pop-on the internet, book a phone, install your app, and you will be testing in minutes.

Right now, the Nokia Asha 311 is available for you to test your apps.

EXPLORE IN-APP PURCHASING IN YOUR JAVA APPS



Using the Nokia SDK 1.1 for Java or later you can explore adding in-app purchase features to your apps, a feature available on phones with Java Runtime 1.1.0 for Series 40 or later. Now you can generate revenue by offering users digital assets and content as part of the app experience — and allow them to purchase these items without leaving your application.

ADD A NEW DIMENSION TO LOCATION WITH THE MAPS API



Leveraging the location information provided by the Location API for J2ME™ (JSR-179) on Series 40 phones, you can add rich maps to your apps with the Maps API for Java ME. With exciting features, such as custom overlays, you can create a unique experience. And with the release of the Nokia SDK 2.0 for Java, you now get the Maps APIs delivered ready for your use without additional downloads. Find out more about creating location aware applications with Java technology ›

NOKIA WEB - TOOLS

SERIES 40 WEB APPS TOOLS

NOKIA WEB TOOLS 2.3

DESCRIPTION

Nokia Web Tools provides a set of tools that enable the creation of Series 40 web apps. The tools included are:

- Web Developer Environment (WDE) — enabling web apps to be created, edited, packaged and deployed.
- Web App Simulator (WAS) — enabling web apps to be previewed and debugged on a computer.
- Web Developer Channel (WDC) — included in Web Developer Environment, to deliver information and tools to facilitate web app development.

WHAT'S NEW

This new version of Nokia Web Tools provides:

- UI Designer in WDE offering drag-and-drop population of the web app's UI.
- the ability to deploy a web app over a USB connection from a PC running Microsoft Windows.
- automatic reloading of the simulator for locally previewed web apps as code changes are saved.
- additional templates upon which to base new web apps, including web apps for trivia games, shopping, and video browsing among others.
- more sample web apps and snippets to help developers use the platform capabilities easily.

SYMBIAN WRT WIDGET DEVELOPMENT

This version of Nokia Web Tools no longer supports Symbian WRT widget development. If you wish to continue using [Nokia Web Tools 1.2](#) for Symbian WRT widget development, please refer to the Installation Guide for details on the setup requirements.

APP PUBLICATION

Web apps created with Nokia Web Tools can be submitted for publication in Nokia Store.

VERSIONS AVAILABLE

Nokia Web Tools are available in versions for:

- 32- or 64-bit Microsoft Windows XP Service Pack 2, Windows Vista, or Windows 7.
- 32-bit Ubuntu Linux 10.04.
- 64-bit Apple Mac OS X 10.6.

To make your Series 40 web apps development as straightforward as possible Nokia Web Tools, Bluetooth Launcher, and the Nokia Xpress Browser are available in the Series 40 web apps section.

Alternatively, Xpress Web App Builder is an online tool that enables content owners to create web apps from clipped content, RSS feeds, and social media content using a wide selection of formats.

You create Series 40 web apps using Nokia Web Tools. Based on Eclipse, Nokia Web Tools builds on the powerful web development features of the Eclipse Web Tools Platform to create the Web Developer Environment (WDE). WDE includes features to create, edit, validate, test, package, and deploy Series 40 web apps. Testing is supported by the Web Apps Simulator (WAS) that enables web apps to be run and tested on a computer. WAS includes an implementation of Web Inspector, so you can perform debugging and examine of a web app's content and performance. This getting started guide takes you through installing Nokia Web Tools, creating a web app from a template, testing it on your computer, and running it on a phone, before providing links to the resources you need to build great web apps and deliver them to Nokia Store.

WDE offers a number of templates you can use to create Series 40 web apps easily. These templates range from the Basic web app template, which contains the core web app files with no functionality, through a selection of templates offering basic UI constructions to fully functional web apps, such as the Videos browsing project template that offers a working web app to browse videos. You can work with web app examples or a web app project you have already created as well. For more details on importing web apps, see [Importing a web app or web app project](#) in the Series 40 Web App Developer's Library.

During development, transferring a web app onto a phone each time you make code changes isn't a practical way of previewing and testing your web app. To simplify testing of a web app, WDE integrates with the Web App Simulator (WAS) to enable testing on your computer.

You have two options for running your web app in WAS:

- a server (cloud) based preview, this option provides a simulation that is very close to the experience that will be seen on a phone.
- a local preview, which is useful when you are working offline or want to debug your web app.

Having tested your web app in the simulator the next stage is to run it on a Series 40 phone. You have three ways to do this::

- deploying the web app to a phone from WDE over a Bluetooth connection.
- deploying the web app from WDE to a phone over a USB connection (but only if you are working on a Microsoft Windows PC as [Nokia Suite](#) or [Nokia PC Suite](#) is required).
- running the web app by entering a short URL into the Nokia Xpress Browser on a phone.

SERIES 40 WEB APP DEVELOPER'S LIBRARY

The Series 40 Web App Developer's Library describes the Series 40 web apps development environment for Series 40 phones that run the Xpress Browser for Series 40, the tools for developing Series 40 web apps, and the design considerations for Series 40 web apps.

XPRESS WEB APP BUILDER

Xpress Web App Builder is an online tool that guides you through the process of creating rich web apps, with no coding required. Select from a variety of templates, customise your theme, and then add clipped web content, RSS feeds, and social media information. The key features of the tool are:

- layout templates to present content, including single pane, tabbed view, and accordion view, as well as focused templates for news, pictures, and video content.
- a wide range of content widgets for clipped web content; RSS feeds; video from YouTube; pictures from Flickr, Picasa, and other photo sharing sites; and blogs from Tumblr and WordPress.
- the ability to add SMMS and call capabilities, static HERE Maps, and in-app advertising from [Nokia Ad Exchange](#).
- the option to customise your app's colour scheme, including header and font colours.
- static and dynamic previews of your app, for all supported screen resolutions.

When you've completed your web app, the tool provides a short URL for testing the app on your phone, and lets you submit the app to Nokia Publish to start the process of publication in Nokia store. However, if you want to customise your web app further, you can download the source code and import it into Nokia Web Tools.

TEST YOUR SERIES 40 WEB APPS

If you don't have access to a Series 40 phone, you can test your web content and apps by making use of the Remote Device Access service. This service provides you with access to ten Series 40 phone models, more than 30 phones, over an internet connection. The service is available free to all Nokia Developer members.

MOBILE WEB COMPONENTS

Make the most of the latest HTML5 feature in Nokia browsers.

Add rich, HTML5 based components to your web pages and web apps for Symbian Anna phones and the Nokia N9 Smartphone. Included are components for collapsible content blocks, scrollable large content item windows, pop-up menus, expandable sliding menus, slideshows, and others.

LEVERAGE THE POWER OF QTWEBKIT

Using Qt WebKit technology, Web developer can easily transform web apps and web services into powerful native applications. Qt offers HTML5 and CSS3 support today. The quick and powerful way to use web assets and skills to produce apps for smartphones and mobile computers.

DEVICE APIS::API BRIDGE

APIBridge is a component for Nokia Symbian devices that enables WRT widgets, Adobe Flash Lite content, and Java applications to access device features through a plug-in architecture. The APIBridge package ships with a set of plug-ins and the components to enable the features of the plug-ins to be used. Developers can extend the APIBridge component with their own plug-ins.

OTHER TECHNOLOGIES

GITHUB – REPOSITORY MANAGEMENT TOOL



GitHub is a web-based hosting service for software development projects that use the Git revision control system. GitHub offers both paid plans for private repositories, and free accounts for open source projects. As of May 2011, GitHub was the most popular open source code repository site. GitHub Inc. was founded in 2008 and is based in San Francisco, California.

DESCRIPTION

The site provides social networking functionality such as feeds, followers and the network graph to display how developers work on their versions of a repository.

GitHub also operates other services: a pastebin-style site called Gist that provides wikis for individual repositories and web pages that can be edited through a Git repository, a slide hosting service called Speaker Deck, and a web analytics platform called Gauges.

As of January 2010, GitHub is operated under the name GitHub, Inc.

The software that runs GitHub was written using Ruby on Rails and Erlang by GitHub, Inc. (previously known as Logical Awesome) developers Chris Wanstrath, PJ Hyett, and Tom Preston-Werner.

LIMITATIONS AND CONSTRAINTS

According to the terms of service, if an account's bandwidth usage significantly exceeds the average of other GitHub customers, the account's file hosting service may be immediately disabled or throttled until bandwidth consumption is reduced. In addition, while there is no hard limit, the guideline for the maximum size of a repository is one gigabyte.

DIA FOR DIAGRAM DRAWING & MODELING

Dia is free and open source general-purpose diagramming software, developed as part of the GNOME project's office suite and was originally created by Alexander Larsson. Dia uses a controlled single document interface (CSDI) similar to GIMP and Sodipodi.

Dia has a modular design with several shape packages available for different needs: flowchart, network diagrams, circuit diagrams, and more. It does not restrict symbols and connectors from various categories from being placed together.

Dia is a gtk+ based diagram creation program released under the GPL license.

Dia is inspired by the commercial Windows program 'Visio', though more geared towards informal diagrams for casual use. It can be used to draw many different kinds of diagrams. It currently has special objects to help draw entity relationship diagrams, UML diagrams, flowcharts, network diagrams, and many other diagrams. It is also possible to add support for new shapes by writing simple XML files, using a subset of SVG to draw the shape.

It can load and save diagrams to a custom XML format (gzipped by default, to save space), can export diagrams to a number of formats, including EPS, SVG, XFIG, WMF and PNG, and can print diagrams (including ones that span multiple pages).

CACOO: ONLINE DRAWING TOOL



Cacoo is a diagram creation tool that runs in your web browser. Multiple people can work together on the same diagram in real time. Diagrams can be published directly to websites, wikis, and blogs.

CREATING DIAGRAMS

- Elements can be dragged and drop to easily create diagrams.
- Elements can be linked together with connectors.
- Connectors automatically move when elements are repositioned.
- You can use a text box and put text anywhere you like.
- You can upload images from your PC and include them in Diagrams.
- You can take screenshots of your computer from within Cacoo.
- Smart styles can easily be applied to stencils.
- You can have multiple sheets in a diagram and use them as backgrounds or layers.
- When you move the objects on your canvas, they will be snapped at the objects or grids nearby and align automatically.
- Copying, pasting and other functionality of basic drawing software is also built in to Cacoo.
- All actions are stored so there are unlimited levels of undo.
- You can import an image from the other websites by indicating the URL.
- The imported image can be easily trimmed only using your mouse.
- According to your editing status, tips will be shown on the right bottom corner of the canvas.

COLLABORATION

- You can invite collaborators to work with you in Cacoo.
- Multiple people can edit a diagram in real time.
- There is a chat function in the editor so people can communicate while creating diagrams.
- People can leave comments about the diagrams.
- Each user can set their own user icon.
- When editing with multiple people, users icons appear on selected objects.
- Sharing diagrams become much smoother. Diagrams in the shared folders can be accessible and editable by people who you have shared the folder with.

SHARING DIAGRAMS

- If you keep the diagram private then other users can't see it.
- If you make the diagram URL public, then anyone who knows the URL can see it.

- Publishing a diagram to a blog can be useful in various ways.
- You can place code into blogs to create a slideshow
- Published images always display the most recent version.
- Diagrams can be exported to SVG format (Plus Plan users only) and PNG format. (More formats will be available in the future.)
- Diagrams can be posted to Twitter/Facebook/GoogleBuzz
- Diagrams can be displayed in SVG format for printing. (Plus Plan users only. A few browsers are not supported.)

MANAGING DIAGRAMS

- Diagrams can be placed into folders.
- Diagrams can be copied.
- Diagrams can be displayed as thumbnails or as a list.

LANGUAGES AND TIME ZONES

- All pages and notification e-mails support English and Japanese
- Users can enter text from almost all languages.
- Dates are displayed relative to your local time zone.

SECURITY

- Private diagrams can only be seen by users you select.
- URLs which you do not share cannot be found by other users or search engines.
- All editing and management is protected by SSL.
- In order to access information about diagrams a Cacoo ID and password are required.
- User passwords are encrypted on Cacoo's server.

API

- You can access Cacoo using the API.
- The Cacoo API supports OAuth and an API Key.

By using the Cacoo API you are able to interact with Cacoo from other services and applications.

Authorization Methods

There are two ways to access the Cacoo API.

1. API KEY

The API key allows you make requests to the Cacoo API. You can make an API key here.

API KEY

Append your API key to requests to the API to return data from your account.(Parameter name "apiKey")

Example: <https://cacoo.com/api/v1/diagrams.json?apiKey=abcdefghijklmn>

OAUTH

OAuth 1.0a is supported as an authorization method for Cacoo. You can register applications [here](#).

APPLICATIONS

Access Token:https://cacoo.com/oauth/access_token

Authorize:<https://cacoo.com/oauth/authorize>

Request Token:https://cacoo.com/oauth/request_token

GOOGLE SPREADSHEET INTERFACE:

With Google Spreadsheets, we can easily create, share, and edit spreadsheets online. Here are a few specific things we can do:

- Import and export these file types: .xls, .csv, .txt and .ods. We can also export data to a PDF or an HTML file.*
- Format cells and edit formulas so we can calculate results and make data look the way we want it.*
- Chat in real time with others who are editing our spreadsheet.*
- Embed a spreadsheet, or a section of a spreadsheet, in our blog or website.*

GLOSSARY

SMMS Shopping Mall Management System

Apps Application

SRS Software Requirement Specification

DFD Data Flow Diagram

ERD Entity Relationship Diagram

GUI Graphical User Interface

UI User Interface

DB Database

COCOMO Constructive Cost Model

SDK Software Development Kit

WPF Windows Presentation Framework

XAML Extensible application Markup Language

IDE Integrated Development Environment

HTML Hyper Text Markup Language

www World Wide Web

DBMS Database Management System

Sync Synchronization

cs C Sharp

KLOC Estimated size of the software product expressed in Kilo

Tdev Estimated time to develop the software, expressed in months.

Effort Total effort required to develop the software product, expressed in person-month (PM).

PM Person-month