# Contents

[Contents 1](#_Toc352080037)

[Introduction: 6](#_Toc352080038)

[Objective: 7](#_Toc352080039)

[System Analysis 7](#_Toc352080040)

[Identification of Need: 7](#_Toc352080041)

[Preliminary Investigation: 7](#_Toc352080042)

[Feasibility Study: 8](#_Toc352080043)

[Project Planning & Scheduling: 8](#_Toc352080044)

[Gantt chart 8](#_Toc352080045)

[Tracking Gantt 8](#_Toc352080046)

[Pert Chart 9](#_Toc352080047)

[Software requirement specifications (SRS): 10](#_Toc352080048)

[Functional Requirement 10](#_Toc352080049)

[Non-functional Requirements 12](#_Toc352080050)

[Software Engineering Paradigm applied 12](#_Toc352080051)

[Data models 14](#_Toc352080052)

[Context Diagram 14](#_Toc352080053)

[0-Level DFD 14](#_Toc352080054)

[1-Level DFD 15](#_Toc352080055)

[2-Level DFD 16](#_Toc352080056)

[Sequence diagrams 16](#_Toc352080057)

[Interaction Event 16](#_Toc352080058)

[Entity Relationship Model, 17](#_Toc352080059)

[Class Diagrams 19](#_Toc352080060)

[Activity Diagrams 20](#_Toc352080061)

[System Design 21](#_Toc352080062)

[Modularisation details 21](#_Toc352080063)

[**CMS** Engine: 21](#_Toc352080064)

[**CMS** GUI: 22](#_Toc352080065)

[**CMS** Storage: 22](#_Toc352080066)

[Google Calendar: 22](#_Toc352080067)

[Facebook/Twitter/LinkedIn API: 22](#_Toc352080068)

[Data integrity and constraints 22](#_Toc352080069)

[Entity integrity 22](#_Toc352080070)

[Referential Integrity 22](#_Toc352080071)

[Domain Integrity 22](#_Toc352080072)

[User Defined Integrity 23](#_Toc352080073)

[Database design 23](#_Toc352080074)

[User Interface Design 24](#_Toc352080075)

[Main Window without Login 24](#_Toc352080076)

[Create New User Account 25](#_Toc352080077)

[Fields not filled properly: error message 25](#_Toc352080078)

[Miss match Password 26](#_Toc352080079)

[Incorrect user Id or Password 26](#_Toc352080080)

[After successfully login 26](#_Toc352080081)

[When the Event Field is Empty 27](#_Toc352080082)

[Write text in text field 28](#_Toc352080083)

[After Event Submitted 28](#_Toc352080084)

[After Event clear 29](#_Toc352080085)

[View All Event 29](#_Toc352080086)

[Search By date 30](#_Toc352080087)

[Tweet from Twitter 30](#_Toc352080088)

[After Facebook Sharing 31](#_Toc352080089)

[All contacts 31](#_Toc352080090)

[Password Login 32](#_Toc352080091)

[Password Info 32](#_Toc352080092)

[Load location in map 33](#_Toc352080093)

[Facebook Login Page 33](#_Toc352080094)

[Daily Notebook Login with Facebook 34](#_Toc352080095)

[Facebook allow permission 34](#_Toc352080096)

[Test Cases (Unit Test Cases and System Test Cases) 35](#_Toc352080097)

[Unit Test Cases 35](#_Toc352080098)

[System Test Cases 44](#_Toc352080099)

[Coding 49](#_Toc352080100)

[Complete Project Coding 49](#_Toc352080101)

[**CMS** GUI Design Coding: **CMS**GUI 49](#_Toc352080102)

[GUI Style : **CMS**Styles 66](#_Toc352080103)

[C# coding 72](#_Toc352080104)

[Datbase Classes : **CMS**Data 110](#_Toc352080105)

[Database connector: **CMS**Db 112](#_Toc352080106)

[Comments and Description of Coding segments 123](#_Toc352080107)

[Code Commenting 123](#_Toc352080108)

[Description of coding 124](#_Toc352080109)

[Comments and API Documentation 125](#_Toc352080110)

[Facebook Connect 125](#_Toc352080111)

[Standardization of the coding 128](#_Toc352080112)

[Code Efficiency 129](#_Toc352080113)

[Error handling 129](#_Toc352080114)

[Exceptions Overview 130](#_Toc352080115)

[Validation checks 130](#_Toc352080116)

[Testing 133](#_Toc352080117)

[Testing techniques and testing strategies used 133](#_Toc352080118)

[Database & Data Integrity Testing 133](#_Toc352080119)

[Functional Testing: 133](#_Toc352080120)

[Regression Testing: 134](#_Toc352080121)

[User Interface Testing: 134](#_Toc352080122)

[Performance Profiling: 134](#_Toc352080123)

[Load Testing: 135](#_Toc352080124)

[Stress Testing: 135](#_Toc352080125)

[Volume Testing: 136](#_Toc352080126)

[Security & Access Control Testing: 136](#_Toc352080127)

[Failover & Recovery Testing: 136](#_Toc352080128)

[Configuration Testing: 136](#_Toc352080129)

[Installation/Deploy & Back out Testing: 136](#_Toc352080130)

[Post Production Testing: 137](#_Toc352080131)

[Unit Testing: 137](#_Toc352080132)

[Smoke Testing: 137](#_Toc352080133)

[Data Migration Testing: 137](#_Toc352080134)

[Testing Plan used 138](#_Toc352080135)

[Test reports for Unit Test Cases and System Test Cases 142](#_Toc352080136)

[Test reports for Unit Test Cases 142](#_Toc352080137)

[Test reports for System Test Cases 144](#_Toc352080138)

[Debugging and Code improvement: 145](#_Toc352080139)

[Create a Sample with the Debug Class 146](#_Toc352080140)

[Using the Trace Class 147](#_Toc352080141)

[Verify That It Works 148](#_Toc352080142)

[Complete Code Listing 149](#_Toc352080143)

[Troubleshoot 150](#_Toc352080144)

[System Security measures: 151](#_Toc352080145)

[Database/data security: 151](#_Toc352080146)

[Creation of User profiles and access rights 151](#_Toc352080147)

[Cost Estimation of the Project along with Cost Estimation Model 151](#_Toc352080148)

[Estimation of development effort 153](#_Toc352080149)

[Estimation of development time 153](#_Toc352080150)

[Reports 153](#_Toc352080151)

[Future scope and further enhancement of the Project 154](#_Toc352080152)

[Bibliography 154](#_Toc352080153)

[Website 154](#_Toc352080154)

[Books 156](#_Toc352080155)

[Appendices 156](#_Toc352080156)

[IDE Used: 156](#_Toc352080157)

[Visual Studio 2010 156](#_Toc352080158)

[Front End - WPF (Windows Presentation Framework) 158](#_Toc352080159)

[Extensible application Markup Language (XAML) 161](#_Toc352080160)

[Programming Framework 161](#_Toc352080161)

[.NET 4.5 161](#_Toc352080162)

[Database/backend: 170](#_Toc352080163)

[MySQL 170](#_Toc352080164)

[ide for Database 173](#_Toc352080165)

[MySQL workbench 173](#_Toc352080166)

[Programming Language 175](#_Toc352080167)

[C# - C sharp 175](#_Toc352080168)

[Dia for Diagram Drawing &Modeling 177](#_Toc352080169)

[Google Spreadsheet Interface: 178](#_Toc352080170)

[Cacoo:: online drawing tool 178](#_Toc352080171)

[Creating Diagrams 178](#_Toc352080172)

[Collaboration 179](#_Toc352080173)

[Sharing Diagrams 179](#_Toc352080174)

[Managing Diagrams 179](#_Toc352080175)

[Languages and Time Zones 179](#_Toc352080176)

[Security 179](#_Toc352080177)

[API 180](#_Toc352080178)

[Version Control System : GitHub 180](#_Toc352080179)

[Description 180](#_Toc352080180)

[Limitations and constraints 181](#_Toc352080181)

[Glossary. 181](#_Toc352080182)

# Introduction & Objective

## Introduction

With the progress of advanced technologies, managing a construction site is being more

and more complicated these days. Whether it’s a personal residence or a huge multiplex, every person wants it to be developed as fast as possible. Constructors struggle to complete

their projects in time as they need to manage lots of resource, i.e. various raw materials, labors, engineers, money etc.

So, it is really necessary to develop an application that would let contractor/company mange each and every resource through his project sufficiently.

Construction Site Management System is developed to minimize all the problems that might arise in a construction site and it helps users to complete their esteemed projects in time. This software is based on work of constructor.

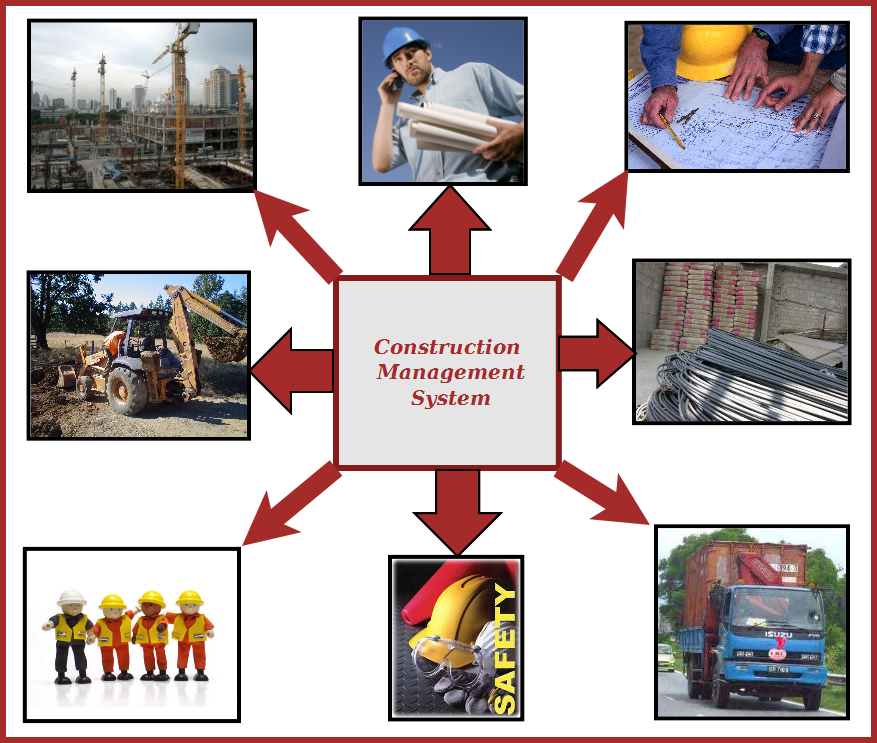
## Objective

Construction Management System is software for managing and maintaining a construction site. A construction site has lots of activity going on parallel. Construction management software will enable the site manager to monitor workers, materials used, work progress, planning & estimation of future work. In simple words, Construction Site Management System is developed to erase all the problems and complexity of a construction site and help user monitor his project(s) and plan for the future projects as well. Go through the main features of the software described below to know more.

It has some advantage that helps the admin head of the construction to enhance his/her knowledge on construction and the effort of their employees and also helps the user to get there answer on their query such as how long their job has done, how much money they need to invest to enjoy the construction work etc. some other advantages are discussed below :

* **Advantage –**

1. **Fully user friendly :** The software is fully user friendly. That is the software is prepared with user friendly scope (like keyboard facility). It is also help full for the user that they can get there entire query result on line.
2. This software provides the security as much as possible.
3. The tools used in the software are easily portable. That is easily installable.
4. It helps the user, admin and manager of the work to make a proper decision. That helps him/her to get good result.
5. It is remove the network criteria. That is the software is not stand alone type software. The connections used in this project will not only work for that machine.
6. It also save a lot of time of admin, employee etc. and the interested users of the construction.



**Figure: Overview of Construction Management System**

# System Analysis

## Identification of Need:

It is time to globalisation and the peoples are trying to create high rising. Multi storage building, shopping mall are rising everywhere. And the related things to create the giantic plant are very difficult to maintain only by paper calculation or dairy maintainence. So I thought to create a application to maintain all the plan, calculation and everything in one place by this.

## Preliminary Investigation:

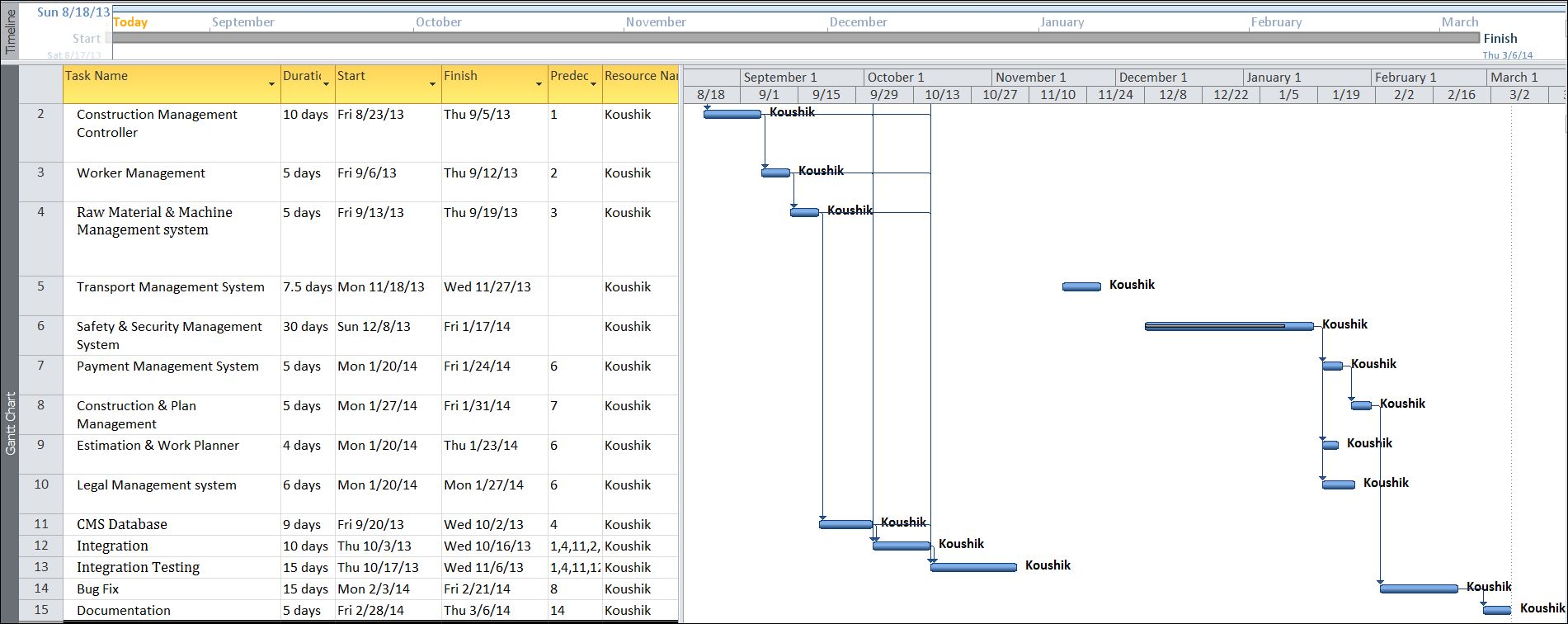
I spoke with many of my friends who use social networking sites regularly and most of them face similar kind of problem. I thought a desktop cum web application could be developed to minimize theses shortcomings of social networking sites. I then started gathering opinion of my friends and seniors among whom some are IT professionals. I gathered all the important points including my own opinion and decided to develop Daily Notebook.

## Feasibility Study:

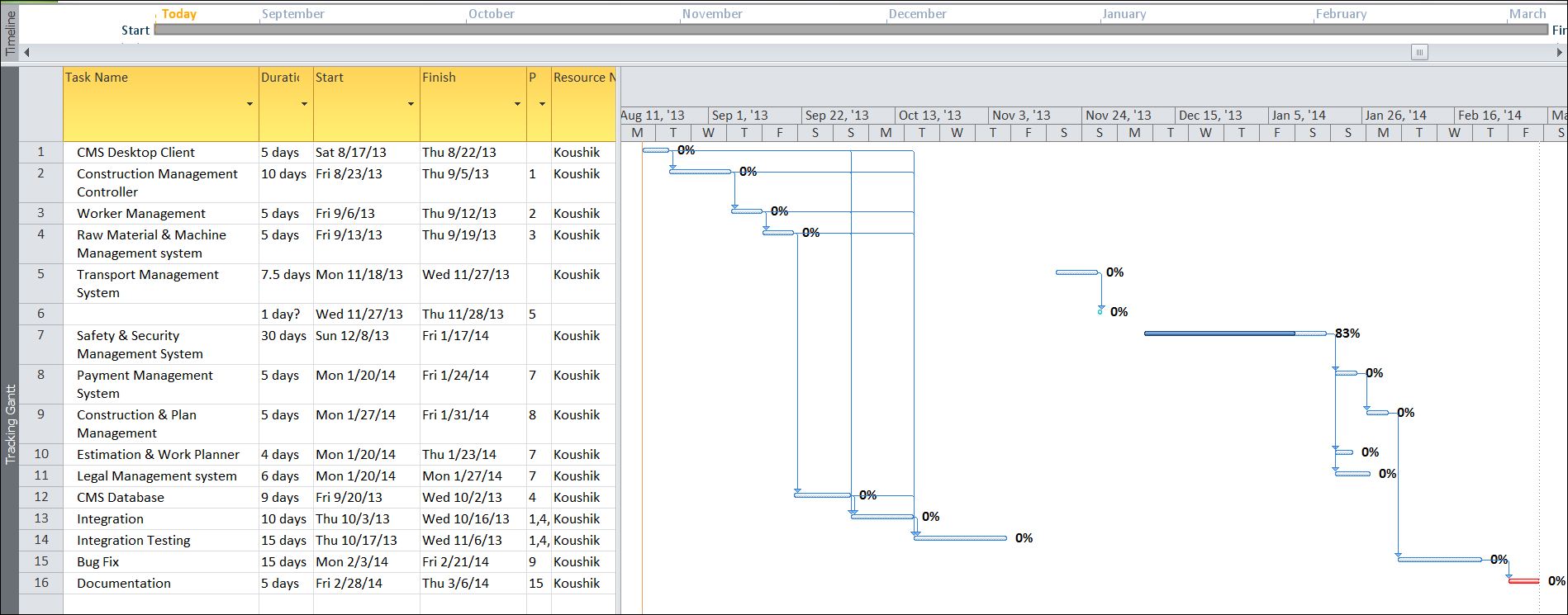
It is an admitted fact that people are becoming more and more addicted to social networking sites day by day. People would love an application that would make their social networking experience more interesting and flawless. I have decided to provide a password manager that to keep track of the id and passwords created in various websites on the internet. So, undoubtedly it is going to be a popular web cum desktop application.

## Project Planning & Scheduling:

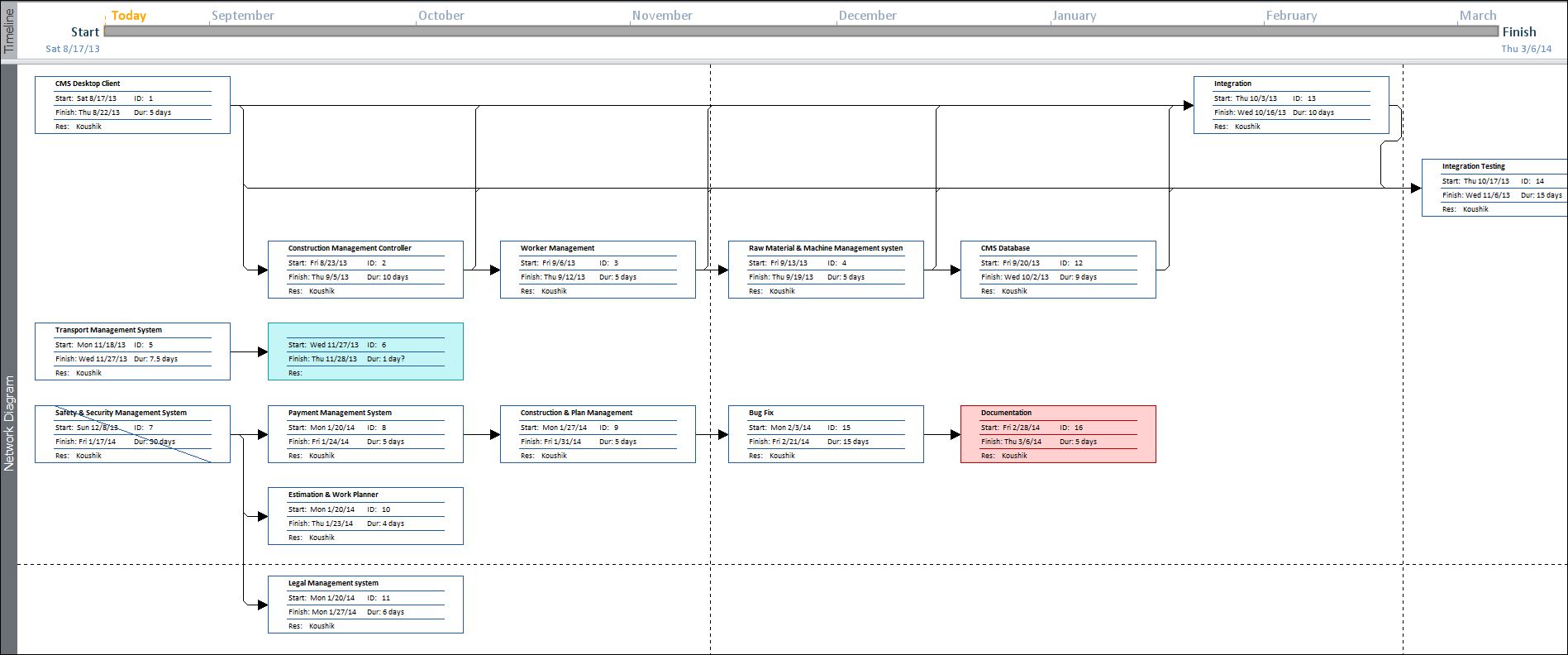
### Gantt chart



### Tracking Gantt



### Pert Chart



## Software requirement specifications (SRS):

## Requirements Specification

### Functional Requirement

#### Add Worker/Labor

Inroduction :

Add a new Worker or labor.

Input :

Relevant worker or labor information like name, address, qualification, contact, experiance etc.

Processing:

Admin will enter data in the **CMS** and create a new worker or labor entry.

Output:

**CMS** will create a new worker or labor for future referance.

#### Add Raw material/Machine

Inroduction :

Add raw material or machine.

Input :

Related raw material or machine information like name, quantity, vandor, etc.

Processing:

Admin will enter data in the **CMS** and create a new raw material or machine entry.

Output:

**CMS** will create a new raw material or machine for construction.

#### Assignment of work to Worker/Labor

Inroduction :

Assign work for Worker or labor.

Input :

Related info of job and worker or labour details etc.

Processing:

Admin will assign job in the **CMS** and create a new job profile entry.

Output:

**CMS** will create a new job index by assigning worker or labor for construction.

#### Apply for security and safety

Inroduction :

Ensure safety and security for Worker and labor.

Input :

Any kind of task assign for the worker and labor.

Processing:

Superviser will check security and safety for the task.

Output:

If the security and safety process is not well the system will generate a reminder in **CMS**.

#### Search Worker/Labor

Inroduction :

Anyone can search for worker or labor.

Input :

He will enter data like worker name, address, etc.

Processing:

CMS will search for the labor or Worker.

Output:

**CMS** will display the search result.

#### Payment for Worker/Labor

Inroduction :

Payment calculation for the Worker or labor.

Input :

Get the job details of the worker or labour etc.

Processing:

Admin will calculate the payment in the **CMS** and send notification to the accounts department.

Output:

**CMS** will create a coupon, by which worker or labor can get payment from account.

#### Get design and plan

Inroduction :

Design and proper planning for the particular task.

Input :

Assign the plan and design job to the planner and designer.

Processing:

Planner and designer will plan and design the proper task schedule.

Output:

**CMS** will get the plan and design.

### Technical Specification

**Front End/ GUI Tools:** Windows Presentation Framework (WPF)

**IDE:** Visual Studio 2010

**Framework:** Microsoft .NET 4.0

**Database:** MySQL

**Database Tool:** MySQL workbench CE

**Operating Systems**: Windows XP, Windows 7

**Cloud Technology**: Google Drive, Google forms

## 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..

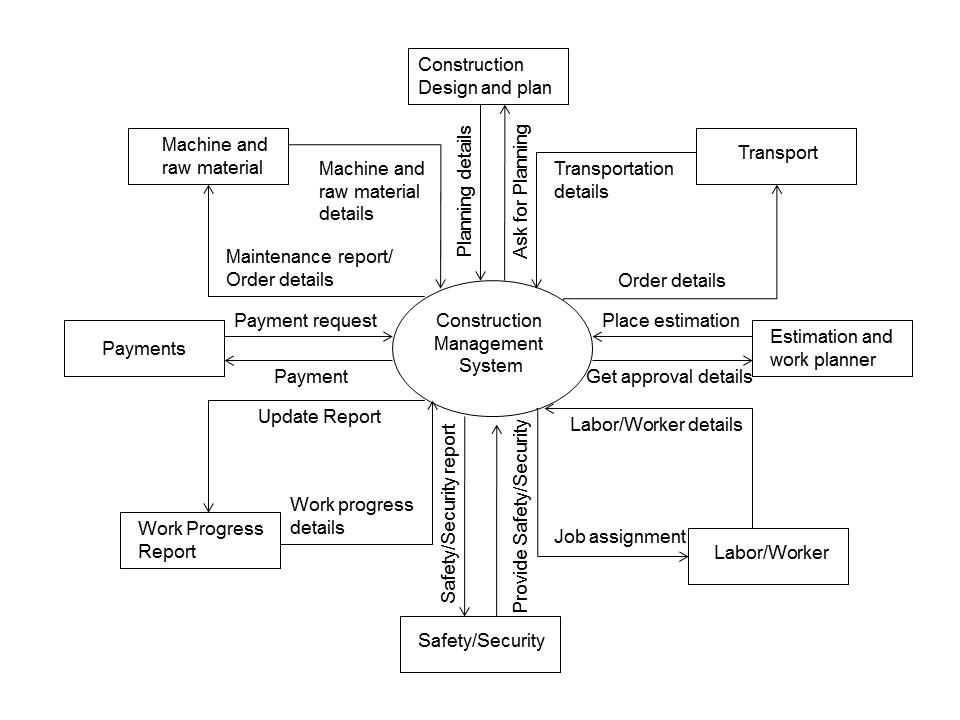


Figure AMDD Through the Agile Development Lifecycle.

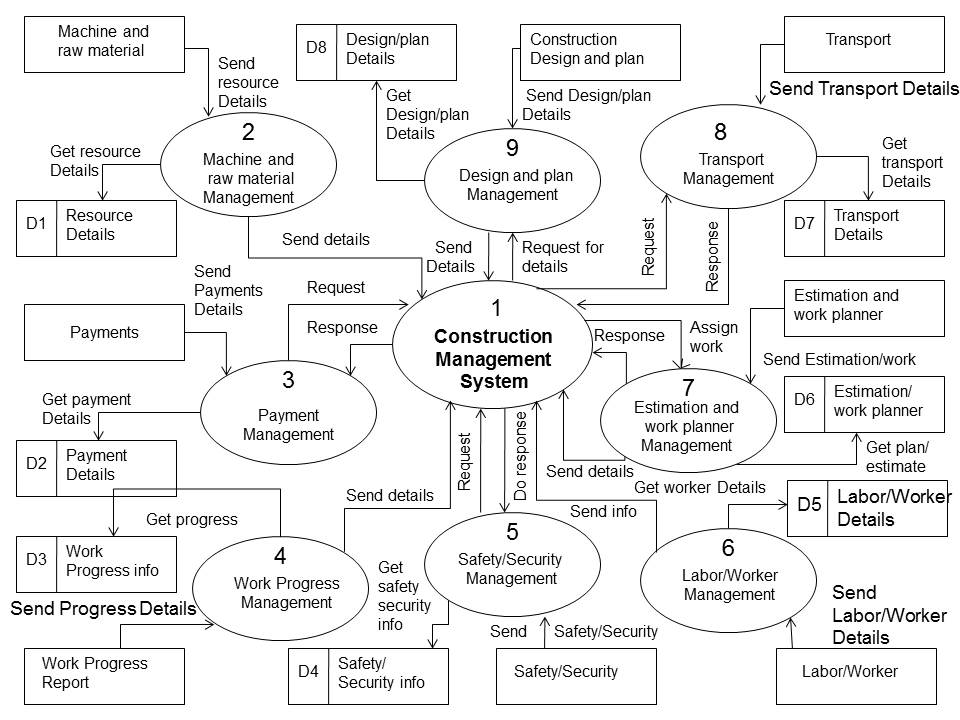
Above Figure depicts how the AMDD activities fit into the various iterations of the agile software development lifecycle.  It's simply another way to show that an agile project begins with some initial modelling and that modelling still occurs in each construction’s iteration.

# Data models

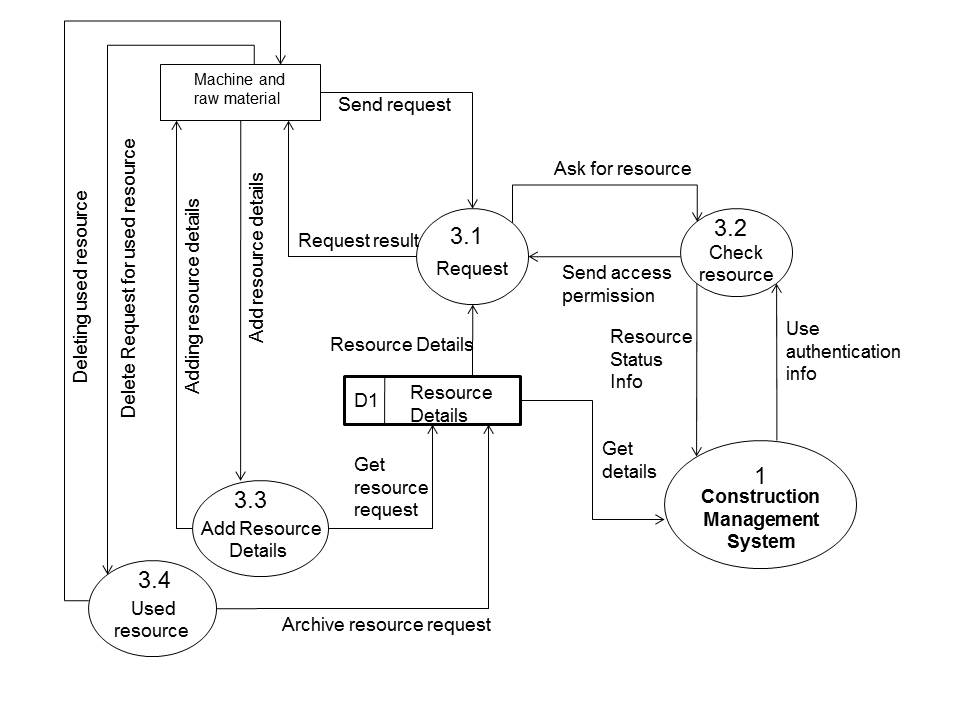
## Context Diagram

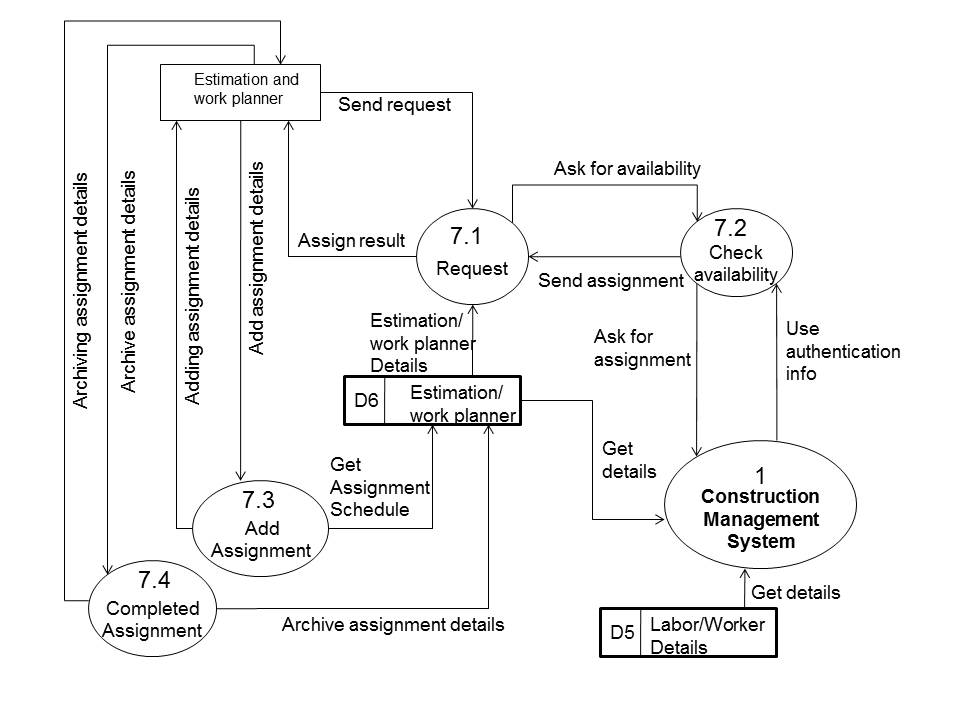


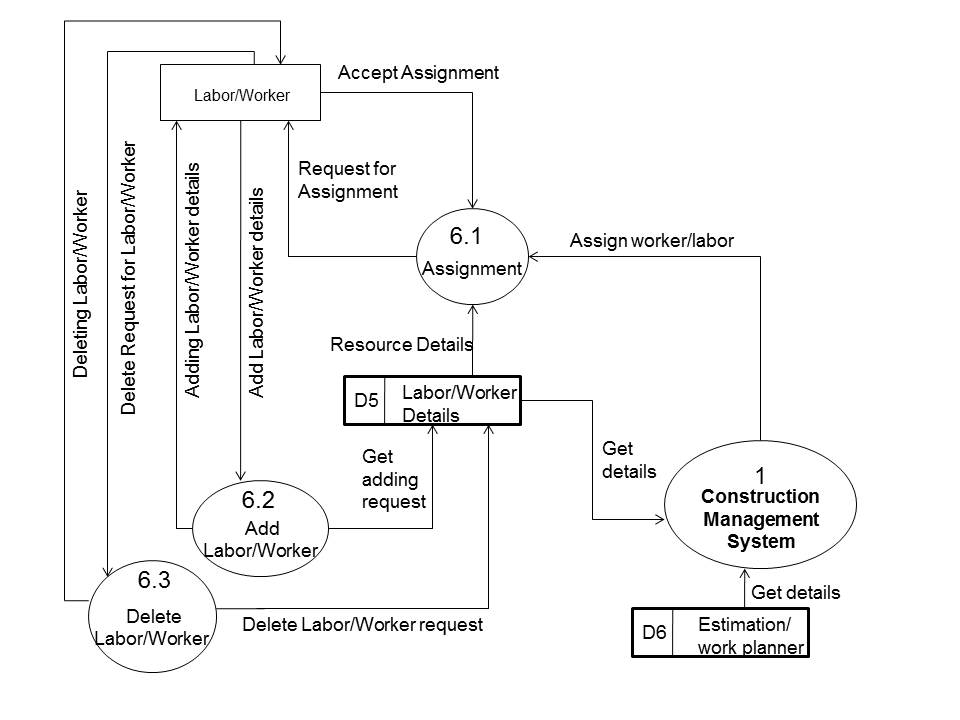
## 0-Level DFD



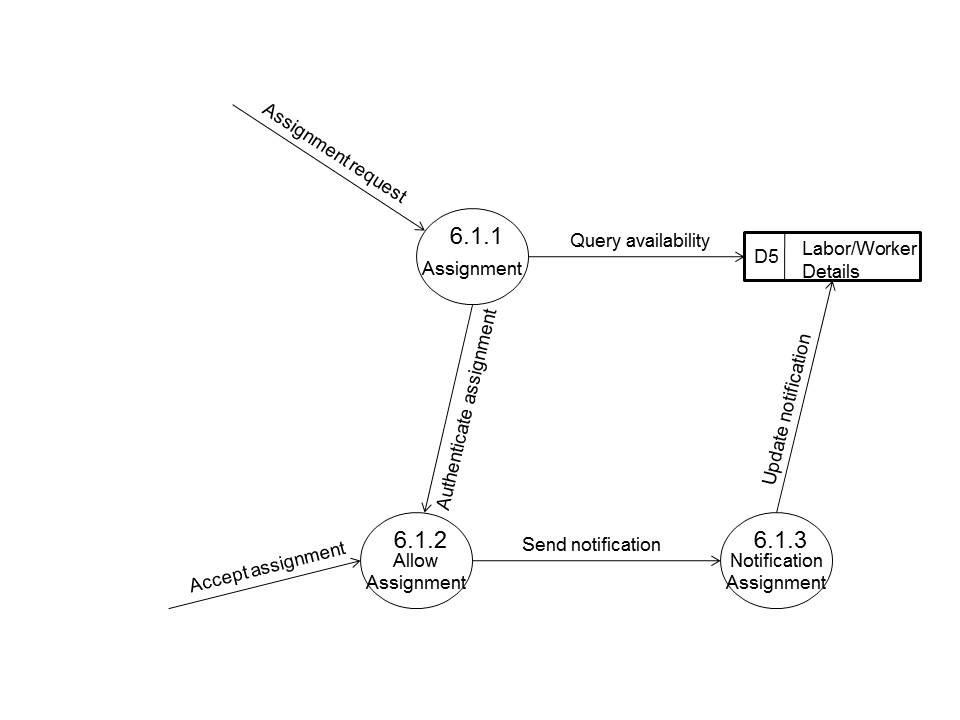
## 1-Level DFD







## 2-Level DFD



## Sequence diagrams

### Interaction Event

Share Event in Social Site

**: Login**

**User**

**: Register**

**User**

**Controller**

**: Update**

**Event**

**: Events**

**Report**

**: View**

**Update Event**

**Report**

**register**

**register**

**modifyEvent**

**addEvent**

**deleteEvent**

**showError**

**showError**

**ShowError**

**updateEvent**

**updateEventReport**

**updateEventsReport**

**showError**

**showError**

**showError**

**viewReport**

**viewEventReport**

**displayEventReport**

**displayReport**

**register**

**register**

**updateEvent**

**syncNotComplete**

**syncComplete**

**syncEvent**

**syncEvent**

**displayEvent**

**syncComplete**

**: Login**

**User**

**: Register**

**User**

**Controller**

**: Update**

**Desktop**

**Event**

**: Update**

**Web**

**Application**

**:Update**

**Desktop**

**Application**

## Entity Relationship Model,

## E-R Diagram:

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

|  |  |
| --- | --- |
| **Entities** | **Attributes** |
| Worker | **User Id**, Name, Address , Contact Number, skillset, Photo ID Num, Photo |
| Construction Management System | **Construction site Id** , Name, Address, Registered no |
| Construction Machine | **Machine Id**, Name, purpose |
| Work Session | **Session Id, worker** Id, Time, Expense amount |
| Engineer | **engineer Id**, Name, address, contact number |
| Design Preference | **Preference Id,** Type, Description |
| Raw Material | **Product id,** stock, name, price |

**Relationship between Entities:**

* Construction Management System has Workers 1 : N
* Construction Management System has Machines 1 : N
* Workers does Work Session 1 : 1
* Construction Management System manages Raw Material  1 : N
* Engineer provides Design Preferences  M : N

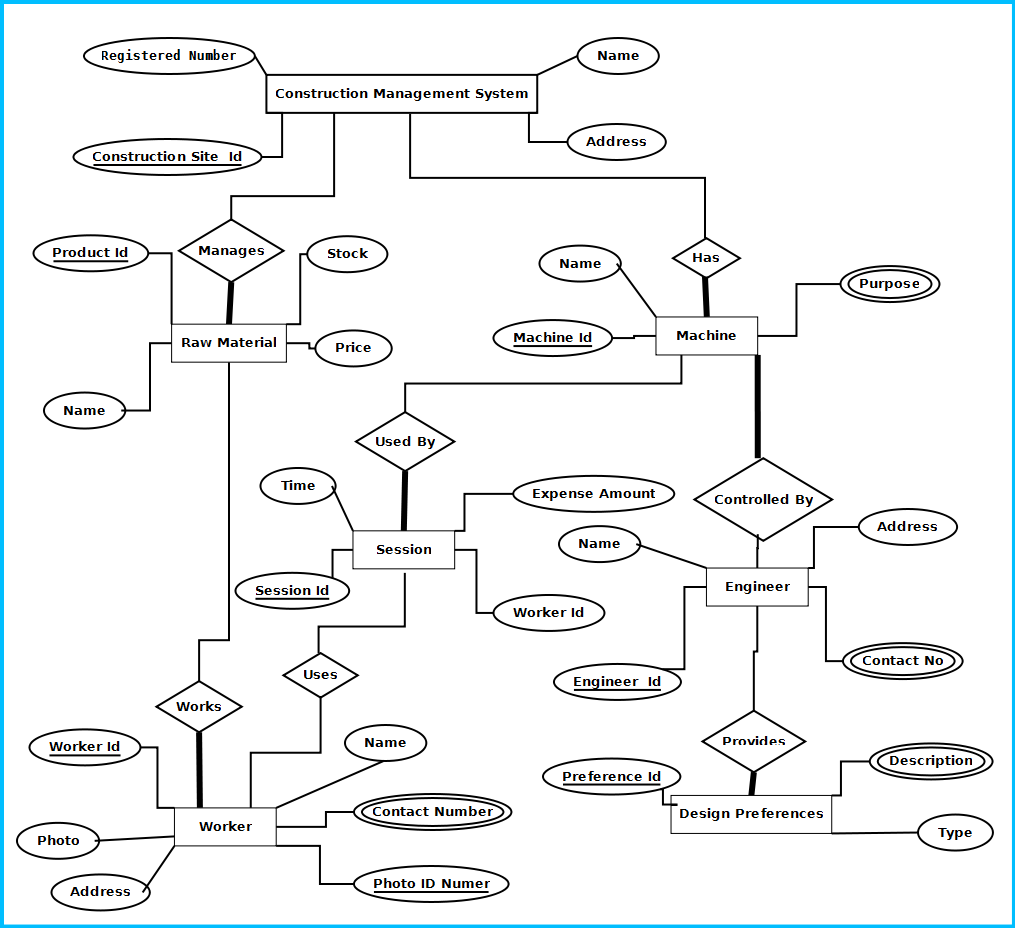
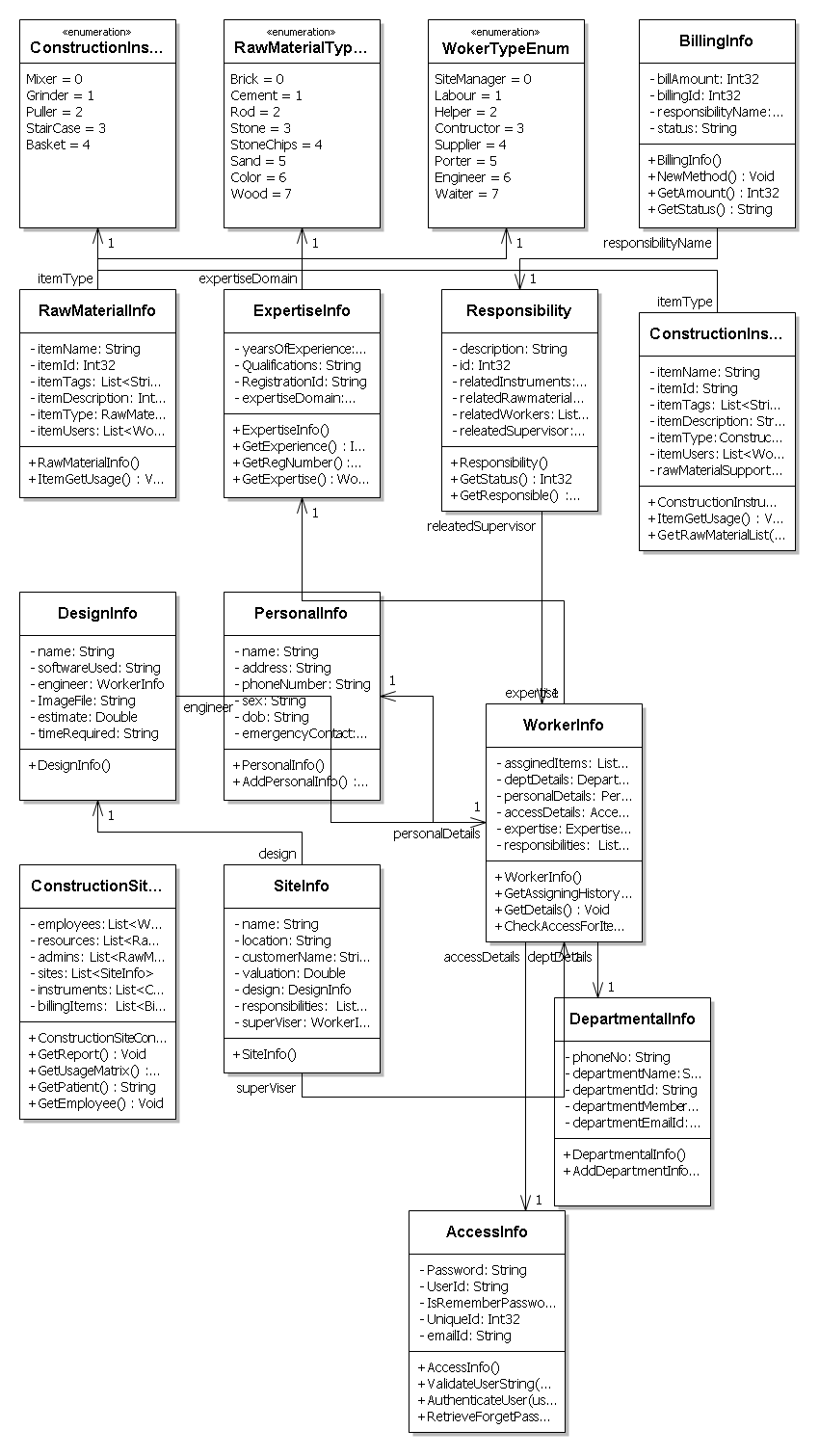
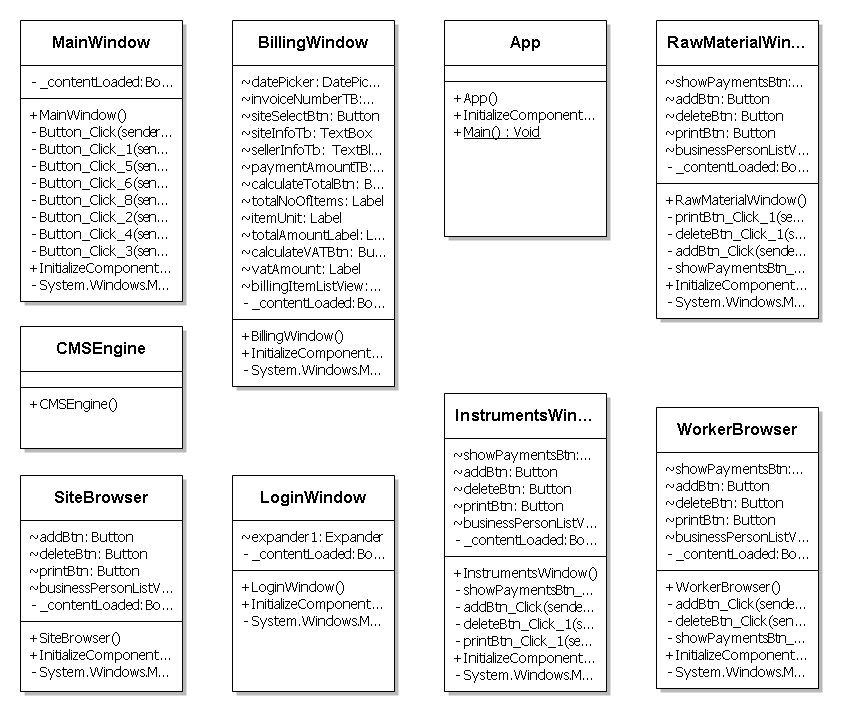


Figure: **E-R Diagram of Construction Management system**

## Class Diagram





## Activity Diagrams



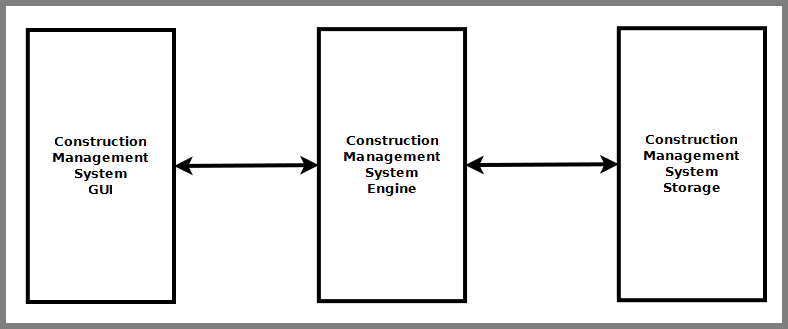
# System Design

## Modularisation details

## Module Description

Construction Management System is divided into three main components. Such as:

* Construction Management System GUI
* Construction Management System Engine
* Construction Management System Database

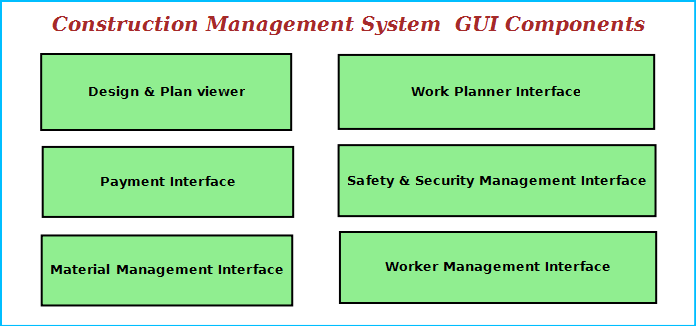


**Figure:** Components of Construction Management System

### Construction Management System GUI

Construction Management System GUI will allow users to view existing construction data, enter new data & modify existing data. It will have a user friendly interface so that user can use the software efficiently. It is divided into six modules. Such as:

* Design & Plan viewer
* Work Planner Interface
* Payment Interface
* Material Management Interface
* Worker Management Interface
* Safety & Security Management Interface



**Figure: Construction Management System GUI Components**

#### Design & Plan Viewer

It will allow to view design & planning file using some external editors & viewer. It will also allow scanning existing plan and view it as a image file. Users can add comments for modification and also generate reports.

#### Work Planner Interface

This interface will allow user to plan and estimate future work. It will have link to worker management interface so that workers can be assigned certain work. It will also enable to add costing and budget data for the work. It will have a option to add legal data associated with a work item and instruction to be followed.

#### Material Management Interface

This interface will allow user to manage the material used for construction.

* + - to search any kind of material of the Construction and view price information and stock details.
    - Enter & modify material availability and usage data.

#### Payment Interface

This interface will allow users to enter payment data associated with various aspects construction work.

* + - Workers payment
    - Raw materials purchase information
    - Future budgets
    - Infrastructure payments.
    - Loan information
    - cash flow and cheque information
    - bank information

#### Worker Management Interface

This module will maintain information about workers.

* + - Construction sites need to maintain the worker personal data, professional experience info.
    - This will allow to track workers attendance and work hours data.
    - This interface will allow managing security measures to taken about the workers. It will have a interface for entering the worker’s security data, photo and scan the photo identity card of the worker.

#### Safety &Security Management Interface

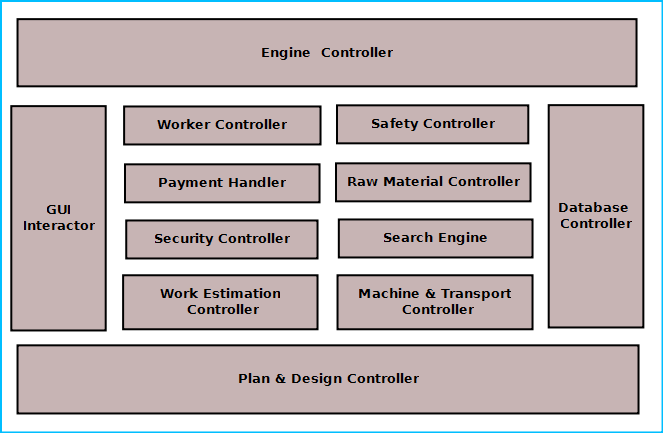
This interface will ensure safety & security of the construction site.

* + - Video Surveillance Interface: This interface will display the video of specific areas of the Construction and save the video for future reference. The site manager can monitor from video surveillance window.
    - First Aid browser: This displays the location first aid boxes and instructions on what to do in case of emergency.
    - Emergency contact interface: This will allow user to call ambulance driver, doctor & hospital in case of emergency.

### Construction Management System Engine

Construction Management System Engine controls the overall system. It provides logical and tactical solutions for managing the whole system. The Construction Management System is divided into 12 divided modular components. Such as:

|  |  |
| --- | --- |
| * Engine controller | * Raw Material Controller |
| * GUI Interactor | * Machine & Transport Controller |
| * Database Controller | * Payment Handler |
| * Plan & Design Controller | * Search Engine |
| * Worker Controller | * Safety Controller |
| * Work Estimation Handler | * Security Controller |

Figure: **Components of Engine**

#### Engine controller:

This controls the overall interaction between all the backend modules. It schedules the priority of the actions in case of overlapping. It helps user to consolidate overall reports and print them from GUI.

#### GUI Interactor:

It interacts with the GUI and polls GUI calls. It exposes APIs and events for GUI to use. GUI Interactor helps Engine to maintain wrapper around the Engine modules so that the GUI can be ported to any other framework without much changes in Engine code.

#### Database Controller:

It controls the database interactions. It forms query to fetch information from the database. It also sends data to be saved in database for future use.

#### Plan & Design Controller:

It allows integration with other designer software to view the design. It processes the design comments and helps generating reports.

#### Worker Controller:

It keeps track of worker information. It has an algorithm for maintaining worker data like

Attendance information

Work hours

Area of expertise

Payment details

Personal data

All these data will BE saved and are available for other modules to use and modify.

#### Security Controller:

It saves information for security measures. It can send it to the authority or police whenever required. It will ensure state of the existing security measures and generate warning if one of the steps is breached.

#### Safety Controller:

It ensures the safety of the construction site. It controls the safety alarms, fire alarms and instructions to be displayed at the hazardous places.

#### Work Estimation Controller:

This will allow users to estimate future work depending on the design,deadline and workers expertise. It will have algorithms to control estimation and assignment of work to the workers.

#### Raw Material Controller:

This module will maintain the information about raw materials. It will derive the figures about available materials, required amount to meet requirements, purchase deadline and allowable stock. It will save the information in database and retrieve them when required.

#### Search Engine:

Search engine enables rapid and efficient searching of data about workers, raw materials and suppliers. Search Engine prepares search indexes depending most accessed data. It improves the efficiency and performance of the software.

#### Payment Handler:

Payment Handler allows generating the report for material purchases. It calculates the salary of a worker. It will get the work & salary information from the database depending on worker data and calculates the payable amount.

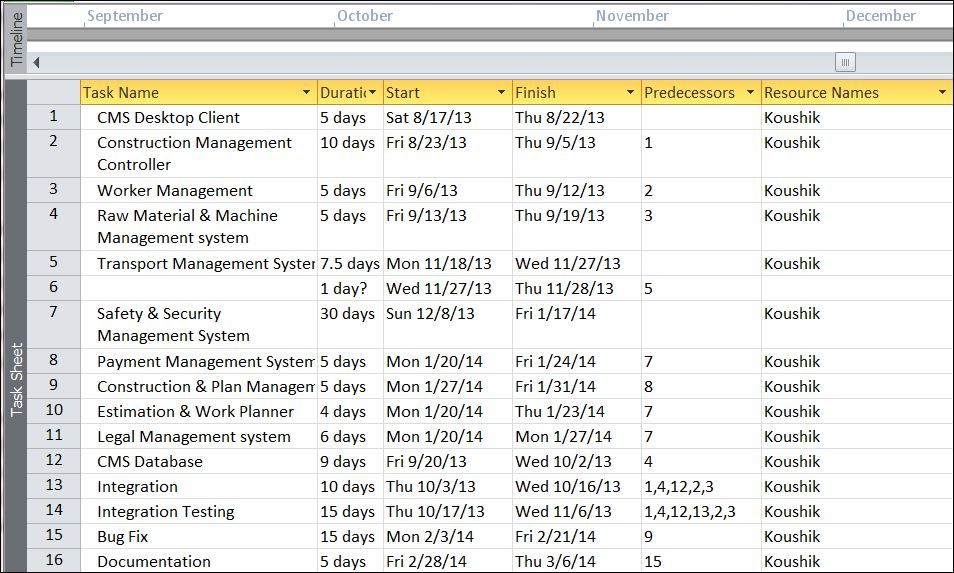
#### Machine & Transport Controller:

This module keeps track machine to be used by the construction workers, machine purchase or hiring information, vendor information. This also handles logistics transport, worker transport.

### Construction Management System Database

Construction Management System will maintain a centralized database for storing information. We will design a RDBMS to manage the database and engine interaction. It will have optimized design and archive older data to save space and increase performance.

## Estimation

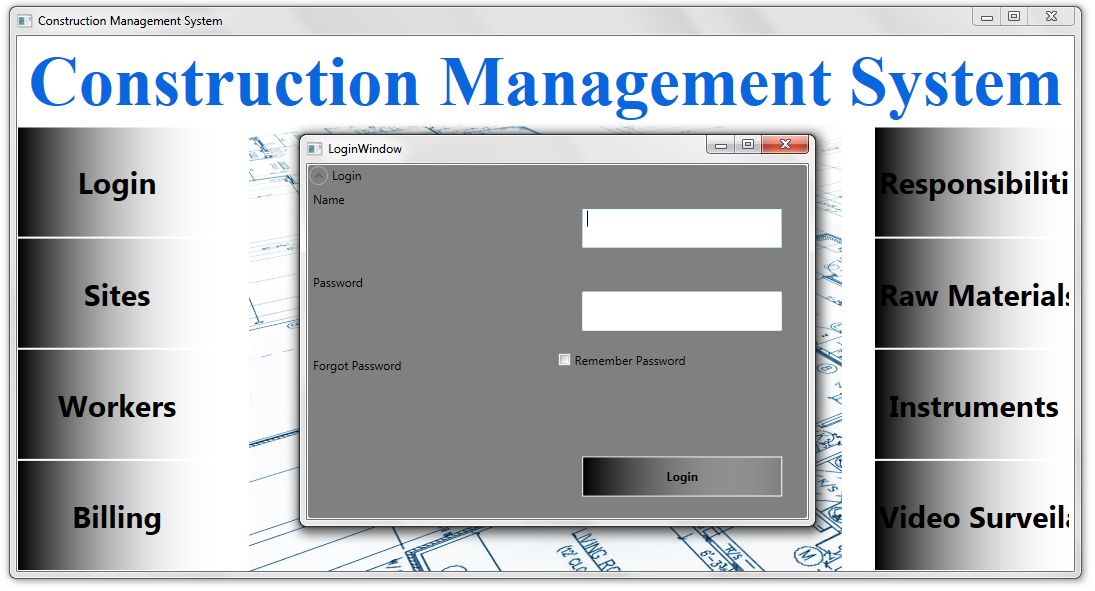


## User Interface Design

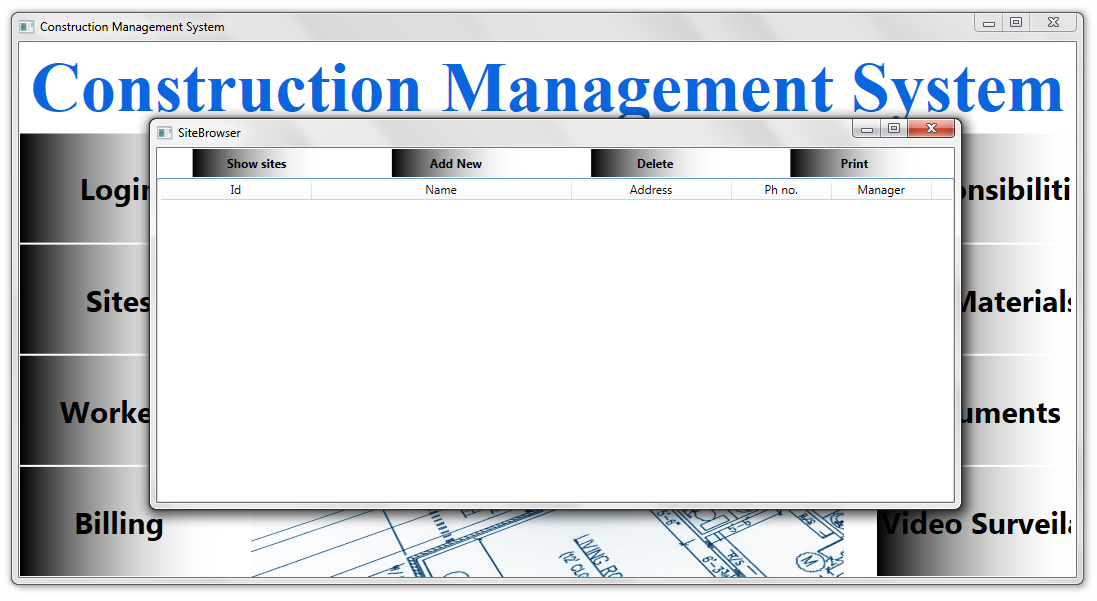
### Main Window



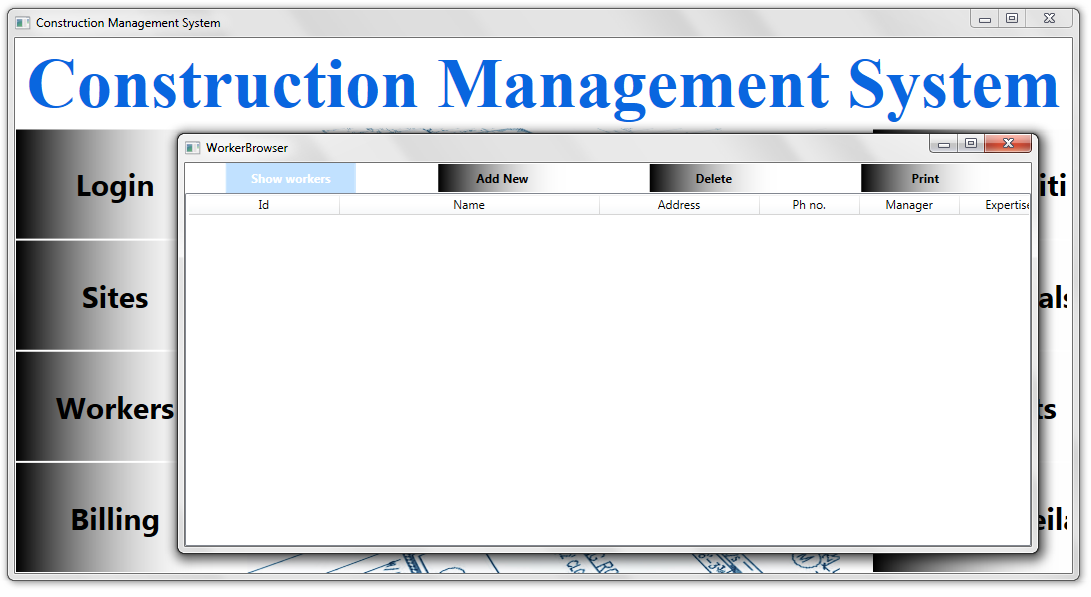
### Login



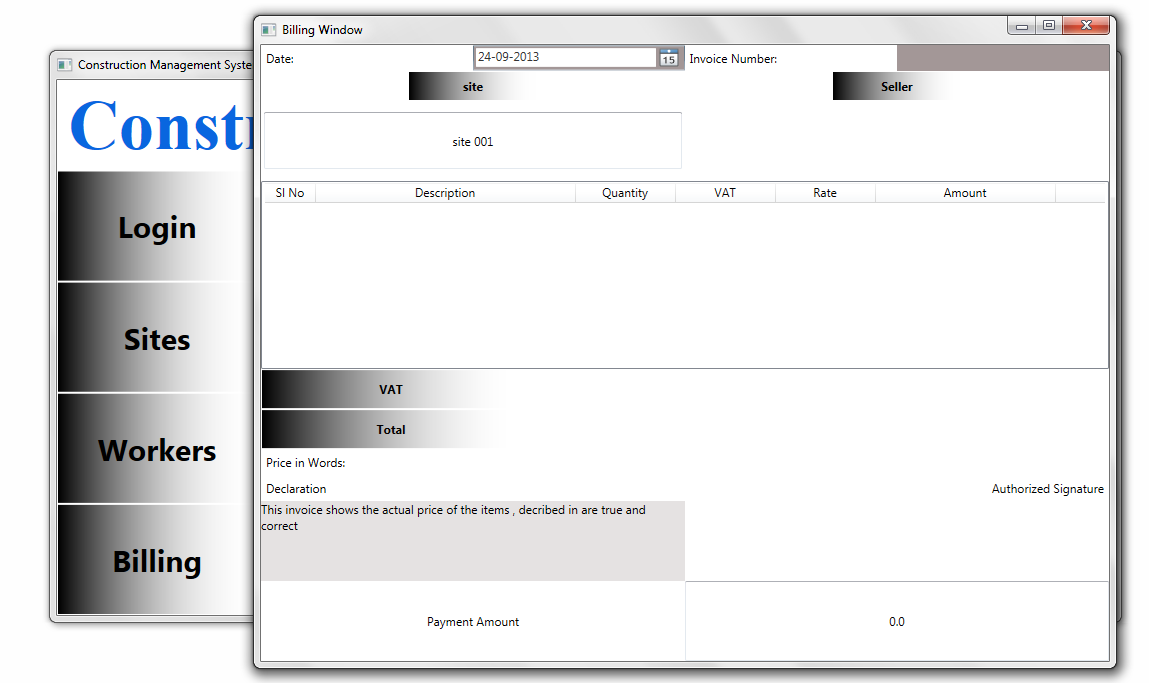
### Sites



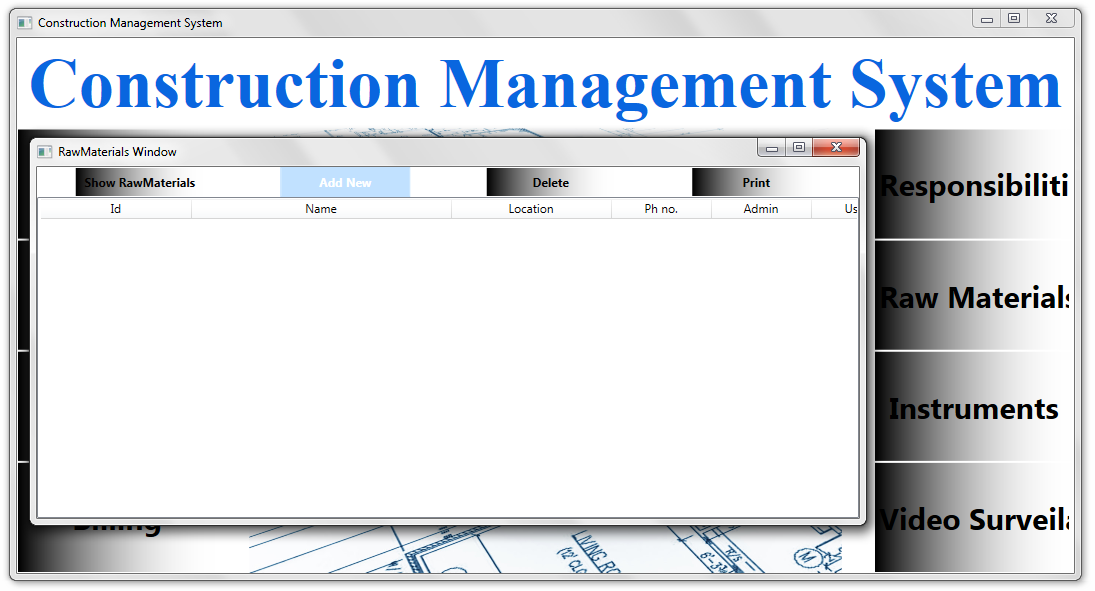
### Workers



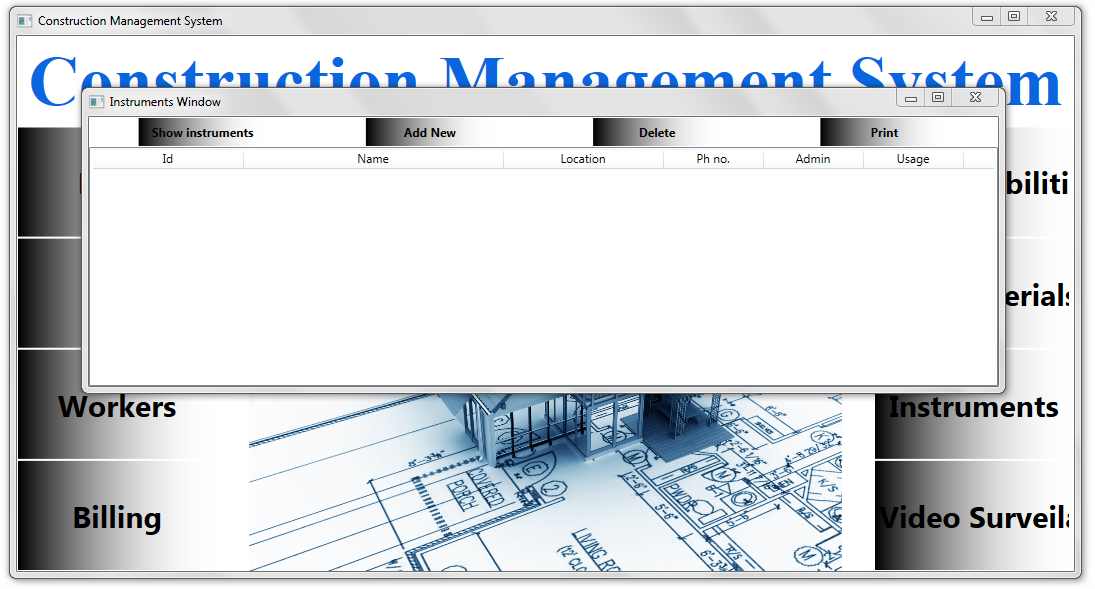
### Billing



### Raw material



### Instruments



## Test Cases (Unit Test Cases and System Test Cases)

### Unit Test Cases

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case Id | Type | Github ID | Subject | Test Name | Test Description | Step Name | Description | Expected Result |
| **CMS**-001 | Manual | f3563be0a9c431104f52839039e86043cf640cf1 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful Login for **CMS** | The purpose of this test is to verify that the User Id and Password of user is valid. | Step1 | Insert wrong User Id and Password. And Click on Login Button. | **CMS** will display error message. And Failed to Login. |
| **CMS**-002 |  |  |  |  |  | Step2 | Insert Wrong User Id and valid Password. And Click on Login Button. | **CMS** will display error message. And Failed to Login. |
| **CMS**-003 |  |  |  |  |  | Step3 | Insert Valid User Id and Wrong Password. And Click on Login Button. | **CMS** will display error message. And Failed to Login. |
| **CMS**-004 |  |  |  |  |  | Step4 | Insert Nothing in User Id and Password fields. And Click on Login Button. | **CMS** will display error message. And Failed to Login. |
| **CMS**-005 |  |  |  |  |  | Step5 | Insert Nothing in User Id and insert Valid Password fields. And Click on Login Button. | **CMS** will display error message. And Failed to Login. |
| **CMS**-006 |  |  |  |  |  | Step6 | Insert Nothing in Password and insert Valid User Id fields. And Click on Login Button. | **CMS** will display error message. And Failed to Login. |
| **CMS**-007 |  |  |  |  |  | Step7 | Insert Nothing in User Id and insert invalid Password fields. And Click on Login Button. | **CMS** will display error message. And Failed to Login. |
| **CMS**-008 |  |  |  |  |  | Step8 | Insert Nothing in Password and insert invalid User Id fields. And Click on Login Button. | **CMS** will display error message. And Failed to Login. |
| **CMS**-009 |  |  |  |  |  | Step9 | Insert valid User Id and Password. And Click on Login Button. | Successfully login. |
| **CMS**-018 | Manual | f0657bbdf47e26ec481fa172b0fa76f9becb2681 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful Show site in Sites of **CMS**. | The purpose of this test is to check the show site is work properly. | Step1 | Click on Sites And click show sites Button. | Successfully display site details. |
| **CMS**-019 | Manual | f0657bbdf47e26ec481fa172b0fa76f9becb2656 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful Add site in Sites of **CMS**. | The purpose of this test is to check the Add site is work properly. | Step1 | Click on Sites And click Add sites Button. | Successfully display Add site Window. |
| **CMS**-020 |  |  |  |  |  | Step2 | Click on submit button without putting information. | Display error message. |
| **CMS**-021 |  |  |  |  |  | Step3 | Insert the details in Field like Name, Address, Phone No., Manager etc. And click submit button. | Successful to Add new site. |
| **CMS**-022 | Manual | f0657bbdf47e26ec481fa172b0fa76f9ffffb2656 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful delete site in Sites of **CMS**. | The purpose of this test is to check the Delete site is work properly. | Step1 | Click on delete button without selecting site. | Display error message. |
| **CMS**-023 |  |  |  |  |  | Step2 | Click on delete button by selecting site. | Deleted successfully. |
| **CMS**-025 | Manual | e22bd0e470f145f3db336ei0e28d474d8f4637d7 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful print the site details. | The purpose of this test is to verify that the print is work properly. | Step1 | Click the print button. | The information of the site is successfully ready for print. |
| **CMS**-026 | Manual | f0657bbdf47e26ec481fa172b0fa76f9becb2640 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful Show workers in workers of **CMS**. | The purpose of this test is to check the Show workers is work properly. | Step1 | Click on workers And click Show workers Button. | Successfully display workers details. |
| **CMS**-027 | Manual | f0657bbdf47e26ec481fa172b0fa76f9becb2627 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful Add workers in Sites of **CMS**. | The purpose of this test is to check the Add workers is workers properly. | Step1 | Click on workers And click Add workers Button. | Successfully display Add workers Window. |
| **CMS**-028 |  |  |  |  |  | Step2 | Click on submit button without putting information. | Display error message. |
| **CMS**-029 |  |  |  |  |  | Step3 | Insert the details in Field like Name, Address, Phone No., Manager, Duty cycle etc. And click submit button. | Successful to Add new worker. |
| **CMS**-030 | Manual | f0657bbdf47e26ec481fa172b0fa76f9ffew2656 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful delete workers in workers of **CMS**. | The purpose of this test is to check the Delete workers is work properly. | Step1 | Click on delete button without selecting workers. | Display error message. |
| **CMS**-031 |  |  |  |  |  | Step2 | Click on delete button by selecting workers. | Deleted successfully. |
| **CMS**-032 | Manual | e22bd0e470f145f3db336ei0e28d474d8f4437d7 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful print the workers details. | The purpose of this test is to verify that the print is work properly. | Step1 | Click the print button. | The information of the workers is successfully ready for print. |
| **CMS**-033 |  |  |  |  |  |  |  |  |
| **CMS**-034 |  |  |  |  |  |  |  |  |
| **CMS**-035 |  |  |  |  |  |  |  |  |
| **CMS**-036 |  |  |  |  |  |  |  |  |
| **CMS**-037 |  |  |  |  |  |  |  |  |
| **CMS**-038 |  |  |  |  |  |  |  |  |
| **CMS**-039 |  |  |  |  |  |  |  |  |
| **CMS**-040 |  |  |  |  |  |  |  |  |
| **CMS**-041 |  |  |  |  |  |  |  |  |
| **CMS**-042 |  |  |  |  |  |  |  |  |
| **CMS**-043 |  |  |  |  |  |  |  |  |
| **CMS**-044 |  |  |  |  |  |  |  |  |
| **CMS**-045 |  |  |  |  |  |  |  |  |
| **CMS**-046 |  |  |  |  |  |  |  |  |
| **CMS**-047 |  |  |  |  |  |  |  |  |
| **CMS**-048 |  |  |  |  |  |  |  |  |
| **CMS**-049 |  |  |  |  |  |  |  |  |
| **CMS**-050 |  |  |  |  |  |  |  |  |
| **CMS**-051 |  |  |  |  |  |  |  |  |
| **CMS**-052 |  |  |  |  |  |  |  |  |
| **CMS**-053 |  |  |  |  |  |  |  |  |
| **CMS**-054 |  |  |  |  |  |  |  |  |
| **CMS**-055 |  |  |  |  |  |  |  |  |
| **CMS**-056 |  |  |  |  |  |  |  |  |
| **CMS**-057 |  |  |  |  |  |  |  |  |
| **CMS**-058 |  |  |  |  |  |  |  |  |
| **CMS**-059 |  |  |  |  |  |  |  |  |
| **CMS**-060 |  |  |  |  |  |  |  |  |
| **CMS**-061 |  |  |  |  |  |  |  |  |
| **CMS**-062 |  |  |  |  |  |  |  |  |
| **CMS**-063 |  |  |  |  |  |  |  |  |
| **CMS**-064 |  |  |  |  |  |  |  |  |
| **CMS**-065 |  |  |  |  |  |  |  |  |
| **CMS**-066 |  |  |  |  |  |  |  |  |

### System Test Cases

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case Id | Type | Github ID | Subject | Test Name | Test Description | Step Name | Description | Expected Result |
| **CMS**-067 | Manual | f3563be0a9c431104f52839039e86043cf640cf1 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Log in. | It is to check that Login works properly. | Step1 | Click on Login button after inserting invalid User id and password from **CMS**. | Login failed to **CMS**. And can’t able to use the feature. |
| **CMS**-068 |  |  |  |  |  | Step2 | Click on Login button after inserting valid User id and password from **CMS**. | Successfully Login to **CMS**. And can able to use the feature. |
| **CMS**-069 | Manual | d01197ee4cd3bee9245874b5937ba740019fd131 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful Registration for New **CMS** User. | The purpose of this test is to verify that the all new connection could be creating new Account By Registration. | Step1 | Click on Registration link. | New Account creation area is opened. |
| **CMS**-070 |  |  |  |  |  | Step2 | Click on Registration button after inserting invalid information from **CMS**. | Registration failed to **CMS**. And can’t able to use the feature. |
| **CMS**-071 |  |  |  |  |  | Step3 | Click on Registration button after inserting valid information from **CMS**. | Registration Successfully done to **CMS**. |
| **CMS**-072 |  |  |  |  |  | Step4 | Click on Login button after inserting newly created valid User id and password from **CMS**. | Successfully Login to by new User Id And password **CMS**. And can able to use the feature. |
| **CMS**-073 | Manual | f0657bbdf47e26ec481fa172b0fa76f9becb2681 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check Successful Event Manage in **CMS**. | The purpose of this test is to check event insertion process. | Step1 | Login to **CMS** with valid User Id And Password. And compose event select the social site. Click on Save Note button. | Successfully Inserted event. Display the event in shared social site. |
| **CMS**-074 |  |  |  |  |  | Step2 | Check various view like monthly view, view all, weekly view, daily view to Show saved event. | Successfully Display events in monthly view, view all, daily view, weekly view. |
| **CMS**-075 |  |  |  |  |  | Step3 | Select an event from monthly view or view all or daily view or weekly view .And click on Delete Event. | Successfully Deleted events from monthly view, view all, daily view, weekly view. |
| **CMS**-076 |  |  |  |  |  | Step4 | Select an event from monthly view or view all or daily view or weekly view .And click on Rewrite Event. | Event ready in Edit field and after editing click on save note button to update the event. And Successfully Display edited event in monthly view, view all, daily view, weekly view. |
| **CMS**-077 | Manual | 47fb570f63ffec837a49e235f629c49cc55a70f0 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check social site status and update event. | It is to check that we can get and send post or tweet to a particular social site. | Step1 | Click Facebook under Social Site tab. And enter valid Login id And Password to connect and create event and click on post button. | Successfully connected to Facebook and can post in Facebook wall. |
| **CMS**-078 |  |  |  |  |  | Step2 | Select a post and click on delete button to delete post from Facebook wall. | Post Successfully deleted. |
| **CMS**-079 |  |  |  |  |  | Step3 | Click Twitter under Social Site tab. And enter valid Login id And Password to connect and create event and click on Tweet button. | Successfully connected to Twitter and can Tweet in Twitter wall. |
| **CMS**-080 |  |  |  |  |  | Step4 | Select a Tweet and click on delete button to delete post from Twitter wall. | Tweet Successfully deleted. |
| **CMS**-081 | Manual | b8a65899fa408a4d99a8ee8cabbe2ac0b54226b0 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check social Google map. | It is to check that we can get the location from Google map site. | Step1 | Without any location in location field and click on Load Map button. | Load the entire Google map. |
| **CMS**-082 |  |  |  |  |  | Step2 | Enter location and click on Load Map button. | Successfully find the particular location. |
| **CMS**-083 | Manual | 6c53de9297011864bd6c07bdcce83415a76e8bd4 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check everything about contact information. | It is to check that contact works properly. | Step1 | Click on Contact tab. Compose contact information and click on Save Button. | Contact display all saved data. And created contact successfully. |
| **CMS**-084 |  |  |  |  |  | Step2 | Select Contact from list view area .And click on Delete Event. | Successfully Deleted Contact from list view area. |
| **CMS**-085 |  |  |  |  |  | Step3 | 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. |
| **CMS**-086 | Manual | 90330b92328d862892fc77436539081cc2b7f70d | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check everything about Password Manager information. | It is to check that Password Manager works properly. | Step1 | Click on Password tab. Compose Password information and click on Save Button. | Password display all saved data. And created Password successfully. |
| **CMS**-087 |  |  |  |  |  | Step2 | Select Password from list view area .And click on Delete Password. | Successfully Deleted Password from list view area. |
| **CMS**-089 |  |  |  |  |  | Step3 | Select Password from list view area. And click on Edit button. | Password ready in Edit field and after editing click on Save button to update the Password. And Successfully Display edited Password in list view area. |
| **CMS**-090 | Manual | 90330b92328d862892fc77436539081cc2b7f70d | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check everything about Task information. | It is to check that Task works properly. | Step1 | Click on Task tab Under Extra Tab. Compose Task information and click on Save Button. | Task display all saved data. And created Task successfully. |
| **CMS**-091 |  |  |  |  |  | Step2 | Select Task from list view area .And click on Delete Task. | Successfully Deleted Task from list view area. |
| **CMS**-092 |  |  |  |  |  | Step3 | Select Task from list view area. And click on Edit button. | Password ready in Edit field with status option and after editing click on Save button to update the Task. And Successfully Display edited Task with status in list view area. |
| **CMS**-093 | Manual | f3563be0a9c431104f52839039e86043cf640cf1 | E:\DEVELOPERS\_ZONE\GitHub\ConstructionManagementSystem\code | Check logout. | It is to check that Logout works properly. | Step1 | Click on logout button to logout from **CMS** section. | Successfully Logout from **CMS**. And can’t able to use the feature any more. |

# Coding

## Complete Project Coding

### CMS GUI Design Coding: CMSGUI

|  |
| --- |
| MainWindow.xaml |
| <Window x:Class="**CMS**GUI.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:sys="clr-namespace:System;assembly=mscorlib"  xmlns:ViewEvent="clr-namespace:**CMS**GUI"  DataContext="{Binding RelativeSource={RelativeSource Self}}"  Title="**CMS**"  MinHeight="650" MinWidth="1250" AllowsTransparency="True" ResizeMode="CanResizeWithGrip" Background="Transparent"  WindowStyle="None" WindowStartupLocation="CenterScreen" Icon="/**CMS**GUI;component/Images/monotone\_pen\_write.png" Loaded="Window\_Loaded" >  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**CMS**Styles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <Border Margin="10">  <Border.Effect>  <DropShadowEffect Color="Black"  Direction="270"  BlurRadius="15"  ShadowDepth="3" />  </Border.Effect>  <DockPanel LastChildFill="True" Margin="2" Background="White">  <DockPanel DockPanel.Dock="Top" LastChildFill="True">  <UniformGrid DockPanel.Dock="Right" HorizontalAlignment="Right" Rows="1">  <Button Width="30" Height="30" Name="minimizeBtn" ToolTip="minimize" Style="{StaticResource TitlebarBTN}" Click="minimizeBtn\_Click">\_</Button>  <Button Width="30" Height="30" Name="closeBtn" ToolTip="close" Style="{StaticResource TitlebarBTN}" Click="closeBtn\_Click">X</Button>  </UniformGrid>  <UniformGrid DockPanel.Dock="Right" HorizontalAlignment="Right">  <Button Margin="0" Width="70" Height="30" Name="logout**CMS**Btn" IsEnabled="False" Style="{StaticResource TitlebarBTN}" Click="logout**CMS**Btn\_Click"></Button>  </UniformGrid>  <Label DockPanel.Dock="Left" Name="hntLginLbl" Style="{StaticResource LblStyle}"> Login first to use Daily Note Book with Social Networking Updater </Label>  </DockPanel>  <UniformGrid DockPanel.Dock="Bottom" Rows="1" Background="WhiteSmoke">  <TextBlock Height="25" Padding="7,1,0,0" VerticalAlignment="Center" Text="{Binding Source={x:Static sys:DateTime.Today}, StringFormat='{}{0:dddd, MMMM dd, yyyy}'}" Foreground="Black" ></TextBlock>  <TextBlock Height="25" Padding="7,1,10,0" HorizontalAlignment="Right">Copyright 2013, Anirban Nandy</TextBlock>  </UniformGrid>  <DockPanel DockPanel.Dock="Top" LastChildFill="True">  <DockPanel >  <Expander Name="loginExpndr" DockPanel.Dock="Top" IsExpanded="True" Header="Login in **CMS**" Padding="2,0,0,0">  <UniformGrid Columns="1" >  <DockPanel Height="149" Width="186">  <DockPanel LastChildFill="True" DockPanel.Dock="Top">  <Label DockPanel.Dock="left" Width="70" Style="{StaticResource LblStyle}">User ID :</Label>  <TextBox x:Name="**CMS**UserIDTB" DockPanel.Dock="Right" Margin="2" ToolTip="enter user id" Tag="enter user name" Style="{StaticResource commonTBtyle}"/>  </DockPanel>  <DockPanel LastChildFill="True" DockPanel.Dock="Top">  <Label DockPanel.Dock="left" Width="70" Style="{StaticResource LblStyle}">Password :</Label>  <PasswordBox Name="**CMS**passPB" Margin="2"></PasswordBox>  </DockPanel>  <DockPanel LastChildFill="True" DockPanel.Dock="Top">  <TextBlock Text="New User !! " DockPanel.Dock="left" VerticalAlignment="Center" >  <Hyperlink Click="createAcHp\_Click"> Register</Hyperlink>  </TextBlock>  <Button Name="**CMS**LoginBtn" ToolTip="login" Style="{StaticResource PlainBtnStyle}" Click="**CMS**LoginBtn\_Click">Login</Button>  </DockPanel>  <DockPanel DockPanel.Dock="Bottom">  <UniformGrid Rows="2">  <Label Name="errorMsgLbl" Foreground="Red"></Label>  <Label Name="hintsMsgLbl" Foreground="Red"></Label>  </UniformGrid>  </DockPanel>  </DockPanel>  </UniformGrid>  </Expander>  <Expander Name="newAcExpndr" DockPanel.Dock="Top" IsExpanded="False" Visibility="Collapsed" Header="Create New Account" Padding="2,0,0,0">  <UniformGrid Columns="1" >  <DockPanel Height="175" Width="186">  <DockPanel LastChildFill="True" DockPanel.Dock="Top">  <Label DockPanel.Dock="left" Width="70" Style="{StaticResource LblStyle}">User ID :</Label>  <TextBox x:Name="**CMS**NewUserIDTB" DockPanel.Dock="Right" Margin="2" ToolTip="enter user id" Tag="enter user name" Style="{StaticResource commonTBtyle}"/>  </DockPanel>  <DockPanel LastChildFill="True" DockPanel.Dock="Top">  <Label DockPanel.Dock="left" Width="70" Style="{StaticResource LblStyle}">Password :</Label>  <PasswordBox Name="**CMS**NewPassPB" Margin="2"></PasswordBox>  </DockPanel>  <DockPanel LastChildFill="True" DockPanel.Dock="Top">  <Label DockPanel.Dock="left" Width="70" Style="{StaticResource LblStyle}">Re Type :</Label>  <PasswordBox Name="**CMS**NewRepassPB" Margin="2"></PasswordBox>  </DockPanel>  <DockPanel LastChildFill="True" DockPanel.Dock="Top"></DockPanel>  <DockPanel LastChildFill="True" DockPanel.Dock="Top">  <Label DockPanel.Dock="left" Width="70" Style="{StaticResource LblStyle}">Hints :</Label>  <TextBox x:Name="**CMS**HintsTB" Margin="2" ToolTip="enter hints" Tag="enter hints" Style="{StaticResource commonTBtyle}"></TextBox>  </DockPanel>  <DockPanel DockPanel.Dock="Top">  <TextBlock Text="Already User !! " DockPanel.Dock="left" VerticalAlignment="Center" >  <Hyperlink Click="goLogin\_Click">Login</Hyperlink>  </TextBlock>  <Button Name="dNBSSubmitBtn" ToolTip="submit" Style="{StaticResource PlainBtnStyle}" Click="dNBSSubmitBtn\_Click">Submit</Button>  </DockPanel>  <DockPanel DockPanel.Dock="Bottom">  <Label Name="createErrorMsgLbl" Foreground="Red"></Label>  </DockPanel>  </DockPanel>  </UniformGrid>  </Expander>  <Expander Name="mainLeftExpndr" IsEnabled="False" IsExpanded="False" Width="200" DockPanel.Dock="Top" Padding="2,0,0,0" ExpandDirection="Right" Header="H&#x0a; I&#x0a;D&#x0a;E&#x0a;/&#x0a;U&#x0a;N&#x0a;H&#x0a; I&#x0a;D&#x0a;E">  <UniformGrid Columns="1">  <DockPanel>  <Calendar DockPanel.Dock="Top" BorderBrush="#FFE9E0E0">    </Calendar>  <TextBox Height="100" TextWrapping="Wrap" AcceptsReturn="True" VerticalScrollBarVisibility="Visible" SpellCheck.IsEnabled="True">  <TextBox.Style>  <Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">  <Style.Resources>  <VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Top" Stretch="None">  <VisualBrush.Visual>  <Label Content="what's on your mind?" Foreground="LightGray" />  </VisualBrush.Visual>  </VisualBrush>  </Style.Resources>  <Style.Triggers>  <Trigger Property="Text" Value="{x:Static sys:String.Empty}">  <Setter Property="Background" Value="{StaticResource CueBannerBrush}" />  </Trigger>  <Trigger Property="Text" Value="{x:Null}">  <Setter Property="Background" Value="{StaticResource CueBannerBrush}" />  </Trigger>  <Trigger Property="IsKeyboardFocused" Value="True">  <Setter Property="Background" Value="White" />  </Trigger>  </Style.Triggers>  </Style>  </TextBox.Style>  </TextBox>  </DockPanel>  <DockPanel LastChildFill="True">  <UniformGrid DockPanel.Dock="Bottom" Rows="1">  <TextBox DockPanel.Dock="Bottom" Height="40" Width="85" TextWrapping="Wrap" AcceptsReturn="True" VerticalScrollBarVisibility="Visible" SpellCheck.IsEnabled="True">  <TextBox.Style>  <Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">  <Style.Resources>  <VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Top" Stretch="None">  <VisualBrush.Visual>  <Label Content="keyword" Foreground="LightGray" />  </VisualBrush.Visual>  </VisualBrush>  </Style.Resources>  <Style.Triggers>  <Trigger Property="Text" Value="{x:Static sys:String.Empty}">  <Setter Property="Background" Value="{StaticResource CueBannerBrush}" />  </Trigger>  <Trigger Property="Text" Value="{x:Null}">  <Setter Property="Background" Value="{StaticResource CueBannerBrush}" />  </Trigger>  <Trigger Property="IsKeyboardFocused" Value="True">  <Setter Property="Background" Value="White" />  </Trigger>  </Style.Triggers>  </Style>  </TextBox.Style>  </TextBox>  <Button Name="goBtn" ToolTip="go" Content="Go" Style="{StaticResource PlainBtnStyle}">  <Button.LayoutTransform>  <RotateTransform Angle="45" />  </Button.LayoutTransform>  </Button>  </UniformGrid>  <Button Name="updateBtn" DockPanel.Dock="Right" ToolTip="update" Height="150" Style="{StaticResource PlainBtnStyle}">Update</Button>  <UniformGrid Columns="1">  <CheckBox Name="facebookCheckBox" VerticalAlignment="Center" Style="{StaticResource CheckBtyle}">Facebook</CheckBox>  <CheckBox Name="twitterCheckBox" VerticalAlignment="Center" Style="{StaticResource CheckBtyle}">Twitter</CheckBox>  <CheckBox Name="linkedInCheckBox" VerticalAlignment="Center" Style="{StaticResource CheckBtyle}">LinkedIn</CheckBox>  <CheckBox Name="googlePlusCheckBox" VerticalAlignment="Center" Style="{StaticResource CheckBtyle}">Google+</CheckBox>  <CheckBox Name="foursquareCheckBox" VerticalAlignment="Center" Style="{StaticResource CheckBtyle}">Foursquare</CheckBox>  <CheckBox Name="googlecalendarCheckBox" VerticalAlignment="Center" Style="{StaticResource CheckBtyle}">  <TextBlock TextWrapping="Wrap" Text="Google Calendar"/>  </CheckBox>  <CheckBox Name="myNoteBookCheckBox" VerticalAlignment="Center" Style="{StaticResource CheckBtyle}" IsChecked="True">  <TextBlock TextWrapping="Wrap" Text="My Note Book"/>  </CheckBox>  </UniformGrid>  </DockPanel>  </UniformGrid>  </Expander>  </DockPanel>  <TabControl Name="mainTabControl" IsEnabled="False" Margin="3" BorderBrush="#FFE9E0E0">  <TabItem Name="notesTabItem" Style="{StaticResource TItemStyle}" Header="Note Book" Background="red">    <DockPanel LastChildFill="True">  <DockPanel DockPanel.Dock="Top" >    <DockPanel DockPanel.Dock="Bottom" LastChildFill="True">    <UniformGrid Rows="1" >  <DatePicker ToolTip="today" Margin="10" Name="noteDateDP" SelectedDate="{x:Static sys:DateTime.Now}" Style="{StaticResource DatepkrStyle}"></DatePicker>  <Label Name="noteSuccessMsgLvl" Foreground="Red" Style="{StaticResource LblStyle}"></Label>  <Button Name="clearFieldsBtn" Content="Clear Fields" Style="{StaticResource PlainBtnStyle}" Click="clearFieldsBtn\_Click" />  <Button Name="saveNotesBtn" Content="Save Notes" Style="{StaticResource PlainBtnStyle}" Click="saveNotesBtn\_Click" />  </UniformGrid>    </DockPanel>  <TextBox Name="notesTB" DockPanel.Dock="Top" TextBlock.LineHeight="20" TextBlock.LineStackingStrategy="BlockLineHeight" Padding="20,10,20,20" Height="100">    <TextBox.Background>  <DrawingBrush TileMode="Tile" Stretch="None" Viewport="0,0,20,20" ViewportUnits="Absolute" Opacity="1">  <DrawingBrush.Drawing>  <GeometryDrawing>  <GeometryDrawing.Pen>  <Pen Brush="RoyalBlue" Thickness=".1" />  </GeometryDrawing.Pen>  <GeometryDrawing.Geometry>  <LineGeometry StartPoint="0,0" EndPoint="20,0"/>  </GeometryDrawing.Geometry>  </GeometryDrawing>  </DrawingBrush.Drawing>  </DrawingBrush>  </TextBox.Background>  <TextBox.Style>  <Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">  <Style.Resources>  <VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Top" Stretch="None">  <VisualBrush.Visual>  <Label Content="what's on your mind?" Foreground="LightGray" />  </VisualBrush.Visual>  </VisualBrush>  </Style.Resources>  <Style.Triggers>  <Trigger Property="Text" Value="{x:Static sys:String.Empty}">  <Setter Property="Background" Value="{StaticResource CueBannerBrush}" />  </Trigger>  <Trigger Property="Text" Value="{x:Null}">  <Setter Property="Background" Value="{StaticResource CueBannerBrush}" />  </Trigger>  <Trigger Property="IsKeyboardFocused" Value="True">  <Setter Property="Background" Value="White" />  </Trigger>  </Style.Triggers>  </Style>  </TextBox.Style>  <!--Start writing from here...-->  </TextBox>  </DockPanel>    <DockPanel DockPanel.Dock="Left">  <UniformGrid DockPanel.Dock="Left" Columns="1">  <DockPanel LastChildFill="True">  <UniformGrid Rows="1" DockPanel.Dock="Bottom">  <Button Content="&lt;" Style="{StaticResource PlainBtnStyle}" Name="previousDateBtn" Click="previousDateBtn\_Click"></Button>  <Button Content="&gt;" Style="{StaticResource PlainBtnStyle}"></Button>  </UniformGrid>  <TextBlock Style="{StaticResource TxtblkStyle}" DockPanel.Dock="Top" Text="{Binding Source={x:Static sys:DateTime.Today}, StringFormat='{}{0:MMMM yyyy}'}" FontSize="18"></TextBlock>  </DockPanel>  <Button Name="rewriBtnBtn" ToolTip="save to cxcel" Style="{StaticResource PlainBtnStyle}">Rewrite Event</Button>  <Button Name="dltevntBtnBtn" ToolTip="save to cxcel" Style="{StaticResource PlainBtnStyle}">Delete Event</Button>  <Button Name="saveToExcelBtn" ToolTip="save to cxcel" Style="{StaticResource PlainBtnStyle}">Save To Excel</Button>  <DatePicker ToolTip="select a date" Margin="10" Name="goToDateDP" SelectedDate="{x:Static sys:DateTime.Now}" Style="{StaticResource DatepkrStyle}" ></DatePicker>  <Button Name="goToDateBtn" ToolTip="go to date" Style="{StaticResource PlainBtnStyle}" Click="goToDateBtn\_Click">Go To Date</Button>  </UniformGrid>  </DockPanel>    <DockPanel>  <TabControl Name="noteControlsTab" Margin="3" BorderBrush="#FFE9E0E0" TabStripPlacement="Right" HorizontalContentAlignment="Left" >  <TabItem Name="monthlyViewTC" Style="{StaticResource TItemStyle}" Header="Monthly View" Width="150" MouseRightButtonDown="monthlyViewTC\_click">  <DockPanel LastChildFill="True">    <UniformGrid Rows="1" Columns="7" DockPanel.Dock="Top">  <Label>Sunday</Label>  <Label>Monday</Label>  <Label>Tuesday</Label>  <Label>Wednesday</Label>  <Label>Thursday</Label>  <Label>Friday</Label>  <Label>Saturday</Label>  </UniformGrid>  <UniformGrid Name="dateUG" Rows="6" Columns="7" DockPanel.Dock="Bottom">    </UniformGrid>  </DockPanel>  </TabItem>  <TabItem Name="bItem" Style="{StaticResource TItemStyle}" Header="View All" Width="150">  <DockPanel LastChildFill="True">  <ListView Name="allnoteView" HorizontalAlignment="Stretch" ItemsSource="{Binding allnoteCollection}" >  <ListView.View>  <GridView>  <GridViewColumn Width="100" Header="Date" DisplayMemberBinding="{Binding noteDate}"/>  <GridViewColumn Width="650" Header="Event" DisplayMemberBinding="{Binding note}" />  </GridView>  </ListView.View>  </ListView>  </DockPanel>  </TabItem>  <TabItem Name="linktem" Style="{StaticResource TItemStyle}" Header="Daily View" Width="150">  <DockPanel LastChildFill="True">    <ListView></ListView>  </DockPanel>  </TabItem>  <TabItem Name="googlItem" Style="{StaticResource TItemStyle}" Header="Weekly View" Width="150">  <DockPanel LastChildFill="True">    <ListView></ListView>  </DockPanel>  </TabItem>    </TabControl>  </DockPanel>  </DockPanel>  </TabItem>  <TabItem Name="mainSocialSiteTabItem" Style="{StaticResource TItemStyle}" Header="Social Site" >  <TabControl Name="socialSiteTabControl" Margin="3" BorderBrush="#FFE9E0E0" TabStripPlacement="Left" HorizontalContentAlignment="Left" >  <TabItem Name="fbTabItem" Style="{StaticResource TItemStyle}" Header="Facebook" ToolTip="facebook">  <DockPanel LastChildFill="True">  <StackPanel DockPanel.Dock="Top" Orientation="Horizontal">  <Expander Name="loginToFBExpndr" DockPanel.Dock="Top" ExpandDirection="Down" IsExpanded="False" Header="Login to Facebook" VerticalAlignment="Center">  <UniformGrid Columns="1">  <DockPanel Width="600" >  <UniformGrid Columns="3" DockPanel.Dock="Left">  <Label Margin="10" Width="150">Facebook User Name</Label>  <Label Margin="10" Width="150">Password</Label>  <Label></Label>  <TextBox Margin="10" Width="150" Tag="enter user name" Style="{StaticResource commonTBtyle}" />  <PasswordBox Name="fbPB" Tag="••••••" Style="{StaticResource pBstyle}" Width="150" Margin="10"></PasswordBox>  <Button Name="fbLoginBtn" Width="100" ToolTip="login" Style="{StaticResource PlainBtnStyle}">Login</Button>  <Label></Label>  <CheckBox Name="fbStayLoginCheckBox" Margin="10" Width="150">Stay logged in</CheckBox>  <Label></Label>  </UniformGrid>  </DockPanel>  </UniformGrid>  </Expander>  <Button Content="Refresh Facebook Status" Style="{StaticResource PlainBtnStyle}" Name="fbRefreshButton" />  </StackPanel>  <ListView></ListView>  </DockPanel>  </TabItem>  <TabItem Name="twitterTabItem" Style="{StaticResource TItemStyle}" Header="Twitter" ToolTip="twitter">  <DockPanel LastChildFill="True">  <StackPanel DockPanel.Dock="Top" Orientation="Horizontal">  <Expander Name="logintotwitterExpndr" DockPanel.Dock="Top" ExpandDirection="Down" IsExpanded="False" Header="Login to Twitter" VerticalAlignment="Center">  <UniformGrid Columns="1">  <DockPanel Width="600" >  <UniformGrid Rows="1">  <Button Content="Launch Twitter App" Name="launchTwitterAppButton" Style="{StaticResource PlainBtnStyle}" Click="launchTwitterAppButton\_Click" />  <TextBox Name="verificationCodeTxtBox" Tag="enter apps code" Margin="20,8,0,8" Style="{StaticResource commonTBtyle}"></TextBox>  <Button Name="verifyTwitterApp" Style="{StaticResource PlainBtnStyle}" Click="verifyTwitterApp\_Click">Verify Twitter Apps Code</Button>  </UniformGrid>    </DockPanel>  </UniformGrid>  </Expander>  </StackPanel>  <TextBlock Name="tweetsTextBlock" Foreground="#FF8E6A6A" Background="#FFD06D6D" OpacityMask="#FF2F545F" />  </DockPanel>  </TabItem>  <TabItem Name="linkedInTabItem" Style="{StaticResource TItemStyle}" Header="LinkedIn" ToolTip="linkedIn" >  <DockPanel LastChildFill="True">  <StackPanel DockPanel.Dock="Top" Orientation="Horizontal">  <Expander DockPanel.Dock="Top" ExpandDirection="Down" IsExpanded="False" Header="Login" VerticalAlignment="Center">  <UniformGrid Columns="1">  <DockPanel Width="600" >  <UniformGrid Columns="3" DockPanel.Dock="Left">  <Label Margin="10" Width="150">LinkedIn User Name</Label>  <Label Margin="10" Width="150">Password</Label>  <Label></Label>  <TextBox Margin="10" Width="150" Tag="enter user name" Style="{StaticResource commonTBtyle}" />  <PasswordBox Width="150" Margin="10"></PasswordBox>  <Button Width="100" ToolTip="login" Style="{StaticResource PlainBtnStyle}">Login</Button>  <Label></Label>  <CheckBox Margin="10" Width="150">Stay logged in</CheckBox>  <Label></Label>  </UniformGrid>  </DockPanel>  </UniformGrid>  </Expander>  <Button Content="Refresh LinkedIn Status" ToolTip="refresh" Style="{StaticResource PlainBtnStyle}"/>  </StackPanel>  <ListView></ListView>  </DockPanel>  </TabItem>  <TabItem Name="googlePlusTabItem" Style="{StaticResource TItemStyle}" Header="Google Plus" ToolTip="google+">  <DockPanel LastChildFill="True">  <StackPanel DockPanel.Dock="Top" Orientation="Horizontal">  <Expander DockPanel.Dock="Top" ExpandDirection="Down" IsExpanded="False" Header="Login" VerticalAlignment="Center">  <UniformGrid Columns="1">  <DockPanel Width="600" >  <UniformGrid Columns="3" DockPanel.Dock="Left">  <Label Margin="10" Width="150">Google+ User Name</Label>  <Label Margin="10" Width="150">Password</Label>  <Label></Label>  <TextBox Margin="10" Width="150" Tag="enter user name" Style="{StaticResource commonTBtyle}" />  <PasswordBox Width="150" Margin="10"></PasswordBox>  <Button Width="100" ToolTip="login" Style="{StaticResource PlainBtnStyle}">Login</Button>  <Label></Label>  <CheckBox Margin="10" Width="150">Stay logged in</CheckBox>  <Label></Label>  </UniformGrid>  </DockPanel>  </UniformGrid>  </Expander>  <Button Content="Refresh Google+ Status" ToolTip="refresh" Style="{StaticResource PlainBtnStyle}"/>  </StackPanel>  <ListView></ListView>  </DockPanel>  </TabItem>  <TabItem Name="foursquareTabItem" Style="{StaticResource TItemStyle}" Header="Foursquare" ToolTip="foursquare">  <DockPanel LastChildFill="True">  <StackPanel DockPanel.Dock="Top" Orientation="Horizontal">  <Expander DockPanel.Dock="Top" ExpandDirection="Down" IsExpanded="False" Header="Login to Foursqare" VerticalAlignment="Center">  <UniformGrid Columns="1">  <DockPanel Width="600" >  <UniformGrid Columns="3" DockPanel.Dock="Left">  <Label Margin="10" Width="150">LinkedIn User Name</Label>  <Label Margin="10" Width="150">Password</Label>  <Label></Label>  <TextBox Margin="10" Width="150" Tag="enter user name" Style="{StaticResource commonTBtyle}" />  <PasswordBox Width="150" Margin="10"></PasswordBox>  <Button Width="100" ToolTip="login" Style="{StaticResource PlainBtnStyle}" >Login</Button>  <Label></Label>  <CheckBox Margin="10" Width="150">Stay logged in</CheckBox>  <Label></Label>  </UniformGrid>  </DockPanel>  </UniformGrid>  </Expander>  <Button Content="Refresh Foursqare" ToolTip="refresh" Style="{StaticResource PlainBtnStyle}"/>  </StackPanel>  <ListView></ListView>  </DockPanel>  </TabItem>  </TabControl>  </TabItem>  <TabItem Name="googleCalendarTabItem" Style="{StaticResource TItemStyle}" Header="Calendar">  <DockPanel LastChildFill="True">  <UniformGrid Rows="1" DockPanel.Dock="Bottom" Height="50">  <Button Style="{StaticResource PlainBtnStyle}" Name="gcalenderBtn" Click="gcalenderBtn\_Click">Load Calendar</Button>  <Label></Label>  <Label></Label>  <Label></Label>  <Label></Label>  </UniformGrid>  <WebBrowser Name="gCalenderWb"></WebBrowser>  </DockPanel>  </TabItem>  <TabItem Name="liveChatTabItem" Style="{StaticResource TItemStyle}" Header="Chat">  <DockPanel LastChildFill="True">  <Expander DockPanel.Dock="Top" ExpandDirection="Down" IsExpanded="False" Header="Login">  <UniformGrid Columns="1">  <DockPanel Width="600" >  <UniformGrid Columns="4" DockPanel.Dock="Left">  <Label Margin="10" Width="100">Chat Via</Label>  <Label Margin="10" Width="150">User Name</Label>  <Label Margin="10" Width="150">Password</Label>  <Label></Label>  <ComboBox Name="linkediBtn" Margin="10" Width="100" SelectedIndex="0">  <ComboBoxItem>Google+</ComboBoxItem>  <ComboBoxItem>Facebook</ComboBoxItem>  <ComboBoxItem>Yahoo</ComboBoxItem>  </ComboBox>  <TextBox Margin="10" Width="150" Tag="enter user name" Style="{StaticResource commonTBtyle}" />  <PasswordBox Width="150" Margin="10"></PasswordBox>  <Button Width="100" Style="{StaticResource PlainBtnStyle}">Login</Button>  <Label></Label>  <Label></Label>  <CheckBox Margin="10" Width="150">Stay logged in</CheckBox>  <Label></Label>  </UniformGrid>  </DockPanel>  </UniformGrid>  </Expander>  <ListView></ListView>  </DockPanel>  </TabItem>  <TabItem Name="googleMapTabItem" Style="{StaticResource TItemStyle}" Header="Map" >  <DockPanel LastChildFill="True">  <UniformGrid Rows="1" DockPanel.Dock="Bottom" Height="50">    <TextBox Name="loadMaplocationTB" Margin="10" Width="140" Tag="enter location" Style="{StaticResource commonTBtyle}" />  <Button Name="kolkataBtn" Style="{StaticResource PlainBtnStyle}" Click="kolkataBtn\_Click">Load Map</Button>  <Label></Label>  <Label></Label>  <Label></Label>  </UniformGrid>  <WebBrowser Name="mapWb"></WebBrowser>  </DockPanel>  </TabItem>  <TabItem Name="contactBookTabItem" Style="{StaticResource TItemStyle}" Header="Contacts" >  <DockPanel LastChildFill="True">  <StackPanel DockPanel.Dock="Top" Orientation="Horizontal">  <Expander Name="contacXpndr" DockPanel.Dock="Top" ExpandDirection="Down" IsExpanded="False" Header="Add Contact" VerticalAlignment="Center" >  <UniformGrid Columns="1">  <DockPanel Width="650" >  <UniformGrid Columns="4" DockPanel.Dock="Left" Width="647">  <Label Margin="10" Width="150">Name</Label>  <TextBox Name="contactnameTB" Margin="10" Width="140" Tag="enter name" Style="{StaticResource commonTBtyle}" />  <Label Margin="10" Width="150">Mobile Number</Label>  <TextBox Name="mobnoTB" Margin="10" Width="140" Tag="enter mobile number" Style="{StaticResource commonTBtyle}" />  <Label Margin="10" Width="150">EMail</Label>  <TextBox Name="emailTb" Margin="10" Width="140" Tag="enter email address" Style="{StaticResource commonTBtyle}" />  <Label Margin="10" Width="150">Home Number</Label>  <TextBox Name="homnoTB" Margin="10" Width="140" Tag="enter home number" Style="{StaticResource commonTBtyle}" />  <Label Margin="10" Width="150">Address</Label>  <TextBox Name="addressTB" Margin="10" Width="140" Tag="enter address" Style="{StaticResource commonTBtyle}" />  <Label Margin="10" Width="150">Office Number</Label>  <TextBox Name="ofcNoTb" Margin="10" Width="140" Tag="enter office number" Style="{StaticResource commonTBtyle}" />  <Label Margin="10" Width="150">Remarks</Label>  <TextBox Name="remrkTB" Margin="10" Width="140" Tag="enter remarks" Style="{StaticResource commonTBtyle}" />  <Label Margin="10" Width="150">Fax Number</Label>  <TextBox Name="faxNoTb" Margin="10" Width="140" Tag="enter fax number" Style="{StaticResource commonTBtyle}" />  <Label></Label>  <Label></Label>  <Button Content="Clear Fields" ToolTip="clear fields" Style="{StaticResource PlainBtnStyle}" />  <Button Content="Save" ToolTip="save info" Style="{StaticResource PlainBtnStyle}" Name="saveContactBtn" Click="saveContactBtn\_Click" />  </UniformGrid>  </DockPanel>  </UniformGrid>  </Expander>  </StackPanel>  <DockPanel DockPanel.Dock="Bottom">  <UniformGrid Rows="1">  <Label></Label>  <Label></Label>  <Label></Label>  <Button Name="contactrfrshBtn" Content="Refresh" ToolTip="Refresh details" Style="{StaticResource PlainBtnStyle}" Click="contactrfrshBtn\_Click" />  <Button Content="Edit Contacts" ToolTip="edit selected contact details" Style="{StaticResource PlainBtnStyle}" />  <Button Content="Delete Contact" ToolTip="delete selected contact details" Style="{StaticResource PlainBtnStyle}" />  </UniformGrid>  </DockPanel>  <ListView Name="contactView" HorizontalAlignment="Stretch" ItemsSource="{Binding contactCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="180" Header="Name" DisplayMemberBinding="{Binding name}" />  <GridViewColumn Width="150" Header="Mobile Number" DisplayMemberBinding="{Binding mobileno}" />  <GridViewColumn Width="130" Header="EMail" DisplayMemberBinding="{Binding email}" />  <GridViewColumn Width="150" Header="Home Number" DisplayMemberBinding="{Binding homeno}" />  <GridViewColumn Width="300" Header="Address" DisplayMemberBinding="{Binding address}" />  <GridViewColumn Width="150" Header="Office Number" DisplayMemberBinding="{Binding oficeno}" />  <GridViewColumn Width="150" Header="Fax Number" DisplayMemberBinding="{Binding faxno}" />  <GridViewColumn Width="100" Header="Remarks" DisplayMemberBinding="{Binding remark}" />  </GridView>  </ListView.View>  </ListView>  </DockPanel>  </TabItem>  <TabItem Name="passTabItem" Style="{StaticResource TItemStyle}" Header="Password">    <DockPanel LastChildFill="True">  <StackPanel DockPanel.Dock="Top" Orientation="Horizontal">  <Expander Name="loginpassPassExpndr" DockPanel.Dock="Top" ExpandDirection="Down" IsExpanded="True" Header="Login" VerticalAlignment="Center" >  <UniformGrid Columns="1">  <DockPanel Width="600" >  <UniformGrid Columns="3" DockPanel.Dock="Left">  <Label Margin="10" Width="150">User Name</Label>  <Label Margin="10" Width="150">Password</Label>  <Label></Label>  <TextBox Name="passwordMUserIdTB" Margin="10" Width="150" Tag="enter user name" Style="{StaticResource commonTBtyle}" />  <PasswordBox Name="passwordMPassBtnPB" Width="150" Margin="10"></PasswordBox>  <Button Name="passwordMLoginBtn" ToolTip="login" Width="100" Style="{StaticResource PlainBtnStyle}" Click="passwordMLoginBtn\_Click">Login</Button>  </UniformGrid>  </DockPanel>  </UniformGrid>  </Expander>  </StackPanel>  <DockPanel Name="mainPassDP" IsEnabled="False" LastChildFill="True">  <Expander Header="Login To Add Info" Name="addpassInfoExpndr" DockPanel.Dock="Top">  <UniformGrid Columns="4">  <Label Margin="10">Name</Label>  <TextBox Name="passNameTB" Margin="10" Tag="enter name" Style="{StaticResource commonTBtyle}" />  <Label Margin="10">EMail</Label>  <TextBox Name="emailforpassTB" Margin="10" Tag="enter email address" Style="{StaticResource commonTBtyle}" />  <Label Margin="10">User ID</Label>  <TextBox Name="userIdTB" Margin="10" Tag="enter user id" Style="{StaticResource commonTBtyle}" />  <Label Margin="10">Password</Label>  <TextBox Name="pawsrdTB" Margin="10" Tag="enter password" Style="{StaticResource commonTBtyle}" />  <Label Margin="10">Secret Question</Label>  <TextBox Name="scrtQstnTB" Margin="10" Tag="enter secret question" Style="{StaticResource commonTBtyle}" />  <Label Margin="10">Answer</Label>  <TextBox Name="secrtAnsTB" Margin="10" Tag="enter answer" Style="{StaticResource commonTBtyle}" />  <Label Margin="10">Others Info</Label>  <TextBox Name="othersInfoTB" Margin="10" Tag="enter others info" Style="{StaticResource commonTBtyle}" />  <Button Name="clearPassInfoFieldBtn" ToolTip="clear all fields" Width="150" Style="{StaticResource PlainBtnStyle}" Click="clearPassInfoFieldBtn\_Click">Clear Fields</Button>  <Button Name="paswrdSaveBtn" ToolTip="save info" Width="150" Style="{StaticResource PlainBtnStyle}" Click="paswrdSave\_Click">Save</Button>  </UniformGrid>  </Expander>  <UniformGrid DockPanel.Dock="Bottom" Columns="6" Name="logoutDP">  <Label></Label>  <Label></Label>  <Button Content="Refresh" Name="refrshpsdwBtn" ToolTip="refresh details" Style="{StaticResource PlainBtnStyle}" Click="refrshpsdwBtn\_Click" />  <Button Content="Edit Details" ToolTip="edit selected details" Style="{StaticResource PlainBtnStyle}" />  <Button Content="Delete Details" ToolTip="delete selected details" Style="{StaticResource PlainBtnStyle}" />  <Button Content="Please Log In First" Name="logoutPassBtn" IsEnabled="False" ToolTip="logout" Style="{StaticResource PlainBtnStyle}" Click="logoutPassBtn\_Click" />  </UniformGrid>  <UniformGrid DockPanel.Dock="Top">  <Expander Header="Login To Show All Details" Name="allDetailsExpndr">  <ListView Name="passView" HorizontalAlignment="Stretch" ItemsSource="{Binding passwordCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="100" Header="Name" DisplayMemberBinding="{Binding name}" />  <GridViewColumn Width="150" Header="Mail" DisplayMemberBinding="{Binding email}" />  <GridViewColumn Width="100" Header="ID" DisplayMemberBinding="{Binding userId}" />  <GridViewColumn Width="150" Header="Password" DisplayMemberBinding="{Binding password}" />  <GridViewColumn Width="300" Header="Secret Question" DisplayMemberBinding="{Binding scrtqstn}" />  <GridViewColumn Width="100" Header="Answer" DisplayMemberBinding="{Binding scrtans}"/>  <GridViewColumn Width="100" Header="Others Info" DisplayMemberBinding="{Binding otherInfo}" />  </GridView>  </ListView.View>  </ListView>  </Expander>  </UniformGrid>  </DockPanel>  </DockPanel>  </TabItem>  <TabItem Name="extraTabItem" Style="{StaticResource TItemStyle}" Header="Extra">    <TabControl Name="expnceControl" Margin="3" BorderBrush="#FFE9E0E0" TabStripPlacement="Left" HorizontalContentAlignment="Left" >  <TabItem Name="taskTabItem" Style="{StaticResource TItemStyle}" Header="Task" ToolTip="task">  <DockPanel LastChildFill="True">  <StackPanel DockPanel.Dock="Top" Orientation="Horizontal">  <Expander Name="tskxpndr" DockPanel.Dock="Top" ExpandDirection="Down" IsExpanded="False" Header="Add Task" VerticalAlignment="Center">  <UniformGrid Columns="1">  <DockPanel Width="650" Height="197">  <UniformGrid Columns="4" DockPanel.Dock="Left" Width="647" Height="198">  <Label Margin="10" Width="150" VerticalAlignment="Center">Task Value</Label>  <ComboBox Name="taskValueCB" Margin="10" Width="142" Height="30" ToolTip="task value" VerticalContentAlignment="Center" HorizontalContentAlignment="Center" SelectedIndex="0">  <ComboBoxItem>High</ComboBoxItem>  <ComboBoxItem>Medium</ComboBoxItem>  <ComboBoxItem>Low</ComboBoxItem>  </ComboBox>  <Label Margin="10" Width="150" VerticalAlignment="Center">About Task</Label>  <TextBox Name="taskDetailsTB" Height="100" TextWrapping="Wrap" AcceptsReturn="True" VerticalScrollBarVisibility="Visible" SpellCheck.IsEnabled="True">  <TextBox.Style>  <Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">  <Style.Resources>  <VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Top" Stretch="None">  <VisualBrush.Visual>  <Label Content="what to do?" Foreground="LightGray" />  </VisualBrush.Visual>  </VisualBrush>  </Style.Resources>  <Style.Triggers>  <Trigger Property="Text" Value="{x:Static sys:String.Empty}">  <Setter Property="Background" Value="{StaticResource CueBannerBrush}" />  </Trigger>  <Trigger Property="Text" Value="{x:Null}">  <Setter Property="Background" Value="{StaticResource CueBannerBrush}" />  </Trigger>  <Trigger Property="IsKeyboardFocused" Value="True">  <Setter Property="Background" Value="White" />  </Trigger>  </Style.Triggers>  </Style>  </TextBox.Style>  </TextBox>  <Label></Label>  <Label></Label>  <Button Content="Clear" ToolTip="clear field" Style="{StaticResource PlainBtnStyle}" Height="25" Width="130" />  <Button Name="addTaskBtn" Content="Save" ToolTip="save task" Style="{StaticResource PlainBtnStyle}" Height="25" Width="130" Click="addTaskBtn\_Click" />  </UniformGrid>  </DockPanel>  </UniformGrid>  </Expander>  </StackPanel>  <DockPanel DockPanel.Dock="Bottom">  <UniformGrid Rows="1">  <Label></Label>  <Label></Label>  <Label></Label>  <Button Content="Refresh" Name="refrshTskBtn" ToolTip="refresh task" Style="{StaticResource PlainBtnStyle}" Click="contactrfrshBtn\_Click" />  <Button Content="Edit Task" ToolTip="edit selected task" Style="{StaticResource PlainBtnStyle}" />  <Button Content="Delete Task" ToolTip="delete selected task" Style="{StaticResource PlainBtnStyle}" />  </UniformGrid>  </DockPanel>  <ListView Name="taskView" HorizontalAlignment="Stretch" ItemsSource="{Binding taskCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="100" Header="Value" DisplayMemberBinding="{Binding value}"/>  <GridViewColumn Width="650" Header="Task" DisplayMemberBinding="{Binding taskDetails}" />  </GridView>  </ListView.View>  </ListView>  </DockPanel>  </TabItem>  <TabItem Name="expenseTabItem" Style="{StaticResource TItemStyle}" Header="Expense" ToolTip="expense">  </TabItem>  </TabControl>  </TabItem>  </TabControl>  </DockPanel>  </DockPanel>  </Border>  </Window> |

|  |
| --- |
| ShowEvent.xaml |
| <UserControl x:Class="**CMS**GUI.ShowEvent"  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">  <DockPanel LastChildFill="True">  <UniformGrid DockPanel.Dock="Top">  <Label Name="sequenceLabel"></Label>  </UniformGrid>  <UniformGrid DockPanel.Dock="Top">  <ListView Background="Gray"></ListView>  </UniformGrid>  </DockPanel>  </UserControl> |

### GUI Style : CMSStyles

|  |
| --- |
| ControlStyle.xaml |
| <ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">    <Style x:Key="PlainBtnStyle" TargetType="Button">  <Setter Property="Background" Value="#D8D8D8" />  <Setter Property="Foreground" Value="Black" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="BorderBrush" Value="#FFC4C458" />  <Setter Property="Margin" Value="10"/>  <Setter Property="OpacityMask" Value="White" />  <Setter Property="Padding" Value="7" />  <Setter Property="Template">  <Setter.Value>  <ControlTemplate TargetType="Button">  <Border Name="border"  BorderThickness="1"  Padding="4,2"  BorderBrush="White"  CornerRadius="0"  Background="{TemplateBinding Background}">  <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center" />  </Border>  <ControlTemplate.Triggers>  <Trigger Property="IsMouseOver" Value="True">  <Setter TargetName="border" Property="BorderBrush" Value="#2E9AFE" />  <Setter Property="Button.Background" Value="#2E9AFE" />  <Setter Property="Button.Foreground" Value="White" />  </Trigger>  <Trigger Property="IsPressed" Value="True">  <Setter TargetName="border" Property="BorderBrush" Value="White" />  <Setter Property="Button.Background" Value="#D8D8D8" />  </Trigger>    </ControlTemplate.Triggers>    </ControlTemplate>  </Setter.Value>  </Setter>  </Style>  <Style x:Key="TItemStyle" TargetType="TabItem">  <Setter Property="Width" Value="115" />  <Setter Property="Height" Value="33" />  <Setter Property="Background" Value="#D8D8D8" />  <Setter Property="BorderBrush" Value="white" />  <Setter Property="Template">  <Setter.Value>  <ControlTemplate TargetType="{x:Type TabItem}">  <Grid>  <Border Name="Border" Margin="0,0,0,0" Background="#D8D8D8"  BorderBrush="#D8D8D8" BorderThickness="1,1,1,1" CornerRadius="0">  <ContentPresenter x:Name="ContentSite" VerticalAlignment="Center"  HorizontalAlignment="Center"  ContentSource="Header" Margin="12,2,12,2"  RecognizesAccessKey="True">  <ContentPresenter.LayoutTransform>  <RotateTransform Angle="0" />  </ContentPresenter.LayoutTransform>  </ContentPresenter>  </Border>  </Grid>  <ControlTemplate.Triggers>  <Trigger Property="IsSelected" Value="True">  <Setter TargetName="Border" Property="Background" Value="#2E9AFE" />  <Setter Property="Button.Foreground" Value="White" />  </Trigger>  <Trigger Property="IsMouseOver" Value="True">  <Setter TargetName="Border" Property="BorderBrush" Value="#2E9AFE" />  <Setter TargetName="Border" Property="Background" Value="#2E9AFE" />  <Setter Property="Button.Foreground" Value="White" />  </Trigger>  </ControlTemplate.Triggers>  </ControlTemplate>  </Setter.Value>  </Setter>  <Setter Property="HeaderTemplate">  <Setter.Value>  <DataTemplate>  <Border x:Name="grid" >  <ContentPresenter>  <ContentPresenter.Content>  <TextBlock Margin="4" FontSize="15" Text="{TemplateBinding Content}"/>  </ContentPresenter.Content>  </ContentPresenter>  </Border>  </DataTemplate>  </Setter.Value>  </Setter>  </Style>  <Style x:Key="CheckBtyle" TargetType="CheckBox">  <Setter Property="Width" Value="80" />  <Setter Property="Background" Value="#FFF9F9EC" />  <Setter Property="BorderBrush" Value="#FFC4C458" />  <Setter Property="Foreground" Value="#000000" />  </Style>  <Style x:Key="pBstyle" TargetType="PasswordBox">  <Setter Property="Control.Template" >  <Setter.Value>  <ControlTemplate TargetType="{x:Type PasswordBox}">  <Border x:Name="border" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" Background="{TemplateBinding Background}" SnapsToDevicePixels="True">  <Grid>  <ScrollViewer x:Name="PART\_ContentHost" Focusable="False" HorizontalScrollBarVisibility="Hidden" VerticalScrollBarVisibility="Hidden"/>  <TextBlock x:Name="InternalWatermarkLabel"  Text="{TemplateBinding Tag}"  Visibility="Collapsed" Focusable="False"  VerticalAlignment="Top" Margin=" 5 1 0 0"  Foreground="Silver"  Background="Transparent"/>  </Grid>  </Border>  <ControlTemplate.Triggers>  <MultiTrigger>  <MultiTrigger.Conditions>  <Condition Property="IsFocused" Value="False" />  </MultiTrigger.Conditions>  <MultiTrigger.Setters>  <Setter Property="Visibility" TargetName="InternalWatermarkLabel"  Value="Visible" />  </MultiTrigger.Setters>  </MultiTrigger>  <Trigger Property="IsEnabled" Value="False">  <Setter Property="Opacity" TargetName="border" Value="0.56"/>  </Trigger>  <Trigger Property="IsMouseOver" Value="True">  <Setter Property="BorderBrush" TargetName="border" Value="#FF7EB4EA"/>  </Trigger>  <Trigger Property="IsKeyboardFocused" Value="True">  <Setter Property="BorderBrush" TargetName="border" Value="#FF569DE5"/>  </Trigger>  </ControlTemplate.Triggers>  </ControlTemplate>  </Setter.Value>  </Setter>  </Style>  <Style x:Key="TitlebarBTN" TargetType="Button">  <Setter Property="OverridesDefaultStyle" Value="True"/>  <Setter Property="Margin" Value="0"/>  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="Template">  <Setter.Value>  <ControlTemplate TargetType="Button">  <Border Name="border"  BorderThickness="1"  Padding="4,2"  BorderBrush="White"  CornerRadius="0"  Background="{TemplateBinding Background}">  <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center" />  </Border>  <ControlTemplate.Triggers>  <Trigger Property="IsMouseOver" Value="True">  <Setter TargetName="border" Property="BorderBrush" Value="White" />  <Setter Property="Button.Background" Value="#D8D8D8" />  </Trigger>  <Trigger Property="IsPressed" Value="True">  <Setter TargetName="border" Property="BorderBrush" Value="White" />  <Setter Property="Button.Background" Value="#2E9AFE" />  </Trigger>  </ControlTemplate.Triggers>  </ControlTemplate>  </Setter.Value>  </Setter>  </Style>  <Style x:Key="{x:Type ToolTip}" TargetType="{x:Type ToolTip}">  <Setter Property="Background" Value="Black"/>  <Setter Property="BorderBrush" Value="#D8D8D8"/>  <Setter Property="Foreground" Value="White" />  <Setter Property="Padding" Value="7" />  <Setter Property="FontWeight" Value="Bold" />    </Style>  <Style x:Key="commonTBtyle" TargetType="TextBox">    <Setter Property="Control.Template" >  <Setter.Value>  <ControlTemplate TargetType="{x:Type TextBox}">  <Border x:Name="border" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" Background="{TemplateBinding Background}" SnapsToDevicePixels="True">  <Grid>  <ScrollViewer x:Name="PART\_ContentHost" Focusable="False" HorizontalScrollBarVisibility="Hidden" VerticalScrollBarVisibility="Hidden"/>  <TextBlock x:Name="InternalWatermarkLabel"  Text="{TemplateBinding Tag}"  Visibility="Collapsed" Focusable="False"  VerticalAlignment="Top" Margin=" 5 1 0 0"  Foreground="Silver"  Background="Transparent"/>  </Grid>  </Border>  <ControlTemplate.Triggers>  <MultiTrigger>  <MultiTrigger.Conditions>  <Condition Property="IsFocused" Value="False" />  <Condition Property="Text" Value="" />  </MultiTrigger.Conditions>  <MultiTrigger.Setters>  <Setter Property="Visibility" TargetName="InternalWatermarkLabel"  Value="Visible" />  </MultiTrigger.Setters>  </MultiTrigger>  <Trigger Property="IsEnabled" Value="False">  <Setter Property="Opacity" TargetName="border" Value="0.56"/>  </Trigger>  <Trigger Property="IsMouseOver" Value="True">  <Setter Property="BorderBrush" TargetName="border" Value="#FF7EB4EA"/>    </Trigger>  <Trigger Property="IsKeyboardFocused" Value="True">  <Setter Property="BorderBrush" TargetName="border" Value="#FF569DE5"/>  </Trigger>  </ControlTemplate.Triggers>  </ControlTemplate>  </Setter.Value>  </Setter>  </Style>  <Style TargetType="{x:Type ScrollBar}">  <Setter Property="Stylus.IsFlicksEnabled" Value="True" />  <Setter Property="Background" Value="#D8D8D8" />  <Setter Property="Width" Value="15"/>  <Setter Property="MinWidth" Value="8" />  <Setter Property="Template">    <Setter.Value>  <ControlTemplate TargetType="{x:Type ScrollBar}">  <Grid x:Name="GridRoot" Width="19" Background="{TemplateBinding Background}">  <Grid.RowDefinitions>  <RowDefinition Height="0.00001\*" />  </Grid.RowDefinitions>  <Track x:Name="PART\_Track" Grid.Row="0" IsDirectionReversed="true" Focusable="false">  <Track.Thumb>  <Thumb x:Name="Thumb" Background="Gray" Style="{DynamicResource ScrollThumbs}" />  </Track.Thumb>  <Track.IncreaseRepeatButton>  <RepeatButton x:Name="PageUp" Command="ScrollBar.PageDownCommand" Opacity="0" Focusable="false" />  </Track.IncreaseRepeatButton>  <Track.DecreaseRepeatButton>  <RepeatButton x:Name="PageDown" Command="ScrollBar.PageUpCommand" Opacity="0" Focusable="false" />  </Track.DecreaseRepeatButton>  </Track>  </Grid>  <ControlTemplate.Triggers>  <Trigger SourceName="Thumb" Property="IsMouseOver" Value="true">  <Setter Value="#2E9AFE" TargetName="Thumb" Property="Background" />  </Trigger>  <Trigger SourceName="Thumb" Property="IsDragging" Value="true">  <Setter Value="#D8D8D8" TargetName="Thumb" Property="Background" />  </Trigger>  <Trigger Property="IsEnabled" Value="false">  <Setter TargetName="Thumb" Property="Visibility" Value="Collapsed" />  </Trigger>  <Trigger Property="Orientation" Value="Horizontal">  <Setter TargetName="GridRoot" Property="LayoutTransform">  <Setter.Value>  <RotateTransform Angle="-90" />  </Setter.Value>  </Setter>  <Setter TargetName="PART\_Track" Property="LayoutTransform">  <Setter.Value>  <RotateTransform Angle="-90" />  </Setter.Value>  </Setter>  <Setter Property="Width" Value="Auto" />  <Setter Property="Height" Value="15" />  <Setter TargetName="Thumb" Property="Tag" Value="Horizontal" />  <Setter TargetName="PageDown" Property="Command" Value="ScrollBar.PageLeftCommand" />  <Setter TargetName="PageUp" Property="Command" Value="ScrollBar.PageRightCommand" />  </Trigger>  </ControlTemplate.Triggers>  </ControlTemplate>  </Setter.Value>  </Setter>  </Style>  <Style x:Key="LblStyle" TargetType="Label">  <Setter Property="VerticalContentAlignment" Value="Center" />  <Setter Property="HorizontalContentAlignment" Value="Left" />  <Setter Property="Foreground" Value="Black" />  <Setter Property="FontWeight" Value="Bold" />    </Style>  <Style x:Key="TxtblkStyle" TargetType="TextBlock">  <Setter Property="VerticalAlignment" Value="Center" />  <Setter Property="HorizontalAlignment" Value="Center" />  <Setter Property="Foreground" Value="Black" />  <Setter Property="FontWeight" Value="Bold" />  </Style>  <Style x:Key="DatepkrStyle" TargetType="DatePicker">  <Setter Property="VerticalAlignment" Value="Center" />  <Setter Property="HorizontalAlignment" Value="Left" />  <Setter Property="Foreground" Value="Black" />  <Setter Property="FontWeight" Value="Bold" />  </Style>  </ResourceDictionary> |

### C# coding

|  |
| --- |
| 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;  using **CMS**Data;  using **CMS**Db;  using System.Collections.ObjectModel;  using TweetSharp;  using System.Diagnostics;  namespace **CMS**GUI  {  /// <summary>  /// Interaction logic for MainWindow.xaml  /// </summary>  public partial class MainWindow : Window  {  public MainWindow()  {  InitializeComponent();  this.MouseLeftButtonDown += delegate { this.DragMove(); };  FocusManager.SetFocusedElement(this, **CMS**UserIDTB);  }  private void Window\_Loaded(object sender, RoutedEventArgs e)  {  fetchTaskData();  fetchContactData();  fetchPasswordData();  fetchNoteData();  for (int i = 1; i <= 30; i++)  dateUG.Children.Add(new ShowEvent(i));  }  private void kolkataBtn\_Click(object sender, RoutedEventArgs e)  {  mapWb.NavigateToString("<html><body><iframe src=\"https://maps.google.co.in/?ll=20.98352,82.752628&spn=43.617141,86.572266&t=h&z=4\" width=\"1100\" height=\"480\" frameborder=\"0\" marginheight=\"0\" marginwidth=\"0\">Loading...</iframe></body></html>");  }  private void gcalenderBtn\_Click(object sender, RoutedEventArgs e)  {  gCalenderWb.NavigateToString("<html><body><iframe src=\"https://www.facebook.com/connect/login\_success.html#access\_token=THE\_TOKEN&expires\_in=7180\" width=\"1100\" height=\"480\" frameborder=\"0\" marginheight=\"0\" marginwidth=\"0\">Loading...</iframe></body></html>");    }  private void Image\_ImageFailed(object sender, ExceptionRoutedEventArgs e)  {  }  private void closeBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  private void virtualKeyboardDateBtn\_Click(object sender, RoutedEventArgs e)  {  System.Diagnostics.Process.Start("calc");  }  private void minimizeBtn\_Click(object sender, RoutedEventArgs e)  {  this.WindowState = WindowState.Minimized;  }  //private void maximizeBtn\_Click(object sender, RoutedEventArgs e)  //{  // this.WindowState = WindowState.Maximized;  //}  private void passwordMLoginBtn\_Click(object sender, RoutedEventArgs e)  {  if ((passwordMUserIdTB.Text.Equals("1")) && (passwordMPassBtnPB.Password.Equals("1")))  {  mainPassDP.IsEnabled = true;  passwordMPassBtnPB.Password = string.Empty;  loginpassPassExpndr.IsExpanded = false;  loginpassPassExpndr.IsEnabled = false;  loginpassPassExpndr.Header = "Logged In";  loginpassPassExpndr.ToolTip = "Logged In";  allDetailsExpndr.IsExpanded = true;  logoutPassBtn.Content = "Logout";  allDetailsExpndr.Header = "All Details";  addpassInfoExpndr.Header = "Add Info";  logoutPassBtn.IsEnabled = true;  }  else  {  //MessageBox.Show("Please Enter correct Password");  //**CMS**GUI.ErrorMessage ErrorMessageObj = new **CMS**GUI.ErrorMessage();  //ErrorMessageObj.ShowDialog();  passwordMPassBtnPB.Password = string.Empty;  }  }  private void logout**CMS**Btn\_Click(object sender, RoutedEventArgs e)  {  **CMS**UserIDTB.Text = string.Empty;  **CMS**passPB.Password = string.Empty;  mainLeftExpndr.IsExpanded = false;  mainLeftExpndr.IsEnabled = false;  mainTabControl.IsEnabled = false;  logout**CMS**Btn.IsEnabled = false;  logout**CMS**Btn.Content = "";  loginExpndr.IsExpanded = true;  loginExpndr.IsEnabled = true;  hntLginLbl.Content = "Login first to use Daily Note Book with Social Networking Updater";  loginExpndr.Visibility = Visibility.Visible;  loginExpndr.IsExpanded = true;  **CMS**UserIDTB.Clear();  **CMS**passPB.Clear();  }  private void **CMS**LoginBtn\_Click(object sender, RoutedEventArgs e)  {  if (**CMS**UserIDTB.Text.Equals(**CMS**Db.DbInteraction.FetcheId()) && **CMS**passPB.Password.Equals(**CMS**Db.DbInteraction.FetchePassword()))  {  mainLeftExpndr.IsEnabled = true;  mainLeftExpndr.IsExpanded = true;  **CMS**passPB.Password = string.Empty;  mainTabControl.IsEnabled = true;  loginExpndr.IsEnabled = false;  //loginExpndr.Header = "Logged In";  //loginExpndr.IsExpanded = false;  loginExpndr.Visibility = Visibility.Collapsed;  logout**CMS**Btn.IsEnabled = true;  logout**CMS**Btn.Content = "LOG OUT";  logout**CMS**Btn.ToolTip = "log out";  hntLginLbl.Content = "";  }  else  {  //MessageBox.Show("Please Enter correct Password");  //**CMS**GUI.ErrorMessage ErrorMessageObj = new **CMS**GUI.ErrorMessage();  //ErrorMessageObj.ShowDialog();  errorMsgLbl.Content = "Wrong User ID or Password";  **CMS**passPB.Password = string.Empty;  }  }  private void logoutPassBtn\_Click(object sender, RoutedEventArgs e)  {  mainPassDP.IsEnabled = false;  passwordMPassBtnPB.Password = string.Empty;  passwordMUserIdTB.Text = string.Empty;  loginpassPassExpndr.IsExpanded = true;  loginpassPassExpndr.IsEnabled = true;  //loginpassPassExpndr.Header = "Logged In";  //loginpassPassExpndr.ToolTip = "Logged In";  allDetailsExpndr.IsExpanded = false;  logoutPassBtn.Content = "Please Log In First";  addpassInfoExpndr.Header = "Login To Add Info";  allDetailsExpndr.Header = "Login To Show All Details";  //allDetailsExpndr.Header = "All Details";  //addInfoExpndr.Header = "Add Info";  logoutPassBtn.IsEnabled = false;    }  private void createAcHp\_Click(object sender, RoutedEventArgs e)  {  newAcExpndr.Visibility = Visibility.Visible;  newAcExpndr.IsExpanded = true;  loginExpndr.Visibility = Visibility.Collapsed;  **CMS**NewUserIDTB.Clear();  **CMS**NewPassPB.Clear();  **CMS**NewRepassPB.Clear();  **CMS**HintsTB.Clear();  createErrorMsgLbl.Content = string.Empty;  }  private void goLogin\_Click(object sender, RoutedEventArgs e)  {  newAcExpndr.Visibility = Visibility.Collapsed;  loginExpndr.Visibility = Visibility.Visible;  loginExpndr.IsExpanded = true;  **CMS**UserIDTB.Clear();  **CMS**passPB.Clear();  errorMsgLbl.Content = string.Empty;  hintsMsgLbl.Content = string.Empty;  }  TwitterService service;  OAuthRequestToken requestToken;  private void launchTwitterAppButton\_Click(object sender, RoutedEventArgs e)  {  // Pass your credentials to the service  service = new TwitterService("rfeDMlcxMQSqDSdmXDT3A", "7ZUD4YtVqOfFJcCHwWz7q0qTFjcKyVGSNyN4ElCsZZE");  // Step 1 - Retrieve an OAuth Request Token  requestToken = service.GetRequestToken();  // Step 2 - Redirect to the OAuth Authorization URL  Uri uri = service.GetAuthorizationUri(requestToken);  Process.Start(uri.ToString());  }  private void verifyTwitterApp\_Click(object sender, RoutedEventArgs e)  {  // Step 3 - Exchange the Request Token for an Access Token  OAuthAccessToken access = service.GetAccessToken(requestToken, verificationCodeTxtBox.Text);  // Step 4 - User authenticates using the Access Token  service.AuthenticateWith(access.Token, access.TokenSecret);  var tweets = service.ListTweetsOnHomeTimeline(new ListTweetsOnHomeTimelineOptions());  foreach (var tweet in tweets)  {  tweetsTextBlock.Text += tweet.User.ScreenName + " says " + tweet.Text + "\n";  }  }  private void dNBSSubmitBtn\_Click(object sender, RoutedEventArgs e)  {  if (!**CMS**NewUserIDTB.Text.Equals(string.Empty) &&  !**CMS**NewPassPB.Password.Equals(string.Empty) &&  !**CMS**NewRepassPB.Password.Equals(string.Empty) &&  !**CMS**HintsTB.Text.Equals(string.Empty))  {  if (!**CMS**NewPassPB.Password.Equals(**CMS**NewRepassPB.Password))  {  **CMS**NewPassPB.Clear();  **CMS**NewRepassPB.Clear();  createErrorMsgLbl.Content = "Please Enter same Password";  }  else  {  **CMS**Data.UserInfo newUser = new **CMS**Data.UserInfo();  newUser.id = GenerateId();  newUser.userId = **CMS**NewUserIDTB.Text;  newUser.pass = **CMS**NewPassPB.Password;  newUser.hints = **CMS**HintsTB.Text;  **CMS**Db.DbInteraction.DoRegisterNewUser(newUser);  newAcExpndr.Visibility = Visibility.Collapsed;  loginExpndr.Visibility = Visibility.Visible;  loginExpndr.IsExpanded = true;  **CMS**UserIDTB.Clear();  **CMS**passPB.Clear();  }  }  else  {  createErrorMsgLbl.Content = "Correctly Enter Info ";  }  }  private void previousDateBtn\_Click(object sender, RoutedEventArgs e)  {  }  private string GenerateId()  {  return DateTime.Now.ToOADate().ToString();  }  #region Note  private void saveNotesBtn\_Click(object sender, RoutedEventArgs e)  {  if (!notesTB.Text.Equals(string.Empty) && !notesTB.Text.Equals("Start writing from here..."))  {  **CMS**Data.NoteInfo newNote = new **CMS**Data.NoteInfo();  newNote.id = GenerateId();  newNote.gotoDate = noteDateDP.SelectedDate.Value;  newNote.note = notesTB.Text;  **CMS**Db.DbInteraction.DoEnterNewNote(newNote);  notesTB.Clear();  noteSuccessMsgLvl.Content = "Event Submited";  fetchNoteData();  }  else  {  noteSuccessMsgLvl.Content = "Please Enter Proper Note";  }  }  private void clearFieldsBtn\_Click(object sender, RoutedEventArgs e)  {  noteSuccessMsgLvl.Content = "";  notesTB.Clear();  noteSuccessMsgLvl.Content = "All Clear";  }  ObservableCollection<NoteInfo> \_allnoteCollection = new ObservableCollection<NoteInfo>();  public ObservableCollection<NoteInfo> allnoteCollection  {  get  {  return \_allnoteCollection;  }  }  private void fetchNoteData()  {  List<NoteInfo> notes = DbInteraction.GetAllNoteList();  \_allnoteCollection.Clear();  foreach (NoteInfo note in notes)  {  \_allnoteCollection.Add(note);  }  }  #endregion  #region Password  private void paswrdSave\_Click(object sender, RoutedEventArgs e)  {  **CMS**Data.PasswordInfo newPassword = new **CMS**Data.PasswordInfo();  newPassword.id = GenerateId();  newPassword.name = passNameTB.Text;  newPassword.email = emailforpassTB.Text;  newPassword.userId = userIdTB.Text;  newPassword.password = pawsrdTB.Text;  newPassword.scrtqstn = scrtQstnTB.Text;  newPassword.scrtans = secrtAnsTB.Text;  newPassword.otherInfo = othersInfoTB.Text;  **CMS**Db.DbInteraction.DoRegisterNewPassword(newPassword);  passNameTB.Text = emailforpassTB.Text = userIdTB.Text = pawsrdTB.Text = scrtQstnTB.Text = secrtAnsTB.Text = othersInfoTB.Text = "";  addpassInfoExpndr.IsExpanded = false;  fetchPasswordData();  }  private void clearPassInfoFieldBtn\_Click(object sender, RoutedEventArgs e)  {  passNameTB.Text = emailforpassTB.Text = userIdTB.Text = pawsrdTB.Text = scrtQstnTB.Text = secrtAnsTB.Text = othersInfoTB.Text = "";  }  ObservableCollection<PasswordInfo> \_passwordCollection = new ObservableCollection<PasswordInfo>();  public ObservableCollection<PasswordInfo> passwordCollection  {  get  {  return \_passwordCollection;  }  }  private void fetchPasswordData()  {  List<PasswordInfo> passwords = DbInteraction.GetAllPasswordsList();  \_passwordCollection.Clear();  foreach (PasswordInfo password in passwords)  {  \_passwordCollection.Add(password);  }  }  private void refrshpsdwBtn\_Click(object sender, RoutedEventArgs e)  {  fetchPasswordData();  }  #endregion  #region Task  private void addTaskBtn\_Click(object sender, RoutedEventArgs e)  {  **CMS**Data.TaskInfo newTask = new **CMS**Data.TaskInfo();  newTask.id = GenerateId();  newTask.value = taskValueCB.Text;  newTask.taskDetails = taskDetailsTB.Text;  **CMS**Db.DbInteraction.DoRegisterNewTask(newTask);  taskValueCB.Text = taskDetailsTB.Text = "";  fetchTaskData();  tskxpndr.IsExpanded = false;  }  ObservableCollection<TaskInfo> \_taskCollection = new ObservableCollection<TaskInfo>();  public ObservableCollection<TaskInfo> taskCollection  {  get  {  return \_taskCollection;  }  }  private void fetchTaskData()  {  List<TaskInfo> tasks = DbInteraction.GetAllTaskList();  \_taskCollection.Clear();  foreach (TaskInfo task in tasks)  {  \_taskCollection.Add(task);  }  }  private void refrshTskBtn\_Click(object sender, RoutedEventArgs e)  {  fetchContactData();  }  #endregion  #region Contact  private void saveContactBtn\_Click(object sender, RoutedEventArgs e)  {  **CMS**Data.ContactInfo newContact = new **CMS**Data.ContactInfo();  newContact.id = GenerateId();  newContact.name = contactnameTB.Text;  newContact.mobileno = mobnoTB.Text;  newContact.homeno = homnoTB.Text;  newContact.oficeno = ofcNoTb.Text;  newContact.faxno = faxNoTb.Text;  newContact.address = addressTB.Text;  newContact.remark = remrkTB.Text;  newContact.email = emailTb.Text;  **CMS**Db.DbInteraction.DoRegisterNewContact(newContact);  contactnameTB.Text = mobnoTB.Text = homnoTB.Text = ofcNoTb.Text = faxNoTb.Text = addressTB.Text = remrkTB.Text = emailTb.Text = "";  fetchContactData();  contacXpndr.IsExpanded = true;  }  ObservableCollection<ContactInfo> \_contactCollection = new ObservableCollection<ContactInfo>();  public ObservableCollection<ContactInfo> contactCollection  {  get  {  return \_contactCollection;  }  }  private void fetchContactData()  {  List<ContactInfo> contacts = DbInteraction.GetAllContactList();  \_contactCollection.Clear();  foreach (ContactInfo Contacts in contacts)  {  \_contactCollection.Add(Contacts);  }  }  private void contactrfrshBtn\_Click(object sender, RoutedEventArgs e)  {  fetchContactData();  }  #endregion  #region Go to Date  private void goToDateBtn\_Click(object sender, RoutedEventArgs e)  {  noteControlsTab.SelectedIndex = 1;  NoteInfo noteinfo = new NoteInfo();  noteinfo.gotoDate = goToDateDP.SelectedDate.Value;  List<NoteInfo> notes = DbInteraction.searchNoteList(noteinfo);  \_allnoteCollection.Clear();  foreach (NoteInfo note in notes)  {  \_allnoteCollection.Add(note);  }  }  #endregion  private void monthlyViewTC\_click(object sender, RoutedEventArgs e)  {  List<NoteInfo> months = DbInteraction.searchMnthlyNoteList( goToDateDP.SelectedDate.Value);  \_allnoteCollection.Clear();  foreach (NoteInfo note in months)  {  \_allnoteCollection.Add(note);  }  }  }  } |

|  |
| --- |
| FacebookLoginDialog.cs |
| namespace fb\_csharp\_sdk\_winforms  {  using System;  using System.Dynamic;  using System.Windows.Forms;  using Facebook;  public partial class FacebookLoginDialog : Form  {  private readonly Uri \_loginUrl;  protected FacebookClient \_fb;  public FacebookOAuthResult FacebookOAuthResult { get; private set; }  public FacebookLoginDialog(string appId, string extendedPermissions)  : this(new FacebookClient(), appId, extendedPermissions)  {  }  public FacebookLoginDialog(FacebookClient fb, string appId, string extendedPermissions)  {  if (fb == null)  throw new ArgumentNullException("fb");  if (string.IsNullOrWhiteSpace(appId))  throw new ArgumentNullException("appId");  \_fb = fb;  \_loginUrl = GenerateLoginUrl(appId, extendedPermissions);  InitializeComponent();  }  private Uri GenerateLoginUrl(string appId, string extendedPermissions)  {  dynamic parameters = new ExpandoObject();  parameters.client\_id = appId;  parameters.redirect\_uri = "https://www.facebook.com/connect/login\_success.html";  // The requested response: an access token (token), an authorization code (code), or both (code token).  parameters.response\_type = "token";  // list of additional display modes can be found at http://developers.facebook.com/docs/reference/dialogs/#display  parameters.display = "popup";  // add the 'scope' parameter only if we have extendedPermissions.  if (!string.IsNullOrWhiteSpace(extendedPermissions))  parameters.scope = extendedPermissions;  // when the Form is loaded navigate to the login url.  return \_fb.GetLoginUrl(parameters);  }  private void FacebookLoginDialog\_Load(object sender, EventArgs e)  {  // make sure to use AbsoluteUri.  webBrowser1.Navigate(\_loginUrl.AbsoluteUri);  }  private void webBrowser1\_Navigated(object sender, WebBrowserNavigatedEventArgs e)  {  // whenever the browser navigates to a new url, try parsing the url.  // the url may be the result of OAuth 2.0 authentication.  FacebookOAuthResult oauthResult;  if (\_fb.TryParseOAuthCallbackUrl(e.Url, out oauthResult))  {  // The url is the result of OAuth 2.0 authentication  FacebookOAuthResult = oauthResult;  DialogResult = FacebookOAuthResult.IsSuccess ? DialogResult.OK : DialogResult.No;  }  else  {  // The url is NOT the result of OAuth 2.0 authentication.  FacebookOAuthResult = null;  }  }  }  } |

|  |
| --- |
| FacebookLoginDialog.Designer.cs |
| namespace fb\_csharp\_sdk\_winforms  {  partial class FacebookLoginDialog  {  /// <summary>  /// Required designer variable.  /// </summary>  private System.ComponentModel.IContainer components = null;  /// <summary>  /// Clean up any resources being used.  /// </summary>  /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>  protected override void Dispose(bool disposing)  {  if (disposing && (components != null))  {  components.Dispose();  }  base.Dispose(disposing);  }  #region Windows Form Designer generated code  /// <summary>  /// Required method for Designer support - do not modify  /// the contents of this method with the code editor.  /// </summary>  private void InitializeComponent()  {  this.webBrowser1 = new System.Windows.Forms.WebBrowser();  this.SuspendLayout();  //  // webBrowser1  //  this.webBrowser1.Dock = System.Windows.Forms.DockStyle.Fill;  this.webBrowser1.Location = new System.Drawing.Point(0, 0);  this.webBrowser1.MinimumSize = new System.Drawing.Size(20, 20);  this.webBrowser1.Name = "webBrowser1";  this.webBrowser1.Size = new System.Drawing.Size(661, 314);  this.webBrowser1.TabIndex = 0;  this.webBrowser1.Navigated += new System.Windows.Forms.WebBrowserNavigatedEventHandler(this.webBrowser1\_Navigated);  //  // FacebookLoginDialog  //  this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);  this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;  this.ClientSize = new System.Drawing.Size(661, 314);  this.Controls.Add(this.webBrowser1);  this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;  this.MaximizeBox = false;  this.MinimizeBox = false;  this.Name = "FacebookLoginDialog";  this.Text = "Login to Facebook";  this.Load += new System.EventHandler(this.FacebookLoginDialog\_Load);  this.ResumeLayout(false);  }  #endregion  private System.Windows.Forms.WebBrowser webBrowser1;  }  } |

|  |
| --- |
| MainForm.cs |
| namespace fb\_csharp\_sdk\_winforms  {  using System;  using System.Windows.Forms;  using Facebook;  public partial class MainForm : Form  {  private const string AppId = "182467085133744";  /// <summary>  /// Extended permissions is a comma separated list of permissions to ask the user.  /// </summary>  /// <remarks>  /// For extensive list of available extended permissions refer to  /// https://developers.facebook.com/docs/reference/api/permissions/  /// </remarks>  private const string ExtendedPermissions = "user\_about\_me,publish\_stream";  public MainForm()  {  InitializeComponent();  }  private void btnLogin\_Click(object sender, EventArgs e)  {  // open the Facebook Login Dialog and ask for user permissions.  var fbLoginDlg = new FacebookLoginDialog(AppId, ExtendedPermissions);  fbLoginDlg.ShowDialog();  // The user has taken action, either allowed/denied or cancelled the authorization,  // which can be known by looking at the dialogs FacebookOAuthResult property.  // Depending on the result take appropriate actions.  TakeLoggedInAction(fbLoginDlg.FacebookOAuthResult);  }  private void TakeLoggedInAction(FacebookOAuthResult facebookOAuthResult)  {  if (facebookOAuthResult == null)  {  // the user closed the FacebookLoginDialog, so do nothing.  MessageBox.Show("Cancelled!");  return;  }  // Even though facebookOAuthResult is not null, it could had been an  // OAuth 2.0 error, so make sure to check IsSuccess property always.  if (facebookOAuthResult.IsSuccess)  {  // since our respone\_type in FacebookLoginDialog was token,  // we got the access\_token  // The user now has successfully granted permission to our app.  var dlg = new InfoDialog(facebookOAuthResult.AccessToken);  dlg.ShowDialog();  }  else  {  // for some reason we failed to get the access token.  // most likely the user clicked don't allow.  MessageBox.Show(facebookOAuthResult.ErrorDescription);  }  }  }  } |

|  |
| --- |
| MainForm .Designer.cs |
| namespace fb\_csharp\_sdk\_winforms  {  partial class MainForm  {  /// <summary>  /// Required designer variable.  /// </summary>  private System.ComponentModel.IContainer components = null;  /// <summary>  /// Clean up any resources being used.  /// </summary>  /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>  protected override void Dispose(bool disposing)  {  if (disposing && (components != null))  {  components.Dispose();  }  base.Dispose(disposing);  }  #region Windows Form Designer generated code  /// <summary>  /// Required method for Designer support - do not modify  /// the contents of this method with the code editor.  /// </summary>  private void InitializeComponent()  {  this.btnLogin = new System.Windows.Forms.Button();  this.SuspendLayout();  //  // btnLogin  //  this.btnLogin.Location = new System.Drawing.Point(111, 75);  this.btnLogin.Name = "btnLogin";  this.btnLogin.Size = new System.Drawing.Size(200, 41);  this.btnLogin.TabIndex = 0;  this.btnLogin.Text = "Login to Facebook";  this.btnLogin.UseVisualStyleBackColor = true;  this.btnLogin.Click += new System.EventHandler(this.btnLogin\_Click);  //  // MainForm  //  this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);  this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;  this.ClientSize = new System.Drawing.Size(417, 186);  this.Controls.Add(this.btnLogin);  this.Name = "MainForm";  this.Text = "Login to Facebook";  this.ResumeLayout(false);  }  #endregion  private System.Windows.Forms.Button btnLogin;  }  } |

|  |
| --- |
| Program.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Windows.Forms;  namespace fb\_csharp\_sdk\_winforms  {  static class Program  {  /// <summary>  /// The main entry point for the application.  /// </summary>  [STAThread]  static void Main()  {  Application.EnableVisualStyles();  Application.SetCompatibleTextRenderingDefault(false);  Application.Run(new MainForm());  }  }  } |

|  |
| --- |
| InfoDialog.cs |
| namespace fb\_csharp\_sdk\_winforms  {  using System;  using System.Collections.Generic;  using System.Dynamic;  using System.IO;  using System.Threading.Tasks;  using System.Windows.Forms;  using Facebook;  public partial class InfoDialog : Form  {  private readonly TaskScheduler \_ui;  private readonly string \_accessToken;  public InfoDialog(string accessToken)  {  \_accessToken = accessToken;  \_ui = TaskScheduler.FromCurrentSynchronizationContext();  InitializeComponent();  }  private void btnLogout\_Click(object sender, EventArgs e)  {  var fb = new FacebookClient();  var logoutUrl = fb.GetLogoutUrl(new  {  next = "https://www.facebook.com/connect/login\_success.html",  access\_token = \_accessToken  });  var webBrowser = new WebBrowser();  webBrowser.Navigated += (o, args) =>  {  if (args.Url.AbsoluteUri == "https://www.facebook.com/connect/login\_success.html")  Close();  };  webBrowser.Navigate(logoutUrl.AbsoluteUri);  }  private void InfoDialog\_Load(object sender, EventArgs e)  {  GetUserProfilePicture();  GraphApiExample();  GraphApiAsyncExample();  GraphApiAsyncDynamicExample();  GraphApiParametersInPathExample();  FqlAsyncExample();  FqlMultiQueryAsyncExample();  BatchRequestExample();  LegacyRestApiAsyncExample();  }  private void GetUserProfilePicture()  {  // note: avoid using synchronous methods if possible as it will block the thread until the result is received  try  {  var fb = new FacebookClient(\_accessToken);  // Note: the result can either me IDictionary<string,object> or IList<object>  // json objects with properties can be casted to IDictionary<string,object> or IDictionary<string,dynamic>  // json arrays can be casted to IList<object> or IList<dynamic>  // for this particular request we can guarantee that the result is  // always IDictionary<string,object>.  var result = (IDictionary<string, object>)fb.Get("me");  // make sure to cast the object to appropriate type  var id = (string)result["id"];  // FacebookClient's Get/Post/Delete methods only supports JSON response results.  // For non json results, you will need to use different mechanism,  // here is an example for pictures.  // available picture types: square (50x50), small (50xvariable height), large (about 200x variable height) (all size in pixels)  // for more info visit http://developers.facebook.com/docs/reference/api  string profilePictureUrl = string.Format("https://graph.facebook.com/{0}/picture?type={1}", id, "square");  picProfile.LoadAsync(profilePictureUrl);  }  catch (FacebookApiException ex)  {  MessageBox.Show(ex.Message);  }  }  private void GraphApiExample()  {  // note: avoid using synchronous methods if possible as it will block the thread until the result is received  try  {  var fb = new FacebookClient(\_accessToken);  // instead of casting to IDictionary<string,object> or IList<object>  // you can also make use of the dynamic keyword.  dynamic result = fb.Get("me");  // You can either access it this way, using the .  dynamic id = result.id;  dynamic name = result.name;  // if dynamic you don't need to cast explicitly.  lblUserId.Text = "User Id: " + id;  lnkName.Text = "Hi " + name;  lnkName.LinkClicked += (o, e) => System.Diagnostics.Process.Start(result.link);  // or using the indexer  dynamic firstName = result["first\_name"];  dynamic lastName = result["last\_name"];  // checking if property exist  var localeExists = result.ContainsKey("locale");  // you can also cast it to IDictionary<string,object> and then check  var dictionary = (IDictionary<string, object>)result;  localeExists = dictionary.ContainsKey("locale");  }  catch (FacebookApiException ex)  {  MessageBox.Show(ex.Message);  }  }  private void GraphApiAsyncExample()  {  // avoid using XAsync methods as it is marked obsolete.  // use XAsync only for .net 3.5/SL4/WP7  // use XTaskAsync methods instead.  var fb = new FacebookClient(\_accessToken);  // make sure to add the appropriate event handler  // before calling the async methods.  // GetCompleted => GetAsync  // PostCompleted => PostAsync  // DeleteCompleted => DeleteAsync  fb.GetCompleted += (o, e) =>  {  // incase you support cancellation, make sure to check  // e.Cancelled property first even before checking (e.Error != null).  if (e.Cancelled)  {  // for this example, we can ignore as we don't allow this  // example to be cancelled.  }  else if (e.Error != null)  {  // error occurred  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show(e.Error.Message);  }));  }  else  {  // the request was completed successfully  // now we can either cast it to IDictionary<string, object> or IList<object>  // depending on the type.  // For this example, we know that it is IDictionary<string,object>.  var result = (IDictionary<string, object>)e.GetResultData();  var firstName = (string)result["first\_name"];  var lastName = (string)result["last\_name"];  // since this is an async callback, make sure to be on the right thread  // when working with the UI.  this.BeginInvoke(new MethodInvoker(  () =>  {  lblFirstName.Text = "First Name: " + firstName;  }));  }  };  // additional parameters can be passed and  // must be assignable from IDictionary<string, object> or anonymous object  var parameters = new Dictionary<string, object>();  parameters["fields"] = "first\_name,last\_name";  fb.GetAsync("me", parameters);  // or  //fb.GetAsync("me", new { fields = new[] { "first\_name", "last\_name" } });  }  private void GraphApiAsyncDynamicExample()  {  var fb = new FacebookClient(\_accessToken);  // make sure to add the appropriate event handler  // before calling the async methods.  // GetCompleted => GetAsync  // PostCompleted => PostAsync  // DeleteCompleted => DeleteAsync  fb.GetCompleted += (o, e) =>  {  // incase you support cancellation, make sure to check  // e.Cancelled property first even before checking (e.Error != null).  if (e.Cancelled)  {  // for this example, we can ignore as we don't allow this  // example to be cancelled.  }  else if (e.Error != null)  {  // error occurred  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show(e.Error.Message);  }));  }  else  {  // the request was completed successfully  // now we can either cast it to IDictionary<string, object> or IList<object>  // depending on the type. or we could use dynamic.  dynamic result = e.GetResultData();  // you can use either the .  var firstName = result.first\_name;  // or you can use indexer.  var lastName = result["last\_name"];  // since this is an async callback, make sure to be on the right thread  // when working with the UI.  this.BeginInvoke(new MethodInvoker(  () =>  {  lblLastName.Text = "Last Name: " + lastName;  }));  }  };  // additional parameters can be passed and  // must be assignable from IDictionary<string, object> or anonymous objects  // You can use ExpandoObject if you want to use dynamic  dynamic parameters = new ExpandoObject();  parameters.fields = "first\_name,last\_name";  fb.GetAsync("me", parameters);  // or  //fb.GetAsync("me", new { fields = new[] { "first\_name", "last\_name" } });  }  private void GraphApiParametersInPathExample()  {  // rather then creating a new object for parameter  // you can also embed simple parameters as part of the path.  try  {  var fb = new FacebookClient(\_accessToken);  dynamic result = fb.Get("me?fields=first\_name,last\_name");  dynamic firstName = result.first\_name;  dynamic lastName = result.last\_name;  //// this is especially useful for paged data (result.paging.next and result.paging.previous)  //// and your path can also contain the full graph url (https://graph.facebook.com/"  //var nextPath = "https://graph.facebook.com/me/likes?limit=3&access\_token=xxxxxxxxxxx&offset=3";  //dynamic nextResult = fb.Get(nextPath);  }  catch (FacebookApiException ex)  {  MessageBox.Show(ex.Message);  }  }  private void FqlAsyncExample()  {  var fb = new FacebookClient(\_accessToken);  // since FQL is internally a GET request,  // make sure to add the GET event handler.  fb.GetCompleted += (o, e) =>  {  // incase you support cancellation, make sure to check  // e.Cancelled property first even before checking (e.Error!=null).  if (e.Cancelled)  {  // for this example, we can ignore as we don't allow this  // example to be cancelled.  // you can check e.Error for reasons behind the cancellation.  var cancellationError = e.Error;  }  else if (e.Error != null)  {  // error occurred  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show(e.Error.Message);  }));  }  else  {  // the request was completed successfully  // now we can either cast it to IDictionary<string, object> or IList<object>  // depending on the type. or we could use dynamic.  var result = (IDictionary<string, object>)e.GetResultData();  var data = (IList<object>)result["data"];  var count = data.Count;  // since this is an async callback, make sure to be on the right thread  // when working with the UI.  this.BeginInvoke(new MethodInvoker(  () =>  {  lblTotalFriends.Text = string.Format("You have {0} friend(s).", count);  }));  }  };  // query to get all the friends  var query = string.Format("SELECT uid,pic\_square FROM user WHERE uid IN (SELECT uid2 FROM friend WHERE uid1={0})", "me()");  // call the Query or QueryAsync method to execute a single fql query.  fb.GetAsync("fql", new { q = query });  }  private void FqlMultiQueryAsyncExample()  {  var fb = new FacebookClient(\_accessToken);  // since FQL multi-query is internally a GET request,  // make sure to add the GET event handler.  fb.GetCompleted += (o, e) =>  {  // incase you support cancellation, make sure to check  // e.Cancelled property first even before checking (e.Error!=null).  if (e.Cancelled)  {  // for this example, we can ignore as we don't allow this  // example to be cancelled.  // you can check e.Error for reasons behind the cancellation.  var cancellationError = e.Error;  }  else if (e.Error != null)  {  // error occurred  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show(e.Error.Message);  }));  }  else  {  // the request was completed successfully  // now we can either cast it to IDictionary<string, object> or IList<object>  // depending on the type. or we could use dynamic.  dynamic result = e.GetResultData();  dynamic resultForQuery1 = result.data[0].fql\_result\_set;  dynamic resultForQuery2 = result.data[1].fql\_result\_set;  var uid = resultForQuery1[0].uid;  this.BeginInvoke(new MethodInvoker(  () =>  {  // make sure to be on the right thread when working with ui.  }));  }  };  var query1 = "SELECT uid FROM user WHERE uid=me()";  var query2 = "SELECT profile\_url FROM user WHERE uid=me()";  // call the Query or QueryAsync method to execute a single fql query.  // if there is more than one query Query/QueryAsync method will automatically  // treat it as multi-query.  fb.GetAsync("fql", new { q = new[] { query1, query2 } });  }  private void BatchRequestExample()  {  try  {  var fb = new FacebookClient(\_accessToken);  dynamic result = fb.Batch(  new FacebookBatchParameter { HttpMethod = HttpMethod.Get, Path = "/4" },  new FacebookBatchParameter(HttpMethod.Get, "/me/friend", new Dictionary<string, object> { { "limit", 10 } }), // this should throw error  new FacebookBatchParameter("/me/friends", new { limit = 1 }) { Data = new { name = "one-friend", omit\_response\_on\_success = false } }, // use Data to add additional parameters that doesn't exist  new FacebookBatchParameter { Parameters = new { ids = "{result=one-friend:$.data.0.id}" } },  new FacebookBatchParameter("{result=one-friend:$.data.0.id}/feed", new { limit = 5 }),  new FacebookBatchParameter("fql", new { q = "SELECT name FROM user WHERE uid=" }), // fql  new FacebookBatchParameter("fql", new { q = new[] { "SELECT first\_name FROM user WHERE uid=me()", "SELECT last\_name FROM user WHERE uid=me()" } }) // fql multi-query  //,new FacebookBatchParameter(HttpMethod.Post, "/me/feed", new { message = "test status update" })  );  // always remember to check individual errors for the batch requests.  if (result[0] is Exception)  MessageBox.Show(((Exception)result[0]).Message);  dynamic first = result[0];  string name = first.name;  // note: incase the omit\_response\_on\_success = true, result[x] == null  // for this example, just comment it out.  //if (result[1] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[2] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[3] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[4] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[5] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[6] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[7] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  }  catch (FacebookApiException ex)  {  MessageBox.Show(ex.Message);  }  }  private void BatchRequestAsyncExample()  {  var fb = new FacebookClient(\_accessToken);  // since batch request is actually a POST request internally,  // make sure to add the event handler for PostCompleted.  fb.PostCompleted += (o, e) =>  {  // incase you support cancellation, make sure to check  // e.Cancelled property first even before checking (e.Error!=null).  if (e.Cancelled)  {  // for this example, we can ignore as we don't allow this  // example to be cancelled.  // you can check e.Error for reasons behind the cancellation.  var cancellationError = e.Error;  }  else if (e.Error != null)  {  // error occurred  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show(e.Error.Message);  }));  }  else  {  // the request was completed successfully  // now we can either cast it to IDictionary<string, object> or IList<object>  // depending on the type. or we could use dynamic.  dynamic result = e.GetResultData();  // make sure to be on the right thread when working with ui.  this.BeginInvoke(new MethodInvoker(  () =>  {  // always remember to check individual errors for the batch requests.  if (result[0] is Exception)  MessageBox.Show(((Exception)result[0]).Message);  dynamic first = result[0];  string name = first.name;  // note: incase the omit\_response\_on\_success = true, result[x] == null  // for this example just comment it out  //if (result[1] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[2] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[3] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[4] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[5] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[6] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  //if (result[7] is Exception)  // MessageBox.Show(((Exception)result[1]).Message);  }));  }  };  fb.BatchAsync(new[]{  new FacebookBatchParameter { HttpMethod = HttpMethod.Get, Path = "/4" },  new FacebookBatchParameter(HttpMethod.Get, "/me/friend", new Dictionary<string, object> { { "limit", 10 } }), // this should throw error  new FacebookBatchParameter("/me/friends", new { limit = 1 }) { Data = new { name = "one-friend", omit\_response\_on\_success = false } }, // use Data to add additional parameters that doesn't exist  new FacebookBatchParameter { Parameters = new { ids = "{result=one-friend:$.data.0.id}" } },  new FacebookBatchParameter("{result=one-friend:$.data.0.id}/feed", new { limit = 5 }),  new FacebookBatchParameter("fql", new { q = "SELECT name FROM user WHERE uid=" }), // fql  new FacebookBatchParameter("fql", new { q = new[] { "SELECT first\_name FROM user WHERE uid=me()", "SELECT last\_name FROM user WHERE uid=me()" } }) // fql multi-query  //,new FacebookBatchParameter(HttpMethod.Post, "/me/feed", new { message = "test status update" })  });  }  private void LegacyRestApiAsyncExample()  {  var fb = new FacebookClient(\_accessToken);  // make sure to add the appropriate event handler  // before calling the async methods.  // GetCompleted => GetAsync  // PostCompleted => PostAsync  // DeleteCompleted => DeleteAsync  fb.GetCompleted += (o, e) =>  {  // incase you support cancellation, make sure to check  // e.Cancelled property first even before checking (e.Error != null).  if (e.Cancelled)  {  // for this example, we can ignore as we don't allow this  // example to be cancelled.  }  else if (e.Error != null)  {  // error occurred  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show(e.Error.Message);  }));  }  else  {  // the request was completed successfully  // now we can either cast it to IDictionary<string, object> or IList<object>  // depending on the type. or we could use dynamic.  dynamic result = e.GetResultData();  // since this is an async callback, make sure to be on the right thread  // when working with the UI.  this.BeginInvoke(new MethodInvoker(  () =>  {  chkCSharpSdkFan.Checked = result;  }));  }  };  //dynamic parameters = new ExpandoObject();  //// any parameter that has "method" automatically is treated as rest api.  //parameters.method = "pages.isFan";  //parameters.page\_id = "162171137156411"; // id of http://www.facebook.com/csharpsdk official page  //// for rest api only, parameters is enough  //// the rest method is determined by parameters.method  //fb.GetAsync(parameters);  fb.GetAsync(new { method = "pages.isFan", page\_id = "162171137156411" });  }  private string \_lastMessageId;  private void btnPostToWall\_Click(object sender, EventArgs args)  {  var fb = new FacebookClient(\_accessToken);  // make sure to add event handler for PostCompleted.  fb.PostCompleted += (o, e) =>  {  // incase you support cancellation, make sure to check  // e.Cancelled property first even before checking (e.Error!=null).  if (e.Cancelled)  {  // for this example, we can ignore as we don't allow this  // example to be cancelled.  // you can check e.Error for reasons behind the cancellation.  var cancellationError = e.Error;  }  else if (e.Error != null)  {  // error occurred  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show(e.Error.Message);  }));  }  else  {  // the request was completed successfully  // now we can either cast it to IDictionary<string, object> or IList<object>  // depending on the type. or we could use dynamic.  dynamic result = e.GetResultData();  \_lastMessageId = result.id;  // make sure to be on the right thread when working with ui.  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show("Message Posted successfully");  txtMessage.Text = string.Empty;  btnDeleteLastMessage.Enabled = true;  }));  }  };  dynamic parameters = new ExpandoObject();  parameters.message = txtMessage.Text;  fb.PostAsync("me/feed", parameters);  }  private void btnDeleteLastMessage\_Click(object sender, EventArgs args)  {  btnDeleteLastMessage.Enabled = false;  var fb = new FacebookClient(\_accessToken);  // make sure to add event handler for DeleteCompleted.  fb.DeleteCompleted += (o, e) =>  {  // incase you support cancellation, make sure to check  // e.Cancelled property first even before checking (e.Error!=null).  if (e.Cancelled)  {  // for this example, we can ignore as we don't allow this  // example to be cancelled.  // you can check e.Error for reasons behind the cancellation.  var cancellationError = e.Error;  }  else if (e.Error != null)  {  // error occurred  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show(e.Error.Message);  }));  }  else  {  // the request was completed successfully  // make sure to be on the right thread when working with ui.  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show("Message deleted successfully");  btnDeleteLastMessage.Enabled = false;  }));  }  };  fb.DeleteAsync(\_lastMessageId);  }  private void bntPostPicture\_Click(object sender, EventArgs args)  {  var ofd = new OpenFileDialog  {  Filter = "JPEG Files|\*.jpg",  Title = "Select picture to upload"  };  if (ofd.ShowDialog() != DialogResult.OK)  {  return;  }  var fb = new FacebookClient(\_accessToken);  // make sure to add event handler for PostCompleted.  fb.PostCompleted += (o, e) =>  {  // incase you support cancellation, make sure to check  // e.Cancelled property first even before checking (e.Error!=null).  if (e.Cancelled)  {  // for this example, we can ignore as we don't allow this  // example to be cancelled.  // you can check e.Error for reasons behind the cancellation.  var cancellationError = e.Error;  }  else if (e.Error != null)  {  // error occurred  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show(e.Error.Message);  }));  }  else  {  // the request was completed successfully  // make sure to be on the right thread when working with ui.  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show("Picture uploaded successfully");  }));  }  };  dynamic parameters = new ExpandoObject();  parameters.message = txtMessage.Text;  parameters.source = new FacebookMediaObject  {  ContentType = "image/jpeg",  FileName = Path.GetFileName(ofd.FileName)  }.SetValue(File.ReadAllBytes(ofd.FileName));  fb.PostAsync("me/photos", parameters);  }  private void btnPostVideo\_Click(object sender, EventArgs args)  {  var ofd = new OpenFileDialog  {  Filter = "MP4 Files|\*.mp4",  Title = "Select video to upload"  };  if (ofd.ShowDialog() != DialogResult.OK)  {  return;  }  var fb = new FacebookClient(\_accessToken);  var attachment = new FacebookMediaStream  {  ContentType = "video/mp4",  FileName = Path.GetFileName(ofd.FileName)  }.SetValue(File.OpenRead(ofd.FileName));  // make sure to add event handler for PostCompleted.  fb.PostCompleted += (o, e) =>  {  attachment.Dispose();  // incase you support cancellation, make sure to check  // e.Cancelled property first even before checking (e.Error!=null).  if (e.Cancelled)  {  // for this example, we can ignore as we don't allow this  // example to be cancelled.  // you can check e.Error for reasons behind the cancellation.  var cancellationError = e.Error;  }  else if (e.Error != null)  {  // error occurred  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show(e.Error.Message);  }));  }  else  {  // the request was completed successfully  // make sure to be on the right thread when working with ui.  this.BeginInvoke(new MethodInvoker(  () =>  {  MessageBox.Show("Video uploaded successfully");  }));  }  };  dynamic parameters = new ExpandoObject();  parameters.message = txtMessage.Text;  parameters.source = attachment;  fb.PostAsync("me/videos", parameters);  }  private void btnProgressAndCancellation\_Click(object sender, EventArgs e)  {  var dlg = new UploadProgressCancelForm(\_accessToken);  dlg.ShowDialog();  }  }  } |

|  |
| --- |
| InfoDialog.Designer.cs |
| namespace fb\_csharp\_sdk\_winforms  {  partial class InfoDialog  {  /// <summary>  /// Required designer variable.  /// </summary>  private System.ComponentModel.IContainer components = null;  /// <summary>  /// Clean up any resources being used.  /// </summary>  /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>  protected override void Dispose(bool disposing)  {  if (disposing && (components != null))  {  components.Dispose();  }  base.Dispose(disposing);  }  #region Windows Form Designer generated code  /// <summary>  /// Required method for Designer support - do not modify  /// the contents of this method with the code editor.  /// </summary>  private void InitializeComponent()  {  this.btnProgressAndCancellation = new System.Windows.Forms.Button();  this.lnkFacebokSdkFan = new System.Windows.Forms.LinkLabel();  this.btnDeleteLastMessage = new System.Windows.Forms.Button();  this.btnPostVideo = new System.Windows.Forms.Button();  this.bntPostPicture = new System.Windows.Forms.Button();  this.btnPostToWall = new System.Windows.Forms.Button();  this.label1 = new System.Windows.Forms.Label();  this.txtMessage = new System.Windows.Forms.TextBox();  this.lblTotalFriends = new System.Windows.Forms.Label();  this.chkCSharpSdkFan = new System.Windows.Forms.CheckBox();  this.lblLastName = new System.Windows.Forms.Label();  this.lblFirstName = new System.Windows.Forms.Label();  this.lblUserId = new System.Windows.Forms.Label();  this.lnkName = new System.Windows.Forms.LinkLabel();  this.picProfile = new System.Windows.Forms.PictureBox();  this.btnLogout = new System.Windows.Forms.Button();  ((System.ComponentModel.ISupportInitialize)(this.picProfile)).BeginInit();  this.SuspendLayout();  //  // btnProgressAndCancellation  //  this.btnProgressAndCancellation.Location = new System.Drawing.Point(167, 340);  this.btnProgressAndCancellation.Name = "btnProgressAndCancellation";  this.btnProgressAndCancellation.Size = new System.Drawing.Size(178, 23);  this.btnProgressAndCancellation.TabIndex = 29;  this.btnProgressAndCancellation.Text = "Upload Progress and Cancellation Sample";  this.btnProgressAndCancellation.UseVisualStyleBackColor = true;  this.btnProgressAndCancellation.Click += new System.EventHandler(this.btnProgressAndCancellation\_Click);  //  // lnkFacebokSdkFan  //  this.lnkFacebokSdkFan.AutoSize = true;  this.lnkFacebokSdkFan.Location = new System.Drawing.Point(79, 175);  this.lnkFacebokSdkFan.Name = "lnkFacebokSdkFan";  this.lnkFacebokSdkFan.Size = new System.Drawing.Size(200, 13);  this.lnkFacebokSdkFan.TabIndex = 28;  this.lnkFacebokSdkFan.TabStop = true;  this.lnkFacebokSdkFan.Text = "Like us on Facebook at our official page.";  this.lnkFacebokSdkFan.Visible = false;  //  // btnDeleteLastMessage  //  this.btnDeleteLastMessage.Enabled = false;  this.btnDeleteLastMessage.Location = new System.Drawing.Point(252, 276);  this.btnDeleteLastMessage.Name = "btnDeleteLastMessage";  this.btnDeleteLastMessage.Size = new System.Drawing.Size(131, 23);  this.btnDeleteLastMessage.TabIndex = 27;  this.btnDeleteLastMessage.Text = "Delete Last Message";  this.btnDeleteLastMessage.UseVisualStyleBackColor = true;  this.btnDeleteLastMessage.Click += new System.EventHandler(this.btnDeleteLastMessage\_Click);  //  // btnPostVideo  //  this.btnPostVideo.Location = new System.Drawing.Point(252, 311);  this.btnPostVideo.Name = "btnPostVideo";  this.btnPostVideo.Size = new System.Drawing.Size(103, 23);  this.btnPostVideo.TabIndex = 26;  this.btnPostVideo.Text = "Post Video";  this.btnPostVideo.UseVisualStyleBackColor = true;  this.btnPostVideo.Click += new System.EventHandler(this.btnPostVideo\_Click);  //  // bntPostPicture  //  this.bntPostPicture.Location = new System.Drawing.Point(143, 311);  this.bntPostPicture.Name = "bntPostPicture";  this.bntPostPicture.Size = new System.Drawing.Size(103, 23);  this.bntPostPicture.TabIndex = 25;  this.bntPostPicture.Text = "Post Picture";  this.bntPostPicture.UseVisualStyleBackColor = true;  this.bntPostPicture.Click += new System.EventHandler(this.bntPostPicture\_Click);  //  // btnPostToWall  //  this.btnPostToWall.Location = new System.Drawing.Point(115, 276);  this.btnPostToWall.Name = "btnPostToWall";  this.btnPostToWall.Size = new System.Drawing.Size(131, 23);  this.btnPostToWall.TabIndex = 24;  this.btnPostToWall.Text = "Post To Wall";  this.btnPostToWall.UseVisualStyleBackColor = true;  this.btnPostToWall.Click += new System.EventHandler(this.btnPostToWall\_Click);  //  // label1  //  this.label1.AutoSize = true;  this.label1.Location = new System.Drawing.Point(26, 202);  this.label1.Name = "label1";  this.label1.Size = new System.Drawing.Size(50, 13);  this.label1.TabIndex = 23;  this.label1.Text = "Message";  //  // txtMessage  //  this.txtMessage.Location = new System.Drawing.Point(82, 202);  this.txtMessage.Multiline = true;  this.txtMessage.Name = "txtMessage";  this.txtMessage.Size = new System.Drawing.Size(354, 62);  this.txtMessage.TabIndex = 22;  //  // lblTotalFriends  //  this.lblTotalFriends.AutoSize = true;  this.lblTotalFriends.Location = new System.Drawing.Point(79, 128);  this.lblTotalFriends.Name = "lblTotalFriends";  this.lblTotalFriends.Size = new System.Drawing.Size(81, 13);  this.lblTotalFriends.TabIndex = 21;  this.lblTotalFriends.Text = "[lblTotalFriends]";  //  // chkCSharpSdkFan  //  this.chkCSharpSdkFan.AutoSize = true;  this.chkCSharpSdkFan.Enabled = false;  this.chkCSharpSdkFan.Location = new System.Drawing.Point(82, 155);  this.chkCSharpSdkFan.Name = "chkCSharpSdkFan";  this.chkCSharpSdkFan.Size = new System.Drawing.Size(280, 17);  this.chkCSharpSdkFan.TabIndex = 20;  this.chkCSharpSdkFan.Text = "Is fan of the official Facebok C# SDK facebook page.";  this.chkCSharpSdkFan.UseVisualStyleBackColor = true;  //  // lblLastName  //  this.lblLastName.AutoSize = true;  this.lblLastName.Location = new System.Drawing.Point(79, 99);  this.lblLastName.Name = "lblLastName";  this.lblLastName.Size = new System.Drawing.Size(71, 13);  this.lblLastName.TabIndex = 19;  this.lblLastName.Text = "[lblLastName]";  //  // lblFirstName  //  this.lblFirstName.AutoSize = true;  this.lblFirstName.Location = new System.Drawing.Point(79, 75);  this.lblFirstName.Name = "lblFirstName";  this.lblFirstName.Size = new System.Drawing.Size(70, 13);  this.lblFirstName.TabIndex = 18;  this.lblFirstName.Text = "[lblFirstName]";  //  // lblUserId  //  this.lblUserId.AutoSize = true;  this.lblUserId.Location = new System.Drawing.Point(79, 49);  this.lblUserId.Name = "lblUserId";  this.lblUserId.Size = new System.Drawing.Size(54, 13);  this.lblUserId.TabIndex = 17;  this.lblUserId.Text = "[lblUserId]";  //  // lnkName  //  this.lnkName.AutoSize = true;  this.lnkName.Font = new System.Drawing.Font("Microsoft Sans Serif", 14.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));  this.lnkName.Location = new System.Drawing.Point(78, 12);  this.lnkName.Name = "lnkName";  this.lnkName.Size = new System.Drawing.Size(95, 24);  this.lnkName.TabIndex = 16;  this.lnkName.TabStop = true;  this.lnkName.Text = "[lnkName]";  //  // picProfile  //  this.picProfile.Location = new System.Drawing.Point(12, 12);  this.picProfile.Name = "picProfile";  this.picProfile.Size = new System.Drawing.Size(50, 50);  this.picProfile.TabIndex = 15;  this.picProfile.TabStop = false;  //  // btnLogout  //  this.btnLogout.Location = new System.Drawing.Point(361, 12);  this.btnLogout.Name = "btnLogout";  this.btnLogout.Size = new System.Drawing.Size(75, 23);  this.btnLogout.TabIndex = 30;  this.btnLogout.Text = "Logout";  this.btnLogout.UseVisualStyleBackColor = true;  this.btnLogout.Click += new System.EventHandler(this.btnLogout\_Click);  //  // InfoDialog  //  this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);  this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;  this.ClientSize = new System.Drawing.Size(448, 376);  this.Controls.Add(this.btnLogout);  this.Controls.Add(this.btnProgressAndCancellation);  this.Controls.Add(this.lnkFacebokSdkFan);  this.Controls.Add(this.btnDeleteLastMessage);  this.Controls.Add(this.btnPostVideo);  this.Controls.Add(this.bntPostPicture);  this.Controls.Add(this.btnPostToWall);  this.Controls.Add(this.label1);  this.Controls.Add(this.txtMessage);  this.Controls.Add(this.lblTotalFriends);  this.Controls.Add(this.chkCSharpSdkFan);  this.Controls.Add(this.lblLastName);  this.Controls.Add(this.lblFirstName);  this.Controls.Add(this.lblUserId);  this.Controls.Add(this.lnkName);  this.Controls.Add(this.picProfile);  this.MaximizeBox = false;  this.Name = "InfoDialog";  this.Text = "Facebook C# SDK WinForms sample";  this.Load += new System.EventHandler(this.InfoDialog\_Load);  ((System.ComponentModel.ISupportInitialize)(this.picProfile)).EndInit();  this.ResumeLayout(false);  this.PerformLayout();  }  #endregion  private System.Windows.Forms.Button btnProgressAndCancellation;  private System.Windows.Forms.LinkLabel lnkFacebokSdkFan;  private System.Windows.Forms.Button btnDeleteLastMessage;  private System.Windows.Forms.Button btnPostVideo;  private System.Windows.Forms.Button bntPostPicture;  private System.Windows.Forms.Button btnPostToWall;  private System.Windows.Forms.Label label1;  private System.Windows.Forms.TextBox txtMessage;  private System.Windows.Forms.Label lblTotalFriends;  private System.Windows.Forms.CheckBox chkCSharpSdkFan;  private System.Windows.Forms.Label lblLastName;  private System.Windows.Forms.Label lblFirstName;  private System.Windows.Forms.Label lblUserId;  private System.Windows.Forms.LinkLabel lnkName;  private System.Windows.Forms.PictureBox picProfile;  private System.Windows.Forms.Button btnLogout;  }  } |

|  |
| --- |
| UploadProgressCancelForm.cs |
| using System;  using System.Collections.Generic;  using System.Windows.Forms;  namespace fb\_csharp\_sdk\_winforms  {  using System.IO;  using Facebook;  public partial class UploadProgressCancelForm : Form  {  private readonly string \_accessToken;  private string \_filename;  private FacebookClient \_fb;  public UploadProgressCancelForm(string accessToken)  {  \_accessToken = accessToken;  InitializeComponent();  }  private void btnSelectFile\_Click(object sender, EventArgs e)  {  var ofd = new OpenFileDialog { CheckFileExists = true, Filter = "jpeg (\*.jpg) | \*.jpg" };  if (ofd.ShowDialog() == DialogResult.OK)  {  \_filename = ofd.FileName;  }  else  {  \_filename = null;  }  }  private void btnUpload\_Click(object sender, EventArgs e)  {  if (string.IsNullOrWhiteSpace(\_filename))  {  MessageBox.Show("Please select the image file first.");  return;  }  var mediaObject = new FacebookMediaObject  {  ContentType = "image/jpeg",  FileName = Path.GetFileName(\_filename)  }  .SetValue(File.ReadAllBytes(\_filename));  progressBar1.Value = 0;  var fb = new FacebookClient(\_accessToken);  fb.UploadProgressChanged += fb\_UploadProgressChanged;  fb.PostCompleted += fb\_PostCompleted;  // for cancellation  \_fb = fb;  fb.PostAsync("/me/photos", new Dictionary<string, object> { { "source", mediaObject } });  }  public void fb\_UploadProgressChanged(object sender, FacebookUploadProgressChangedEventArgs e)  {  progressBar1.BeginInvoke(new MethodInvoker(() =>  {  progressBar1.Value = e.ProgressPercentage;  }));  }  public void fb\_PostCompleted(object sender, FacebookApiEventArgs e)  {  if (e.Cancelled)  {  MessageBox.Show("Upload cancelled");  }  else if (e.Error == null)  {  // upload successful.  MessageBox.Show(e.GetResultData().ToString());  }  else  {  // upload failed  MessageBox.Show(e.Error.Message);  }  progressBar1.BeginInvoke(new MethodInvoker(() =>  {  progressBar1.Value = 0;  }));  }  private void btnCancel\_Click(object sender, EventArgs e)  {  if (\_fb != null)  {  \_fb.CancelAsync();  }  }  }  } |

|  |
| --- |
| UploadProgressCancelForm.Designer.cs |
| namespace fb\_csharp\_sdk\_winforms  {  partial class UploadProgressCancelForm  {  /// <summary>  /// Required designer variable.  /// </summary>  private System.ComponentModel.IContainer components = null;  /// <summary>  /// Clean up any resources being used.  /// </summary>  /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>  protected override void Dispose(bool disposing)  {  if (disposing && (components != null))  {  components.Dispose();  }  base.Dispose(disposing);  }  #region Windows Form Designer generated code  /// <summary>  /// Required method for Designer support - do not modify  /// the contents of this method with the code editor.  /// </summary>  private void InitializeComponent()  {  this.btnUpload = new System.Windows.Forms.Button();  this.btnCancel = new System.Windows.Forms.Button();  this.progressBar1 = new System.Windows.Forms.ProgressBar();  this.btnSelectFile = new System.Windows.Forms.Button();  this.SuspendLayout();  //  // btnUpload  //  this.btnUpload.Location = new System.Drawing.Point(126, 65);  this.btnUpload.Name = "btnUpload";  this.btnUpload.Size = new System.Drawing.Size(75, 23);  this.btnUpload.TabIndex = 11;  this.btnUpload.Text = "Start Upload";  this.btnUpload.UseVisualStyleBackColor = true;  this.btnUpload.Click += new System.EventHandler(this.btnUpload\_Click);  //  // btnCancel  //  this.btnCancel.Location = new System.Drawing.Point(225, 151);  this.btnCancel.Name = "btnCancel";  this.btnCancel.Size = new System.Drawing.Size(105, 23);  this.btnCancel.TabIndex = 10;  this.btnCancel.Text = "Cancel Upload";  this.btnCancel.UseVisualStyleBackColor = true;  this.btnCancel.Click += new System.EventHandler(this.btnCancel\_Click);  //  // progressBar1  //  this.progressBar1.Location = new System.Drawing.Point(20, 109);  this.progressBar1.Name = "progressBar1";  this.progressBar1.Size = new System.Drawing.Size(310, 23);  this.progressBar1.TabIndex = 9;  //  // btnSelectFile  //  this.btnSelectFile.Location = new System.Drawing.Point(78, 18);  this.btnSelectFile.Name = "btnSelectFile";  this.btnSelectFile.Size = new System.Drawing.Size(207, 23);  this.btnSelectFile.TabIndex = 8;  this.btnSelectFile.Text = "Select Jpeg Image File to Upload";  this.btnSelectFile.UseVisualStyleBackColor = true;  this.btnSelectFile.Click += new System.EventHandler(this.btnSelectFile\_Click);  //  // UploadProgressCancelForm  //  this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);  this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;  this.ClientSize = new System.Drawing.Size(352, 191);  this.Controls.Add(this.btnUpload);  this.Controls.Add(this.btnCancel);  this.Controls.Add(this.progressBar1);  this.Controls.Add(this.btnSelectFile);  this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;  this.MaximizeBox = false;  this.MinimizeBox = false;  this.Name = "UploadProgressCancelForm";  this.Text = "UploadProgressCancelForm";  this.ResumeLayout(false);  }  #endregion  private System.Windows.Forms.Button btnUpload;  private System.Windows.Forms.Button btnCancel;  private System.Windows.Forms.ProgressBar progressBar1;  private System.Windows.Forms.Button btnSelectFile;  }  } |

|  |
| --- |
| ShowEvent.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 **CMS**GUI  {  /// <summary>  /// Interaction logic for ShowEvent.xaml  /// </summary>  public partial class ShowEvent : UserControl  {  public ShowEvent(int no)  {  InitializeComponent();  sequenceLabel.Content = no;  }  }  } |

### Datbase Classes : CMSData

|  |
| --- |
| **AccessInfo** |
| public class AccessInfo  {  string Password;  string UserId;  bool IsRememberPassword;  int UniqueId;  string emailId;  } |

|  |
| --- |
| **BillingInfo** |
| public class BillingInfo  {  int billAmount;  int billingId;  Responsibility responsibilityName;  string status;  } |

|  |
| --- |
| **ConstructionInstruments** |
| public class ConstructionInstruments  {  string itemName;  string itemId;  List<string> itemTags;  string itemDescription;  ConstructionInstrumentsTypeEnum itemType;  List<WorkerInfo> itemUsers;  List<RawMaterialInfo> rawMaterialSupportedList;  } |

|  |
| --- |
| **ConstructionInstrumentsTypeEnum** |
| public enum ConstructionInstrumentsTypeEnum  {  Mixer,  Grinder,  Puller,  StairCase,  Basket  } |

|  |
| --- |
| **ConstructionSiteController** |
| public class ConstructionSiteController  {  List<WorkerInfo> employees;  List<RawMaterialInfo> resources;  List<RawMaterialInfo> admins;  List<SiteInfo> sites;  List<ConstructionInstruments> instruments;  List<BillingInfo> billingItems;  } |

|  |
| --- |
| DepartmentalInfo |
| public class DepartmentalInfo  {  string phoneNo;  string departmentName;  string departmentId;  string departmentMemberId;  string departmentEmailId;  } |

|  |
| --- |
| **DesignInfo** |
| public class DesignInfo  {  string name;  string softwareUsed;  WorkerInfo engineer;  string ImageFile;  double estimate;  string timeRequired;  }  } |

|  |
| --- |
| **ExpertiseInfo** |
| public class ExpertiseInfo  {  int yearsOfExperience;  string Qualifications;  string RegistrationId;  WokerTypeEnum expertiseDomain;  } |

|  |
| --- |
| **PersonalInfo** |
| public class PersonalInfo  {  string name;  string address;  string phoneNumber;  string sex;  string dob;  string emergencyContact;  } |

|  |
| --- |
| **RawMaterialInfo** |
| public class RawMaterialInfo  {  string itemName;  int itemId;  List<string> itemTags;  int itemDescription;  RawMaterialTypeEnum itemType;  List<WorkerInfo> itemUsers;  } |

|  |
| --- |
| **RawMaterialTypeEnum** |
| public enum RawMaterialTypeEnum  {  Brick,  Cement,  Rod,  Stone,  StoneChips,  Sand,  Color,  Wood  } |

|  |
| --- |
| **Responsibility** |
| public class Responsibility  {  string description;  int id;  List<ConstructionInstruments> relatedInstruments;  List<RawMaterialInfo> relatedRawmaterials;  List<WorkerInfo> relatedWorkers;  WorkerInfo releatedSupervisor;  } |

|  |
| --- |
| **SiteInfo** |
| public class SiteInfo  {  string name;  string location;  string customerName;  double valuation;  DesignInfo design;  List<Responsibility> responsibilities;  WorkerInfo superViser;  } |

|  |
| --- |
| **WokerTypeEnum** |
| public enum WokerTypeEnum  {  SiteManager,  Labour,  Helper,  Contructor,  Supplier,  Porter,  Engineer,  Waiter  } |

|  |
| --- |
|  |
| public class WorkerInfo  {  List<ConstructionInstruments> assginedItems;  DepartmentalInfo deptDetails;  PersonalInfo personalDetails;  AccessInfo accessDetails;  ExpertiseInfo expertise;  List<Responsibility> responsibilities;  } |

### Database connector: CMSDb

|  |
| --- |
| DbInteraction.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using **CMS**Data;  namespace **CMS**Db  {  public class DbInteraction  {  static string passwordCurrent = "technicise";  static string dbmsCurrent = "**CMS**db";  private static MySql.Data.MySqlClient.MySqlConnection OpenDbConnection()  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;  msqlConnection = new MySql.Data.MySqlClient.MySqlConnection("server=localhost;user id=root;Password=" + passwordCurrent + ";database=" + dbmsCurrent + ";persist security info=False");  //open the connection  if (msqlConnection.State != System.Data.ConnectionState.Open)  msqlConnection.Open();  return msqlConnection;  }  #region User  public static int DoRegisterNewUser(UserInfo NewUser)  {  return DoRegisterNewuserindb(NewUser);  }  private static int DoRegisterNewuserindb(UserInfo NewUser)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO user(id,userid,passwrd,hints) " + "VALUES(@id,@userid,@passwrd,@hints)";  msqlCommand.Parameters.AddWithValue("@id", NewUser.id);  msqlCommand.Parameters.AddWithValue("@userid", NewUser.userId);  msqlCommand.Parameters.AddWithValue("@passwrd", NewUser.pass);  msqlCommand.Parameters.AddWithValue("@hints", NewUser.hints);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  #endregion  #region Note  public static int DoEnterNewNote(NoteInfo NewNote)  {  return DoEnterNewNoteindb(NewNote);  }  private static int DoEnterNewNoteindb(NoteInfo NewNote)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO note(id,date,note) " + "VALUES(@id,@date,@note)";  msqlCommand.Parameters.AddWithValue("@id", NewNote.id);  msqlCommand.Parameters.AddWithValue("@date", NewNote.gotoDate);  msqlCommand.Parameters.AddWithValue("@note", NewNote.note);    msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<NoteInfo> GetAllNoteList()  {  return QueryAllNoteList();  }  private static List<NoteInfo> QueryAllNoteList()  {  List<NoteInfo> NoteList = new List<NoteInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From note ;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  NoteInfo Note = new NoteInfo();  //Note.id = msqlReader.GetString("id");  Note.noteDate = msqlReader.GetDateTime("date");  Note.note = msqlReader.GetString("note");  NoteList.Add(Note);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return NoteList;  }    #endregion  #region ID password  public static string FetcheId()  {  string idStr = string.Empty;  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select userid from user;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  msqlReader.Read();  idStr = msqlReader.GetString("userid");  }  catch (Exception er)  {  //Assert//.Show(er.Message);  }  finally  {  //always close the connection  msqlConnection.Close();  }  return idStr;  }  public static string FetchePassword()  {  string passwordStr = string.Empty;  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select passwrd from user;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  msqlReader.Read();  passwordStr = msqlReader.GetString("passwrd");  }  catch (Exception er)  {  //Assert//.Show(er.Message);  }  finally  {  //always close the connection  msqlConnection.Close();  }  return passwordStr;  }  #endregion  #region Contact  public static int DoRegisterNewContact(ContactInfo contactDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO contact(contactId,name,mobile,homePhone,officePhone,email,address,faxNumber,remark) "  + "VALUES(@contactId,@name,@mobile,@homePhone,@officePhone,@email,@address,@faxNumber,@remark)";  msqlCommand.Parameters.AddWithValue("@contactId", contactDetails.id);  msqlCommand.Parameters.AddWithValue("@name", contactDetails.name);  msqlCommand.Parameters.AddWithValue("@mobile", contactDetails.mobileno);  msqlCommand.Parameters.AddWithValue("@homePhone", contactDetails.homeno);  msqlCommand.Parameters.AddWithValue("@officePhone", contactDetails.oficeno);  msqlCommand.Parameters.AddWithValue("@email", contactDetails.email);  msqlCommand.Parameters.AddWithValue("@address", contactDetails.address);  msqlCommand.Parameters.AddWithValue("@faxNumber", contactDetails.faxno);  msqlCommand.Parameters.AddWithValue("@remark", contactDetails.remark);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<ContactInfo> GetAllContactList()  {  return QueryAllContactList();  }  private static List<ContactInfo> QueryAllContactList()  {  List<ContactInfo> ContactList = new List<ContactInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From contact ;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  ContactInfo Contact = new ContactInfo();  Contact.id = msqlReader.GetString("contactId");  Contact.name = msqlReader.GetString("name");  Contact.mobileno = msqlReader.GetString("mobile");  Contact.homeno = msqlReader.GetString("homePhone");  Contact.oficeno = msqlReader.GetString("officePhone");  Contact.email = msqlReader.GetString("email");  Contact.address = msqlReader.GetString("address");  Contact.faxno = msqlReader.GetString("faxNumber");  Contact.remark = msqlReader.GetString("remark");    ContactList.Add(Contact);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return ContactList;  }  #endregion  #region Task  public static int DoRegisterNewTask(TaskInfo TaskDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO task(taskId,priority,details) "  + "VALUES(@taskId,@priority,@details)";  msqlCommand.Parameters.AddWithValue("@taskId", TaskDetails.id);  msqlCommand.Parameters.AddWithValue("@priority", TaskDetails.value);  msqlCommand.Parameters.AddWithValue("@details", TaskDetails.taskDetails);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<TaskInfo> GetAllTaskList()  {  return QueryAllTaskList();  }  private static List<TaskInfo> QueryAllTaskList()  {  List<TaskInfo> TaskList = new List<TaskInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From task;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  TaskInfo Task = new TaskInfo();  Task.id = msqlReader.GetString("taskId");  Task.value = msqlReader.GetString("priority");  Task.taskDetails = msqlReader.GetString("details");    TaskList.Add(Task);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return TaskList;  }    #endregion  #region Password Manager  public static int DoRegisterNewPassword(PasswordInfo passwordDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO password(passwordId,name,email,userId,password,secretQuestion,secretAnswer,otherInfo) "  + "VALUES(@passwordId,@name,@email,@userId,@password,@secretQuestion,@secretAnswer,@otherInfo)";  msqlCommand.Parameters.AddWithValue("@passwordId", passwordDetails.id);  msqlCommand.Parameters.AddWithValue("@name", passwordDetails.name);  msqlCommand.Parameters.AddWithValue("@email", passwordDetails.email);  msqlCommand.Parameters.AddWithValue("@userId", passwordDetails.userId);  msqlCommand.Parameters.AddWithValue("@password", passwordDetails.password);  msqlCommand.Parameters.AddWithValue("@secretQuestion", passwordDetails.scrtqstn);  msqlCommand.Parameters.AddWithValue("@secretAnswer", passwordDetails.scrtans);  msqlCommand.Parameters.AddWithValue("@otherInfo", passwordDetails.otherInfo);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<PasswordInfo> GetAllPasswordsList()  {  return QueryAllPasswordList();  }  private static List<PasswordInfo> QueryAllPasswordList()  {  List<PasswordInfo> PasswordList = new List<PasswordInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From password ;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  PasswordInfo Password = new PasswordInfo();  Password.id = msqlReader.GetString("passwordId");  Password.name = msqlReader.GetString("name");  Password.email = msqlReader.GetString("email");  Password.userId = msqlReader.GetString("userId");  Password.password = msqlReader.GetString("password");  Password.scrtqstn = msqlReader.GetString("secretQuestion");  Password.scrtans = msqlReader.GetString("secretAnswer");  Password.otherInfo = msqlReader.GetString("otherInfo");    PasswordList.Add(Password);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return PasswordList;  }  #endregion  #region Go To Date / Search  public static List<NoteInfo> searchNoteList(NoteInfo noteinfo)  {  return searchAllNoteList(noteinfo);  }  private static List<NoteInfo> searchAllNoteList(NoteInfo noteinfo)  {  List<NoteInfo> NoteList = new List<NoteInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From note where date = @date ; ";  msqlCommand.Parameters.AddWithValue("@date", noteinfo.gotoDate);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  NoteInfo Note = new NoteInfo();  Note.noteDate = msqlReader.GetDateTime("date");    NoteList.Add(Note);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return NoteList;  }  #endregion  #region Monthly View  public static List<NoteInfo> searchMnthlyNoteList(DateTime date)  {  DateTime day1st = new DateTime(date.Year, date.Month, 1);  DateTime dayLast = new DateTime(date.Year, date.Month + 1, 1);  dayLast = dayLast.Subtract(new TimeSpan(1, 0, 0, 0));  List<NoteInfo> NoteList = new List<NoteInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "SELECT \* FROM note where date(note.date) >= DATE\_SUB( @dayLast, INTERVAL @diff DAY) group by date;";  msqlCommand.Parameters.AddWithValue("@dayLast", dayLast);  msqlCommand.Parameters.AddWithValue("@diff", dayLast.Subtract(day1st));  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  NoteInfo Note = new NoteInfo();  Note.noteDate = msqlReader.GetDateTime("date");  Note.note = msqlReader.GetString("note");  NoteList.Add(Note);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return NoteList;  }  #endregion    }  } |

## 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:
* //always close the connection
* //Note.id = msqlReader.GetString("id");
* //define the command reference
* Always applied C# comment-blocks (///) to public, protected, and internal declarations.
* Only used C# comment-blocks for documenting the API.
* Included #region and #endregion where possible for whole sections to have a #region-like thing and collapse them.
* Example:

|  |
| --- |
| #region User  public static int DoRegisterNewUser(UserInfo NewUser)  {  return DoRegisterNewuserindb(NewUser);  }  private static int DoRegisterNewuserindb(UserInfo NewUser)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO user(id,userid,passwrd,hints) " + "VALUES(@id,@userid,@passwrd,@hints)";  msqlCommand.Parameters.AddWithValue("@id", NewUser.id);  msqlCommand.Parameters.AddWithValue("@userid", NewUser.userId);  msqlCommand.Parameters.AddWithValue("@passwrd", NewUser.pass);  msqlCommand.Parameters.AddWithValue("@hints", NewUser.hints);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  #endregion |

### Description of coding

#### CMSGUI Namespace

##### Classes

|  |  |  |
| --- | --- | --- |
|  | Class | Description |
| Public class | [App](file:///C:\Users\Anirban\Documents\GitHub\DailyNoteBook\Help\html\521639525.htm) | Interaction logic for App.xaml |
| Public class | [MainWindow](file:///C:\Users\Anirban\Documents\GitHub\DailyNoteBook\Help\html\494305234.htm) | Interaction logic for MainWindow.xaml |
| Public class | [ShowEvent](file:///C:\Users\Anirban\Documents\GitHub\DailyNoteBook\Help\html\481492962.htm) | Interaction logic for ShowEvent.xaml |

##### **CMS**GUI.Properties Namespace

##### Classes

|  |  |  |
| --- | --- | --- |
|  | Class | Description |
| Protected class | [Settings](file:///C:\Users\Anirban\Documents\GitHub\DailyNoteBook\Help\html\1271527439.htm) |  |
| Protected class | [Resources](file:///C:\Users\Anirban\Documents\GitHub\DailyNoteBook\Help\html\1102861162.htm) | A strongly-typed resource class, for looking up localized strings, etc. |

### Comments and API Documentation

#### Desktop Application Coding

We used NuGet Package Manager for integration facebook apps with my Appplication.

**NuGet 2.2**

NuGet is a free, open source developer focused package management system for the .NET platform intent on simplifying the process of incorporating third party libraries into a .NET application during development.

#### Facebook App:

### Facebook Connect

Facebook Connect is a set of [APIs](http://en.wikipedia.org/wiki/Application_programming_interface) from Facebook that enable Facebook members to log onto third-party websites, applications, mobile devices and gaming systems with their Facebook identity. While logged in, users can connect with friends via these media and post information and updates to their Facebook profile. Developers can use these services to help their users connect and share with their Facebook friends on and off of Facebook and increase engagement for their website or application.

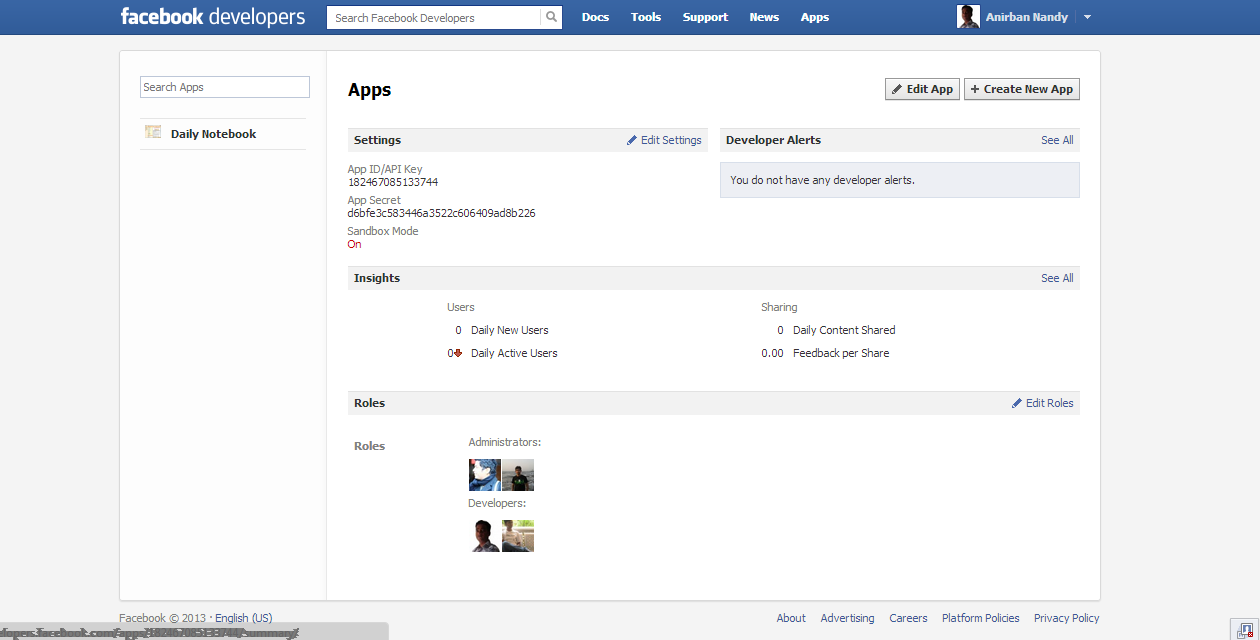
Originally unveiled during Facebook’s developer conference, F8, in July 2008, Facebook Connect became generally available in December 2008. According to an article from The New York Times, "Some say the services are representative of surprising new thinking in Silicon Valley. Instead of trying to hoard information about their users, the Internet companies (including Facebook, Google, MySpace and Twitter) all share at least some of that data so people do not have to enter the same identifying information again and again on different sites."

Since launching Facebook Connect, the company has rolled out additional features related to the services some of which include: Translations for Connect;Facebook Connect Wizard and Facebook Connect for the Mobile Web.

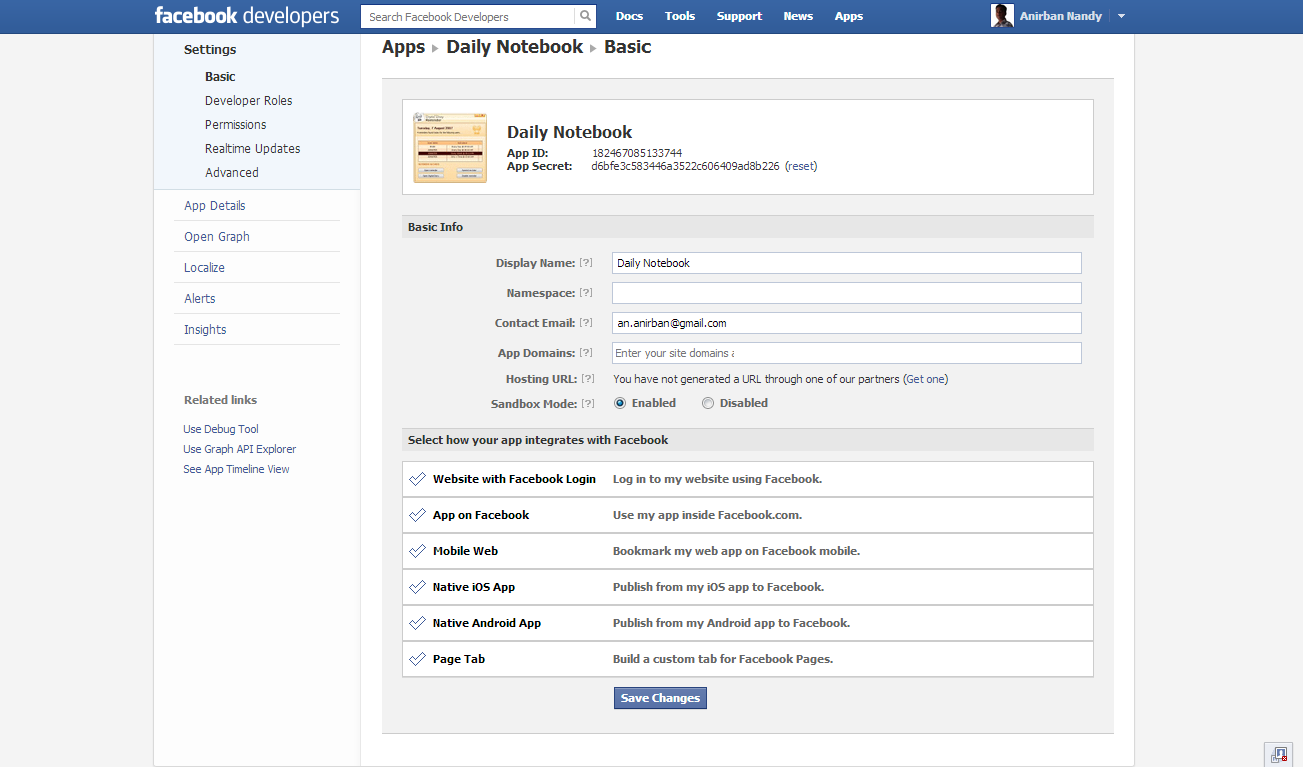
Facebook Connect cannot be used by users in locations that cannot access Facebook (e.g. China), even if the third-party site is otherwise accessible from that location and to other location.

We used NuGet Package Manager for integration facebook apps with my Appplication.

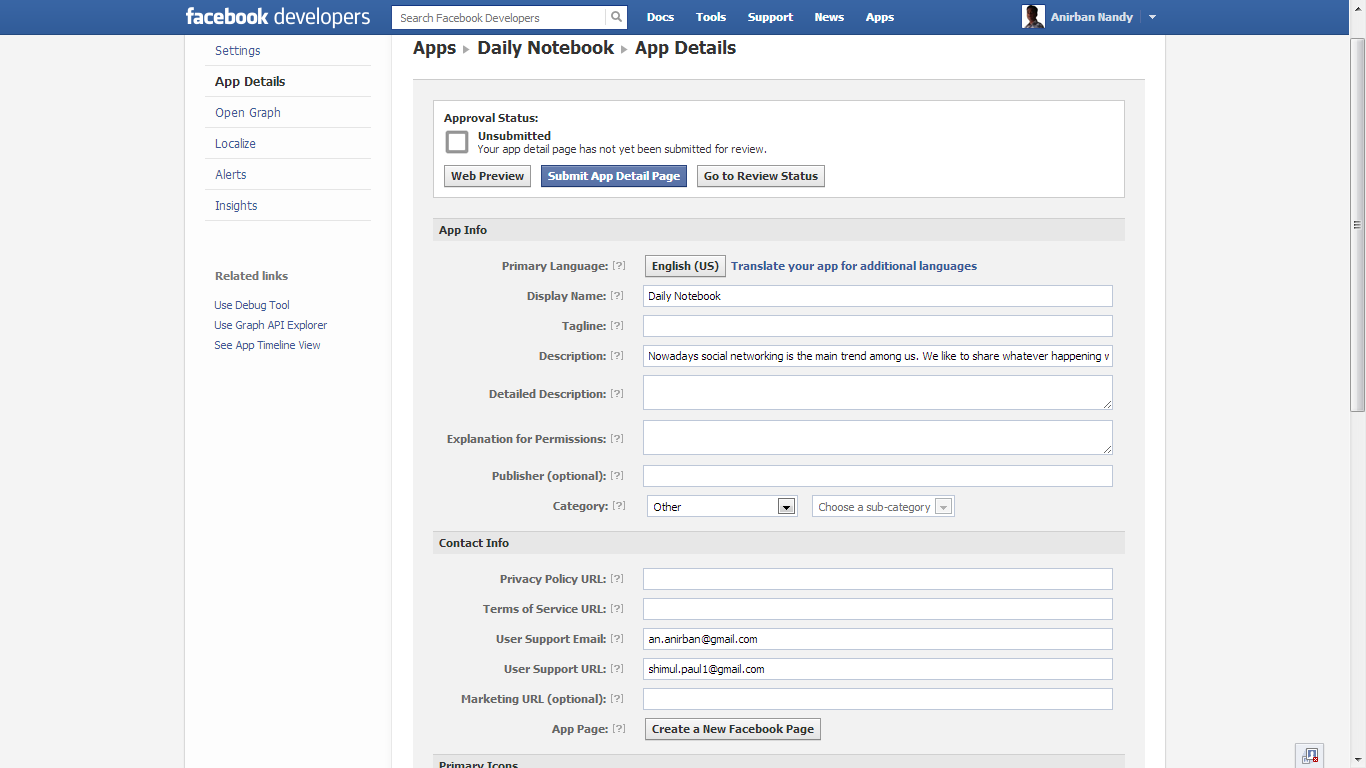
##### Apps Details for Facebook



##### Daily NoteBook Basic for Facebook Apps



##### Daily NoteBook Apps Details for Facebook Apps



#### Twitter App

Great reliance is placed on [open-source software](http://en.wikipedia.org/wiki/Open-source_software).The Twitter Web interface uses the [Ruby on Rails](http://en.wikipedia.org/wiki/Ruby_on_Rails) framework,deployed on a performance enhanced [Ruby Enterprise Edition](http://en.wikipedia.org/w/index.php?title=Ruby_Enterprise_Edition&action=edit&redlink=1)implementation of [Ruby](http://en.wikipedia.org/wiki/Ruby_(programming_language)).

As of April 6, 2011, Twitter engineers confirmed they had switched away from their Ruby on Rails search-stack, to a [Java](http://en.wikipedia.org/wiki/Java_(programming_language)) server they call Blender.

From spring 2007 to 2008 the messages were handled by a Ruby [persistent](http://en.wikipedia.org/wiki/Persistent_data_structure) queue server called [Starling](http://en.wikipedia.org/wiki/Starling_(software)),  but since 2009 implementation has been gradually replaced with software written in[Scala](http://en.wikipedia.org/wiki/Scala_(programming_language)).The service's [application programming interface](http://en.wikipedia.org/wiki/Application_programming_interface) (API) allows other [web services](http://en.wikipedia.org/wiki/Web_service) and applications to integrate with Twitter.

Individual tweets are registered under unique IDs using software called [snowflake](http://en.wikipedia.org/w/index.php?title=Snowflake_(software)&action=edit&redlink=1) and geolocation data is added using 'Rockdove'. The URL shortner t.co then checks for a spam link and shortens the URL. The tweets are stored in a [MySQL](http://en.wikipedia.org/wiki/MySQL) database using [Gizzard](http://en.wikipedia.org/w/index.php?title=Gizzard_(software)&action=edit&redlink=1) and acknowledged to users as having been sent. They are then sent to search engines via the [Firehose](http://en.wikipedia.org/w/index.php?title=Firehose_(software)&action=edit&redlink=1) API. The process itself is managed by [FlockDB](http://en.wikipedia.org/wiki/FlockDB) and takes an average of 350 ms.

**CMS** uses twitter application to let users visit his/ her twiter account through **CMS**. A user can also update his status and share his thouths to Twitter via **CMS**

#### Google Map API

Google Maps (formerly Google Local) is a [web mapping](http://en.wikipedia.org/wiki/Web_mapping) service application and technology provided by [Google](http://en.wikipedia.org/wiki/Google), that powers many map-based services, including the Google Maps website, Google Ride Finder, Google Transit,and maps embedded on third-party websites via the Google Maps [API](http://en.wikipedia.org/wiki/Application_programming_interface).It offers street maps, a [route planner](http://en.wikipedia.org/wiki/Route_planner) for traveling by foot, car, bike (beta), or with public transportation and a locator for urban businesses in numerous countries around the world. Google Maps satellite images are not updated in real time, but rather they are several months or years old.

Google Maps uses a close variant of the [Mercator projection](http://en.wikipedia.org/wiki/Mercator_projection), so it cannot show areas around the poles. A related product is [Google Earth](http://en.wikipedia.org/wiki/Google_Earth), a stand-alone program which offers more globe-viewing features, including showing polar areas.

Google Map API has been used in **CMS** desktop application sothat a user can see his/ her location through **CMS**. A user do not need to open internet browser, while using **CMS** features he can check his/ her location as well by simply clicking on the Maps option.

## Standardization of the coding

12

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:

Member variables.

Constructors & Finalizers.

Nested Enums, Structs, and Classes.

Properties

Methods

* Sequence declarations within type groups based upon access modifier and visibility:

Public

Protected

Internal

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 braces.
* 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 separatedeclaration.

Example:

// Bad!

[Attrbute1, Attrbute2, Attrbute3]

public class MyClass

{…}

// Good!

[Attrbute1, RelatedAttribute2]

[Attrbute3]

[Attrbute4]

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 unnecessary 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 resources 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 if 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](javascript:void(0))

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#, thecatch 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.

#### 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.

#### 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

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

### Database & Data Integrity Testing

The databases and the database processes should be tested as a subsystem within the **CMS** 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: | SQL queries will be executed in the DB to verify the data content and correctness. |
| Completion Criteria: | All planned tests have been executed.  All defects that have been identified have been resolved  All resolutions have been implemented. |

### 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 **CMS**:

|  |  |
| --- | --- |
| Test Objective: | Ensure proper target-of-test functionality, including business process validation. |
| Technique: | Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following:   * 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: | * 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 **CMS**.

|  |  |
| --- | --- |
| 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: | * 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: | • All planned regression tests have been executed.  • All identified defects have been resolved. |

### 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: | * 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: | * 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 **CMS** application is up to the desired level. |
| Technique: | * 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: | * 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: | * 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: | * 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.

### 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: | * After initial deployment to the test environment validate all critical components of the application prior to proceeding with testing. |
| Completion Criteria: | * Navigation through the application at high level is possible, testing can continue. |

### 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 newdata Model in the Destination DB(s) will be migrated from the existing systemssource 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: | * 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: | * 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

Testing is a set of activities that can be planned in advance and conducted systematically. During testing, the program to tested is executed what a set of test cases, and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to.

In a software development project, errors can be injected at any stage during the development. Some requirement errors and design errors are likely to remain undetected. Ultimately, these errors will be reflected in the code. Hence, testing Performs a very critical role for quality assurance and for ensuring the reliability of software.

#### Cause of Testing:

The first test of the system is to see whether it produces the correct output. Following this a variety of other tests are conducted:

**Response time:** this test is conducted to measure the processing of the software.

**Volume testing:** In this test, we create as many data as would normally be used to a variety that the hardware and the software will function correctly.

**Stress Testing:** The purpose of this is to prove that the candidate system should not malfunction under peak load.

**Recovery & Security:** A forced system is induced to test a backup recovery procedure.

System testing is verification of the workability of a system. For this purpose the system is used experimentally to ensure that it will run according to its specification and in the way users expect. These are two stages to this:

* The testing of the individual programs by their programmers called Unit testing and
* The testing of the overall System testing.

#### Unit testing includes the following:

* Test for number of input parameters equal to number of arguments.
* Test the parameter and argument attributes match.
* Feasibility and validity checks on input data.
* Checks for interpretation of symbols correctly.
* Checks for accurate branching and looping.
* Checks on logical file addressing and searching.
* Checks on the capacity of storage areas and buffers.
* Checks on updating procedure.
* Checks on contents and layout of printed and displayed output.
* Checks for batch control totals.
* Checks on interfacing with other programs, software, database and operating system.
* Checks on documentation.

#### System testing includes the following:

* Interfacing of run within the system.
* Compilation and continuity of control totals.
* Capacity of logical file storage areas and the handling of overflow.
* Error correction procedures including user involvement.
* User request for amendments and output.
* Timing of runs and routines for the data volumes to be actually handled.
* Output preparation and distribution.
* Audit requirements.
* Logical physical file housekeeping and control.

The usual procedures in testing are to create data for the initial tests and to use live data for later testing.

#### The following points should be remembered primarily:

* Both the artificial and live data should be representative of reality;
* Live data can often be checked against the previous system’s result;
* If the previous and new system differ, the two sets of result should be reconciled if at all possible;
* Logical files are usually needed to fully test the programs and routines;
* Data generating techniques are useful for simulating large volume of input data file records;
* In the final trial run of the complete routine, asset of input data is passed through to the resultant output and/or file updating stage(s);
* Test data should include known incorrect data with a view to test the validation and control procedure.

#### Any engineered product (and most other things) can be tested in one of two ways:

* Knowing the specified function that a product has been designed to perform, test can be conducted that demonstrate each function is fully operational, at the same time searching for errors in each function;
* Knowing the internal workings of a product, tests can be conducted to ensure that all internal operation performs according to specification and all internal components have been adequately exercised. The first test approach is called Black-box and the second, White-box testing.

#### White-box testing:

White-box testing sometimes called glass-box testing, is a test case design to derive test cases. Using white-box testing method, the software engineer can test:

* Guarantee that all independent paths within a module have been exercised at least one;
* Exercise all logical decisions on their true and false sides;
* Execute all loops at there boundaries and within there operational bounds; and
* Exercise internal data structures to assure there validity.

A reasonable question might be posed at this juncture: “why spend time and energy worrying about (and testing) logical minutes when we might better spend effort ensuring that program requirements have been met?” stated another way, why don’t we spend all of our energies on black-box testing? The answer lies in the nature of software defects.

* Logical errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed. Errors tend to creep into our work when we design and implement function, conditions and control that are out of the mainstream. Every processing tends to be well understood (and well scrutinized), while “special case” processing tends to fall into the cracks.
* We often believe that a logical path is not likely to be executed when, in fact, it may be executed in regular basis. The logical flow of a program is sometimes counterintuitive, meaning that our unconscious assumption about flow of control and data may lead us to make design errors that are uncovered only when path testing commences.
* Typographical errors are random. When a program is translated into programming language source code, it is likely that some typing error will occur. Many will be uncovered by syntax checking mechanism, but others will go undetected until testing begins. It is likely that a type will exist on an obscure logical path as on a mainstream path.

Each of these reasons provides an argument for conducting white-box tests. Black-boxtesting, no matter how through, may miss the kinds of errors noted above. As Beizer has stated: “Bugs lurk in corners and congregate at boundaries”. White-box testing is far more likely to uncover them.

#### Basis Path Testing:

Basis path testing is a **White-box testing** technique first proposed by Tom MeCabe. The basis path methods enables the test case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basic set that are guaranteed to execute every statement in the program at least one time during testing.

#### Black-box Testing:

It focuses on the functional requirements of the software. That is Black-box testing enables software engineer to derive sets of input conditions that will fully exercise all functional requirements of a program. Black-box testing is not a alternative of white box testing. Rather, it is a complementary approach that is likely to uncover a different class of errors than White-box methods.

Black-box testing attempts to find errors in the following categories.

* Incorrect or missing function.
* Interface errors
* Errors in data structures or external data base access
* Performance errors
* Initializations or termination errors

Unlike **White-box** testing, which is performed early in the testing process, **black-box** testing tends to be applied during later stage of testing. Because **black-box** testing purposely disregards control structure, attention is focused on the information domain. Tests are based on source data from a

previous period so that the result from the new system can be compared with that of the old one.

With those of the previous ones to answer the following questions:

* How is functional validity testing?
* What classes of input will make good test cases?
* Is the system particularly sensitive to certain input values?
* How are the boundaries of a data class isolated?
* What data rates and data volume can the system tolerate?
* What effect will specific combinations of the data have on system operation?

By applying **Black-box** techniques, we derive a set of test cases that satisfy the following criteria:

* Test cases that reduce, by errors and a designed to achieve reasonableness testing.
* Test cases that tell us something about the presence or absence of classes of errors, than errors associated only with the specific test at hand.

## Test reports for Unit Test Cases and System Test Cases

### Test reports for Unit Test Cases

|  |  |  |
| --- | --- | --- |
| Test Case Id | Comment | Status |
| **CMS**-001 | NA | PASS |
| **CMS**-002 | NA | PASS |
| **CMS**-003 | NA | PASS |
| **CMS**-004 | NA | PASS |
| **CMS**-005 | NA | PASS |
| **CMS**-006 | NA | PASS |
| **CMS**-007 | NA | PASS |
| **CMS**-008 | NA | PASS |
| **CMS**-009 | NA | PASS |
| **CMS**-010 | NA | PASS |
| **CMS**-011 | NA | PASS |
| **CMS**-012 | NA | PASS |
| **CMS**-013 | NA | PASS |
| **CMS**-014 | NA | PASS |
| **CMS**-015 | NA | PASS |
| **CMS**-016 | NA | PASS |
| **CMS**-017 | NA | PASS |
| **CMS**-018 | NA | PASS |
| **CMS**-019 | NA | PASS |
| **CMS**-020 | NA | PASS |
| **CMS**-021 | NA | PASS |
| **CMS**-022 | NA | PASS |
| **CMS**-023 | NA | PASS |
| **CMS**-024 | NA | PASS |
| **CMS**-025 | NA | PASS |
| **CMS**-026 | NA | PASS |
| **CMS**-027 | NA | PASS |
| **CMS**-028 | NA | PASS |
| **CMS**-029 | NA | PASS |
| **CMS**-030 | NA | PASS |
| **CMS**-031 | NA | PASS |
| **CMS**-032 | NA | PASS |
| **CMS**-033 | NA | PASS |
| **CMS**-034 | NA | PASS |
| **CMS**-035 | NA | PASS |
| **CMS**-036 | NA | PASS |
| **CMS**-037 | NA | PASS |
| **CMS**-038 | NA | PASS |
| **CMS**-039 | NA | PASS |
| **CMS**-040 | NA | PASS |
| **CMS**-041 | NA | PASS |
| **CMS**-042 | NA | PASS |
| **CMS**-043 | NA | PASS |
| **CMS**-044 | NA | PASS |
| **CMS**-045 | NA | PASS |
| **CMS**-046 | NA | PASS |
| **CMS**-047 | NA | PASS |
|  |  |  |
| **CMS**-048 | NA | PASS |
| **CMS**-049 | NA | PASS |
| **CMS**-050 | NA | PASS |
| **CMS**-051 | NA | PASS |
| **CMS**-052 | NA | PASS |
| **CMS**-053 | NA | PASS |
| **CMS**-054 | NA | PASS |
| **CMS**-055 | NA | PASS |
| **CMS**-056 | NA | PASS |
| **CMS**-057 | NA | PASS |
| **CMS**-058 | NA | PASS |
| **CMS**-059 | NA | PASS |
| **CMS**-060 | NA | PASS |
| **CMS**-061 | NA | PASS |
| **CMS**-062 | NA | PASS |
| **CMS**-063 | NA | PASS |
| **CMS**-064 | NA | PASS |
| **CMS**-065 | NA | PASS |
| **CMS**-066 | NA | PASS |

### Test reports for System Test Cases

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Test Case Id | Comment | Status |
| **CMS**-067 | NA | PASS |
| **CMS**-068 | NA | PASS |
| **CMS**-069 | NA | PASS |
| **CMS**-070 | NA | PASS |
| **CMS**-071 | NA | PASS |
| **CMS**-072 | NA | PASS |
| **CMS**-073 | NA | PASS |
| **CMS**-074 | NA | PASS |
| **CMS**-075 | NA | PASS |
| **CMS**-076 | NA | PASS |
| **CMS**-077 | NA | PASS |
| **CMS**-078 | NA | PASS |
| **CMS**-079 | NA | PASS |
| **CMS**-080 | NA | PASS |
| **CMS**-081 | NA | PASS |
| **CMS**-082 | NA | PASS |
| **CMS**-083 | NA | PASS |
| **CMS**-084 | NA | PASS |
| **CMS**-085 | NA | PASS |
| **CMS**-086 | NA | PASS |
| **CMS**-087 | NA | PASS |
| **CMS**-089 | NA | PASS |
| **CMS**-090 | NA | PASS |
| **CMS**-091 | NA | PASS |
| **CMS**-092 | NA | PASS |
| **CMS**-093 | 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 produces 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;

1. To initialize variables to contain information about a product, add the following declaration statements to **Main** method:
2. string sProdName = "Widget";
3. int iUnitQty = 100;

double dUnitCost = 1.03;

1. Specify the message that the class produces 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 ");

1. For readability, use the **Indent** method to indent subsequent messages in the Output window:

Debug.Indent();

1. To display the content of selected variables, use the **WriteLine** method as follows:
2. Debug.WriteLine("The product name is " + sProdName);
3. Debug.WriteLine("The available units on hand are" + iUnitQty.ToString());

Debug.WriteLine("The per unit cost is " + dUnitCost.ToString());

1. 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:
2. System.Xml.XmlDocument oxml = new System.Xml.XmlDocument();

Debug.WriteLine(oxml);

1. 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:
2. Debug.WriteLine("The product name is " + sProdName,"Field");
3. Debug.WriteLine("The units on hand are" + iUnitQty,"Field");
4. Debug.WriteLine("The per unit cost is" + dUnitCost.ToString(),"Field");

Debug.WriteLine("Total Cost is " + (iUnitQty \* dUnitCost),"Calc");

1. 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.
2. Debug.WriteLineIf(iUnitQty > 50, "This message WILL appear");
3. Debug.WriteLineIf(iUnitQty < 50, "This message will NOT appear");
4. Use the **Assert** method of the **Debug** class so that the Output window displays the message only if a specified condition evaluates to false:
5. Debug.Assert(dUnitCost > 1, "Message will NOT appear");
6. Debug.Assert(dUnitCost < 1, "Message will appear since dUnitcost < 1 is false");
7. 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:
8. TextWriterTraceListener tr1 = new TextWriterTraceListener(System.Console.Out);
9. Debug.Listeners.Add(tr1);
10. TextWriterTraceListener tr2 = new TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));

Debug.Listeners.Add(tr2);

1. 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.
2. Debug.Unindent();

Debug.WriteLine("Debug Information-Product Ending");

1. 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. at Class1.Main(String[] args) <%Path%>\class1.cs(34)
24. The product name is Widget
25. The available units on hand are100
26. The per unit cost is 1.03
27. Debug Information-Product Ending
28. Trace Information-Product Starting
29. The product name is Widget
30. Field: The product name isWidget
31. This message WILL appear
32. Trace Information-Product Ending
33. The Console window and the Output.txt file should display the following output:
34. The product name is Widget
35. The available units on hand are 100
36. The per unit cost is 1.03
37. Debug Information-Product Ending
38. Trace Information-Product Starting
39. The product name is Widget
40. Field: The product name is Widget
41. This message WILL appear
42. 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:

## Database/data security:

It encrypts the data stored in the database so that even if someone succeeds to hack the database still not much harm could be done.

The application will use Google open-id authentication for web interface.

## Creation of User profiles and access rights

The software requires a predefined username and password to login.

It allows a guest login as well which lets a guest user this application withvery limited access to the user data.

# Cost Estimation of the Project along with Cost Estimation Model

Software development is a highly labor intensive activity. A project of large dimension can easily turn into chaos if proper management controls are not imposed. Therefore the cost/expenditure and the profit gained after implementing the project has to be taken into account. That is we have to consider the cost benefit analysis.

This cost/benefit may be tangible or intangible, direct or indirect, fixed or variable. To build up a large software all the elements required, are estimated to get the development cost of the considering project. When we consider all this requirements we can develop a cost estimation model to find proposed cost of the developing project. And from this model we can track down the expenditure during the course of development.

Now after implementing the project we have to consider gain from it in terms of benefits, that is how much person month can be saved from this project. Therefore we have to consider the total expenditure and the benefit gain from the project once it has been implemented. Here we express the benefits in the terms of person month that is monthly salary of the person concerned for the system, which has to be replaced. Therefore this cost/benefit analysis report gives us a total picture of how a company gets benefit from candidate system once that has replace a older one.

The project developing components like hardware, personnel, facility and supply cost are also taken into consideration during the cost estimation. Then we identify the cost and benefit of a given system and categories them or analysis. And from that estimated cost we track the expenditure and then calculate the benefits.

In developing the cost estimation of a project we need to consider several cost elements. Among them is hardware, personal, facility, operating and supply cost are noteworthy.

The model for estimating cost is mainly based on the total lines of cop1 delivered. As this is not such code based rather than a p1ign based project so we estimate the cost on the consideration of how much time it can take in p1igning the user interfaces and how many interfaces are required. The cost is then calculated from the total p1ign hours and as it is a single handed project, so this is the time taken by a single person.

The cost of man-power involved in this project is not considered in this estimation. We should consider the cost when we shall go for any live project. This cost is depending on several factors like skill set, location of country etc. e.g. man-hour cost is around Rs.250.00 to Rs.300.00 in India whereas for USA it varies from US$60.00 to US$200.00. Most of the cases, Man (person) power cost is considerable higher than all other costs. Software cost and effort estimation will never be an exact science. Too many variables human, technical, environmental can affect the ultimate cost of the software and effort applied to develop it. To achieve reliable cost and effort estimates, a number of option arise, 1) Base estimates on similar projects that have been already completed; 2) Using relatively simple “decomposition techniques” to generate project cost and effort estimates; 3) Using one or more empirical models for software cost and effort estimation.

We used the basic COCOMO model, which gives an approximate estimate of our **CMS** project parameters. The basic COCOMO estimation model is given by the following expressions:

Effort = a1 \* (KLOC)a2 PM

Tdev = b1 \* (Effort)b2months

Where

KLOC is the estimated size of the software product expressed in Kilo Lines of Code a1, a2, b1, b2 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 **CMS**, the formula for estimating the effort based on the code size is shown below:

Semi-detached **CMS**: Tdev = 3.0\*(KLOC)1.12 PM

## Estimation of development time

For our Semi-detached class software product **CMS**, the formula for estimating the development time based on the effort is given below:

Semi-detached **CMS**: Tdev = 2.5\*(Effort)0.35 months

Assume that the size of a Semi-detached **CMS**product has been estimated to be 3,200 lines of source code. Assume that the average salary of software engineer(me) is Rs. 20,000 per month.

Assume that the size of our

The basic COCOMO estimation formula for **CMS** semidetached software:

Our Effort =3.0\*(3.2)1.12PM

= 11 PM

Normal Development time = 2.5\*(11)0.35months

=6 months

Cost required to develop the product = Rs. 6 \* 20000

= Rs. 120,000

# Reports

* List of Facebook updates could be generated.
* List of twitter update could be generated.
* A list of events could be generated.
* List of LinkedIn update could be generated.
* List of google plus update could be generated.

# Future scope and further enhancement of the Project

To sink with multiple Construction sites with a centralized database.

To support UNIX / Linux, MAC OSX Operating systems.

Mobile application for querying the progress of works assigned to the workers.

# Bibliography

## Website

* [http://www.google.co.in](http://www.google.co.in/)
* <http://en.wikipedia.org>
* <http://msdn.microsoft.com/en-us/>
* <http://www.microsoft.com/en-us/default.aspx>
* <http://www.codeplex.com/>
* <http://stackoverflow.com/>
* <http://www.codeguru.com/>
* [http://www.w3schools.com](http://www.w3schools.com/)

## Books

* 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 Used:

### Visual Studio 2010

visual_studio_logo

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](http://en.wikipedia.org/wiki/Code_editor) supporting [IntelliSense](http://en.wikipedia.org/wiki/IntelliSense) as well as [code refactoring](http://en.wikipedia.org/wiki/Code_refactoring). The integrated [debugger](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building [GUI](http://en.wikipedia.org/wiki/GUI) applications, web designer, [class](http://en.wikipedia.org/wiki/Class_(computing)) designer, and [database schema](http://en.wikipedia.org/wiki/Database_schema) designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for [source-control](http://en.wikipedia.org/wiki/Source_control) systems (like [Subversion](http://en.wikipedia.org/wiki/Subversion_(software)) and [Visual SourceSafe](http://en.wikipedia.org/wiki/Visual_SourceSafe)) and adding new toolsets like editors and visual designers for [domain-specific languages](http://en.wikipedia.org/wiki/Domain-specific_language) or toolsets for other aspects of the [software development lifecycle](http://en.wikipedia.org/wiki/Software_development_lifecycle) (like the [Team Foundation Server](http://en.wikipedia.org/wiki/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 toString.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](http://msdn.microsoft.com/en-IN/library/system.windows.aspx) 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](http://msdn.microsoft.com/en-IN/library/system.windows.controls.button.aspx) and [RepeatButton](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.repeatbutton.aspx).
* Data Display: [DataGrid](http://msdn.microsoft.com/en-IN/library/system.windows.controls.datagrid.aspx), [ListView](http://msdn.microsoft.com/en-IN/library/system.windows.controls.listview.aspx),and [TreeView](http://msdn.microsoft.com/en-IN/library/system.windows.controls.treeview.aspx).
* Date Display and Selection: [Calendar](http://msdn.microsoft.com/en-IN/library/system.windows.controls.calendar.aspx) and [DatePicker](http://msdn.microsoft.com/en-IN/library/system.windows.controls.datepicker.aspx).
* Dialog Boxes: [OpenFileDialog](http://msdn.microsoft.com/en-IN/library/microsoft.win32.openfiledialog.aspx), [PrintDialog](http://msdn.microsoft.com/en-IN/library/system.windows.controls.printdialog.aspx), and [SaveFileDialog](http://msdn.microsoft.com/en-IN/library/microsoft.win32.savefiledialog.aspx).
* Digital Ink: [InkCanvas](http://msdn.microsoft.com/en-IN/library/system.windows.controls.inkcanvas.aspx) and [InkPresenter](http://msdn.microsoft.com/en-IN/library/system.windows.controls.inkpresenter.aspx).
* Documents: [DocumentViewer](http://msdn.microsoft.com/en-IN/library/system.windows.controls.documentviewer.aspx), [FlowDocumentPageViewer](http://msdn.microsoft.com/en-IN/library/system.windows.controls.flowdocumentpageviewer.aspx), [FlowDocumentReader](http://msdn.microsoft.com/en-IN/library/system.windows.controls.flowdocumentreader.aspx), [FlowDocumentScrollViewer](http://msdn.microsoft.com/en-IN/library/system.windows.controls.flowdocumentscrollviewer.aspx), and[StickyNoteControl](http://msdn.microsoft.com/en-IN/library/system.windows.controls.stickynotecontrol.aspx).
* Input: [TextBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.textbox.aspx), [RichTextBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.richtextbox.aspx), and [PasswordBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.passwordbox.aspx).
* Layout: [Border](http://msdn.microsoft.com/en-IN/library/system.windows.controls.border.aspx), [BulletDecorator](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.bulletdecorator.aspx), [Canvas](http://msdn.microsoft.com/en-IN/library/system.windows.controls.canvas.aspx), [DockPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.dockpanel.aspx), [Expander](http://msdn.microsoft.com/en-IN/library/system.windows.controls.expander.aspx), [Grid](http://msdn.microsoft.com/en-IN/library/system.windows.controls.grid.aspx), [GridView](http://msdn.microsoft.com/en-IN/library/system.windows.controls.gridview.aspx), [GridSplitter](http://msdn.microsoft.com/en-IN/library/system.windows.controls.gridsplitter.aspx), [GroupBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.groupbox.aspx), [Panel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.panel.aspx),[ResizeGrip](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.resizegrip.aspx), [Separator](http://msdn.microsoft.com/en-IN/library/system.windows.controls.separator.aspx), [ScrollBar](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.scrollbar.aspx), [ScrollViewer](http://msdn.microsoft.com/en-IN/library/system.windows.controls.scrollviewer.aspx), [StackPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.stackpanel.aspx), [Thumb](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.thumb.aspx), [Viewbox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.viewbox.aspx), [VirtualizingStackPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.virtualizingstackpanel.aspx), [Window](http://msdn.microsoft.com/en-IN/library/system.windows.window.aspx), and[WrapPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.wrappanel.aspx).
* Media: [Image](http://msdn.microsoft.com/en-IN/library/system.windows.controls.image.aspx), [MediaElement](http://msdn.microsoft.com/en-IN/library/system.windows.controls.mediaelement.aspx), and [SoundPlayerAction](http://msdn.microsoft.com/en-IN/library/system.windows.controls.soundplayeraction.aspx).
* Menus: [ContextMenu](http://msdn.microsoft.com/en-IN/library/system.windows.controls.contextmenu.aspx), [Menu](http://msdn.microsoft.com/en-IN/library/system.windows.controls.menu.aspx), and [ToolBar](http://msdn.microsoft.com/en-IN/library/system.windows.controls.toolbar.aspx).
* Navigation: [Frame](http://msdn.microsoft.com/en-IN/library/system.windows.controls.frame.aspx), [Hyperlink](http://msdn.microsoft.com/en-IN/library/system.windows.documents.hyperlink.aspx), [Page](http://msdn.microsoft.com/en-IN/library/system.windows.controls.page.aspx), [NavigationWindow](http://msdn.microsoft.com/en-IN/library/system.windows.navigation.navigationwindow.aspx), and [TabControl](http://msdn.microsoft.com/en-IN/library/system.windows.controls.tabcontrol.aspx).
* Selection: [CheckBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.checkbox.aspx), [ComboBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.combobox.aspx), [ListBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.listbox.aspx), [RadioButton](http://msdn.microsoft.com/en-IN/library/system.windows.controls.radiobutton.aspx), and [Slider](http://msdn.microsoft.com/en-IN/library/system.windows.controls.slider.aspx).
* User Information: [AccessText](http://msdn.microsoft.com/en-IN/library/system.windows.controls.accesstext.aspx), [Label](http://msdn.microsoft.com/en-IN/library/system.windows.controls.label.aspx), [Popup](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.popup.aspx), [ProgressBar](http://msdn.microsoft.com/en-IN/library/system.windows.controls.progressbar.aspx), [StatusBar](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.statusbar.aspx), [TextBlock](http://msdn.microsoft.com/en-IN/library/system.windows.controls.textblock.aspx), and [ToolTip](http://msdn.microsoft.com/en-IN/library/system.windows.controls.tooltip.aspx).

#### 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](http://msdn.microsoft.com/en-IN/library/system.windows.controls.canvas.aspx) : Child controls provide their own layout.
* [DockPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.dockpanel.aspx) : Child controls are aligned to the edges of the panel.
* [Grid](http://msdn.microsoft.com/en-IN/library/system.windows.controls.grid.aspx) : Child controls are positioned by rows and columns.
* [StackPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.stackpanel.aspx) : Child controls are stacked either vertically or horizontally.
* [VirtualizingStackPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.virtualizingstackpanel.aspx) : Child controls are virtualized and arranged on a single line that is either horizontally or vertically oriented.
* [WrapPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.wrappanel.aspx) : 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. Its a simple language based on XML to create and initialize .NET objects with hierarchical relations. Although it was originally invented for WPF it can by 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 parameter less constructors and make excessive usage 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 ist 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. Declare 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](http://msdn.microsoft.com/en-us/library/system.aspx) | The [System](http://msdn.microsoft.com/en-us/library/system.aspx) 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](http://msdn.microsoft.com/en-us/library/gg145022.aspx) | The System.Activities namespaces contain all the classes necessary to create and work with activities in Window Workflow Foundation. |
| [System.AddIn](http://msdn.microsoft.com/en-us/library/gg145020.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145034.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145035.aspx) | The System.Collections namespaces contain types that define various standard, specialized, and generic collection objects. |
| [System.ComponentModel](http://msdn.microsoft.com/en-us/library/gg145042.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145027.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145028.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145029.aspx) | The System.Deployment namespaces contain types that support deployment of ClickOnce applications. |
| [System.Device.Location](http://msdn.microsoft.com/en-us/library/system.device.location.aspx) | The [System.Device.Location](http://msdn.microsoft.com/en-us/library/system.device.location.aspx) 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](http://msdn.microsoft.com/en-us/library/system.device.location.aspx) 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 developers who use this API do not need to tailor applications to specific hardware configurations. |
| [System.Diagnostics](http://msdn.microsoft.com/en-us/library/gg145030.aspx) | 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. |
| [System.DirectoryServices](http://msdn.microsoft.com/en-us/library/gg145037.aspx) | The System.DirectoryServices namespaces contain types that provide access to Active Directory from managed code. |
| [System.Drawing](http://msdn.microsoft.com/en-us/library/gg145023.aspx) | 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. |
| [System.Dynamic](http://msdn.microsoft.com/en-us/library/system.dynamic.aspx) | The [System.Dynamic](http://msdn.microsoft.com/en-us/library/system.dynamic.aspx) namespace provides classes and interfaces that support Dynamic Language Runtime. |
| [System.EnterpriseServices](http://msdn.microsoft.com/en-us/library/gg145047.aspx) | 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. |
| [System.Globalization](http://msdn.microsoft.com/en-us/library/system.globalization.aspx) | The [System.Globalization](http://msdn.microsoft.com/en-us/library/system.globalization.aspx) 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](http://msdn.microsoft.com/en-us/library/system.globalization.stringinfo.aspx) and[TextInfo](http://msdn.microsoft.com/en-us/library/system.globalization.textinfo.aspx) provide advanced globalization functionalities, including surrogate support and text element processing. |
| [System.IdentityModel](http://msdn.microsoft.com/en-us/library/gg145031.aspx) | The System.IdentityModel namespaces contain types that are used to provide authentication and authorization for .NET applications. |
| [System.IO](http://msdn.microsoft.com/en-us/library/gg145019.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145016.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145024.aspx) | 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](http://msdn.microsoft.com/en-us/library/system.media.aspx) | The [System.Media](http://msdn.microsoft.com/en-us/library/system.media.aspx) namespace contains classes for playing sound files and accessing sounds provided by the system. |
| [System.Messaging](http://msdn.microsoft.com/en-us/library/gg145046.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145039.aspx) | 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](http://msdn.microsoft.com/en-us/library/system.numerics.aspx) | The [System.Numerics](http://msdn.microsoft.com/en-us/library/system.numerics.aspx) namespace contains numeric types that complement the numeric primitives, such as [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx), [Double](http://msdn.microsoft.com/en-us/library/system.double.aspx), and [Int32](http://msdn.microsoft.com/en-us/library/system.int32.aspx), that are defined by the .NET Framework. |
| [System.Printing](http://msdn.microsoft.com/en-us/library/gg145044.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145033.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145043.aspx) | The System.Resources namespaces contain types that enable developers to create, store, and manage an application's culture-specific resources. |
| [System.Runtime](http://msdn.microsoft.com/en-us/library/gg145017.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145025.aspx) | 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 crytographic services, control access to operations and resources based on policy, and support rights management of application-created content. |
| [System.ServiceModel](http://msdn.microsoft.com/en-us/library/gg145010.aspx) | The System.ServiceModel namespaces contain the types necessary to build Windows Communication Foundation (WCF) service and client applications. |
| [System.ServiceProcess](http://msdn.microsoft.com/en-us/library/gg145038.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145021.aspx) | The System.Speech namespaces contain types that support speech recognition. |
| [System.Text](http://msdn.microsoft.com/en-us/library/gg145012.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145014.aspx) | 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](http://msdn.microsoft.com/en-us/library/system.timers.aspx) | The [System.Timers](http://msdn.microsoft.com/en-us/library/system.timers.aspx) namespace provides the [Timer](http://msdn.microsoft.com/en-us/library/system.timers.timer.aspx) component, which allows you to raise an event on a specified interval. |
| [System.Transactions](http://msdn.microsoft.com/en-us/library/gg145032.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145018.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145013.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145026.aspx) | 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](http://msdn.microsoft.com/en-us/library/gg145048.aspx) | The System.Xaml namespaces contain types that support parsing and processing the Extensible Application Markup Language (XAML). |
| [System.Xml](http://msdn.microsoft.com/en-us/library/gg145036.aspx) | 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](http://msdn.microsoft.com/en-us/library/accessibility.aspx) | The [Accessibility](http://msdn.microsoft.com/en-us/library/accessibility.aspx) and all of its exposed members are part of a managed wrapper for the Component Object Model (COM) accessibility interface. |
| [Microsoft.Activities](http://msdn.microsoft.com/en-us/library/hh135392.aspx) | The Microsoft.Activities namespaces contain types that support MSBuild and debugger extensions for Windows Workflow Foundation applications. |
| [Microsoft.Aspnet.Snapin](http://msdn.microsoft.com/en-us/library/microsoft.aspnet.snapin.aspx) | The [Microsoft.Aspnet.Snapin](http://msdn.microsoft.com/en-us/library/microsoft.aspnet.snapin.aspx) 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](http://go.microsoft.com/fwlink/?linkid=37118). |
| [Microsoft.Build](http://msdn.microsoft.com/en-us/library/gg145008.aspx) | The Microsoft.Build namespaces contain types that provide programmatic access to, and control of, the MSBuild engine. |
| [Microsoft.CSharp](http://msdn.microsoft.com/en-us/library/gg145015.aspx) | 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 betwen the dynamic language runtime (DLR) and C#. |
| [Microsoft.Data.Entity.Build.Tasks](http://msdn.microsoft.com/en-us/library/microsoft.data.entity.build.tasks.aspx) | The [Microsoft.Data.Entity.Build.Tasks](http://msdn.microsoft.com/en-us/library/microsoft.data.entity.build.tasks.aspx) namespace contains two MSBuild tasks that are used by the ADO.NET Entity Data Model Designer (Entity Designer). |
| [Microsoft.JScript](http://msdn.microsoft.com/en-us/library/gg145041.aspx) | The Microsoft.JScript namespaces contain classes that support compilation and code generation using the JScript language. |
| [Microsoft.SqlServer.Server](http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.server.aspx) | The [Microsoft.SqlServer.Server](http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.server.aspx) 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](http://msdn.microsoft.com/en-us/library/gg145009.aspx) | 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.VisualC](http://msdn.microsoft.com/en-us/library/gg145040.aspx) | The Microsoft.VisualC 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](http://msdn.microsoft.com/en-us/library/gg145011.aspx) | 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](http://msdn.microsoft.com/en-us/library/hh135393.aspx) | The Microsoft.Windows namespaces contain types that support themes and preview in Windows Presentation Framework (WPF) applications. |
| [UIAutomationClientsideProviders](http://msdn.microsoft.com/en-us/library/uiautomationclientsideproviders.aspx) | Contains a single type that maps client automation providers. |
| [XamlGeneratedNamespace](http://msdn.microsoft.com/en-us/library/xamlgeneratednamespace.aspx) | 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](http://searchenterpriselinux.techtarget.com/definition/MySQL-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 evelopment,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](http://www.mysql.com/products/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

## Programming Language

### C# - C sharp



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](http://en.wikipedia.org/wiki/Multi-paradigm_programming_language) encompassing [imperative](http://en.wikipedia.org/wiki/Imperative_programming), [declarative](http://en.wikipedia.org/wiki/Declarative_programming), [functional](http://en.wikipedia.org/wiki/Functional_programming), [generic](http://en.wikipedia.org/wiki/Generic_programming), [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming)([class-based](http://en.wikipedia.org/wiki/Class_(computer_science))), and [component-oriented](http://en.wikipedia.org/wiki/Component-based_software_engineering) programming disciplines. It was developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) within the [.NET](http://en.wikipedia.org/wiki/.NET_Framework) initiative and later approved as a standard by [Ecma](http://en.wikipedia.org/wiki/Ecma_International) (ECMA-334) and [ISO](http://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO/IEC 23270). C# is one of the programming languages designed for the [Common Language Infrastructure](http://en.wikipedia.org/wiki/Common_Language_Infrastructure).

C# is intended to be a simple, modern, general-purpose, object-oriented programming language.

#### Characteristics of C#:

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 initialized to zero.  
Products an error message while using an uninitialized 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.

## 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).

## 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.*

## 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 can not 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 requited.
* 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

#### 2. OAuth

OAuth 1.0a is supported as an authorization method for Cacoo. You can register applications here.

You can get your Access Token from the following links.

#### applications

Access Token:https://cacoo.com/oauth/access\_token

Authorize:https://cacoo.com/oauth/authorize

Request Token:https://cacoo.com/oauth/request\_token

## http://t1.gstatic.com/images?q=tbn:ANd9GcS-CmbHGLD4MH83JH1oNIr_acREqblVhrcFuvQfYZR8HFi1Upaqlg Version Control System : GitHub

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.

# Glossary.

**CMS** Daily Notebook & Social Networking Updater

Apps Application

FB Facebook

SRS Software Requirement Specification

DFD Data Flow Diagram

ERD Entity Relationship Diagram

GUI Graphical User Interface

UI User Interface

DB Database

API Application Programming Interface

COCOMO Constructive Cost Model

SDK Sweater 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

--------------------**Thank You**--------------------