MCA 6th Semester
MCSP-060(Project Final Report)

Construction Management System

Arindam Ghosh

Enrolment Number: 093052638

Contents

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. 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.

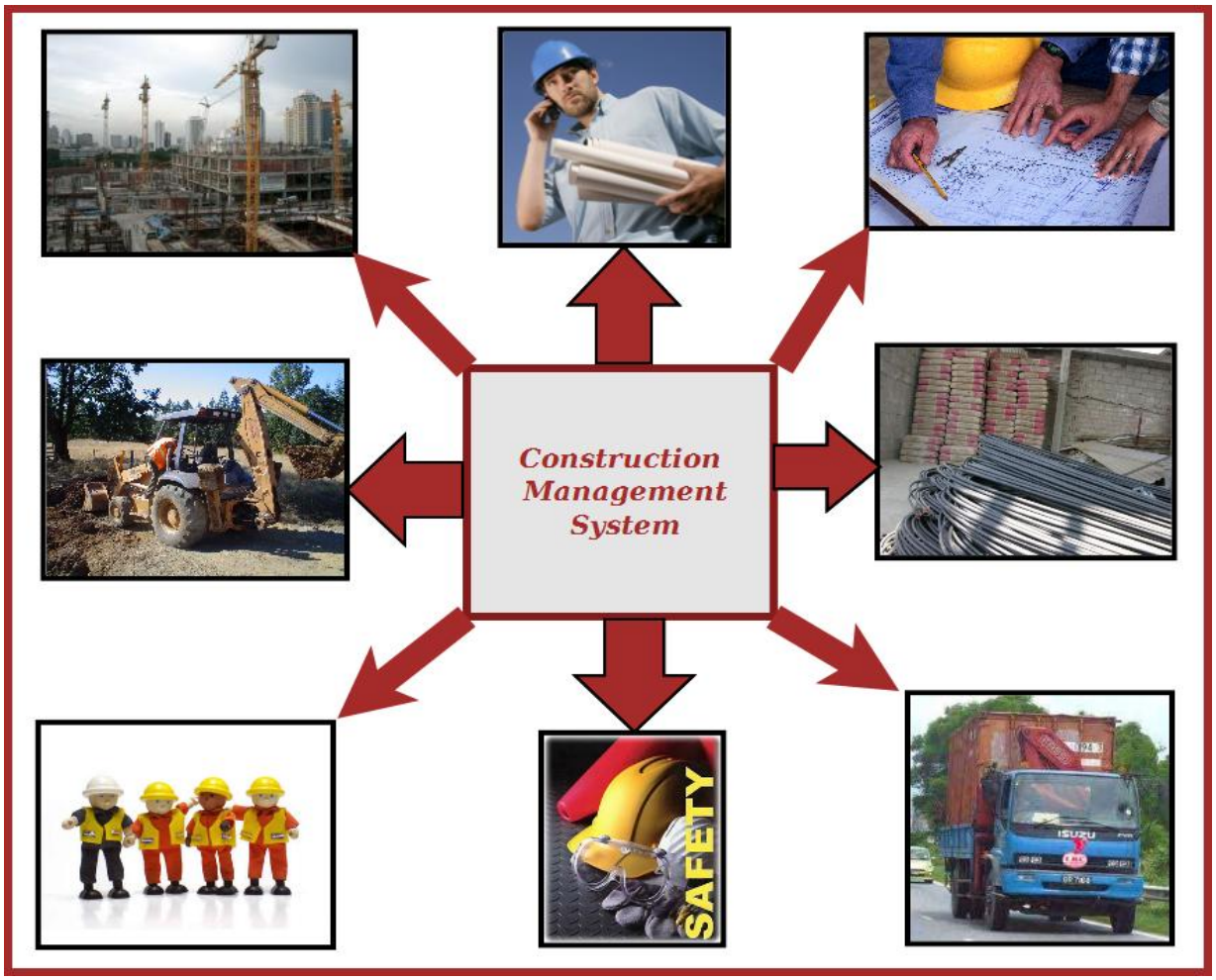


Figure: Overview of Construction Management System

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 an application to maintain all the plan, calculation and everything in one place by this.

Preliminary Investigation:

I spoke with many of my relatives who are associate with building construction and facing management problems. A desktop application could be developed to minimize theses effort. 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 Construrtion Management System.

Feasibility Study:

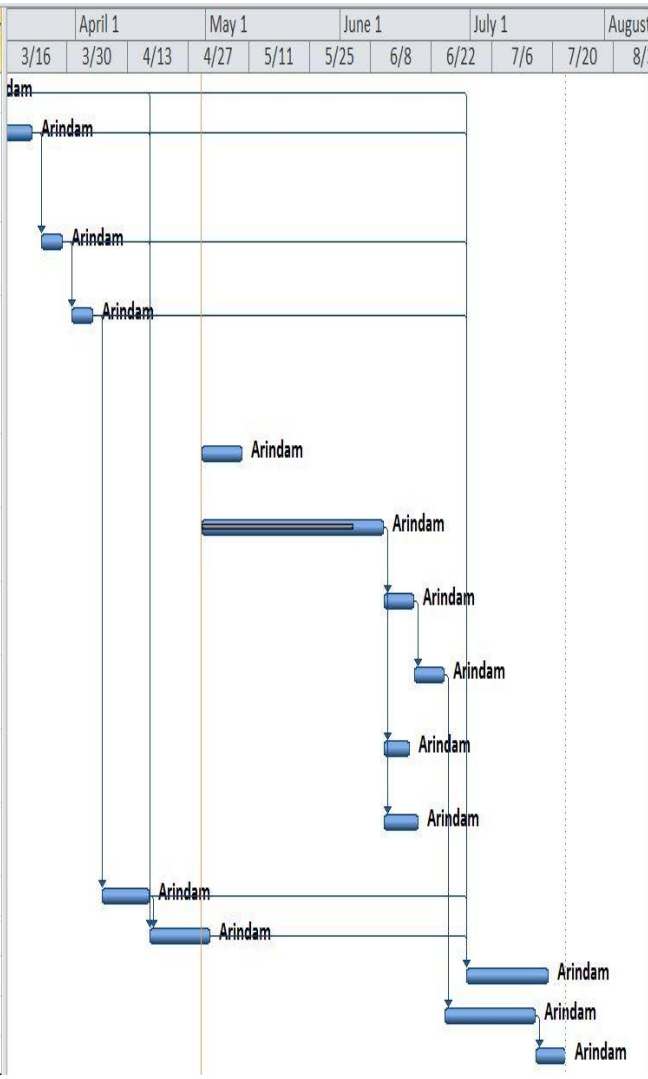
It is an admitted fact that people are becoming more and more addicted with technological usages day by day. People would love an application that would make their work procedure easier. So, undoubtedly it is going to be a popular desktop application.

Project Planning & Scheduling:

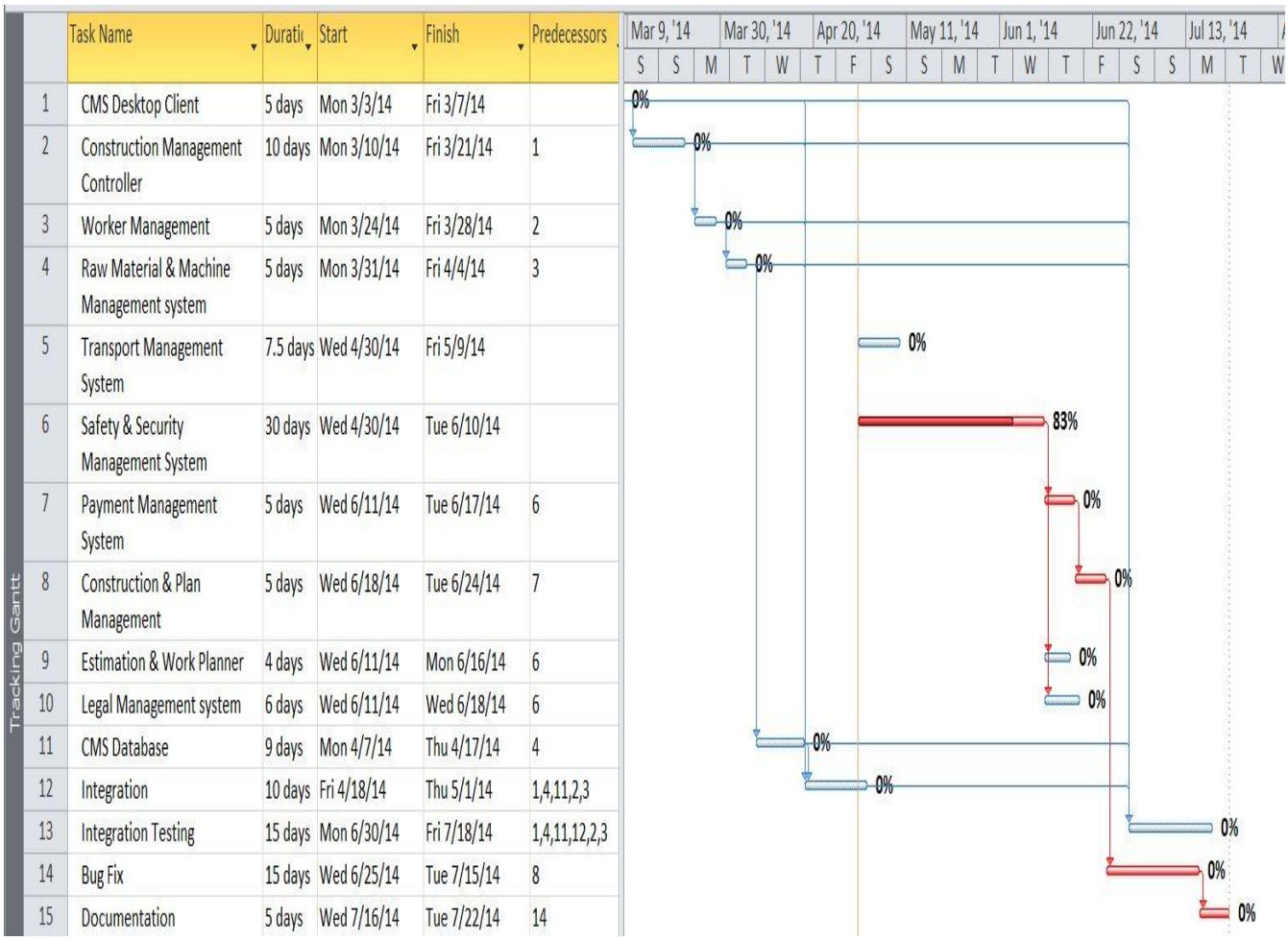
Gantt chart

Gantt Chart

	Task Name	Duratic	Start	Finish	Predecessors	Resource Nam
1	CMS Desktop Client	5 days	Mon 3/3/14	Fri 3/7/14		Arindam
2	Construction Management Controller	10 days	Mon 3/10/14	Fri 3/21/14	1	Arindam
3	Worker Management	5 days	Mon 3/24/14	Fri 3/28/14	2	Arindam
4	Raw Material & Machine Management system	5 days	Mon 3/31/14	Fri 4/4/14	3	Arindam
5	Transport Management System	7.5 days	Wed 4/30/14	Fri 5/9/14		Arindam
6	Safety & Security Management System	30 days	Wed 4/30/14	Tue 6/10/14		Arindam
7	Payment Management System	5 days	Wed 6/11/14	Tue 6/17/14	6	Arindam
8	Construction & Plan Management	5 days	Wed 6/18/14	Tue 6/24/14	7	Arindam
9	Estimation & Work Planner	4 days	Wed 6/11/14	Mon 6/16/14	6	Arindam
10	Legal Management system	6 days	Wed 6/11/14	Wed 6/18/14	6	Arindam
11	CMS Database	9 days	Mon 4/7/14	Thu 4/17/14	4	Arindam
12	Integration	10 days	Fri 4/18/14	Thu 5/1/14	1,4,11,2,3	Arindam
13	Integration Testing	15 days	Mon 6/30/14	Fri 7/18/14	1,4,11,12,2,3	Arindam
14	Bug Fix	15 days	Wed 6/25/14	Tue 7/15/14	8	Arindam
15	Documentation	5 days	Wed 7/16/14	Tue 7/22/14	14	Arindam

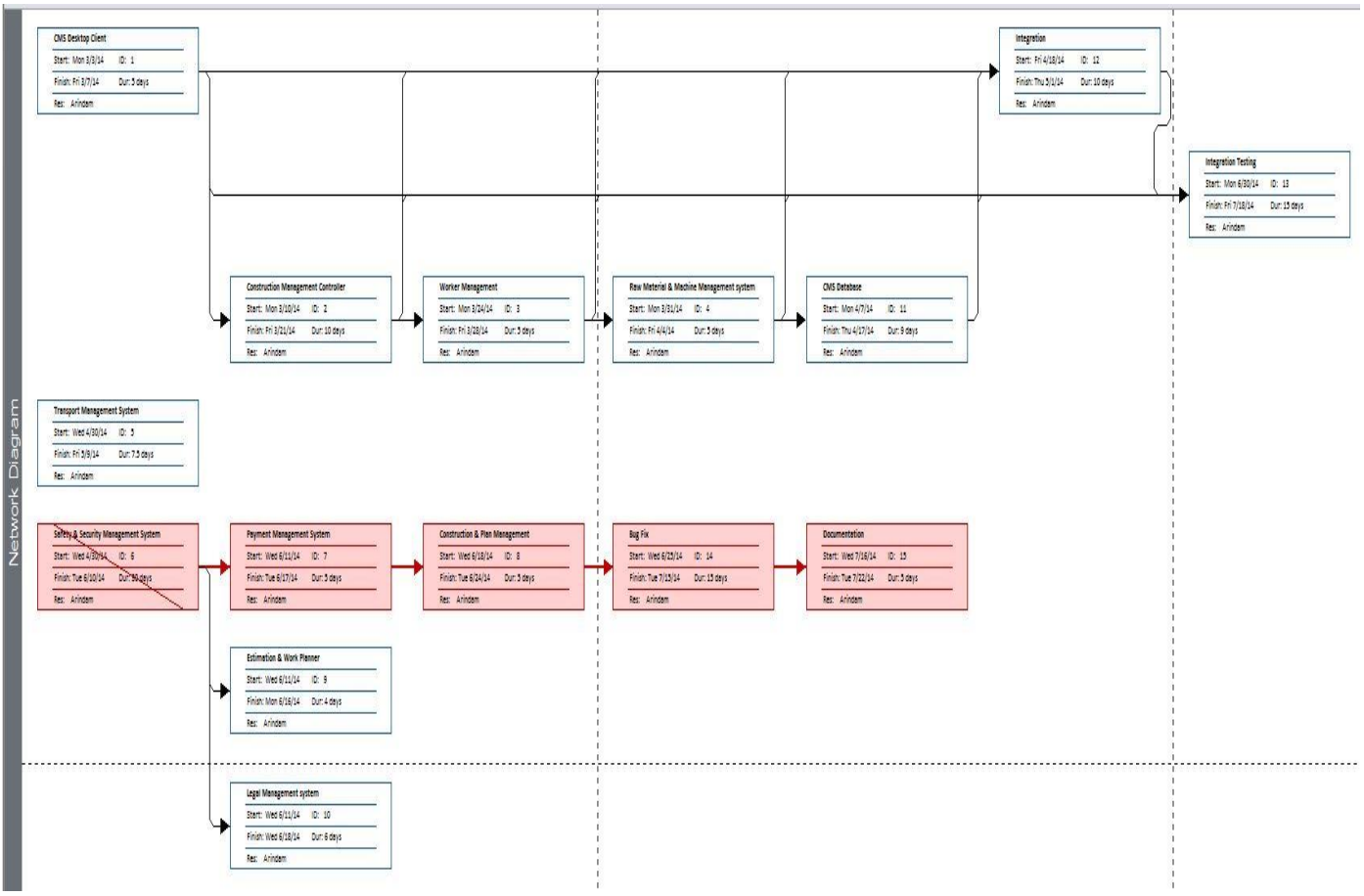


Tracking Gantt



	Task Name	Duration	Start	Finish	Predecessors
1	CMS Desktop Client	5 days	Mon 3/3/14	Fri 3/7/14	
2	Construction Management Controller	10 days	Mon 3/10/14	Fri 3/21/14	1
3	Worker Management	5 days	Mon 3/24/14	Fri 3/28/14	2
4	Raw Material & Machine Management system	5 days	Mon 3/31/14	Fri 4/4/14	3
5	Transport Management System	7.5 days	Wed 4/30/14	Fri 5/9/14	
6	Safety & Security Management System	30 days	Wed 4/30/14	Tue 6/10/14	
7	Payment Management System	5 days	Wed 6/11/14	Tue 6/17/14	6
8	Construction & Plan Management	5 days	Wed 6/18/14	Tue 6/24/14	7
9	Estimation & Work Planner	4 days	Wed 6/11/14	Mon 6/16/14	6
10	Legal Management system	6 days	Wed 6/11/14	Wed 6/18/14	6
11	CMS Database	9 days	Mon 4/7/14	Thu 4/17/14	4
12	Integration	10 days	Fri 4/18/14	Thu 5/1/14	1,4,11,2,3
13	Integration Testing	15 days	Mon 6/30/14	Fri 7/18/14	1,4,11,12,2,3
14	Bug Fix	15 days	Wed 6/25/14	Tue 7/15/14	8
15	Documentation	5 days	Wed 7/16/14	Tue 7/22/14	14

Pert Chart



Software requirement specifications (SRS):

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.

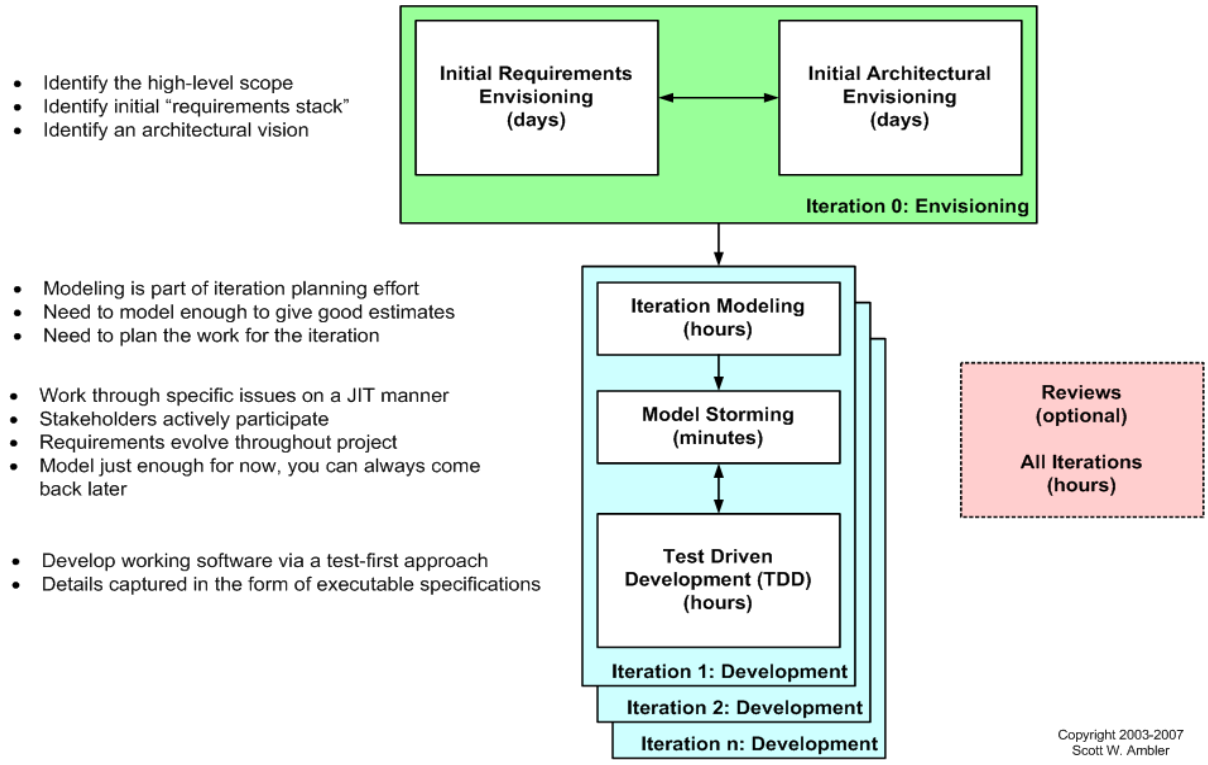


Figure 1: The AMDD lifecycle: Modeling activities throughout the lifecycle of a project

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..

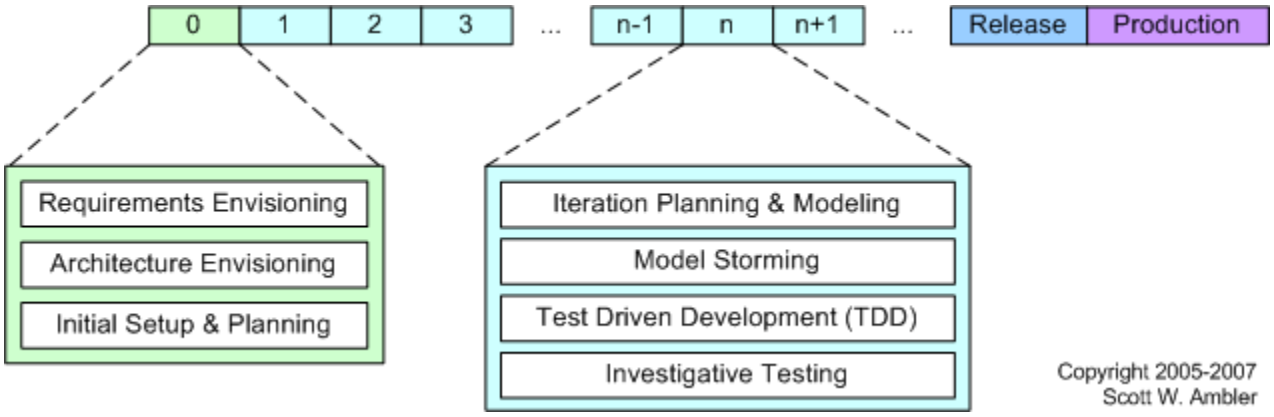
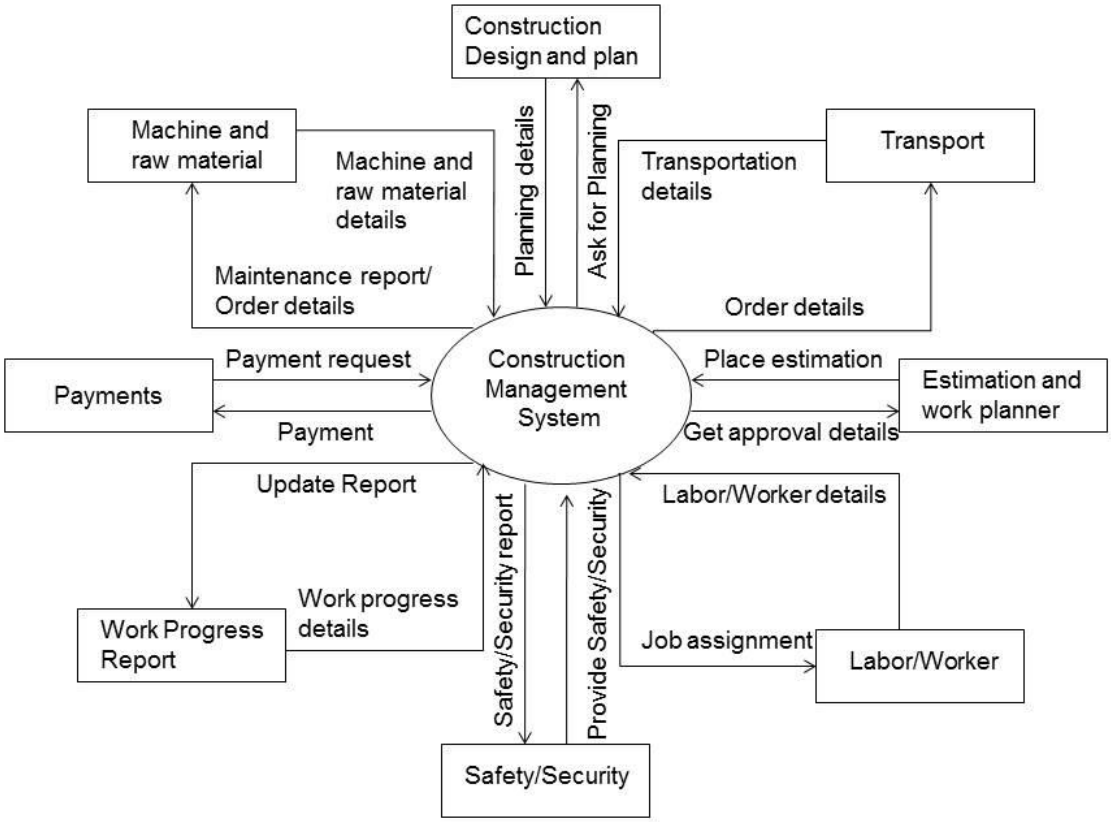


Figure 2AMDD Through the Agile Development Lifecycle.

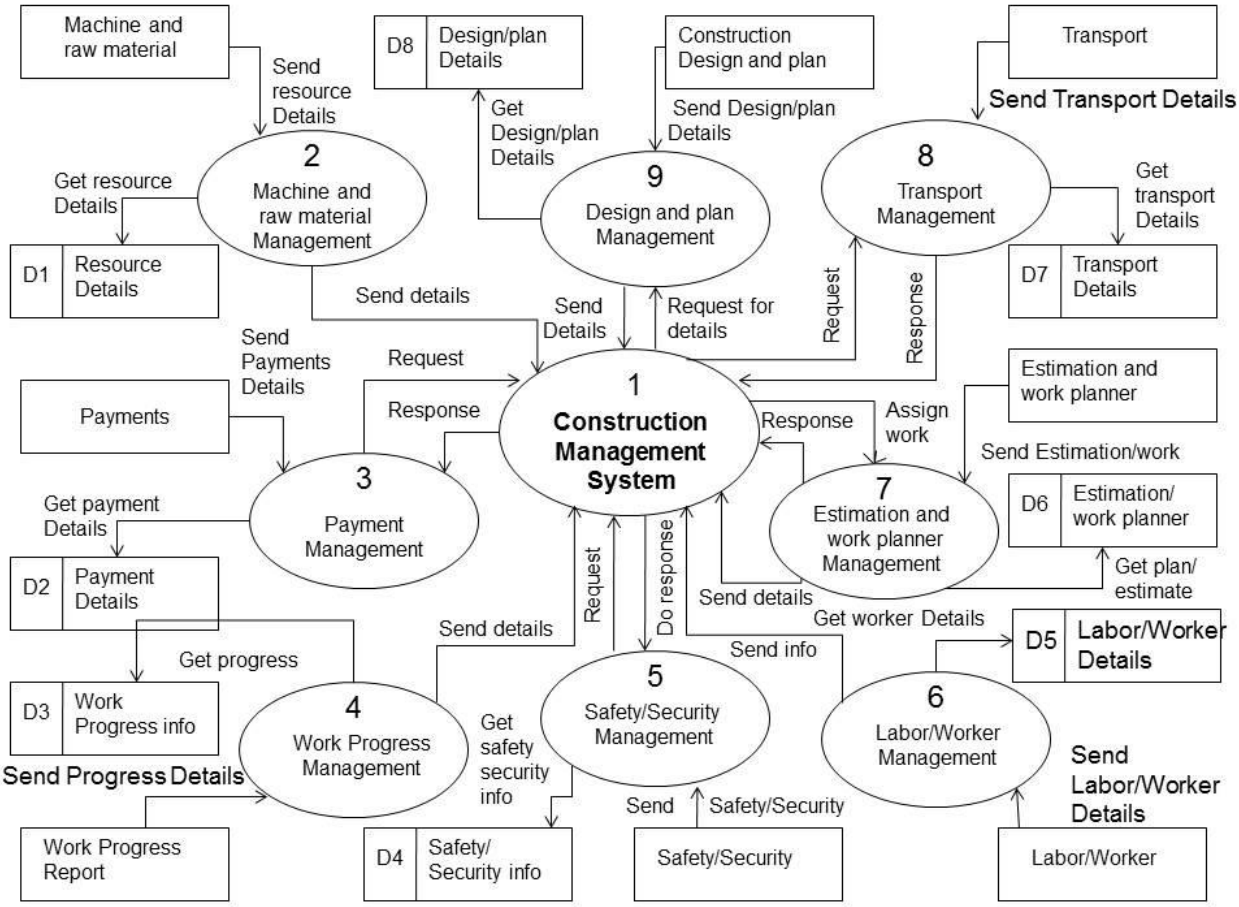
Above Figure depicts how the AMDD activities fit into the various iterations of the agile software development lifecycle. It's simply another way to show that an agile project begins with some initial modelling and that modelling still occurs in each construction's iteration.

Data models

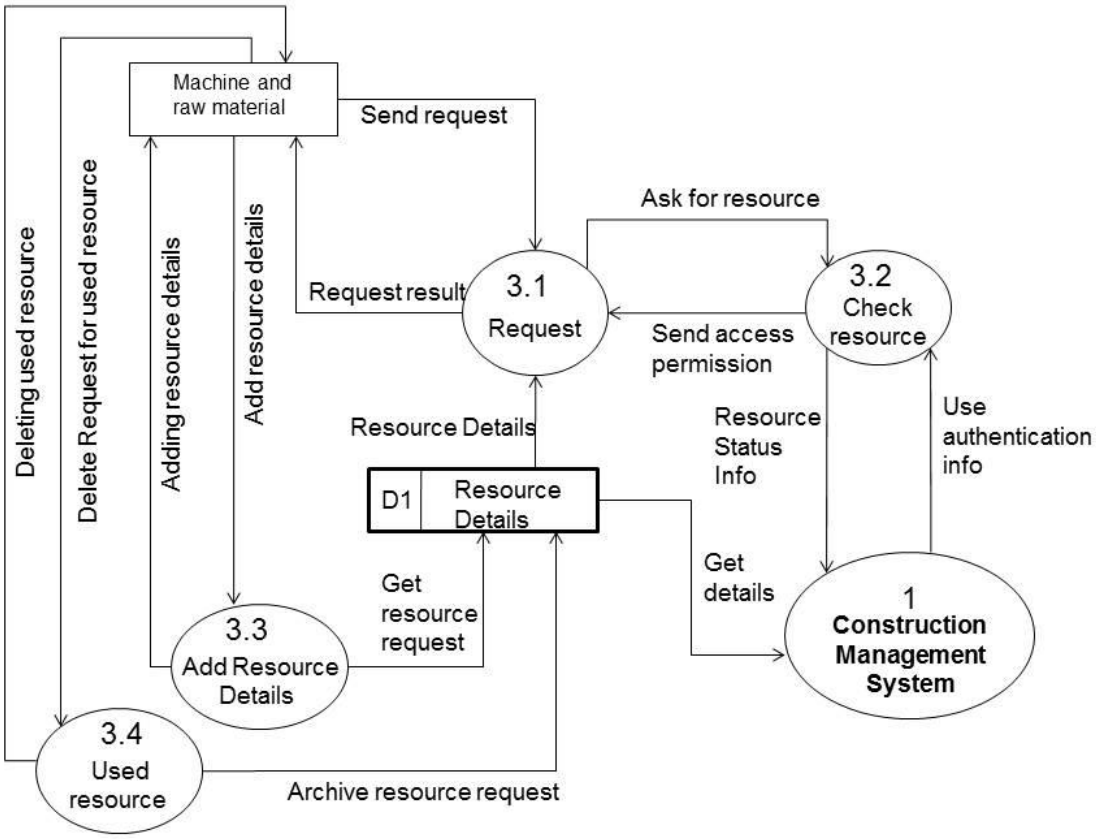
Context Diagram

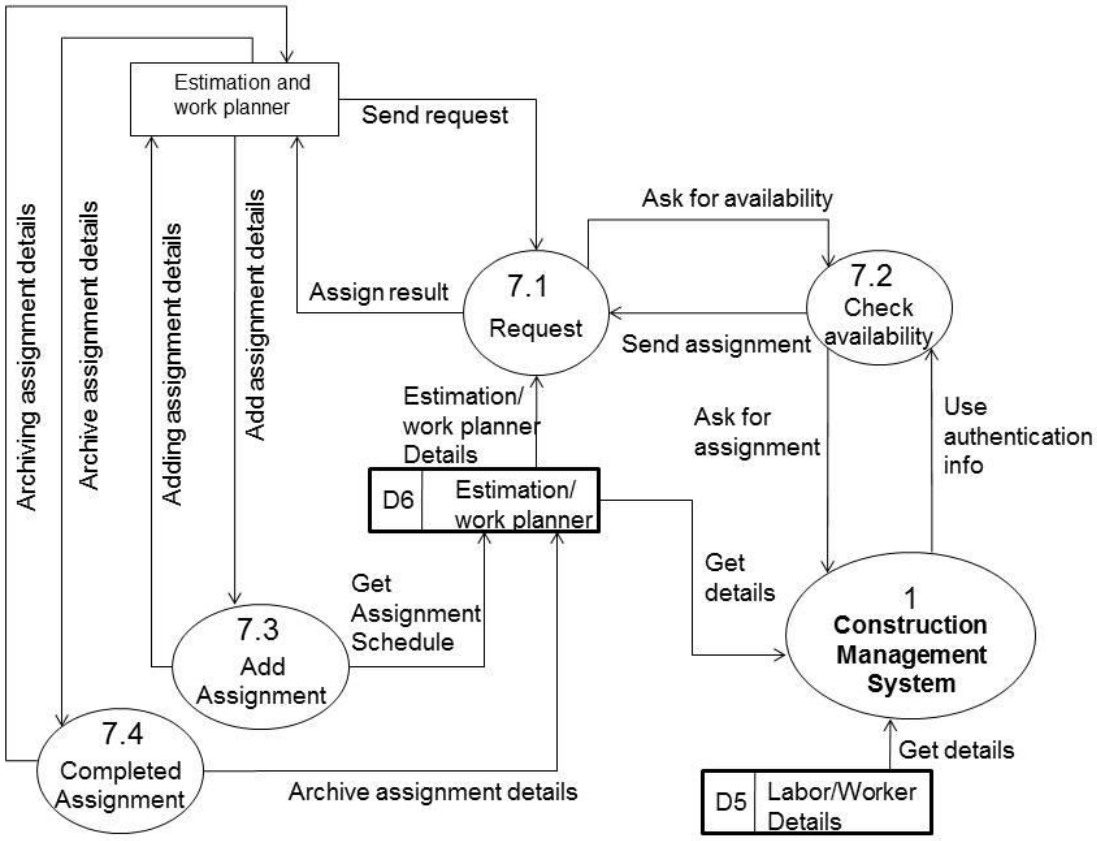


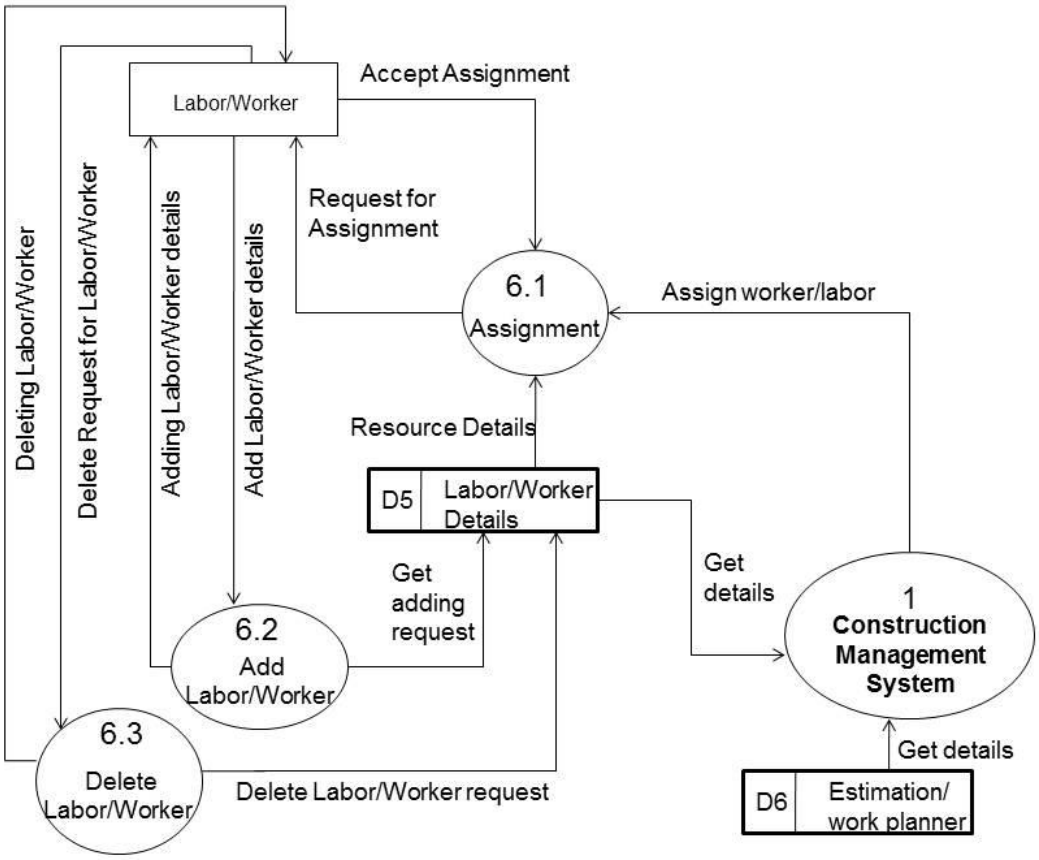
0-Level DFD


Machine and raw material
Send resource Details
D8
Design/plan Details
Construction Design and plan
Transport
Send Transport Details
Get Design/plan Details
Send Design/plan Details
2
Machine and raw material Management
9
Design and plan Management
8
Transport Management
Get resource Details
Get transport Details
D1
Resource Details
D7
Transport Details
Send details
Send Details
Request for details
Request
Response
Send Payments Details
Request
1
Construction Management System
Estimation and work planner
Payments
Response
Response
Assign work
Send Estimation/work
3
Payment Management
7
Estimation and work planner Management
D6
Estimation/ work planner
Get payment Details
Get plan/ estimate
D2
Payment Details
Send details
Request
Do response
Send details
Get worker Details
D5
Labor/Worker Details
Get progress
Send info
D3
Work Progress info
4
Work Progress Management
5
Safety/Security Management
6
Labor/Worker Management
Get safety security info
Send Labor/Worker Details
Send Progress Details
Send
Safety/Security
Work Progress Report
D4
Safety/ Security info
Safety/Security
Labor/Worker

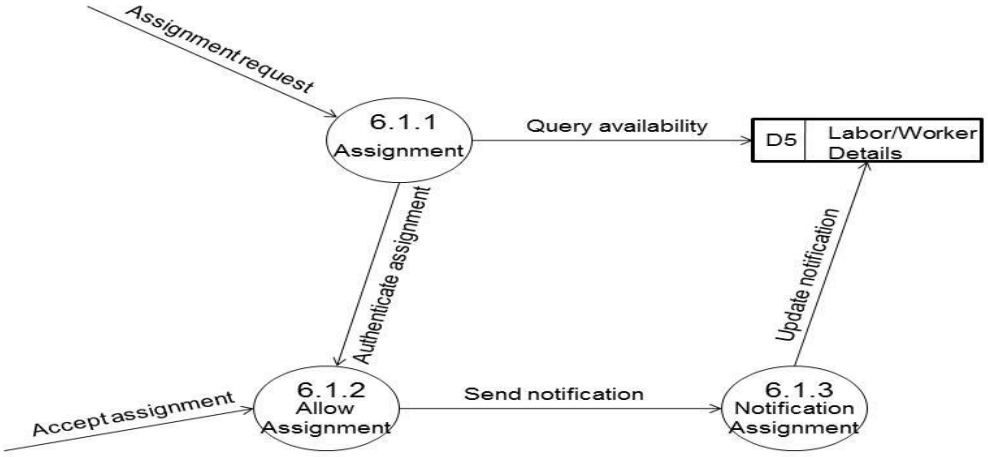
1-Level DFD


Machine and raw material
Send request
Ask for resource
Deleting used resource
Delete Request for used resource
Adding resource details
Add resource details
Request result
3.1 Request
3.2 Check resource
Send access permission
Resource Details
Resource Status Info
Use authentication info
D1 Resource Details
Get details
Get resource request
3.3 Add Resource Details
1 Construction Management System
3.4 Used resource
Archive resource request


Estimation and work planner
Send request
Ask for availability
Archiving assignment details
Archive assignment details
Adding assignment details
Add assignment details
Assign result
7.1
Request
7.2
Check availability
Send assignment
Estimation/ work planner Details
Ask for assignment
Use authentication info
D6
Estimation/ work planner
Get details
Get Assignment Schedule
7.3
Add Assignment
1
Construction Management System
7.4
Completed Assignment
Get details
Archive assignment details
D5
Labor/Worker Details

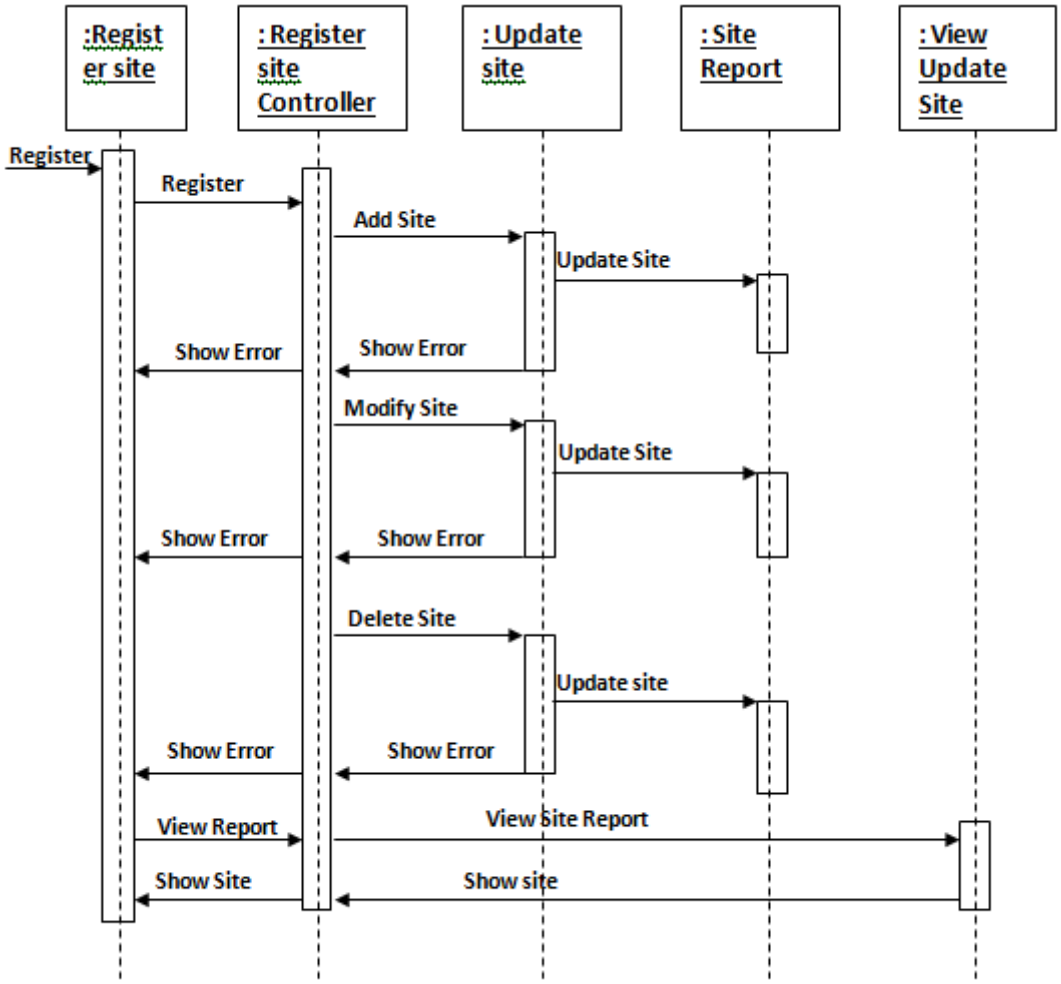

Labor/Worker
Accept Assignment
Request for Assignment
Deleting Labor/Worker
Delete Request for Labor/Worker
Adding Labor/Worker details
Add Labor/Worker details
6.1
Assignment
Assign worker/labor
Resource Details
D5
Labor/Worker Details
Get adding request
6.2
Add Labor/Worker
Get details
1
Construction Management System
6.3
Delete Labor/Worker
Delete Labor/Worker request
Get details
D6
Estimation/ work planner

2-Level DFD

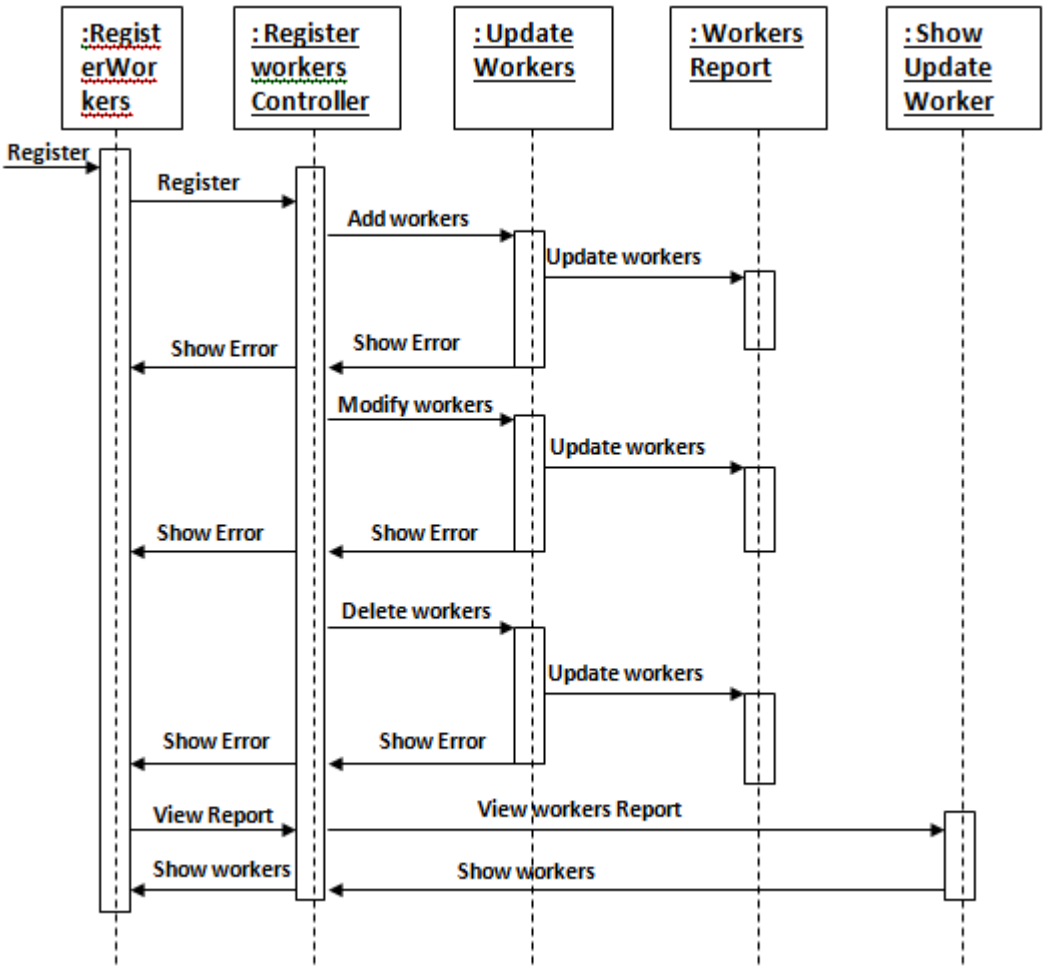


Sequence diagrams

Interaction Sites



Interaction Workers



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

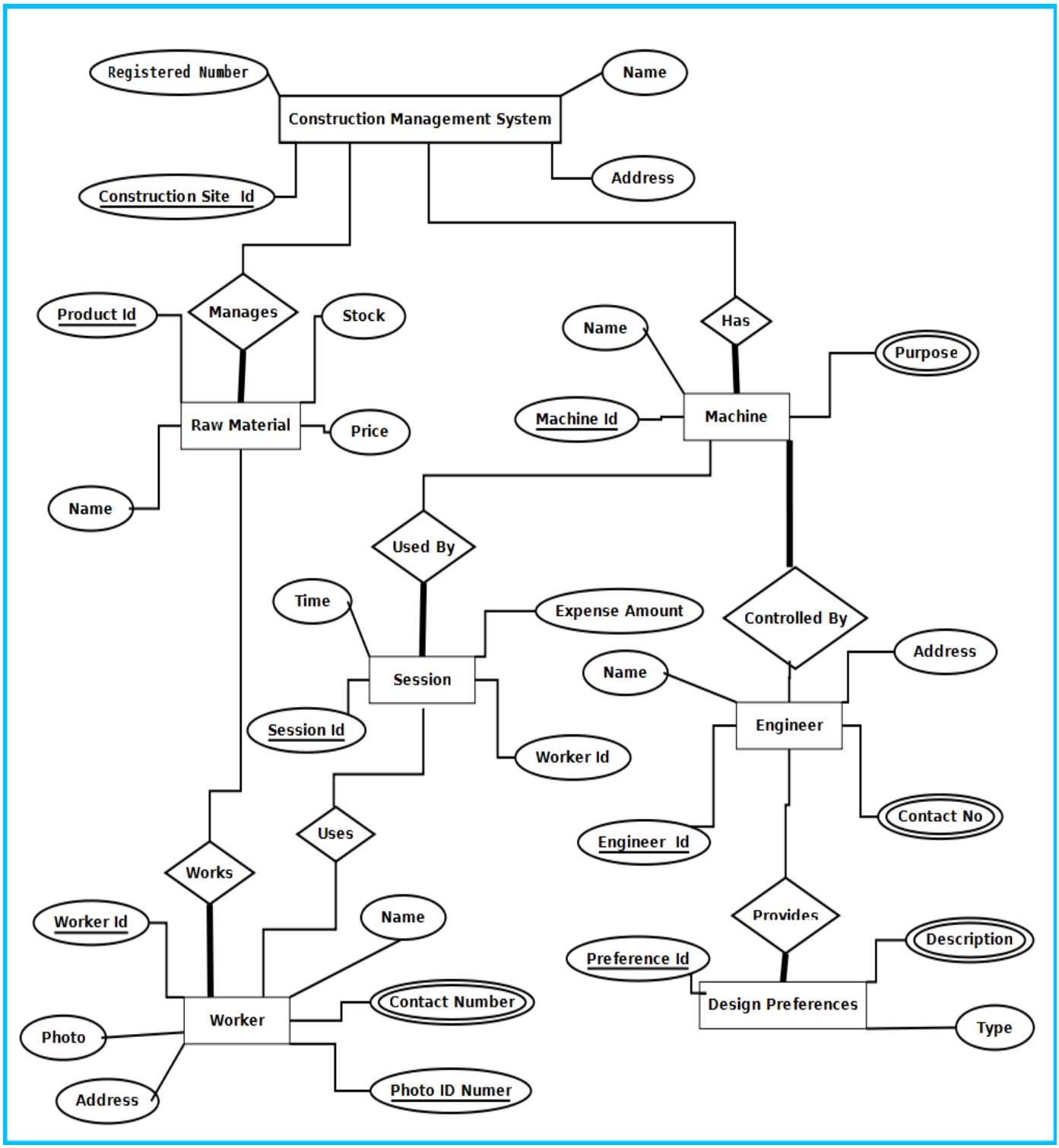
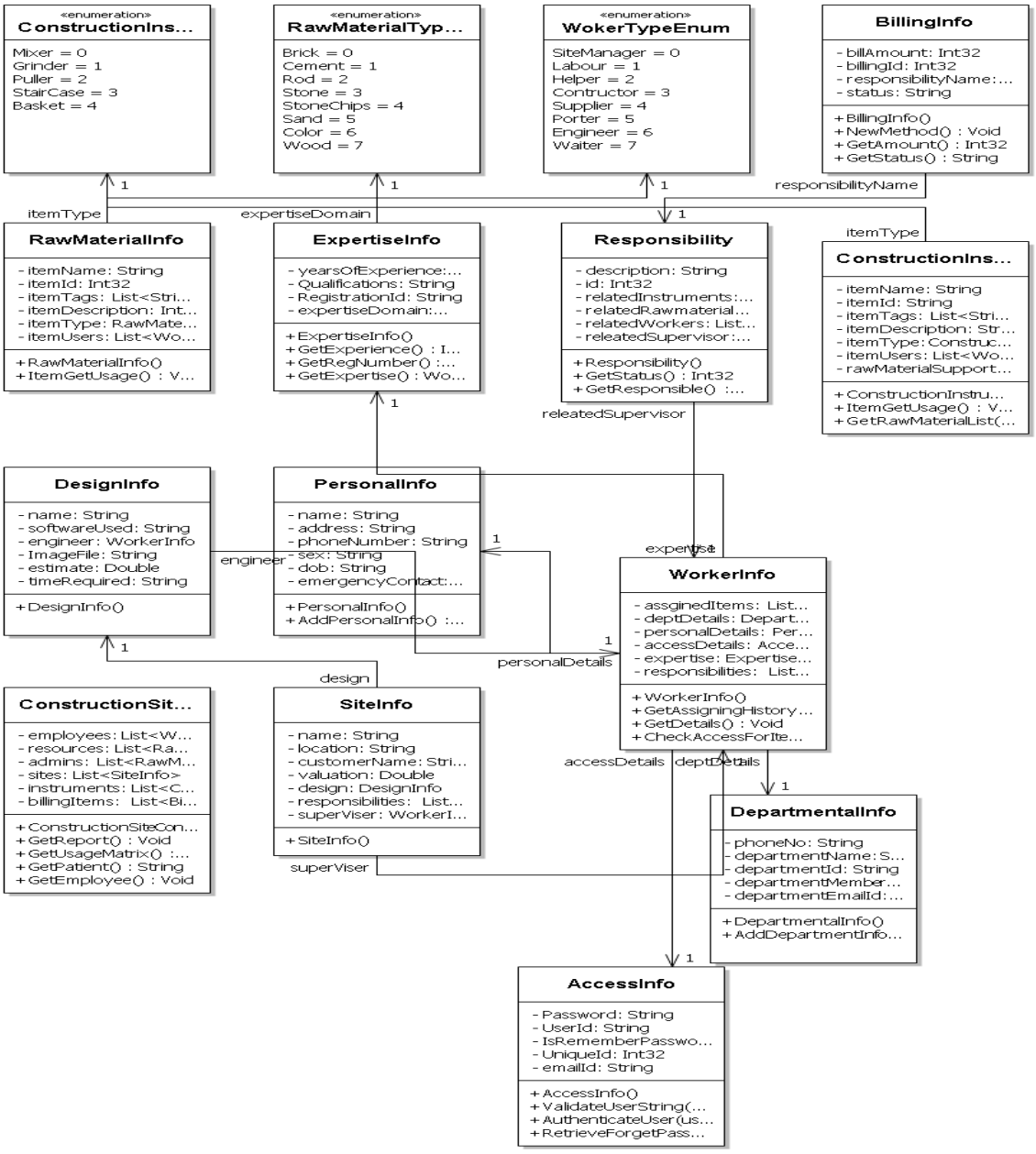
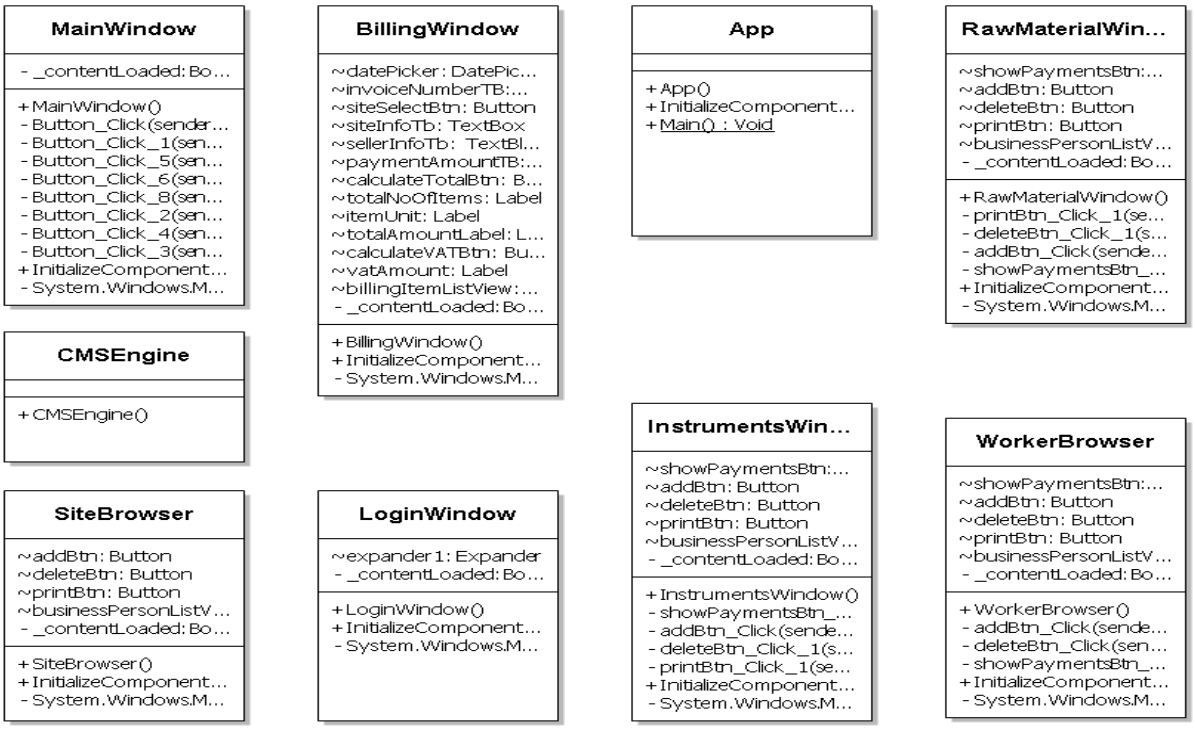


Figure: **E-R Diagram of Construction Management system**

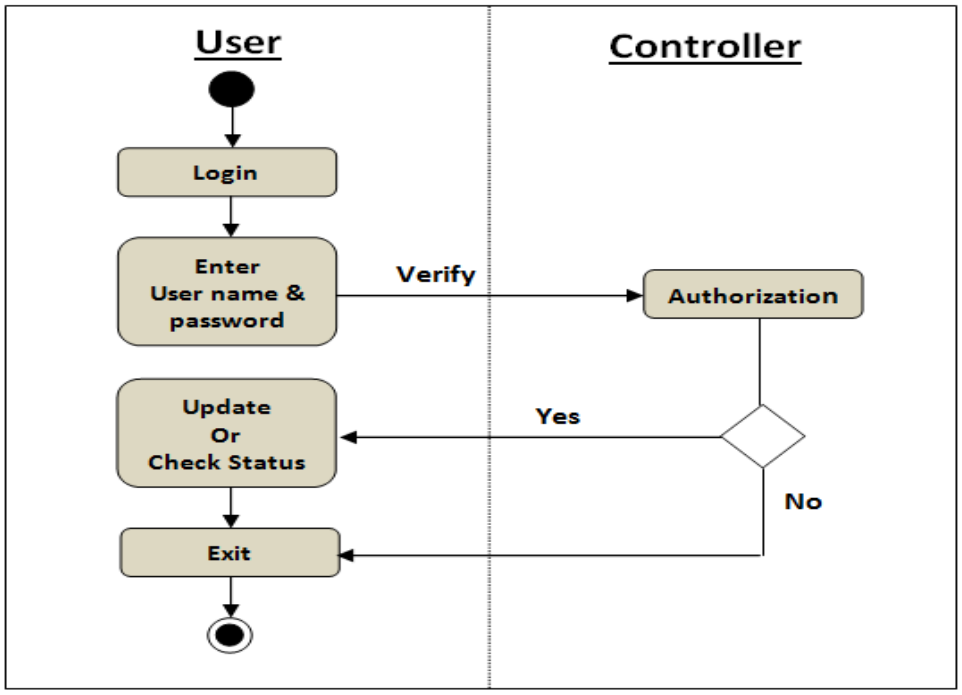
Class Diagram



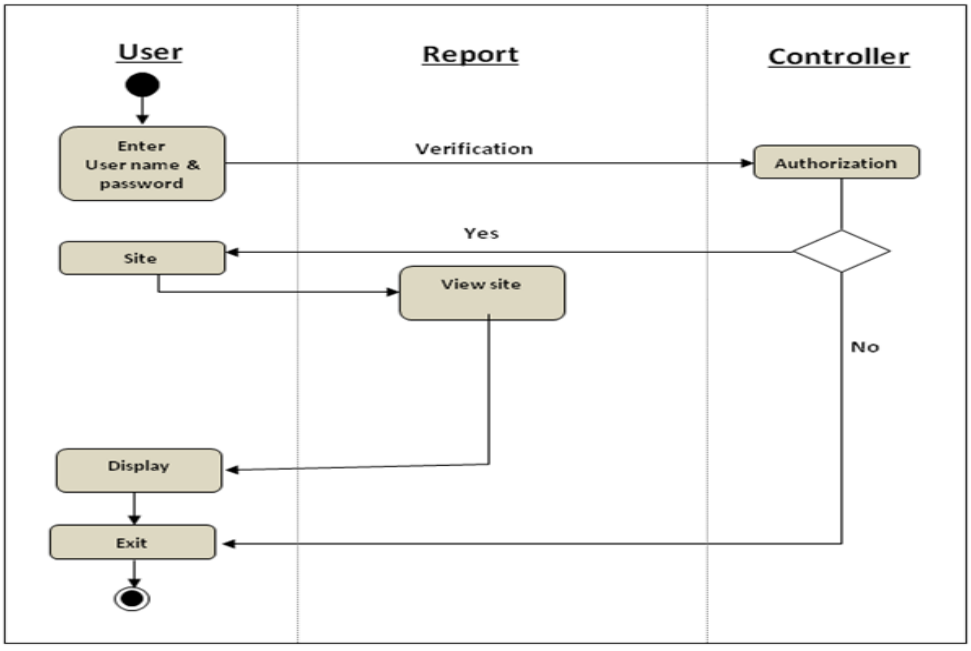


Activity Diagrams

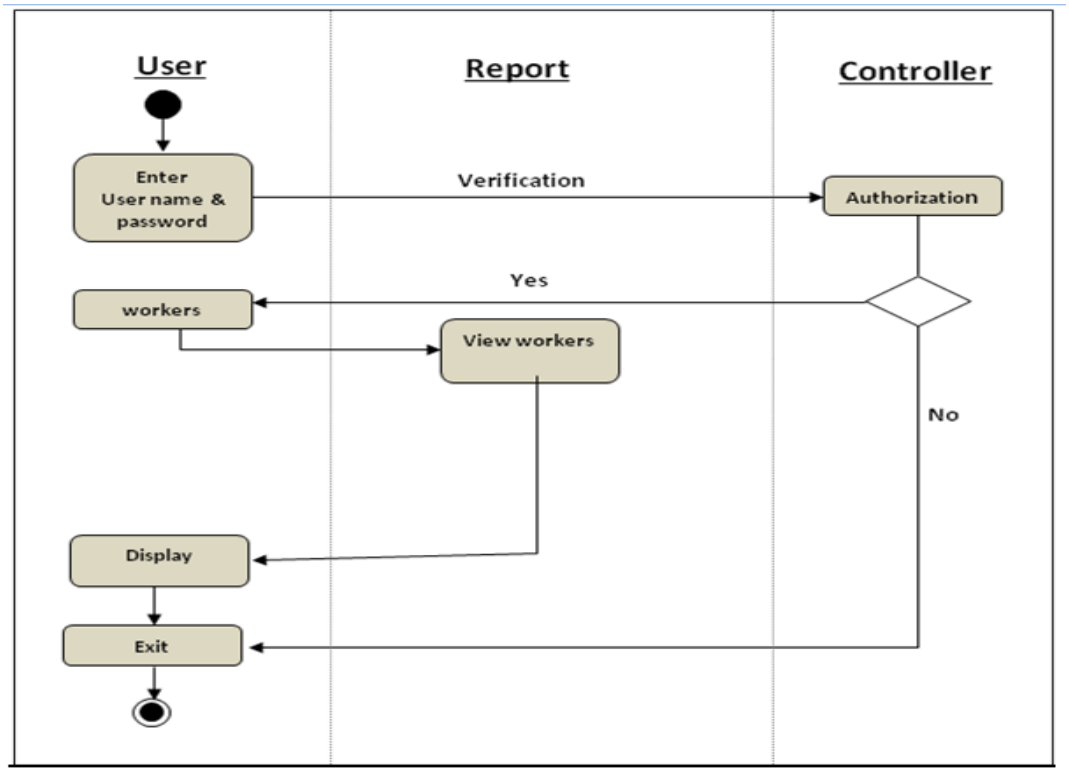
User Login



View site



View workers


User
Report
Controller
Enter
User name &
password
Verification
Authorization
Yes
workers
View workers
No
Display
Exit

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

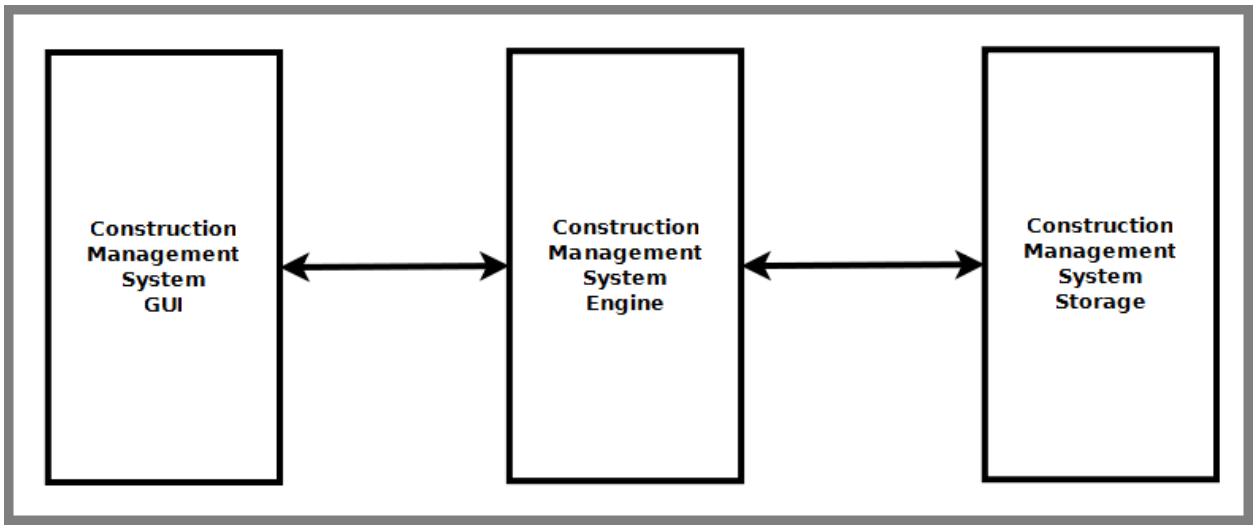


Figure: Components of Construction Management System

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

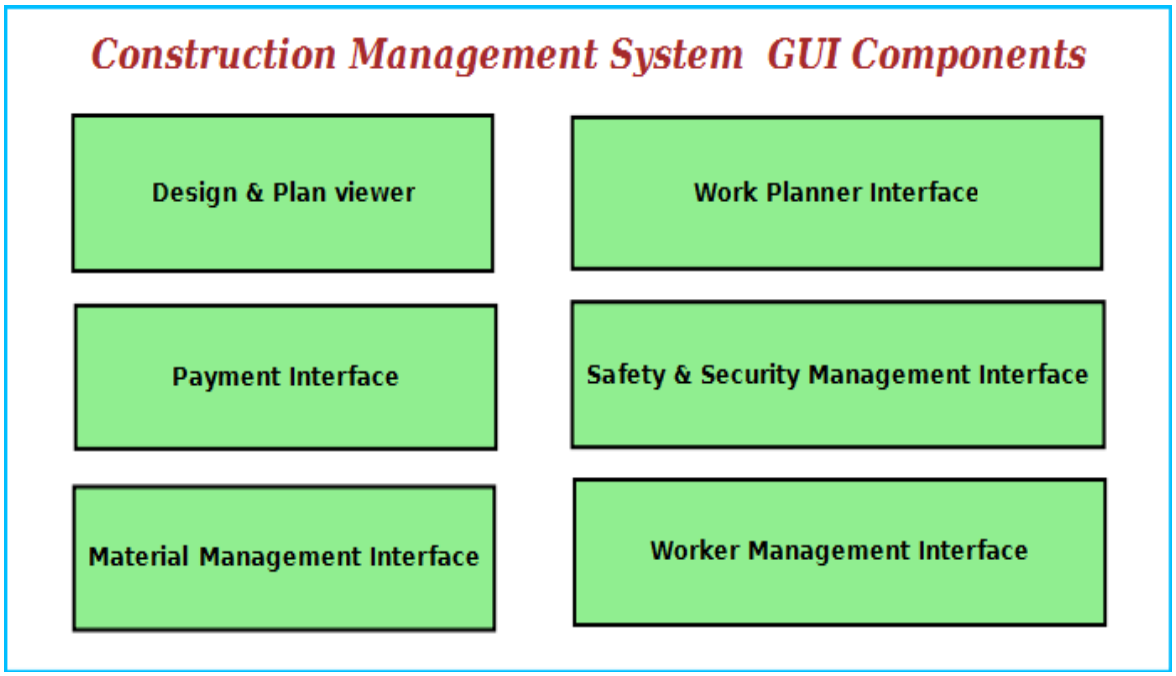


Figure: Construction Management System GUI Components

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

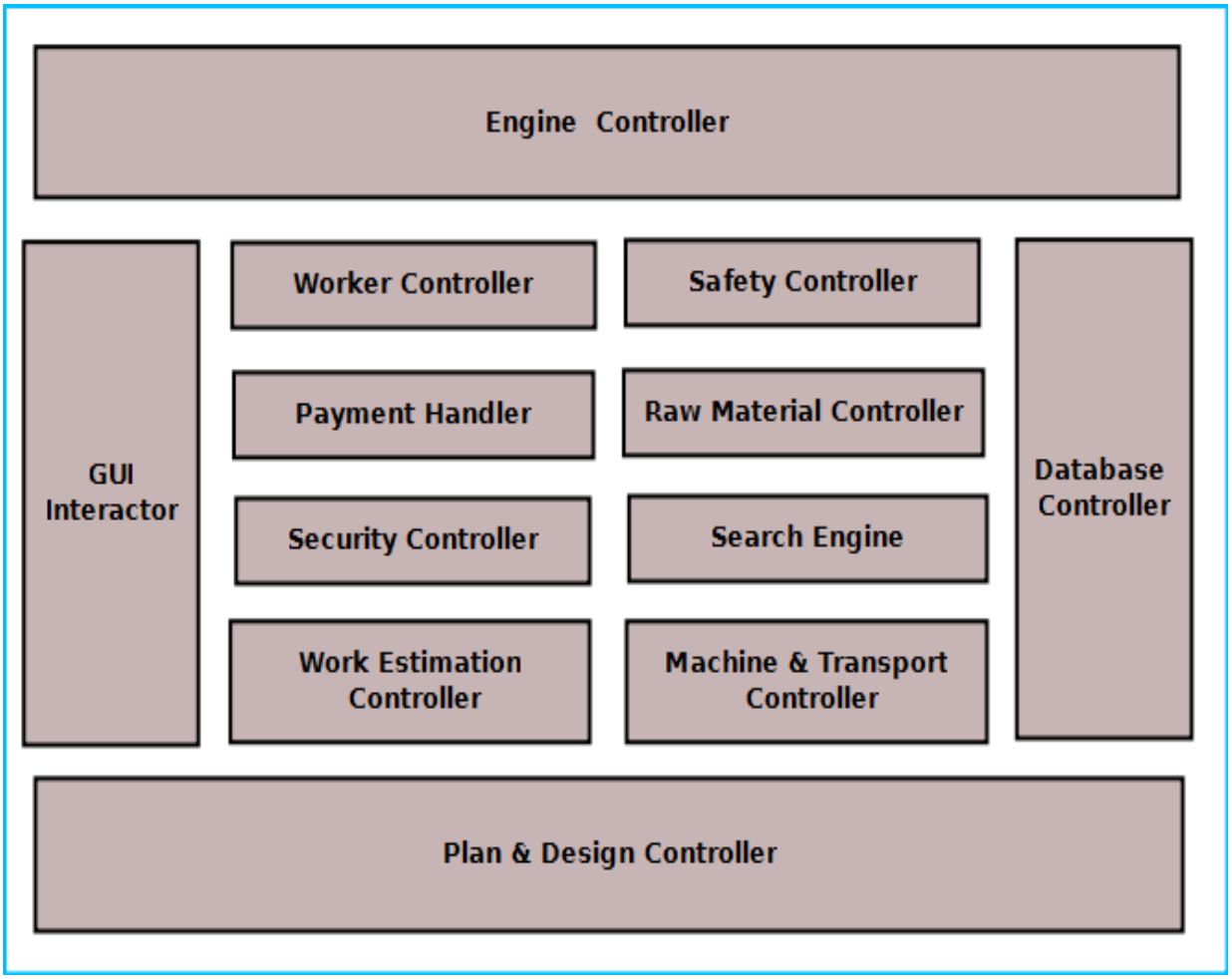


Figure: **Components of Engine**

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

	Task Name	Duratic	Start	Finish	Predecessors	Resource Names
1	CMS Desktop Client	5 days	Mon 3/3/14	Fri 3/7/14		Arindam
2	Construction Management Controller	10 days	Mon 3/10/14	Fri 3/21/14	1	Arindam
3	Worker Management	5 days	Mon 3/24/14	Fri 3/28/14	2	Arindam
4	Raw Material & Machine Management system	5 days	Mon 3/31/14	Fri 4/4/14	3	Arindam
5	Transport Management Syster	7.5 days	Wed 4/30/14	Fri 5/9/14		Arindam
6	Safety & Security Management System	30 days	Wed 4/30/14	Tue 6/10/14		Arindam
7	Payment Management Syster	5 days	Wed 6/11/14	Tue 6/17/14	6	Arindam
8	Construction & Plan Manager	5 days	Wed 6/18/14	Tue 6/24/14	7	Arindam
9	Estimation & Work Planner	4 days	Wed 6/11/14	Mon 6/16/14	6	Arindam
10	Legal Management system	6 days	Wed 6/11/14	Wed 6/18/14	6	Arindam
11	CMS Database	9 days	Mon 4/7/14	Thu 4/17/14	4	Arindam
12	Integration	10 days	Fri 4/18/14	Thu 5/1/14	1,4,11,2,3	Arindam
13	Integration Testing	15 days	Mon 6/30/14	Fri 7/18/14	1,4,11,12,2,3	Arindam
14	Bug Fix	15 days	Wed 6/25/14	Tue 7/15/14	8	Arindam
15	Documentation	5 days	Wed 7/16/14	Tue 7/22/14	14	Arindam

Database design

The database used for this software is called **CMS**db. Database tables and corresponding keys are shown in tabular form. It shows the tables and its columns. A key in underline is the primary key.

Screenshots of table structures:

Table: Worker

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	userId	int(10)			No	*None*		
	name	varchar(100)	latin1_swedish_ci		No	*None*		
	address	varchar(200)	latin1_swedish_ci		No	*None*		
	contactNumber	int(10)			No	*None*		
	skillset	varchar(100)	latin1_swedish_ci		No	*None*		
	photoIdNum	varchar(10)	latin1_swedish_ci		No	*None*		
	photo	blob		BINARY	No	*None*		

Table: ConstructionManagementSystem

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	constructionSiteId	int(10)			No	*None*		
	name	varchar(100)	latin1_swedish_ci		No	*None*		
	address	varchar(200)	latin1_swedish_ci		No	*None*		
	registeredNo	varchar(10)	latin1_swedish_ci		No	*None*		

Table: Construction Machine

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	machineId	int(10)			No	*None*		
	name	varchar(100)	latin1_swedish_ci		No	*None*		
	purpose	varchar(100)	latin1_swedish_ci		No	*None*		

Table: WorkSession

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	sessionId	int(10)			No	*None*		
	workerId	int(10)			No	*None*		
	workTime	datetime			No	*None*		
	expenseAmount	float			No	*None*		

Table: Engineer

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	engineerId	int(10)			No	None		
	name	varchar(100)	latin1_swedish_ci		No	None		
	address	varchar(200)	latin1_swedish_ci		No	None		
	contactNumber	int(12)			No	None		

Table: DesignPreference

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	preferenceId	int(11)			No	None		
	type	varchar(50)	latin1_swedish_ci		No	None		
	description	varchar(200)	latin1_swedish_ci		No	None		

Table: RawMaterial

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	productId	int(10)			No	None		
	stock	varchar(200)	latin1_swedish_ci		No	None		
	name	varchar(100)	latin1_swedish_ci		No	None		
	price	double			No	None		

Tables	Keys
Worker	**User Id**, Name, Address , Contact Number, skillset, Photo ID Num, Photo
ConstructionManagementSystem	**Construction site Id** , Name, Address, Registered no
Construction Machine	**Machine Id**, Name, purpose
WorkSession	**Session Id, worker** Id, Time, Expense amount
Engineer	**engineer Id**, Name, address, contact number
DesignPreference	**Preference Id,** Type, Description
RawMaterial	**Product id,** stock, name, price

User Interface Design

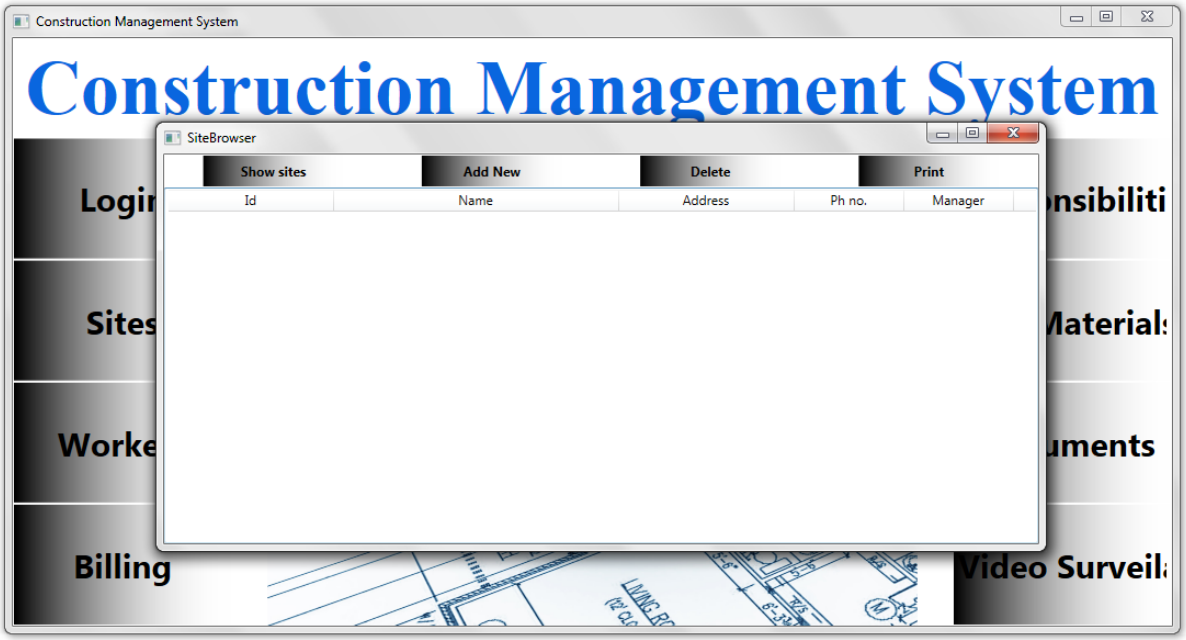
Main Window


Construction Management System
Construction Management System
Login
Sites
Workers
Billing
Responsibiliti
Raw Material
Instruments
Video Surveil

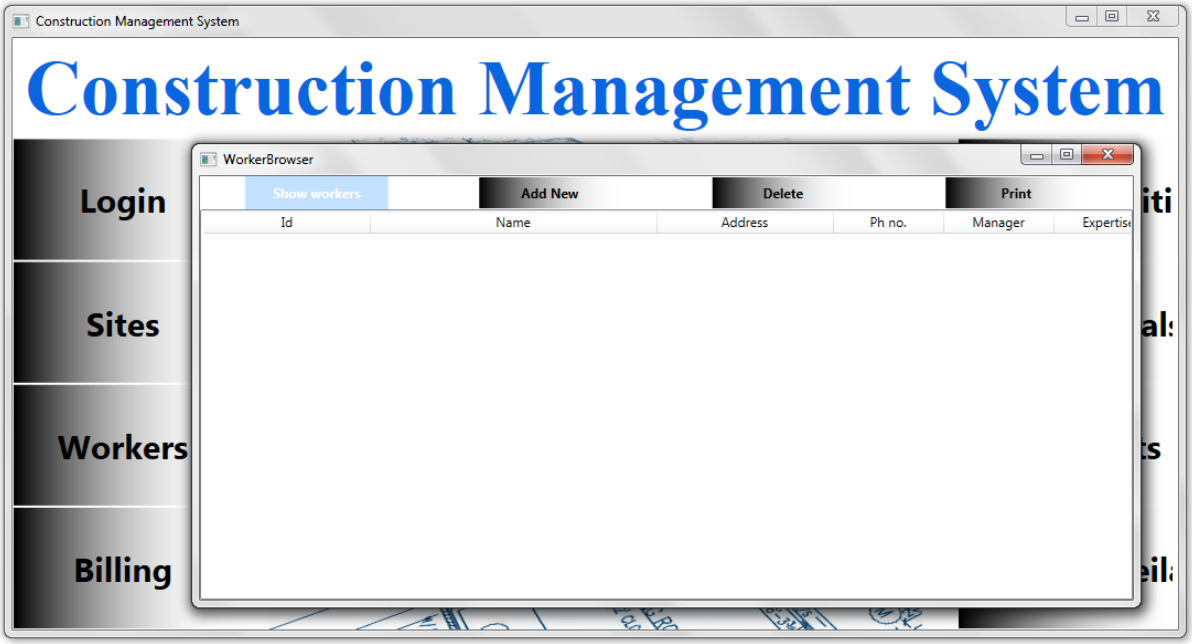
Login


Construction Management System
Construction Management System
Login
Sites
Workers
Billing
LoginWindow
Login
Name
Password
Forgot Password
Remember Password
Login
Responsibiliti
Raw Material
Instruments
Video Surveil

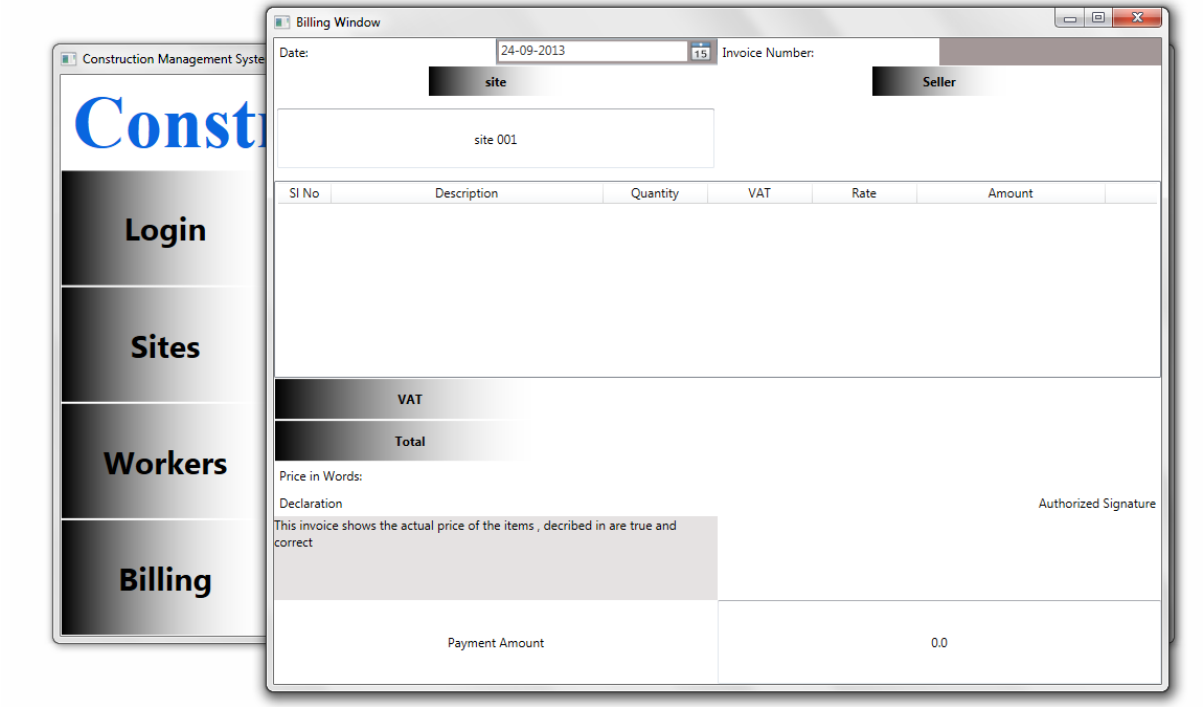
Sites


Construction Management System
Construction Management System
SiteBrowser
Show sites
Add New
Delete
Print
Id
Name
Address
Ph no.
Manager
Billing
Video Surveil

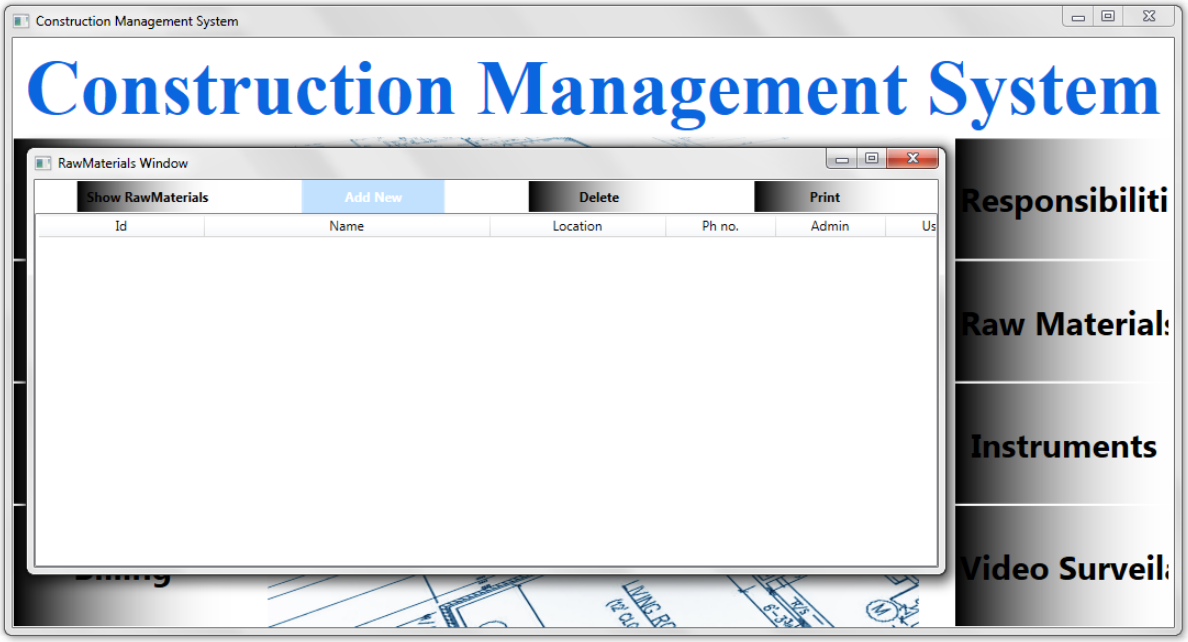
Workers


Construction Management System
Construction Management System
WorkerBrowser
Show workers
Add New
Delete
Print
Id
Name
Address
Ph no.
Manager
Login
Sites
Workers
Billing

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	anual	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	Step 1	Insert wrong User Id and Password. And Click on Login Button.	**CMS** will display error message. And Failed to Login.

					of user is valid.			
CMS-002						Step 2	Insert Wrong User Id and valid Password. And Click on Login Button.	**CMS** will display error message. And Failed to Login.
CMS-003						Step 3	Insert Valid User Id and Wrong Password. And Click on Login Button.	**CMS** will display error message. And Failed to Login.
CMS-004						Step 4	Insert Nothing in User Id and Password fields. And Click on Login Button.	**CMS** will display error message. And Failed to Login.
CMS-005						Step 5	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						Step 6	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						Step 7	Insert Nothing in User Id and insert invalid Password fields. And Click on	**CMS** will display error message. And Failed to Login.

							Login Button.	
CMS-008						Step 8	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						Step 9	Insert valid User Id and Password. And Click on Login Button.	Successfully login.
CMS-010	anual	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.	Step 1	Click on Sites And click show sites Button.	Successfully display site details.
CMS-011	anual	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.	Step 1	Click on Sites And click Add sites Button.	Successfully display Add site Window.
CMS-012						Step 2	Click on submit button without putting information.	Display error message.
CMS-013						Step 3	Insert the details in Field like Name, Address,	Successful to Add new site.

							Phone No., Manager etc. And click submit button.	
CMS-014	anual	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.	Step 1	Click on delete button without selecting site.	Display error message.
CMS-015						Step 2	Click on delete button by selecting site.	Deleted successfully.
CMS-016	anual	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.	Step 1	Click the print button.	The information of the site is successfully ready for print.
CMS-017	anual	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.	Step 1	Click on workers And click Show workers Button.	Successfully display workers details.
CMS-018	anual	f0657bbdf47e26ec481fa172b0fa76f9becb2627	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\cod	Check Successful Add workers in Sites of **CMS**.	The purpose of this test is to check the Add workers	Step 1	Click on workers And click Add workers Button.	Successfully display Add workers Window.

			e		is workers properly.			
CMS-019						Step 2	Click on submit button without putting information.	Display error message.
CMS-020						Step 3	Insert the details in Field like Name, Address, Phone No., Manager, Duty cycle etc. And click submit button.	Successful to Add new worker.
CMS-021	a n u a l	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.	Step 1	Click on delete button without selecting workers.	Display error message.
CMS-022						Step 2	Click on delete button by selecting workers.	Deleted successfully.
CMS-023	a n u a l	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.	Step 1	Click the print button.	The information of the workers is successfully ready for print.

CMS-024	anual	f0657bbdf47e26ec481fa172b0fa76f9becb2627	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successful Add Raw Materials in Sites of **CMS**.	The purpose of this test is to check the Add Raw Materials is workers properly.	Step 1	Click on Raw Materials And click Add Raw Materials Button.	Successfully display Add Raw Materials Window.
CMS-025						Step 2	Click on submit button without putting information.	Display error message.
CMS-026						Step 3	Insert the details in Field like Name, Usages etc. And click submit button.	Successful to Add new Raw Materials.
CMS-027	anual	f0657bbdf47e26ec481fa172b0fa76f9ffew2656	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successful delete Raw Materials in Raw Materials of **CMS**.	The purpose of this test is to check the Delete Raw Materials is work properly.	Step 1	Click on delete button without selecting Raw Materials.	Display error message.
CMS-028						Step 2	Click on delete button by selecting Raw Materials.	Deleted successfully.

CMS-029	anual	e22bd0e470f145f3db336ei0e28d474d8f4437d7	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successful print the Raw Materials details.	The purpose of this test is to verify that the print is work properly.	Step 1	Click the print button.	The information of the Raw Materials is successfully ready for print.
CMS-030	anual	f0657bbdf47e26ec481fa172b0fa76f9becb2640	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successful Show Instruments in workers of **CMS**.	The purpose of this test is to check the Show Instruments is work properly.	Step 1	Click on Instruments And click Show Instruments Button.	Successfully display Instruments details.
CMS-031	anual	f0657bbdf47e26ec481fa172b0fa76f9becb2627	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successful Add Instruments in Instruments of **CMS**.	The purpose of this test is to check the Add Instruments is work properly.	Step 1	Click on Instruments And click Add Instruments Button.	Successfully display Add Instruments Window.
CMS-032						Step 2	Click on submit button without putting information.	Display error message.
CMS-033						Step 3	Insert the details in Field like Instruments Name, Usages, etc. And click submit button.	Successful to Add new Instruments.

CMS-034	anual	f0657bbdf47e26ec481fa172b0fa76f9ffew2656	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successful delete Instruments in Instruments of **CMS**.	The purpose of this test is to check the Delete Instruments is work properly.	Step 1	Click on delete button without selecting Instruments.	Display error message.
CMS-035						Step 2	Click on delete button by selecting Instruments.	Deleted successfully.
CMS-036	anual	e22bd0e470f145f3db336ei0e28d474d8f4437d7	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successful print the Instruments details.	The purpose of this test is to verify that the print is work properly.	Step 1	Click the print button.	The information of the Instruments is successfully ready for print.
CMS-037	anual	e22bd0e470f145f3db336ei0e28d474d8f4437d7	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successful Billing works properly.	The purpose of this test is to verify that the Billing is work properly.	Step 1	Click the Billing from **CMS**.	The Billing window display successfully.
CMS-001						Step 2	Click on Add Item amd select the item with the details information and click on ok. Repeat this process until all	All Item Added in billing view.

							required item is added.	
CMS-048						Step 3	Click on Total.	Total amount display successfully.
CMS-049						Step 4	Click on VAT.	Total VAT amount display successfully.
CMS-050						Step 5	Click on Print.	Bill is ready for print.

System Test Cases

Test Case Id	type	Github ID	Subject	Test Name	Test Description	Step Name	Description	Expected Result
CMS-051	anual	f3563be0a9c431104f52839039e86043cf640cf1	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Log in.	It is to check that Login works properly.	Step 1	Click on Login button after inserting invalid User id and password from **CMS**.	Login failed to **CMS**. And can't able to use the feature.

CMS-052						Step 2	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-053	a n u a l	d01197ee4cd3bee9245874b5937ba740019fd131	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successfuly work of Sites on **CMS**.	The purpose of this test is to verify that the all function of Sites works properly.	Step 1	Click on Sites and Add New link.	New Site creation area is opened.
CMS-054						Step 2	Click on Submit button after inserting invalid information.	Adding failed to **CMS**. And can't able to use the feature.
CMS-055						Step 3	Click on Submit button after inserting valid information.	New Site added Successfully to **CMS**.
CMS-056						Step 4	Click on delete button without selecting Raw Materials.	Display error message.
CMS-057						Step 5	Click on delete button by selecting Raw Materials.	Deleted successfully.
CMS-058						Step 6	Click the print button.	The information of the site is successfully ready for print.

CMS-059	anual	d01197ee4cd3bee9245874b5937ba740019fr531	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successfuly work of workers on **CMS**.	The purpose of this test is to verify that the all function of workers works properly.	Step 1	Click on workers And click Show workers Button.	Successfully display workers details.
CMS-060						Step 2	Click on workers And click Add workers Button.	Successfully display Add workers Window.
CMS-061						Step 3	Click on submit button without putting information.	Display error message.
CMS-062						Step 4	Insert the details in Field like Name, Address, Phone No., Manager, Duty cycle etc. And click submit button.	Successful to Add new worker.
CMS-063						Step 5	Click on delete button without selecting workers.	Display error message.
CMS-064						Step 6	Click on delete button by selecting workers.	Deleted successfully.
CMS-065						Step 7	Click the print button.	The information of the workers is successfully ready for

								print.
CMS-066	manual	f0657bbdf47e26ec481fa172b0fa76f9becb2627	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successfuly work of Raw Materials on **CMS**.	The purpose of this test is to verify that the all function of Raw Materials works properly.	Step 1	Click on Raw Materials And click Add Raw Materials Button.	Successfully display Add Raw Materials Window.
CMS-067						Step 2	Click on submit button without putting information.	Display error message.
CMS-068						Step 3	Insert the details in Field like Name, Usages etc. And click submit button.	Successful to Add new Raw Materials.
CMS-069						Step 4	Click on delete button without selecting Raw Materials.	Display error message.
CMS-070						Step 5	Click on delete button by selecting Raw Materials.	Deleted successfully.
CMS-071						Step 6	Click the print button.	The information of the Raw Materials is successfully ready for print.

CMS-072	anual	f0657bbdf47e26ec481fa172b0fa76f9becb2627	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successfuly work of Instruments on **CMS**.	The purpose of this test is to verify that the all function of Instruments works properly.	Step 1	Click on Instruments And click Show Instruments Button.	Successfully display Instruments details.
CMS-073						Step 2	Click on Instruments And click Add Instruments Button.	Successfully display Add Instruments Window.
CMS-074						Step 3	Click on submit button without putting information.	Display error message.
CMS-075						Step 4	Insert the details in Field like Instruments Name, Usages, etc. And click submit button.	Successful to Add new Instruments.
CMS-076						Step 5	Click on delete button without selecting Instruments.	Display error message.
CMS-077						Step 6	Click on delete button by selecting Instruments.	Deleted successfully.
CMS-078						Step 7	Click the print button.	The information of the Instruments is successfully ready for print.

CMS-079	manual	f0657bbdf47e26ec481fa172b0fa76f9becb2627	E:\DEVELOPERS_ZONE\GitHub\ConstructionManagementSystem\code	Check Successfuly work of Billing on **CMS**.	The purpose of this test is to verify that the all function of Billing works properly.	Step 1	Click the Billing from **CMS**.	The Billing window display successfully.
CMS-080						Step 2	Click on Add Item amd select the item with the details information and click on ok. Repeat this process until all required item is added.	All Item Added in billing view.
CMS-081						Step 3	Click on Total.	Total amount display successfully.
CMS-082						Step 4	Click on VAT.	Total VAT amount display successfully.
CMS-083						Step 5	Click on Print.	Bill is ready for print.

Coding

Complete Project Coding

CMS GUI Design Coding: **CMS**GUI

MainWindow.xaml

```
    <Window x:Class="CMSGUI.MainWindow"
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            Title="Construction Management System"  WindowState="Minimized">

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

        <DockPanel LastChildFill="True">
            <UniformGrid DockPanel.Dock="Top" Columns="1">
                <Label Name="lable1" VerticalContentAlignment="Center"
HorizontalContentAlignment="Center" FontFamily="Times New Roman" FontSize="70"
Foreground="#FF0966DF"
                  Content="Construction Management System" FontWeight="Bold"></Label>
            </UniformGrid>

            <UniformGrid DockPanel.Dock="Right" Columns="1">
                <Button  Style="{StaticResource PlainBtnStyle}"
Click="Button_Click_8">Responsibilities</Button>
                <Button  Style="{StaticResource PlainBtnStyle}"
Click="Button_Click_3">Raw Materials</Button>
                <Button  Style="{StaticResource PlainBtnStyle}"
Click="Button_Click_2">Instruments</Button>
                <Button  Style="{StaticResource PlainBtnStyle}"
Click="Button_Click_4">Video Surveilance</Button>
            </UniformGrid>


            <UniformGrid DockPanel.Dock="Left" Columns="1">
                <Button  Style="{StaticResource PlainBtnStyle}"
Click="Button_Click_1">Login</Button>
                <Button  Style="{StaticResource PlainBtnStyle}"
Click="Button_Click">Sites</Button>
                <Button  Style="{StaticResource PlainBtnStyle}"
Click="Button_Click_5">Workers</Button>
```

```
                <Button  Style="{StaticResource PlainBtnStyle}"
Click="Button_Click_6">Billing</Button>
            </UniformGrid>

            <UniformGrid DockPanel.Dock="Left" Columns="1">
                <Image Source="/CMSGUI;component/Images/13594766413d.jpg" />
            </UniformGrid>


        </DockPanel>


    </Window>
```

MainWindow.xaml.cs

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

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

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            SiteBrowser sb = new SiteBrowser();
            sb.ShowDialog();
        }

        private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            LoginWindow lw = new LoginWindow();
            lw.ShowDialog();
        }
```

```
        private void Button_Click_5(object sender, RoutedEventArgs e)
        {
            WorkerBrowser wb = new WorkerBrowser();
            wb.ShowDialog();
        }

        private void Button_Click_6(object sender, RoutedEventArgs e)
        {
            BillingWindow bw = new BillingWindow();
            bw.ShowDialog();
        }

        private void Button_Click_8(object sender, RoutedEventArgs e)
        {

        }

        private void Button_Click_2(object sender, RoutedEventArgs e)
        {
            InstrumentsWindow inw = new InstrumentsWindow();
            inw.ShowDialog();
        }

        private void Button_Click_4(object sender, RoutedEventArgs e)
        {

        }

        private void Button_Click_3(object sender, RoutedEventArgs e)
        {
            RawMaterialWindow rmw = new RawMaterialWindow();
            rmw.ShowDialog();
        }
    }
}
```

BillingWindow.xaml

```
<Window x:Class="CMSGUI.BillingWindow"
        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"
        Title="Billing Window" Height="857" Width="864">

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

```
        <DockPanel>
            <UniformGrid DockPanel.Dock="Top" Rows="1">
                <Label Content="Date:"></Label>
                <DatePicker Height="25" Name="datePicker" Background="#FFA39797"
SelectedDate="{x:Static sys:DateTime.Now}" />
                <Label Content="Invoice Number:"></Label>
                <TextBlock Name="invoiceNumberTB" Background="#FFA39797"></TextBlock>
            </UniformGrid>
            <UniformGrid Rows="1" Height="30" DockPanel.Dock="Top">
                <Button  Style="{StaticResource BtnStyle}" Content="site"
Name="siteSelectBtn"></Button>
                <Button  Style="{StaticResource BtnStyle}" Content="Seller" ></Button>
            </UniformGrid>
            <UniformGrid Height="80"  DockPanel.Dock="Top" Rows="1">
                <TextBox Name="siteInfoTb" VerticalAlignment="Center"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center" Margin="3"
Height="57">site 001</TextBox>
                <TextBlock Name="sellerInfoTb"  Margin="3"
TextWrapping="Wrap"></TextBlock>
            </UniformGrid>
            <UniformGrid Height="160"  DockPanel.Dock="Bottom" Rows="2">
                <TextBlock Background="#FFE5E2E2" TextWrapping="Wrap">This invoice
shows the actual price of the items , decribed in are true and correct</TextBlock>
                <Label></Label>
                <Label HorizontalContentAlignment="Center"
VerticalContentAlignment="Center">Payment Amount</Label>
                <TextBox Name="paymentAmountTB"  HorizontalContentAlignment="Center"
VerticalContentAlignment="Center">0.0</TextBox>

            </UniformGrid>
            <UniformGrid DockPanel.Dock="Bottom" Rows="1">
                <Label>Declaration</Label>
                <Label HorizontalContentAlignment="Right">Authorized Signature</Label>
            </UniformGrid>
            <DockPanel DockPanel.Dock="Bottom" LastChildFill="True">
                <Label>Price in Words:</Label>
                <Label></Label>
            </DockPanel>

            <DockPanel DockPanel.Dock="Bottom" Height="40">
                <Button  Style="{StaticResource BtnStyle}" DockPanel.Dock="Left"
Name="calculateTotalBtn" Content="Total" Width="260"
HorizontalContentAlignment="Right" FontWeight="Bold"></Button>
                <Label  DockPanel.Dock="Left"  Name="totalNoOfItems"
Width="100"></Label>
                <Label DockPanel.Dock="Left"  Name="itemUnit" Width="300"
HorizontalAlignment="Left" ></Label>
                <Label  DockPanel.Dock="Left" Name="totalAmountLabel"
Width="100"></Label>
            </DockPanel>
            <DockPanel DockPanel.Dock="Bottom" Height="40">
                <Button  Style="{StaticResource BtnStyle}" DockPanel.Dock="Left"
Content="VAT" Name="calculateVATBtn" Width="260" HorizontalContentAlignment="Right"
```

```
FontWeight="Bold"></Button>
                <Label DockPanel.Dock="Left"  Width="100"></Label>
                <Label DockPanel.Dock="Left" Width="300" HorizontalAlignment="Left"
></Label>
                <Label  DockPanel.Dock="Left" Name="vatAmount" Width="100"></Label>
            </DockPanel>
            <ListView Name="billingItemListView" DockPanel.Dock="Bottom"
ItemsSource="{Binding billingCollection}">
                <ListView.View>
                    <GridView>
                        <GridViewColumn Width="50" Header="Sl No"
DisplayMemberBinding="{Binding serialNo}"  />
                        <GridViewColumn Width="260" Header="Description"
DisplayMemberBinding="{Binding description}"  />
                        <GridViewColumn Width="100" Header="Quantity"
DisplayMemberBinding="{Binding quantity}"  />
                        <GridViewColumn Width="100" Header="VAT"
DisplayMemberBinding="{Binding vat}"  />
                        <GridViewColumn Width="100" Header="Rate"
DisplayMemberBinding="{Binding rate}"  />
                        <GridViewColumn Width="180" Header="Amount"
DisplayMemberBinding="{Binding amount}"  />
                    </GridView>
                </ListView.View>
            </ListView>

        </DockPanel>
    </Window>
```

InstrumentsWindow.xaml

```
    <Window x:Class="CMSGUI.InstrumentsWindow"
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            Title="Instruments Window" Height="307" Width="925">

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


        <DockPanel LastChildFill="True">
            <UniformGrid Height="30"  DockPanel.Dock="Top" Rows="1">
                <Button Style="{StaticResource BtnStyle}" Content="Show instruments"
Name="showPaymentsBtn" Click="showPaymentsBtn_Click"  Margin="10 0" ></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Add New"
Name="addBtn" Click="addBtn_Click"  Margin="10 0" ></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Delete" Margin="10
0" Name="deleteBtn" Click="deleteBtn_Click_1"></Button>
```

```
                <Button Style="{StaticResource BtnStyle}" Content="Print" Margin="10
0" Name="printBtn" IsEnabled="True" Click="printBtn_Click_1"></Button>
            </UniformGrid>
            <ListView Name="businessPersonListView" DockPanel.Dock="Bottom"
                      IsSynchronizedWithCurrentItem="True">
                <ListView.View>
                    <GridView>
                        <!-- GridViewColumn Width="50" Header="Sl No"
DisplayMemberBinding="{Binding serialNo}"  /-->
                        <GridViewColumn Width="150" Header="Id"
DisplayMemberBinding="{Binding instrumentId}"  />
                        <GridViewColumn Width="260" Header="Name"
DisplayMemberBinding="{Binding instrumentName}"/>
                        <GridViewColumn Width="160" Header="Location"
DisplayMemberBinding="{Binding instrumentAdress}"  />
                        <GridViewColumn Width="100" Header="Ph no."
DisplayMemberBinding="{Binding phoneNumber}"  />
                        <GridViewColumn Width="100" Header="Admin"
DisplayMemberBinding="{Binding instrumentVatNo}"  />
                        <GridViewColumn Width="100" Header="Usage"
DisplayMemberBinding="{Binding instrumentTurnOver}"  />
                    </GridView>
                </ListView.View>
            </ListView>
        </DockPanel>
    </Window>
```

LoginWindow.xaml

```
    <Window x:Class="CMSGUI.LoginWindow"
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            Title="LoginWindow" Height="462" Width="516">

        <Window.Resources>
            <ResourceDictionary>
                <ResourceDictionary.MergedDictionaries>
                    <ResourceDictionary
Source="/CMSCommonStyle;component/Commonstyle.xaml" />
                </ResourceDictionary.MergedDictionaries>
            </ResourceDictionary>
        </Window.Resources>
        <Expander IsExpanded="True" Background="Gray" Header="Login" Name="expander1"
DockPanel.Dock="Left" >
            <UniformGrid Rows="4" HorizontalAlignment="Stretch"
VerticalAlignment="Stretch">
                <Label>Name</Label>
                <TextBox Width="200" Height="40"></TextBox>
                <Label>Password</Label>
                <PasswordBox  Width="200" Height="40"></PasswordBox>
                <Label >Forgot Password</Label>
                <CheckBox>Remember Password</CheckBox>
                <Label></Label>
```

```
                <Button Style="{StaticResource BtnStyle}" Width="200"
Height="40">Login</Button>
            </UniformGrid>
        </Expander>
    </Window>
```

RawMaterialWindow.xaml

```
    <Window x:Class="CMSGUI.RawMaterialWindow"
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            Title="RawMaterials Window" Height="307" Width="925">

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


        <DockPanel LastChildFill="True">
            <UniformGrid Height="30"  DockPanel.Dock="Top" Rows="1">
                <Button Style="{StaticResource BtnStyle}" Content="Show RawMaterials"
Name="showPaymentsBtn" Click="showPaymentsBtn_Click"  Margin="10 0" ></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Add New"
Name="addBtn" Click="addBtn_Click"  Margin="10 0" ></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Delete" Margin="10
0" Name="deleteBtn" Click="deleteBtn_Click_1"></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Print" Margin="10
0" Name="printBtn" IsEnabled="True" Click="printBtn_Click_1"></Button>
            </UniformGrid>
            <ListView Name="businessPersonListView" DockPanel.Dock="Bottom"
                      IsSynchronizedWithCurrentItem="True">
                <ListView.View>
                    <GridView>
                        <!-- GridViewColumn Width="50" Header="Sl No"
DisplayMemberBinding="{Binding serialNo}"  /-->
                        <GridViewColumn Width="150" Header="Id"
DisplayMemberBinding="{Binding instrumentId}"  />
                        <GridViewColumn Width="260" Header="Name"
DisplayMemberBinding="{Binding instrumentName}"/>
                        <GridViewColumn Width="160" Header="Location"
DisplayMemberBinding="{Binding instrumentAdress}"  />
                        <GridViewColumn Width="100" Header="Ph no."
DisplayMemberBinding="{Binding phoneNumber}"  />
                        <GridViewColumn Width="100" Header="Admin"
DisplayMemberBinding="{Binding instrumentVatNo}"  />
                        <GridViewColumn Width="100" Header="Usage"
DisplayMemberBinding="{Binding instrumentTurnOver}"  />
                    </GridView>
                </ListView.View>
```

```
            </ListView>
        </DockPanel>
    </Window>
```

SiteBrowser.xaml

```
    <Window x:Class="CMSGUI.SiteBrowser"
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            Title="SiteBrowser" Height="303" Width="1023">

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


        <DockPanel LastChildFill="True">
            <UniformGrid Height="30"  DockPanel.Dock="Top" Rows="1">
                <Button Style="{StaticResource BtnStyle}" Content="Show sites"
Margin="10 0" ></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Add New"
Name="addBtn"  Margin="10 0" ></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Delete" Margin="10
0" Name="deleteBtn" ></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Print" Margin="10
0" Name="printBtn" IsEnabled="True" ></Button>
            </UniformGrid>
            <ListView Name="businessPersonListView" DockPanel.Dock="Bottom"
                      IsSynchronizedWithCurrentItem="True">
                <ListView.View>
                    <GridView>
                        <!-- GridViewColumn Width="50" Header="Sl No"
DisplayMemberBinding="{Binding serialNo}"  /-->
                        <GridViewColumn Width="150" Header="Id"
DisplayMemberBinding="{Binding siteId}"  />
                        <GridViewColumn Width="260" Header="Name"
DisplayMemberBinding="{Binding siteName}"/>
                        <GridViewColumn Width="160" Header="Address"
DisplayMemberBinding="{Binding siteAdress}"  />
                        <GridViewColumn Width="100" Header="Ph no."
DisplayMemberBinding="{Binding phoneNumber}"  />
                        <GridViewColumn Width="100" Header="Manager"
DisplayMemberBinding="{Binding siteVatNo}"  />
                        <GridViewColumn Width="100" Header="Design"
DisplayMemberBinding="{Binding siteTurnOver}"  />
                        <GridViewColumn Width="100" Header="Turn Over"
DisplayMemberBinding="{Binding siteDue}"  />
                    </GridView>
                </ListView.View>
```

```
            </ListView>
        </DockPanel>
    </Window>
```

WorkerBrowser.xaml

```
    <Window x:Class="CMSGUI.WorkerBrowser"
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            Title="WorkerBrowser" Height="346" Width="992">

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

        <DockPanel LastChildFill="True">
            <UniformGrid Height="30"  DockPanel.Dock="Top" Rows="1">
                <Button Style="{StaticResource BtnStyle}" Content="Show workers"
Name="showPaymentsBtn" Click="showPaymentsBtn_Click"  Margin="10 0" ></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Add New"
Name="addBtn" Click="addBtn_Click"  Margin="10 0" ></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Delete" Margin="10
0" Name="deleteBtn" Click="deleteBtn_Click"></Button>
                <Button Style="{StaticResource BtnStyle}" Content="Print" Margin="10
0" Name="printBtn" IsEnabled="True" Click="showPaymentsBtn_Click"></Button>
            </UniformGrid>
            <ListView Name="businessPersonListView" DockPanel.Dock="Bottom"
                      IsSynchronizedWithCurrentItem="True">
                <ListView.View>
                    <GridView>
                        <!-- GridViewColumn Width="50" Header="Sl No"
DisplayMemberBinding="{Binding serialNo}"  /-->
                        <GridViewColumn Width="150" Header="Id"
DisplayMemberBinding="{Binding workerId}"  />
                        <GridViewColumn Width="260" Header="Name"
DisplayMemberBinding="{Binding workerName}"/>
                        <GridViewColumn Width="160" Header="Address"
DisplayMemberBinding="{Binding workerAdress}"  />
                        <GridViewColumn Width="100" Header="Ph no."
DisplayMemberBinding="{Binding phoneNumber}"  />
                        <GridViewColumn Width="100" Header="Manager"
DisplayMemberBinding="{Binding workerVatNo}"  />
                        <GridViewColumn Width="100" Header="Expertise"
DisplayMemberBinding="{Binding workerTurnOver}"  />
                        <GridViewColumn Width="100" Header="Duty Cycle"
DisplayMemberBinding="{Binding workerDue}"  />
                    </GridView>
                </ListView.View>
            </ListView>
```

```
        </DockPanel>
    </Window>
```

GUI Style : **CMS**Styles

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>
```

CMS Engine

WorkItemsService.cs

```
#region Using Directives
using System;
using System.ComponentModel;
using System.Collections;
using System.Xml.Serialization;
using System.Data;

using Microsoft.Practices.EnterpriseLibrary.Logging;

#endregion

namespace CMS.App
{

    ///<summary>
    /// An component type implementation of the 'sf_WorkItems' table.
    ///</summary>
    /// <remarks>
    /// All custom implementations should be done here.
    /// </remarks>
    [CLSCompliant(true)]
    public partial class WorkItemsService : WorkItemsServiceBase
    {
         /// <summary>
         /// Used to return the status of the attempt to create a new ....
         /// </summary>
         public enum CreateNewWorkItemStatus {
             Success,  //All others are considered a failure
             InsufficientPermisssions,
```

```
            }

            /// <summary>
            /// Used to return the status of the attempt to update ....
            /// </summary>
            public enum UpdateWorkItemStatus {
                Success,  //All others are considered a failure
                InsufficientPermisssions,
            }

              /// <summary>
              /// Initializes a new instance of the WorkItemsService class.
              /// </summary>
              public WorkItemsService() : base()
              {
              }


            public static void CreateWorkItems(WorkItems workItem, Users creator,
Projects inProject, out CreateNewWorkItemStatus status) {

                //must hold required permissions
                if (!ProjectPermissionsService.UserHoldsPermission(creator, inProject,
ProjectPermissionsColumn.WorkItemCreate)) {
                    status = CreateNewWorkItemStatus.InsufficientPermisssions;
                    return;
                }



                workItem.TotalVotes = 0;
                workItem.LastUpdated = DateTime.Now;
                DataRepository.WorkItemsProvider.Insert(workItem);

                //vote for this work item
                if (creator != null) {
                    //anonymous users can't vote

                    WorkItemVotes newVote = new WorkItemVotes();
                    newVote.WorkItemGroupId = workItem.WorkItemGroupId;
                    newVote.WorkItemId = workItem.WorkItemId;
                    newVote.UserId = creator.UserId;

                    WorkItemManagerService.CreateNewWorkItemVoteStatus voteStatus;
                    WorkItemManagerService.CreateWorkItemVotes(newVote, creator, out
voteStatus);
                }



                status = CreateNewWorkItemStatus.Success;
            }


            public static void UpdateWorkItems(WorkItems workItem, Users user, out
```

```
UpdateWorkItemStatus status) {

                //Check permisison TODO
                workItem.LastUpdated = DateTime.Now;

                DataRepository.WorkItemsProvider.Update(workItem);
                status = UpdateWorkItemStatus.Success;
            }

        }//End Class


    } // end namespace
```

WorkItemManagerService.cs

```
    #region Using Directives
    using System;
    using System.ComponentModel;
    using System.Collections;
    using System.Xml.Serialization;
    using System.Data;

    using Microsoft.Practices.EnterpriseLibrary.Logging;

    #endregion

    namespace CMS.App
    {

        ///<summary>
        /// An component type implementation of the 'sf_WorkItemVotes' table.
        ///</summary>
        /// <remarks>
        /// All custom implementations should be done here.
        /// </remarks>
        [CLSCompliant(true)]
        public partial class WorkItemManagerService :
SharpForge.App.WorkItemManagerServiceBase
        {
             /// <summary>
             /// Used to return the status of the attempt to create a new ....
             /// </summary>
             public enum CreateNewWorkItemVoteStatus {
                 Success,  //All others are considered a failure
                 UserAlreadyVoted,
                 InsufficientPermisssions,
             }


               /// <summary>
               /// Initializes a new instance of the WorkItemManagerService class.
               /// </summary>
```

```
            public WorkItemManagerService() : base()
            {
            }


        /// <summary>
        ///  PRE: If the user has already voted don't add another vote TODO
        ///  PRE: Check permissions
        ///
        /// Returns the workitem after its been updated with the new total
        /// </summary>
        /// <param name="workItemVote"></param>
        /// <param name="creator"></param>
        /// <param name="status"></param>
        public static WorkItems CreateWorkItemVotes(WorkItemVotes workItemVote,
Users creator, out CreateNewWorkItemVoteStatus status) {



            if
(!WorkItemManagerService.HasVoted(workItemVote.WorkItemId.ToString(),
creator.UserId.ToString())) {
                //Check permisison to create TODO
                DataRepository.WorkItemVotesProvider.Insert(workItemVote);

                //update the number of votes for this work item
                WorkItems workItem =
DataRepository.WorkItemsProvider.GetByWorkItemId(workItemVote.WorkItemId);
                workItem.TotalVotes = workItem.TotalVotes + 1;
                DataRepository.WorkItemsProvider.Update(workItem);



                status = CreateNewWorkItemVoteStatus.Success;
                return workItem;
            }
            else
                status = CreateNewWorkItemVoteStatus.UserAlreadyVoted;

            return null;
        }


        public static bool HasVoted(WorkItems workItem, Users user) {
            return HasVoted(workItem.WorkItemId.ToString(),
user.UserId.ToString());
        }

        /// <summary>
        /// Has the user voted for this work item.
        /// </summary>
        /// <param name="workItem"></param>
        /// <param name="user"></param>
        /// <returns></returns>
        public static bool HasVoted(string workItemId, string userId) {
```

```
                int count = -1;
                string sql = "WorkItemId = " + workItemId + " and UserId = '" + userId
+ "'";
                TList<WorkItemVotes> votes =
DataRepository.WorkItemVotesProvider.GetPaged(sql, null, 0, 50, out count);

                if (count == 1)
                    return true;
                else
                    return false;
            }

        }//End Class


    } // end namespace
```

Database connector: **CMS**Db

DbInteraction.cs

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

    namespace CMSDb
    {
      public class DbInteraction
      {
        static string passwordCurrent = "technicise";
        static string dbmsCurrent = "CMSdb";

        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
```

Datbase Classes : **CMS**Data

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;

    }
```

Comments and Description of Coding segments

Code Commenting

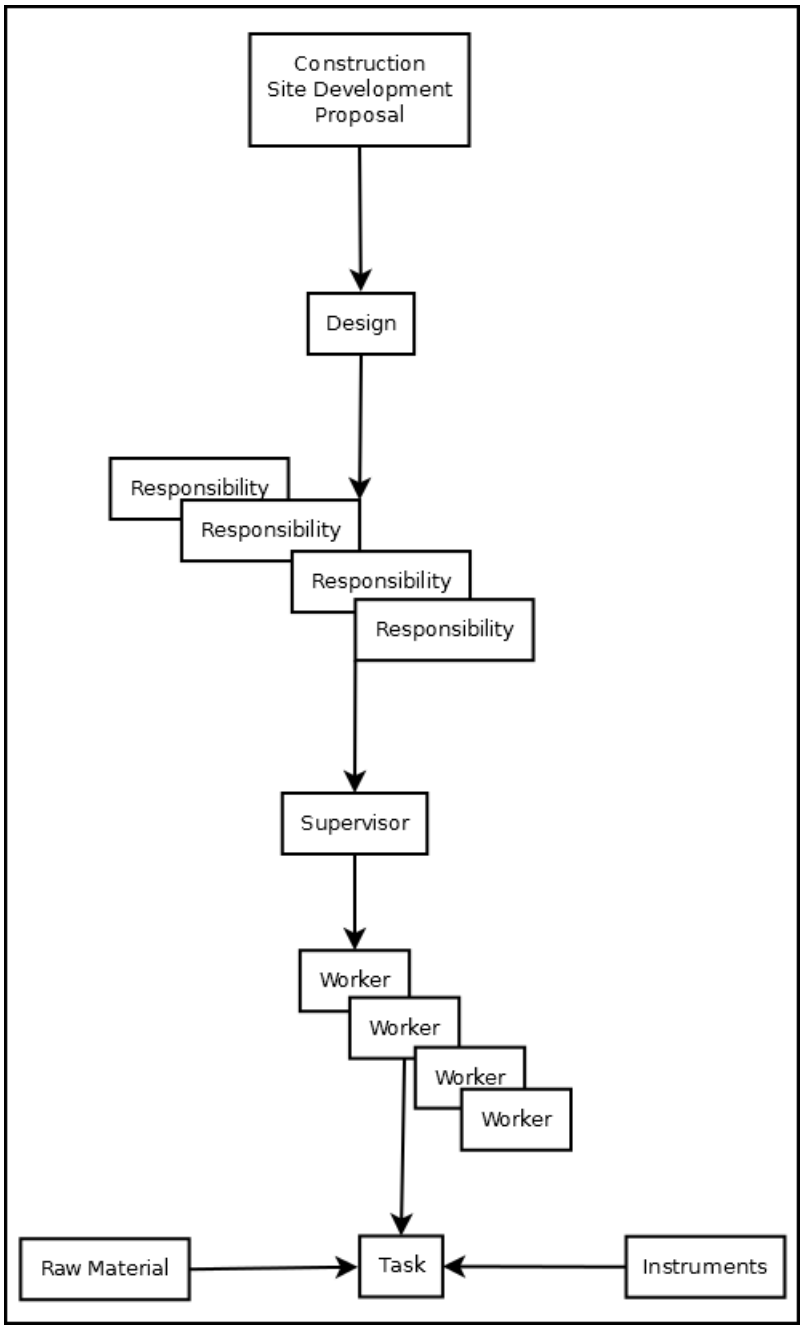
- All comments have been written in the same language, be grammatically correct, and contain appropriate punctuation.
- Used // or /// but never /* … */
- Did not “flowerbox” comment blocks.
- Example:
- // ***********************************
- // Comment block
- // ***********************************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

Coding of this project is done using object oriented methodology… I followed MVC pattern. Here is the overall process logic followed in coding.



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.

Standardization of the coding

Coding style causes the most inconsistency and controversy between developers. Each developer has a preference, and rarely are two the same. However, consistent layout, format, and organization are key to creating maintainable code. The following sections describe the preferred way to implement C# source code in order to create readable, clear, and consistent code that is easy to understand and maintain.

Formatting

- Never declared more than 1 namespace per file.
- Avoided putting multiple classes in a single file.
- Always placed curly braces ({ and }) on a new line.
- Always used curly braces ({ and }) in conditional statements.
- Always used a Tab & Indention size of 4.
- Declared each variable independently – not in the same statement.
- Placed namespace “using” statements together at the top of file. Group .NET namespaces above custom namespaces.
- Grouped internal class implementation by type in the following order:

  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

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:

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

Unit testing includes the following:

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

System testing includes the following:

15. Interfacing of run within the system.
16. Compilation and continuity of control totals.

17. Capacity of logical file storage areas and the handling of overflow.
18. Error correction procedures including user involvement.
19. User request for amendments and output.
20. Timing of runs and routines for the data volumes to be actually handled.
21. Output preparation and distribution.
22. Audit requirements.
23. 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:

24. Both the artificial and live data should be representative of reality;
25. Live data can often be checked against the previous system's result;
26. If the previous and new system differ, the two sets of result should be reconciled if at all possible;
27. Logical files are usually needed to fully test the programs and routines;
28. Data generating techniques are useful for simulating large volume of input data file records;
29. 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);
30. 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:

31. 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;

32. 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:

33. Guarantee that all independent paths within a module have been exercised at least one;
34. Exercise all logical decisions on their true and false sides;
35. Execute all loops at there boundaries and within there operational bounds; and
36. 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.

37. Incorrect or missing function.
38. Interface errors
39. Errors in data structures or external data base access
40. Performance errors
41. 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:

1 How is functional validity testing?

2 What classes of input will make good test cases?

3 Is the system particularly sensitive to certain input values?

4 How are the boundaries of a data class isolated?

5 What data rates and data volume can the system tolerate?

6 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

Test reports for System Test Cases

Test Case Id	Comment	Status
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
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

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;
```

4. To initialize variables to contain information about a product, add the following declaration statements to **Main** method:

```
string sProdName = "Widget";
int iUnitQty = 100;
double dUnitCost = 1.03;
```

7. 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 ");
```

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

```
Debug.Indent();
```

9. To display the content of selected variables, use the **WriteLine** method as follows:

```
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());
```

12. We can also use the **WriteLine** method to display the namespace and the class name for an existent object. For example, the following code displays the **System.Xml.XmlDocument** namespace in the Output window:

```
System.Xml.XmlDocument oxml = new System.Xml.XmlDocument();
```

```
Debug.WriteLine(oxml);
```

14. To organize the output, we can include a category as an optional, second input parameter of the **WriteLine** method. If we specify a category, the format of the Output window message is "category: message." For example, the first line of the following code displays "Field: The product name is Widget" in the Output window:

```
15.     Debug.WriteLine("The product name is " + sProdName,"Field");
16.     Debug.WriteLine("The units on hand are" + iUnitQty,"Field");
17.     Debug.WriteLine("The per unit cost is" + dUnitCost.ToString(),"Field");
Debug.WriteLine("Total Cost is  " + (iUnitQty * dUnitCost),"Calc");
```

18. The Output window can display messages only if a designated condition evaluates to true by using the **WriteLineIf**method of the **Debug** class. The condition to be evaluated is the first input parameter of the **WriteLineIf** method. The second parameter of **WriteLineIf** is the message that appears only if the condition in the first parameter evaluates to true.

```
19.     Debug.WriteLineIf(iUnitQty > 50, "This message WILL appear");
20.     Debug.WriteLineIf(iUnitQty < 50, "This message will NOT appear");
```

21. Use the **Assert** method of the **Debug** class so that the Output window displays the message only if a specified condition evaluates to false:

```
22.     Debug.Assert(dUnitCost > 1, "Message will NOT appear");
23.     Debug.Assert(dUnitCost < 1, "Message will appear since dUnitcost < 1 is false");
```

24. Create the **TextWriterTraceListener** objects for the Console window (tr1) and for a text file named Output.txt (tr2), and then add each object to the **Debug Listeners** collection:

```
25.     TextWriterTraceListener tr1 = new TextWriterTraceListener(System.Console.Out);
26.     Debug.Listeners.Add(tr1);
27.
28.     TextWriterTraceListener tr2 = new
TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));
Debug.Listeners.Add(tr2);
```

29. For readability, use the **Unindent** method to remove the indentation for subsequent messages that the **Debug** class generates. When we use the **Indent** and the **Unindent** methods together, the reader can distinguish the output as group.

```
30.      Debug.Unindent();
```

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

31. To make sure that each **Listener** object receives all its output, call the **Flush** method for the **Debug** class buffers:

```
Debug.Flush();
```

Using the Trace Class

We can also use the **Trace** class to produce messages that monitor the execution of an application. The **Trace** and **Debug**classes share most of the same methods to produce output, including the following:

- **WriteLine**
- **WriteLineIf**
- **Indent**
- **Unindent**
- **Assert**
- **Flush**

We can use the **Trace** and the **Debug** classes separately or together in the same application. In a Debug Solution Configuration project, both **Trace** and **Debug** output are active. The project generates output from both of these classes to all**Listener** objects. However, a Release Solution Configuration project only generates output from a **Trace** class. The Release Solution Configuration project ignores any **Debug** class method invocations.

```
Trace.WriteLine("Trace Information-Product Starting ");

Trace.Indent();


Trace.WriteLine("The product name is "+sProdName);

Trace.WriteLine("The product name is"+sProdName,"Field" );

Trace.WriteLineIf(iUnitQty > 50, "This message WILL appear");

Trace.Assert(dUnitCost > 1, "Message will NOT appear");


Trace.Unindent();

Trace.WriteLine("Trace Information-Product Ending");
```

```
Trace.Flush();


Console.ReadLine();
```

Verify That It Works

1. Make sure that **Debug** is the current solution configuration.
2. If the **Solution Explorer** window is not visible, press the CTRL+ALT+L key combination to display this window.
3. Right-click **conInfo**, and then click **Properties**.
4. In the left pane of the conInfo property page, under the **Configuration** folder, make sure that the arrow points to**Debugging**.

   Note In Visual C# 2005 and in Visual C# 2005 Express Edition, click **Debug** in the **conInfo** page.
5. Above the **Configuration** folder, in the **Configuration** drop-down list box, click **Active (Debug)** or **Debug**, and then click **OK**. In Visual C# 2005 and in Visual C# 2005 Express Edition, click **Active (Debug)** or **Debug** in the **Configuration**drop-down list box in the **Debug** page, and then click **Save** on the **File** menu.
6. Press CTRL+ALT+O to display the Output window.
7. Press the F5 key to run the code. When the **Assertion Failed** dialog box appears, click **Ignore**.
8. In the Console window, press ENTER. The program should finish, and the Output window should display the output that resembles the following

```
9.      Debug Information-Product Starting
10.     The product name is Widget
11.     The available units on hand are100
12.     The per unit cost is 1.03
13.     System.Xml.XmlDocument
14.     Field: The product name is Widget
15.     Field: The units on hand are100
```

```
    Field: The per unit cost is1.03
    Calc: Total Cost is  103
    This message WILL appear
    ---- DEBUG ASSERTION FAILED ----
---- Assert Short Message ----
Message will appear since dUnitcost  < 1 is false
---- Assert Long Message ----


    at Class1.Main(String[] args)  <%Path%>\class1.cs(34)

    The product name is Widget
    The available units on hand are100
    The per unit cost is 1.03
Debug Information-Product Ending
Trace Information-Product Starting
    The product name is Widget
    Field: The product name isWidget
    This message WILL appear
Trace Information-Product Ending

```

The Console window and the Output.txt file should display the following output:

```
The product name is Widget
    The available units on hand are 100
    The per unit cost is 1.03
Debug Information-Product Ending
Trace Information-Product Starting
    The product name is Widget
```

```
44.         Field: The product name is Widget
45.         This message WILL appear
46.      Trace Information-Product Ending
```

Note The Output.txt file is located in the same directory as the conInfo executable (conInfo.exe). Typically, this is the \bin folder where the project source is stored. By default, this is C:\Documents and Settings*User login*\My Documents\Visual Studio Projects\conInfo\bin. In Visual C# 2005 and in Visual C# 2005 Express Edition, the Output.txt file is located in the following folder:

C:\Documents and Settings*User login*\My Documents\Visual Studio 2005\Projects\conInfo\conInfo\bin\Debug

Complete Code Listing

```
using System;
using System.Diagnostics;


class Class1
  {
    [STAThread]
static void Main(string[] args)
    {
string sProdName = "Widget";
int iUnitQty = 100;
double  dUnitCost = 1.03;
Debug.WriteLine("Debug Information-Product Starting ");
Debug.Indent();
Debug.WriteLine("The product name is "+sProdName);
Debug.WriteLine("The available units on hand are"+iUnitQty.ToString());
Debug.WriteLine("The per unit cost is "+ dUnitCost.ToString());


      System.Xml.XmlDocument oxml = new System.Xml.XmlDocument();
```

```
Debug.WriteLine(oxml);


Debug.WriteLine("The product name is "+sProdName,"Field");
Debug.WriteLine("The units on hand are"+iUnitQty,"Field");
Debug.WriteLine("The per unit cost is"+dUnitCost.ToString(),"Field");
Debug.WriteLine("Total Cost is  "+(iUnitQty * dUnitCost),"Calc");

Debug.WriteLineIf(iUnitQty > 50, "This message WILL appear");
Debug.WriteLineIf(iUnitQty < 50, "This message will NOT appear");

Debug.Assert(dUnitCost > 1, "Message will NOT appear");
Debug.Assert(dUnitCost < 1, "Message will appear since dUnitcost  < 1 is false");

      TextWriterTraceListener tr1 = new TextWriterTraceListener(System.Console.Out);
Debug.Listeners.Add(tr1);

      TextWriterTraceListener tr2 = new
TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));
Debug.Listeners.Add(tr2);



Debug.WriteLine("The product name is "+sProdName);
Debug.WriteLine("The available units on hand are"+iUnitQty);
Debug.WriteLine("The per unit cost is "+dUnitCost);
Debug.Unindent();
Debug.WriteLine("Debug Information-Product Ending");
Debug.Flush();
```

```
Trace.WriteLine("Trace Information-Product Starting ");
Trace.Indent();

Trace.WriteLine("The product name is "+sProdName);
Trace.WriteLine("The product name is"+sProdName,"Field" );
Trace.WriteLineIf(iUnitQty > 50, "This message WILL appear");
Trace.Assert(dUnitCost > 1, "Message will NOT appear");



Trace.Unindent();
Trace.WriteLine("Trace Information-Product Ending");

Trace.Flush();

Console.ReadLine();
    }
  }
```

Troubleshoot

- If the solution configuration type is **Release**, the **Debug** class output is ignored.
- After we create a **TextWriterTraceListener** class for a particular target, **TextWriterTraceListener** receives output from the **Trace** and the **Debug** classes. This occurs regardless of whether we use the **Add** method of the **Trace** or the**Debug** class to add **TextWriterTraceListener** to the **Listeners** class.
- If we add a **Listeners** object for the same target in the **Trace** and the **Debug** classes, each line of output is duplicated, regardless of whether **Debug** or **Trace** generates the output.

```
      TextWriterTraceListener myWriter = new
TextWriterTraceListener(System.Console.Out);
Debug.Listeners.Add(myWriter);
```

```
        TextWriterTraceListener myCreator = new
TextWriterTraceListener(System.Console.Out);
Trace.Listeners.Add(myCreator);
```

System Security measures:

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 reports that are likely to be generated in this software are given below:

- List of Raw materials used/purchased can be generated
- List of Labors/workers can be generated
- List of buyers can be generated
- Work progress details can be generated
- Fund details can 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://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

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

Microsoft Visual Studio is a powerful IDE that ensures quality code throughout the entire application lifecycle, from design to deployment. Whether we are developing applications for SharePoint, the web, Windows, Windows Phone, and beyond, Visual Studio is the ultimate all-in-one solution. Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

Standout Features

- User interface built on Windows Presentation Foundation (WPF)

- Improved Start page
- Improved code editor
- Improved IntelliSense
- Call Hierarchy Viewer

What problems does it solve?

The newly designed user experience is refreshing for an application showing its age. The user interface is built on WPF and no longer relies on the limited MDI interface in previous versions; this allows for better multi-monitor support with fly-out windows. The first thing you might notice when opening Visual Studio 2010 is the new Start page. As an xaml file, this page is completely customizable and includes the ability to remove and pin project files in the Recent Projects section.

The code editor has a number of enhancements. You can scale the font by holding down [Ctrl] while scrolling the mouse wheel. In previous versions of Visual Studio, users had to set the font size through a dialog and exit to see if the changes were correct.

In Visual Studio 2010, Box Selection is enhanced to allow for zero-length boxes and improved pasting.

The feature that will see the most use (by accident if not design) is Highlight References. By selecting any symbol, such as a variable or a property, all references to the symbol are highlighted. The symbols can then be navigated by holding down [Ctrl][Shift] and pressing the up/down keys.

IntelliSense has been improved to allow for acronyms based on Pascal casing. For example, typing String.INOE and then a non-alphanumeric character will convert the call

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 namespace. If you have previously built applications with .NET Framework using managed technologies like ASP.NET and Windows Forms, the fundamental WPF programming experience should be familiar; you instantiate classes, set properties, call methods, and handle events, all using your favorite .NET Framework programming language, such as C# or Visual Basic.

Markup & code-behind

WPF offers additional programming enhancements for Windows client application development. One obvious enhancement is the ability to develop an application using both markup and code-behind, an experience that ASP.NET developers should be familiar with. You generally use Extensible Application Markup Language (XAML) markup to implement the appearance of an application while using managed programming languages (code-behind) to implement its behavior.

security

Because XBAPs are hosted in a browser, security is important. In particular, a partial-trust security sandbox is used by XBAPs to enforce restrictions that are less than or equal to the restrictions imposed on HTML-based applications. Furthermore, each HTML feature that is safe to run from XBAPs in partial trust has been tested using a comprehensive security process.

controls

The user experiences that are delivered by the application model are constructed controls. In WPF, "control" is an umbrella term that applies to a category of WPF classes that are hosted in either a window or a page, have a user interface (UI), and implement some behavior.

Wpf controls by function

The built-in WPF controls are listed here.

- Buttons: Button and RepeatButton.
- Data Display: DataGrid, ListView,and TreeView.
- Date Display and Selection: Calendar and DatePicker.
- Dialog Boxes: OpenFileDialog, PrintDialog, and SaveFileDialog.
- Digital Ink: InkCanvas and InkPresenter.
- Documents: DocumentViewer, FlowDocumentPageViewer, FlowDocumentReader, Flow DocumentScrollViewer, andStickyNoteControl.
- Input: TextBox, RichTextBox, and PasswordBox.
- Layout: Border, BulletDecorator, Canvas, DockPanel, Expander, Grid, GridView, GridSplit ter, GroupBox, Panel,ResizeGrip, Separator, ScrollBar, ScrollViewer, StackPanel, Thumb, Viewbox, VirtualizingStackPanel, Window, andWrapPanel.
- Media: Image, MediaElement, and SoundPlayerAction.
- Menus: ContextMenu, Menu, and ToolBar.
- Navigation: Frame, Hyperlink, Page, NavigationWindow, and TabControl.
- Selection: CheckBox, ComboBox, ListBox, RadioButton, and Slider.
- User Information: AccessText, Label, Popup, ProgressBar, StatusBar, TextBlock, and ToolTip.

layout

When you create a UI, you arrange your controls by location and size to form a layout. A key requirement of any layout is to adapt to changes in window size and display settings. Rather than forcing you to write the code to adapt a layout in these circumstances, WPF provides a first-class, extensible layout system for you.
The cornerstone of the layout system is relative positioning, which increases the ability to adapt to changing window and display conditions. In addition, the layout system manages the negotiation between controls to determine the layout. The negotiation is a two-step process:

first, a control tells its parent what location and size it requires; second, the parent tells the control what space it can have.
The layout system is exposed to child controls through base WPF classes. For common layouts such as grids, stacking, and docking, WPF includes several layout controls:

- Canvas : Child controls provide their own layout.
- DockPanel : Child controls are aligned to the edges of the panel.
- Grid : Child controls are positioned by rows and columns.
- StackPanel : Child controls are stacked either vertically or horizontally.
- VirtualizingStackPanel : Child controls are virtualized and arranged on a single line that is either horizontally or vertically oriented.
- WrapPanel : Child controls are positioned in left-to-right order and wrapped to the next line when there are more controls on the current line than space allows.

Graphics

WPF introduces an extensive, scalable, and flexible set of graphics features that have the following benefits:

- **Resolution-independent and device-independent graphics**. The basic unit of measurement in the WPF graphics system is the device independent pixel, which is 1/96th of an inch, regardless of actual screen resolution, and provides the foundation for resolution-independent and device-independent rendering. Each device-independent pixel automatically scales to match the dots-per-inch (dpi) setting of the system it renders on.
- **Improved precision**. The WPF coordinate system is measured with double-precision floating-point numbers rather than single-precision. Transformations and opacity values are also expressed as double-precision. WPF also supports a wide color gamut (scRGB) and provides integrated support for managing inputs from different color spaces.
- **Advanced graphics and animation support**. WPF simplifies graphics programming by managing animation scenes for you; there is no need to worry about scene processing, rendering loops, and bilinear interpolation. Additionally, WPF provides hit-testing support and full alpha-compositing support.
- **Hardware acceleration**. The WPF graphics system takes advantage of graphics hardware to minimize CPU usage.

Extensible application Markup Language (XAML)

XAML stands for Extensible Application Markup Language. 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	The System namespace contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.
System.Activities	The System.Activities namespaces contain all the classes necessary to create and work with activities in Window Workflow Foundation.
System.AddIn	The System.AddIn namespaces contain types used to identify, register, activate, and control add-ins, and to allow add-ins to communicate with a host application.
System.CodeDom	The System.CodeDom namespaces contain classes that represent the elements of a source code document and that support the generation and compilation of source code in supported programming languages.
System.Collections	The System.Collections namespaces contain types that define various standard, specialized, and generic collection objects.
System.ComponentModel	The System.ComponentModel namespaces contain types that implement the run-time and design-time behavior of components and controls. Child namespaces support the Managed Extensibility Framework (MEF), provide attribute classes that define metadata for ASP.NET Dynamic Data controls, and contain types that let you define the design-time behavior of components

	and their user interfaces.
System.Configuration	The System.Configuration namespaces contain types for handling configuration data, such as data in machine or application configuration files. Child namespaces contain types that are used to configure an assembly, to write custom installers for components, and to support a pluggable model for adding functionality to, or removing functionality from, both client and server applications.
System.Data	The System.Data namespaces contain classes for accessing and managing data from diverse sources. The top-level namespace and a number of the child namespaces together form the ADO.NET architecture and ADO.NET data providers. For example, providers are available for SQL Server, Oracle, ODBC, and OleDB. Other child namespaces contain classes used by the ADO.NET Entity Data Model (EDM) and by WCF Data Services.
System.Deployment	The System.Deployment namespaces contain types that support deployment of ClickOnce applications.
System.Device.Location	The System.Device.Location namespace allows application developers to easily access the computer's location by using a single API. Location information may come from multiple providers, such as GPS, Wi-Fi triangulation, and cell phone tower triangulation. The System.Device.Location classes provide a single API to encapsulate the multiple location providers on a computer and support seamless prioritization and transitioning between them. As a result, application developers who use this API do not need to tailor applications to specific hardware configurations.
System.Diagnostics	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	The System.DirectoryServices namespaces contain types that provide access to Active Directory from managed code.
System.Drawing	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	The System.Dynamic namespace provides classes and interfaces that support Dynamic Language Runtime.
System.EnterpriseServices	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	The System.Globalization namespace contains classes that define culture-related information, including language, country/region, calendars in use, format patterns for dates, currency, and numbers, and sort order for strings. These classes are useful for writing globalized (internationalized) applications. Classes such as StringInfo andTextInfo provide advanced globalization functionalities, including surrogate support and text element processing.
System.IdentityModel	The System.IdentityModel namespaces contain types that are used to provide authentication and authorization for .NET applications.
System.IO	The System.IO namespaces contain types that support input and output, including the ability to read and write data to streams either synchronously or asynchronously, to compress data in streams, to create and use isolated stores, to map files to an application's logical address space, to store multiple data objects in a single container, to communicate using anonymous or named pipes, to implement custom logging, and to handle the flow of data to and from serial ports.
System.Linq	The System.Linq namespaces contain types that support queries that use Language-Integrated Query (LINQ). This includes types that represent queries as objects in expression trees.
System.Management	The System.Management namespaces contain types that provide access to management information and management events about the system, devices, and applications instrumented to the Windows Management Instrumentation (WMI) infrastructure. These namespaces also contain types necessary for instrumenting applications so that they expose their management

	information and events through WMI to potential customers.
System.Media	The System.Media namespace contains classes for playing sound files and accessing sounds provided by the system.
System.Messaging	The System.Messaging namespaces contain types that enable you to connect to, monitor, and administer message queues on the network and to send, receive, or peek messages. A child namespace contains classes that can be used to extend design-time support for messaging classes.
System.Net	The System.Net namespaces contain classes that provide a simple programming interface for a number of network protocols, programmatically access and update configuration settings for the System.Net namespaces, define cache policies for web resources, compose and send e-mail, represent Multipurpose Internet Mail Exchange (MIME) headers, access network traffic data and network address information, and access peer-to-peer networking functionality. Additional child namespaces provide a managed implementation of the Windows Sockets (Winsock) interface and provide access to network streams for secure communications between hosts.
System.Numerics	The System.Numerics namespace contains numeric types that complement the numeric primitives, such as Byte, Double, and Int32, that are defined by the .NET Framework.
System.Printing	The System.Printing namespaces contain types that support printing, that provide access to the properties of

	print system objects and enable rapid copying of their property settings to another object of the same type, and that support the interconversion of managed System.PrintTicket objects and unmanaged GDI DEVMODE structures.
System.Reflection	The System.Reflection namespaces contain types that provide a managed view of loaded types, methods, and fields, and that can dynamically create and invoke types. A child namespace contains types that enable a compiler or other tool to emit metadata and Microsoft intermediate language (MSIL).
System.Resources	The System.Resources namespaces contain types that enable developers to create, store, and manage an application's culture-specific resources.
System.Runtime	The System.Runtime namespaces contain types that support an application's interaction with the common language runtime, and types that enable features such as application data caching, advanced exception handling, application activation within application domains, COM interop, distributed applications, serialization and deserialization, and versioning. Additional namespaces enable compiler writers to specify attributes that affect the run-time behavior of the common language runtime, define a contract for reliability between a set of code and other code that takes a dependency on it, and implement a persistence provider for Windows Communication Foundation (WCF).
System.Security	The System.Security namespaces contain classes that represent the .NET Framework security system and permissions. Child namespaces provide types that control access to and audit securable objects, allow authentication, provide crytographic services, control

	access to operations and resources based on policy, and support rights management of application-created content.
System.ServiceModel	The System.ServiceModel namespaces contain the types necessary to build Windows Communication Foundation (WCF) service and client applications.
System.ServiceProcess	The System.ServiceProcess namespaces contain types that enable you to implement, install, and control Windows service applications and extend design-time support for Windows service applications.
System.Speech	The System.Speech namespaces contain types that support speech recognition.
System.Text	The System.Text namespaces contain types for character encoding and string manipulation. A child namespace enables you to process text using regular expressions.
System.Threading	The System.Threading namespaces contain types that enable multithreaded programming. A child namespace provides types that simplify the work of writing concurrent and asynchronous code.
System.Timers	The System.Timers namespace provides the Timer component, which allows you to raise an event on a specified interval.
System.Transactions	The System.Transactions namespaces contain types that support transactions with multiple, distributed participants, multiple phase notifications, and durable

	enlistments. A child namespace contains types that describe the configuration options used by the System.Transactions types.
System.Web	The System.Web namespaces contain types that enable browser/server communication. Child namespaces include types that support ASP.NET forms authentication, application services, data caching on the server, ASP.NET application configuration, dynamic data, HTTP handlers, JSON serialization, incorporating AJAX functionality into ASP.NET, ASP.NET security, and web services.
System.Windows	The System.Windows namespaces contain types used in Windows Presentation Foundation (WPF) applications, including animation clients, user interface controls, data binding, and type conversion. System.Windows.Forms and its child namespaces are used for developing Windows Forms applications.
System.Workflow	The System.Workflow namespaces contain types used to develop applications that use Windows Workflow Foundation. These types provide design time and run-time support for rules and activities, to configure, control, host, and debug the workflow runtime engine.
System.Xaml	The System.Xaml namespaces contain types that support parsing and processing the Extensible Application Markup Language (XAML).
System.Xml	The System.Xml namespaces contain types for processing XML. Child namespaces support serialization of XML documents or streams, XSD schemas, XQuery 1.0 and XPath 2.0, and LINQ to XML, which is an in-memory XML programming interface that enables easy

	modification of XML documents.
Accessibility	The Accessibility and all of its exposed members are part of a managed wrapper for the Component Object Model (COM) accessibility interface.
Microsoft.Activities	The Microsoft.Activities namespaces contain types that support MSBuild and debugger extensions for Windows Workflow Foundation applications.
Microsoft.Aspnet.Snapin	The Microsoft.Aspnet.Snapin namespace defines the types necessary for the ASP.NET management console application to interact with Microsoft Management Console (MMC). For more information, see "MMC Programmer's Guide" in the MSDN Library.
Microsoft.Build	The Microsoft.Build namespaces contain types that provide programmatic access to, and control of, the MSBuild engine.
Microsoft.CSharp	The Microsoft.CSharp namespaces contain types that support compilation and code generation of source code written in the C# language, and types that support interoperation betwen the dynamic language runtime (DLR) and C#.
Microsoft.Data.Entity.Build.Tasks	The Microsoft.Data.Entity.Build.Tasks namespace contains two MSBuild tasks that are used by the ADO.NET Entity Data Model Designer (Entity Designer).
Microsoft.JScript	The Microsoft.JScript namespaces contain classes that support compilation and code generation using the JScript language.

Microsoft.SqlServer.Server	The Microsoft.SqlServer.Server namespace contains classes, interfaces, and enumerations that are specific to the integration of the Microsoft .NET Framework common language runtime (CLR) into Microsoft SQL Server, and the SQL Server database engine process execution environment.
Microsoft.VisualBasic	The Microsoft.VisualBasic namespaces contain classes that support compilation and code generation using the Visual Basic language. Child namespaces contain types that provide services to the Visual Basic compiler and types that include support for the Visual Basic application model, the My namespace, lambda expressions, and code conversion.
Microsoft.VisualC	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	The Microsoft.Win32 namespaces provide types that handle events raised by the operating system, that manipulate the system registry, and that represent file and operating system handles.
Microsoft.Windows	The Microsoft.Windows namespaces contain types that support themes and preview in Windows Presentation Framework (WPF) applications.
UIAutomationClientsideProviders	Contains a single type that maps client automation providers.
XamlGeneratedNamespace	Contains compiler-generated types that are not

	intended to be used directly from your code.

Database/backend:

MySQL

MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout its history.

The MySQL Community Edition includes:

- Pluggable Storage Engine Architecture
- Multiple Storage Engines: InnoDB , MyISAM, NDB (MySQL Cluster),Memory ,Merge , Archive, CSV
- MySQL Replication to improve application performance and scalability
- MySQL Partitioning to improve performance and management of large database applications
- Stored Procedures to improve developer productivity

Detailed features of mysql

The following list shows the most important properties of MySQL. This section is directed to the reader who already has some knowledge of relational databases. We will use some terminology from the relational database world without defining our terms exactly. On the other hand, the explanations should make it possible for database novices to understand to some extent what we are talking about.

Relational Database System: Like almost all other database systems on the market, MySQL is a relational database system.

Client/Server Architecture: MySQL is a client/server system. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc. The clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

Almost all of the familiar large database systems (Oracle, Microsoft SQL Server, etc.) are client/server systems. These are in contrast to the file-server systems, which include Microsoft Access, dBase and FoxPro. The decisive drawback to file-server systems is that when run over a network, they become extremely inefficient as the number of users grows.

SQL compatibility: MySQL supports as its database language -- as its name suggests – SQL (Structured Query Language). SQL is a standardized language for querying and updating data and for the administration of a database. There are several SQL dialects (about as many as there are database systems). MySQL adheres to the current SQL standard (at the moment SQL:2003), although with significant restrictions and a large number of extensions.

Through the configuration setting sql-mode you can make the MySQL server behave for the most part compatibly with various database systems. Among these are IBM DB/2 and Oracle. (The setting sql-mode changes some of the syntax conventions, and performs no miracles.

SubSELECTs: Since version 4.1, MySQL is capable of processing a query in the form SELECT * FROM table1 WHERE x IN (SELECT y FROM table2) (There are also numerous syntax variants for subSELECTs.)

Views: Put simply, views relate to an SQL query that is viewed as a distinct database object and makes possible a particular view of the database. MySQL has supported views since version 5.0.

Stored procedures: Here we are dealing with SQL code that is stored in the database system.

Stored procedures (SPs for short) are generally used to simplify certain steps, such as inserting or deleting a data record. For client programmers this has the advantage that they do not have to process the tables directly, but can rely on SPs. Like views, SPs help in the administration of large database projects. SPs can also increase efficiency. MySQL has supported SPs since version 5.0.

Triggers: Triggers are SQL commands that are automatically executed by the server in certain database operations (INSERT, UPDATE, and DELETE). MySQL has supported triggers in a limited form from version 5.0, and additional functionality is promised for version 5.1.

Unicode: MySQL has supported all conceivable character sets since version 4.1, including Latin-1, Latin-2, and Unicode (either in the variant UTF8 or UCS2).

User interface: There are a number of convenient user interfaces for administering a MySQL server.

Full-text search: Full-text search simplifies and accelerates the search for words that are located within a text field. If you employ MySQL for storing text (such as in an Internet discussion group), you can use full-text search to implement simply an efficient search function.

Replication: Replication allows the contents of a database to be copied (replicated) onto a number of computers. In practice, this is done for two reasons: to increase protection against system failure (so that if one computer goes down, another can be put into service) and to improve the speed of database queries.

Transactions: In the context of a database system, a transaction means the execution of several database operations as a block. The database system ensures that either all of the operations are correctly executed or none of them. This holds even if in the middle of a transaction there is a power failure, the computer crashes, or some other disaster occurs. Thus, for example, it cannot occur that a sum of money is withdrawn from account A but fails to be deposited in account B due to some type of system error.

Transactions also give programmers the possibility of interrupting a series of already executed commands (a sort of revocation). In many situations this leads to a considerable simplification of the programming process. In spite of popular opinion, MySQL has supported transactions for a long time. One should note here that MySQL can store tables in a variety of formats. The default table format is called MyISAM, and this format does not support transactions. But there are a number of additional formats that do support transactions. The most popular of these is InnoDB, which will be described extensively in this book.

Foreign key constraints: These are rules that ensure that there are no cross references in linked tables that lead to nowhere. MySQL supports foreign key constraints for InnoDB tables.

GIS functions: Since version 4.1, MySQL has supported the storing and processing of two-dimensional geographical data. Thus MySQL is well suited for GIS (geographic information systems) applications.

Programming languages: There are quite a number of APIs (application programming interfaces) and libraries for the development of MySQL applications. For client programming you can use, among others, the languages C, C++, Java, Perl, PHP, Python, and Tcl.

ODBC: MySQL supports the ODBC interface Connector/ODBC. This allows MySQL to be addressed by all the usual programming languages that run under Microsoft Windows (Delphi, Visual Basic, etc.). The ODBC interface can also be implemented under Unix, though that is seldom necessary.

Windows programmers who have migrated to Microsoft's new .NET platform can, if they wish, use the ODBC provider or the .NET interface Connector/NET.

Platform independence: It is not only client applications that run under a variety of operating systems; MySQL itself (that is, the server) can be executed under a number of operating systems. The most important are Apple Macintosh OS X, Linux, Microsoft Windows, and the countless Unix variants, such as AIX, BSDI, FreeBSD, HP-UX, OpenBSD, Net BSD, SGI Iris, and Sun Solaris.

Speed: MySQL is considered a very fast database program. This speed has been backed up by a large number of benchmark.

ide for Database

MySQL workbench

MySQL Workbench is a visual database design tool that integrates SQL 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 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 encompassing imperative, declarative, functional, generic, object-oriented(class-based), and component-oriented programming disciplines. It was developed by Microsoft within the .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270). C# is one of the programming languages designed for the Common Language Infrastructure.

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

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.

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	Construrtion Management System
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
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----------