Child Education Software

Arindam Ghosh

Index

Contents

[Introduction: 4](#_Toc351941211)

[Objective 6](#_Toc351941212)

[System Analysis 6](#_Toc351941213)

[Identification of Need: 6](#_Toc351941214)

[Preliminary Investigation: 7](#_Toc351941215)

[Feasibility Study: 7](#_Toc351941216)

[Technical Feasibility 7](#_Toc351941217)

[Operational Feasibility 8](#_Toc351941218)

[Economic Feasibility 8](#_Toc351941219)

[Project Planning & Scheduling: 8](#_Toc351941220)

[Gantt chart 8](#_Toc351941221)

[Tracking Gantt 9](#_Toc351941222)

[Pert Chart 9](#_Toc351941223)

[Software requirement specifications (SRS): 10](#_Toc351941224)

[Functional Requirement 10](#_Toc351941225)

[Non-functional Requirements 12](#_Toc351941226)

[Data models 13](#_Toc351941227)

[Context Diagram 13](#_Toc351941228)

[0-Level DFD 13](#_Toc351941229)

[1-Level DFD 14](#_Toc351941230)

[2-Level DFD 16](#_Toc351941231)

[Sequence diagrams 17](#_Toc351941232)

[Entity Relationship Model, 17](#_Toc351941233)

[E-R Diagram 18](#_Toc351941234)

[Class Diagrams 19](#_Toc351941235)

[Activity Diagrams 20](#_Toc351941236)

[User Login 20](#_Toc351941237)

[Lesson Management 21](#_Toc351941238)

[Exercise Management 21](#_Toc351941239)

[View Report 22](#_Toc351941240)

[Sync 23](#_Toc351941241)

[Payment Management 24](#_Toc351941242)

[System Design 25](#_Toc351941243)

[Modularisation details 25](#_Toc351941244)

[Database & Table Details 25](#_Toc351941245)

[Complete Structure 26](#_Toc351941246)

[Module Description 26](#_Toc351941247)

[Child Education Software GUI: 26](#_Toc351941248)

[Child Education Software Engine: 30](#_Toc351941249)

[Tutorial Controller: 30](#_Toc351941250)

[User Interface Design 30](#_Toc351941251)

[Main window 30](#_Toc351941252)

[Open Video 30](#_Toc351941253)

[Play video 31](#_Toc351941254)

[Pause video 31](#_Toc351941255)

[Photo 32](#_Toc351941256)

[Typing 32](#_Toc351941257)

[Brush size 33](#_Toc351941258)

[Select color 33](#_Toc351941259)

[Erase 34](#_Toc351941260)

[Selection 34](#_Toc351941261)

[Test Cases (Unit Test Cases and System Test Cases) 35](#_Toc351941262)

[Coding 35](#_Toc351941263)

[Complete Project Coding 35](#_Toc351941264)

[Comments and Description of Coding segments 35](#_Toc351941265)

[Standardization of the coding 36](#_Toc351941266)

[Code Efficiency 36](#_Toc351941267)

[Error handling 36](#_Toc351941268)

[Parameters calling/passing 36](#_Toc351941269)

[Validation checks 36](#_Toc351941270)

[Testing 36](#_Toc351941271)

[Testing techniques and Testing strategies used 36](#_Toc351941272)

[Testing Plan used 36](#_Toc351941273)

[Test reports for Unit Test Cases and System Test Cases 36](#_Toc351941274)

[Debugging and Code improvement: 36](#_Toc351941275)

[System Security measures: 36](#_Toc351941276)

[Database/data security: 36](#_Toc351941277)

[Creation of User profiles and access rights 37](#_Toc351941278)

[Cost Estimation of the Project along with Cost Estimation Model 37](#_Toc351941279)

[Estimation of development effort 37](#_Toc351941280)

[Estimation of development time 37](#_Toc351941281)

[Reports 38](#_Toc351941282)

[Future scope and further enhancement of the Project 38](#_Toc351941283)

[Bibliography 39](#_Toc351941284)

[Website 39](#_Toc351941285)

[Books 40](#_Toc351941286)

[Appendices 40](#_Toc351941287)

[Mobile App Development 40](#_Toc351941288)

[Nokia SDK 2.0 for Java — for Series 40 apps 40](#_Toc351941289)

[Nokia Web - Tools 41](#_Toc351941290)

[Cacoo:: online drawing tool 45](#_Toc351941291)

[Creating Diagrams 45](#_Toc351941292)

[Collaboration 46](#_Toc351941293)

[Sharing Diagrams 46](#_Toc351941294)

[Managing Diagrams 46](#_Toc351941295)

[Languages and Time Zones 46](#_Toc351941296)

[Security 46](#_Toc351941297)

[API 46](#_Toc351941298)

[GitHub 47](#_Toc351941299)

[Description 47](#_Toc351941300)

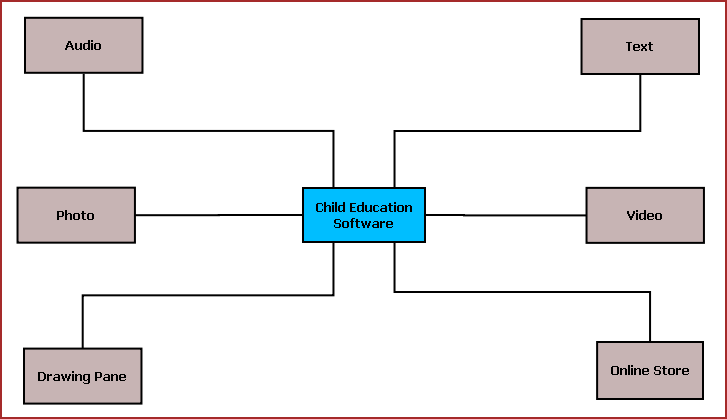
[Limitations and constraints 47](#_Toc351941301)

[Glossary. 47](#_Toc351941302)

# Introduction:

Child Education Software will enable a new aspect of teaching children with the help of modern technology. It will incorporate audio visual teaching method with user friendly interface. Almost every family is having a computer now. We are used to computerized system in every walk of our life. So kids also will be attracted towards computer. We can use computer as a mean of teaching new things. Because of rich multi-media content kids will be happy to use computer for learning new topics.

In Child Education Software users can use the existing tutorial as well as they can download new tutorial from online store. It will be an eco-friendly option. Children wastes paper a lot and they get bored as the colour of the fades gradually. The child education software will make learning more efficient, attractive and tech-savvy for the children.



**Figure 1:** Overview of Child Education Software

The main features of this software are listed below:

* + Textual display of teaching material.
  + Video display for video tutorial.
  + Sound system to play audio clips.
  + Drawing Pane or Handwriting practice area.
  + Photo viewer.
  + Easy navigation.
  + Interactive UI.
  + Printing of selected content.
  + Online store for buying new tutorials and upgrades.

## Objective

The objectives of the projects are given below:

* To provide a new kind of medium for educating children
* To cop up with technological revolutions happening around us and to involve the kids with new technology
* To make education process more interesting to the kids
* To utilize our electronic gadgets to do more meaningful work like teaching children
* To save paper waste and to take more green initiatives
* By accomplishing this project I could learn new technologies like .NET, C#, XML and I am able to be involved with the complete software development lifecycle.

# System Analysis

## Identification of Need:

Now we are used to computerized system in every walk of our life and kids also will be attracted towards computer. We can use computer as a mean of teaching new things. Because of rich multi-media content kids will be happy to use computer for learning new topics. Since we are also dedicated to the education we are always searching for ways to create educational material which starts from "simple" alphabets to complex grammar exercises and study material for other subjects such as science, mathematics, history etc. Our primary focus is on schoolchildren aged 3 to 18, and the specialized user interface needs of young users. However, we will also have programs to aid teachers in planning lessons, and others that are of interest to university students and anyone else with a desire to learn.

CES will allow participants to have lessons from offline resources as well as online study materials. They can also do a hands-on with the exercises and evaluate their progress. Additional study material and tool can be availed by purchasing modules from various online resources.

## Preliminary Investigation:

From everywhere I tried to spoke with the parents and guardian about this modern system of education. All they agree with may think about this application. I tried to collect opinion from software experts and took their opinion to develop the child education system.

## Feasibility Study:

Feasibility study is an essential requirement of any proposed system. It proposes one or more conceptual solutions to the problem set for the project. It is the review of the findings analyzed so far. Before proceeding directly into the design phase I had to check whether this project is worth doing. For this I carried out feasibility study of the proposed system in 3 different ways

* To check whether the project is technically feasible that is whether I have the necessary skills and know-how to complete the project.
* To check whether the proposed system is easy to use and that it satisfies the user objectives (operational feasibility) and can be fitted into current system operation.
* To determine whether the project’s goal can be achieved within the resource limits allocated that is to find out whether the project is economically feasible.

### Technical Feasibility

It has been already mentioned that ‘**Child Education System’** is purely a desktop based project with some mobile application and related web-based project. A lots of such types of projects have already been made or been running to make every day. So it is not even technically impossible rather difficult to build such a software. The technical software knowledge that is required for desktop is C#, Mysql, WPF for mobile apps is nokia web tool and some anyone who has a basic knowledge of computer software and has a good sense of design can almost be an expert in handling these software. As the project will proceed one can easily make himself informative about ASP to do the project. So not very much technical capability is required by the project. We see that the project is technically feasible and worth to do because of the reasons specified above.

### Operational Feasibility

To know whether the proposed system is acceptable to the end user, they were subjected to a mini prototype. The users were asked to respond as to how they feel about the system. To a large extent the users were satisfied with the prototype, which I have shown to them, minor modifications were also done to closely match with the users’ requirement.

To implement the user requirements in a software system is the true goal of a systems analyst. I rigorously followed the requirements of the user, what they want from the system and how it should help them. Prototyping results show that the proposed system is acceptable to the user.

### Economic Feasibility

If it is seen from real life view then MY PROJECT IS ECONOMICALLY FEASIBLE as this sort of software is going to get the market. As these types of engineering colleges emerge they all will try to communicate with at most people they can. When this communication is required portal will be the most wanted communication medium that they rely for. So this type of web-based project will always have a commercial aspect. So I am not wasting my time in a non-economic feasible project.

Technology is growing up day by day. The parents can easily afford to buy various software for their children’s education. If we keep the cost in comfortable label with the basic feature, which help to maintain the child education system, we can sell the application. And according to their need we can modify the applications and customize it for their help.

## Project Planning & Scheduling:

### Gantt chart



### Tracking Gantt



### 

### Pert Chart



## Software requirement specifications (SRS):

### Functional Requirement

#### Audio visual display of study materials:

**Introduction**:

User should be able to watch animated video lessons as well as audio lessons.

**Input**:

Click on show lesion button.

**Processing**:

System will find and fetch a predefined lesion and display that.

**Output**:

Kids can watch them and learn new things.

#### Exercise for students:

**Introduction**:

There should be relevant and user friendly exercise topics.

**Input**:

User will click on start practice button.

**Processing**:

System will open relevant practice set for kids from database.

**Output**:

Kids can interact with the graphical practice sets and solve them.

#### Drawing pane for practice

**Introduction**:

There should be a drawing pane for kids with an attractive GUI.

**Input**:

User will click on start drawing button and kids will start interacting with the GUI.

**Processing**:

Drawn shapes will be saved temporarily in memory and could be permanently kept in database.

**Output**:

Stored drawings will be shown to the user whenever is asked.

#### Download new lesions:

**Introduction**:

New lesions could be downloaded from the web if user wants.

**Input**:

User clicks on download new lesions button.

**Processing**:

System searches for relevant data in the previously stored online database and copies them in the local system.

**Output**:

System shows the newly downloaded lesions to the user.

#### Learn about various topics like nature, economy etc.

**Introduction**:

There should be various GK tutorials with relevant images and examples.

**Input**:

User clicks on the learn button.

**Processing**:

Various options provided to the user and as per his choice a new image and description gets opened.

**Output**:

Students get to see topics with images and examples that help them remember the topics better.

#### Online payment

**Introduction**:

For purchasing lesions, user should be able to pay online using debit card, internet banking, credit card etc.

**Input**:

User opts for online payment option in the application GUI.

**Processing**:

After getting required details, system takes user to his banker’s website and deducts the required amount. Generates a online bill as well.

**Output**:

User gets the digital bill’s copy in his email inbox and gets the new lesion in his system as well.

#### Easily sharable downloaded lesions:

**Introduction**:

People should be able to share the lesions to other computers.

**Input**:

User clicks on share the lesions button

**Processing**:

System takes the tutorials from database and sends them to the portable device as per user choice for sharing.

**Output**:

User gets the data and could easily copy them to the other portable device.

#### Generate grade card

**Introduction**:

After completing exercises, a grade card should be generated.

**Input**:

Kids complete the exercises within the given time.

**Processing:**

After matching the answers with predefined answers, system generates a report card and saves them in database.

**Output**:

User can see the report card after the exam and see them in future as well.

## Non-functional Requirements

* The application will be self-dependent and no dependency on other parties required.
* There will be a digital backup and restore system.
* There will be more opportunity to extend the application in various type of device in future.
* The response time will be low and the system will response fast.
* It will be very user friendly and usable by any person with minimal computer knowledge.
* In terms of security unauthorized access will be denied and register user will be able to change as necessary.
* It will be efficient as it reduces manual labor and searching.
* CES will have user manual and help documents.
* It is designed such a way that it can be maintained with minimal effort.

## Data models

### Context Diagram

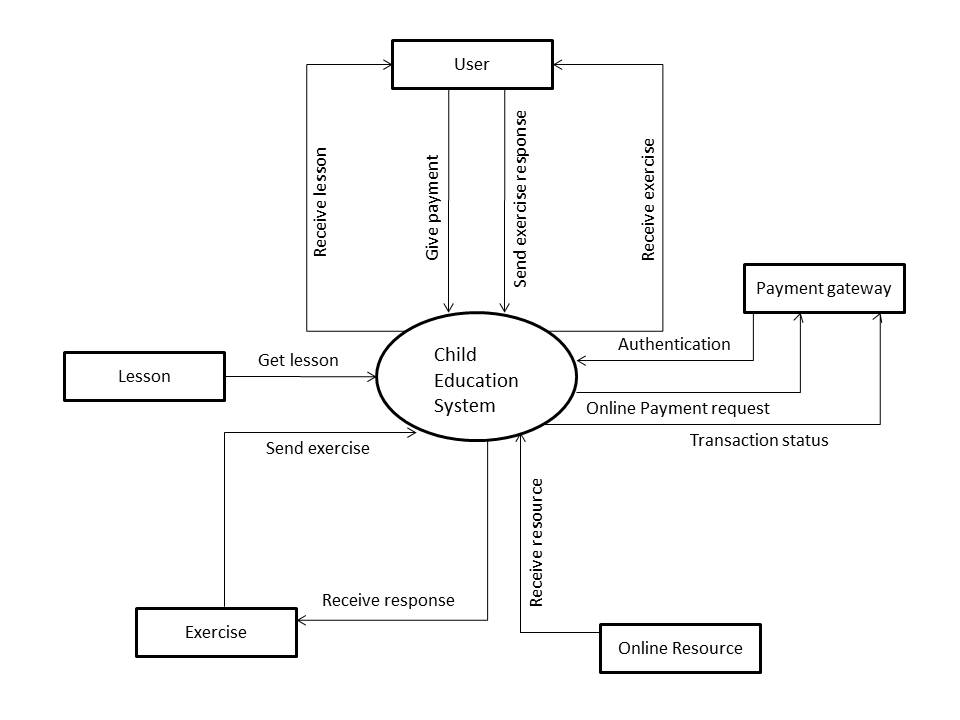


Figure 1: Context Diagram of CES

### 0-Level DFD

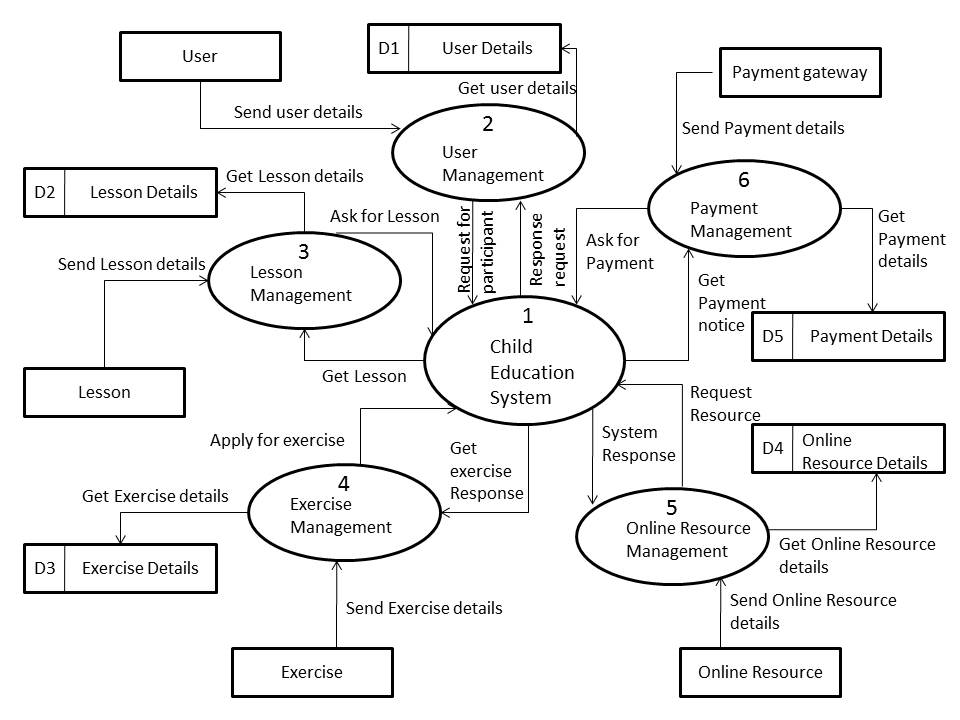


Figure 2: DFD level 0 of CES

### 1-Level DFD

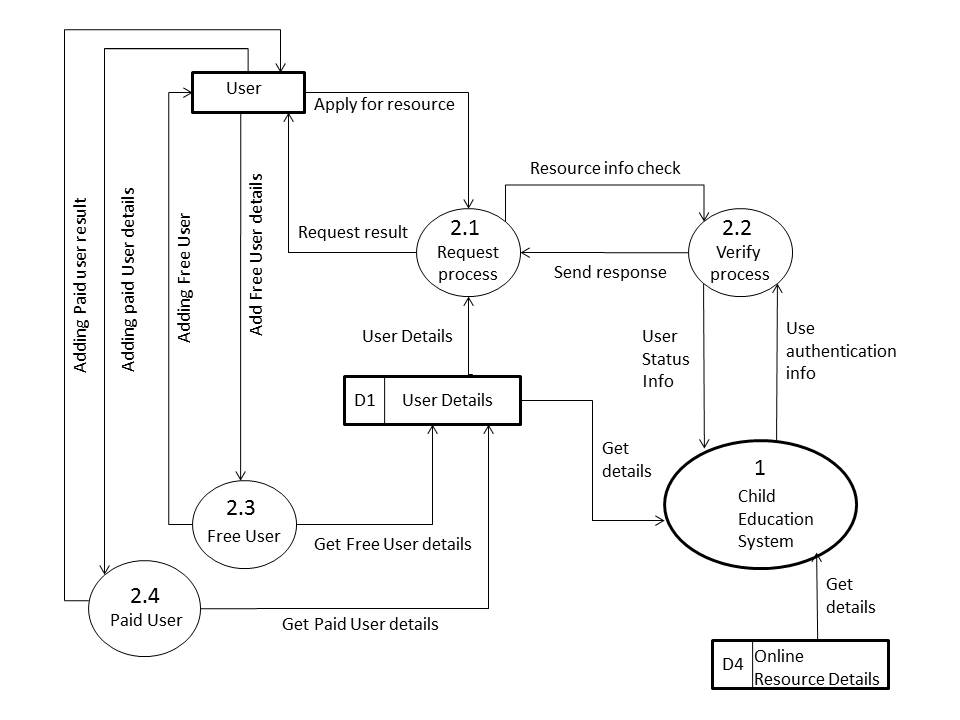


Figure 3: DFD level 1 of CES

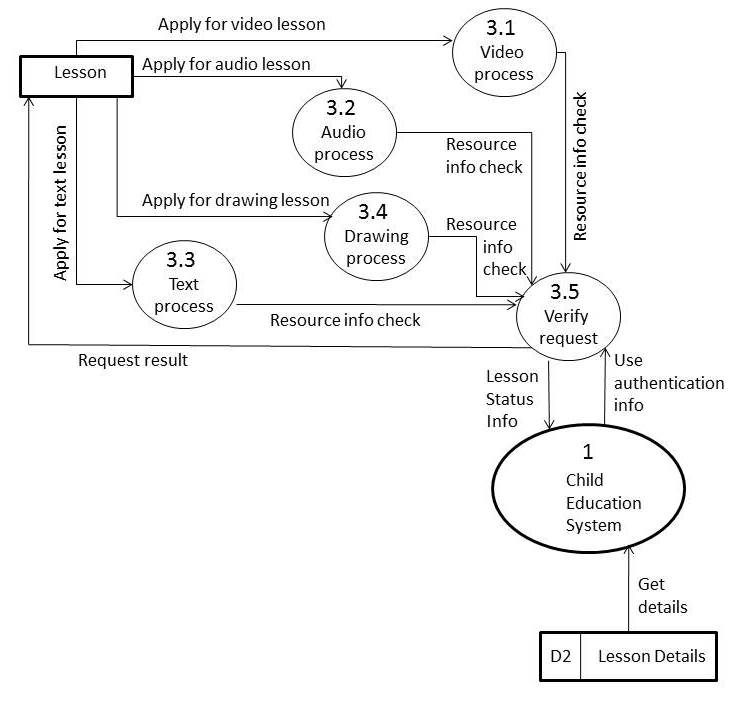


Figure 4: DFD level 1 of CES

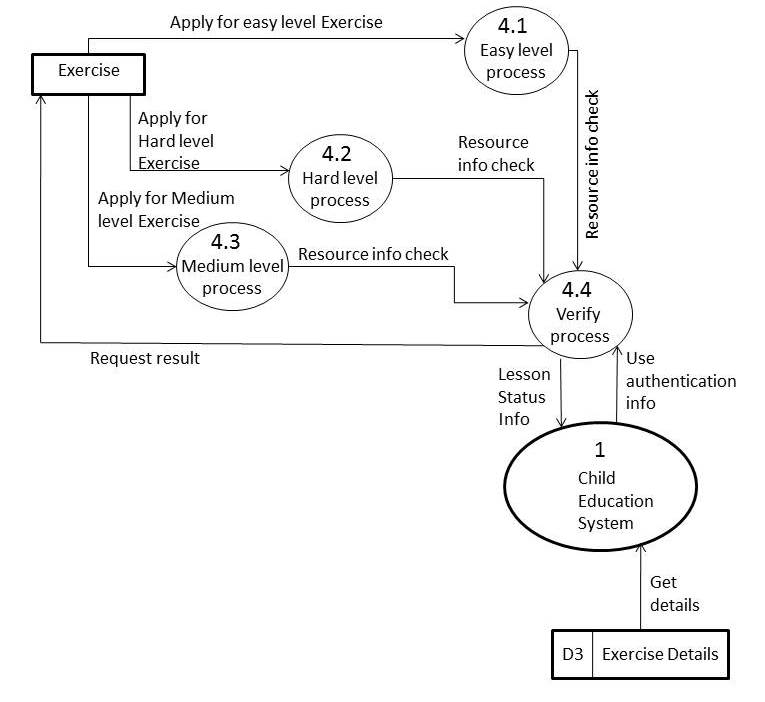


Figure 5: DFD level 1 of CES

### 2-Level DFD

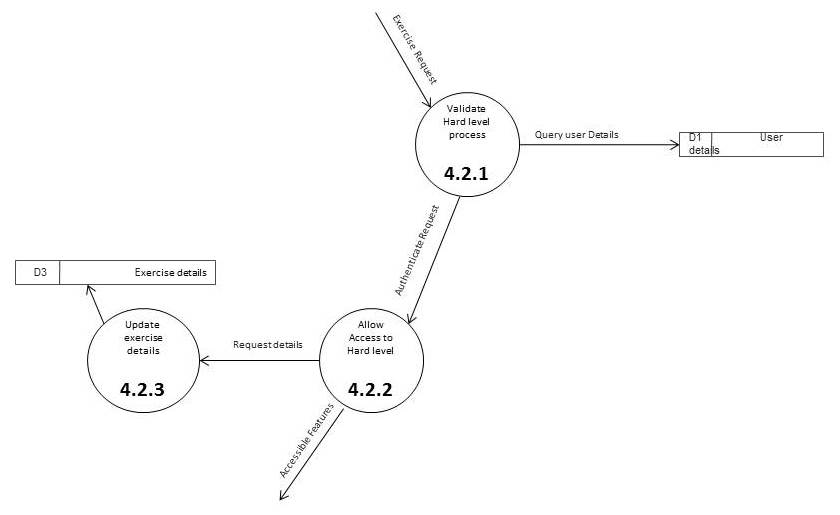
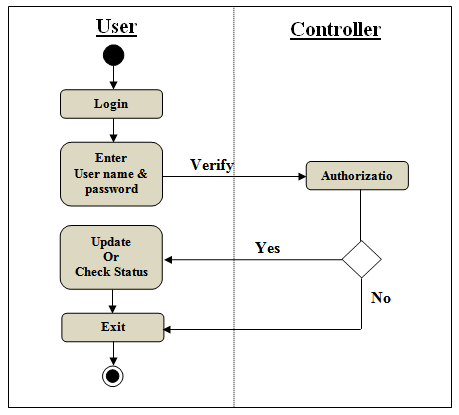


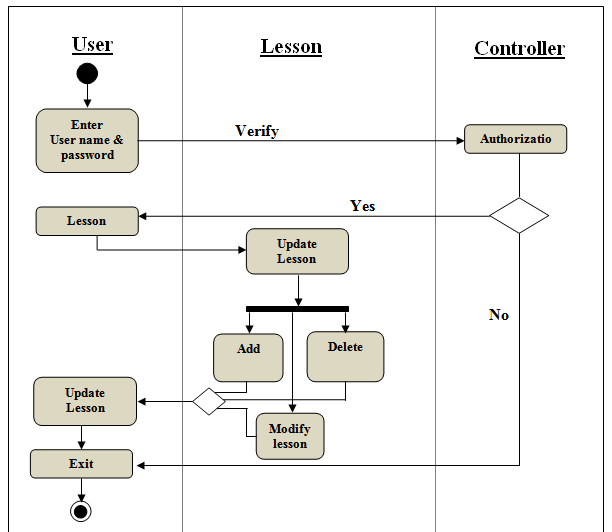
Figure 6: DFD level 2 of CES

## Control Flow Diagram

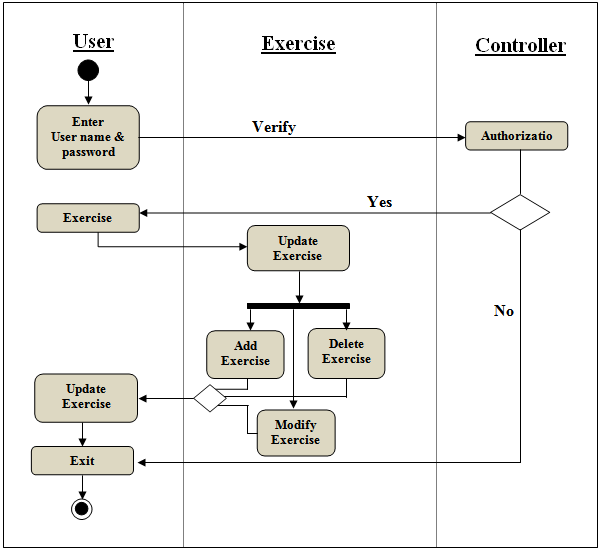
### User Login



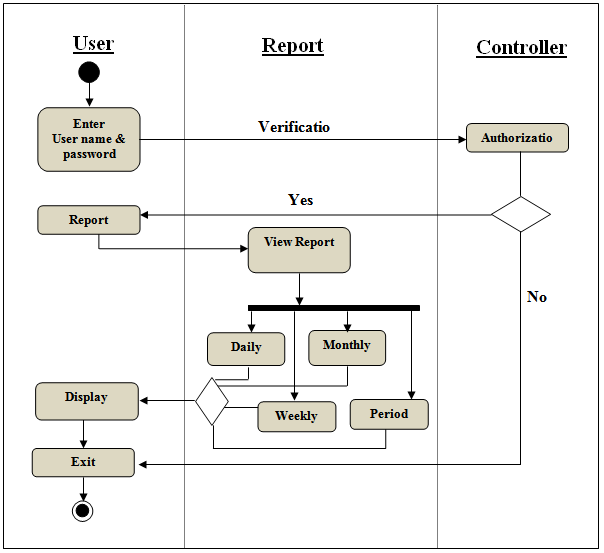
### Lesson Management



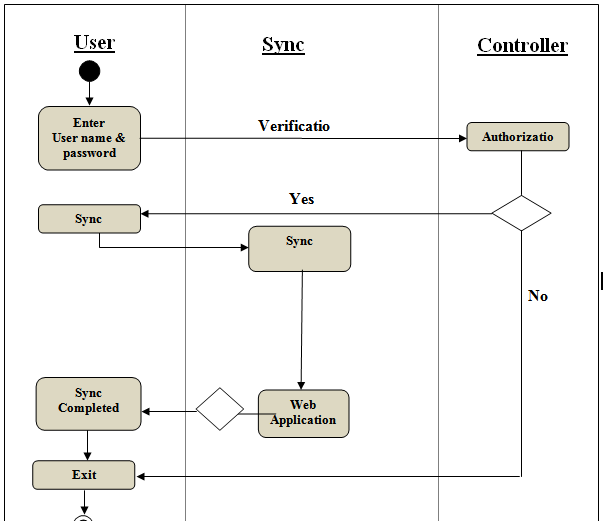
### Exercise Management



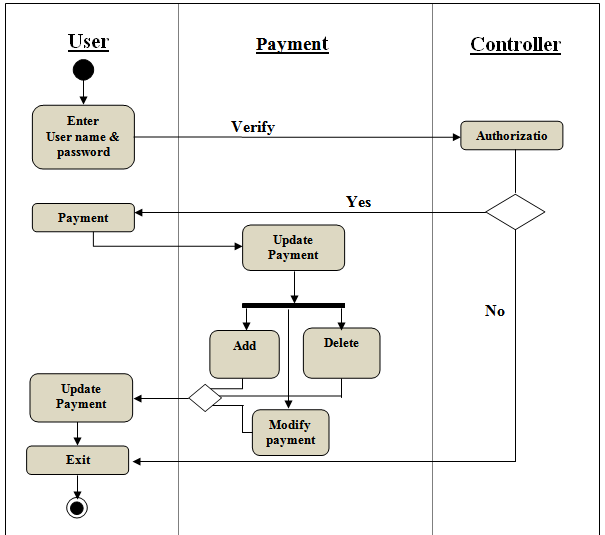
### View Report



### Sync



### Payment Management



## Sequence diagrams

## Entity Relationship Model,

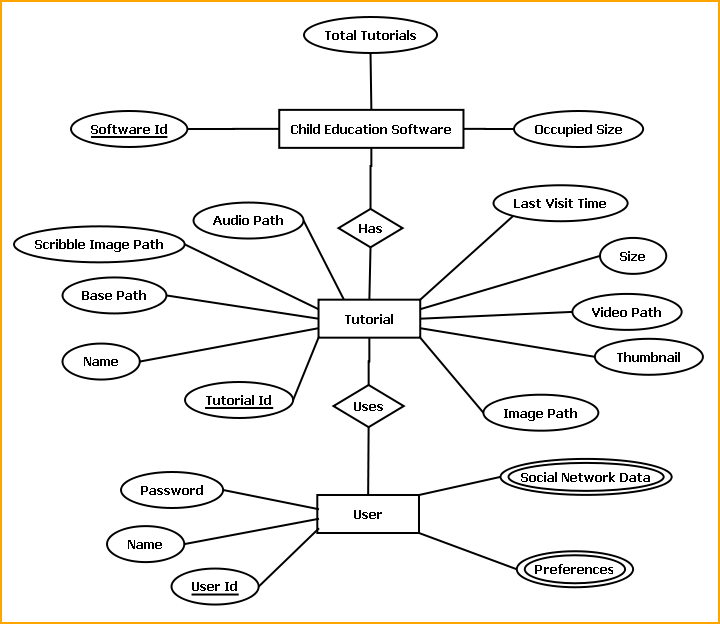
We will design a RDBMS for Child Education Software. The entities and their attributes are listed below. Attributes in Bold letter is the unique key.

|  |  |
| --- | --- |
| **Entities** | **Attributes** |
| Tutorial | **Tutorial Id**, Name, Base Path, Last Visit Time, Size, Thumbnail, video path, image path, audio path, scribble image path |
| Child Education Software | **Software Id**, Total tutorials , Occupied Size |
| User | **User Id**, Name, Social Network Data, password, Preferences. |

**Relationship between Entities:**

* Child Education Software has Tutorials 🡪 1 : N
* User uses tutorials 🡪 M : N

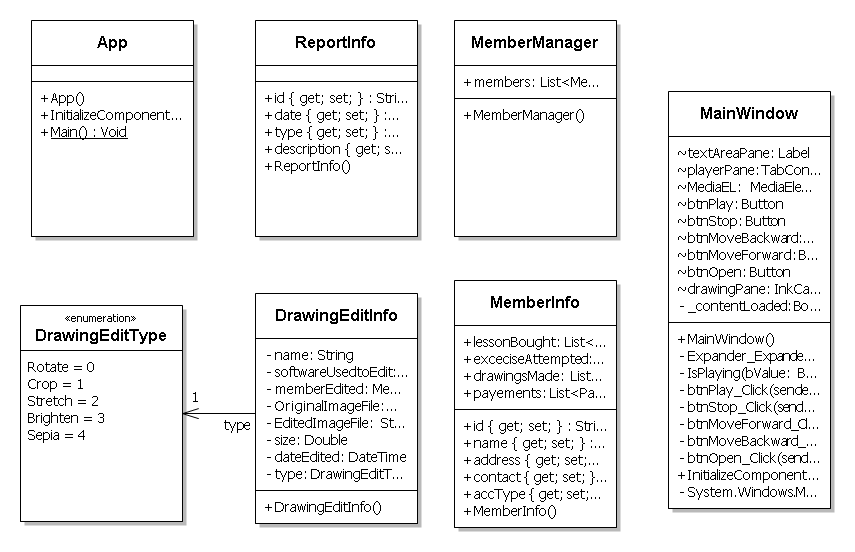
## E-R Diagram

****

E-R Diagram of Child Education Software

## Class Diagrams

****



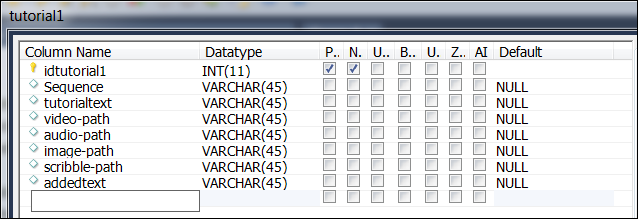
# System Design

## Modularisation details

# Database & Table Details

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

**Table:: Tutorial1**



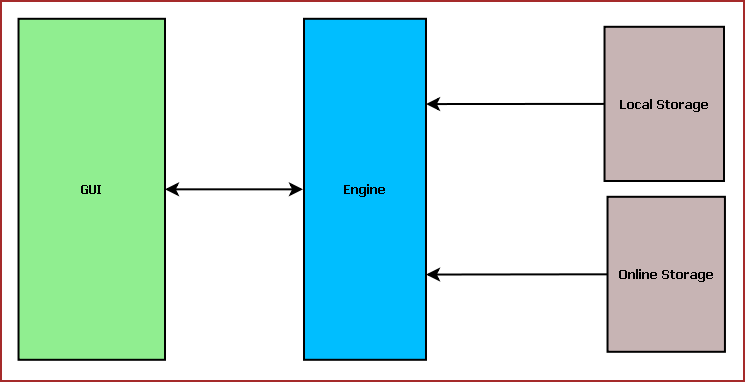
|  |  |
| --- | --- |
| **Tables** | **Attributes** |
| Tutorial1 | `tutorial1`.`idtutorial1`,`tutorial1`.`Sequence`, `tutorial1`.`tutorialtext`,`tutorial1`.`video-path`, `tutorial1`.`audio-path`,`tutorial1`.`image-path`, `tutorial1`.`scribble-path`, `tutorial1`.`addedtext` |

# Complete Structure

## Module Description

Child Education Software is divided into three main modules.

* GUI
* Engine
* Local Storage & Online Storage



**Figure:** Child Education System Components

### Child Education Software GUI:

Child Education Software GUI will display tutorial, navigator, online store interface. The main components of GUI are:

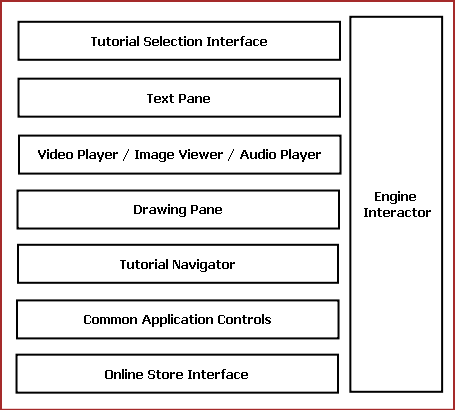
* Tutorial Selection Interface
* Text Pane
* Video Player / Image Viewer / Audio Player
* Drawing Pane
* Tutorial Navigator
* Common Application Control
* Online Store Interface
* Engine Interactor

#### Tutorial Selection Interface:

This interface will display the available tutorials from local storage and the tutorials bought from Online Store. From here the user will select the tutorial to be viewed. The tutorials will be displayed according to different category such as: Age, Subject, culture etc.

#### Text Pane:

This area will display the text portion of the tutorial. This will contain the description about the subject if different colours. Users can select text, copy and paste the text.



**Fig:** Components of Child Education Software GUI

#### Common Controls:

Common controls include several controls of the application. Such as: Menu Bar, Toolbar, status bar, Context Menu and progress bar.

#### Engine Interaction:

This module handles the interaction between GUI and Engine. This module defines all callback for the Engine events.

#### Video Player / Image Viewer / Audio Player:

This Pane displays the image, video & audio artwork associated with the tutorial content. This will have two components such as: Display Area & Control Pane. Display Area will display the content. Control Pane contains the control like Play, Pause, Previous, Next for controlling the displayed content.

#### Drawing Pane:

Drawing Pane allows user to draw, sketch & scribble. This will help the kids to draw and practice while learning. User can save the drawing as a Image or load a image to scribble.

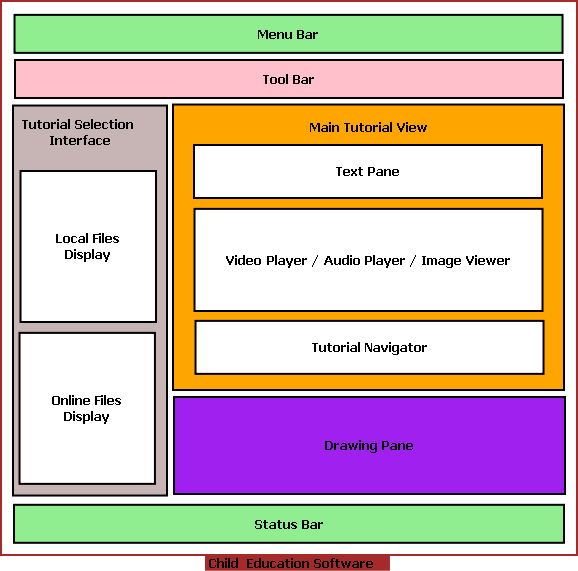
#### Tutorial Navigator:

This pane contains the controls for navigating within the tutorial. It will have “Next”, “Previous” and “Go to Home” options.

#### Online Store Interface:

This pane allows user to login into Online Store. Then it will display the available tutorial online for download. User may need to purchase or download it for free.

The GUI layout of Child Education Software is shown below:



### Child Education Software Engine:

Child Education Software Engine provides multimedia framework for displaying Video/Image, tutorial controlling logic. It is mainly divided into following components:

* Tutorial Controller
* Image Controller
* Archive Manager
* Video /Audio Controller
* Online Store Controller
* GUI Interactor
* Database Manager

### Tutorial Controller:

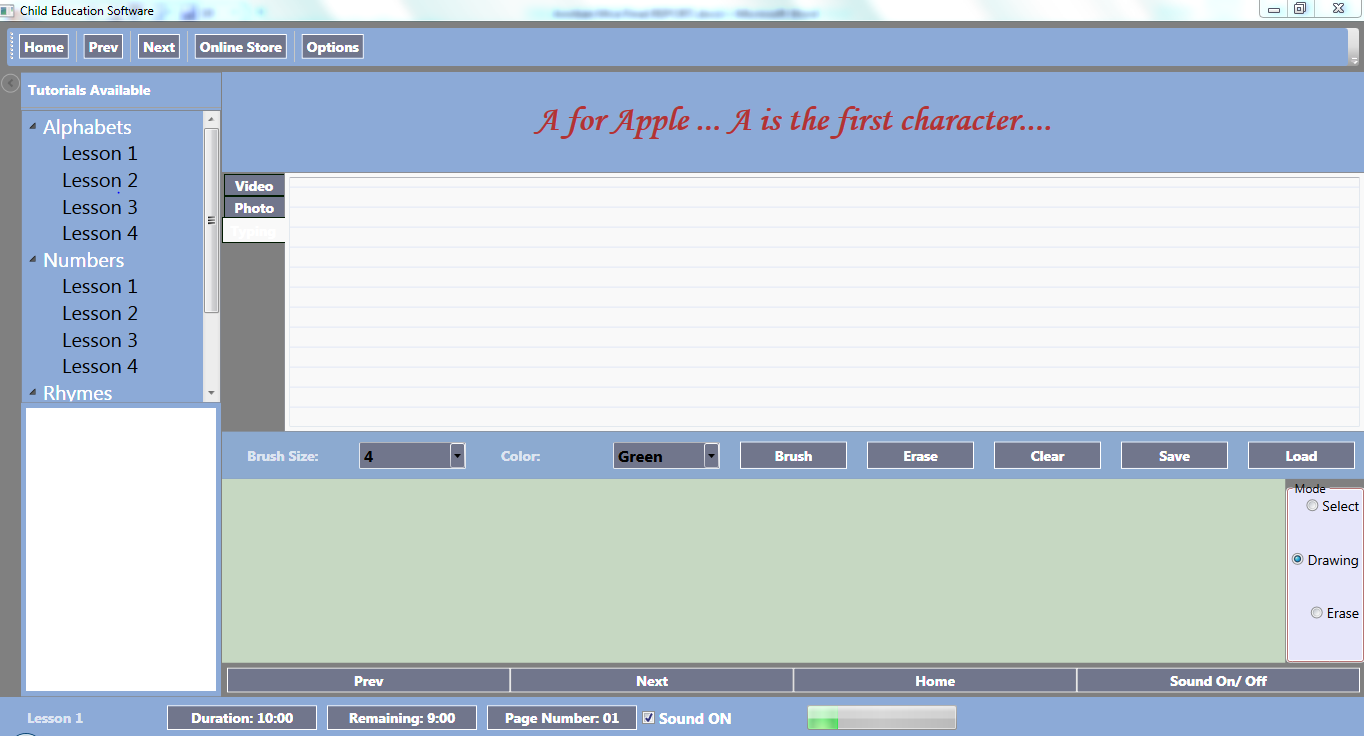
This module defines the tutorial format. It controls tutorial flow, content to be displayed. Every tutorial is internally a XML file mentioning the tutorial contents.

Tutorial controller has a XML parser to read the XML and load tutorial contents accordingly. Tutorial Controller controls the other engine components to coordinate tutorial display.

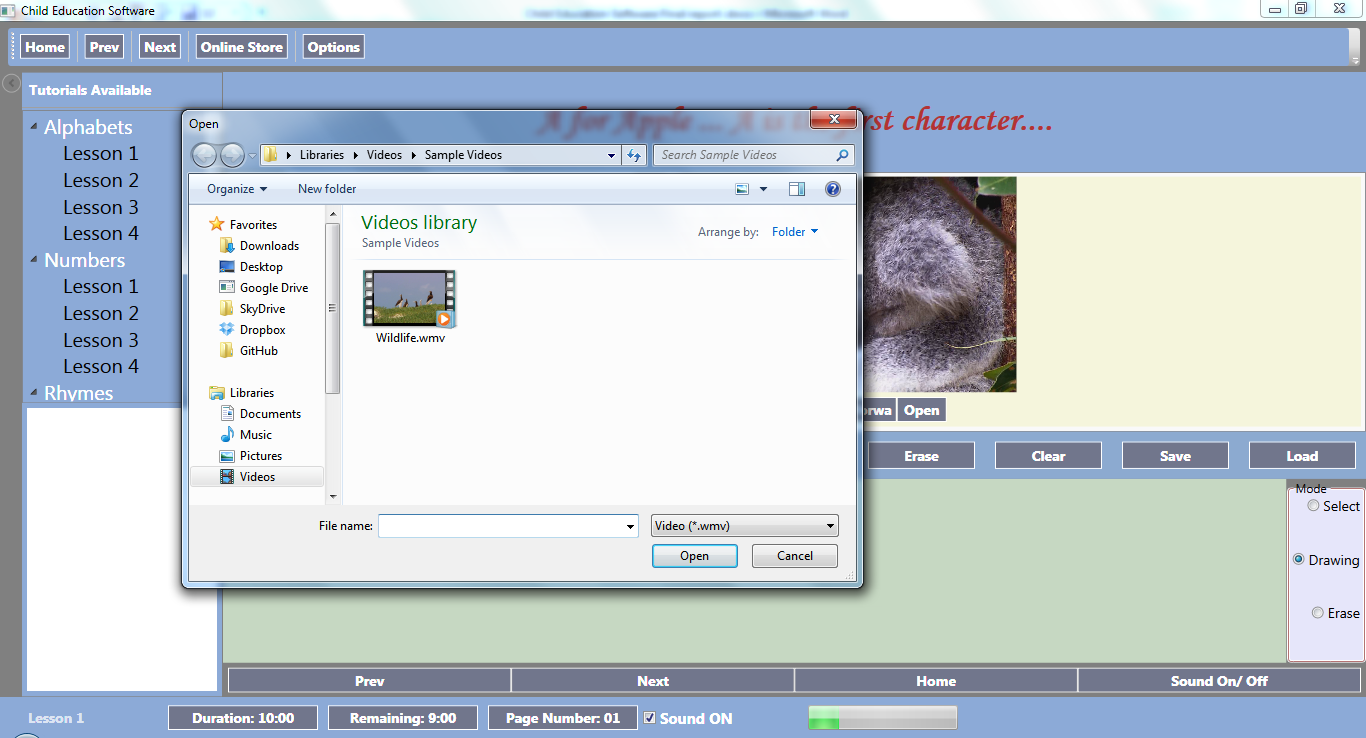
Different components of Child Education Software Engine and their interaction with other modules are displayed in the following diagram.

## User Interface Design

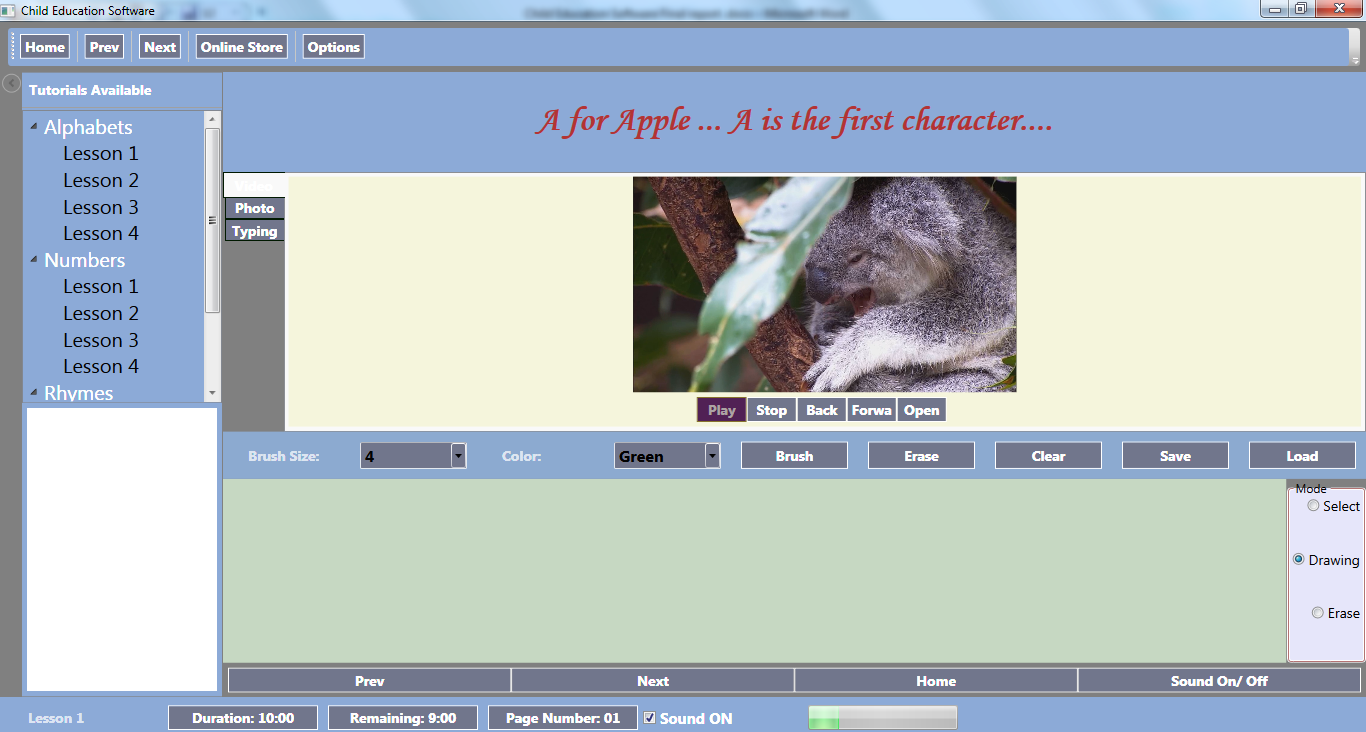
### Main window



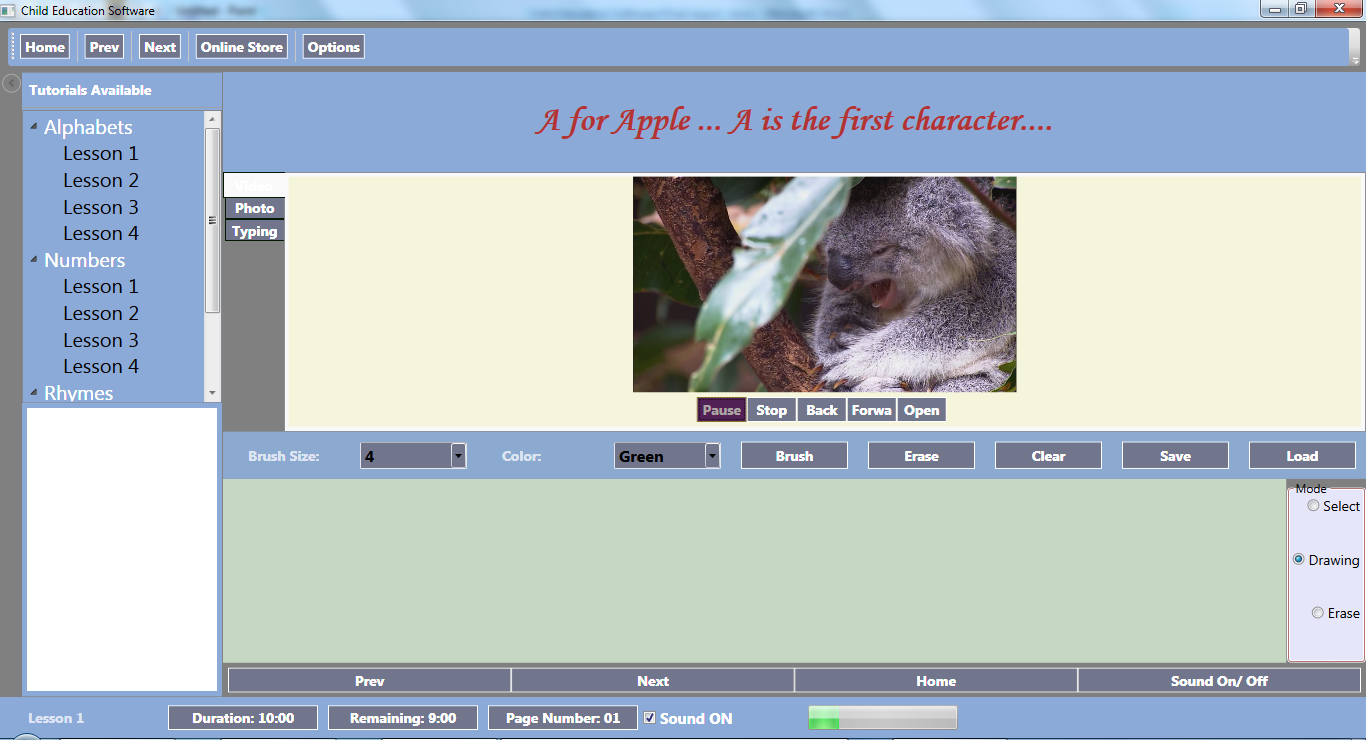
### Open Video



### Play video



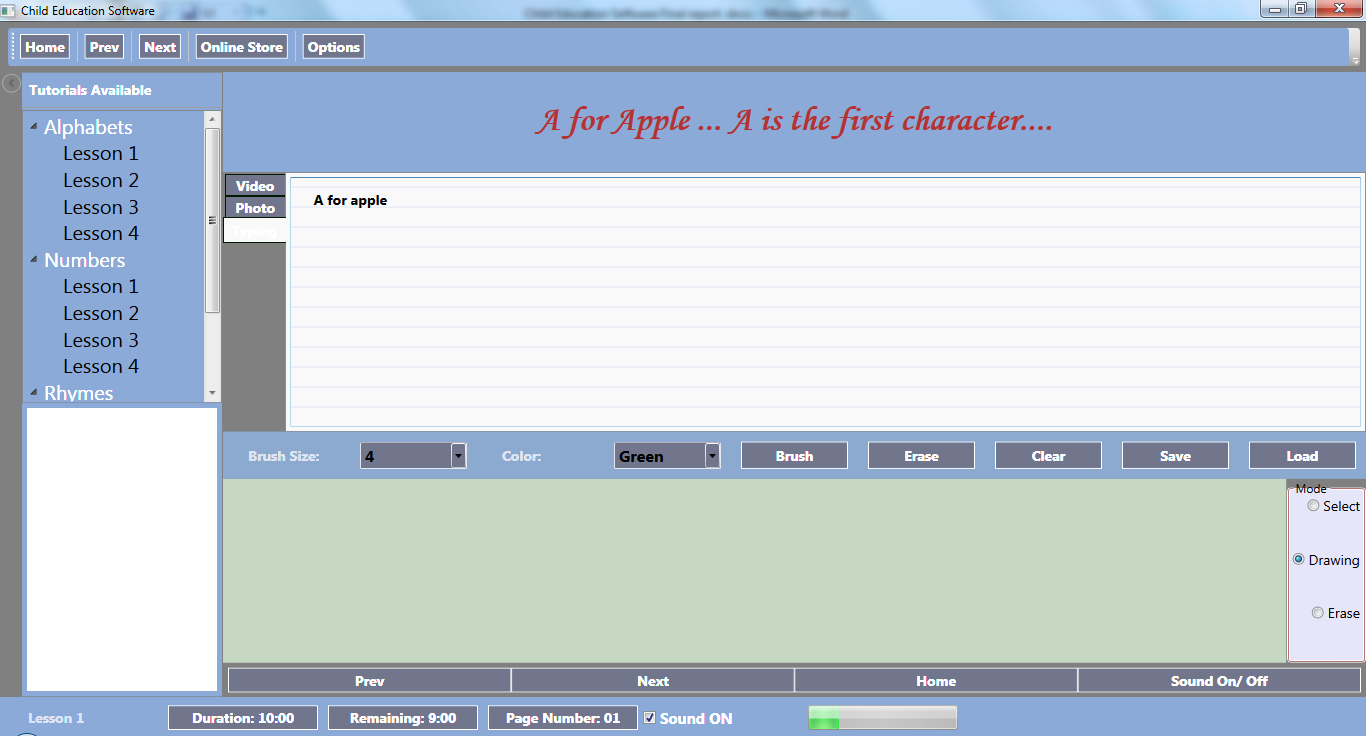
### Pause video



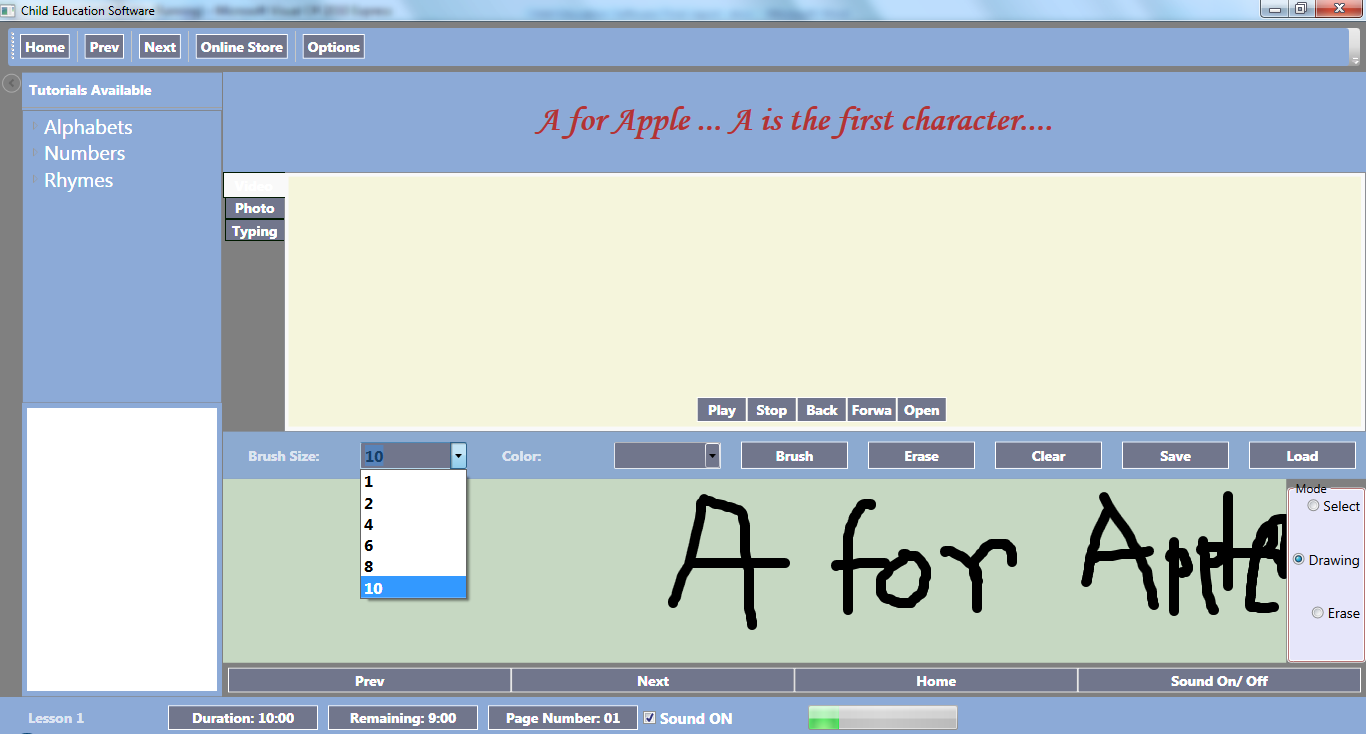
### Photo



### Typing



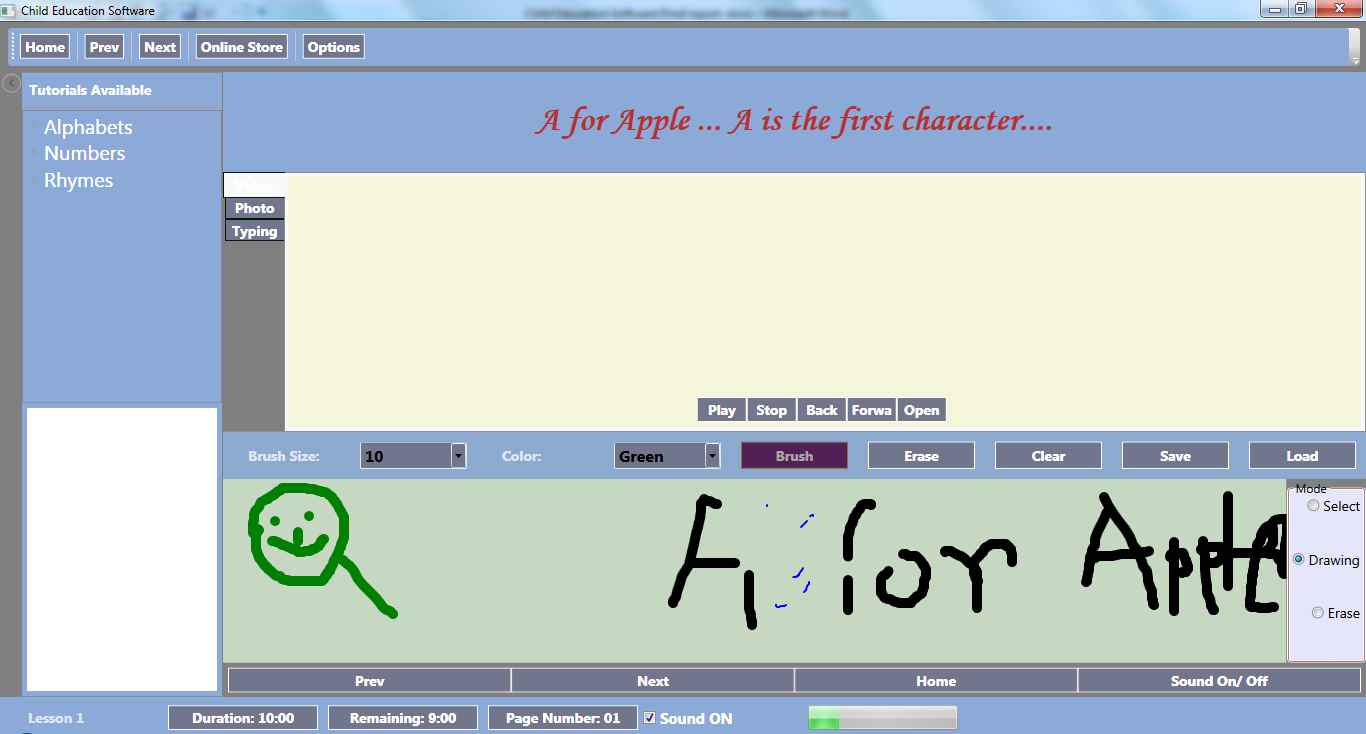
### Brush size



### Select color



### Erase



### Selection



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

# Coding

## Complete Project Coding

### Desktop Application Coding

### Mobile Application Coding

## Comments and Description of Coding segments

#### Code Commenting

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

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Comment block

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Example:

// TODO: Place Database Code Here

// UNDONE: Removed P\Invoke Call due to errors

// HACK: Temporary fix until able to refactor

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

Example:

/// <example>

/// Add the following key to the “appSettings” section of your config:

/// <code><![CDATA[

/// <configuration>

/// <appSettings>

/// <add key=”mySetting” value=”myValue”/>

/// </appSettings>

/// </configuration>

/// ]]></code> >

/// </example>

#### ChildEducationSoftwareEngine

##### Classes

|  |  |  |
| --- | --- | --- |
|  | Class | Description |
| Public class | [CESController](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\541674045.htm) |  |
| Public class | [MemberManager](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\644999019.htm) |  |
| Public class | [ExerciseManager](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\149433560.htm) |  |
| Public class | [LessonManager](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\741694114.htm) |  |
| Public class | [DrawingEditController](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\477062516.htm) |  |
| Public class | [MemberInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1373341276.htm) |  |
| Public class | [PaymentInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\346412356.htm) |  |
| Public class | [LessonInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\304630710.htm) |  |
| Public class | [ExerciseInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\2090516743.htm) |  |
| Public class | [ScoreInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1339366881.htm) |  |
| Public class | [DrawingInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\424069616.htm) |  |
| Public class | [TextInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\2072101315.htm) |  |
| Public class | [AudioInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1582667939.htm) |  |
| Public class | [VideoInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1488063546.htm) |  |
| Public class | [ImageInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1131969157.htm) |  |
| Public class | [DrawingEditInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1062276712.htm) |  |
| Public class | [ReportInfo](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\905045251.htm) |  |

##### Enumerations

|  |  |  |
| --- | --- | --- |
|  | Enumeration | Description |
| Public enumeration | [AccountType](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1041126080.htm) |  |
| Public enumeration | [DrawingEditType](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\745124001.htm) |  |
| Public enumeration | [ReportType](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1246053874.htm) |  |

#### ChildEducationSoftwareGUI Namespace

##### Classes

|  |  |  |
| --- | --- | --- |
|  | Class | Description |
| Public class | [App](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1973931447.htm) | Interaction logic for App.xaml |
| Public class | [MainWindow](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1917943267.htm) | Interaction logic for MainWindow.xaml |

#### ChildEducationSoftwareGUI.Properties Namespace

##### Classes

|  |  |  |
| --- | --- | --- |
|  | Class | Description |
| Protected class | [Settings](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\114054946.htm) |  |
| Protected class | [Resources](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\2064594663.htm) | A strongly-typed resource class, for looking up localized strings, etc. |

#### ChildEducationSoftwareStyle.Properties Namespace

##### Classes

|  |  |  |
| --- | --- | --- |
|  | Class | Description |
| Protected class | [Settings](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\343051425.htm) |  |
| Protected class | [Resources](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\121570590.htm) | A strongly-typed resource class, for looking up localized strings, etc. |

#### ChildEducationSoftwareStorage Namespace

##### Classes

|  |  |  |
| --- | --- | --- |
|  | Class | Description |
| Public class | [ChildEducationSoftwareStorage](file:///C:\Users\chandra\Documents\GitHub\ChildEducationSoftware\Help\html\1153410324.htm) |  |

## 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:

1. Member variables.
2. Constructors & Finalizers.
3. Nested Enums, Structs, and Classes.
4. Properties
5. Methods

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

1. Public
2. Protected
3. Internal
4. 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 separate declaration.

Example:

// Bad!

[Attrbute1, Attrbute2, Attrbute3]

public class MyClass

{…}

// Good!

[Attrbute1, RelatedAttribute2]

[Attrbute3]

[Attrbute4]

public class MyClass

{…}

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

## Code Efficiency

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

## Error handling

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

In many cases, an exception may be thrown not by a method that your code has called directly, but by another method further down in the call stack. When this happens, the CLR will unwind the stack, looking for a method with a **catch** block for the specific exception type, and it will execute the first such **catch** block that if finds. If it finds no appropriate **catch** block anywhere in the call stack, it will terminate the process and display a message to the user.

[Exceptions Overview](javascript:void(0))

Exceptions have the following properties:

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

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

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

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

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

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

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

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

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

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

## Parameters calling/passing

## Validation checks

# Testing

## Testing techniques and testing strategies used

**CES** 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 **CES** 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 **CES**:

|  |  |
| --- | --- |
| 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 **CES**.

|  |  |
| --- | --- |
| 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 **CES** 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

### Creation of Test Plan

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

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

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

**Figure 2.3   Creating a Test Plan**



### Testing Scope and Objectives

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

### Testing Methodology

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

* The domain architecture used for testing
* The tools and techniques used to conduct the tests or to measure results
* Automated techniques we plan to use during testing

### Features and Functions to Test

Included tests that verify or address:

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

### Risk Factors

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

### Testing Schedule

We drafted a preliminary schedule that includes each test listed in the test plan. The schedule can help us coordinate test lab use among sub teams. Assign a team member, ideally the test lab manager, if our team has one, to maintain and update the lab schedule. Having an up-to-date schedule is critical when unscheduled lab requests are submitted.

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

### Test reports for Unit Test Cases

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

### Test reports for System Test Cases

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

## Debugging and Code improvement:

The steps in the bellow section demonstrate how to create a console application that uses the **Debug** class to provide information about the program execution.   
  
When the program is run, we can use methods of the **Debug** class to produce messages that help we to monitor the program execution sequence, to detect malfunctions, or to provide performance measurement information. By default, the messages that the **Debug** class produces appear in the Output window of the Visual Studio Integrated Development Environment (IDE).   
  
The sample code uses the **WriteLine** method to produce a message that is followed by a line terminator. When we use this method to produce a message, each message appears on a separate line in the Output window.   
  
When we use the **Assert** method of the **Debug** class, the Output window displays a message only if a specified condition evaluates to false. The message also appears in a modal dialog box to the user. The dialog box includes the message, the project name, and the **Debug.Assert** statement number. The dialog box also includes the following three command buttons: 

* **Abort:**The application stops running.
* **Retry:**The application enters debug mode.
* **Ignore:**The application proceeds.

The user must click one of these buttons before the application can continue.   
  
We can also direct output from the **Debug** class to destinations other than the Output window. The **Debug** class has a collection named **Listeners** that includes **Listener** objects.  
  
Each **Listener** object monitors **Debug** output and directs the output to a specified target.   
  
Each **Listener** in the **Listener** collection receives any output that the **Debug** class generates. Use the **TextWriterTraceListener**class to define **Listener** objects. We can specify the target for a **TextWriterTraceListener** class through its constructor.   
  
Some possible output targets include the following:

* The Console window by using the **System.Console.Out** property.
* A text (.txt) file by using the **System.IO.File.CreateText("FileName.txt")** statement.

After we create a **TextWriterTraceListener** object, we must add the object to the **Debug.Listeners** collection to receive Debug output. 

### Create a Sample with the Debug Class

1. Start Visual Studio or Visual C# Express Edition.
2. Create a new Visual C# Console Application project named **conInfo**. Class1 is created in Visual Studio .NET. Program.cs is created in Visual Studio 2005.
3. Add the following namespace at top in Class1 or Program.cs.

using System.Diagnostics;

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

double dUnitCost = 1.03;

1. Specify the message that the class produces as the first input parameter of the **WriteLine** method. Press the CTRL+ALT+O key combination to make sure that the Output window is visible.

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

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

Debug.Indent();

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

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

1. We can also use the **WriteLine** method to display the namespace and the class name for an existent object. For example, the following code displays the **System.Xml.XmlDocument** namespace in the Output window:
2. System.Xml.XmlDocument oxml = new System.Xml.XmlDocument();

Debug.WriteLine(oxml);

1. To organize the output, we can include a category as an optional, second input parameter of the **WriteLine** method. If we specify a category, the format of the Output window message is "category: message." For example, the first line of the following code displays "Field: The product name is Widget" in the Output window:
2. Debug.WriteLine("The product name is " + sProdName,"Field");
3. Debug.WriteLine("The units on hand are" + iUnitQty,"Field");
4. Debug.WriteLine("The per unit cost is" + dUnitCost.ToString(),"Field");

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

1. The Output window can display messages only if a designated condition evaluates to true by using the **WriteLineIf**method of the **Debug** class. The condition to be evaluated is the first input parameter of the **WriteLineIf** method. The second parameter of **WriteLineIf** is the message that appears only if the condition in the first parameter evaluates to true.
2. Debug.WriteLineIf(iUnitQty > 50, "This message WILL appear");
3. Debug.WriteLineIf(iUnitQty < 50, "This message will NOT appear");
4. Use the **Assert** method of the **Debug** class so that the Output window displays the message only if a specified condition evaluates to false:
5. Debug.Assert(dUnitCost > 1, "Message will NOT appear");
6. Debug.Assert(dUnitCost < 1, "Message will appear since dUnitcost < 1 is false");
7. Create the **TextWriterTraceListener** objects for the Console window (tr1) and for a text file named Output.txt (tr2), and then add each object to the **Debug Listeners** collection:
8. TextWriterTraceListener tr1 = new TextWriterTraceListener(System.Console.Out);
9. Debug.Listeners.Add(tr1);
11. TextWriterTraceListener tr2 = new TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));

Debug.Listeners.Add(tr2);

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

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

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

Debug.Flush();

### Using the Trace Class

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

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

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

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

Trace.Indent();

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

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

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

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

Trace.Unindent();

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

Trace.Flush();

Console.ReadLine();

### Verify That It Works

1. Make sure that **Debug** is the current solution configuration.
2. If the**Solution Explorer** window is not visible, press the CTRL+ALT+L key combination to display this window.
3. Right-click **conInfo**, and then click **Properties**.
4. In the left pane of the conInfo property page, under the **Configuration** folder, make sure that the arrow points to**Debugging**.  
     
   **Note** In Visual C# 2005 and in Visual C# 2005 Express Edition, click **Debug** in the **conInfo** page.
5. Above the **Configuration** folder, in the **Configuration** drop-down list box, click **Active (Debug)** or **Debug**, and then click **OK**. In Visual C# 2005 and in Visual C# 2005 Express Edition, click **Active (Debug)** or **Debug** in the **Configuration**drop-down list box in the **Debug** page, and then click **Save** on the **File** menu.
6. Press CTRL+ALT+O to display the Output window.
7. Press the F5 key to run the code. When the **Assertion Failed** dialog box appears, click **Ignore**.
8. In the Console window, press ENTER. The program should finish, and the Output window should display the output that resembles the following
9. Debug Information-Product Starting
10. The product name is Widget
11. The available units on hand are100
12. The per unit cost is 1.03
13. System.Xml.XmlDocument
14. Field: The product name is Widget
15. Field: The units on hand are100
16. Field: The per unit cost is1.03
17. Calc: Total Cost is 103
18. This message WILL appear
19. ---- DEBUG ASSERTION FAILED ----
20. ---- Assert Short Message ----
21. Message will appear since dUnitcost < 1 is false
22. ---- Assert Long Message ----
23. at Class1.Main(String[] args) <%Path%>\class1.cs(34)
24. The product name is Widget
25. The available units on hand are100
26. The per unit cost is 1.03
27. Debug Information-Product Ending
28. Trace Information-Product Starting
29. The product name is Widget
30. Field: The product name isWidget
31. This message WILL appear
32. Trace Information-Product Ending
34. The Console window and the Output.txt file should display the following output:
35. The product name is Widget
36. The available units on hand are 100
37. The per unit cost is 1.03
38. Debug Information-Product Ending
39. Trace Information-Product Starting
40. The product name is Widget
41. Field: The product name is Widget
42. This message WILL appear
43. 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 user this application with very limited access to the user data.

# Cost Estimation of the Project along with Cost Estimation Model

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

Effort = a1 \* (KLOC)a2 PM

Tdev = b1 \* (Effort)b2 months

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 **CES**, the formula for estimating the effort based on the code size is shown below:

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

## Estimation of development time

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

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

Assume that the size of a Semi-detached CES product has been estimated to be 4,000 lines of source code. Assume that the average salary of software engineer(me) is Rs. 15,000 per month.

Assume that the size of our

The basic COCOMO estimation formula for CES semidetached software:

Our Effort = 3.0 \* (4)1.12 PM

= 14 PM

Normal Development time = 2.5 \* (14)0.35 months

= 6 months

Cost required to develop the product = Rs. 6 \* 15,000

= Rs. 90,000

# Reports

* List of Lesion updates could be downloading.
* Hand writing could be generated.
* A list of events could be generated.
* Typing test could be generated.
* List of google plus update could be generated.

# Future scope and further enhancement of the Project

* Now it will display the text based RSS feeds and link of the multimedia contents. We will display the Multimedia contents like Video, Audio & Image in future.
* To support UNIX / Linux Based Operating systems.
* To Support Mobile Operating systems for Symbian, Meego & Android.

# Bibliography

## Website

* <http://en.wikipedia.org>
* http://en.wikipedia.org/wiki/Windows\_Presentation\_Foundation
* <http://msdn.microsoft.com/en-us/>
* http://www.c-sharpcorner.com/beginners/
* <http://www.microsoft.com/en-us/default.aspx>
* <http://www.codeplex.com/>
* <http://stackoverflow.com/>
* <http://www.codeguru.com/>
* http://www.csharpcourse.com/
* <http://www.w3schools.com>
* <http://blogs.technicise.com/>
* http://connect.technicise.com/
* <http://www.mysql.com/>
* http://www.mysql.com/support/
* <http://dev.mysql.com/>
* http://dev.mysql.com/support/
* <http://www.homeandlearn.co.uk/csharp/csharp.html>
* <http://www.wpftutorial.net/Home.html>
* <http://www.youtube.com/?gl=IN>
* <http://blendinsider.com/>
* https://github.com/
* <https://github.com/anirban-nandy>
* [https://github.com/anirban-nandy/DailyNoteBook](https://github.com/%20anirban-nandy%20/DailyNoteBook)
* <http://learn.github.com/p/intro.html>
* <http://www.vogella.com/articles/Git/article.html>
* <http://try.github.com/levels/1/challenges/1>
* <https://enterprise.github.com/support>
* <https://support.enterprise.github.com/home>

## Books

* 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

## Mobile App Development

Quickly and efficiently create and test Java™ applications with the Nokia SDKs for Java and the Series 40 platform SDKs.

The Nokia SDKs for Java provide the development tools for phones containing a Java Runtime for Series 40. Each release of the Java Runtime has a corresponding Nokia SDK.

For earlier platform releases Series 40 platform SDKs offer versions to support specific editions and feature packs.

Within both families of tools, each SDK includes Java APIs, an emulator, documentation, code examples, and emulator based debugging tools. The SDKs can be used with either the NetBeans or Eclipse IDEs to create, compile, and package applications and content. Applications can be tested using the emulator.

### Nokia SDK 2.0 for Java — for Series 40 apps

|  |
| --- |
|  |

Create apps for Series 40 phones with the Java Runtime 2.0.0, including the full-touch UI equipped Nokia Asha 305, Nokia Asha 306, and Nokia Asha 311 using the Nokia SDK 2.0 for Java. Then test our apps in an emulator based on the Nokia Asha 305. In addition to the features of the Nokia SDK 1.1 for Java, the 2.0 SDK offers:

#### LWUIT for Series 40 arrives at 1.0

LUWIT for Series 40 has graduated beta to a full productised release. With a number of new APIs — such as PopUpChoiceGroup, ContextMenu, and NokiaListCellRenderer — the 1.0 release includes significant improvements in performance, particularly in lists, themes loading, and the HTMLComponent. Compatibility with the native full-touch UI has been fine-tuned and many bugs fixed, particularly in command handling and text input. A LWUIT Developer's Library has also been released, providing full technical and design guides. There are many new examples too.  
  
This release is delivered to the Nokia SDK 2.0 for Java through the SDK Manager, while a download for the Nokia SDK 1.1 for Java is available from LWUIT for Series 40 project. 

#### The Nokia SDK 2.0 for Java adds new features

The Nokia SDK 2.0 for Java delivers everything you need to develop apps for the exciting new full-touch UI equipped Nokia Asha 305, Nokia Asha 306, and Nokia Asha 311 phones. And now the SDK has graduated from beta.

The SDK delivers the updated Nokia UI API for advanced touch interaction, the Mobile Sensor API (JSR-256) to take advantage of the orientation sensors on the latest Series 40 phones, and the in-app purchase APIs. Now LWUIT is included as a plug-in, so you can create slick UIs, faster. In addition, the emulator gains features for the simulation of multipoint-touch gestures, such as pinch-to-zoom, and PC keyboard input. Based on the Nokia Asha 305, the emulator also provides improved sensor and location support so you can test more of our app on a PC.

The Nokia IDE for Java ME (Eclipse) has also been enhanced with improved searching in the Device SDK Manager and a tool that lets you pull code examples directly into the IDE. Building on the power of the Eclipse platform for Java development, the Nokia IDE makes delivering our Series 40 Java apps easier with features such as an editor for Nokia specific JAD attributes.

#### Testing our Java apps for Series 40 using Remote Device Access

Testing our Java apps on several Series 40 phones is easy and cheap with remote Device Access. Remote Device Access offers a range of Series 40 phones that you access over the internet free-of-charge. So when you need to test our app you can simply pop-on the internet, book a phone, install our app, and you will be testing in minutes.   
  
Right now, the Nokia Asha 311 is available for you to test our apps.

#### Explore in-app purchasing in our Java apps

Using the Nokia SDK 1.1 for Java or later you can explore adding in-app purchase features to our apps, a feature available on phones with Java Runtime 1.1.0 for Series 40 or later. Now you can generate revenue by offering users digital assets and content as part of the app experience — and allow them to purchase these items without leaving our application.

#### Add a new dimension to location with the Maps API

Leveraging the location information provided by the Location API for J2ME™ (JSR-179) on Series 40 phones, you can add rich maps to our apps with the Maps API for Java ME. With exciting features, such as custom overlays, you can create a unique experience. And with the release of the Nokia SDK 2.0 for Java, you now get the Maps APIs delivered ready for our use without additional downloads. Find out more about creating location aware applications with Java technology ›

### Nokia Web - Tools

#### Series 40 web apps tools

##### Nokia Web Tools 2.3

###### **Description**

Nokia Web Tools provides a set of tools that enable the creation of Series 40 web apps. The tools included are:

* Web Developer Environment (WDE) — enabling web apps to be created, edited, packaged and deployed.
* Web App Simulator (WAS) — enabling web apps to be previewed and debugged on a computer.
* Web Developer Channel (WDC) — included in Web Developer Environment, to deliver information and tools to facilitate web app development.

###### **What's new**

This new version of Nokia Web Tools provides:

* UI Designer in WDE offering drag-and-drop population of the web app’s UI.
* the ability to deploy a web app over a USB connection from a PC running Microsoft Windows.
* automatic reloading of the simulator for locally previewed web apps as code changes are saved.
* additional templates upon which to base new web apps, including web apps for trivia games, shopping, and video browsing among others.
* more sample web apps and snippets to help developers use the platform capabilities easily.

###### **Symbian WRT widget development**

This version of Nokia Web Tools no longer supports Symbian WRT widget development. If you wish to continue using [Nokia Web Tools 1.2](http://www.developer.nokia.com/info/sw.nokia.com/id/054b94d5-4cdf-4b17-b268-bedcfc421ba6/Nokia_Web_Tools_1_2_for_Symbian.html) for Symbian WRT widget development, please refer to the Installation Guide for details on the setup requirements.

###### **App publication**

Web apps created with Nokia Web Tools can be submitted for publication in Nokia Store.

###### **Versions available**

Nokia Web Tools are available in versions for:

* 32- or 64-bit Microsoft Windows XP Service Pack 2, Windows Vista, or Windows 7.
* 32-bit Ubuntu Linux 10.04.
* 64-bit Apple Mac OS X 10.6.

To make your Series 40 web apps development as straightforward as possible Nokia Web Tools, Bluetooth Launcher, and the Nokia Xpress Browser are available in the Series 40 web apps section.

Alternatively, Xpress Web App Builder is an online tool that enables content owners to create web apps from clipped content, RSS feeds, and social media content using a wide selection of formats.

You create Series 40 web apps using Nokia Web Tools. Based on Eclipse, Nokia Web Tools builds on the powerful web development features of the Eclipse Web Tools Platform to create the Web Developer Environment (WDE). WDE includes features to create, edit, validate, test, package, and deploy Series 40 web apps. Testing is supported by the Web Apps Simulator (WAS) that enables web apps to be run and tested on a computer. WAS includes an implementation of Web Inspector, so you can perfom debugging and examine of a web app's content and performance. This getting started guide takes you through installing Nokia Web Tools, creating a web app from a template, testing it on your computer, and running it on a phone, before providing links to the resources you need to build great web apps and deliver them to Nokia Store.

WDE offers a number of templates you can use to create Series 40 web apps easily. These templates range from the Basic web app template, which contains the core web app files with no functionality, through a selection of templates offering basic UI constructions to fully functional web apps, such as the Videos browsing project template that offers a working web app to browse videos. You can work with web app examples or a web app project you have already created as well. For more details on importing web apps, see [Importing a web app or web app project](http://www.developer.nokia.com/Resources/Library/Series_40_web_apps_library/#%21tools-library/creating-a-web-app-project/importing-an-existing-web-app-or-web-app-project.html) in the Series 40 Web App Developer's Library.

During development, transferring a web app onto a phone each time you make code changes isn’t a practical way of previewing and testing your web app. To simplify testing of a web app, WDE integrates with the Web App Simulator (WAS) to enable testing on your computer.

You have two options for running your web app in WAS:

* a server (cloud) based preview, this option provides a simulation that is very close to the experience that will be seen on a phone.
* a local preview, which is useful when you are working offline or want to debug your web app.

Having tested your web app in the simulator the next stage is to run it on a Series 40 phone. You have three ways to do this::

* deploying the web app to a phone from WDE over a Bluetooth connection.
* deploying the web app from WDE to a phone over a USB connection (but only if you are working on a Microsoft Windows PC as [Nokia Suite](http://www.nokia.com/nokiasuite) or [Nokia PC Suite](http://www.nokia.com/global/support/nokia-pc-suite/) is required).
* running the web app by entering a short URL into the Nokia Xpress Browser on a phone.

##### Series 40 Web App Developer’s Library

The Series 40 Web App Developer’s Library describes the Series 40 web apps development environment for Series 40 phones that run the Xpress Browser for Series 40, the tools for developing Series 40 web apps, and the design considerations for Series 40 web apps.

##### Nokia design principles

These design principles are common for all Nokia platforms and represent Nokia’s idea of good design. Keep these things in mind to make sure your application will be a success.

###### **Less is more.**

* Keep it simple, easy, and intuitive to use. Make it obvious what your application does and how it works.
* The application is lean. There is no extra content. The application works fast and is powerful.
* The structure is flat. No deep hierarchy and endless scrolling.
* The design is elegant and simple and works without exceptions.

###### **Every pixel counts.**

* Remember that mobile devices have limited screen real estate.
* Consider ergonomics and the size of peoples’ fingers when laying out your application and designing controls. The minimum dimensions for touch areas are 8 mm for thumb, 7 mm for finger, and 5 x 4.5 mm for pen.

###### **Natural interaction.**

* The interaction is familiar, clear, and trustworthy.
* Basic interaction should be achieved with touch.
* Be consistent, logical, and coherent both within the application and within your target platform.
* Make sure that you use terms consistently.

###### **Remember the user.**

* What does the user want to do with the application? Keep the big picture in mind when designing the application.
* Mobile devices are used in different situations. Check that the application can be used in those conditions where it should be used, e.g. in a bus, outdoors, in a noisy environment...
* You are creating the application for the end user, not for yourself.
* The application is intuitive and fun to use.
* The application makes the user smile but not laugh.

###### **Use fonts and colours wisely.**

* The fonts you use are clear and easy to read from small screen.
* Check that the contrast is clear enough.
* Use a limited number of colours.
* Remember the colour metaphors and cultural differences in perceiving colours.

###### **Don’t be offensive.**

* You have only one chance to make the first impression. Make the most of it.
* Check that the application is in line with the [Ovi Store content guidelines](https://publish.ovi.com/account/content_guidelines" \t "_blank).

#### Xpress Web App Builder

Xpress Web App Builder is an online tool that guides you through the process of creating rich web apps, with no coding required. Select from a variety of templates, customise your theme, and then add clipped web content, RSS feeds, and social media information. The key features of the tool are:

* layout templates to present content, including single pane, tabbed view, and accordion view, as well as focused templates for news, pictures, and video content.
* a wide range of content widgets for clipped web content; RSS feeds; video from YouTube; pictures from Flickr, Picasa, and other photo sharing sites; and blogs from Tumblr and WordPress.
* the ability to add PSMS and call capabilities, static HERE Maps, and in-app advertising from [Nokia Ad Exchange](http://www.developer.nokia.com/NAX).
* the option to customise your app's colour scheme, including header and font colours.
* static and dynamic previews of your app, for all supported screen resolutions.

When you've completed your web app, the tool provides a short URL for testing the app on your phone, and lets you submit the app to Nokia Publish to start the process of publication in Nokia store. However, if you want to customise your web app further, you can download the source code and import it into Nokia Web Tools.

#### Test your Series 40 web apps

If you don't have access to a Series 40 phone, you can test your web content and apps by making use of the Remote Device Access service. This service provides you with access to ten Series 40 phone models, more than 30 phones, over an internet connection. The service is available free to all Nokia Developer members.

#### Mobile Web Components

Make the most of the latest HTML5 feature in Nokia browsers.

Add rich, HTML5 based components to your web pages and web apps for Symbian Anna phones and the Nokia N9 Smartphone. Included are components for collapsible content blocks, scrollable large content item windows, pop-up menus, expandable sliding menus, slideshows, and others.

#### Leverage the power of QtWebKit

Using Qt WebKit technology, Web developer can easily transform web apps and web services into powerful native applications. Qt offers HTML5 and CSS3 support today. The quick and powerful way to use web assets and skills to produce apps for smartphones and mobile computers.

#### Device APIs::API Bridge

APIBridge is a component for Nokia Symbian devices that enables WRT widgets, Adobe Flash Lite content, and Java applications to access device features through a plug-in architecture. The APIBridge package ships with a set of plug-ins and the components to enable the features of the plug-ins to be used. Developers can extend the APIBridge component with their own plug-ins.

## Cacoo:: online drawing tool

 Cacoo is a diagram creation tool that runs in your web browser.Multiple people can work together on the same diagram in real time.Diagrams can be published directly to websites, wikis, and blogs.

### Creating Diagrams

* Elements can be dragged and drop to easily create diagrams.
* Elements can be linked together with connectors.
* Connectors automatically move when elements are repositioned.
* You can use a text box and put text anywhere you like.
* You can upload images from your PC and include them in Diagrams.
* You can take screenshots of your computer from within Cacoo.
* Smart styles can easily be applied to stencils.
* You can have multiple sheets in a diagram and use them as backgrounds or layers.
* When you move the objects on your canvas, they will be snapped at the objects or grids nearby and align automatically.
* Copying, pasting and other functionality of basic drawing software is also built in to Cacoo.
* All actions are stored so there are unlimited levels of undo.
* You can import an image from the other websites by indicating the URL.
* The imported image can be easily trimmed only using your mouse.
* According to your editing status, tips will be shown on the right bottom corner of the canvas.

### Collaboration

* You can invite collaborators to work with you in Cacoo.
* Multiple people can edit a diagram in real time.
* There is a chat function in the editor so people can communicate while creating diagrams.
* People can leave comments about the diagrams.
* Each user can set their own user icon.
* When editing with multiple people, users icons appear on selected objects.
* Sharing diagrams become much smoother. Diagrams in the shared folders can be accessible and editable by people who you have shared the folder with.

### Sharing Diagrams

* If you keep the diagram private then other users can't see it.
* If you make the diagram URL public, then anyone who knows the URL can see it.
* Publishing a diagram to a blog can be useful in various ways.
* You can place code into blogs to create a slideshow
* Published images always display the most recent version.
* Diagrams can be exported to SVG format (Plus Plan users only) and PNG format. (More formats will be available in the future.)
* Diagrams can be posted to Twitter/Facebook/GoogleBuzz
* Diagrams can be displayed in SVG format for printing. (Plus Plan users only. A few browsers are not supported.)

### Managing Diagrams

* Diagrams can be placed into folders.
* Diagrams can be copied.
* Diagrams can be displayed as thumbnails or as a list.

### Languages and Time Zones

* All pages and notification e-mails support English and Japanese
* Users can enter text from almost all languages.
* Dates are displayed relative to your local time zone.

### Security

* Private diagrams can only be seen by users you select.
* URLs which you do not share can not be found by other users or search engines.
* All editing and management is protected by SSL.
* In order to access information about diagrams a Cacoo ID and password are requited.
* User passwords are encrypted on Cacoo's server.

### API

* You can access Cacoo using the API.
* The Cacoo API supports OAuth and an API Key.

By using the Cacoo API you are able to interact with Cacoo from other services and applications.

Authorization Methods

There are two ways to access the Cacoo API.

#### 1. API Key

The API key allows you make requests to the Cacoo API. You can make an API key here.

#### API Key

Append your API key to requests to the API to return data from your account.(Parameter name "apiKey")

Example: https://cacoo.com/api/v1/diagrams.json?apiKey=abcdefghijklmn

#### 2. OAuth

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

You can get your Access Token from the following links.

#### applications

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

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

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

## http://t1.gstatic.com/images?q=tbn:ANd9GcS-CmbHGLD4MH83JH1oNIr_acREqblVhrcFuvQfYZR8HFi1UpaqlgGitHub

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.