Web Programming Languages

Online Accounting Management System

**Report Phase #1**

Linghuan Hu (UTD-ID: 2021179513)

Linrui Zhang (UTD-ID:2021188217)

Yisheng Zhou (UTD-ID:2021185183)

**Table of Contents**

[**1.** **General Information** 1](#_Toc369284322)

[**1.1.** **Group Information** 1](#_Toc369284323)

[**1.2.** **Meeting Schedule** 1](#_Toc369284324)

[**2.** **Source Control and Repository Policy** 1](#_Toc369284325)

[**2.1.** **Repository tool and server** 1](#_Toc369284326)

[**2.2.** **Repository commit strategy** 1](#_Toc369284327)

[**2.3.** **Repository branch strategy** 1](#_Toc369284328)

[**3.** **Coding Standard** 2](#_Toc369284329)

[**3.1.** **C#** 2](#_Toc369284330)

[**3.1.1.** **Clarity and Consistency** 2](#_Toc369284331)

[**3.1.2.** **Formatting and style** 2](#_Toc369284332)

[**3.1.3.** **Libraries** 2](#_Toc369284333)

[**3.1.4.** **Variable Declarations and Initializations** 2](#_Toc369284334)

[**3.1.5.** **Statements** 2](#_Toc369284335)

[**3.1.6.** **Whitespace** 2](#_Toc369284336)

[**3.1.7.** **Spaces** 3](#_Toc369284337)

[**3.1.8.** **Braces** 3](#_Toc369284338)

[**3.1.9.** **Comments** 3](#_Toc369284339)

[**3.1.10.** **Inline Code Comments** 3](#_Toc369284340)

[**3.1.11.** **Capitalization Naming Rules for Identifiers** 4](#_Toc369284341)

[**3.2.** **HTML** 5](#_Toc369284342)

[**3.2.1.** **Formatting** 5](#_Toc369284343)

[**3.2.2.** **Doctype and layout** 6](#_Toc369284344)

[**3.2.3.** **Forms** 6](#_Toc369284345)

[**3.2.4.** **Including meta data** 6](#_Toc369284346)

[**3.2.5.** **Line breaks** 6](#_Toc369284347)

[**3.3.** **CSS** 6](#_Toc369284348)

[**3.3.1.** **Structure** 6](#_Toc369284349)

[**3.3.2.** **Selectors** 7](#_Toc369284350)

[**3.3.3.** **Properties** 8](#_Toc369284351)

[**3.3.4.** **Property Ordering** 8](#_Toc369284352)

[**3.3.5.** **Vendor Prefixes** 9](#_Toc369284353)

[**3.3.6.** **Values** 10](#_Toc369284354)

[**3.3.7.** **Media Query** 11](#_Toc369284355)

[**3.3.8.** **Commenting** 11](#_Toc369284356)

[**4.** **Requirements** 12](#_Toc369284357)

[**4.1.** **User Stories** 12](#_Toc369284358)

[**4.1.1.** **User (Following features are shared by all kind of user, including guest, normal user, ultimate user).** 12](#_Toc369284359)

[**4.1.2.** **Guest** 12](#_Toc369284360)

[**4.1.3.** **Normal User** 12](#_Toc369284361)

[**4.1.4.** **Ultimate User** 12](#_Toc369284362)

[**4.2.** **Use Case Diagram** 13](#_Toc369284363)

[**4.3.** **User Definition** 13](#_Toc369284364)

[**4.3.1.** **Guest** 13](#_Toc369284365)

[**4.3.2.** **Normal User** 13](#_Toc369284366)

[**4.3.3.** **Ultimate User** 14](#_Toc369284367)

[**4.4.** **Use Case Specification** 14](#_Toc369284368)

[**Use case: Create Plan** 14](#_Toc369284369)

[**Use case: Process Plan** 14](#_Toc369284370)

[**Use case: Delete Plan** 15](#_Toc369284371)

[**Use case: Generate Plan Report** 15](#_Toc369284372)

[**Use case: Create Collaboration Plan** 15](#_Toc369284373)

[**Use case: Search Plan** 16](#_Toc369284374)

[**Use case: Edit Account Profile** 16](#_Toc369284375)

[**Use case: Modify Plan** 16](#_Toc369284376)

[**Use case: Upgrade Account** 17](#_Toc369284377)

[**Use case: Register Account** 17](#_Toc369284378)

[**4.5.** **Specification Functional Requirements** 18](#_Toc369284379)

[**4.5.1.** **Guest Requirements** 18](#_Toc369284380)

[**4.5.2.** **Normal User Requirements** 18](#_Toc369284381)

[**4.5.3.** **Ultimate User Requirements** 19](#_Toc369284382)

[**5.** **Web Page Design** 20](#_Toc369284383)

[**5.1.** **Page Category** 20](#_Toc369284384)

[**5.1.1.** **Brief Introduction** 20](#_Toc369284385)

[**5.1.2.** **Welcome Page** 20](#_Toc369284386)

**[5.1.3.](#_Toc369284387)****[Register Page](#_Toc369284387)** [20](#_Toc369284387)

**[5.1.4.](#_Toc369284388)****[Personal Profile Page](#_Toc369284388)** [20](#_Toc369284388)

[**5.1.5.** **Personal Page** 20](#_Toc369284389)

[**5.1.6.** **Report Page** 20](#_Toc369284390)

[**5.2.** **Page Wire Frame Design** 21](#_Toc369284391)

[**5.2.1.** **Welcome Page** 21](#_Toc369284392)

[**5.2.2.** **Register Page** 21](#_Toc369284393)

[**5.2.3.** **Personal Page** 22](#_Toc369284394)

[**5.2.4.** **Personal Profile Page** 22](#_Toc369284395)

[**5.2.5.** **Report Page** 23](#_Toc369284396)

[**6.** **Database Design** 23](#_Toc369284397)

[**6.1.** **Table Design** 23](#_Toc369284398)

[**6.1.1.** **User\_Login** 23](#_Toc369284399)

[**6.1.2.** **User\_Profile** 23](#_Toc369284400)

[**6.1.3.** **User\_Plan** 24](#_Toc369284401)

[**6.1.4.** **Plan** 24](#_Toc369284402)

[**6.1.5.** **User Collaboration Plan** 24](#_Toc369284403)

[**6.2.** **E-R Model** 25](#_Toc369284404)

# **General Information**

## **Group Information**

* Linghuan Hu Email: hulinghuan@gmail.com Phone: 214-533-6171
* Linrui Zhang Email: 1246615160@qq.com Phone: 214-206-5712
* Yisheng Zhou Email: wuxinguoren@hotmail.com Phone: 469-422-9996

## **Meeting Schedule**

Our group will have two basic meetings every week.

* Meeting 1: Monday 3 pm ~ 4:30 pm
* Meeting 2: Wednesday 3 pm ~ 4:30 pm

# **Source Control and Repository Policy**

## **Repository tool and server**

* Tool Name: Git
* Tool Version: 1.8.4 or later
* Remote Git Server: https://github.com/hulinghuan/Web-Programming-Languages.git

## **Repository commit strategy**

Each commit should contain clear and helpful commit message.

Each commit message should have:

1. commit version, include:
2. main version number
3. function version number
4. specific commit description, include:
5. commit task type: working on/add/delete/modify file/function
6. object name
7. task description: the specific description about what’s the goal or purpose of the task and the abstract description about what have been done in this commit.
8. name of the person who did the commit

## **Repository branch strategy**

* Branch name format:

branch type + task name.

* Branch type:

the type of each branch should be one of following:

1. master
2. function
3. hotfix

* Task naming strategy

The name of tasks should be the one of the subsystem name.

* Branch merge strategy

Master branch can be merged with other branches when and only when the other branch pass the black box test and has been proved it is stable.

# **Coding Standard**

## **C#**

### **Clarity and Consistency**

Do ensure that clarity, readability and transparency are paramount. These coding standards strive to ensure that the resultant code is easy to understand and maintain, but nothing beats fundamentally clear, concise, self-documenting code.

### **Formatting and style**

* Tabs

Do not use tabs. It's generally accepted across Microsoft that tabs shouldn't be used in source files - different text editors use different spacing to render tabs, and this causes formatting confusion. All code should be written using four spaces for indentation. Visual Studio text editor can be configured to insert spaces for tabs.

* Length of line

The length of lines of code should be 86 column.

* Font

Use a fixed-width font, typically Courier New, in your code editor.

### **Libraries**

Do not reference unnecessary libraries, include unnecessary header files, or reference unnecessary assemblies.

### **Variable Declarations and Initializations**

Do declare local variables in the minimum scope block that can contain them, typically justbefore use if the language allows; otherwise, at the top of that scope block.

Initialize variables when they are declared.

Do not declare multiple variables in a single line. One declaration per line is recommended

since it encourages commenting, and could avoid confusion.

### **Statements**

Use an enum to strongly type parameters, properties, and return values that representsets of values.

### **Whitespace**

Use blank lines to separate groups of related statements. Omit extra blanklines that do not make the code easier to read. For example, you can have a blank linebetween variable declarations and code.

### **Spaces**

Spaces improve readability by decreasing code density. Here are some guidelines for the use of space characters within code:

1. CreateFoo(); // No space between function name and parenthesis
2. Method(myChar, 0, 1); // Single space after a comma
3. x = array[index]; // No spaces inside brackets
4. while (x == y) // Single space before flow control statements
5. if (x == y) // Single space separates operators
6. ' VB.NET sample:
7. CreateFoo() ' No space between function name and parenthesis
8. Method(myChar, 0, 1) ' Single space after a comma
9. x = array(index) ' No spaces inside brackets
10. While (x = y) ' Single space before flow control statements
11. If (x = y)Then ' Single space separates operators

### **Braces**

Use the brace rule as following format:

if (x > 5) {

y = 0;

}

### **Comments**

Use comments that sumCmarize what a piece of code is designed to do andwhy.

Do not use comments to repeat the code.

Use ‘//’ comments instead of ‘/\* \*/’ for comments for C++ and C# codecomments.

The single-line syntax (// …) is preferred even when a comment spans multiplelines.

Indent comments at the same level as the code they describe.

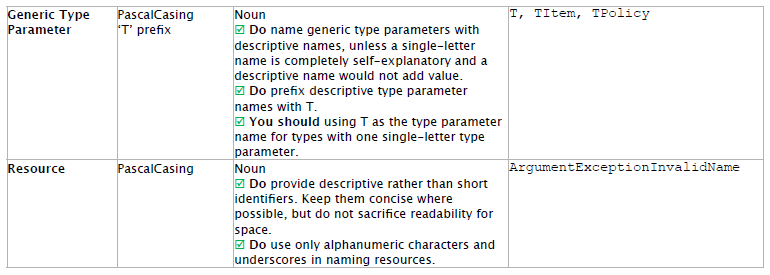
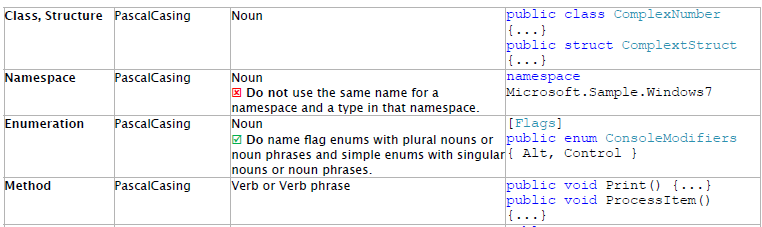
Use full sentences with initial caps, a terminating period and properpunctuation and spelling in comments.

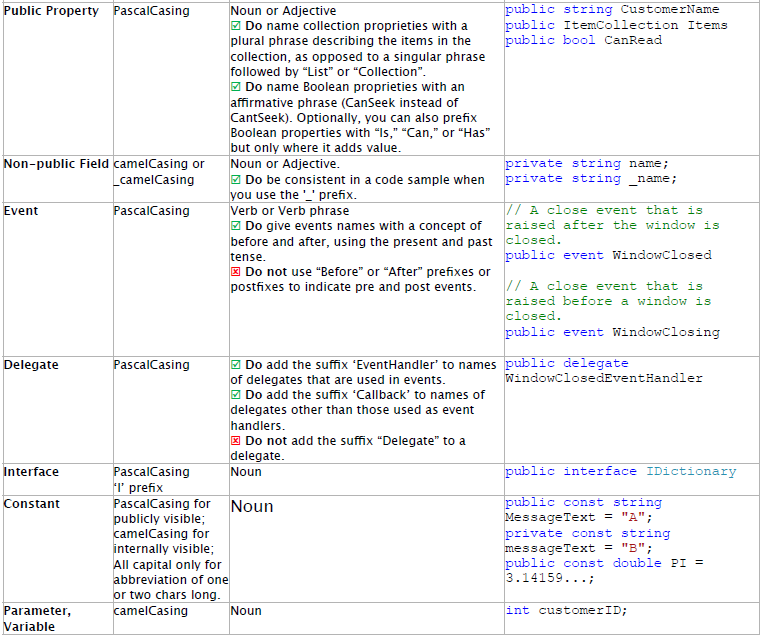
### **Inline Code Comments**

Inline comments should be included on their own line and should be indented at the same level as the code they are commenting on, with a blank line before, but none after.

Comments describing a block of code should appear on a line by themselves, indented as the code they describe, with one blank line before it and one blank line after it.

### **Capitalization Naming Rules for Identifiers**





## **HTML**

### **Formatting**

All HTML documents must use two spaces for indentation and there should be no trailing whitespace. XHTML syntax must be used (this is more a Genshi requirement) and all attributes must use double quotes around attributes.

<!-- XHTML boolean attributes must still have values and self closing tags must have a closing / -->

<video autoplay="autoplay" poster="poster\_image.jpg">

<source src="foo.ogg" type="video/ogg" />

</video>

HTML5 elements should be used where appropriate reserving <div> and <span> elements for situations where there is no semantic value (such as wrapping elements to provide styling hooks).

### **Doctype and layout**

All documents shall use the HTML 4.01 doctype and the <html> element should have the "lang" attribute. The <head> should also at a minimum include "viewport" and "charset" meta tags.

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

### **Forms**

Form fields mCust always include a <label> element with a "for" attribute matching the "id" on the input. This helps accessibility by focusing the input when the label is clicked, it also helps screen readers match labels to their respective inputs.

<label for="field-email">email</label>

<input type="email" id="field-email" name="email" value="" />

Each <input> should have an "id" that is unique to the page. It does not have to match the "name" attribute.

### **Including meta data**

Ideally, classes should only be used as styling hooks. If you need to include additional data in the html document, for example to pass data to JavaScript, then the HTML5 data- attributes should be used.

<a class="btn" data-format="csv">Download CSV</a>

### **Line breaks**

Do not include line breaks within <p> blocks.

Do like this:

<p>Blah foo blah</p>

<p>New paragraph, blah</p>

Do not do like this:

<p>Blah foo blah

New paragraph, blah</p>

## **CSS**

### **Structure**

There are plenty of different methods for structuring a stylesheet. With the CSS in core, it is important to retain a high degree of legibility. This enables subsequent contributors to have a clear understanding of the flow of the document.

Use tabs, not spaces, to indent each property.

Add two blank lines between sections and one blank line between blocks in a section.

Each selector should be on its own line, ending in either a comma or an opening curly brace. Property-value pairs should be on their own line, with one tab of indentation and an ending semicolon. The closing brace should be flush left, using the same level of indentation as the opening selector.

|  |
| --- |
| #selector-1,  #selector-2,  #selector-3{      background: #fff;      color: #000;  } |

Incorrect:

|  |
| --- |
| #selector-1, #selector-2, #selector-3{      background: #fff;      color: #000;      }    #selector-1{ background: #fff; color: #000; } |

### **Selectors**

With specificity, comes great responsibility. Broad selectors allow us to be efficient, yet can have adverse consequences if not tested. Location-specific selectors can save us time, but will quickly lead to a cluttered stylesheet. Exercise your best judgment to create selectors that find the right balance between contributing to the overall style and layout of the DOM.

Similar to the [WordPress Coding Standards](http://codex.wordpress.org/WordPress_Coding_Standards) for file names, use lowercase and separate words with hyphens when naming selectors. Avoid camel case and underscores.

Use human readable selectors that describe what element(s) they style.

Attribute selectors should use double quotes around values

Refrain from using over-qualified selectors, div.container can simply be stated as .container

Correct:

|  |
| --- |
| #comment-form {      margin: 1em0;  }  input[type="text"] {      line-height: 1.1;  } |

Incorrect:

|  |
| --- |
| #commentForm{ /\* Avoid camelcase. \*/      margin: 0;  }  #comment\_form{ /\* Avoid underscores. \*/      margin: 0;  }    div#comment\_form { /\* Avoid over-qualification. \*/      margin: 0;  }  #c1-xr { /\* What is a c1-xr?! Use a better name. \*/      margin: 0;  }  input[type=text] { /\* Should be [type="text"] \*/      line-height: 110%/\* Also doubly incorrect \*/  } |

### **Properties**

Similar to selectors, properties that are too specific will hinder the flexibility of the design. Less is more. Make sure you are not repeating styling or introducing fixed dimensions (when a fluid solution is more acceptable).

* Properties should be followed by a colon and a space.
* All properties and values should be lowercase, except for font names and vendor-specific properties.
* Use hex code for colors, or rgba() if opacity is needed. Avoid RGB format and uppercase, and shorten values when possible: #fff instead of #FFFFFF.
* Use shorthand (except when overriding styles) for background, border, font, list-style, margin, and padding values as much as possible. (For a shorthand reference, see [CSS Shorthand](http://codex.wordpress.org/CSS_Shorthand).)

### **Property Ordering**

Above all else, choose something that is meaningful to you and semantic in some way. Random ordering is chaos, not poetry. In WordPress Core, our choice is logical or grouped ordering, wherein properties are grouped by meaning and ordered specifically within those groups. The properties within groups are also strategically ordered to create transitions between sections, such as background directly before color. The baseline for ordering is:

* Display
* Positioning
* Box model
* Colors and Typography
* Other

Things that are not yet used in core itself, such as CSS3 animations, may not have a prescribed place above but likely would fit into one of the above in a logical manner. Just as CSS is evolving, so our standards will evolve with it.

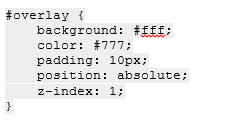
Top/Right/Bottom/Left (TRBL/trouble) should be the order for any relevant properties (e.g. margin), much as the order goes in values. Corner specifiers (e.g. border-radius-\*-\*) should be top-left, top-right, bottom-right, bottom-left. This is derived from how shorthand values would be ordered.

Example:

|  |
| --- |
| #overlay {      position: absolute;      z-index: 1;      padding: 10px;      background: #fff;      color: #777;  } |

Another method that is often used, including by the Automattic/WordPress.com Themes Team, is to order properties alphabetically, with or without certain exceptions.

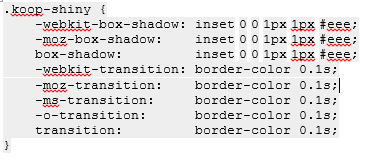
Example:



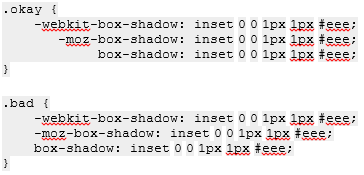
### **Vendor Prefixes**

Vendor prefixes should go longest (-webkit-) to shortest (unprefixed). Values should be left aligned with spaces after the colon provided that all the values are the same across all prefixes.

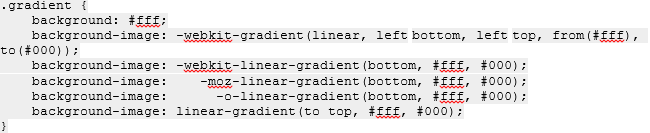
Preferred method:



Not preferred:



[Special case](http://core.trac.wordpress.org/ticket/16461#comment:43) for CSS gradients:

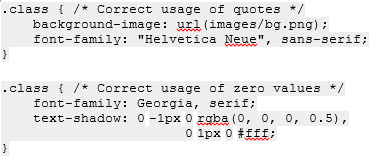


### **Values**

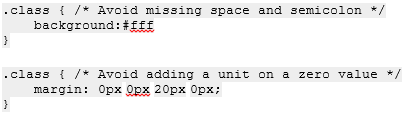
There are numerous ways to input values for properties. Follow the guidelines below to help us retain a high degree of consistency.

* Space before the value, after the colon
* Do not pad parentheses with spaces
* Always end in a semicolon
* Use double quotes rather than single quotes, and only when needed, such as when a font name has a space.
* 0 values should not have units unless necessary, such as with transition-duration.
* Line height should also be unit-less, unless necessary to be defined as a specific pixel value. This is more than just a style convention, but is worth mentioning here. More information: <http://meyerweb.com/eric/thoughts/2006/02/08/unitless-line-heights/>
* Use a leading zero for decimal values, including in rgba().
* Multiple comma-separated values for one property should be separated by either a space or a newline, including within rgba(). Newlines should be used for lengthier multi-part values such as those for shorthand properties like box-shadow and text-shadow. Each subsequent value after the first should then be on a new line, indented to the same level as the selector and then spaced over to left-align with the previous value.

Correct:



Incorrect:



### **Media Query**

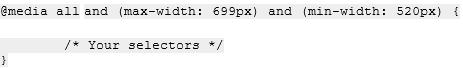
Media queries allow us to gracefully degrade the DOM for different screen sizes. If you are adding any, be sure to test above and below the break-point you are targeting.

It is generally advisable to keep media queries grouped by media at the bottom of the stylesheet.

* An exception is made for the wp-admin.css file in core, as it is very large and each section essentially represents a stylesheet of its own. Media queries are therefore added at the bottom of sections as applicable.

Rule sets for media queries should be indented one level in.

Example:



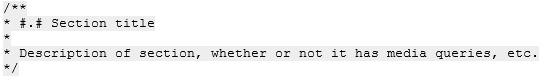
### **Commenting**

Comment, and comment liberally. If there are concerns about file size, utilize minified files and the SCRIPT\_DEBUG constant. Long comments should manually break the line length at 80 characters.

A table of contents should be utilized for longer stylesheets, especially those that are highly sectioned. Using an index number (1.0, 1.1, 2.0, etc.) aids in searching and jumping to a location.

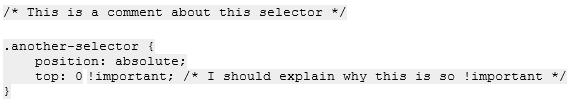
Comments should be formatted much as PHPDoc is. The [CSSDoc](http://cssdoc.net/) standard is not necessarily widely accepted or used but some aspects of it may be adopted over time. Section/subsection headers should have newlines before and after. Inline comments should not have empty newlines separating the comment from the item to which it relates.

For sections and subsections:





For inline:



# **Requirements**

## **User Stories**

### **User (Following features are shared by all kind of user, including guest, normal user, ultimate user).**

* + - 1. As a user, I want to create accounting plans.
      2. As a user, I want to add purchased items to the temporary accounting plan.
      3. As a user, I want to edit the purchased items in the temporary accounting plan.
      4. As a user, I want to remove the purchased items in the temporary accounting plan.
      5. As a user, I want to generate full accounting report of each accounting plan.
      6. As a user, I want to view my every accounting plan.
      7. As a user, I want to get the purchase amount, category amount in a time period from full accounting report.

### **Guest**

* + - 1. As a guest, I want the accounting plan will be deleted when I leave the website.
      2. As a guest, I want to register my own account.
      3. As a guest, I want to save my temporary account plan if I register the account.

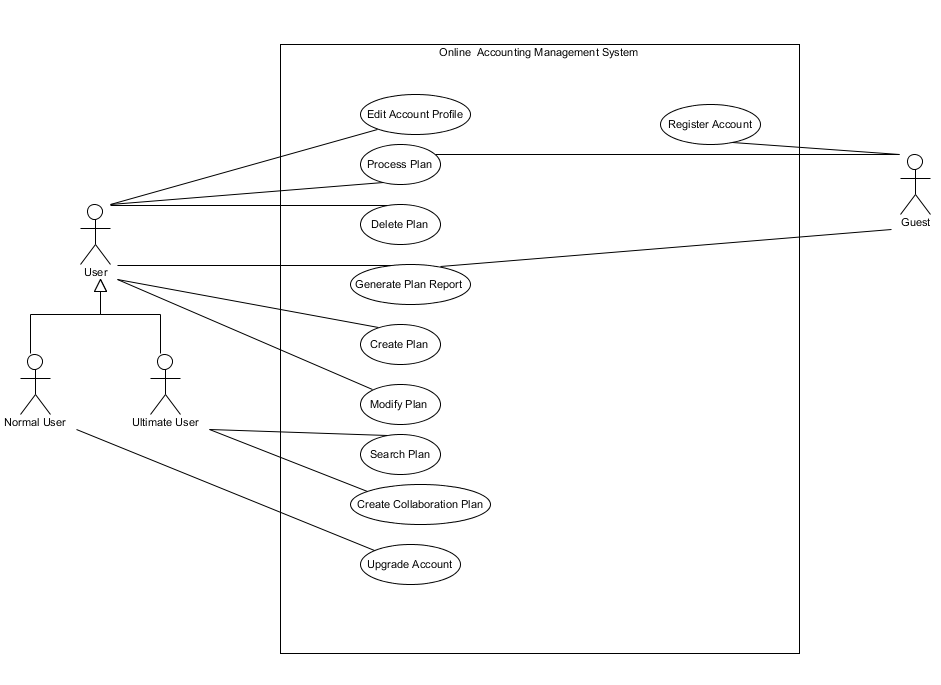
### **Normal User**

* + - 1. As a normal user, I want be able to login my account.
      2. As a normal user, I want be able to log out my account.
      3. As a normal user, I want to set my personal profile.
      4. As a normal user, I want to have the authority to change the profile of my accounting plan.
      5. As a normal user, I want to set the plan shared setting so that I can allow designated users to browse my accounting plan.
      6. As a normal user, I want to browse the designated accounting plans which are shared to me by others.
      7. As a normal user, I want all my accounting plans are encrypted so that it is safe.

### **Ultimate User**

* + - 1. As an ultimate user, I want be able to login my account.
      2. As an ultimate user, I want be able to log out my account.
      3. As an ultimate user, I want to set my profile.
      4. As an ultimate user, I want to set the plan shared setting so that I can allow designated users to browse my accounting plan.
      5. As an ultimate user, I want to browse the designated accounting plans which are shared to me by other users.
      6. As an ultimate user, I want to set the collaboration setting so that I can allow designated normal and ultimate user to edit my accounting plan.

## **Use Case Diagram**



## **User Definition**

*(This is the high-level and general description of 3 different kind of user. It only descript the main feature and may not give enough specific detail)*

### **Guest**

* + - The guest user can create one and only one individual accounting plan.
    - The guest will lose his individual accounting plan when guest leave the web site.
    - The guest can generate the accounting report of the only one individual accounting plan.

### **Normal User**

* + - The normal user can create no more than 5 individual accounting plan.
    - The normal user can generate individual accounting report of every individual plan.
    - All individual plan will be stored on server after encrypted forever.

### **Ultimate User**

* + - The ultimate user can create unlimited amount of individual plan. The ultimate user can create unlimited amount of collaboration plan.
    - The ultimate user can collaborate with other designated normal user or ultimate user and edit the collaboration plan together.
    - The ultimate user can generate individual accounting report of every individual and collaboration plan.
    - All individual plan and collaboration plan will be stored on server after encrypted forever.

## **Use Case Specification**

|  |
| --- |
| **Use case: Create Plan** |
| **ID:** UC1 |
| **Actors:**  User, Guest |
| **Preconditions:**  User is in personal page. |
| **Flow of events:**   1. The user click “Create Plan” button on the website. 2. The users insert the plan name in a text field. 3. The user click “Confirm” 4. The system validates the creation request of the plan. 5. If the creation request pass the system validation then    1. The system creates the plan for the user 6. If the creation request does not pass the system validation then    1. The system will inform the user of the creation failed by showing error message. |
| **Post conditions:**   1. The system will show the plan which is created just before immediately |

|  |
| --- |
| **Use case: Process Plan** |
| **ID:** UC2 |
| **Actors:**  User, Guest |
| **Preconditions:**  User is in personal page. |
| **Flow of events:**   1. The user click “plan name”. 2. The system validate the process plan request. 3. If process plan request validation success then    1. The user enter the editing condition.    2. While the user do not click submit button       1. The user is editing the plan.    3. The user click submit button 4. If the process plan request validation failed then    1. The system will inform the user of the process plan failed by showing error message. |
| **Postconditions:**   1. The system show the personal page. |

|  |
| --- |
| **Use case: Delete Plan** |
| **ID:** UC3 |
| **Actors:**  User |
| **Preconditions:**  User has login to the system and is in the personal page. |
| **Flow of events:**   1. User click the “delete” button of a plan. 2. The system validates the delete plan request. 3. If the delete plan request success then    1. The system warnings user by showing caution message and provides “Delete” and “Cancel” button.    2. If user click “Delete” button then       1. The system delete the plan.    3. If user click “Cancel” button.       1. The system cancel the delete request. 4. If the delete plan request failed then 5. The system will inform of the delete plan failed by showing error message. |
| **Postconditions:**   1. The system show the personal page. |

|  |
| --- |
| **Use case: Generate Plan Report** |
| **ID:** UC4 |
| **Actors:**  User, Guest |
| **Preconditions:**  User or Guest already had a plan. |
| **Flow of events:**   1. The user or guest clicks “Generate Report” in personal page or plan process page. 2. System generates plan report. 3. System shows the plan report to users or guest. |
| **Postconditions:**   1. The system stays on the report page and waits for instruction. |

|  |
| --- |
| **Use case: Create Collaboration Plan** |
| **ID:** UC5 |
| **Actors:**  Ultimate User |
| **Preconditions:**  User has login to the system and is in the personal page. |
| **Flow of events:**   1. The ultimate user click “Edit” button of a plan. 2. The system shows the plan profile. 3. The ultimate user click the checkbox of “Change this plan to Collaboration Plan” 4. The system shows Collaboration setting section to ultimate user. 5. The ultimate User input others user’s username or alias. 6. The system searches the username in database. 7. If designed user can not be found in database then    1. The system will inform ultimate user that the user can not be found. 8. If designed user is found in database then    1. The ultimate user clicks designed user name to add it to the collaboration plan.    2. The ultimate user clicks “invite” button.    3. The system sends invitation to the users who are invited to the collaboration plan. |
| **Postconditions:**   1. The system stays on the personal pages. |

|  |
| --- |
| **Use case: Search Plan** |
| **ID:** UC6 |
| **Actors:**  Ultimate User |
| **Preconditions:**  User has login to the system and is in the personal page. |
| **Flow of events:**   1. The ultimate user click “Search” button. 2. The ultimate user input plan name. 3. The system shows the search result. |
| **Postconditions:**   1. The system stays on the personal pages. |

|  |
| --- |
| **Use case: Edit Account Profile** |
| **ID:** UC7 |
| **Actors:**  User |
| **Preconditions:**  User has login to the system and is in the personal page. |
| **Flow of events:**   1. The user clicks “Edit Profile” button. 2. System shows profile page to users. 3. The user changes the content of profile page. 4. The user clicks “Save” button to save the changes. |
| **Postconditions:**   1. The system stays on the personal pages. |

|  |
| --- |
| **Use case: Modify Plan** |
| **ID:** UC8 |
| **Actors:**  User |
| **Preconditions:**  User has login to the system and is in the personal page. |
| **Flow of events:**   1. The user click “Edit” button of a plan. 2. The system shows the plan profile. 3. The user change the plan profile. 4. The user click “Save” button to save the changes. |
| **Postconditions:**   1. The system stays on the personal pages. |

|  |
| --- |
| **Use case: Upgrade Account** |
| **ID:** UC9 |
| **Actors:**  Normal User |
| **Preconditions:**  User has login to the system and is in the personal page. |
| **Flow of events:**   1. The normal user click “Upgrade Account” 2. The normal user input the upgrade key. 3. The normal user click “Upgrade” button. 4. The system validate the account upgrade request. 5. If account upgrade request validation success then    1. The system upgrade the normal user to ultimate user. 6. If account upgrade request validation failed then    1. The system informs user of account upgrade request validation fail. |
| **Postconditions:**   1. The system stays on the personal pages. |

|  |
| --- |
| **Use case: Register Account** |
| **ID:** UC10 |
| **Actors:**  Guest |
| **Preconditions:**   1. The guest is in the welcome page. |
| **Flow of events:**   1. The guest clicks “Sign up” button on the Welcome Page. 2. The guest inputs the login email address. 3. The guest inputs the login password. 4. The system validates the password and email. 5. While the password and email validation failed    1. The system informs guest the password and email validation failed.    2. The guest inputs the login email address.    3. The guest inputs the login password. 6. The system informs guest registration success. 7. The system show the personal profile page. |
| **Postconditions:**   1. The system stays on the personal pages. |

## **Specification Functional Requirements**

### **Guest Requirements**

#### Register Account

* + - The guest shall be able to register normal user account by inputting the valid login email address and password.
    - The guest shall get the register failed error message when the guest user input the invalid login email address or password.
    - The guest shall get the register success message when the guest user input he valid long email address and password.

#### Create Plan

* + - The guest shall create one and only one individual accounting plan.

#### Process Plan

* + - The guest shall be able to process the individual accounting plan.
    - The guest shall add, modify and delete rows in the individual accounting plan.
    - Each row in the individual accounting plan shall contain “Name”, “Category”, “Price” and “Date”.

#### Delete Plan

* + - The guest shall not be able to delete the individual accounting plan.

#### Generate Plan Report

* + - The guest shall be able to generate the individual accounting plan report.
    - The individual accounting plan report shall contain purchase amount, category amount in a time period.

#### Modify Plan

* + - The guest shall not be able to modify the plan.

### **Normal User Requirements**

#### Create Plan

* + - The normal user shall be able to create no more than 5 individual accounting plan.
    - The normal user shall not be able to create an individual accounting plan which has the same name with another’s.

#### Process Plan

* + - The normal user shall be able to process the individual accounting plan.
    - The normal user shall add, modify and delete rows in the individual accounting plan.
    - Each row in the individual accounting plan shall contain “Name”, “Category”, “Price” and “Date”.

#### Delete Plan

* + - The normal user shall be able to delete the individual accounting plan.

#### Modify Plan

* + - The normal user shall be able to modify the plan profile.

#### Generate Plan Report

* + - The normal user shall be able to generate the individual accounting plan report.
    - The individual accounting plan report shall contain purchase amount, category amount between a time period.

#### Edit Account Profile

* + - The normal user shall be able to edit his personal account profile.

#### Upgrade Account

* + - The normal user shall upgrade account by entering purchased upgrade key.

### **Ultimate User Requirements**

#### Create Plan

* + - The ultimate user shall be able to create no limited amount individual accounting plan.
    - The ultimate user shall not be able to create an individual accounting plan which has the same name with his other plan.

#### Process Plan

* + - The ultimate user shall be able to process the individual accounting plan and collaboration plan.
    - The ultimate user shall add, modify and delete rows in the individual accounting plan and collaboration plan.
    - Each row in the individual accounting plan shall contain “Name”, “Category”, “Price” and “Date”.
    - Each row in collaboration accounting plan shall contain “Name”, “Category”, “Price”, “Date” and “Purchased By”.

#### Delete Plan

* + - The ultimate user shall be able to delete the individual and collaboration accounting plan.

#### Modify Plan

* + - The ultimate user shall be able to modify the plan profile.
    - The ultimate user shall be able to specify the collaboration user to join or leave designated collaboration plan.

#### Generate plan Report

* + - The ultimate user shall be able to generate the plan report.
    - The individual accounting plan report shall contain purchase amount, category amount between a time period.
    - The collaboration account plan report shall contain purchase amount, category amount and amount of every collaborators between a time period.

#### Search Plan

* + - The ultimate user shall be able to search specific plan by plan name.

#### Edit Account Profile

* + - The ultimate user shall be able to edit his persona account profile.

# **Web Page Design**

## **Page Category**

### **Brief Introduction**

There are five pages in the website. Welcome page, register page, personal page, personal page, and report page. By clicking the buttons on each page, users can navigate from one page to the other. Each pages have its own functions, 5.1.2-5.1.6 shows the detail of each pages.

### **Welcome Page**

There are four sections in this page. In the top, is the name and logo of the software.

In the left side of the page, people can sign in by using their email and password, and for new users they can sign up by clicking sign up button. People can also find us on Facebook, Twitter, YouTube and google+.

In the middle of the page, there is a picture to tell people the feature of the software and a video to show a demo to tell user how to use this software.

In the bottom of the page, are the new feature that are added in this software.

### **Register Page**

By clicking on the sign up page, users can register in the web. For new users they can put email, username and password and choose the gender, and click the sign up button to sign up.

### **Personal Profile Page**

After a new user register, they can set up their profile in the profile page. They can choose each categories in the left column，and set it up in the right column.

### **Personal Page**

Personal page is still divided into two column, in the left column, user can check their picture, and set their plan by clicking each button under the image.

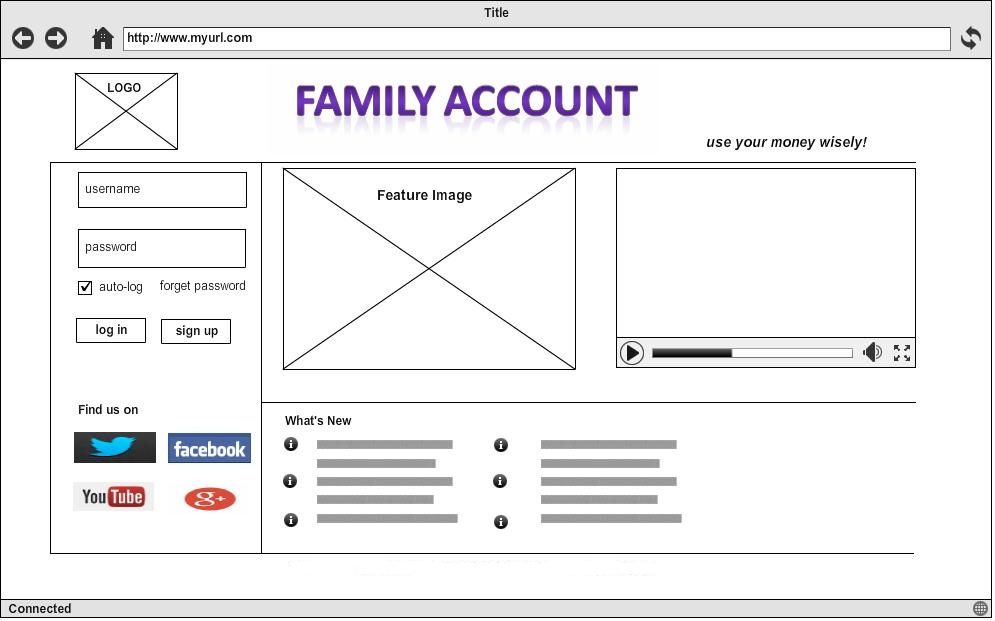
In the right column, is the whole plan that the current user has already set. Users can change the details of each plan by clicking the button below them.

### **Report Page**

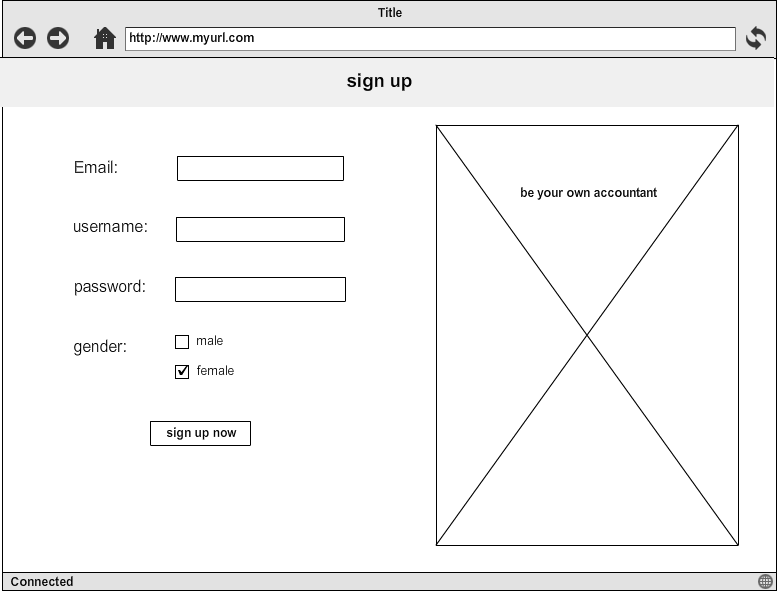
By clicking the “generate report” button on personal page, users can check their weekly report. The individual plan is shown in the left side by histogram，and the collaborate plan is shown on the right side by pie mapping.

## **Page Wire Frame Design**

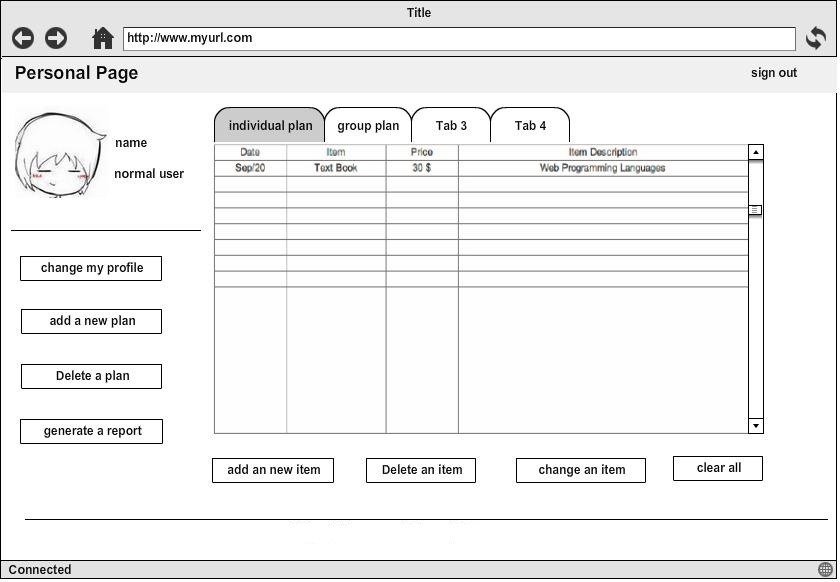
### **Welcome Page**



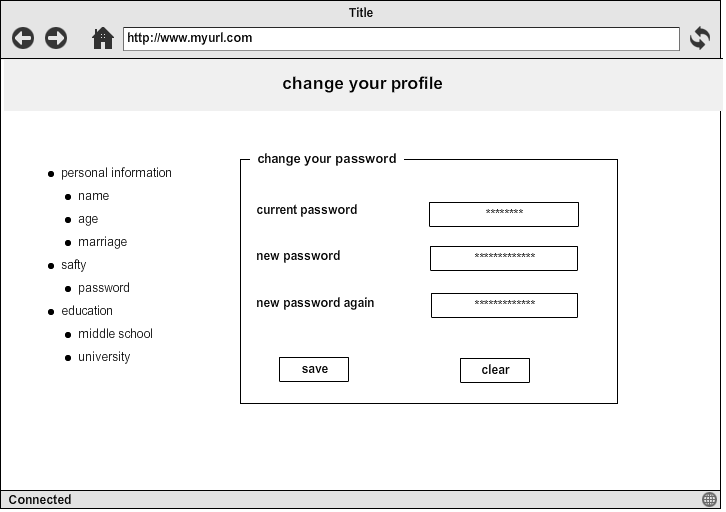
### **Register Page**



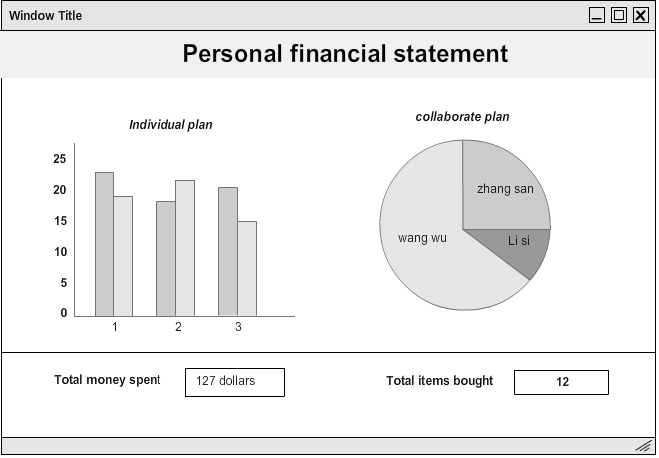
### **Personal Page**



### **Personal Profile Page**



### **Report Page**



# **Database Design**

## **Table Design**

### **User\_Login**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | user\_id | user\_name | email\_addr | user\_pwd |
| **Data Type** | int | varchar(25) | varchar(50) | varchar(25) |
| **Null** | not null | N/A | not null | not null |
| **Primary Key** | yes | N/A | N/A | N/A |
| **Foreign Key** | N/A | N/A | N/A | N/A |
| **Default** | N/A | N/A | N/A | N/A |
| **Identity** | yes | N/A | N/A | N/A |
| **Identity Seed** | 1 |  |  |  |
| **Identity Increment** | 1 |  |  |  |

### **User\_Profile**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | user\_id | user\_name | email\_addr | user\_age | phone | Profile\_image | user\_classification |
| **Data Type** | int | varchar(25) | varchar(50) | int | varchar(20) | Image | int |
| **Null** | not null | N/A | not null | N/A | N/A | N/A | not null |
| **Primary Key** | yes | N/A | N/A | N/A | N/A | N/A | N/A |
| **Foreign Key** | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| **Default** | N/A | N/A | N/A | N/A | N/A | N/A | 0 |
| **Identity** | yes | N/A | N/A | N/A | N/A | N/A | N/A |
| **Identity Seed** | 1 |  |  |  |  |  |  |
| **Identity Increment** | 1 |  |  |  |  |  |  |

### **User\_Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | map\_id | user\_id | user\_name | plan\_id |
| **Data Type** | int | int | varchar(25) | int |
| **Null** | not null | not null | N/A | not null |
| **Primary Key** | yes | N/A | N/A | N/A |
| **Foreign Key** | N/A | N/A | N/A | yes |
| **Default** | N/A | N/A | N/A | N/A |
| **Identity** | yes | N/A | N/A | N/A |
| **Identity Seed** | 1 | 1 | N/A |  |
| **Identity Increment** | 1 | 1 | N/A |  |

### **Plan**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | plan\_id | is\_co | item\_name | price | date | catalog | description | buyer\_name | buyer\_id |
| **Data Type** | int | int | varchar(50) | float | datetime | varchar(25) | varchar(100) | varchar(25) |  |
| **Null** | not null | not null | N/A | N/A | N/A | N/A | N/A | N/A | not null |
| **Primary Key** | yes | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| **Foreign Key** | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| **Default** | N/A | 0 | N/A | 0 | N/A | N/A | N/A | N/A | N/A |
| **Identity** | yes | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| **Identity Seed** | 1 |  |  |  |  |  |  | N/A |  |
| **Identity Increment** | 1 |  |  |  |  |  |  | N/A |  |

### **User Collaboration Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | map\_id | plan\_id | owner\_id | co\_user\_id | authority |
| **Data Type** | int | int | int | int | int |
| **Null** | not null | not null | not null | not null | not null |
| **Primary Key** | yes | N/A | N/A | N/A | N/A |
| **Foreign Key** | N/A | N/A | N/A | N/A | N/A |
| **Default** | N/A | N/A | N/A | N/A | 0 |
| **Identity** | yes | N/A | N/A | N/A | N/A |
| **Identity Seed** | 1 |  |  |  |  |
| **Identity Increment** | 1 |  |  |  |  |

## **E-R Model**