EASYLIFE - ONLINE COMMUNITY TRADING SYSTEM: REQUIREMENTS ANALYSIS,PROTOTYPE DESIGN,DATABASE DESIGN AND SEARCH ENGINE IMPLEMENTATION

by

Weiwei Chen

a Report submitted in partial fulfillment
of the requirements for the SFU-ZU dual degree of
Bachelor of Science
in the School of Computing Science
Simon Fraser University
and
the College of Computer Science and Technology
Zhejiang University

APPROVAL

Name:	Weiwei Chen
Degree:	Bachelor of Science
Title of Report:	EasyLife - Online Community Trading System: Requirements Analysis,Prototype Design,Database Design and Search Engine Implementation

Examining Committee:

Dr. Qianping Gu, Supervisor

Dr. Ramesh Krishnamurti, SFU Examiner

Date Approved:

Abstract

Nowadays, there are many trading service web sites, more and more people choose to do their transactions online. However, on the current web sites, sellers cannot customize their web pages to make them more attractive. In order to solve this problem, in this document, we describe the development process of a new web site named EasyLife.

After the requirement analysis, we achieved the basic functions of a business web site, such as the page design, database building and search engine. In additional to these, we implemented a smart page design tool inside our web site. By using this tool, users can create web pages by using different templates and widgets.

As a result, EasyLife can become a successful business web site and get more attention from host of users, especially users who want to build small online shops of their own.

Acknowledgments

I am writing these acknowledgments to express my gratitude to all the people who gave me help and support. Without them, I could not have finished my project and the technical report.

I would like to first thank my supervisor, Qianping Gu, Professor of SFU Computing Science, who gave me a lot of guidance and suggestions about the project design and the document production.

My partner, Kefu Zhao, is a responsible and creative talent. We worked together to make our idea become a real product. He could always give me great advice when I was trapped by some problem.

Edward Chin, one of my best friends, dedicated his own time to help me edit my report, and make it as perfect as he can. I want to express my deepest gratitude to him and praise his outstanding writing skill.

SFU Computing Science MSc student, Yan Tan, delivered his experience in data retrieval and graphic interface design when I was implementing the search engine.

At last, I will give my thankful to all the following people: Jieying Li, Beijing Mu, Vito Pun and Darren Zhao. They tried their best to support me; without them, I would not have the strength to finish my project.

Contents

List of Tables

List of Figures

List of Programs

Preface

This document, combined with Kefu Zhao's capstone project report[1], aims to describe the procedure of a website development for people who have a certain level of computing science background, especially for those who are interested in designing and implementing their own website.

Unlike some computing science textbooks which only introduce the knowledge with many small examples, we are trying to teach readers the technology through describing the procedure of a website development during our document, so that they can understand how to produce a website from requirement analysis, design, to implementation. After that, they will know the basic process of software engineering.

In order to deliver the necessary knowledge to readers, before describing each developing step of the website, we will the technology first.

This document includes six main parts:

1. Part I Introduction(Chapter 1) and preliminaries(Chapter 2).
2. Part II Requirement analysis (Chapters 3 and 4). This part outlines the process of the software requirement analysis, especially how to use the UML model to define different user classes and design use cases.
3. Part III Prototype design (Chapters 5 and 6). This part outlines the prototype design of different pages in EasyLife by using Adobe Fireworks.
4. Part IV Database design (Chapters 7 and 8). This part outlines the database design and implementation of EasyLife in MySQL.

[1]Please refer to Kefu Zhao's capstone project report for more details[17]

5. Part V Search engine implementation (Chapters 9 and 10). This part outlines the background for data searching, the algorithm in Apache Lucene, and the implementation process of a search engine by using Lucene.

6. Part VI Conclusion of the whole report.

Also, all the source codes used in the search engine implementation are listed in the appendix for users who want to create a mini search engine themselves.

As mentioned before, this document only describes part of the developing process of EasyLife(Requirement Analysis, prototype design, database design and search engine implementation). Readers who want to get the whole picture and to produce a website themselves, are advised to also refer to KefuZhao's capstone project report[17].

In addition, the technology introduction chapters only have brief descriptions. People who want to learn more about the technology and apply it into actual operation, should read the detailed manuals listed in the bibliography.

Part I

Overview

Chapter 1

Introduction

1.1 Background

On 25 December 1990, Sir Tim Berner-Lee, a computer scientist at MIT, realized the first successful communication between a HTTP client and a server via the internet[7]. During the past 20 years, our lives have been transformed due to the development of Information Technology (IT).

In 1995, Craig Newmark, a graduate of Case Western Reserve University, began an email distribution service at his residential place in the San Francisco Bay Area[1]. The service became a web-based service in 1996, and was incorporated as a private for-profit company in 1999[1]. After 15 years of development, the company expanded into approximately 700 cities in 70 different countries[1]. His website, also known as "Craigslist", has become one of the most popular centralized network of online communities, featuring free online classified advertisements .

Online trades and advertisements are now an indispensable part of everyday life. In these online trades and advertisements, people can search products or services according to interests. They can also contact the suppliers via the contact information on the web pages, or complete the whole transaction process online.

1.2 Analysis of the existing online trade websites

There are several kinds of online trade websites today. They all have their own advantages and disadvantages:

- Online auction websites

  A typical example of an online auction website is eBay. The majority of the sales in an online-auction website are in a set-time auction format. The seller can post the product or service on the website by some specified bidding method: for example, eBay uses proxy bidding. The winning bidder will get the product or service.

  It supplies individual users a platform on which to sell their products and services, and maximizes the benefit for both the seller and buyer. However, the integrity of the trade is a serious problem. The website is not responsible for the quality of the products, and legality issues during the transaction and payment process. For example, the product the buyer gets maybe different from what was described on the website; however, the website is not responsible for this. Another disadvantage is that all the products listing pages on the website look identical. Sellers cannot customize their own page or open an online shop inside.

- Online retailers

  A typical example of an online retailer is Amazon.com. As the name suggests, an online retailer can be viewed as a virtual store. On these websites, only the owners have the right to post and sell products. The customers can also choose to buy the products they like and pay a price set by the website. Online retailers avoid integrity problems on the online-auction websites, but the rights of individual sellers are compromised.

- Online classified advertising websites

  A typical example of an online classified advertising website is Craigslist. An online classified advertising website is mostly an online website for individuals to sell their personal belongings (i.e., used goods, house renting and other services). The advertising page is usually text-only and may consist of a short description of the product with the necessary contact information.

  Individuals can post their advertisements online. To some extent, online classified advertising websites also avoid the integrity problems since the website itself does not supply the order and payment service. However, the classified advertisements are normally oversimplified; it cannot attract commercial corporations who want to make many online transactions.

1.3 Introduction of EasyLife

From the comparisons in the last section, we find that each of the existing online trading websites has its own advantages and disadvantages. One of the common disadvantages is that the sellers cannot design and customize their webpages. To overcome this problem, the idea of creating a new site called EasyLife is proposed.

1.3.1 Overview of EasyLife

As we mentioned, EasyLife is an online trading website. It aims at producing a simple webpage design tool - EasyWeb. This tool allows users to design their own pages by dragging the templates or widgets into the canvas from the tool bar. After finishing the design, users can obtain a URL and publish the webpage inside EasyLife.

Essentially, EasyLife is an online trade information centre which includes its own web page design tool. By using EasyLife, all individual sellers or companies can create and post their sub webpages or sub websites inside EasyLife. The buyers can also search the pages they need in EasyLife to obtain information, or contact the seller to complete the transactions.

1.3.2 Target users

The target users of EasyLife include:

- People who want to retrieve information.
- People who don't have extensive web development experience but want to sell stuff online.
- Companies who want to build simple websites but do not want to turn to IT consulting companies.

Chapter 2

Preliminaries

2.1 Technologies

2.1.1 Technologies used in requirement analysis

During the software requirements analysis phase, we will use Unified Modeling Language (UML)[9] as the main tool to specify user definitions.

UML is a standardized modeling language which is popularly used in software engineering. Details about UML will be outlined in Chapter 3. In addition, Windows Office Visio 2007 will be used to draw UML diagrams.

2.1.2 Technologies used in interface prototypes design

During the interface prototypes design phase, we will use Adobe Fireworks CS4 (FW CS4)[5] as the main tool to design the prototype of all pages in EasyLife.

Adobe Fireworks (FW), is a bitmap and vector graphic editor which is popularly used to quickly design the prototype of an interface or even the whole website. Details about FW will be outlined in Chapter 5.

2.1.3 Technologies used in database design

During the database design phase, we will use MySQL[13] as the main tool to design the data tables and to specify their relationships.

MySQL is a relational database management system which is popularly used to design a database and to implement data mining. Details about MySQL will be outlined in Chapter

7.

2.1.4 Technologies used in search engine implementation

During the Search Engine Implementation phase, we will use Apache Lucene[6] as the source library to build the search engine in EasyLife.

Apache Lucene is an open source information retrieval library which is built in Java and used to implement the search engine for websites. Details about Apache Lucene will be outlined in Chapter 9.

2.2 Requirements of the developing environment

The requirement of the developing environment of EasyLife includes:

2.2.1 Hardware requirements

- Processor: 600 MHz Pentium III-class processor or higher
- RAM: 1 Gigabytes or larger
- Disk space: At least 10 GB free space for storage and software installation.
- Display: Super VGA (1024x768) or higher resolution display with 256 colors
- Mouse: Microsoft mouse or compatible pointing device

2.2.2 Software requirements

- Operating system: Windows 2000 or higher
- Application software:
  - Microsoft Office 2003 or higher (Word, Excel, Visio)
  - Adobe Suite CS3 or higher
  - MySQL 1.4 or higher
  - Apache Lucene 2.3 or higher
  - MyEclipse 7.5 or higher

2.3 Knowledge preparation

The readers should have the basic knowledge of the following technologies before reading this document:

- Software engineering: readers should already know what software engineering is, and some basic concepts of software engineering.
- Interactive arts technology: readers should have some basic interface design experience and knowledge.
- SQL and database: readers should have some basic knowledge of SQL and some database knowledge.
- Java: readers should have some experience of Java programming.
- HTML, JSP and CSS: Users should have some experience of graphic interface design with HTML, JSP and CSS.

Part II

Requirements Analysis

Chapter 3

UML

3.1 Introduction of UML

Unified Modeling Language (UML), is a standardized modeling language in software engineering[9]. It can be used during the process of object-oriented software engineering, and gives a standard and straightforward way of describing the software architecture, user interactions, the running process and the developing cycle.

UML 2.2,the latest version of UML, has 14 types of diagrams in two categories: Structure Diagrams and Behavior Diagrams.

Structure diagrams are used to describe the structure information of systems and tell readers what components must be in the system and the relationships between them. There are seven types of structure diagrams in UML: Class diagram, Component diagram, Composite structure diagram, Deployment diagram, Object diagram, Package diagram and Profile diagram[9].

Behavior diagrams are used to describe the behavior information of systems and tell readers what actions will occur in the system and the sequence of different functions. There are also seven types of behavior diagrams in UML: Activity diagram, State machine diagram, Use case diagram, Communication diagram, Interaction overview diagram, Sequence diagram and Timing diagrams[9].

3.2 Reasons for using UML

Though some people believe that all the steps during the requirements analysis phase are not important, experienced software engineers can provide many examples of failures caused by poor preparation. During the analysis and discussion phases of the project, UML provides one of the best ways of communicating with other teammates, as well as customers who do not have enough professional technology background.

UML is not a development tool, but is useful for developers to analyze and make decisions prior to development. During software engineering, UML diagrams are important support tools wisely used by project managers as blueprints to analyze requirements, make developing plans, control processes, communicate with customers, and release products.

In addition, as an internationally recognized modeling language, UML has many advantages compared with other modeling languages; some of the advantages are visualized elements and easy to understand descriptions. This is why UML was chosen for EasyLife.

3.3 The usage of UML in EasyLife

During the requirement analysis of EasyLife, use case diagrams were used in user definition and use case analysis. Use case diagrams are one type of behavior diagrams. They are used to present a graphical overview of the functionality by outlining the operations and relations between different types of actors.

In a Use case diagram, there are several elements:

- Actors: there are one or more actors used in different roles in the system, such as visitor, administrator, and assigned user.
- Actions: Use case diagram includes the actions of different actors.
- Environments: Use case diagram includes different environments where the actions take place.
- Relations: Use case diagram uses arrows to show all the relationships between actors, actions, and environments.

Let us look at one example in figure 3.1: in the diagram, there are four different types of actors: waiter, client, cashier and chef. The waiter's functions include: receive order of

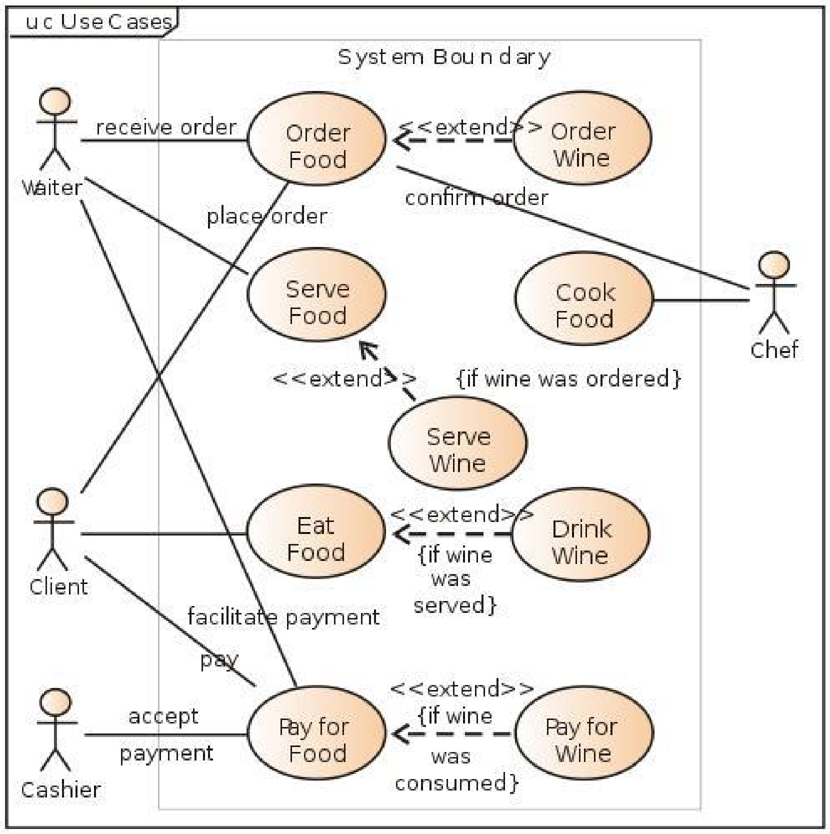


Figure 3.1: Use case diagram of a restaurant[11]

food and wine, serve food and wine. The client's functions include: order food and wine, eat food and drink wine, pay for food and wine. The cashier's functions include accepting the payment of food and wine. The chef's functions include receiving the order of food and cooking the food. In addition, the actors and actions are connected by lines and dashed-lines with arrows. The relations are described by the comments on the line. All the actions take place in an environment called "System Boundary".

3.4 UML Model building in Microsoft Office Visio

There are several different tools used to draw UML diagrams. Microsoft Office Visio is one of the best choices. Take Microsoft Office 2007 for example: users can create different file types and drag the components into the canvas instead of drawing them one by one. What is more, the widgets in Visio are internationally recognized and render professional diagrams.

All UML diagrams during the requirements analysis of EasyLife were built using Microsoft Office Visio 2007.

Chapter 4

User definition

As we mentioned in Chapter 3, use case diagrams will be used to express the actions of different users. We will draw one use case diagram for each type of user with some minor changes to make the diagrams easier to understand.

4.1 User classes

There are three types of users in this system and they have different rights and requirements when they enter the system.

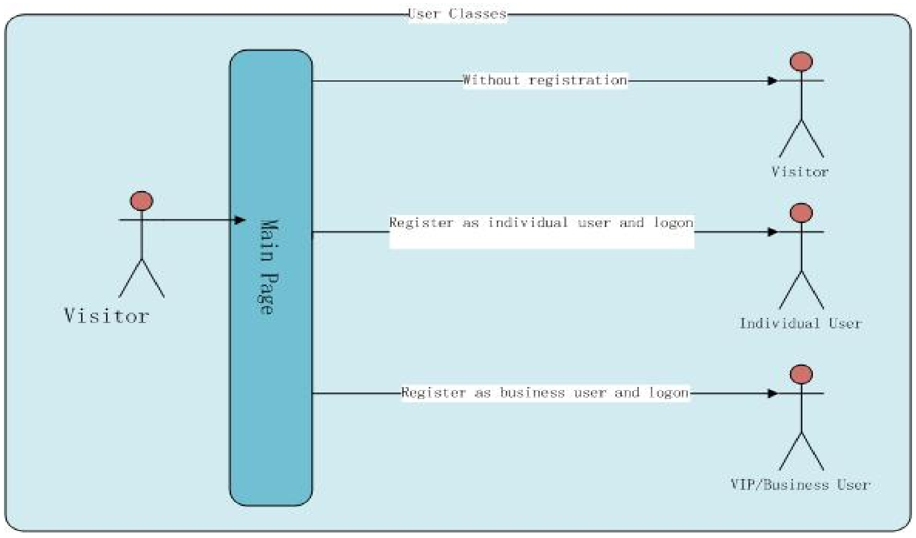


Figure 4.1: User classes of EasyLife

As Figure 4.1 shows, there are three types of users in EasyLife: visitors, individual users and VIP/business users. When a visitor enters the website, he/she will first be led to the home page: the information list page. A visitor has some basic rights, if he/she wants to get more rights, he/she can choose to register and login to become an individual user, or even a VIP/business user.

The detailed definitions of visitors, individual users and VIP/business users are outlined in Sections 4.2, Section 4.3 and Section 4.4.

4.2 Visitors

Visitors are customers who do not register, so they only have some basic rights.

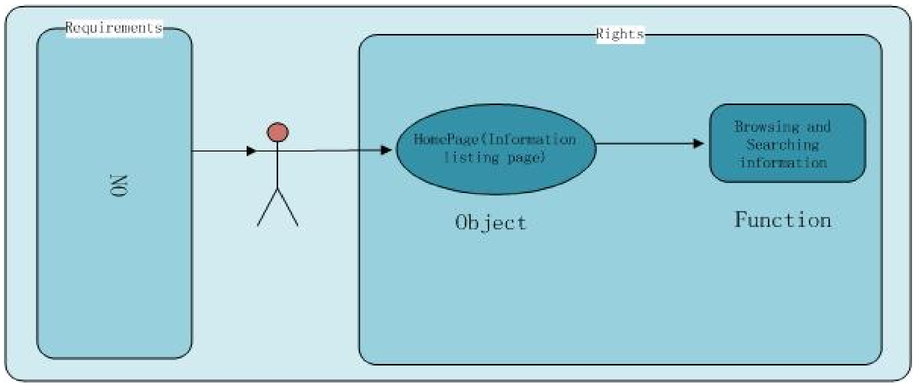

Requirements
NO
Rights
HomePage(Information listing page)
Browsing and Searching information
Object
Function

Figure 4.2: Use case diagram_visitor

As per Figure 4.2, anyone can become a visitor without any requirement. At the same time, a visitor only has basic rights: 1. Browse the home page (Information listing page). 2. Search information.

The following is the use scenario of a visitor:

User: Luke

Time: 2010/5/20

Goal: Get house renting information in Burnaby, B.C.

Background: Luke is a new international student who will begin his academic career in SFU this September. Before he comes to Vancouver, he decides to get house rental information near SFU Burnaby campus by using EasyLife.

Scenario: When Luke first enters the website, he sees the Register/Login buttons that advise him to register or login. However, Luke doesn't want to register,he feels that he only needs to do a basic search. So he decides to browse the website as a visitor.

Firstly, he finds there is a section on the home page called house rental. When he clicks on the title, an information list of house rental appears. After scanning the page, Luke realizes that he should search houses located in North Burnaby because there are too many rental advertisements. So in the search section of the homepage, he chooses his location as Burnaby and clicks on the search button. Luckily, he finds the house he wants and signs the lease shortly after contacting the landlord.

4.3 Individual users

Individual users are registered members with more rights compared to visitors.

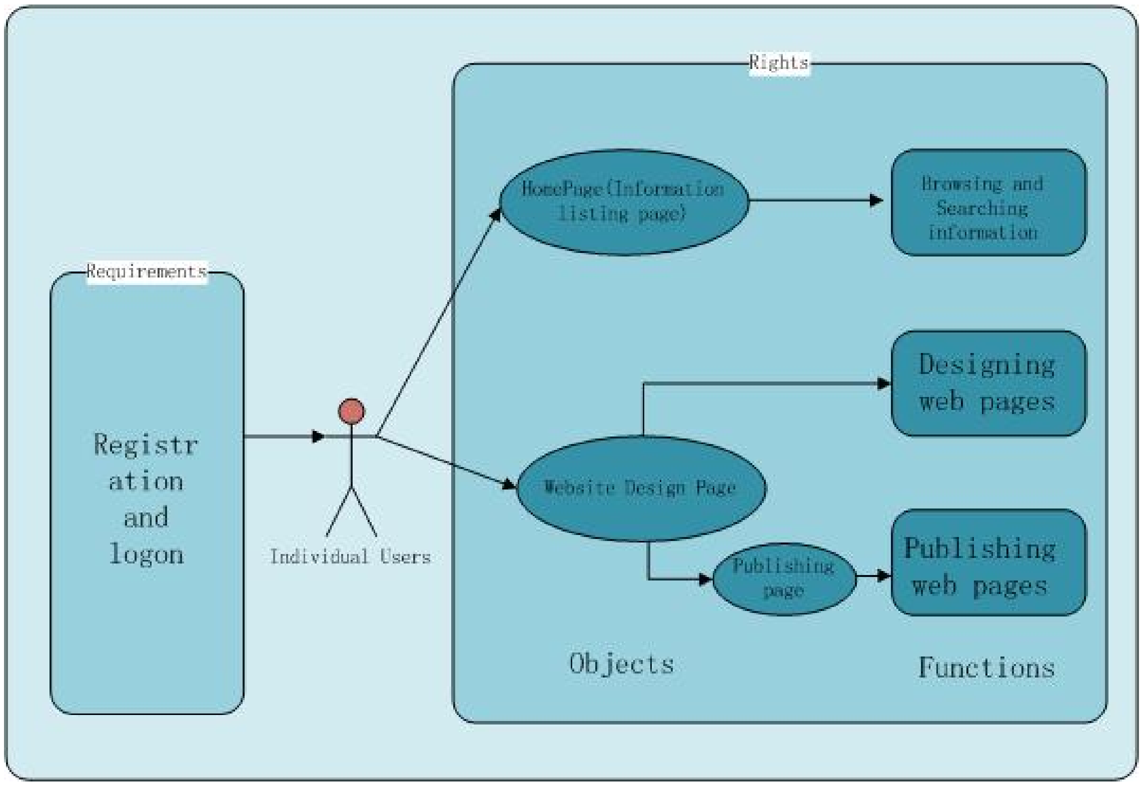


Figure 4.3: Use case diagram_individual user

According to Figure 4.3, the requirements of becoming an individual user include: Registration and login. During the registration, he/she needs to provide a nickname and the password to login, and the email address to activate his/her account. An individual user has the following rights:

1. Browse the home page (Information listing page), search information.
2. Design his own page by using basic templates and widgets.
3. Publish the page under EasyLife.

The following is the use scenario of an individual user:

User: Luke

Time: 2010/12/10

Goal: Create an online page to sell Christmas gifts.

Background: After studying one semester at SFU, Luke starts to get used to his life in Canada. However, the international tuition fees are very high, Luke thinks he needs to earn some money. With Christmas coming, he decides to sell the gifts he bought from China online in Vancouver. Once again, EasyLife would be a good way to do this.

Scenario: Luke enters the homepage of EasyLife, then clicks the register button and goes to the registration page. After filling out the necessary information, such as nickname, password and email address, he completes the registration and logins in. Because Luke wants to create his own webpage, he goes to the design page by clicking the “Design your own webpage” link.

Now, Luke is on the design page; on this page, he chooses the personal sale template, and designs his own webpage by following the instruction shown on the page. After reviewing and making sure the layout and information are what he wants, he clicks on the “Publish” button to go to the publish page. On the publish page, he fills the summary information of his online shop, such as name, location, brief description, and then publishes the webpage. Finally, he sells out all his stock before the end of the holidays.

4.4 VIP/Business users

The business users, or VIP Users, have some more rights compared with individual users. At the same time, they need to meet more requirements to get these rights.

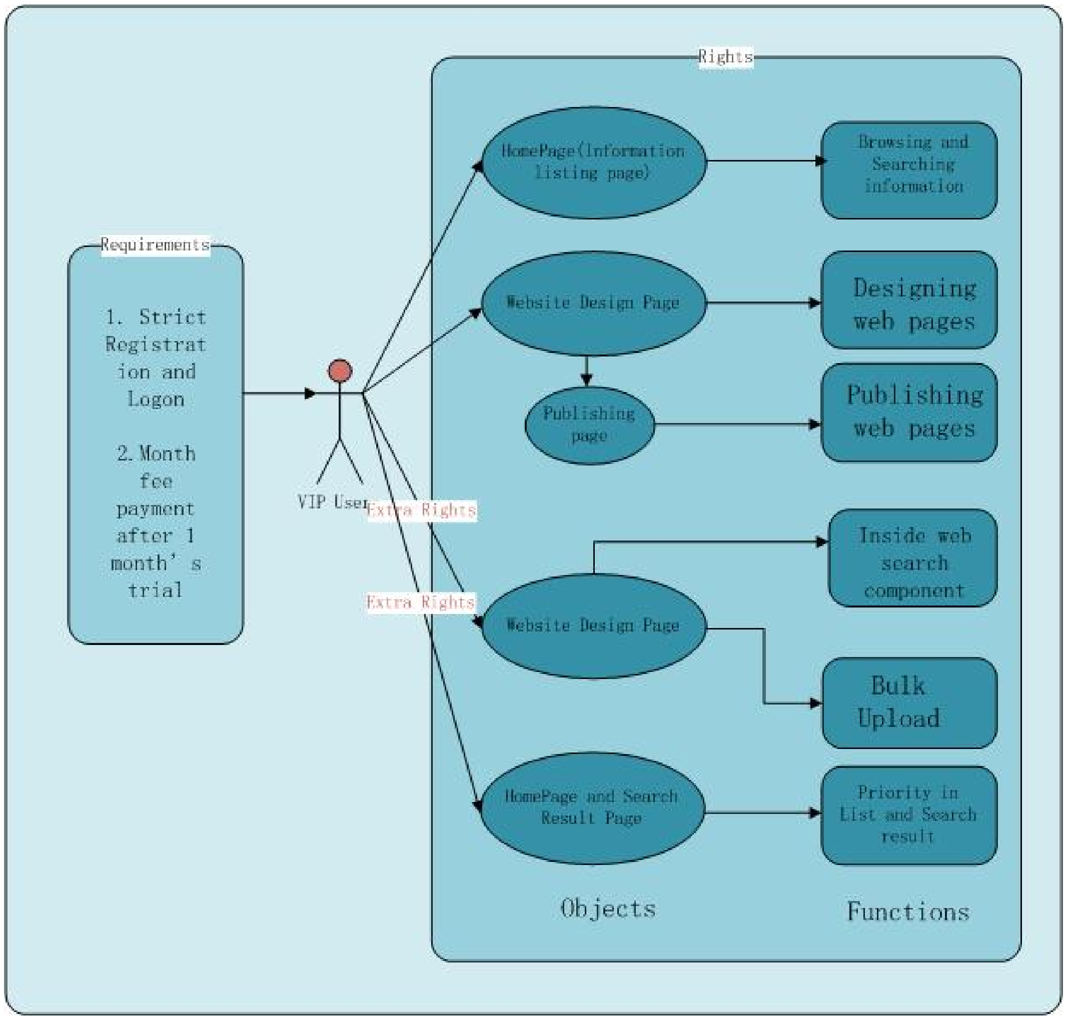

Rights
Requirements
1. Strict Registration and Logon
2. Month fee payment after 1 month's trial
VIP User
Extra Rights
Extra Rights
HomePage(Information listing page)
Browsing and Searching information
Website Design Page
Designing web pages
Publishing page
Publishing web pages
Website Design Page
Inside web search component
Bulk Upload
HomePage and Search Result Page
Priority in List and Search result
Objects
Functions

Figure 4.4: Use case diagram_VIP/business user

According to Figure 4.4, a VIP/business user needs to meet the following requirements:

1. Real name registration and login (Information needs to be provided in registration include: last name, first name, nickname, password, email address, number of one piece of his ID)
2. After one month's trial, the user must pay for continued usage.

Accordingly, a VIP/business user has the following rights:

1. Browse the home page (Information listing page), search information.

2. Design his/her page by using both the basic and advanced widgets (e.g. search Bar, and bulk upload).
3. Publish the page under EasyLife.
4. Priority in the search results.

The following is the use scenario of a VIP/business user:

User: Luke
Time: 2014/09/15
Goal: Open an online shop to sell tea leaves.
Background: Luke graduates in June, 2014 with a business degree. After graduation, he chooses to build his own business to import tea leaves from China and sell them in Vancouver. He decides to start by opening a virtual store. There is no doubt that EasyLife is still his first choice. But this time, he wants to be a VIP/business user to get extra priority to promote his business.

Scenario: Once Luke logs in as an individual user, he chooses to upgrade to a VIP/Business user, and on the pop out page, he fills out the required additional information, such as last name, first name, the number on his driver's licence. He then clicks "Upgrade" to become a VIP user.

After that, Luke enters the design page, and chooses a template to create his own page. As a VIP user, he can bulk upload all his products information at one time and decorate his page. After publishing the page, he types tea in the search bar, and find his online shop is at the top of the list.

4.5 Use cases

After defining the user classes, drawing the use case diagrams and analyzing the use scenarios, we can begin to write use cases to describe the main functions of EasyLife.

Use-case: Access online information
Primary actor: Visitor, individual user or VIP user
Goal in context: To see the information of online pages in different section
Pre-conditions: Visitor, individual user or VIP user has landed on the system.
Trigger: Visitor, individual user or VIP user decides to go to the information page of one

section.
Scenario:

1. Visitor, individual user or VIP user: lands on the system.
2. Visitor, individual user or VIP user: clicks the specific section he interested in.

Exceptions: The system will crash if too many users land on the system: improve the quality of the system and limit the max number of landing on users.
Priority: Essential, must be implemented
When available: First increment
Frequency of use: Many times per day
Channel to actor: Via the browser
Secondary actor: Central database server
Channels to secondary actor:
Administrator: Via the browser
Central database server: Intranet (hardwired or wireless)

Use-case: Design page
Primary actor: Individual user or VIP user
Goal in context: Design own webpage in EasyLife
Pre-conditions: The individual user or VIP user has already logged in the system.
Trigger: The individual user or VIP user wants design their own webpage.
Scenario:

1. Individual user or VIP user: double click “Design you own webpage” link in homepage.
2. Individual user or VIP user: choose the template he want, and drag it into the canvas.
3. Individual user or VIP user: design the page following the instruction.
4. VIP user: use extra component and function when design the page.

Priority: Essential, must be implemented
When available: First increment
Frequency of use: Many times per day
Channel to actor: Via the browser

Secondary actor: Central database service
Channels to secondary actor:
Central database service: Intranet (hardwired or wireless)

Use-case: Publish pages
Primary actor: Individual user or VIP user
Goal in context: To publish the pages designed
Preconditions: The individual user or VIP user has already logged into the system and finished design.
Trigger: The individual user or VIP user has finished the design and wants to publish his webpage in EasyLife.
Scenario:

1. Individual user or VIP user: Click "Publish" link in design page to enter the "publishing webpage". page.

2. Individual user or VIP user: Fill the information in the page.

3. Individual user or VIP user: Click "Publish".

Priority: Essential, must be implemented
When available: First increment
Frequency of use: Many times per day
Channel to actor: Via the browser
Secondary actor: Central database service
Channels to secondary actor: Central database service: intranet (hardwired or wireless)

Part III

Interface Design

Chapter 5

Adobe Fireworks

5.1 Introduction of FW

Adobe Fireworks ("FW"), is a bitmap and vector graphic editor originally developed by Macromedia and acquired by Adobe in 2005[5]. FW is popularly used by web designers to quickly design prototypes of interface or even entire websites.

FW inherits the classical features of other products in Adobe Creative Suite, such as the interface, the smart guide and the operation style. Adobe Fireworks CS4, which is the current released version, supplies users some new cool features which can help us design webpage prototypes in an easy way[5, 12].

5.2 Reasons of using Adobe FW

Adobe Fireworks CS4 has many features that make it one of the best tools to design the webpage prototypes.

- Page Set[12]: In website design, many pages may have similar styles or even layouts. For example, in the website of a restaurant, every page may has the same layout, except perhaps with highlights of different tabs. In FW CS4, the user can design one template and set the template as the Master Page. Then he/she can apply the layout to all the sub pages in the set. When changes are required to be made to the layout, the user can just change it in the Master Page. It will be automatically applied to all the pages in the site.

- Smart Guide[12, 16]: Smart Guide is added in FW CS4 to help users align and position objects. When a user creates an object and moves it on the canvas, the smart guide (a dashed cross) will help him/her locate his/her object according to other relative elements). For example, when a user creates a button, and wants to insert the text into the button, he can just move the text box around the button. When the dashed cross appears, it means the text box is in the centre of the button.
- CSS Export[12]: FW is mostly a layout designing software; but if users want to use the CSS code for future development, they can choose to export the layout into clean CSS code.

There are also some more cool features in FW CS4, such as Sample Text, Common Library, and Element Import. All these features make FW CS4 a powerful design tool, so it is selected as the interface prototypes design tool for EasyLife.

5.3 The usage of Adobe FireWorks in EasyLife

FW is used as the interface prototypes design tool during the development of EasyLife.

The prototypes include:

- Home page: the home page of EasyLife with information listing.
- Design page: the page for individual users or VIP Users to design their own webpages.
- Templates: the templates used to drag into the canvas when users design their pages. They include:
  - Grocery webpage template.
  - Restaurant webpage template
  - Personal sale webpage template.
- Submitting page: the submitting page appears when users finish page design and click to publish their page.

FW will only be used in prototype design which aims to express different widgets and their relative positions. Therefore, the prototypes will only include the basic widgets (e.g. text box, image box, buttons and borders of different areas). For the layout design later, it will need more helpful software such as Adobe Photoshop and Flash.

Chapter 6

Interface prototypes design

There are four major interface prototypes in EasyLife: Home Page, Design Page, Templates, and Submitting Page. The details will be outlined from Sections 6.1 to 6.4.

6.1 Home page

Figure 6.1 is the interface prototype of home page:

Home page consists of the following widgets:

Top part:

- Logo: The logo of EasyLife.
- Location bar: user can choose his location to improve the accuracy of the search result and information listing.
- Button1: Design own website: links to webpage design page.
- Button2: Register: links to register page for visitor.
- Button3: Login: links to login page for registered user.
- Search bar: user can type query inside to search the web pages within EasyLife.

Highlight part:

- Display the highlight pictures: e.g. important website information, advertisement and suggested pages.

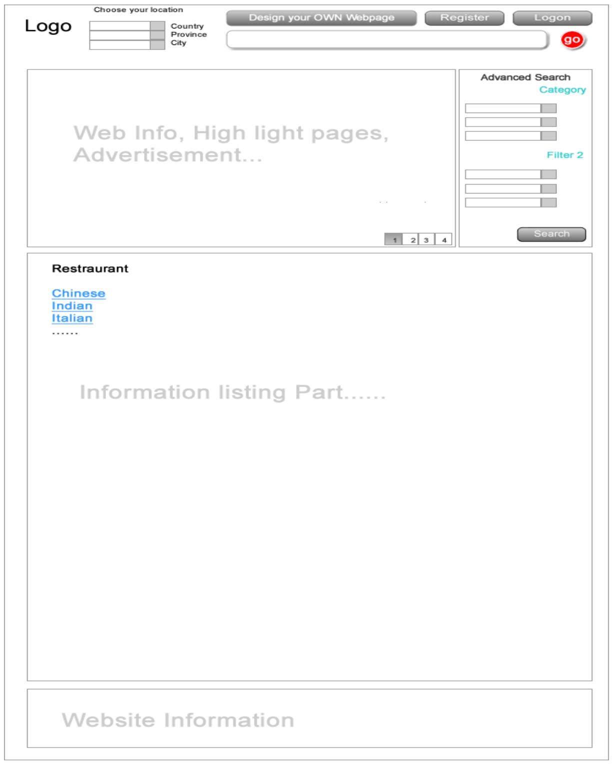


Figure 6.1: Home page prototype

Advanced search bar:

- Includes the advanced search choices to improve the accuracy of search results: e.g. search by the category of website.

Information listing part:

- Contains different sections. In each section, lists the links of websites belong to this section.

Bottom part:

- Website information: includes website information such as the instruction, contact information, and copy right issues.

6.2 Design page

Figure 6.2 is the interface prototype of design page:

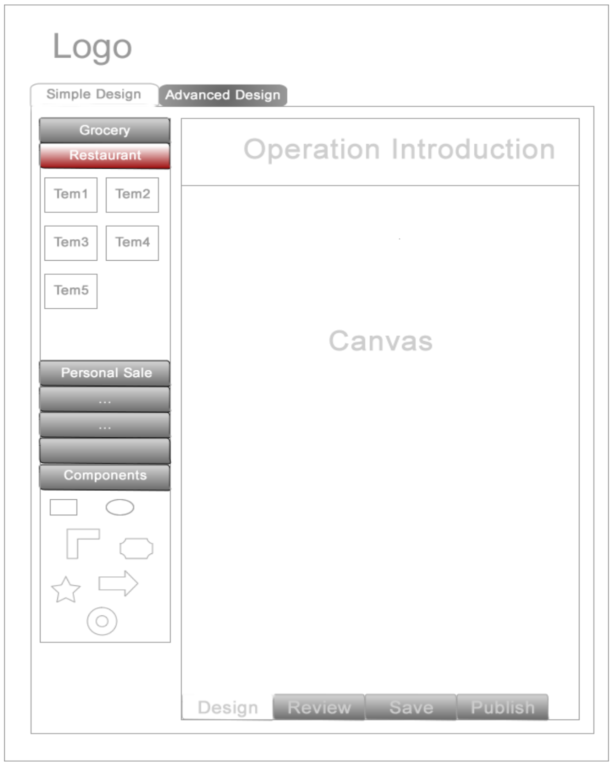


Figure 6.2: Design page prototype

Design page consists of the following components:
Top part:

- Logo: The logo of EasyLife.

Tool Bar:

- Templates of different kind of websites.

- Components section: website components used for further design.

Canvas:

- Instruction box: display the instruction for each templates or components which is currently selected.

- Canvas: Canvas used to design web page.

- Button1: Design: shows during design process.

- Button2: Review: used to review the page during design.
- Button3: Save: save the page during design.
- Button4: Publish: links to the publish page after finishing design.

6.3 Templates

Templates are used for users who do not have much computer science knowledge. By using templates, users can just drag the templates into the canvas, and design inside the templates following the instruction displayed in instruction box.

By now, there are three kinds of templates: Grocery, Restaurant, and Personal Sale. Their interface prototypes are outlined in Sections 6.3.1, 6.3.2 and 6.3.3.

6.3.1 Grocery

Figure 6.3 and Figure 6.4 are the interface prototypes of the grocery webpage:

- Home page

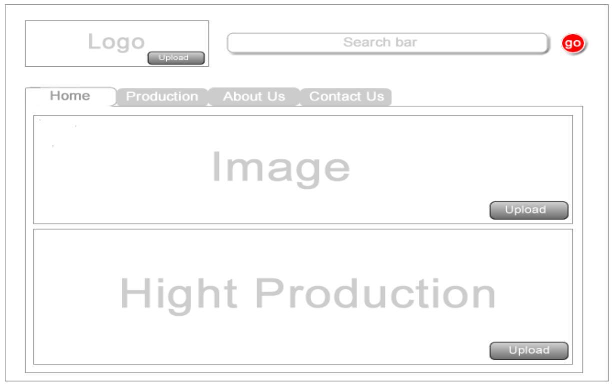


Figure 6.3: Template_grocery_homepage prototype

The home page of grocery interfaces includes:

Top part:

- Logo: The logo of the grocery.
- Search bar: it is used for searching the items in the grocery website.

Main part:

- Image: designer can upload the image of their companies. They can also upload important notices, sales promotionsm etc..
- Highlight Items: the highlight product of the grocery.

• Production page:

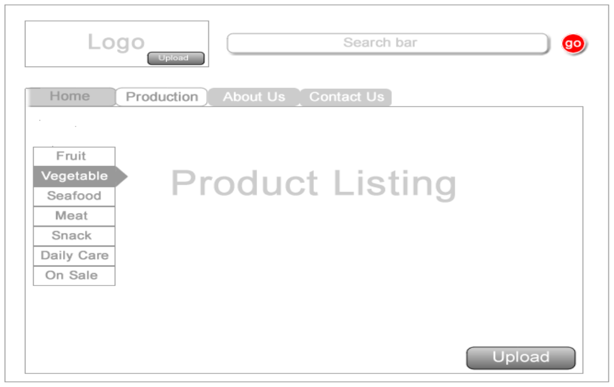


Figure 6.4: Template_grocery_production page prototype

The production page of grocery interfaces includes:

Top part:

- Logo: The logo of the grocery.
- Search bar: it is used for searching the items in the grocery website.

Main part:

- Classified bar: user can browse the products in different categories by clicking the relative button in the classified bar.
- Products listing: the list of the products within this section.

6.3.2 Restaurant

Figure 6.5 and Figure 6.6 are the interface prototypes of the restaurant webpage:

• Home page:

Home page of restaurant interfaces includes:

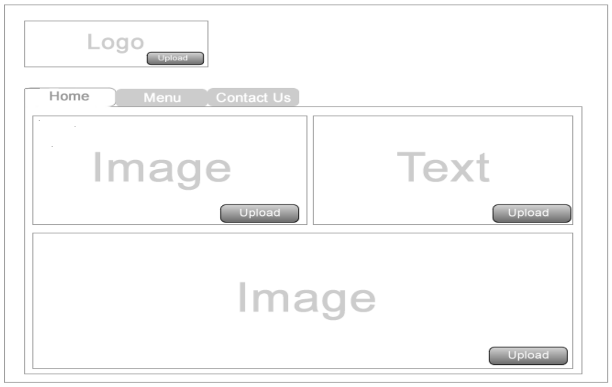


Figure 6.5: Template_restaurant_home page prototype

Top part:

- Logo: The logo of the restaurant.

Main part:

- Image and text boxes: designers can upload the image or input text which is used to introduce their restaurant.

- Menu page:

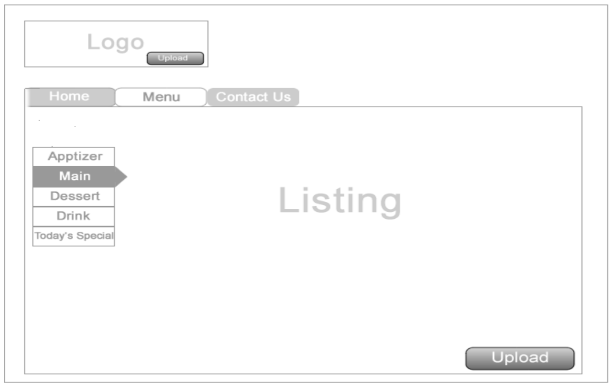


Figure 6.6: Template_restaurant_Manu page prototype

Menu page of restaurant interfaces includes:

Top part:

- Logo: The logo of the restaurant.

Main Part:

- Classified bar: user can browse the menu of different category by click the relative button in classified bar.
- Listing: the menu of the given category.

6.3.3 Personal sale

Figure 6.7 is the interface prototypes of personal sale:

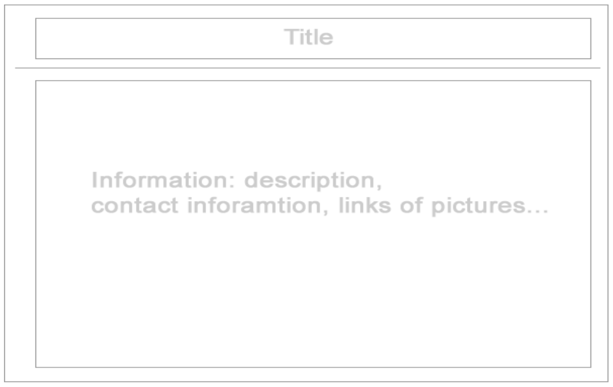

Title
Information: description,
contact inforamtion, links of pictures...

Figure 6.7: Template_Personalsale page prototype

Personal sale interface prototype includes:

- Title: the title of the sale.
- Main part: the description of the sale, contact information, links of pictures and so on.

6.4 Submitting page

Figure 6.8 is the interface prototypes of submitting page:

Submitting page interface prototype includes:

- Information input bars: the designer should input the following information in the page before publishing it in EasyLife: website name, website URL, address, postal code and the category of his/her page.

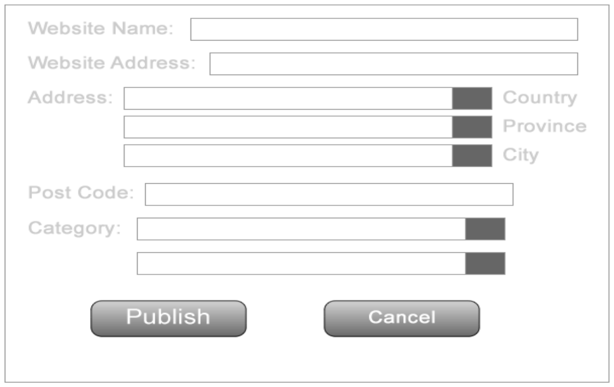


Figure 6.8: Template_submit page prototype

- Buttons: designer can choose to publish his/her page or cancel by clicking the relative buttons.

Part IV

Database Design

Chapter 7

MySQL

7.1 Introduction of MySQL

MySQL is a relational database management system which supports multiple users' access[13]. MySQL was originally developed by two Swedish scientists and a Finnish scientist: David Axmark, Allan Larsson and Michael Widenius[8]. After being acquired by Oracle Corporation and after more than 10 years of development, it has become one of the world's most popular database softwares[8].

MySQL is written in C and C++, and works with most operating system such as Linux, Mac OS X and Microsoft Windows. Because of the well-known advantages, such as fast speed, reliability, and ease of learning, MySQL is widely used by both large organizations and small corporations, even individual users. More importantly, as an open source software, MySQL is used by many computer science students when they are learning courses related to database system and data mining[8].

7.2 The reason of using MySQL

MySQL has its own advantages compared to other popular database software.

- MySQL vs Microsoft Access[2]

  Microsoft access is a database management tool which is used by web maintainers to store information and operate data in a local system. It was developed by Microsoft and first released in 1992[2]. As a Microsoft product, Access is compatible with most Microsoft software, and very easy to use.

The advantages of MySQL compared to Microsoft Access are:

- Cost: MySQL is an open source software and free to use.
- Support for large databases: MySQL is more powerful when required to handle a large database.
- Support for multiple users: MySQL supports multiple users access to the database at the same time while Access only allows one user to use the database at any time.
- Support for multiple operating systems: MySQL can be used with different operating systems while Access is only compatible with Microsoft Windows.

- MySQL vs MS SQL[3]

  MS SQL, which stands for Microsoft SQL Sever, is another Microsoft database software which was first released in 1989[3]. Compared to Access, it is a more professional database product of Microsoft. MS SQL has more functions, such as search, query, analyze and report, and is preferred by web masters.

  The advantages of MySQL compared to MS SQL are:

  - Cost: MySQL is an open source software and free to use.
  - Support for multiple operating systems: MySQL can be used among different operating system while MS SQL is only designed for Microsoft Windows.

- MySQL vs Oracle[3]

  Oracle is a relational database management system developed by Oracle Corporation[4]. It has a free version but does not include all the functions. Oracle is mostly used for very large applications and the users need to have relatively extensive knowledge and skills to deal with large amounts of data.

  Compared to Oracle, MySQL is very simple to use, and powerful enough for the development of EasyLife.

In light of the above arguments, MySQL was selected as the database software for EasyLife.

7.3 The usage of MySQL in EasyLife

MySQL is used in EasyLife development mainly in the following two parts:

- Data Storage: MySQL is used to store different types of data in EasyLife, including user data and webpage data. Details are outlined in Chapter 8.
- Search engine: The data stored in MySQL is used in the search engine implementation. Details are outlined in Chapter 10.

Chapter 8

Database design and implementation in MySQL

8.1 Data specification

The data which used in EasyLife can be divided into two parts:

- User information

  User information(Table 8.1) is the data used to describe any customer who has an account in EasyLife:

Name	**Type**	**Range**	**Description**	**Comment**
User ID	String	>1 char	User ID used to login	Primary Key
Last Name	String	>1 char	It is a user's last name	VIP user only
First Name	String	>1 char	It is a user's first name	VIP user only
ID Number	String	>1 char	It is the number of user's ID	VIP user only
Password	String	>1 char	It is the password to lonin	
Email address	String	>1 char	It is the email address used to register	

Table 8.1: Data table_User

- Webpage information

  Webpage information(Table 8.2) is the data used to describe the webpage created and designed by a specific user:

Name	**Type**	**Range**	**Description**	**Comment**
Page ID	String	>1 char	It is the page id of the web page	Primary key
Owner	String	>1 char	ID of the user who created the page	Foreign key
Web name	String	>1 char	The name of the web page	After publishing
Web address	String	>1 char	It is the address of the web page	After publishing
Add_Country	String	>1 char	Country name of the web site	After publishing
Add_Province	String	>1 char	Province name of the web site	After publishing
Add_City	String	>1 char	City name of the web site	After publishing
Postal Code	String	>1 char	Postal code of the web site	After publishing
Category	String	>1 char	Category of the web site	After publishing
Flag	Boolean	0 or 1	If the web page is published or not	After publishing
Text	String	>1 char	All the text within the web page	After publishing
Last Modified	String	>1 char	The last time the page is modified	After publishing
page_property	String	>1 char	All the property of the web page	After publishing
Description	String	>1 char	Description of the web page	After publishing

Table 8.2: Data table_Page

8.2 Data structure

8.2.1 Overview of Entity-Relationship Diagram

Entity-Relationship Diagram (ERD), is an abstract and graphic representation method of the data base structure based on Entity-Relationship Modeling[14]. It includes three parts:

- Entities

  An entity can be viewed as an object which contains different attributes. It can be an abstract form or a physical object. Normally, an entity is a noun in database modeling (e.g. a house, a user, or a time schedule), and represented as a rectangle in ERD.

- Relationships

  Relationships are the connections between different entities. For examples, the design page action can be viewed as a relationship between entity ”User” and entity ”Webpage” in EasyLife. Normally, a relationship is a verb in database modeling, and represented as a diamond in ERD.

- Attributes

  Both entities and relationships can have their own attributes. They are used to describe the entities or relationships in ERD. In an entity, there should be at least one attribute which can identify the entity, and this attribute is the primary key of the entity. Normally, an attribute is a noun in database modeling, and represented as an ellipse. In addition, the primary key is the ellipse whose name has an underline.

8.2.2 ERD of EasyLife

Figure 8.1 is the ERD of EasyLife:

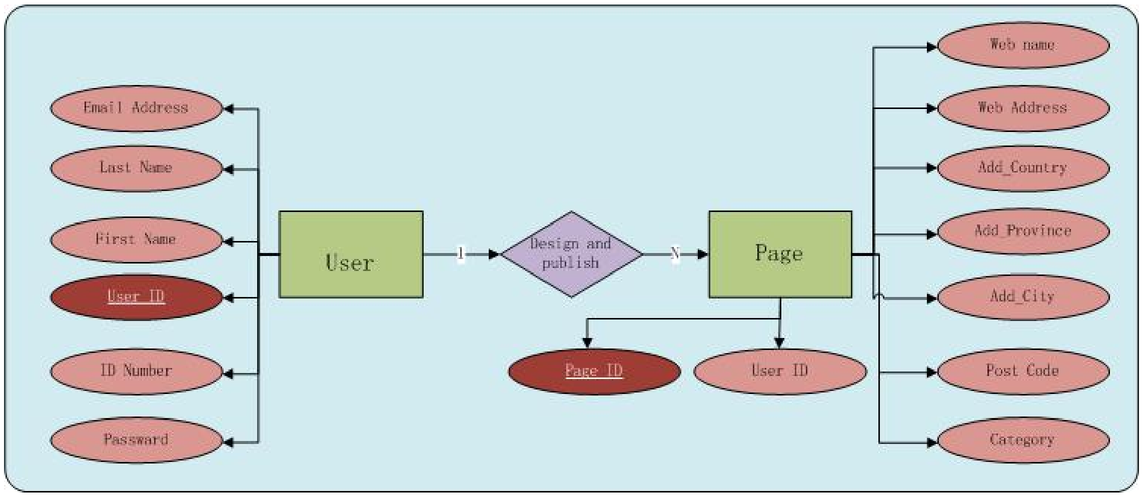


Figure 8.1: ERD of EasyLife

8.3 Database implementation in MySQL

8.3.1 Two ways of database implementation in MySQL

MySQL has two ways to implement database, one way is through the Unix command line interface, the other is through a graphic interface. For a user who is not used to Unix commands, he/she can choose to either download the official MySQL Workbench from the MySQL official website[1], or get some third party software such as phpMyAdmin, HeidiSQL or Adminer.

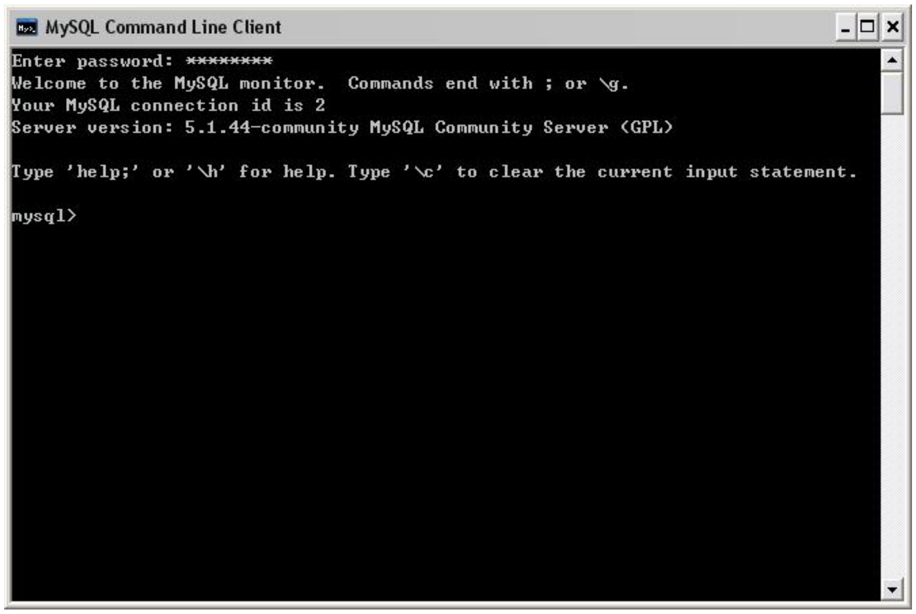
Since the implementation in graphic interface is straightforward, we focus on the introduction of database management through the Command Line Client in MySQL.

[1]To download the MySQL workbench, please go the the official cite of MySQL: http://www.mysql.com/downloads/

8.3.2 Connect to the database server

The first step in creating and managing data in MySQL is to connect to the server. During the installation and deployment of MySQL, the user can create the server under his user name. After that, the user should first run the MySQL Command Line Client in "Start - All Programs - MySQL - MySQL Server 5.1 - MySQL Command Line Client".

As shown in Figure 8.2, in the pop out window, the user can connect to the server by typing in the password. If the server is connected, it will provide some introductory sentences followed by a mysql> prompt (Note: all the commands in the Command Line Client are case sensitive).



```
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.44-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

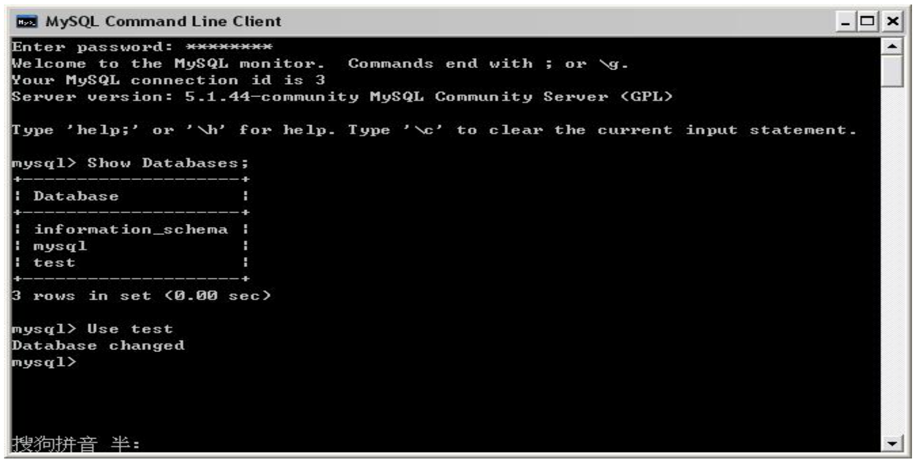
Figure 8.2: Screenshot_MySQL_connect to the sever

Users can exit the Command Line Client by typing "quit" and pressing Enter.

8.3.3 Choose a database or create a new database

- Choose an existing database

  As shown in Figure 8.3, if "**Show Databases;**" is typed, in the Command Line Client, the user can see all the databases in the server. The user can then type "**Use 'database name'**" to choose an existing database to manage. If it succeeds, there will be one line showing "**Database changed**"



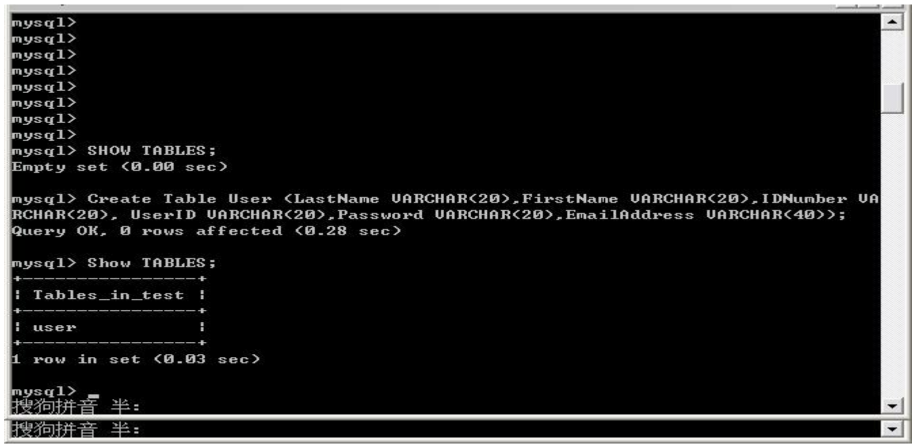
```
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.1.44-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> Show Databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)

mysql> Use test
Database changed
mysql>

搜狗拼音 半:
```

Figure 8.3: Screenshot_MySQL_choose database

- Create a new database

  If you want to create a new database, what you can do is ask your MySQL administrator to give you permission or type **"GRANT ALL ON 'your_database_name'.* TO 'your_mysql_name'@'your_client_host';"** to create a new database.

8.3.4 Create table



```
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql> Create Table User (LastName VARCHAR(20),FirstName VARCHAR(20),IDNumber VA
RCHAR(20), UserID VARCHAR(20),Password VARCHAR(20),EmailAddress VARCHAR(40));
Query OK, 0 rows affected (0.28 sec)

mysql> Show TABLES;
+----------------+
| Tables_in_test |
+----------------+
| user           |
+----------------+
1 row in set (0.03 sec)

mysql> _
搜狗拼音 半:
搜狗拼音 半:
```

Figure 8.4: Screenshot_MySQL_creat table

As shown in Figure 8.4, type "Show Tables;" to see the current tables in the database.

For example, there is no table in the database “test”.

If we want to create a table named “User” , we can type the command:

mysql> **CREATE TABLE User (LastName VARCHAR(20), FirstName VARCHAR(20), IDNumber VARCHAR(20), UserID VARCHAR(20), Password VARCHAR(20), EmailAddress VARCHAR(40));**

If it succeeds, when we type “**Show Tables**”, the table named User will be shown although there is no row in it yet.

8.3.5 Load data into table

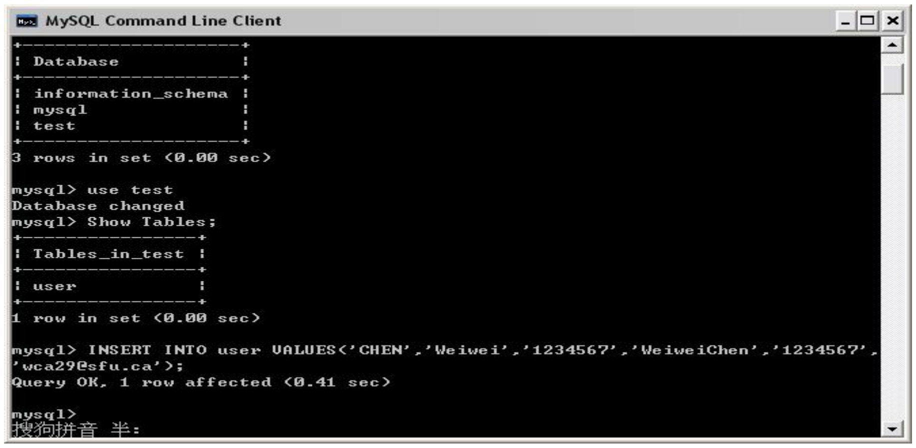
To insert data into a table, the user can choose two different ways. When there are many rows you want to insert, you can first create a txt file or some other type of file, then load the whole file into the table. The command is:

mysql> **LOAD DATA LOCAL INFILE ‘/path/User.txt’ INTO TABLE User;**

If you want to insert just one or two lines, maybe it is better to insert the row in the table directly by using the following command:

mysql> **INSERT INTO User VALUES (‘Chen’,‘Weiwei’,‘1234567’,‘WeiweiChen’, ‘1234567’,‘wca29@sfu.ca’);**

Below is the screen short after inserting the row:



```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)

mysql> use test
Database changed
mysql> Show Tables;
+----------------+
| Tables_in_test |
+----------------+
| user           |
+----------------+
1 row in set (0.00 sec)

mysql> INSERT INTO user VALUES('CHEN','Weiwei','1234567','WeiweiChen','1234567',
'wca29@sfu.ca');
Query OK, 1 row affected (0.41 sec)

mysql>
搜狗拼音 半:
```

Figure 8.5: Screenshot_MySQL_load data

By using the above method, we created two tables in EasyLife: one is named “user” which is used to store user information, the other is named “page” which is used to store page information.

Part V

Search Engine Implementation

Chapter 9

Overview of website search engines and Apache Lucene

Most online trading websites have their own search engines and so does EasyLife. This chapter provides the background for web search engines, and introduces a useful open source tool named Apache Lucene to configure search engine.

9.1 Three methods of putting a search engine in a website

There are three ways to insert a search engine in a website[10]: installing a search engine library, using a free or commercial third party hosted search engine service, or using major search engine. The following three sections will introduce these three methods.

9.1.1 Installing a search engine library

During the website development phase, developers can choose to install the search engine library themselves. There are many free search engine libraries to download, and the developers can choose to use the one which adapts to their websites.

There are two types of search engine libraries. One will search the entire website every time a user sends a search request. The other will build an index of all the web pages first, and only search over the index when a user uses the search engine. The first one is easier to configure, but it will become more inefficient when the website gets larger. The second type of library is more efficient and the web maintainer can choose to refresh the index anytime he wants.

This method has the following advantages:

- Developers can customize the search result and even the score proportion.
- No third party advertisement occurs on the website.
- Developers can index their web pages whenever they want. (For the library that builds an index)

The disadvantages are:

- Need to install the necessary environment platform and other tools.
- Need to have strong computer science background and coding ability.

9.1.2 "Using a free or commercial third party hosted search engine service"

If the developers don't have enough computer science background, or don't want to spend much time on implementing a search engine, they can use a free or commercial third party hosted search engine service.

For example, after you register and login to a website what supplies a third party hosted search engine service, you can send the request by supplying the necessary information such as the URL of your website. After that, the search engine service will index your website and plug the search engine into your HTML file.

This method has the following advantages:

- No need to have the ability to install the library into your website and no need to program or configure the search engine.
- No need to worry about the index and layout of the search result page.

The disadvantages are:

- The time of indexing the webpages is dictated by the third party search engine service, not by the website maintainers.

- Some third party search engine services will require you to insert their own advertisements into the search result display page. Also the URL shown in the address box of the search result page will not be that of your website but the search engine supplier.
- Developers don't have the absolute right of designing the search result page.

9.1.3 "Using major search engines"

Finally, you can use a major search engine such as Google as your website's search engine. To realize this, developers just need to go to the Google Custom Search Engine page and complete the online form. However, by using this method, the developers' rights of indexing and displaying the results will be more limited.

9.1.4 Our choice

After analyzing all the three methods, we decided to use the first method that of installing a search engine library. The library we use is Apache Lucene, which is an open source Java library supplied by Apache Software Foundation.

9.2 Overview of Apache Lucene

9.2.1 Overview

Apache Lucene is an open source/free download information retrieval library, which is purely built in Java. By using Lucene, developer can build its full-text search engine by inserting the library into their main project and building the necessary interfaces[1].

9.2.2 Features

As mentioned in its official site: "Lucene offers powerful features through a simple API"[6]. It has the following features:

1. "Scalable, High-Performance Indexing"[6]

[1]The current version of Apache Lucene is 3.0.1 which was released on Feb 26th, 2010. To download and get started with the latest version of Lucene, please go to the official website of Apache Lucene: http://lucene.apache.org.

As mentioned in section 9.1, there are two types of search engine libraries: one will search the entire website every time a user sends a search request, the other will build an index first and then search the result in the index instead of travelling though the entire website.

Apache Lucene is the second type of library which has a high quality indexing algorithm. By using the fast and accurate indexing method, Lucene can build an index whose size is only around 20% - 30% of the original text[6]. By using the index, the search speed can be improved to over 20MB/minute[6].

2. "Powerful, Accurate and Efficient Search Algorithms"[6]

   Apache Lucene uses several different ways to make the search accurate:

   - "Ranked searching"[6]: after searching, Lucene will give the score of each file according to the query. The score can be done in one of several ways. For example, the frequency of the appearance of a query in the file, the field in the file it appears in and so on.
   - "Field searching"[6]: Lucene defines several fields of a file, e.g. title, author and contents. Different fields will have different weights during the scoring. Also, the user can decide which text belongs to which fields according to his/her own needs. In addition, the user can choose to sort the result by any field.
   - "Powerful query types"[6]: Lucene supplies many kinds of algorithms to deal with queries, such as phrasing, wildcarding and ranging.

   Beside these, there are more features listed in Lucene's site, such as "date-ranging searching"[6], "multiple-index searching with merged results"[6], and "simultaneous update and searching"[6]. All these features make the searching more accurate and effective.

3. "Cross - Platform Solution"[6]

   Lucene is built in Java, but it also supports many other programming languages.

9.3 The process of search engine building with Lucene

As a full-text search engine library, Lucene already has the algorithms for data retrieval, such as indexing, parsing, searching, and scoring. All the users need to do is to write the interface and to connect data flow. It includes the following steps:

1. Adding Lucence library

   After creating a new project in the programming platform, the user needs to add the Lucence library into the project for further use. For example, if we are building the project in MyEclipse and using Lucene 3.0.1 for search engine implementation, we should add the jar file: lucene-core-3.0.1.jar into our project as an external jar file.

2. Accessing a database (If a database is used as a data resource)

   During the search engine building, the source data can be from real text files, or the database. If the developer wants to use the database as the data source, he/she needs to use some application programming interface (API), e.g. JDBC, to access the database.

3. Indexing

   Indexing is one of the most important steps during the implementation. In this step, the user needs to create an in-memory index directory and add all the documents into the IndexWriter.

   "In Lucene, a document is the unit of index and search"[15]. It does not have to be an English text file. For example, when we are searching a data table of a database, each row of the table can be viewed as a document.

   Moreover, a document can consist of one or more fields. Each field may have its own property and weight. For example, a text file may include title, authors, and content. When we add a text file in to the index, it can be represented as a document. We can add the title into the "title" field, add the authors into "author" field, and add the content into the "content" field. During the search, we can choose whether to score each using its weight, to make the search more accurate.

4. Managing the query

After creating the index, the next step is managing the query. As mentioned before, Lucene supplies several ways of dealing with queries, such as phrasing, wildcarding and ranging.

5. Searching and scoring

   This step first hands the query into an IndexSearch. After that, by using Lucene's scoring algorithm, it will return a list of hit documents, which are sorted by the scores.

6. Creating graphic interface

   This is the last step of the search engine building. The developer can choose to input the query and print out different fields of the hit documents in the command line window. But in most situations, we need to create a graphic interface to input the query and display the search result. During this step, the developer can choose the fields of the hit documents he/she wants to display, and implement the input and output interfaces with the help of external tools such as JSP/indexJSP, CSS/indexCSS and HTML/indexHTML.

There is one important thing needs to be pointed out, it is better to separate indexing and searching into different classes when implementing the search engine. The reason is that, if we put these two methods in the same class, it will refresh the index every time the user sends a search request. As a good performance search engine, the index refresh frequency should depend on the maintainer, instead of the user.

Chapter 10 describes the search engine implementation process in EasyLife using Apache Lucene.

Chapter 10

Implementation of search engine in Apache Lucene

In this chapter, we will show the process of searching engine building in EasyLife.

10.1 Preparation

Before we begin to build the search engine in EasyLife, there are some preparation steps we need to do:

1. Create project in MyEclipse. Firstly, we create a Web Project named SearchEngine in MyEclipse.

2. Download and add external libraries. After the project is created, we download and add two external libraries into the Referenced Libraries of the project. One library, named lucene-core-3.0.1.jar, is the latest version of the lucene library. The other library, named mysql-connector-java-5.1.12-bin.jar, is the latest version of the JDBC library.

3. Create classes In the project, we create three classes. One is named IndexManager.java, which is used to implement the database connection and index building. One is named SearchManager.java, which is used to implement the query parsing and searching. The last one is named SearchResult.java, which is used to create the search result object.

Now we describe our implementation of the search engine.

10.2 Database connection

In the IndexManager.java[1], using the following code, we connect the database named test, which is the source data of our searching. The url of the database is “jdbc:mysql://localhost/test”, and the user name and password of the database are both “spring”.

```
static Connection conn;
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    String url = "jdbc:mysql://localhost/test";
    conn = DriverManager.getConnection(url, "spring", "spring");
```

Program 10.1: Source code of database connection

10.3 Index building

```
static Directory dirIndex = new RAMDirectory();
StandardAnalyzer analyzer = new StandardAnalyzer (Version.LUCENE_30);
indexWriter = new IndexWriter(dirIndex,analyzer,true,
                            IndexWriter.MaxFieldLength.UNLIMITED);
ResultSet rset = stmt.executeQuery(``select * from pages'');
while(rset.next()){
Document doc = new Document();
doc.add(new Field("title",rset.getString("Web_Name"),
                    Field.Store.YES, Field.Index.ANALYZED));
     //repeat for each column in result set.
    indexWriter.addDocument(doc);}
```

Program 10.2: Source code of index building

As shown in program 10.2: in the indexing step, we first build an index directory in memory. Second, by using SQL “select * from pages”, we get all the data in the table “pages”. This data can be used in searching and in displaying the search result. After that,

[1]The source file of IndexManager.java is listed in Appendix I

in the "while" loop, we add all the columns which need to be searched or displayed in the result set.

10.4 Query parsing

Now, we switch to SearchManager.java[2]. The following code is used to read the string, parse it and build the query for searching.

```
String querystr = keyWord;
Query q_title = new QueryParser
        (Version.LUCENE_30,"title",analyzer).parse(querystr);
```

Program 10.3: Source code of query parsing

10.5 Searching

In this step, we create an indexSearcher to search the index using the query. After that, by using the TopScoreDocCollector, we get the hit documents which have the highest scores.

```
int hitsPerpage=10;

IndexManager.createIndex();

IndexSearcher searcher =
               new IndexSearcher(IndexManager.getDirIndex(),true);

TopScoreDocCollector collector =
               TopScoreDocCollector.create(hitsPerpage ,true);

searcher.search(q_title ,collector);
ScoreDoc[] hits = collector.topDocs().scoreDocs;
```

Program 10.4: Source code of searching

[2]The source file of SearchManager.java is listed in Appendix I

10.6 Graphic interfaces

The last step is creating graphic interfaces. First, we use SearchResult.java to package the fields we need in the hit documents. Second, we create two .jsp files and two .css files in the WebRoot folder of our project. Index.jsp and layout.css are used to draw the query input page, searchResult.jsp and resultlayout.css are used to draw the result display page[3].

Figure 10.1 and figure 10.2 are the screen shots of the input and output page.

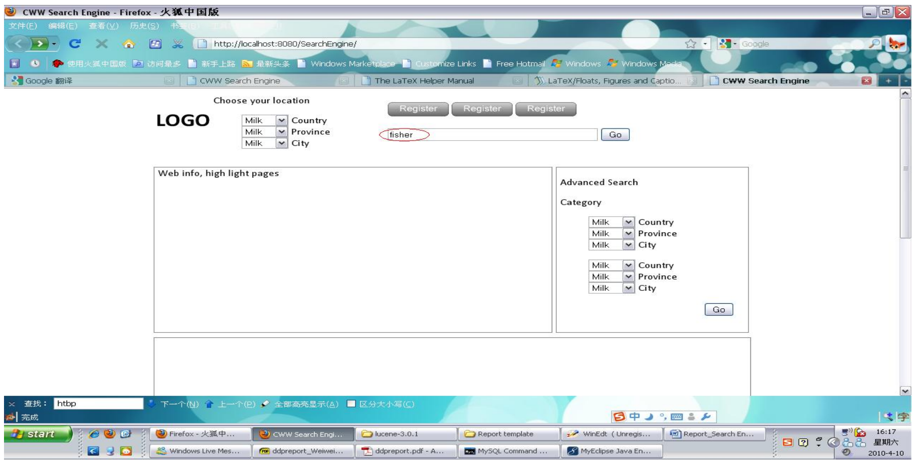
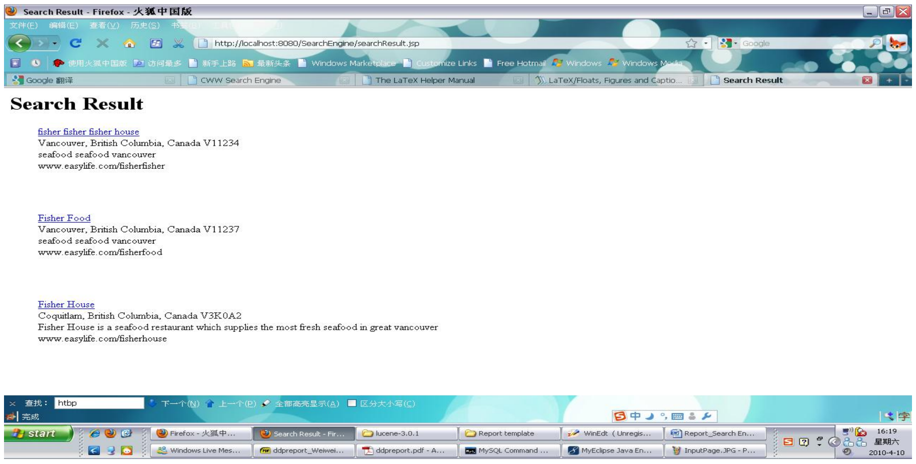


Figure 10.1: Screen shots of input page



Search Result

fisher fisher fisher house
Vancouver, British Columbia, Canada V11234
seafood seafood vancouver
www.easylife.com/fisherfisher

Fisher Food
Vancouver, British Columbia, Canada V11237
seafood seafood vancouver
www.easylife.com/fisherfood

Fisher House
Coquitlam, British Columbia, Canada V3K0A2
Fisher House is a seafood restaurant which supplies the most fresh seafood in great vancouver
www.easylife.com/fisherhouse

Figure 10.2: Screen shots of output page

[3]The source files of Index.jsp, searchResult.jsp, layout.css and resultlayout.css are listed in Appendix I

As we can see, in the input page (the homepage of EasyLife), when the user types ”fisher” in the search bar at the top right of the page and clicks GO, all the hit pages will be returned and shown in the result page.

Part VI

Conclusion

Up to this point, we have finished four parts of the development. These include: requirement analysis, interface design, database design and search engine implementation.

During the requirement analysis phase, we used UML as the main tool to draw the use classes and use case diagrams. Based on the diagram, we wrote the use scenarios and use cases. Requirement analysis is a basic and very important step in software engineering and project development. Through successful requirement analysis, we built a solid foundation for the development of EasyLife in the future.

During the prototype design phase, we used Adobe Fireworks as the main tool to design the prototypes of pages. The prototypes are blueprints for future page beautification. In addition, the prototype design can give us some ideas about database design and other applications within EasyLife.

During the database design phase, we created two tables. Based on the database design, we drew the ERD diagram and built the database using MySQL. Database is the data center of a website. After building the database, we can begin the other development steps such as data input, data output and search engine implementation.

The last development step was the search engine implementation. With the help of Apache Lucene, we successfully built a mini search engine in EasyLife, and achieved basic search functions using a graphic interface.

These four parts, combined with the steps shown in Kefu Zhao's capstone project report[4], describe the whole development process of a Website named EasyLife. Through all the steps, we achieved the main functions of EasyLife, which include:

- Graphic interface, information storage and reading
- EasyWeb - an easy use webpage design tool
- Data searching

However, because of the shortage of time and knowledge, there are still some areas for improvement, they include:

- Page beautification
- Improvement of searching accuracy

[4]Please refer to Kefu Zhao's capstone project report for more details[17]

- Incorporating more widgets and templates.

In conclusion, through all the steps shown in the two reports, we created a new online trade website named EasyLife. However, more effort needs to be put in the future in order to make it a business product.

Appendix A

Source code of searching engine implementation

The following sections include the source code of the implementation of search engine in EasyLife.

A.1 IndexManager.java

```
package search;
import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.analysis.standard.*;
import org.apache.lucene.document.*;
import org.apache.lucene.queryParser.*;
import org.apache.lucene.search.*;
import org.apache.lucene.store.*;
import org.apache.lucene.util.*;
import org.apache.lucene.index.*;

public class IndexManager {
        static Connection conn;
        static IndexWriter indexWriter;
        static Directory dirIndex = new RAMDirectory();
        public static Directory getDirIndex() {
                return dirIndex;
        }
        public void setDirIndex(Directory dirIndex) {
                this.dirIndex = dirIndex;
        }
        protected IndexManager(){
        }
        public static IndexManager instance;

        public static void createInstance(){
                instance = new IndexManager();
        }
        public static IndexManager getInstance(){
                return instance;
        }
        public static void createIndex() throws IOException, Exception{
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            String url = "jdbc:mysql://localhost/test";
            conn = DriverManager.getConnection(url, "spring", "spring");
            StandardAnalyzer analyzer =
                                    new StandardAnalyzer(Version.LUCENE_30);
            indexWriter = new IndexWriter(dirIndex,analyzer,true,
                                    IndexWriter.MaxFieldLength.UNLIMITED);
            Statement stmt = conn.createStatement();
            ResultSet rset = stmt.executeQuery("select␣*␣from␣pages");
            rset.beforeFirst();
            while(rset.next()){
                Document doc = new Document();
                doc.add(new Field("title",rset.getString("Web_Name"),
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("web_add",rset.getString("Web_Address"),
                        Field.Store.YES, Field.Index.NOT_ANALYZED_NO_NORMS));
                doc.add(new Field("add_country",rset.getString("ADD_Country")
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("title",rset.getString("ADD_Country"),
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("add_province",rset.getString("ADD_Province
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("title",rset.getString("ADD_Province"),
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("add_city",rset.getString("ADD_City"),
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("title",rset.getString("ADD_City"),
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("postalcode",rset.getString("PostalCode"),
                        Field.Store.YES, Field.Index.NOT_ANALYZED_NO_NORMS));
                doc.add(new Field("category",rset.getString("Category"),
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("title",rset.getString("Category"),
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("owner",rset.getString("Owner"),
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("title",rset.getString("Owner"),
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("title",rset.getString("Text"),
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("page_Pro",rset.getString("Page_Property"),
                        Field.Store.YES, Field.Index.NOT_ANALYZED_NO_NORMS));
                doc.add(new Field("description",rset.getString("Description")
                        Field.Store.YES, Field.Index.ANALYZED));
                doc.add(new Field("title",rset.getString("Description"),
                        Field.Store.YES, Field.Index.ANALYZED));
                indexWriter.addDocument(doc);
            }
            indexWriter.optimize();
            indexWriter.close();
        } catch (InstantiationException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (IllegalAccessException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (Exception e){
                e.printStackTrace();
        }
        }
        public static void main(String[] args) throws IOException, Exception{
                IndexManager.createInstance();
                IndexManager indexManager = IndexManager.getInstance();
                indexManager.createIndex();
        }
}
```

Program A.1: IndexManager.java

A.2 IndexManager.java

```
package search;
import java.io.File;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.analysis.standard.*;
import org.apache.lucene.document.*;
import org.apache.lucene.queryParser.*;
import org.apache.lucene.search.*;
import org.apache.lucene.store.*;
import org.apache.lucene.util.*;
import org.apache.lucene.index.*;


public class SearchManager {
    public ArrayList<SearchResult> search(String keyWord)throws IOException{
        try{
         ArrayList<SearchResult> result;
         // The result consists of arraylist, including webname, webadd, .etc.
```

```
result = new ArrayList<SearchResult>();
StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_30);

//2.parse query
String querystr = keyWord;
Query q_title = new QueryParser(
        Version.LUCENE_30,"title",analyzer).parse(querystr);
Query q_web_add = new QueryParser(
        Version.LUCENE_30,"web_add",analyzer).parse(querystr);
Query q_add_country = new QueryParser(
        Version.LUCENE_30,"add_country",analyzer).parse(querystr);
Query q_add_province = new QueryParser(
        Version.LUCENE_30,"add_province",analyzer).parse(querystr);
Query q_add_city = new QueryParser(
        Version.LUCENE_30,"add_city",analyzer).parse(querystr);
Query q_postalcode = new QueryParser(
        Version.LUCENE_30,"postalcode",analyzer).parse(querystr);
Query q_category = new QueryParser(
        Version.LUCENE_30,"category",analyzer).parse(querystr);
Query q_owner = new QueryParser(
        Version.LUCENE_30,"owner",analyzer).parse(querystr);
Query q_description = new QueryParser(
        Version.LUCENE_30,"content",analyzer).parse(querystr);

//3. search
int hitsPerpage=10;

IndexManager.createIndex();
IndexSearcher searcher =
        new IndexSearcher(IndexManager.getDirIndex(),true);
TopScoreDocCollector collector =
        TopScoreDocCollector.create(hitsPerpage,true);
searcher.search(q_title,collector);
```

```
ScoreDoc[] hits = collector.topDocs().scoreDocs;

//4. display results

System.out.println("Found " + hits.length + " webpages:");
ArrayList<String> webName = new ArrayList<String>();
ArrayList<String> webAdd = new ArrayList<String>();
ArrayList<String> addCountry = new ArrayList<String>();
ArrayList<String> addProvince = new ArrayList<String>();
ArrayList<String> addCity = new ArrayList<String>();
ArrayList<String> postalcode = new ArrayList<String>();
ArrayList<String> description = new ArrayList<String>();
for(int i=0;i<hits.length;++i) {
      int docId = hits[i].doc;
      Document d = searcher.doc(docId);
      SearchResult sr = new SearchResult();
      sr.setName(d.get("title"));
      sr.setWebAddress(d.get("web_add"));
      sr.setCountry(d.get("add_country"));
      sr.setProvince(d.get("add_province"));
      sr.setCity(d.get("add_city"));
      sr.setPostalCode(d.get("postalcode"));
      sr.setDescription(d.get("description"));
      result.add(sr);
    }
searcher.close();
return result;
}catch (Exception e){
                e.printStackTrace();
}
return null; // if failed, return null;
}
```

```
    public static void main(String[] args) throws IOException, Exception{
        }
}
```

Program A.2: SearchManager.java

A.3 SearchResult.java

```
package search;

public class SearchResult {
        public static final int FIELDS_COUNT = 7;
        protected String webName;
        protected String webAddress;
        protected String country;
        protected String province;
        protected String city;
        protected String postalCode;
        protected String description;
        public SearchResult(String name, String addr, String coun,
                        String prov, String c, String po, String des){
                webName = name;
                webAddress = addr;
                country = coun;
                province = prov;
                city = c;
                postalCode = po;
                description = des;
        }
        public SearchResult(){
        }
        public String getName(){
            return webName;
    }
    public String getWebAddress(){
            return webAddress;
    }
    public String getCountry(){
            return country;
    }
    public String getProvince(){
            return province;
    }
    public String getCity(){
            return city;
    }
    public String getPostalCode(){
            return postalCode;
    }
    public String getDescription(){
            return description;
    }
    public void setName(String name){
            webName = name;
    }
    public void setWebAddress(String addr){
            webAddress = addr;
    }
    public void setCountry(String coun){
            country = coun;
    }
    public void setProvince(String prov){
            province = prov;
    }
    public void setCity(String c){
```

```
                city = c;
        }
        public void setPostalCode(String po){
                postalCode = po;
        }
        public void setDescription(String des){
                description = des;
        }
}
```

Program A.3: SearchResult.java

A.4 index.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="ISO-8859-1"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+
        "://"+request.getServerName()+":"+request.getServerPort()+path+"/";
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <base href="<%=basePath%>">
    <title>CWW Search Engine</title>
    <link rel="stylesheet" type="text/css" href="layout.css" />
    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
</head>
```

```
<body>
    <div id="pg">
        <div id="header">
            <div id="header_left">
                <table border="0">
                    <tr>
                        <td>
                            <h1>
                                LOGO</h1>
                        </td>
                        <td>
                            <p>
                                Choose your location </p>
                            <ul>
                                <li>
                                    <select >
                                        <option>Milk</option>
                                        <option>Coffee</option>
                                        <option>Tea</option>
                                    </select> Country</li>
                                <li>
                                    <select>
                                        <option>Milk</option>
                                        <option>Coffee</option>
                                        <option>Tea</option>
                                    </select> Province</li>
                                <li>
                                    <select>
                                        <option>Milk</option>
                                        <option>Coffee</option>
                                        <option>Tea</option>
                                    </select> City</li>
                              </ul>
                              </td>
                          </tr>
                      </table>
                  </div>
                  <div id="header_right">
                      <p>
                          <a><img src="button1.gif" height="25"/></a>
                              <a><img src="button1.gif" height="25"/></a> <a><img

src="button1.gif" height="25"/></a>
                      </p>
                      <form method="post" action="searchResult.jsp">
                      <p>
                          <input type="text" name="keyword" size="45" />
                          <input type="submit" value="Go" />
                      </p>
                      </form>
                  </div>
              </div>
              <div id="ad">
                  <div id="ad_left">
                      Web info, high light pages
                  </div>
                  <div id="ad_right">
                      <p>Advanced Search</p>
                      <p>Category</p>
                      <ul>
                                      <li>
                                          <select >
                                              <option>Milk</option>
                                              <option>Coffee</option>
                                              <option>Tea</option>
                                          </select> Country</li>
```

```
                                        <li>
                                            <select>
                                                <option>Milk</option>
                                                <option>Coffee</option>
                                                <option>Tea</option>
                                            </select> Province</li>
                                        <li>
                                            <select>
                                                <option>Milk</option>
                                                <option>Coffee</option>
                                                <option>Tea</option>
                                            </select> City</li>
                                      </ul>

                        <ul>
                                        <li>
                                            <select >
                                                <option>Milk</option>
                                                <option>Coffee</option>
                                                <option>Tea</option>
                                            </select> Country</li>
                                        <li>
                                            <select>
                                                <option>Milk</option>
                                                <option>Coffee</option>
                                                <option>Tea</option>
                                            </select> Province</li>
                                        <li>
                                            <select>
                                                <option>Milk</option>
                                                <option>Coffee</option>
                                                <option>Tea</option>
                                            </select> City</li>
                                        </ul>
                                        <div id="advanced_go"><input type="submit" valu

                    </div>
                </div>
                <div id="content">
                </div>
                <div id="footer">
                </div>
        </div>
</body>
</html>
```

Program A.4: index.jsp

A.5 layout.css

```
div#pg{
        width: 900px;
        height: 1000px;
        margin: 0 auto;
        }
div#header{
        float: left;
        margin-left:30px;
        width: 840px;
        height: 130px;
}
div#header_left{
        float: left;
        width: 320px;
        height: 130px;
        }
div#header_right{
        float: left;
        margin-left: 10px;
        width: 500px;
        height: 130px;
        }
div#ad{
        float: left;
        margin-left:30px;
        width: 840px;
        height: 290px;
}
div#ad_left{
        float: left;
        padding-left:5px;
        width: 555px;
        height: 290px;
        border: solid 1px grey;
        }
div#ad_right{
        float: left;
        margin-left: 5px;
        padding-left:5px;
        width: 265px;
        height: 290px;
        border: solid 1px grey;
        }
div#content{
        float: left;
        margin-top:10px;
        margin-left:30px;
        width: 840px;
        height: 560px;
        border: solid 1px grey;
        }
div#footer{
        float: left;
        margin-top:10px;
        margin-left:30px;
        width: 840px;
        height: 80px;
        border: solid 1px grey;
        }
ul{
list-style-type: none;
}
div#advanced_go{
margin-right: 20px;
float:right;
}
body{
font-family:"Calibri", Times, serif;
}
```

Program A.5: layout.css

A.6 searchResult.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="ISO-8859-1"%>
<%@ page language="java" import="search.*" %>
<%
        String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+
        ":"+request.getServerPort()+path+"/";
```

```
String keyword = request.getParameter("keyword");
SearchManager sManager = new SearchManager();
ArrayList<search.SearchResult> searchResults = sManager.search(keyword);
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD␣HTML␣4.01␣Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">
    <link rel="stylesheet" type="text/css" href="resultlayout.css" />
    <title>Search Result</title>
        <meta http-equiv="pragma" content="no-cache">
        <meta http-equiv="cache-control" content="no-cache">
        <meta http-equiv="expires" content="0">
        <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
        <meta http-equiv="description" content="This␣is␣my␣page">
        <!--
        <link rel="stylesheet" type="text/css" href="styles.css">
        -->
  </head>
  <body>
  <h1>Search Result</h1>
    <%
        for(int i = 0; i<searchResults.size(); i++){
                search.SearchResult sr = searchResults.get(i);
    %>
     <ul>
                <li><a href="http://<%=
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sr.getWebAddress()%>"><%=sr.getName() %></a></li>
                <li><%=sr.getCity()%>,
                <%=sr.getProvince()%>,
                <%=sr.getCountry()%> <%=sr.getPostalCode() %></li>
                <li><%=sr.getDescription() %></li>
                <li><%=sr.getWebAddress() %></li>
     </ul><br /><br />
     <%
         }
      %>
  </body>
</html>
```

Program A.6: searchResult.jsp

A.7 resultlayout.css

```
ul{
list-style-type: none;
}
```

Program A.7: resultlayout.css

Bibliography

[1] craigslist - company overview, 2008. http://www.hoovers.com/company/craigslist_inc/rtsjrki-1.html.

[2] Mysql vs. microsoft access, 2009. http://www.bestmysqlwebhosting.com/articles11.html.

[3] Mysql vs. ms sql, 2009. http://www.bestmysqlwebhosting.com/articles8.html.

[4] Mysql vs. oracle, 2009. http://www.bestmysqlwebhosting.com/articles10.html.

[5] Adobe fireworks - wikipedia, 2010. http://en.wikipedia.org/wiki/Adobe_Fireworks.

[6] Apache lucene - features, 2010. http://lucene.apache.org/java/docs/features.html.

[7] Berners-lee biography at the world wide web consortium, 2010. http://www.w3.org/People/Berners-Lee/Longer.html.

[8] *MySQL 5.1 Reference Manual*, 2010.

[9] Allen H. Dutoit Bernd Bruegge. *Object-Oriented Software Engineering, Using UML, Patterns, and Java, Second Edition.* Pearson Education, Inc., 2004.

[10] Christopher Heng. 3 ways of putting a search engine on your website, 2009. http://www.thesitewizard.com/archive/searchengine.shtml.

[11] Kishorekumar. Uml diagrams model of an accommodation online in hotels of particular place, 2002.

[12] Rachna. Cool new features in adobe fireworks cs4. http://www.entheosweb.com/fireworks/CS4/cool_new_features.asp.

[13] Arjen Lentz Robin Schumacher. Dispelling the myths. http://dev.mysql.com/tech-resources/articles/dispelling-the-myths.html.

[14] Peter Pin shan Chen. The entity-relationship model: Toward a unified view of data, 1976. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.123.1085.

[15] Kelvin Tan. Lucene tutorial.com - basic concepts, 2004. http://www.lucenetutorial.com/basic-concepts.html.

[16] Tommi West. Design learning guide for fireworks: Using smart guides and tooltips for precise positioning and layout, 2009. http://www.adobe.com/devnet/fireworks/learning_guide/design/design_guide_pt4.html.

[17] Kefu Zhao. Easylife - online community trading system: Easyweb server, rich client designer and viewer implementation. Technical report, Simon Fraser University, 2010.

Index