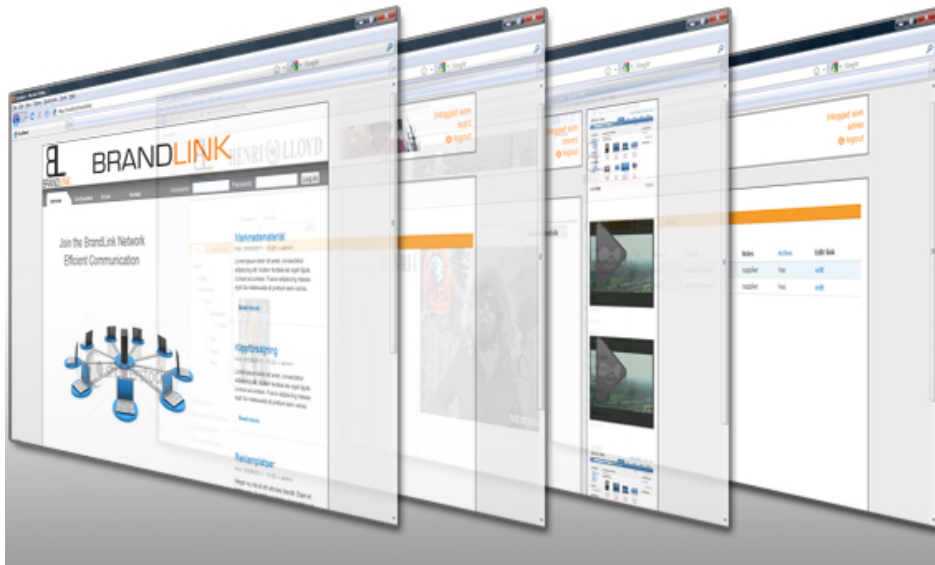


Brandlink B2B



Brandlink Business-to-Business E-commerce web Solution

Master of Science Thesis

MUHAMMAD ALI NASIR JANJUA
MUHAMMAD ARIF

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2013

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Brandlink Business to Business E-commerce web Solution

Master of Science Thesis

Host organization: Speedment AB Kilsgatan 10 Post Code 411 04 Göteborg, Sweden
Tel: 0702 - 69 24 01

Supervisor: Per-Åke Minborg
Chief Technical Officer (CTO), Speedment AB Göteborg
Sweden (<http://speedment.com>).
Telephone +46 702 69 24 02
Email: minborg@speedment.com

Examiner: Graham Kemp
Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone +46-(0)31 772 54 11
Email: kemp@chalmers.se

Muhammad Ali Nasir Janjua
Muhammad Arif

© Muhammad Ali Nasir Janjua, June 2013.
© Muhammad Arif, June 2013.

Department of Computer Science and Engineering
Göteborg, Sweden June 2013

ABSTRACT

Today, business is moving rapidly towards web technologies due to their portability, efficiency and multi-platform compatibility. Web technologies are impacting the way of businesses and their management. It is very difficult to manage many business relationships manually or even using some desktop application because there are multiple and distributed users. Using web based systems it's easy to manage distant users and speed-up business transactions. This project has produced a web based system which aims to make the business transactions speedy, easy to execute, and reliable.

This report presents the importance of web based systems in modern world and provides the mechanism that how business-to-business e-commerce solutions work. Specifically the report discusses the Brandlink Business to Business e-commerce solution which is kind of a complete management system facilitating manufacturers/brands and sellers/retailers to do business transactions between them through a web based e-commerce system. It also addresses the important development phases, resulting outcomes, and technical description of different sections of the Brandlink B2B solution (supplier/brand, store/retailer, and admin).

A formalized specification of the requirements is developed keeping the documentation simple and clear. Testing of solution is performed at the component level and bugs appeared during testing are listed in this report. The end product is a complete web based B2B ecommerce system able to be deployed in real environment. The solution reduces the transaction processing time & cost, improves efficiency, and provides reliability compared to manual business transactions.

ACKNOWLEDGEMENTS

This work is done with Speedment AB at Gothenburg, Sweden. We are very grateful to **Carina Dreifeldt** Chief Executive Officer (CEO), Speedment for giving us this opportunity to work on this project.

Our very special gratitude goes to our supervisor **Per-Åke Minborg** Chief Technical Officer (CTO), Speedment who has always guided and supported us many times during our work. We are lucky enough that he has never hesitated in helping us in our practical work as well as in planning and management tasks. He has been guiding, helping and encouraging us throughout the time. Whenever failures in work depressed us, we always found him to share with.

We are also grateful to our examiner **Graham Kemp** Associate Professor, Chalmers University of Technology who has always helped us with the smile throughout our work despite of his very busy work schedule.

TABLE OF CONTENTS

<i>Abstract</i>	<i>III</i>
<i>Acknowledgements</i>	<i>IV</i>
1. <i>chapter 1: Introduction & Objectives</i>	<i>I</i>
1.1. <i>Introduction to Brandlink</i>	<i>1</i>
1.2. <i>Introduction to Speedment</i>	<i>1</i>
1.3. <i>Problem</i>	<i>2</i>
1.4. <i>Introduction of the Project</i>	<i>2</i>
1.5. <i>Objectives</i>	<i>3</i>
2. <i>chapter 2: Tools & Technologies</i>	<i>5</i>
2.1. <i>Introduction</i>	<i>5</i>
2.2. <i>Web Application Architecture</i>	<i>6</i>
2.3. <i>Tier1</i>	<i>6</i>
2.4. <i>Tier2</i>	<i>7</i>
2.5. <i>Tier3</i>	<i>9</i>
2.6. <i>Supporting Software Tools</i>	<i>9</i>
3. <i>chapter 3: Requirement Gathering and Analysis</i>	<i>11</i>
3.1. <i>Introduction</i>	<i>11</i>
3.2. <i>Brandlink B2B E-commerce Solution requirements</i>	<i>11</i>
4. <i>chapter 4: Supplier Section Development</i>	<i>15</i>
4.1. <i>Introduction</i>	<i>15</i>
4.2. <i>Theme/GUI</i>	<i>15</i>
4.3. <i>Shopping Cart</i>	<i>19</i>
4.4. <i>Product</i>	<i>20</i>
4.5. <i>Product and category permissions</i>	<i>24</i>
4.6. <i>Private Catalog</i>	<i>25</i>
4.7. <i>Product Search</i>	<i>26</i>
4.8. <i>Summary</i>	<i>27</i>
5. <i>chapter 5: Store/ Retailer Section Development</i>	<i>28</i>
5.1. <i>Introduction</i>	<i>28</i>
5.2. <i>Theme/GUI</i>	<i>28</i>
5.3. <i>View Related Suppliers</i>	<i>29</i>
5.4. <i>Advertisements</i>	<i>30</i>
5.5. <i>Catalog</i>	<i>31</i>
5.6. <i>Product Details</i>	<i>32</i>
5.7. <i>Orders History page</i>	<i>32</i>
5.8. <i>Marketing Information</i>	<i>34</i>

5.9.	Summary.....	34
6.	<i>chapter 6: Brandlink Administrator Section Development.....</i>	<i>35</i>
6.1.	Introduction	35
6.2.	Theme/GUI.....	35
6.3.	Home Page	36
6.4.	Orders Page.....	38
6.5.	Statistics Reports	38
6.6.	Advertisements Page	40
6.7.	Relationships Page	41
6.8.	Email Section	42
6.9.	Summary.....	43
7.	<i>Deployment</i>	<i>44</i>
7.1.	Introduction	44
7.2.	Hardware architecture.....	44
7.3.	Installation of server applications.....	46
7.4.	Summary.....	46
8.	<i>Results and Future work</i>	<i>47</i>
8.1.	Introduction	47
8.2.	Results	47
8.3.	Future Work	47
9.	<i>Critical analysis.....</i>	<i>48</i>
9.1.	Introduction	48
9.2.	Implementation framework.....	48
9.3.	The good decision.....	49
9.4.	Major problems	49
9.5.	Learning	49
	<i>References.....</i>	<i>50</i>
	<i>Appendix A: SRS & Analysis.....</i>	<i>52</i>
	<i>Appendix B: Supplier Development.....</i>	<i>58</i>
	<i>Appendix C: Admin Development.....</i>	<i>65</i>
	<i>Appendix D: Deployment.....</i>	<i>69</i>
	<i>D.3 Application deployment</i>	<i>70</i>

LIST OF FIGURES

1. Figure 1.1: Brand link B2B ecommerce system	3
2. Figure 2.1: Web application.....	5
3. Figure 2.2: Three tier web architecture	6
4. Figure 2.3: Drupal Architecture	8
5. Figure 3.1: Supplier Pages.....	11
6. Figure 3.2: Store Pages.....	12
7. Figure 3.3: Admin Pages.....	13
8. Figure 4.1: Supplier section theme layout	15
9. Figure 4.2: Drupal list of blocks	15
10. Figure 4.3: Admin J-Query quick tabs user interface.....	16
10. Figure 4.4: View of supplier home page	17
11. Figure 4.5: Links below products catalog.....	18
12. Figure 4.6: Add product classes.....	19
13. Figure 4.7: JavaScript calculating seller commission	20
14. Figure 4.8: Attributes & options for products	21
15. Figure 4.9: Product discounts.....	22
16. Figure 4.10: Product publishing	22
17. Figure 4.11: Permissions with categories and sub-categories	24
18. Figure 4.12: Product catalog categories and sub-categories.....	25
19. Figure 4.13: Complete supplier section	26
20. Figure 5.1: Store side theme	27
21. Figure 5.2: Relationships block	29
22. Figure 5.3: Products page.....	31
23. Figure 5.4: Store's order history.....	32
24. Figure 5.5: Complete store side	34
25. Figure 6.1: Admin section layout	35
26. Figure 6.2: JQuery tabs to organize content.....	36
27. Figure 6.3: Drupal menus	36
28. Figure 6.4: Drupal Views.....	37
29. Figure 6.5: List of suppliers	37
30. Figure 6.6: List of stores	37
31. Figure 6.7: Orders page.....	38

32. Figure 6.8: <i>Drupal view for statistic reports</i>	39
33. Figure 6.9: <i>Statistics report</i>	40
34. Figure 6.10: <i>Relationships content type</i>	41
35. Figure 6.11: <i>Mass emailing user interface</i>	42
36. Figure 6.12: <i>Admin Section</i>	43
37. Figure 8.1: <i>Hardware architecture</i>	46

1. CHAPTER 1: INTRODUCTION & OBJECTIVES

1.1. Introduction to Brandlink

Brandlink AB is a third party who provides e-commerce services (Electronic commerce, commonly known as e-commerce, consists of the buying and selling products or services over electronic systems such as the internet and other computer networks). Brandlink AB needed a web-based e-commerce system to automate the manual business between different Suppliers/Brands and Stores/Retailers.

The features required in the project were to make every-day business activities between Suppliers/Brands and Stores/Retailers easy and reliable using automated computer systems. Administration section of the system was required with the capability of creating new Supplier or Store accounts and maintaining the system. As users are distant so it was a basic requirement to have a single system that can support all the remote users having different platform (Windows, Linux, MacOs, etc.) based client computers and make business-related transactions very easy. To develop the system Brandlink AB contacted Speedment AB.

1.2. Introduction to Speedment

Speedment AB is an IT specialist who solves problems (related to different domains) by providing desktop and web based computer applications. Speedment deals with following five major areas [1].

- Web Application Development.
- Database Acceleration and web hosting.
- Web development in Drupal content management.
- Mobile application development.
- IT Consulting.

1.3. Problem

There are two types of business entities that are involved in this problem statement: Suppliers/Manufacturers & Stores/Retailers. Suppliers manufacture one or more types of products that are associated with unique brand names, for example shoes, garments, etc. Stores are the shops in the market that sell one or more types of brands. One Supplier can have business relationships with many Stores to sell products and Stores purchases different brands from different Suppliers.

To get the latest information about some product, a store either gets some printed media or visits Supplier's website. An order is placed via email, phone call, or by meeting Supplier. A supplier uses some spreadsheets or applications to keep a record of products, orders, and business relationships with Stores. A business relationship is just a group of different agreements and parameters, for example: A supplier offers different discount rates to each store.

Brandlink AB analyzed that there was a problem for Stores to access each Supplier's catalog by either visiting websites or by using printed media. Both, Suppliers & Stores were keeping record separately for each business relationship. The most difficult task that they were doing was to maintain many to many business relationships.

After analysis, Brandlink AB decided to provide such an e-commerce solution that will allow Suppliers & Stores to connect with each other using a single platform. Stores should be able to select a list of products from Supplier's catalog and to place an order. The system will keep the record of each business activity.

1.4. Introduction of the Project

The project aims to provide a complete web-based e-commerce solution and an administration section. The E-commerce systems are categorized into two main categories.

- B2B (Business-to-Business).
- B2C (Business-to-Consumer).

“**Business-to-business (B2B)** describes commerce transactions between businesses, such as between a manufacturer and a wholesaler, or between a wholesaler and a retailer” [2]. “The term ‘business-to-business’ was originally coined to describe the electronic communications between businesses or enterprises in order to distinguish it from the communications between businesses and consumers (B2C)” [2]. The main difference between these two types of e-commerce systems is that B2B includes multiple sellers and buyers while B2C has one seller and many buyers.

In Brandlink project, Suppliers and Stores are two types of business. There can be multiple sellers (Suppliers) and multiple buyers (Stores). Each Supplier can be a brand that manufactures different kind of products and each Store buys different brands to sell in retail. This shows that Business-to-Business (B2B) e-commerce solution is more suitable for Brandlink which led us to build a B2B ecommerce system for Brandlink.

Brandlink Business-to-Business E-commerce solution will provide a complete management system that allows suppliers and stores to do business transactions through a web interface. Each supplier will have a personal catalog (consisting of categories, sub categories and products added by supplier) and rights to administer its catalog. To access the catalog of a particular supplier and to make an order, stores will need a relationship with a supplier. The relationship between a supplier and a store will ensure that store has access to the right supplier, i.e. a store selling shoes should not see the supplier manufacturing electronic devices.

Brandlink administration will be a super user of the system and will have rights to add new users (suppliers and stores) and manage the relationships between suppliers and stores.

1.5. Objectives

Objective of business to business e-commerce system to facilitate suppliers and stores with a system with interactive web based user interface that can help to perform the business transactions over internet and keep a record of all activates. Figure 1.2 shows an abstract view of the system and how different users will be interacting with different functionalities to complete their business objectives through a totally automated system.

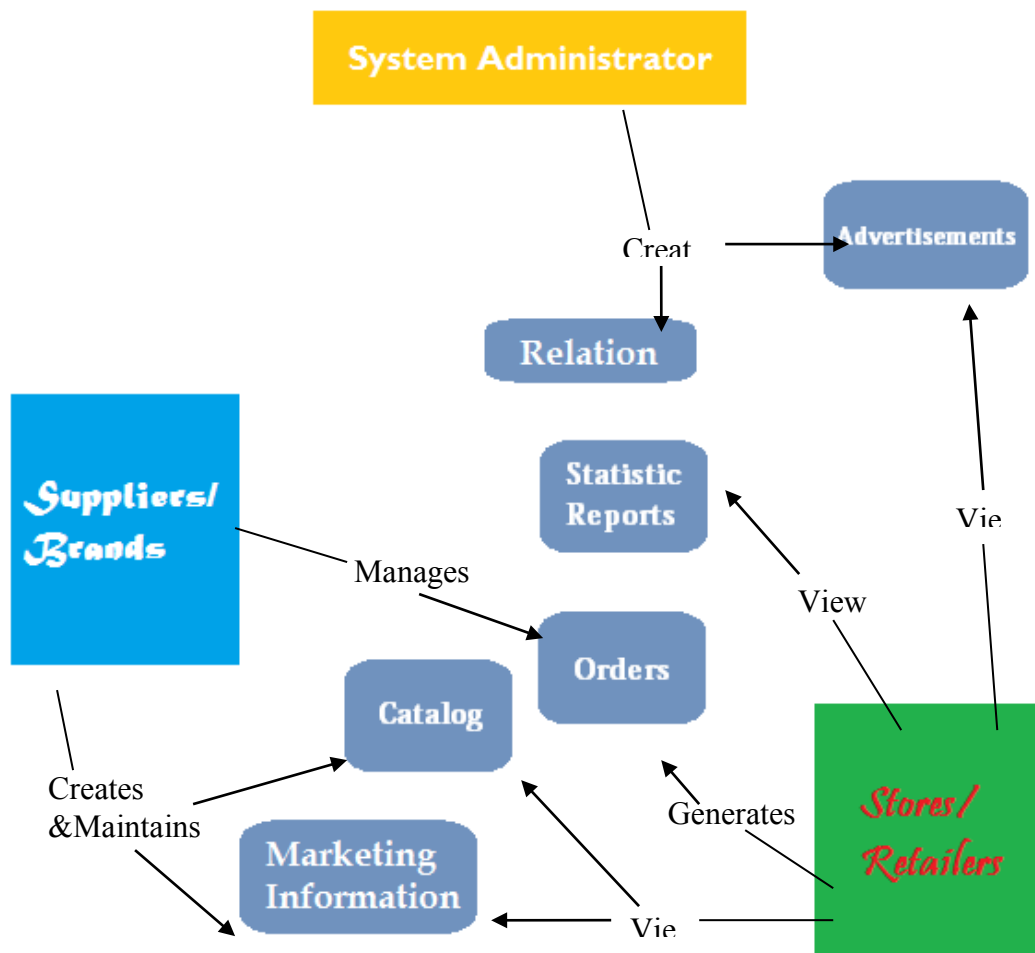


Figure 1.2: Brand link B2B e-commerce system

Figure 1.2: Description

System Administrator (Brandlink)

System administrator with super user rights (as indicated by wide arrow) will be able to

- Add, edit, and delete supplier and store accounts.
- Manage the relationships between suppliers and stores.
- View and Search Orders.
- Access Statistics reports.
- Create image and video advertisements shown on store side.
- Send email to selected users.

Suppliers/Brands

Users with supplier role will have rights to do following transactions.

- Create and maintain their catalog.
- Add new products.
- Change the product permission settings.
- Manage discounts applied to categories or products.
- Track and fulfill their order.
- Monitor their sales reports.
- Add marketing information for their stores.
- Change own account information.

Stores/Customers

Stores will have rights to do following transactions.

- View their related supplier accounts.
- Navigate the supplier's catalog.
- Create their cart and place orders.
- View their previous orders.
- View and download marketing information provided by supplier.
- View scrolling advertisements created by admin and play videos directly.
- Change own account information.

2. CHAPTER 2: TOOLS & TECHNOLOGIES

2.1. Introduction

Software tools and technologies are applications and some standards that help to build some modern application and provide a way to solve some problem. As web application development is a part of this thesis project, it is important to understand about web technology. A web application is an application that executes on a web server [3]. A User interacts by sending the web request over network (the internet) by using some web browser. The Web server receives the request and generates a response by executing a web application. The response is sent it back to the client (user) in a Hyper Text Markup Language (HTML) format that is that is rendered by the web browser. All requests and responses are sent using Hyper Text Transfer Protocol (HTTP).

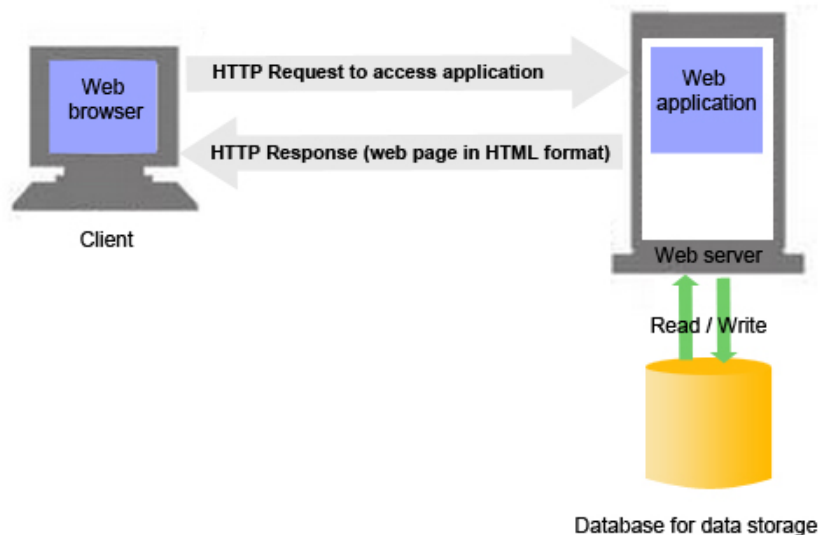


Figure 2.1: Web application, adapted from [3]

These are some benefits of using web applications,

- Distributed application and users from different locations can access it at the same time.
- Cross platform compatibility. Users having different operating systems (Windows, Linux, Mac, etc.) can use it.

- It requires a thin client. Only a web browser is required to run the web based application.

2.2. Web Application Architecture

Web applications are usually divided into different logical components that can be placed on one or multiple machines distributed over a network [4]. Distribution of components over hardware depends on work load on each component. Each logical component is called a “Tier”.

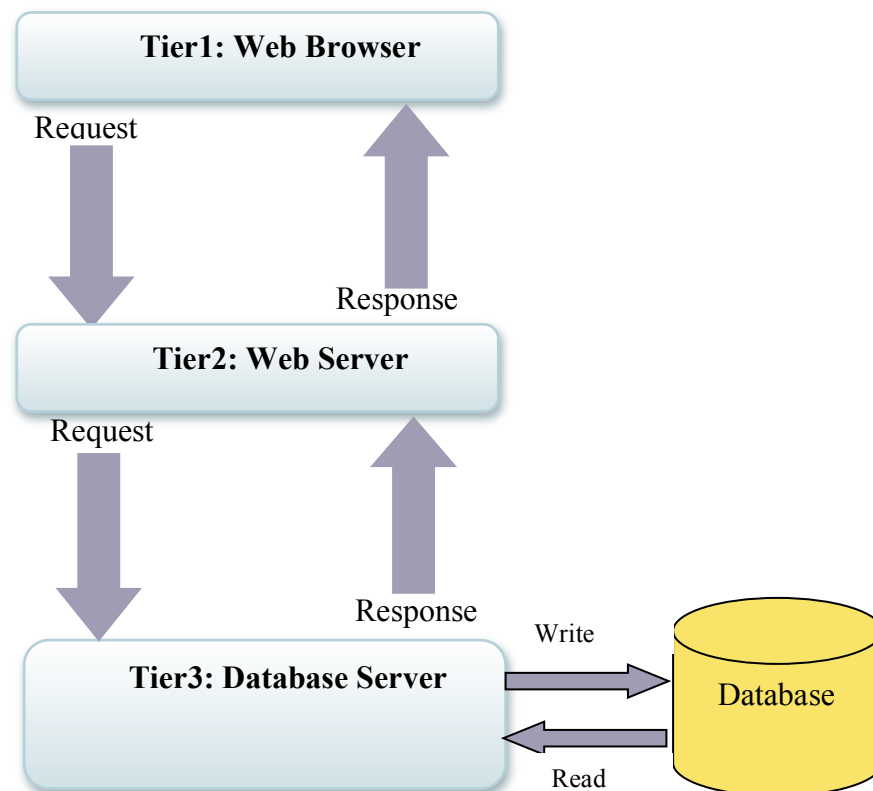


Figure 2.2: Three tier web architecture

2.3. Tier1

Tier1 is also called presentation layer. This layer is implemented as client machine. A web browser is used to render the graphical user interface that allows the user to interact with a web application. Different web browsers are used on the client side like Internet Explorer,

Mozilla Firefox, Safari, etc. Scripting languages are used to render the user interface in a web browser. HTML, JavaScript, and CSS are used in this project as a client side scripting languages.

2.3.1. HTML

“Hyper Text Markup Language (HTML) is written in the form of elements consisting of tags, enclosed in angle brackets (like <html>), within the web page content” [5]. An HTML File is either written by author of the website or can be generated dynamically by a web server and sent to a client after the web browser’s request. Web browser interprets the script and composes into visual content (pages).

2.3.2. CSS

“Web browsers can also refer to Cascading Style Sheet (CSS) to define the appearance and layout of text and other material”[6]. CSS defines the presentation of a web page or complete website. It can be written as a separate document or with HTML, called embedded CSS. Benefit of using CSS is to increase the presentation power of HTML and to isolate the layout from content.

2.3.3. JavaScript

JavaScript is also a client side scripting language that defines the behavior of HTML pages. For example: navigation, data validation before submit, and calculations on a client side. It can be written in a different file or can be embedded in the webpage.

2.4. Tier2

Tier2 is called the application layer that consists of a web server. A web server is an application that executes some web application [7]. It listens to HTTP requests from clients and replies after receiving content from hosted web application. There are different types of web server applications like Apache HTTP, Apache Tomcat, IIS, and Oracle Web Tier. Each web server supports one or more types of scripting languages and databases.

2.4.1. Apache HTTP server

Apache is the most popular open source web server software that can be installed on almost all operating systems, including Linux, UNIX, and Windows. These are some benefits of using Apache.

- It is supported by multiple operating systems.
- It is a freely available & open-source application.
- Easy to install and configure.
- Supports PHP scripting language.

2.4.2. PHP

PHP is a scripting language that executes on a web server to generate dynamic content like HTML, CSS, and JavaScript. It is an Object Oriented language [8]. One can either embed it in HTML code (using `<?php` and `?>` tags) or can write separate documents. There are a lot of content management systems, modules, and scripts which are freely available that are written in PHP and can be a starting point to build a complex web application. Due to this benefit, PHP language is used in this project for web application development.

2.4.3. Drupal

Drupal is an open-source content management system that is written in PHP. It is powered by its community all over the world [9]. It is fully compatible with all web servers that support PHP and database. It has a variety of modules and themes freely available over internet. All the modules and themes are customizable.

Drupal provides a web-based User Interface that is used to configure, customize, and manage the website content. Figure 2.3 describes the architecture of Drupal Content Management System (CMS). Core is a bundle of basic functionalities that makes Drupal a Content Management System. It can be easily customized by adding modules (features written by the Drupal community), themes (different styles for graphical user interface), and translation files. Drupal modules interact with Drupal by using hooks. Hooks are interfaces that activate on some event.

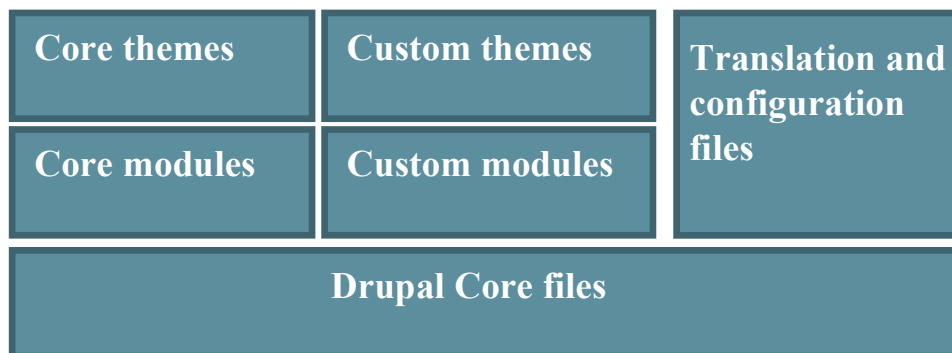


Figure 2.3: Drupal Architecture

Drupal requires a database server to store different records. During Drupal installation, it creates a database structure in selected database that is used to store site's dynamic content and configurations. Variety of database servers can be used with Drupal. MySQL is the most commonly used database server with PHP and Drupal.

It is very time consuming to implement such a complex e-commerce business to business application from scratch. There are a lot of content management systems that are freely

available. Drupal was selected for development in this project because of its following features,

- Customizable & Scalable.
- Most stable content management system.
- Written in PHP, works on many platforms like Apache, IIS.
- Open source.
- Active community.
- Modules & themes are freely available.
- Provides interfaces (hooks) to implement & attach a new functionality.

Drupal has released version 7.0, but it doesn't have enough modules and support to build this complex system. Drupal 6 is the most stable version & has enough material to support this project. So we decided to use it.

2.4.4. CentOS

CentOS is Linux based operating system that is mostly used for web servers [10]. It is an open-source, free, and the most stable version of Linux. CentOS 5.4 version is used for this project.

2.5. Tier3

The third tier is a data storage layer. It is just a database management system. A web application stores all records on this layer. MySQL Server is used in this project because it is very stable and compatible with Drupal.

2.5.1. MySQL

MySQL is a relational database management system (DBMS) [11]. It is most popular (for web applications), free, and compatible with Apache web server. It uses Structured Query Language (SQL) to manipulate databases. MySQL Server 5.0 is used to store application's content & configurations.

2.6. Supporting Software Tools

Different tools are used for this project development,

2.6.1. Adobe Dreamweaver

Adobe Dreamweaver CS 4 is an editor for scripting languages like HTML, CSS, and JavaScript. It visualizes scripts in the easy and readable format.

2.6.2. Adobe Photoshop

Adobe Photoshop CS 5 is graphics designing software. It is used to create themes (layout of graphical user interface), icons and graphics for web applications.

2.6.3. Firebug

Firebug 1.6.2 is a debugging tool that helps a developer to debug problems related to HTML, CSS, and JavaScript. It is an Add-On that works with Firefox web browser. It is also available for other browsers like Google Chrome & Internet Explorer.

2.6.4. PuTTY

PuTTY 0.60 is a free and open-source application that is used to establish SSH connections with remote machines (computers). It is also available in the portable format that can be used directly without installation [12].

3. CHAPTER 3: REQUIREMENT GATHERING AND ANALYSIS

3.1. Introduction

In this chapter focus will be to define the Software Requirement Specification (SRS) document, discuss the purpose of SRS document, specify tangible benefits of the end product, and describe the different sections of the solution. The requirements gathered, analyzed and finalized for the Brandlink B2B e-commerce project are listed in the user manual (Appendix A).

To finalize requirements reading of a document made by Brandlink client and researching & navigating different existing ecommerce systems helped a lot.

3.2. Brandlink B2B E-commerce Solution requirements

As there are three major sections of the Project, Supplier, Store, and Admin section so the requirement gathering process splits into these three sections.

The Supplier user accounts are companies that manufacture brands like garments, shoes, cell phones, etc. The suppliers manufacture and transport their products to different stores to approach potential customers. The Brandlink B2B e-commerce solution aims to facilitate suppliers to do their transactions with stores. Figure 3.1 shows different pages and functions on those pages for supplier users. A complete description of supplier section requirements is listed in table 1 in Appendix A.

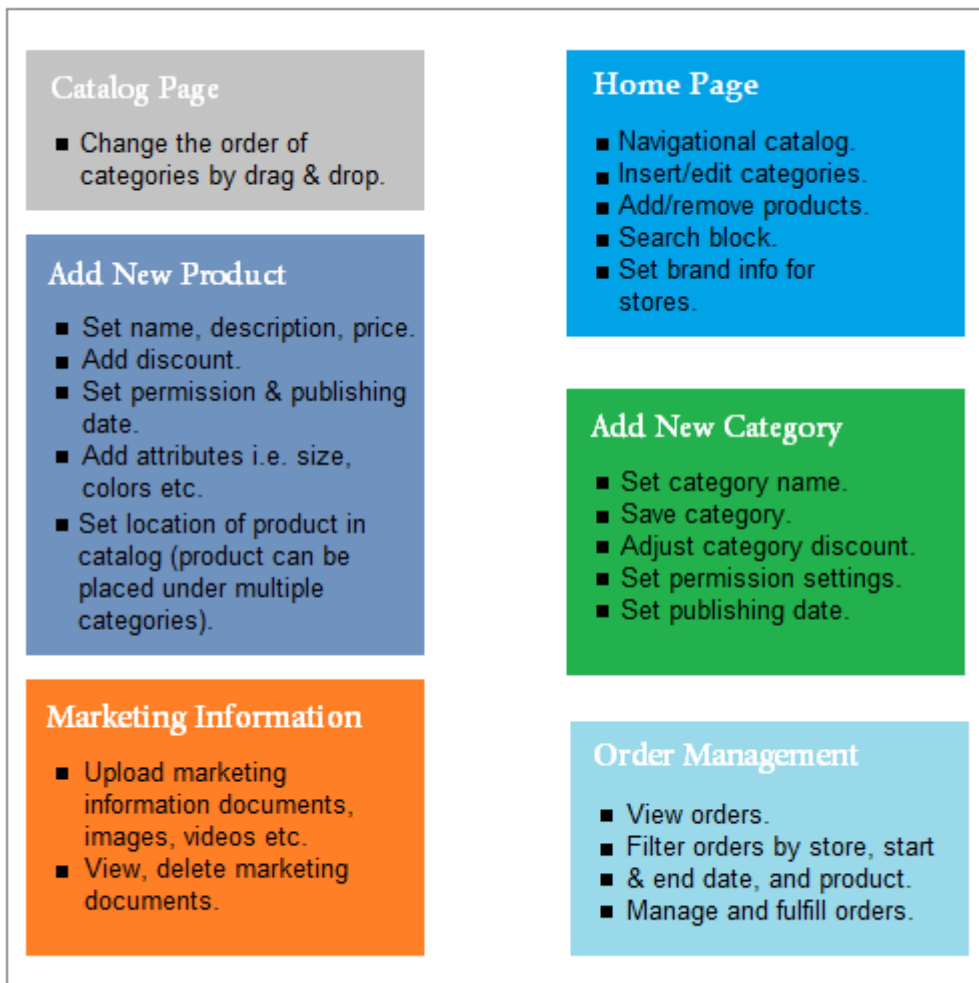


Figure 3.1: Supplier Pages

The Store user accounts are retailers who sell different brands to customers. Figure 3.2 shows different pages and functions on those pages on the store side. A complete description of store side requirements is listed in table 2 in Appendix A.

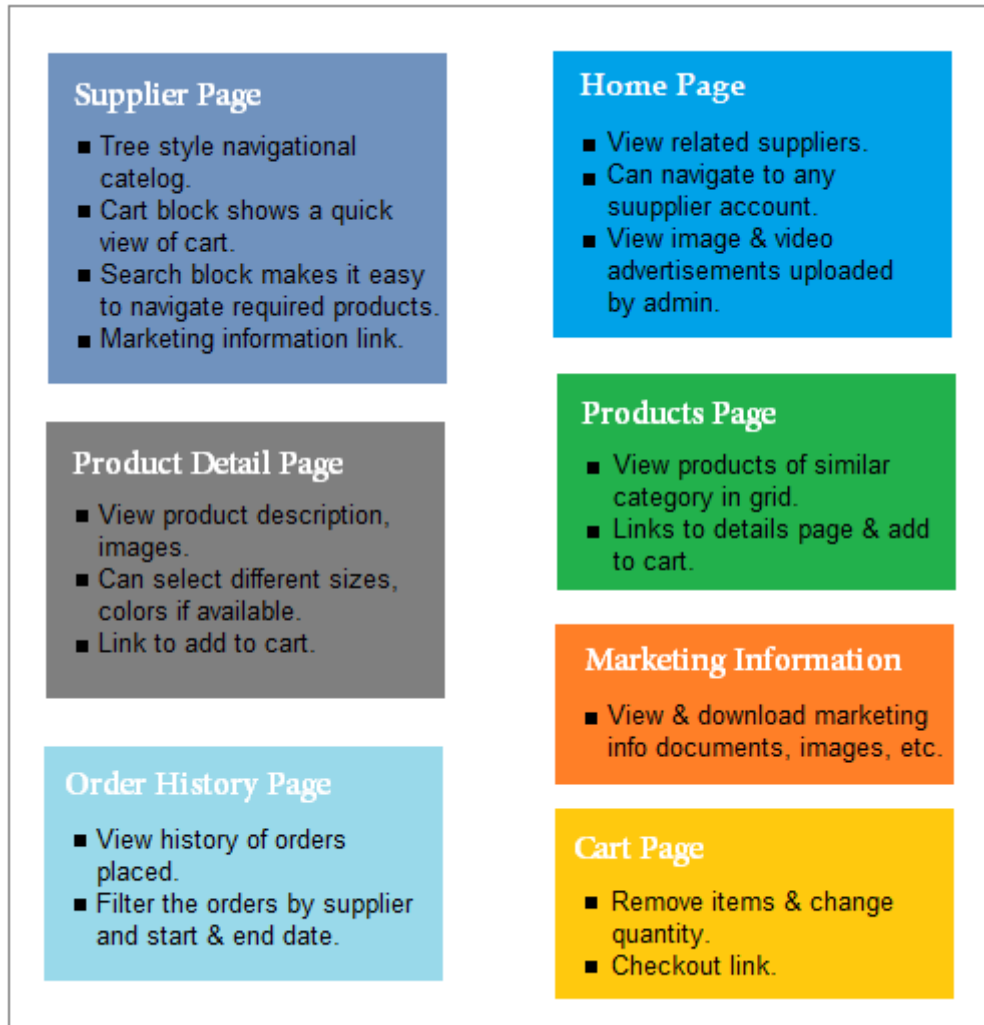


Figure 3.2: Store Pages

The Admin is a super user in Brandlink B2B system. Admin can create new store and supplier user accounts. It can also create relationships between them. The admin will be able to view the different statistical reports, send bulk emails to all or specific users, and create advertisements. Figure 3.3 shows different pages and functions on those pages on admin part of the site. A complete description of admin part requirements is listed in table 3 in Appendix A.

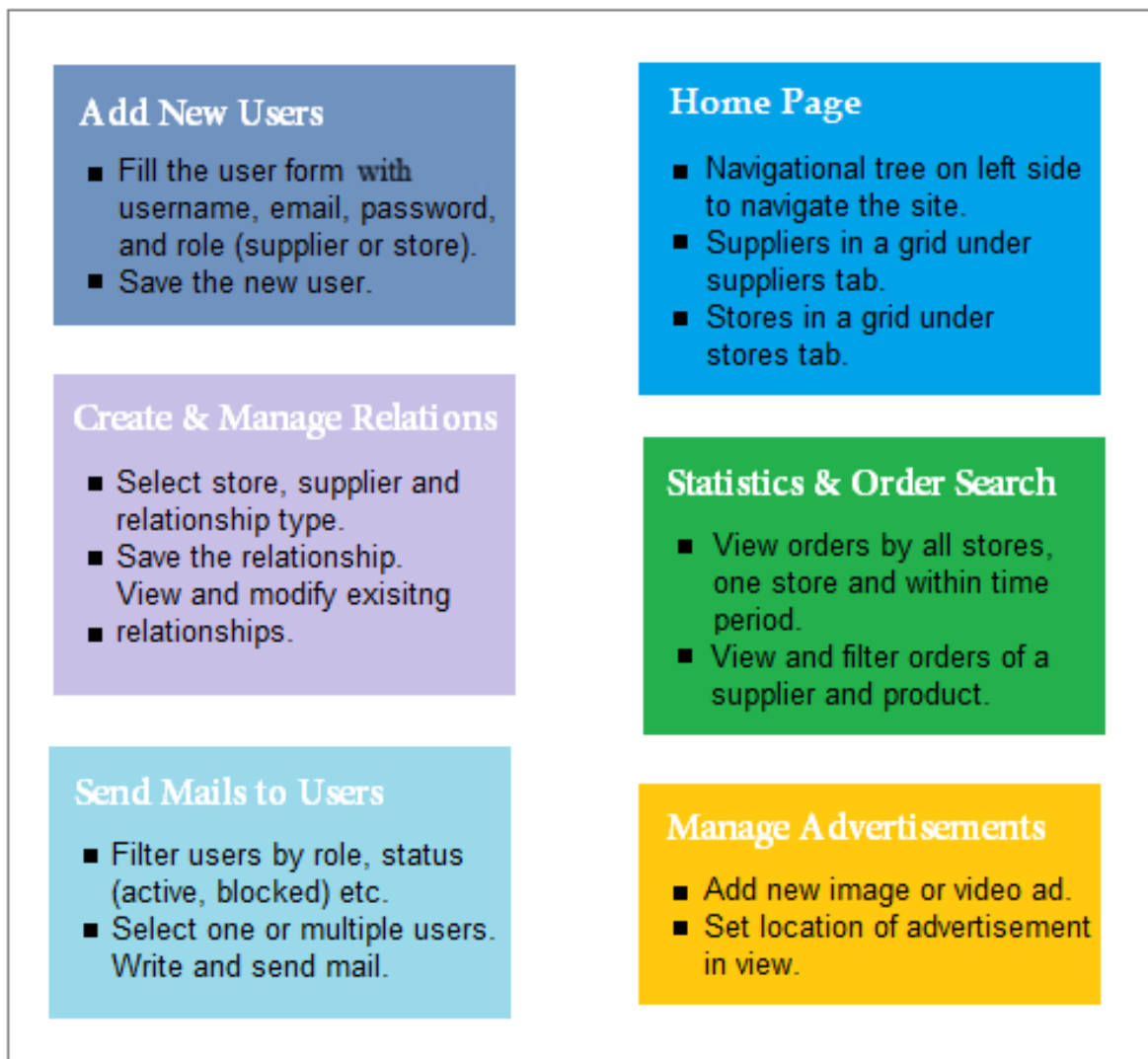


Figure 3.3: Admin Pages

4. CHAPTER 4: SUPPLIER SECTION DEVELOPMENT

4.1. Introduction

Supplier section is a graphical user interface that lets the supplier (role) to create and organize content related to their business. Supplier users are redirected to this interface after login where they can perform following operations,

- Create/Edit profile, which contains logo, and page that will appear to stores.
- Create and edit categories & products.
- Apply permissions and publication conditions.
- Create discount packages for categories and products.
- Upload and edit marketing information files (images, video clips) related to their products.
- Can generate sale and order reports (statistics).

Development of supplier's section consists of two main parts. Theme (the graphical user interface) and Functionality. Shopping cart is the main feature in these sections. Shopping cart is a set of functions that allows online customers to select different products, accumulate a list, calculates all the values, charges, and taxes and let the customer check-out. Development of the shopping cart is further divided into different parts. Product, categories & sub-categories, and discounts.

There are some other important features that are developed, like private catalog, product & category permissions, and discounts. Private catalog and product permission are very important features that allow suppliers to build personal product catalog and apply different permissions on categories and sub-categories.

4.2. Theme/GUI

This section describes the front-end part of supplier's section. This section consists of three main graphical sections: header, left content, and right content. In order to create new sections in Drupal theme (defines layout for graphical user interface), one should register those blocks in theme information file and define an HTML element DIV in the specific part of theme where space is required. Figure 4.1 describes how the supplier interface is divided into three parts for content management.

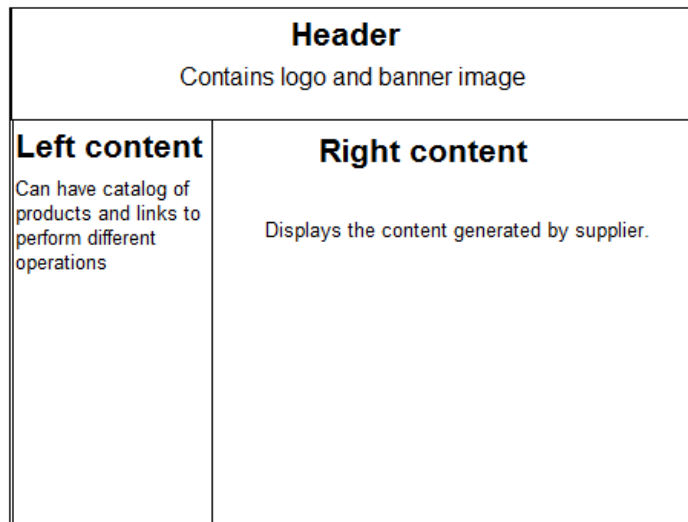


Figure 4.1: Supplier section theme layout

To start with the theme in Drupal, any basic theme can be modified to achieve the desired layout. For this section, “Brandlink” theme is used as a starting point because it looks similar to required layout and takes less effort to modify. Login and navigational blocks were removed from the base theme in “page.tpl.php” file to make it much simple. To display supplier’s banner image in header section, there should be some block registered in Drupal. New block called “header region” was introduced in theme information file, and its position was set in template file. Line 2 and 6 in figure 4.2 shows how to register a section and define its position in web-page.

```

1 Supplier.info
2     regions[header] = Header region
3
4 page.tpl.php
5     <div id="header-region" class="clear-block">
6         <?php print $header ?>
7     </div>

```

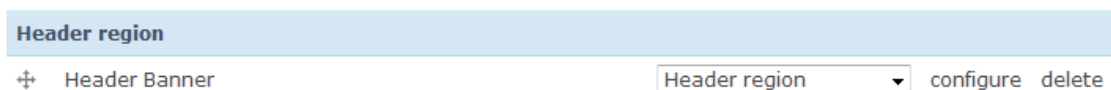


Figure 4.2: Drupal list of blocks (Admin-→ Site Building-→ Blocks)

A Drupal block can render any piece of a script. Either it is written directly or generated by Drupal View. Drupal view is a graphical user interface to generate content blocks. A Drupal view was created that fetches supplier logo from its profile. Here is Drupal generated code for “Drupal View”. Line 6 filters out only logo type content and related to current logged in user.


```

1 SELECT node.nid AS nid,
2   node.language AS node_language,
3   node.title AS node_title,
4   node.type AS node_type
5 FROM node JOIN users ON node.uid = users.uid
6 WHERE (node.type in ('logo')) AND (users.uid = ***CURRENT_USER***)

```

As it is showing in *figure 4.1*, content section consists of two parts. Each content section is a floating panel. Panels are used to manage the content and to make the GUI interactive. Drupal doesn't have any built-in function to generate tabbed panels like desktop applications. "jQuery Quick Tabs" [13] is a module that can be installed to achieve this functionality.

To place tabs, "jQuery Tabs" [13] module is used. It creates tabs and in each tab, one can put block or page.

The Quick Tabs module allows you to create blocks of tabbed content. Clicking on the tabs makes the corresponding content display instantly (it uses jQuery). The content for each tabbed section can be either a node, view, block or another quicktab. It is an ideal way to do something like the Most Popular / Most Emailed stories tabs you see on many news websites.

Once created, the quicktab block show up in your block listing, ready to be configured and enabled like other blocks. Multiple quicktabs can be placed on a single page.

Visit the [configuration page](#) to see the available styles and select the default style for your quicktabs.

Quicktab	Operations		
admin left side	Edit	Clone	Delete
content tabs	Edit	Clone	Delete
Menu	Edit	Clone	Delete
suppliers-stores	Edit	Clone	Delete

Figure 4.3: Admin J-Query quick tabs user interface

Left content contains supplier's product catalog and other information like statistics. Two tabs are created that contains each content category. Left content block is designed using the same method as header part by registering block in information file and theme file.

Product is the most important content in this section. Supplier can add hundreds of products. There must be some searching interface to search desired product by id or name. To display search box, new block was placed using the Drupal information and theme file. Some Cascading Style Sheet script was written to position the search block on the top of main content. Line 2 and 6 shows the registration of search block in Drupal theme and information file.

```

1 Supplier.info
2   regions[content_top] = content_top
3
4 page.tpl.php
5   <?php if ($content_top): ?>
6     <div id="content_top">
7       <?php print $content_top ?>
8     </div>
9   <?php endif; ?>

```



Figure 4.4: View of supplier home page

To make the user interface simple, some hyper-links were placed to perform operations related to products and different parameters like discounts, publishing, etc. As these links are important for all suppliers so template file is best to design and display the shared functionality. The following piece of code can be written in file “page.tpl.php” to achieve dynamic links for each supplier.

```

1 <div id="sidebar-left" class="sidebar">
2   <?php print $left ?>
3   <div id="supplier-links">
4     <ul>
5       <li><a href="<?php global $base_url; print $base_url;?>
6         /admin/content/taxonomy/<?php echo variable_get('uc_catalog_vid','1') ?>
7         /add/term">Infoga Gren</a></li>
8       <li><a href="<?php global $base_url; print $base_url;?>
9         /admin/content/taxonomy/<?php echo variable_get('uc_catalog_vid','1') ?>
10        ">Redigera/Ta bort Gren</a></li>
11       <li><a href="<?php global $base_url; print $base_url;?>/node/add/c01">
12        L&auml;gg till Artikel</a></li>
13       <li><a href="<?php global $base_url; print $base_url;?>
14        /admin/store/uc_discounts">L&auml;gg till Rabatter</a></li>
15     </ul>
16   </div>
17 </div>

```

The code was written in “page.tpl.php” file to generate a list of links. From line 5 onward, the code shows how to generate different links for current logged in user.

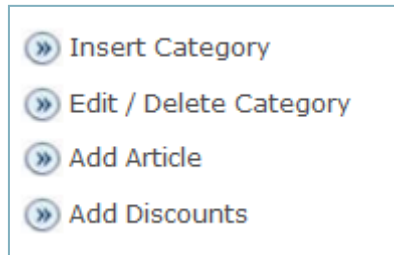


Figure 4.5: Links below products catalog

4.3. Shopping Cart

In e-commerce, a shopping cart is used to assist people making purchases online. This feature works exactly as a real cart in a shop to collect items. It gathers all the products that a customer wants to buy with their accurate quantities. “Upon checkout, the application typically calculates a total of the order, including shipping and handling (i.e. postage and packing) charges and the associated taxes, as possible” [14].

There are different modules for shopping cart that are freely available on the Internet. “Ubertcart” [15] is a bundle of modules that can be used with Drupal to achieve the basic functionality for shopping cart. Here is a list of some important modules related to this project,

- Product
- Catalog
- Cart
- Store
- Attribute
- Discounts

“Ubertcart” is a generic module for e-commerce systems. The following limitations were discovered after studying the implementation of this “Ubertcart.

- It supports only one supplier that can have many stores (one to many relationship).
- All products (relevant and irrelevant) appear to all the stores.
- Catalog is built to support one supplier. There is no discount on the product’s category level.

“Ubertcart” is easy to customize and extend because it is written in PHP. The following features were decided to implement in the existing module.

- Multiple suppliers.
- Private Catalog of products with categories and sub-categories.
- Product and catalog access permissions.

4.4. Product

Product is the main content type in this system. Each supplier creates and manages hundreds of products for stores. For this project, a product can have the following list of fields,

- Article number.
- Article name.
- Color code.
- Style.
- Size.
- Pictures (front & back).
- Cost.
- Retail price.
- Seller Commission.
- Information.
- Discount information.

Each product created by Ubertcart is a Drupal node. Each type of content in Drupal is called “node”. Drupal provides a built-in module called Content Construction Kit (CCK) [16]. CCK provides an interface that allows some user to attach custom fields (text fields, select boxes, etc.) to any content type. However, it doesn’t attach fields directly with “Product”. It means supplier needs to attach for each product. A simple solution for this problem is to create a class of product content type called “Article”.

New product class can be created using following path “Administer → Store Administration → Products → Manage Classes”.

The screenshot displays the 'Manage classes' interface. At the top, there is a table with the following data:

Class ID	Name	Description	Operations
c01	Artikel		edit delete

Below the table is the 'Add a class' form. It contains the following fields and instructions:

- Class ID: *** (text input field)
- The machine-readable name of this content type. This text will be used for constructing the URL of the *create content* page for this content type. This name may consist only of lowercase letters, numbers, and underscores. Dashes are not allowed. Underscores will be converted into dashes when constructing the URL of the *create content* page. This name must be unique to this content type.
- Class name: *** (text input field)
- Description:** (large text area)
- This text describes the content type created for this product class to administrators.
- Submit** (button)

Figure 4.6: Add product classes

CCK module can only attach simple text fields with “Article”. An “Article” can’t be described in a simple text form. It requires some other fields add pictures, attachments, and other attributes. So some helping modules were used with CCK to attach content fields.

4.4.1. Product Pictures

One product can have two image attachments that appear on the store side. CCK module requires some module that provides some image attachment functionality. “ImageField” [17] and “FileField” [18] modules can be installed to enable image field attachment. ImageField provides different parameters like maximum number of images, resizing of images, etc. Settings to image field can be accessed using following drupal path (Admin → Content Management → Article → Manage Fields → Field_Image_Cache).

4.4.2. Seller Commission

The seller commission field is already available so that a supplier can record the desired amount. The problem is that a seller has to calculate and record their commission for each product. Simple solution to this problem is a script implemented in JavaScript that listens to different events and calculates the seller commission automatically when cost or price fields are filled by some user. Line 3, 8, 14 and 20 are functions that listen to different events. For example the function blur() written on line 3 checks if the field “edit-list-price” is changed, calculate the sell price and fill into the “edit-sell-price” text field. Figure 4.7 shows the output of this script in text fields.

```
1 <?php
2     drupal_add_js('$ (document).ready(function () {
3         $('#edit-list-price').blur(function() {
4             var vv=$('#edit-list-price').val();
5             $('#edit-sell-price').val(((vv*$('#edit-cost').val())/100)+vv*1);
6             return false;
7         });
8         $('#edit-cost').blur(function() {
9             var vv=$('#edit-list-price').val();
10            $('#edit-sell-price').val(((vv*$('#edit-cost').val())/100)+vv*1);
11            $('#edit-cost').val(((vv*$('#edit-cost').val())/100));
12        return false;
13        });
14        $('#edit-cost').focus(function() {
15            var vv=$('#edit-sell-price').val();
16            $('#edit-cost').val(((vv-$('#edit-list-price').val()
17            )*100)/$('#edit-list-price').val());
18        });
19        $('#edit-sell-price').blur(function() {
20            var vv=$('#edit-sell-price').val();
21            $('#edit-cost').val(vv- $('#edit-list-price').val());
22            return false;
23        });
24    });
25    ', 'inline');
26 ?>
```

Product information

In pris: **Seller commission: *** % **Rek ut pris: ***

The listed MSRP. The amount you will be paid. Customer purchase price.

Product and its derivatives are shippable.

Figure 4.7: JavaScript calculating seller commission

This script was written in Drupal block and permission was set so that it can only execute for “add product” page.

4.4.3. Product attributes

In the real world, products are differentiated by their properties e.g. size, weight, etc. These properties are called product attributes. In any kind of business, these products have different identification numbers (article numbers) because of these attributes, but all these products belong to same class. In an e-commerce system, all these attributes are virtual, so all products look same. It is important to group all those products that belong to same class and let the user to choose attributes.

Ubertcart provides a module called “uc_attribute (Product Attributes)” to define and attach different attributes to product. After adding this, the administrator can define different attributes for suppliers. Initially, sizes attribute was defined and added options (Small, Medium, Large, etc.). Attributes enable a supplier to configure the extra cost, price, and different article number.

Sun View **Edit**

Product Attributes **Options** Adjustments

Use the checkboxes to enable options for attributes and the radio buttons to specify defaults for the enabled options. Use the other fields to override the default settings for each option. Attributes with no enabled options will be displayed as text fields.

size					
Options	Default	Cost	Price	Weight	
+ <input checked="" type="checkbox"/> S	<input checked="" type="radio"/>	<input type="text" value="0.00"/>	<input type="text" value="0.00"/>	<input type="text" value="0"/>	
+ <input checked="" type="checkbox"/> M	<input type="radio"/>	<input type="text" value="0.00"/>	<input type="text" value="0.00"/>	<input type="text" value="0"/>	
+ <input checked="" type="checkbox"/> L	<input type="radio"/>	<input type="text" value="0.00"/>	<input type="text" value="0.00"/>	<input type="text" value="0"/>	
+ <input checked="" type="checkbox"/> XL	<input type="radio"/>	<input type="text" value="0.00"/>	<input type="text" value="0.00"/>	<input type="text" value="0"/>	
+ <input checked="" type="checkbox"/> XXL	<input type="radio"/>	<input type="text" value="0.00"/>	<input type="text" value="0.00"/>	<input type="text" value="0"/>	

Figure 4.8: Attributes & options for products

4.4.4. Discounts

There can be different kind of discounts with following conditions,

- Time based discount (activates during the specified time span).
- Quantity based (having some condition like overall shopping items exceeded X number of items).

- Fixed number of discounts that defines some discount to expire after a defined number of selling of some product or whole category.
- Shopping subtotal based (if total shopping exceeds some amount).
- Discount coupons for to embed in advertisements.
- Discount on product categories and sub-categories.

Discounts can be enabled by using module “uc_discounts (Discounts)” [19] that is in the optional list of Ubertcar package. By allowing permissions, it allows a supplier to define discounts on products and categories having different conditions.

Active	Name	Short Description	Qualifying Type	Type	Amount	Weight	Created At	Expires At	Operations
✓	test_disc	test discription	Minimum quantity	Percent off	10%	0	04/26/2011 - 14:44	N/A	usage edit copy delete

Figure 4.9: Product discounts

4.4.5. Product Publishing

Product publishing/un-publishing can be a very useful feature in the e-commerce system. It allows a supplier to set publishing and un-publishing dates so that some product appears at the right time (i.e. like some products only appears in some seasons). There is a module called “Scheduler” [20] that enables any content type in Drupal to publish and un-publish on specific dates. It requires enabled “Date” module to function properly. “Date” is built-in Drupal module that displays simple text fields to fill-up dates. Some graphical calendar instead of text field is a better solution. JQueryUI module [21] can be installed that enables Drupal to display more interactive user interfaces. As it can be viewed in figure 4.10, supplier can select and edit publishing and un-publishing dates while create or editing products.

The screenshot shows a 'Scheduling options' form. Under the 'Publish on:' heading, there are two empty text input fields. Below them is a date picker showing 'May' and '2011'. A calendar grid is displayed with days of the week (MO, TU, WE, TH, FR, SA, SU) and dates from 1 to 31. To the right of the calendar, there are two time selection fields, both showing '9:37:15'. The text 'ublishing.' is partially visible between the time fields.

Figure 4.10: Product publishing

4.5. Product and category permissions

One supplier can have hundreds of products organized in tens of categories and sub-categories. All products and categories are not relevant to all stores. It is important to have some permissions system so that a supplier can set permissions for the different kind of stores. There is a question in this concept, how some supplier would know about the type of store to set permissions? The solution is to define some relationship system among suppliers, so each store can only view the relevant suppliers in its list.

For further filtrations, a relationship type can be defined while creating a relationship. For this project, relationships can be divided into four classes called A, B, C, and D. These are just alphabetic labels that define the class of relationship. Implementation of relationships can be read from section 6.7. Same as relationship labels, four classes of permission labels (A, B, C, and D) are defined.

A product or category can be tagged with one of these four classes so that system can filter content for a supplier by using by using relationship-permission matrix shown in table 4.1. These four labels for relationships and permissions were initially defined for this project that can be renamed and extended in the future.

A store having a relationship type “A” can access products and categories tagged with label “A”. “B” type of relationship can only access the content having permission tags B, C, or D. “C” relationship can only access “C” or “D” labeled products and class “D” can access all the products and categories having any permission tag.

		Relationship types			
		A	B	C	D
Permission types	A	Yes	No	No	No
	B	Yes	Yes	No	No
	C	Yes	Yes	Yes	No
	D	Yes	Yes	Yes	Yes

Table 4.1: Shows how relationship and permission tags show/hide products and categories to store users

A simple permission tag can be defined with a product using the CCK module. This field will not be visible to stores. A supplier can choose the permission type for some product while creating a new product. This information should be invisible for stores because it is just for system. Tagging a category and sub-category is not directly possible because a category is not a content type; it is a “Taxonomy” [22]. CCK module is unable to add any extra fields.

A module called “Taxonomy Permissions” was implemented that performs the following actions,

- When this module is installed, it appends a new column called “permission” in database table “term_data” (i.e. Taxonomy module table use to store category information).
- Hide extra field called “details” and show select box to select permissions.
- When some category is edited, it recursively modified permission in sub-categories.

Implementation details of this module can be read in Appendix B. Taxonomy was modified to view permissions with hierarchy of categories and sub-categories.

Name	Operations	Permission
+ Start	edit	D
+ Market	edit	D
+ Clip	edit	D
+ Products	edit	D
+ Mr.	edit	D
+ Tops	edit	A
+ T-Shirt	edit	A
+ test	edit	D
+ Pants	edit	D
+ Score	edit	D

Save Reset to alphabetical

Figure 4.11: Permissions with categories and sub-categories

“Catalog” module was modified so that only permitted stores can access the content. Modification details can be read in Appendix B.

4.6. Private Catalog

Product catalog consists of categories and sub-categories that are arranged in a hierarchical form to organize products. “Catalog” module is built-in “Ubertcart” package that generates “Taxonomy” [15] categories. This module generates a shared catalog for all users. It means there can be only one supplier and many store like other e-commerce websites. After understanding the functionality of “Catalog” module, a customized module can be developing to achieve a private catalog for each supplier user.

“Catalog” module generates an empty shared container and then it calls “Taxonomy” library functions to generate a hierarchy of categories and sub-categories. The “Private Catalog” module was developed to achieve a private catalog for each supplier. Implementation details can be read in Appendix C. This module generates a new catalog when an account is created for supplier user. The module captures different events in the Drupal system and performs following actions,

- When a new user is being registered, if the user supplier, create a new catalog and insert a root category called “Start”.
- When the supplier is logged in, switch the right catalog that belongs to the user.
- When some store accesses some supplier’s catalog, switch the appropriate catalog for it.

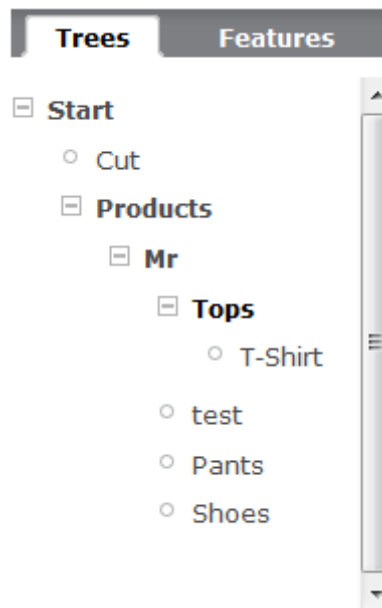


Figure 4.12: Product catalog contains categories and sub-categories

Catalog is just a container for “Taxonomy” categories. To add categories, it requires having private “Taxonomy” categories. By default in Drupal, Taxonomy is shared for all users. To achieve private taxonomy functionality, a module called “Private Taxonomy” [23]. It generates a private taxonomy for each user.

4.7. Product Search

Search functionality is common for supplier and store side. It is used to search articles (products) by name and article number. Drupal has built-in search functionality, but it searches all content types. Product is the only content type that is relevant to supplier user, so it is very important to filter only products.

To achieve filtered search functionality, a special searching module can be used that searches only products. This module is called “Product Search” [24]. This module can’t directly be used in this project for the following reasons,

- This module only searches only “Product” type content while “Article” class is used in this project. Details for this class can be read in section 4.3.1.
- There is no check for permissions (product filtrations) that is implemented in section 4.5.
- It also hides un-published products from suppliers.

So, this module was modified to fit in this project. Details about modification and can be found in Appendix D.

4.8. Summary

The result of all discussion and implementation in this chapter is a complete supplier section having all features described in section 4.1. Figure 4.8.1 shows the complete user interface for supplier section.

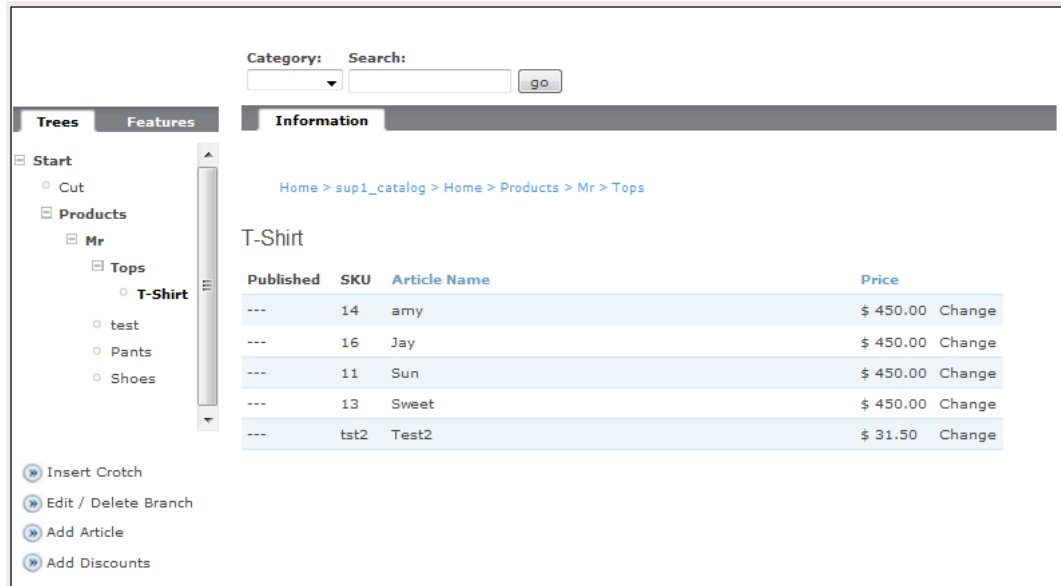


Figure 4.13: Complete supplier section

The most difficult task was to understand Drupal core and to make logic for many to many relationships among suppliers and stores. It took a lot of effort and research to implement complex permissions system and private catalogs for suppliers.

5. CHAPTER 5: STORE/ RETAILER SECTION DEVELOPMENT

5.1. Introduction

This chapter describes the development details of Store section of Brandlink B2B e-commerce solution. Store side of the system contains interfaces, logic and queries for store user. Store users are redirected to store interface after login where they can perform following operations,

- Create/Edit the profile which contains logo, and page that will appear to stores.
- View related supplier profiles, and their catalogs.
- Navigate through the catalog and can search products in catalog.
- Can place an order to a particular supplier.
- View and download marketing information files uploaded by suppliers.
- View the order history report.
- View the advertisements uploaded by Brandlink admin.

5.2. Theme/GUI

Figure 5.1 shows the theme layout on store side.

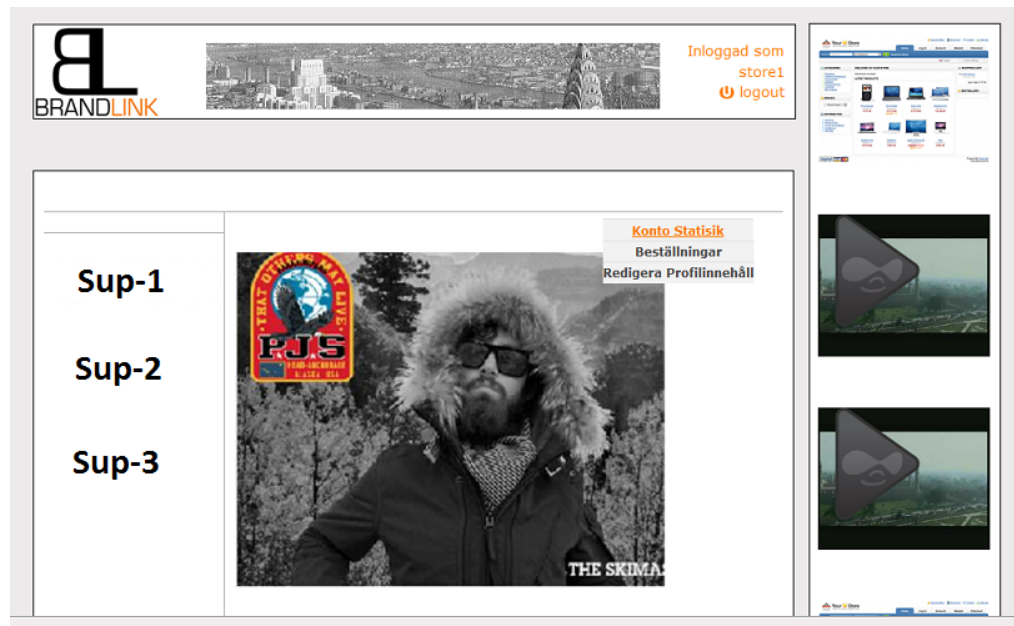


Figure5.1: Store side theme

The store side theme is divided into four main content areas that are,

- Header section

Header contains the Brandlink logo, store logo, user profile link, and logout link. As the header is same as it is on the supplier and admin section so the implementation detail can be found under supplier section 2.11.

- Left sidebar

It contains a list of selected suppliers and the related shopping cart.

- Content area

This area presents the content like products, search block, marketing information, etc.

- Right sidebar

Right sidebar is actually introduced only for the store section. The purpose of this area is to show advertisements to stores.

5.3. View Related Suppliers

Left side bar is used to render a list of supplier's logos that links to supplier's profiles. As the store user sells only specific brands or products, they should not see all supplier accounts in the Brandlink system. Stores have a relation with suppliers that allows to shop from those suppliers. Left side bar of the store's home page contains a list of logos of related suppliers. Following query creates a view which populates related suppliers of current store user.

```
1 SELECT users.uid AS uid,  
2    users.picture AS users_picture,  
3    FROM users LEFT JOIN user_relationships  
4    ON users.uid = user_relationships.requestee_id  
5    WHERE (user_relationships.approved = '1') AND(  
6    (user_relationships.requester_id = ***CURRENT_USER***  
7    OR user_relationships.requestee_id = ***CURRENT_USER***)  
8    AND(users.uid <> ***CURRENT_USER***)
```

Relationships that are used in the query are described in Admin Section under 6.7. In the content area a drop down menu is created, which links to the Store's Orders page and the Store's home page content. If the store has not created any home page, the link will be "add home page" else the link will be "edit home page". Figure 5.2 shows the code used to add this menu.

```

2 <div id="stats-menu"> <ul>
3 <li><a href="#" class="parent">Konto Statistik</a>
4 <ul><li>
5 <?php
6 global $base_url;
7 global $user;
8 echo '<a href="'. $base_url . "/user/" . $user->uid . "/orders" . "'>';
9 echo 'Beställningar'; // Order History
10 echo '</a></li><li>';
11 $result = db_result(db_query("SELECT nid from
12 {node} WHERE $user->uid=uid AND type='prifile_image'"));
13 if($result){
14 echo '<a href="'. $base_url . "/node/" . $result . "/edit" . "'>';
15 echo 'Redigera Profilinnehåll'; // Edit Profile Info
16 }
17 else{
18 echo '<a href="'. $base_url . "/node/add/prifile-image" . "'>';
19 echo 'Lägga till Profilinnehåll'; // Add Profile Info
20 }
21 echo '</a>';
22 ?>
23 </ul></li></ul></div>
24

```

Figure 5.2: Relationships block

5.4. Advertisements

Right sidebar of store section shows scrolling image/video advertisements. The slideshow of images & videos is created by using following Drupal modules.

i. Content Construction Kit (CCK)

In Drupal, CCK module helps to create content types and add fields i.e. image upload field. For advertisements, a content type is created which contains image and video upload fields. The admin can create content of this type, and store can only view the advertisement content.

ii. Views

“View” is a Drupal built-in module that is used to generate views of different content types. “Views” is used with two other modules called “Views Slideshow” and “Views Slideshow ddblock” to generate rolling advertisements.

iii. Views Slideshow

“Slideshow can be used to create a slideshow of any content type that can be display in a View” [25]. Slideshow functionality provides a graphical user interface to modify different parameters of slideshow. For example animation speeds.

iv. Slideshow Ddblock

This module accepts a dynamic list of images, videos and text type content and presents them in a form of a slideshow on a web page. Content changing frequency and other settings can be easily changed by its easy configuration. The module helped us to include videos in our slideshow.

v. **Lightbox2**

It provides a functionality to display images in a popup-window that appears without re-loading the web page. Lightbox2 can display images and videos [26].

5.5. Catalog

By clicking on some supplier logo, the store navigates to Supplier page and accesses the supplier’s catalog. Store can navigate through different categories and products of the supplier.

Relationship type between Supplier and Store will define which products store can view and order. As described in section 4.5, the product added by supplier also falls into one of the four types (A, B, C, and D). If the store has ‘A’ type relationship with a supplier then store will be able to navigate products of all types A to D. And if the relationship is of B type then store will be able to navigate products of type B to D. Similarly with C type relationship store can navigate products of C to D and if the relationship is of type D then only products of type D are accessible to the store. This is also described in table 4.5.1 in supplier section.

5.5.1. Products Page

In the content area, a grid containing products of particular category is placed. Here product’s image, size, price, choose an option add to cart, etc. are shown. Choose option button links to product detail page, and add to cart button will add product to cart.

Clicking on “add to cart” doesn’t refresh the page and cart updates through Ajax. For this functionality Ajax Driven Cart module is used which updates cart without refreshing page using JavaScript [27]. Figure 5.3 describes the product detail page.

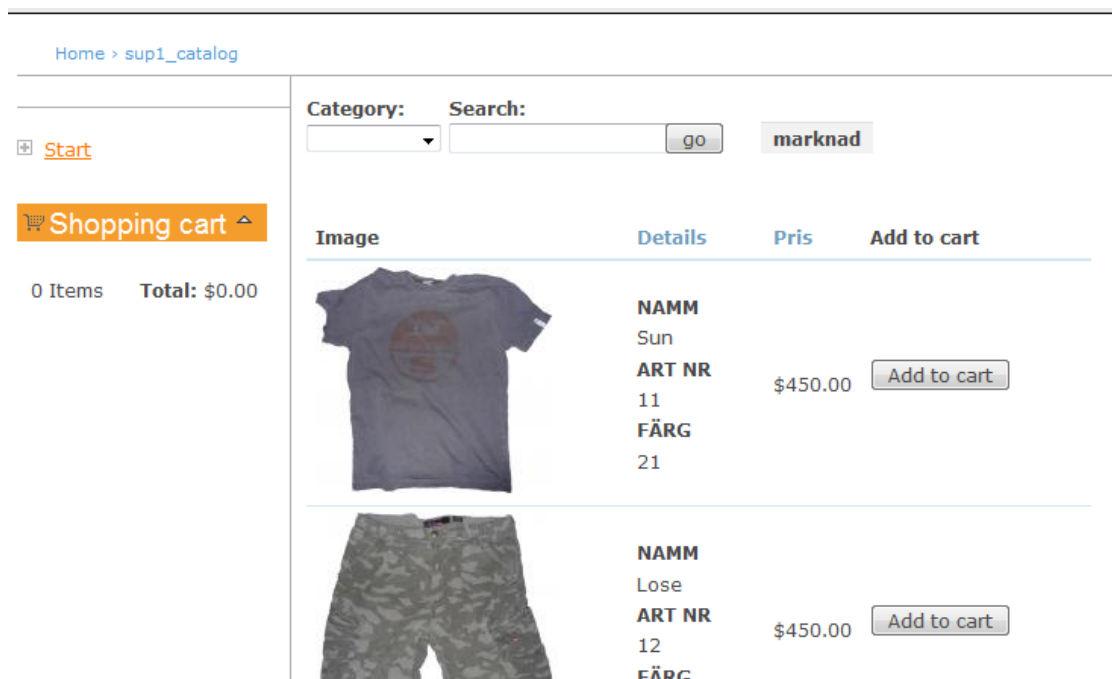


Figure 5.3: Products page

5.5.2. Search Products

At the top of the content area search box will be shown. The purpose of the search box is to make it easy for store to locate desired products. All categories of the supplier will come in a drop down box. A store will be able to select a category and type product name or article number and click go. The result page will show all products in selected search criteria. For searching, a module called “Product Search” was modified. Details can be read from section 4.7.

5.5.3. Shopping Cart

Store will be able to add products to the cart. On supplier page, there is a block on the left side containing cart overview and links. Links on the block are view-cart and checkout.

Here detailed view of the shopping cart appears. User has more control on the cart and can remove or change quantity of selected items in cart. Clicking on “update cart” button will make desired changes in cart. Checkout button links to checkout first step.

5.5.4. Supplier Information

The suppliers can add their logos, offers, important guidelines, interests, and future plans in the form of images and text.

5.6. Product Details

This page shows all details about a selected product. Here again add to cart button will be shown but with enhanced functions. Users are able to select size (s, m, l, xl etc.) and can also enter quantity (number of items to purchase). The cart again updates through Ajax. A product’s front and back side images will also appear here.

5.7. Orders History page

This page shows history of orders made by current store in a grid. Figure 5.4 shows the history of sample transactions in tabular form.

Home > My account

Orders

Date▼	Order #	Status	Supplier	Products	Total
04/08/2011	12	not complete	sup1	2	\$900.00
03/10/2011	11	complete	sup1	1	\$350.00
03/10/2011	2	complete	sup1	1	\$450.00
03/09/2011	1	complete	sup1	1	\$450.00

Figure 5.4: Store's order history

To display the order history, “uc_order” module was used that is a part of Ubertcart module. Uc_order module does not show the supplier/ seller. To add supplier we did the following customization in uc_order module. Line 999 and 1000 were added to display new columns in the history table. Line 1012 is a SQL query that retrieves the history data and stores in a variable.

```

996     $header = array(
997         array('data' => t('Date'), 'field' => 'o.created', 'sort' => 'desc'),
998         array('data' => t('Order #'), 'field' => 'o.order_id'),
999         array('data' => t('Status'), 'field' => 'os.title'),
1000        array('data' => t('Supplier'), 'field' => 'os.title'),
1001        array('data' => t('Products'), 'field' => 'products'),
1002        array('data' => t('Total'), 'field' => 'o.order_total')
1003    );
1004
1005    $rows = array();
1006
1007    $context = array(
1008        'revision' => 'themed-original',
1009        'type' => 'amount',
1010    );
1011
1012    $result = pager_query("
1013        SELECT o.order_id, o.created, mpos.order_status
1014        ,u.name, SUM(op.qty) AS products, o.order_total AS total
1015        FROM {uc_orders} AS o
1016        LEFT JOIN {mp_seller_order_statuses} AS mpos ON o.order_id = mpos.order_id
1017        LEFT JOIN {uc_order_products} AS op ON o.order_id = op.order_id
1018        LEFT JOIN {node} AS onode ON op.nid = onode.nid
1019        LEFT JOIN {users} AS u ON onode.uid = u.uid
1020        WHERE o.uid = $user->uid
1021        GROUP BY o.order_id, o.created, mpos.order_status,
1022            o.order_total" . tablesort_sql($header), 20, 0,
1023        "SELECT COUNT(*) FROM {uc_orders} WHERE uid = $user->uid");

```

5.8. Marketing Information

A supplier can upload advertisements files (images and videos) in a special section or page called “Marketing Information” section. Each supplier has its private marketing section that can be only accessed by its associated store.

A store can view and download different types of media like text, images, and videos. “WebFM”[28] module was used to create a web based file manager/explorer that allows suppliers to manage different advertisement media files. After configuration, the module had these features,

- Interactive graphical user interface.
- Easy to upload files and organize them into different folders.
- Single file upload with version options for file overwrite.
- Easy to download a file or edit in place.

5.9. Summary

Result of all this discussion and implementation is a complete supplier section which performs all functionalities that are described in section 5.1. Figure 5.5 shows the complete user interface for supplier section.

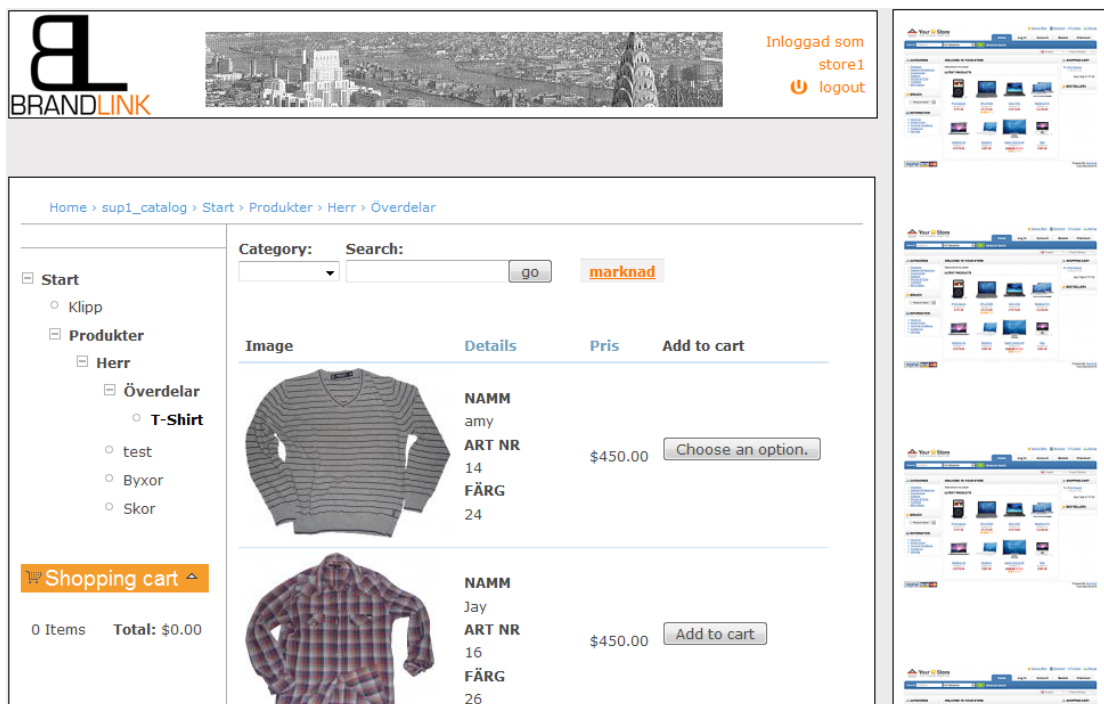


Figure 5.5: Complete store side

6. CHAPTER 6: BRANDLINK ADMINISTRATOR SECTION DEVELOPMENT

6.1. Introduction

Admin section is a graphical user interface that lets the administrator manage content, users, and business transactions. After login, admin user is redirected to this interface where it can view and manage all information. With respect to this project, Drupal's default admin section is very complex and un-organized. It is very important to group all information and operations in defined sections. Here is a list of operations that an administrator can perform,

- Create/Edit/Delete Supplier and Store user accounts.
- Built relationships among suppliers and stores.
- View and search orders from Stores to Suppliers.
- Generate statistic reports.
- Administer advertisement section that appears on Store section. It can upload and organize banners (images) and videos.
- Admin can send personal, multicast, and broadcast emails to users.

Admin section can be divided into two main parts; user interface and functionalities.

6.2. Theme/GUI

This section describes user interface development for admin section. The layout is the same as in the supplier section. As figure 6.1 describes, it consists of three main graphical sections: header, left content, and right content. To develop these sections, supplier's theme was copied here that contains all three sections. Details of layout development can be read from supplier development section 4.2.

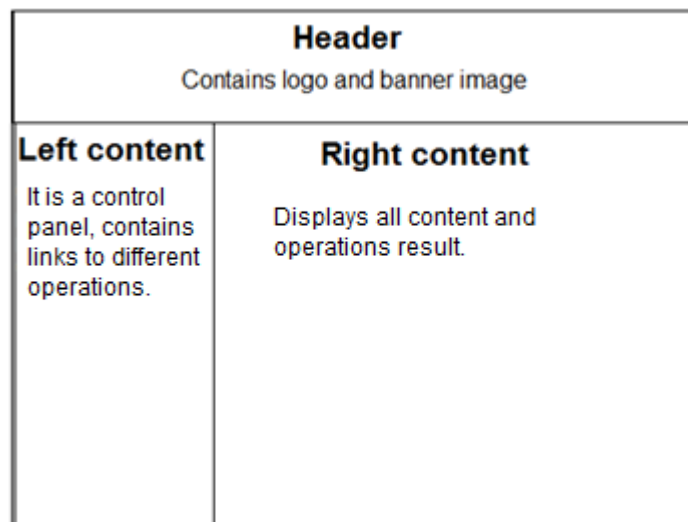


Figure 6.1: Admin section layout

“JQuery-Tabs” [13] can be use in this section to make the user interface more interactive and to group different kind of contents. For left content, only one tab is enough to display hierarchical links to different operations. But for default home page, two tabs were created in main content to display list of suppliers and stores that can be very useful for an administrator.



Figure 6.2: JQuery tabs to organize content

Default admin section provides very complex hierarchical links to different function that are more relevant for a developer than a user. To make it simple, hierarchies of relevant links were created. To create new links, Drupal provides an interface called “Menus”. One can create a new menu by using (Admin → Site Building → Menu) link.

Path: *

The path this menu item links to. This can be an internal Drupal path such as `node/add` or an external URL such as `http://drupal.org`. Enter `<front>` to link to the front page.

Menu link title: *

The link text corresponding to this item that should appear in the menu.

Description:

The description displayed when hovering over a menu item.

Enabled
Menu items that are not enabled will not be listed in any menu.

Expanded
If selected and this menu item has children, the menu will always appear expanded.

Parent item:

The maximum depth for an item and all its children is fixed at 9. Some menu items may not be available as parents if selecting them would exceed this limit.

Figure 6.3: Drupal menus

6.3. Home Page

6.3.1. Content Area: Suppliers & Stores

Content area in the admin section consists of two parts, left and right content. Right content consists of hierarchically organized links and sub-links that are used for easy navigation. Drupal provides a complete functionality related to user creation and management for an administrator. But that is only for all (unfiltered) users.

Drupal “view” can be created to display only filtered lists with administrative links. “View” is just like SQL query. Drupal provides a very easy and graphical interface that lets you to generate views for some content types. One can create views using (Admin → Site Building → Views) Figure 6.4 show Drupal interface to generate and save view.

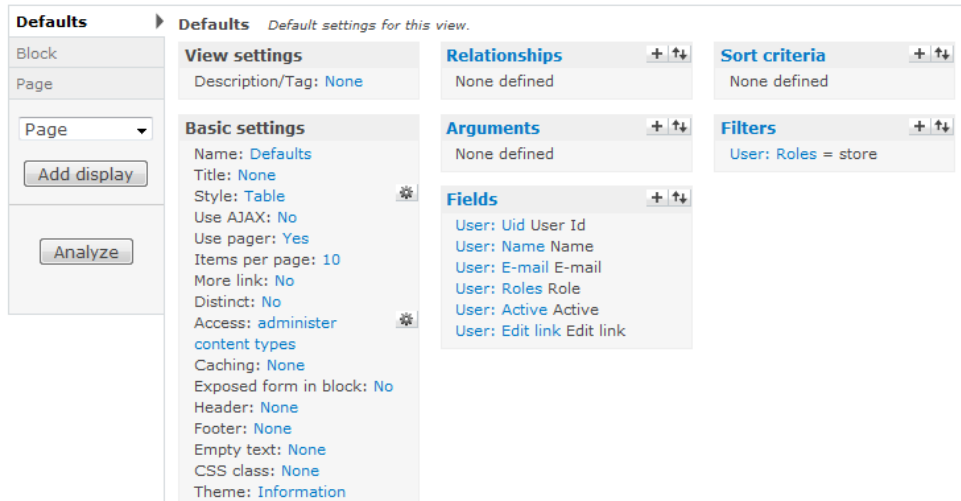


Figure 6.4: Drupal Views

As described in theme section 6.2, two tabs were created to display lists of supplier and store users. Drupal views were created to display lists. Figures 6.5 and 6.6 show the result of “views”.

```

1 SELECT users.uid AS Uid, users.name AS Name, users.mail AS E-mail,
2     users.status AS Active, users_roles.name AS Roles
3 FROM uses JOIN users_roles ON users.uid = users_roles.uid
4 WHERE users_roles.rid = 3
5 ORDER BY users_name ASC
6

```

Uid	Name	E-mail	Roles	Active	
4	sup1	sup1@123.com	supplier	Yes	edit
5	sup2	sup2@123.com	supplier	Yes	edit

Figure 6.5: List of suppliers

User Id	Name	E-mail	Role	Active	
2	store1	store1@123.com	store	Yes	edit
3	store2	store2@123.com	store	Yes	edit

Figure 6.6: List of stores

```

1 SELECT users.uid AS Uid, users.name AS Name, users.mail AS E-mail,
2     users.status AS Active, users_roles.name AS Roles
3 FROM uses JOIN users_roles ON users.uid = users_roles.uid
4 WHERE users_roles.rid = 4
5 ORDER BY users_name ASC

```




Note: Drupal assigns a unique “User ID” to each user that is used in these queries to filter out the records.

6.4. Orders Page

Order page displays a list of orders from stores to suppliers, so that admin can monitor business activities and perform an operation to these orders. Ubertcart package provides this functionality to administer orders, one can access using (admin/store/orders/view) path. This path can be attached with any menu so that it is easily accessible for non-technical user. Figure 6.7 shows list of orders generated by “Orders” function.

Orders

View order: View by status: Active orders ▾

 = View  = Edit  = Delete













Actions	Order ID	Customer	Total	Purchase date	Status
  	12	arif arif	\$900.00	04/08/2011	Completed
  	11	arif arif	\$350.00	03/10/2011	Completed
  	2	arif arif	\$450.00	03/10/2011	Completed
  	1	arif arif	\$450.00	03/09/2011	Completed

Figure 6.7: Orders page

6.5. Statistics Reports

Statistic reports are result of historical records that are organized with respect to different parameters. These reports help a user in decision making in current situation and for future. Three kind of statistic report can be useful for an administrator in this system,

i. Total per Store

It displays a list of stores with the total purchase amount for each store in currency.

ii. Total per Supplier

It displays a list of suppliers with their total selling amount.

iii. Total per Product

Total per product displays list of all products from all suppliers with the total selling amount for each product.

6.5.1. Report Section development

The default Drupal system generates very basic reports that are not useful for this system. Drupal views are used here to generate custom reports. Figure 6.8 shows settings for Drupal view.

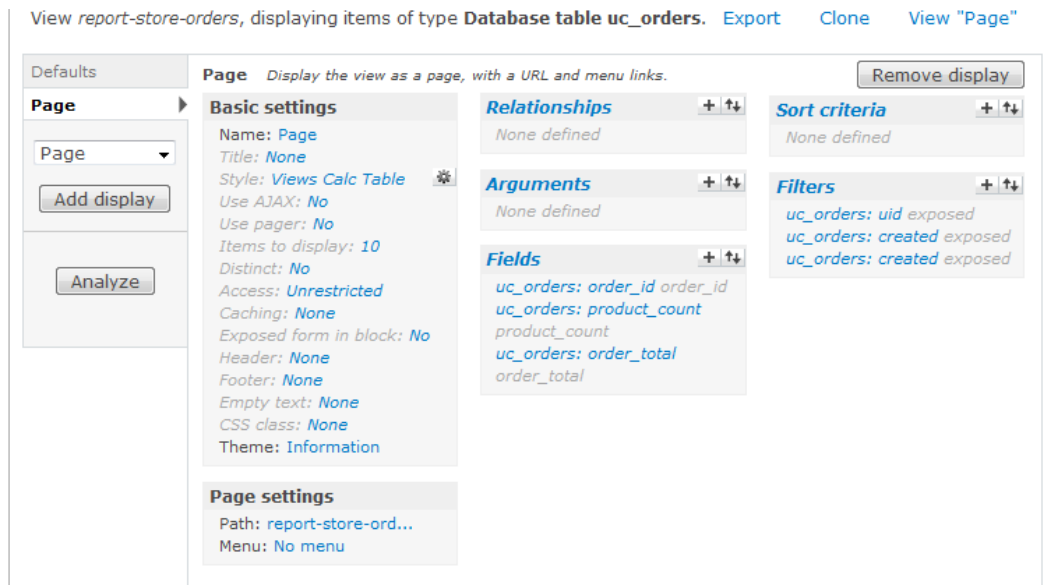


Figure 6.8: Drupal view for statistic reports

This view looks static because it will not change by user input. To make dynamic views, filters are used that gives input to a view to re-generate its results. A filter is used to reduce the result set. For example, the very basic view with no settings will simply give all content available on the site.

6.5.2. Exposed Filter

An exposed filter is a filter that's value can be set by a user. It can be a text-field or some select box to get input from a user. Three filters were used in this project to provide full control to search relevant records. These filters are,

- Start date.
- End date.
- Store.

Start and end filters are date fields and JQuery module were already used by this project that converts simple date fields into popup calendar. Figure 6.5.2.1 shows input to generate some statics report and the result. Other implementation details can be read from Appendix C.2.

The code in Appendix C.2 loads Store/Suppliers/Products and enables the date picker for date fields. The final form is shown in figure 6.9.

Butik: All Butik

Start date: (Format: 2011:05:14)

End date: (Format: 2011:05:14)

Selecting options and clicking apply will show the statistics report like.

Butik: All Butik

Start date: 2011:01:15 (Format: 2011:05:14)

End date: 2011:05:11 (Format: 2011:05:14)

order_id	product_count	order_total
1	1	450
2	1	450
12	2	900
11	1	350
Total SUM		2,150

Figure 6.9: Statistics report

In figure 6.9, last record “Total SUM” is not drupal default functionality. It can be achieved by using “Views Calc” [29] module. It provides a lot of different features like Count of records, Minimum, Maximum, etc.

6.6. Advertisements Page

Advertisements page is a list of rolling banners (images) and video clips that appear to the Store user. These advertisements are dynamic and the administrator can create/edit the list and animation order. A new content type named “iv_ads” was created and enabled image attachment option to manage advertisements. After content creation, Drupal views can be used to render a list of advertisement. Here is a code that generates a view for one rolling banner.

```

1 SELECT node.nid AS nid,
2 node_data_field_video.field_video_embed AS node_data_field_video_field_video_embed,
3 node_data_field_video.field_video_value AS node_data_field_video_field_video_value,
4 node_data_field_video.field_video_provider AS node_data_field_video_field_video_provider,
5 node_data_field_video.field_video_data AS node_data_field_video_field_video_data,
6 node_data_field_video.field_video_version AS node_data_field_video_field_video_version,
7 node_data_field_video.field_video_duration AS node_data_field_video_field_video_duration,
8 node.language AS node_language,
9 node.type AS node_type,
10 node.vid AS node_vid,
11 node.title AS node_title
12 FROM node
13 LEFT JOIN content_type_advertisement node_data_field_video
14 ON node.vid = node_data_field_video.vid
15 WHERE (node.status <> 0) AND (node.type in ('advertisement'))

```


To create a strip of rolling advertisements, same code can be used to create other advertisements.

6.7. Relationships Page

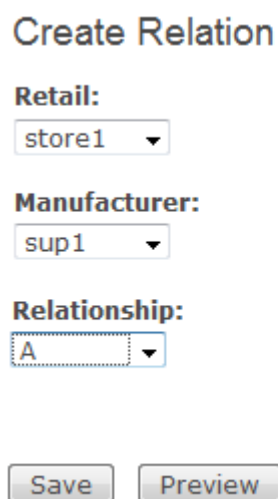
Relationships page allows administrator to connect multiple suppliers to stores. This connection allows store to view related suppliers and filtered catalog of products. Four types of connections (A, B, C, and D) were defined. Purpose and usage of these four kinds of labels can be read from supplier section 4.5. These connections between supplier and store are called relationships.

Implementation of relationships can be possible using “User Relationships” [30] module. By using this module, the administrator can create simple relationships. Admin can give users the option to auto approves relationships on a per-relationship type basis. One important function of User Relationship is its integration with “views” module providing filters, arguments, and fields. This functionality makes it very easy to filter different relationship types for an administrator.

“User Relationships” module doesn’t provide complete functionality to assign relationship types. It is required to extend this solution using some other techniques and programming.

6.7.1. Relationships development

A simple solution to the labeled relationship can be to create a Drupal content type for it. That content type will provide a user interface for an administrator to make a new relationship. New content type can be created using (Administer → Content Management → Content Types → Add Content Type). Figure 6.10 shows implemented content type for relationships.



The image shows a web form titled "Create Relation". It contains three dropdown menus. The first is labeled "Retail:" and has "store1" selected. The second is labeled "Manufacturer:" and has "sup1" selected. The third is labeled "Relationship:" and has "A" selected. Below the dropdowns are two buttons: "Save" and "Preview".

Figure 6.10: Relationships content type

To load the list of suppliers and stores, some PHP code can be written to fetch these lists from the database. Here is PHP code that is used in this project to populate lists.

```

1 $result = db_query("SELECT users.uid, name FROM {users}
2     LEFT JOIN {users_roles} on users.uid=users_roles.uid
3     WHERE users.status=1 AND users_roles.rid=4");
4 while ($title = db_fetch_array($result)){
5     $options[$title[uid]]=$title[name];
6 }
7 return $options;

```

There is a SQL query written on line 1 that fetches the list of users that is filled up in a select box using “while” loop on line 4.

After saving a new content type, it doesn’t create a new relationship because in order to make a relationship in Drupal, one must add relationship record in “User Relationships” module’s table. To solve this problem, a small module was written that records relationship information in a database when the administrator:

1. Creates a new relationship.
2. Edits some existing relationship.
3. Deletes some existing relationship.

Implementation details for that module can be read from Appendix C.1.

6.8. Email Section

Email section is very important for the administrator to communicate with suppliers and stores connected with this system. Drupal doesn’t provide functionality for mass mailing. A simple solution to achieve this function can be a module called “AdvanceUser” [31] having functionality to send mass emailing like selected users and all users. It can also filter users by roles. Figure 6.11 shows user interface for mass emailing.

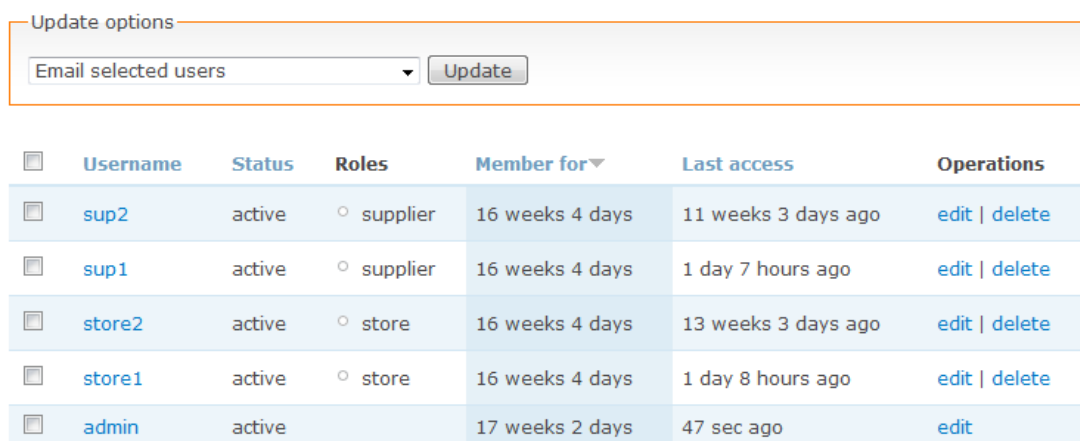
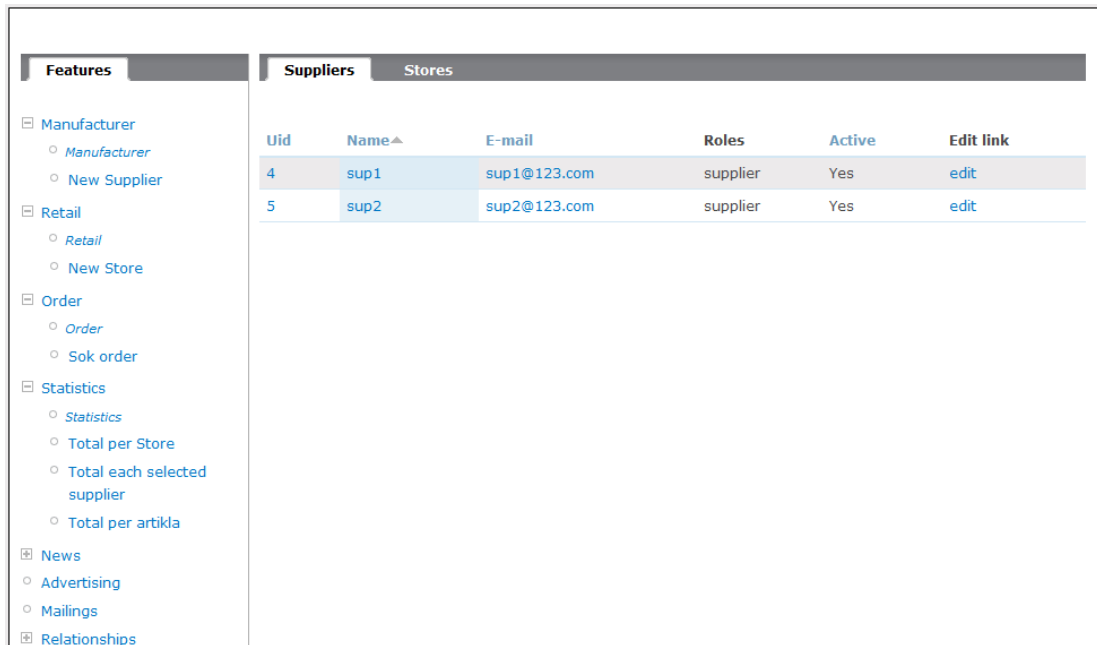


Figure 6.11: Mass emailing user interface

6.9. Summary

Result of admin section development is a customized and easy to use control panel for administrator that provides all information and there controls. A major part of admin section development was already done in supplier and store sections because admin section is just to administer all that functionalities that we build. Figure 6.12 shows complete user interface for administrator.



The screenshot displays the Admin Section user interface. On the left is a sidebar menu with the following items:

- Features
 - Manufacturer
 - Manufacturer
 - New Supplier
 - Retail
 - Retail
 - New Store
 - Order
 - Order
 - Sok order
 - Statistics
 - Statistics
 - Total per Store
 - Total each selected supplier
 - Total per artika
 - News
 - Advertising
 - Mailings
 - Relationships

On the right, there are two tabs: 'Suppliers' (selected) and 'Stores'. Below the tabs is a table with the following data:

Uid	Name▲	E-mail	Roles	Active	Edit link
4	sup1	sup1@123.com	supplier	Yes	edit
5	sup2	sup2@123.com	supplier	Yes	edit

Figure 6.12: Admin Section

7. DEPLOYMENT

7.1. Introduction

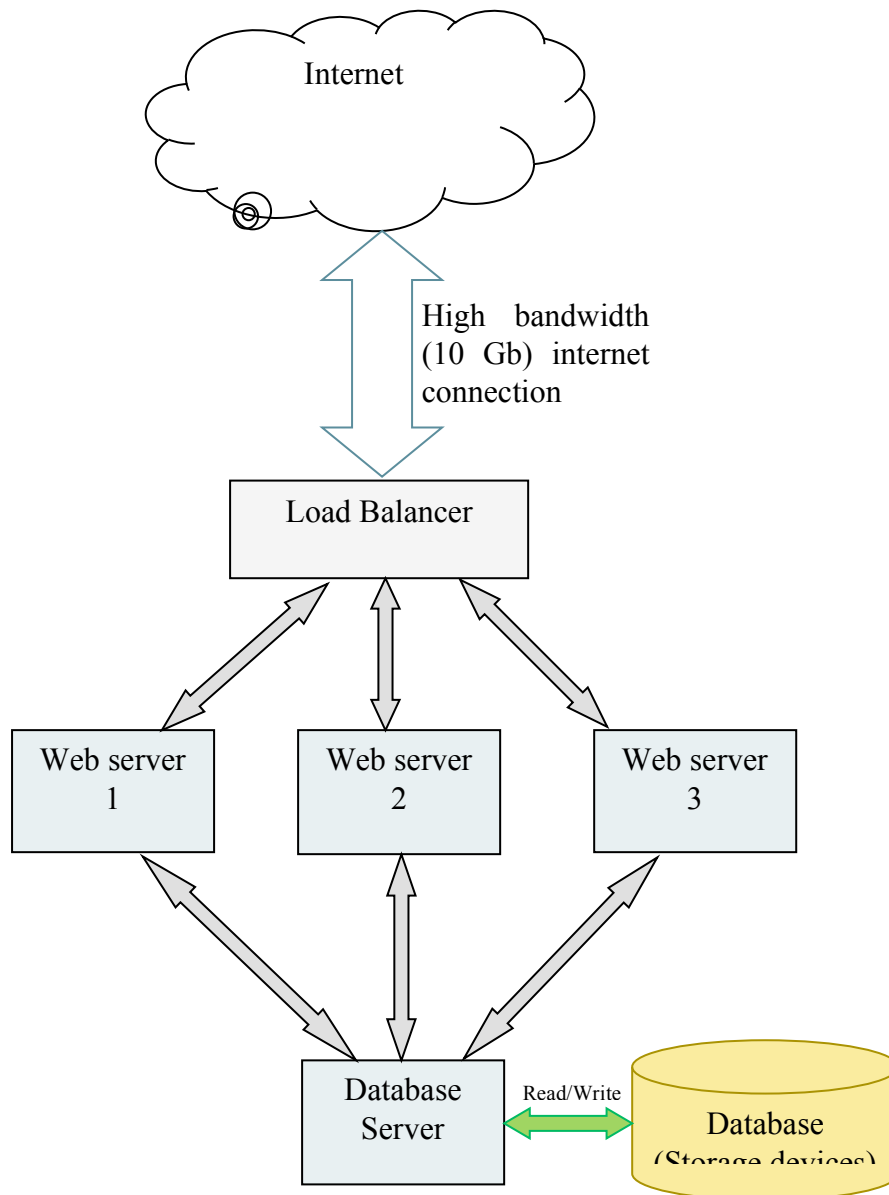
An application deployment process is a set of activities that enables an application to execute in the production (publically available) environment. [32] As this project is a web application, it should be placed on a server machine so that users can access it via network. A web application can be physically divided into many parts called “tiers” that work together as a single application. Either all tiers can be implemented on same machine or separate machines connected via network.

Distribution of tiers over machines depends on how much work load each machine will face in the real environment. Details about web application architecture and tiers can be read from section 2.2. The number of “tiers” depends on which technology is used for web development. If the web application is static, it doesn’t require any record management system or databases. As the development of e-commerce web application is a part of this project, it requires hardware and software resources to sustain in the real environment.

7.2. Hardware architecture

Hardware architecture is a combination of arrangement and configuration of different hardware components in such a way that works as a single hardware device. A web application requires some set of software application configured on either a single machine or multiple connected via network. This machine should have access to public networks (internet) so that users can access it.

Hardware architecture depends on what kind of application is going to be deployed. Figure 7.1 describes available hardware architecture for this project. It shows three server machines called “Web Servers”[4]. These are connected to the internet via “Load Balancer” [33]. There is a database server that is shared for all three web servers.



Symbols:



This symbol shows TCP/IP communication between two machines.

Figure 7.1: Hardware architecture

Initially, web application can be placed on any web server after installation and configuration of web server software on it. Later the same configuration and a copy of the application can be put on other servers so that they can process client requests in parallel with help of the load balancer. All servers use the CentOS operating system.

7.3. Installation of server applications

Applications and their installation commands are presented in Appendix D.

7.4. Summary

After configuring hardware and deployment of web application it was running successfully on the web server. It can be accessed via (<http://213.136.63.111/brandlink>). For now, this application is available only via URL because there is no Domain Name Server (DNS) configured that registers any web address with IP.

8. RESULTS AND FUTURE WORK

8.1. Introduction

This chapter concludes the present work and covers limitations and future possibilities to extend the present work.

8.2. Results

The project uses web-based technology to make the business transactions easy to execute & manage. Goals and objectives of the project have been generally achieved and the following are the tangible benefits of project.

First, processing time of the business transactions between suppliers and stores will reduce. It is hard to write order and delivery documents on paper or even using desktop application. Migration of order and delivery system to Internet reduces the time needed to process transactions.

Second, the accuracy of transaction data improves and requires less rework. The system is more responsive (i.e., interactive) and therefore there is less chances to commit human errors. Through proper validation of input values and intuitive user interface the system becomes more responsible and error free.

Third, complete database of system is a database server which enhances the security, efficiency, and provides a better integrated system equipped with proper backup facility.

8.3. Future Work

For future work, we would like to recommend three distinctive areas. First one is the non-functional testing of Brandlink B2B solution. We did functional testing which establishes the correct operation of the solution (i.e. it matches the expected behavior defined in the design requirements). The non-functional testing verifies that the software functions properly even when it receives unexpected or invalid inputs. Following is the list of non-functional testing phases, which can help to improve the Brandlink B2B solution.

- Software performance testing and load testing.
- Stability & usability testing.
- Security testing.

Second recommendation of future work is to upload new modules developed during this project to the Drupal community site. Drupal has thousands of developer and non-developer active users. Sharing modules on Drupal community will help a large community and also will be helpful to improve modules. Uploading modules on Drupal will require properly organizing code and preparing documentation of modules according to Drupal standards.

Third improvement needed in this project is online payment integration. Currently project has no integration with any online payment system like Paypal, visa etc.

9. CRITICAL ANALYSIS

9.1. Introduction

This chapter describes about the critical analysis about selection of implementation framework, the good decision that was taken for project implementation and what one can learn from this project.

9.2. Implementation framework

A variety of scripting languages was available to implement this project. For example: PHP, Dot NET framework, Ruby on Rails, etc. Each language has some plus and minus points. Some languages are freely available and some are not. As described in section 2.4.2, PHP is free and supported by many web servers. PHP contains all the basic functions to implement a complex web application and most web applications are implemented in PHP. So PHP was selected as the main implementation language for this project.

There are different frameworks that are written in PHP that provide a skeleton to build a web application very easily. All frameworks are not general purpose; they are made to build some type of web application. For example, “WordPress” is a content management system (CMS) written in PHP that helps to build web blogs.

There are a lot of content management systems that can be used to build e-commerce web applications. For example: Drupal, OpenCart, Joomla, etc. After some research, we found that Drupal and OpenCart provides a lot of modules and features to build an e-commerce web application. Here is the comparison of some features in Drupal and OpenCart.

Feature	Drupal	OpenCart
Written in PHP	Yes	Yes
Freely available over internet	Yes	Yes
Variety of modules and web graphics are available	Yes	No
Has built-in features for e-commerce	No	Yes
Framework can be easily modified	Yes	No
New modules can be written easily.	Yes	No
Existing modules can be	Yes	No

change easily		
Well documented	Yes	No
Large community and support	Yes	No
Framework and modules are frequently updated by community	Yes	No

9.3. The good decision

The scope of this project was too big to implement in the limited time period. So we preferred to select already available modules and modify them according to our specifications. This helped to implement the project very fast.

9.4. Major problems

Drupal and its available modules are written by its community, consisting of professional and non-professional users. There are some bugs and performance related issues in the Drupal framework and its modules. Especially those issues appear during the development of large systems. Even the documentation is available, but still takes a lot of time to find and fix those issues.

9.5. Learning

This project learns that how to automate a real life problem from scratch and to reuse already built software applications and modules. The drawback of code re-using that the code may contain errors due to lack of testing. If this project is offered again with longer time period, PHP will be the best implementation language and the written code will be error free, well documented, and completely related to the project.

REFERENCES

- [1]. <http://www.speedment.com>
- [2]. Wikipedia, the free encyclopedia (2011) Business-to-business. <http://en.wikipedia.org/wiki/Business-to-business> (20 May 2011).
- [3]. David Gourley & Brian Totty with Marjorie Sayer, Sailu Reddy & Ansbu Aggarwal: HTTP The Definitive Guide, First Edition, page 4.
- [4]. Wikipedia, the free encyclopedia (2011) Web application. http://en.wikipedia.org/wiki/Web_application (06 May 2011).
- [5]. Ross Shannon (2010) what is HTML? <http://www.yourhtmlsource.com/starthere/whatishtml.html> (05 May 2011).
- [6]. <http://www.whatiscss.com/> (20 May 2011).
- [7]. Online resource accessed on May 2011, from. <http://web-hosting.candidinfo.com/server-operating-system.asp> (07 May 2011).
- [8]. Wikipedia, the free encyclopedia (2011) PHP. <http://en.wikipedia.org/wiki/PHP> (03 April 2011).
- [9]. <http://drupal.org/>
- [10]. <http://wiki.centos.org/>
- [11]. Wikipedia, the free encyclopedia (2011) MySQL. <http://en.wikipedia.org/wiki/MySQL> (23 April 2011).
- [12]. 2011 PuTTY FAQ. <http://www.chiark.greenend.org.uk/~sgtatham/putty/faq.html#faq-meaning> [17].
- [13]. Nedjo Rogers (2008) Tabs (jQuery UI tabs). <http://drupal.org/project/tabs>
- [14]. Wikipedia, the free encyclopedia (2011) Shopping cart software. http://en.wikipedia.org/wiki/Shopping_cart_software (22 April 2011).
- [15]. <http://www.ubercart.org/>
- [16]. Yves Chedemois (2006) Content Construction Kit (CCK). <http://drupal.org/project/ckk>
- [17]. Nathan Haug (2006) ImageField. <http://drupal.org/project/imagefield>
- [18]. Nathan Haug (2006) FileField. <http://drupal.org/project/filefield>
- [19]. Psynaptic (2007) Discounts. <http://www.ubercart.org/contrib/143>
- [20]. Eric Schaefer (2006) Scheduler. <http://drupal.org/project/scheduler>
- [21]. Daniel F. Kudwien (2008) JQuery UI. http://drupal.org/project/jquery_ui

- [22]. Drupal (2011) About Taxonomy. <http://drupal.org/node/774892>
- [23]. Michael Videlgauz (2009) Private Taxonomy Terms.
http://drupal.org/project/private_taxonomy
- [24]. Sivaram M (2009) Ubercart Product search module. <http://drupal.org/node/562892>
- [25]. Aaron Winborn (2007) Views Slideshow. http://drupal.org/project/views_slideshow
- [26]. Stella Power (2006) Lightbox2. <http://drupal.org/project/lightbox2>
- [27]. Erik Seifert (2008) Ubercart ajax cart. http://drupal.org/project/uc_ajax_cart
- [28]. Cameron Eagans (2007) Web File Manager. <http://drupal.org/project/webfm>
- [29]. Karen Stevenson (2006) Views Calc. http://drupal.org/project/views_calc
- [30]. Alex Karshakevich (2007) User Relationships.
[http:// drupal.org/ project/user_relationships](http://drupal.org/project/user_relationships)
- [31]. Earnie Boyd (2006) Advanced User. <http://drupal.org/project/advuser>
- [32]. Wikipedia, the free encyclopedia (2011) Software Development.
http://en.wikipedia.org/wiki/Software_deployment (15 May 2011).
- [33]. Wikipedia, the free encyclopedia (2011) Load balancing (computing).
http://en.wikipedia.org/wiki/Load_balancing_computing (16 May 2011).

A.1 Supplier / Brand side Requirements

<i>Id</i>	<i>Author</i>	<i>Rev</i>	<i>Type</i>	<i>Supplier / Brand Section</i>
10000	arif	1	Functional	Header will be containing the Brandlink logo and Supplier logo.
10001	arif	1	Functional	On right side of header there will be logged in supplier's name linking to supplier profile page.
10002	arif	1	Functional	Search block will have functionality to search articles/branches by article name and article number.
10004	arif	1	Functional	Logout link will be there to log off from supplier account.
10005	arif	1	Functional	Supplier home page content part will allow supplier to set the text/images shown to store user.
10006	arif	1	Functional	Left side bar will contain two tabs navigational tree (trad) and Features (funktioner).
10007	arif	1	Functional	Bottom of left side bar will contain the links like, insert branch, edit branch, add/remove article etc.
10008	arif	1	Functional	The navigational tree will contain hierarchical links to branches and articles at leaf nodes.
10009	arif	1	Functional	Clicking on a branch will lead to branch page.
10010	arif	1	Functional	Add new article link will lead to "add article" page. See Description sheet to find fields of "add article" page.
10011	arif	1	Functional	Pressing the "add article" button will submit the form and "article view" page will open.
10012	arif	1	Functional	Set article publishing date will allow scheduling the publishing date of an article.
10013	arif	1	Functional	The system will automatically publish the article as selected publishing date will reach.
10014	arif	1	Functional	Using edit link supplier will be able to edit the navigational tree settings.
10015	arif	1	Functional	The article can be moved and/or copied to another location in the tree.
10016	arif	1	Functional	Single article can be placed in multiple places.
10017	arif	1	Functional	Clicking save button will update the navigational tree settings.

10018	arif	1	Functional	Permission settings will allow specifying which stores are able to see and order an article. On Description sheet there are permissions.
10019	arif	1	Functional	It shall be possible to add new branch. Branch name field will be required to insert new branch.
10020	arif	1	Functional	Edit branch will allow changing branch name, description or discount rate associated with the branch.
10021	arif	1	Functional	Remove branch option will allow removing a branch. System will allow removing a branch only if it's empty.
10022	arif	1	Functional	There will be option to set branch publish date.
10023	arif	1	Functional	Supplier will indicate where in the tree, branch must be available. Branch can be moved to other places in tree.
10024	arif	1	Functional	It will be possible to place Single branch under multiple parent nodes in tree.
10026	arif	1	Functional	On left sidebar Features tab will contain Orders, and Market information links.
10029	arif	1	Functional	Order management will allow supplier to view all orders.
10030	arif	1	Functional	Selecting a particular store will filter and show orders made by that store.
10031	arif	1	Functional	"Order detail" page will show detailed information of a particular order.
10032	arif	1	Functional	Supplier can upload different documents related to market on market information page. Stores will be able to download and use these.
10033	arif	1	Functional	Documents allowed types will be JPEG, TIFF, Word, GIF, and Video.
10034	arif	1	Functional	Store user will be able to download the uploaded documents.
10036	arif	1	Functional	On "Marketing" page supplier will be able to see documents uploaded by him/her.
10037	arif	1	Functional	The supplier will be able to filter the documents by document type.
10038	arif	1	Functional	The supplier can also delete particular document.

A.2 Store/ Retailer side Requirements

<i>Id</i>	<i>Author</i>	<i>Rev</i>	<i>Type</i>	<i>Store / Retailer Section</i>
20000	ali	1	Functional	The page "Choice of supplier" Will be the home page for store users.
20001	ali	1	Functional	On header store logo will be placed.
20002	ali	1	Functional	On left side bar list of supplier logos will appear which will link to supplier page.
20003	ali	1	Functional	A little navigation menu should appear on the top right side of content area of page that will contain links to profile content and order history. This menu should only appear on store home page i.e. "Choice of supplier" page.
20004	ali	1	Functional	"Choice of supplier" page content area will contain "content picture" retrieved from store's profile.
20005	ali	1	Functional	There should appear at most 10 rolling advertisements on the right side bar of store site.
20006	ali	1	Functional	The advertisements that will appear on store site should be retrieved from BrandLink admin's profile.
20007	ali	2	Additional	JQuery plug-in will be used for efficient client side slideshow of advertisements.
20008	ali	1	Functional	Clicking some advertisement will link to advertisement pop-up page.
20010	ali	0	Functional	Selecting Supplier will lead to supplier profile page.
20011	ali	1	Functional	There should appear a tree style navigation on the left side of "supplier" page and it shall be possible to select branches, sub-branches, and articles.
20012	ali	1	Functional	Picture and text posted by supplier will appear on the supplier's page content area.
20013	ali	1	Functional	Only stores with permitted classification A, B, C, D will have access to branches/articles.
20014	ali	1	Functional	Selection of "article" leave from left side tree menu should lead to the page "Order entry".
20015	ali	1	Functional	The button "Kundvagn/Basket" should lead to the page "shopping cart".
20016	ali	1	Functional	"Home" link should lead to the main page "Choice of Supplier".

20017	ali	1	Functional	A search box should appear on the top & it should be possible to search different branches and articles on some supplier's page.
20018	ali	1	Functional	There shall be a button on the top-center of Supplier's page called "Marknad/Market" that will lead to Marketing information page.
20019	ali	1	Functional	The "Order entry" content is listed in description sheet.
20020	ali	1	Functional	It shall be possible to add selected articles to cart by pressing "Add to cart" button on Order entry page.
20021	ali	1	Functional	"View Cart" button should navigate to Cart page.
20023	ali	1	Functional	Only one order should be handled at a time.
20024	ali	1	Functional	A warning message should appear of lost order if supplier is switched or try to log off without checkout completion
20025	ali	1	Functional	Article price should be decreased by specified discount rate.
20026	ali	1	Functional	Articles with discount should have highest priority, then branch & then top level branches.
20027	ali	1	Functional	It will be possible to change, delete ordered items in cart or empty the cart.
20028	ali	1	Functional	User should be able to save changes in cart after pressing "Update Cart" button on Cart page.
20029	ali	1	Functional	User should navigate to "Order entry" page after pressing "add new article" button on Cart page.
20030	ali	1	Functional	Pressing "Empty Cart" will result removal of all items from the cart.
20031	ali	1	Functional	After order submission, there should be sent a confirmation email to store.
20032	ali	1	Functional	Shipping information should be displayed on cash transaction in Cashier page.
20034	ali	1	Functional	On Order History page, user should be able to find earlier orders.

A.3 Admin/ Brandlink side Requirements

<i>Id</i>	<i>Author</i>	<i>Rev</i>	<i>Type</i>	<i>Admin / Brandlink Section</i>
30000	arif	1	Functional	Brand link logo will come on header.
30002	arif	1	Functional	Link to logout.
30003	arif	1	Functional	Administrator profile link will point to "profile" page of admin. In "profile" page personal and account information can also be changed.
30004	arif	1	Functional	Left sidebar will contain navigational tree containing different sub trees from where admin will navigate the whole site.
30005	arif	1	Functional	Supplier sub tree will allow admin to view Suppliers and add /edit supplier accounts.
30006	arif	1	Functional	Stores sub tree will contain the links to view, add, edit or remove a store user.
30007	arif	1	Functional	Order sub tree will contain the links to view and search orders.
30008	arif	1	Functional	Statistics sub tree will link to different types of reports.
30009	arif	1	Functional	Advertise (reklam) sub tree will help to manage advertisements appearing on store site.
30010	arif	1	Functional	Mailing sub tree will link to mail functionality.
30011	arif	1	Functional	Brand link home page content area will be a grid containing Suppliers and Stores.
30012	arif	1	Functional	Paging on home page will allow admin to navigate All supplier or store accounts.
30014	arif	1	Functional	On "modify header" page it will be possible to upload new branlink logo.
30015	arif	1	Functional	Admin will be able to add more suppliers and activate / deactivate existing suppliers.
30016	arif	1	Functional	"Modify supplier" page will allow admin to change profile and account information of suppliers.
30017	arif	1	Functional	Admin will be able to add more stores and activate / deactivate existing stores.
30018	arif	1	Functional	Modify store page will allow admin to change profile and account information of store accounts.
30020	ali	1	Functional	It should appear list of stores in drop down menu that's selection will result store's details.
30022	ali	1	Functional	Search Orders page content is listed in description sheet.
30023	ali	1	Functional	It should be possible to search orders by providing parameters

				like start and end date, store, supplier etc.
30024	ali	1	Functional	Possible statistics reports are listed Orders/store, Orders/supplier and Orders/product.
30025	ali	1	Functional	It should be able to search records for selected statistics type and store type.
30026	ali	1	Functional	Advertisement content type will help admin to easily upload, modify, or delete advertisements.
30027	ali	1	Functional	Advertising page should have 20 fields to select banners/adds. Advertisement will be a movie clip or a still banner (some image).
30028	ali	1	Functional	On Mailing page admin will be able to filter and select specific users, and send mail.
30030	ali	1	Functional	After selecting some option, it should display contact lists according to option.

B.1 File: *Taxonmy_permissions.install*

Install file consists of two functions that execute during module's installation and un-installation. Database query on line 8 alters the existing Drupal table by adding a new column to store permission values. Reverse query is on line 17 to delete column on module un-installation.

```
1 <?php
2
3 /**
4  * Implementation of hook_install().
5  */
6 function taxonomy_permissions_install() {
7
8     db_query("ALTER TABLE {term_data} ADD permission VARCHAR( 1 ) NOT NULL DEFAULT 'D'");
9
10 } // taxonomy_permissions_install
11
12 /**
13  * Implementation of hook_uninstall().
14  */
15 function taxonomy_permissions_uninstall() {
16
17     db_query("ALTER TABLE {term_data} DROP permission");
18
19     cache_clear_all('variables', 'cache');
20 } // taxonomy_permissions_uninstall
21
22 /**
23  * Implementation of hook_schema().
24  */
```

B.2 File: *Taxonomy_permissions.module*

This file describes the implementation of taxonomy permissions. Details about taxonomy permissions can be read from section 4.5. This module file contains three functions that activate on some specific event. The first one, *taxonomy_permission_form_alter()* activates when a web form appears on the screen to add a new taxonomy. It renders an extra field called “Permission” to that form.

Second function *taxonomy_permission_altrfm_submit()* activates when the user submits the form to create a taxonomy. Line 33 & 37 calls the third function *taxonomy_permission_change_child()* to re-calculate permissions for sub-categories and store in the database. Line 46 is a database query that sets permission for each category.

```
1 <?php
2 function taxonomy_permissions_form_alter(&$form, &$form_state, $form_id) {
3     if ($form_id == 'taxonomy_form_term' && $form_state['values']['op'] != "Delete") {
4         $form['identification']['description'] = array(
5             '#type' => 'hidden',
6         );
7         $form['identification']['image']=array(
8             '#type' => 'hidden',
9         );
10        $form['identification']['permission']=array(
11            '#type' =>'select',
12            '#title' => 'Beh&ouml;righet av gren',
13            '#options' => array('A' => 'A','B' => 'B','C' => 'C','D' => 'D'),
14            '#default_value' => isset($form['#term']['permission']) ? $form['#term']['permission'] : 'D',
15        );
16        $form['advanced']['#collapsed'] = FALSE;
17
18        $form['advanced']['weight'] = array(
19            '#type' => 'hidden',
20            '#title' => t('Weight'),
21            '#value' => $edit['weight'],
22            '#required' => TRUE);
23
24        $form['#submit'][] = 'taxomony_permissions_altrfm_submit';
25    }
26 }
27
28 function taxomony_permissions_altrfm_submit($form, &$form_state){
29     global $user;
30     if (in_array('supplier', array_values($user->roles))){
31         if(isset($form['#term']['tid'])){
32             if($form['#post']['permission'])
33                 taxonomy_permissions_change_child($form['#term']['tid'],$form['#post']['permission']);
34         }else {
35             if($form['#post']['permission'){
36                 $sid= db_result(db_query("SELECT tid from {term_data} WHERE name='".$form['#post']['name']."'"));
37                 taxonomy_permissions_change_child($sid,$form['#post']['permission']);
38             }
39         }
40     }
41 }
42 }
43
44 function taxonomy_permissions_change_child($tid,$perm){
45
46     db_query("UPDATE {term_data} SET permission = '".$perm.'" where tid= $tid");
47     $children= taxonomy_get_children($tid, $vid = 0, $key = 'tid');
48     foreach($children as $child){
49         taxonomy_permissions_change_child($child->tid,$perm);
50     }
51 }
52 }
53 >
```

B.3 File: Uc_Catalog.pages.inc

This code is the application of taxonomy and product permission rules. Details about this module can be read from section 4.5. From line 117 to line 131 is a SQL query that gets a filtered list of products according to product & category's permissions and user relationship types.

```
105 $sql_count = "SELECT COUNT(DISTINCT(n.nid))
106 FROM {node} n
107     INNER JOIN {term_node} tn ON n.vid = tn.vid
108     INNER JOIN {uc_products} AS p ON n.vid = p.vid
109 WHERE tn.tid = %d
110     AND n.status = 1
111     AND n.type IN (". db_placeholders($types, 'varchar') .)";
112 }
113 else {
114     $sql = "SELECT DISTINCT(n.nid), c.field_article_id_value, n.sticky, n.title,
115     n.created, p.model, p.sell_price, p.ordering
116 FROM {node} n
117     INNER JOIN {term_node} tn ON n.vid = tn.vid
118     INNER JOIN {term_data} td ON tn.tid = td.tid
119     INNER JOIN {uc_products} AS p ON n.vid = p.vid
120     INNER JOIN {content_type_c01} AS c ON c.nid = p.nid
121     INNER JOIN {user_relationships} AS ur ON (ur.requester_id = n.uid AND
122     ur.requestee_id = ".$user->uid.")
123 WHERE tn.tid = %d AND n.status = 1 AND ((ur.rtid = 2
124     AND (c.field_article_permission_value = 'A' OR c.field_article_permission_value = 'B'
125     OR c.field_article_permission_value = 'C' OR c.field_article_permission_value = 'D'))
126     OR (ur.rtid = 3 AND (c.field_article_permission_value = 'B'
127     OR c.field_article_permission_value = 'C' OR c.field_article_permission_value = 'D'))
128     OR (ur.rtid = 4 AND (c.field_article_permission_value = 'C'
129     OR c.field_article_permission_value = 'D')) OR (ur.rtid = 5
130     AND (c.field_article_permission_value = 'D'))))
131     AND n.type IN (". db_placeholders($types, 'varchar') .) ". $order;
```

B.4 File: *private_catalog.module*

Private catalog module is the code that generates a new catalog for newly generated user and switches it when that user is login. It also makes sure that the catalog is only visible to the particular user or its related customer. Details about the private catalog can be read from section 4.6. This module consists of different functions that activate on particular events, for example on user registration, login, etc.

In function `private_catalog_user()`, there are different conditions. On line 4, if user logins, then it checks either the user already exists or newly registered. If the user is new, it creates a catalog otherwise switch existing catalog for that user. The second condition is on line 48, if some customer is going to view the supplier's page. If the store has a relationship with a supplier, it sets the catalog that belongs to that supplier.

```
1 <?php
2 //Auto catalog creator & switcher!!!
3 function private_catalog_user($op, &$sedit, &$saccount, $category = NULL){
4   if($op == "login"){
5     global $user;
6     if (in_array('supplier', array_values($user->roles))) {
7       $result = db_query("SELECT p.vid FROM {private_catalog} p WHERE p.uid =".$user->uid);
8       if($result->num_rows > 0){
9         $catalog_id=db_result($result);
10        variable_set('uc_catalog_vid', $catalog_id);
11        $result2 = db_query("SELECT v.name FROM {vocabulary} v WHERE v.vid =".$catalog_id);
12        variable_set('uc_catalog_name', db_result($result2));
13      }
14      else {
15        variable_del('uc_catalog_vid');
16        //drupal_set_message(t('You have no catalog!!!'));
17        $vocabulary = array(
18          'name' => t($user->name.'_catalog'),
19          'multiple' => 1,
20          'is_private' => 1,
21          'required' => 1,
22          'hierarchy' => 1,
23          'relations' => 0,
24          'module' => 'forum',
25          'weight' => 0,
26          'nodes' => array('product' => 1, 'c01' => 1),
27        );
28        taxonomy_save_vocabulary($vocabulary);
29        $result2 = db_query("SELECT p.vid FROM {private_catalog} p WHERE p.uid =".$user->uid);
30        $catalog_id=db_result($result2);
31        db_query('UPDATE {private_vocabularies} SET is_private = 1 WHERE vid = %d',$catalog_id);
32        variable_set('uc_catalog_vid', $catalog_id);
33        $result2 = db_query("SELECT v.name FROM {vocabulary} v WHERE v.vid =".$catalog_id);
```

```

34     variable_set('uc_catalog_name', db_result($result2));
35
36     $term = array(
37       'vid' => $catalog_id,
38       'name' => 'Start', // Default term
39     );
40
41     taxonomy_save_term($term);
42
43   }
44 }
45 }
46
47 //Shows right catalog on the right supplier's page
48 else if($op == "view"){
49   global $user;
50   if (in_array('store', array_values($user->roles))) {
51     $uid = $account->uid;
52     variable_set('supplier_id', $uid);
53     $result = db_query("SELECT p.vid FROM {private_catalog} p WHERE p.uid = ".$uid);
54     if($result->num_rows > 0){
55       $catalog_id=db_result($result);
56       variable_set('uc_catalog_vid', $catalog_id);
57       $result2 = db_query("SELECT v.name FROM {vocabulary} v WHERE v.vid = ".$catalog_id);
58       variable_set('uc_catalog_name', db_result($result2));
59     }
60   }
61 }
62 }

```

```

64 //Activates on create vocabulary event!
65 //This function attaches user information with Catalog(vocabulary) created by current user!
66
67 function private_catalog_taxonomy($op, $type, $array = NULL) {
68   global $user;
69   if($type == "vocabulary" && $op == "insert"){
70     $table = 'private_catalog';
71     $record = new stdClass();
72     $record->vid = $array['vid'];
73     $record->uid = $user->uid;
74     drupal_write_record($table, $record);
75   }
76
77 }
78 //
79 //Restricts other catalogs to display in product form.
80 //
81 function private_catalog_db_rewrite_sql($query, $primary_table, $primary_field, $args) {
82   global $user;
83   if (in_array('supplier', array_values($user->roles))) {
84     if ($primary_field == 'vid') {
85       $ret = array();
86       $ret['join'] = "LEFT JOIN {private_catalog} p ON p.vid = v.vid ";
87       $ret['where'] = "p.uid=".$user->uid;
88       return $ret;
89       break;
90     }
91   }
92 }
93
94 ?>

```

B.5 File: search.php

File search module is used to search products. This functionality is shared for both Supplier and Store section. Details about the product search module can be read from section 4.7. This module consists of a single-function `product_search()` to generate a list of filtered products. Line 41 to 54 is a SQL query that fetches a list of products. Line 26 to 39 is an implementation of product permission and relationships.

```
1 <?php
2 function product_search()
3 {
4     // drupal_add_css(drupal_get_path('module', 'uc_catalog') .'/uc_catalog.css');
5     // Build an ORDER BY clause for the SELECT query based on table sort info.
6     if (empty($_REQUEST['order'])) {
7         $order = 'ORDER BY p.ordering, n.title, n.nid';
8     }
9     else {
10        $order = tapirsort_sql(uc_product_table_header());
11    }
12    $output = '';
13    $links = array();
14    $category = (isset($_REQUEST['uc_search_category'])
15    ? trim($_REQUEST['uc_search_category']) : '');
16    if ($category == '0') $category = '';
17    $word = (isset($_REQUEST['uc_search_word']) ? trim($_REQUEST['uc_search_word']) : '');
18    global $user;
19    $is_sup=0;
20    $rem1="";
21    $rem2="";
22    if (in_array('supplier', array_values($user->roles))){
23        $is_sup=1;
24    }
25    else {
26        $rem1="INNER JOIN {user_relationships} AS ur ON (ur.requester_id = n.uid
27        AND ur.requestee_id = ".$user->uid." ) ";
28        $rem2="AND ((ur.rtid = 2 AND (pt.field_article_permission_value = 'A' OR
29        pt.field_article_permission_value = 'B' OR pt.field_article_permission_value
30        = 'C' OR pt.field_article_permission_value = 'D')) OR (ur.rtid = 3 AND
31        (pt.field_article_permission_value = 'B' OR pt.field_article_permission_value
32        = 'C' OR pt.field_article_permission_value = 'D')) OR (ur.rtid = 4 AND
33        (pt.field_article_permission_value = 'C' OR pt.field_article_permission_value
```

```

34     = 'D')) OR (ur.rtid = 5 AND (pt.field_article_permission_value = 'D')));";
35 $rem2 .= "AND ((ur.rtid = 2 AND (td.permission = 'A' OR td.permission = 'B' OR
36     td.permission = 'C' OR td.permission = 'D')) OR (ur.rtid = 3 AND (td.permission
37     = 'B' OR td.permission = 'C' OR td.permission = 'D')) OR (ur.rtid = 4 AND
38     (td.permission = 'C' OR td.permission = 'D')) OR (ur.rtid = 5 AND (td.permission
39     = 'D')));";
40 }
41 $sql = "SELECT DISTINCT (n.nid), td.name, n.sticky, n.title, n.created, p.model,
42     p.sell_price, p.ordering "
43     ."FROM {node} n "
44     ."INNER JOIN {term_node} tn ON (n.vid = tn.vid) "
45     ."INNER JOIN {uc_products} AS p ON (n.vid = p.vid) "
46     ."INNER JOIN {term_data} td ON (tn.tid = td.tid AND td.vid = '"
47     .variable_get('uc_catalog_vid', '1')."') "
48     ."INNER JOIN {content_type_c01} AS pt ON (n.nid = pt.nid AND n.vid =pt.vid) "
49     .$rem1."WHERE (n.status=1 OR 1='".$$is_sup." ) "
50     .($category ? " AND td.name='".$$category.'" " : "")
51     .($sword ? " AND (n.title LIKE '%".$$sword."%' OR pt.field_article_id_value LIKE '%'"
52     .$sword."%') " : "")
53     .$rem2
54     ."AND (n.type='c01' OR n.type='product_kit')";
55 $catalog->products = array();
56 $result = pager_query($sql, variable_get('uc_product_nodes_per_page', 7));
57 while ($node = db_fetch_object($result)) {
58     $catalog->products[] = $node->nid;
59 }
60 if (count($catalog->products)) {
61     if (count($links)) {
62         $output .= theme('links', $links, array('class' => 'links inline uc-categories'))
63         . "<br />\n";
64     }
65     $output .= $catalog->description;
66     $output .= theme('uc_catalog_products', $catalog->products);
67     $output .= theme('pager');
68 }
69 else {
70     $output .= t("No products found");
71 }
72 return $output;
73 }

```


C.1 Relationships Module

```
1 <?php
2 /**
3  * Implementation of hook_form_alter().
4  */
5 function user_form_form_views_exposed_form_alter(&$form, &$form_state) {
6     global $user;
7     if($user->uid==1){
8         if($form['#id'] == 'views-exposed-form-report-store-orders-page-1' ||
9            $form['#id']=='views-exposed-form-report-supplier-orders-page-1' ||
10           $form['#id']=='views-exposed-form-total-per-artikle-page-1') {
11             $uid_field = 'field_uid_value_many_to_one';
12             if($form['#id'] == 'views-exposed-form-report-store-orders-page-1'){
13                 $result = db_query("SELECT DISTINCT user.uid,name FROM {users} as
14                 user LEFT JOIN {users_roles} as urole ON user.uid = urole.uid
15                 WHERE urole.rid=4");
16                 $options[''] = 'All Butik';
17
18                 while($row = db_fetch_array($result)) {
19                     $meraid = $row['uid'];
20                     $meraname= $row['name'];
21                     if($meraid>1)
22                         $options[$meraid] = $meraid." ".$meraname;
23                 }
24                 asort($options);
25                 // update the make field
26                 $form['uid']['#type']='select';
27                 $form['uid']['#size']=0;
28                 $form['uid']['#options'] = $options;
29             }
30             else if($form['#id']=='views-exposed-form-report-supplier-orders-page-1'){
31                 $result = db_query("SELECT DISTINCT user.uid,name FROM {users} as user LEFT
32                 JOIN {users_roles} as urole ON user.uid = urole.uid WHERE urole.rid=3");
```

```

33     $options[''] = 'All Leverantor';
34     while($row = db_fetch_array($result)) {
35         $meraid = $row['uid'];
36         $meraname= $row['name'];
37         if($meraid>1)
38             $options[$meraid] = $meraid." : ".$meraname;
39     }
40     asort($options);
41     $form['uid']['#type']='select';
42     $form['uid']['#size']=0;
43     $form['uid']['#options'] = $options;
44 }
45
46 else if($form['#id']=='views-exposed-form-total-per-artikle-page-1'){
47     $result = db_query("SELECT DISTINCT title FROM {node} WHERE type='c01'");
48     $options[''] = 'All Artikle';
49
50     while($row = db_fetch_array($result)) {
51         $meraname= $row['title'];
52         $options[$meraname] = $meraname;
53     }
54     asort($options);
55     $form['title']['#type']='select';
56     $form['title']['#size']=0;
57     $form['title']['#options'] = $options;
58 }
59
60 // build and sort options array
61
62 $form['#attributes'] = array('autocomplete' => 'off');
63
64 $form['created']['#type'] = 'date_popup';
65
66 $form['created']['#name'] = 'created';
67 $form['created']['#date_format'] = 'Y:m:d';
68 $form['created_1']['#type'] = 'date_popup';
69 $form['created_1']['#name'] = 'created';
70 $form['created_1']['#date_format'] = 'Y:m:d';
71 $form['submit']['#attributes'] = array('onclick'=>'functest()');
72 }
73 }
74 }

```

C.2 Reports Module

```
1 function user_form_nodeapi(&$node, $op, $a3 = NULL, $a4 = NULL) {
2
3     if($node->type=='relation'){
4
5         switch ($op) {
6             case 'validate':
7                 // form_set_error('title', t('Foo is not allowed!'));
8                 /* print '<pre>';
9                 print var_export($node);
10                print '</pre>';*/
11                if($node->delete=='Delete') {
12                    $butik=$a3['field_butik']['#value']['value'];
13                    $sup=$a3['field_leverantor']['#value']['value'];
14                    $rType=$a3['field_relationship']['#value']['value'];
15                    if($rType=='A')
16                        $rtid=2;
17                    else if($rType=='B') $rtid=3;
18                    else if($rType=='C') $rtid=4;
19                    else $rtid=5;
20                    db_query("UPDATE {user_relationships} SET rtid=$rtid WHERE
21                    requester_id=$sup AND requestee_id=$butik");
22                    db_query("UPDATE {user_relationships} SET rtid=$rtid WHERE
23                    requester_id=$butik AND requestee_id=$sup");
24                    /*print '<pre>';
25                    print var_export($node);
26                    print '</pre>';*/
27                }
28            else{
29                $butik=$a3['field_butik']['#value']['value'];
30                $sup=$a3['field_leverantor']['#value']['value'];
31                $rType=$a3['field_relationship']['#value']['value'];
32                if($rType=='A')
```

```

33         $rtid=2;
34     else if($rType=='B') $rtid=3;
35     else if($rType=='C') $rtid=4;
36     else $rtid=5;
37
38     $result = db_result(db_query("SELECT rid from {user_relationships} WHERE
39     requester_id=$sup AND requestee_id=$butik"));
40
41     if($result) {
42         form_set_error('title', t('Relationship Already exists!'));
43     }
44     else{
45         db_query("INSERT INTO {user_relationships} (requester_id, requestee_id,
46         rtid, approved) VALUES (%d, %d, %d, %d)",intval($sup), intval($butik),
47         $rtid, 1);
48         db_query("INSERT INTO {user_relationships} (requester_id, requestee_id,
49         rtid, approved) VALUES (%d, %d, %d, %d)",intval($butik), intval($sup),
50         $rtid, 1);
51     }
52 }
53 break;
54 case 'delete':
55     $butik=$node->field_butik[0]['value'];
56     $sup=$node->field_leverantor[0]['value'];
57     db_query("DELETE FROM {user_relationships} WHERE ((requester_id=$sup AND
58     requestee_id=$butik) OR (requester_id=$butik AND requestee_id=$sup))");
59     break;
60 }
61 }
62 }

```

D.1 Web Server

Different software applications are used to enable a hardware machine to work as a server that fulfills requests from other machines. This project requires a web server application to execute the application and a database server to manage an application's database.

Apache web server was installed on "Web Server1" so that it can host web application. These are some commands to install "Apache". Given command will install HTTP server that will be able to handle HTTP requests.

```
# yum install httpd
```

"HTTPD" is a simple web server used for static websites. To deploy Drupal project, it requires at least PHP 5.2 to run without any error.

These commands are used to download required packages.

```
# rpm -Uvh http://apt.sw.be/redhat/el5/en/i386/rpmforge/RPMS/rpmforge-release-0.3.6-1.el5.rf.i386.rpm
```

```
# rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-4.noarch.rpm
```

```
# rpm -Uvh http://dl.iuscommunity.org/pub/ius/stable/Redhat/5/i386/ius-release-1.0-6.ius.el5.noarch.rpm
```

Command to install PHP 5.3 on HTTP server.

```
# yum install php53u php53u-pear php53u-cli php53u-common php53u-gd
```

```
# yum install php53u-mbstring php53u-mcrypt php53u-mysql php53u-soap
```

```
# yum install php53u-xml php53u-xmlrpc php53u-bcmath
```

After installation, HTTP service can be started using following command

```
#service httpd start
```

After installation these services on "web server1", a folder structure (/var/www/html) appeared that is used to place web application files.

D.2 Database Server

MySQL database management system or server application is used for this project that is fully compatible with Drupal. These are some commands are used to install and turn-on the database server.

```
# yum install mysql-server mysql
#service mysql start
```

D.3 Application deployment

Drupal was running on local machines during development, and it was having all configurations in the local database, it is very important to migrate from database structure and content. Database was exported in SQL format and transferred it to the database server using File Transfer Protocol (FTP) connection. Following steps shows how to export Drupal database structure in a file to deploy on remote database server.

```
# mysql
# use brandlink
#mysqldump brandlink >brandlink.sql
```

After uploading “brandlink.sql” to server, following commands were feed to turn on the MYSQL command line interface and to import database,

```
# mysql
#create database brandlink
# use brandlink
# source brandlink.sql
```

After the successfully migrating database, next step was to transfer all application files on the web server. Following steps shows how to put application files on the web server,

1. Create a new directory under “/var/www/html” with application name.
mkdir brandlink
2. Compress all application files on local machine using any program.
3. Use any FTP to establish connection to server and put compressed project in /brandlink directory.
4. Use PuTTY[x] to establish SSH[x] connection with server.
5. Uncompress project files using following command
unzip ./brandlink.zip

Now all files are transferred successfully, now it is required to configure Drupal so that it knows on which machine its database is. Drupal configuration file can be found in /sites/default/. Following string can be written in “settings.php” file to register database server to Drupal,

```
$db_url = 'mysql://root:root@192.168.4.16/brandlink';
```

After configuring and putting everything, now it was time to make some final test either it is accessible via internet using this url, [<http://213.136.63.111/brandlink/>].