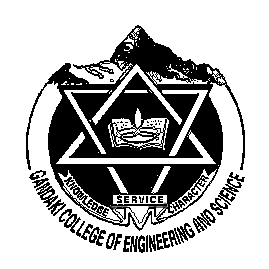
A Major Project Report on  
**SMART HOTEL**

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Software Engineering at Pokhara University

***By*  
ABISH GURUNG  
BAMDEV BHANDARI  
BISHNU SHARMA**

****

**Department of Research and Development  
GANDAKI COLLEGE OF ENGINEERING AND SCIENCE**  
Lamachaur, Kaski, Nepal

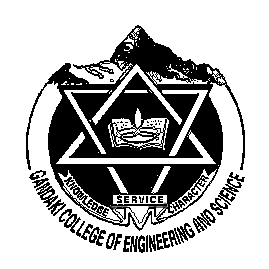
**(October, 2017)**

A Major Project Report on  
**SMART HOTEL**

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Software Engineering at Pokhara University

***By*ABISH GURUNG  
BAMDEV BHANDARI  
BISHNU SHARMA**

***Supervisor*Er. HARI KC**

****

**Department of Research and Development  
GANDAKI COLLEGE OF ENGINEERING AND SCIENCE**  
Lamachaur, Kaski, Nepal

**(October, 2017)**

**ACKNOWLEDGEMENT**

We would like to express our deepest appreciation to all those who provided us the possibility to complete this project. A special gratitude we would like to give to our project head Er. Sujan Tamrakar, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report.

Furthermore, we would like also like to acknowledge with much appreciation the crucial role of our project supervisor Er. HARI KC who helped us in the each and every step of our project, we really enjoyed his involvement in our project. We would also like to thank the whole administration of GCES for giving us the permissions to access the required information for our project. We have to appreciate the guidance given by other supervisors as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices.

ABISH GURUNG   
BAMDEV BHANDARI   
BISHNU SHARMA

Gandaki College of Engineering and Science   
Lamachaur, Pokhara

ABSTRACT

“Smart Hotel” is an E-commerce site for browsing, searching and buying Nepali local goods and materials. This Project aims to promote traditional and modern goods and materials that are produced locally in Nepal.

No login or user authentication is required to view or search products. Any normal guest user can land on our portal and view items and products. He/she can search different products and can even add them to the cart. They can add a quantity of a product or even reduce or remove the entire item from the cart. In order to buy a product, the user needs to be authenticated. New guest user needs to register to our system while the returning customer can just sign in and checkout. Before Checkout, the customer need to enter his/her credit card credentials which is verified and validated by using stripe in our system. The customer/logged-in user can the view his/her order history from the user account dashboard.

Use of Access Control List has allowed the super admin to control, assign or revoke roles and permissions to or from any user. A user can be granted a Merchant role if he/she desires to sell products using our system. The Merchant can CRUD products and items.

Super Admin is responsible for web administration part. The super admin is responsible for assigning or revoking user roles and permissions. Super admin is responsible for overall management of the wholes system.

# TABLE OF CONTENTS

[TABLE OF CONTENTS iii](#_Toc467257131)

[LIST OF FIGURES vi](#_Toc467257132)

[LIST OF TABLES ix](#_Toc467257133)

[LIST OF ABBREVIBIATION x](#_Toc467257134)

[Chapter 1 INTRODUCTION 1](#_Toc467257135)

[1.1. BACKGROUND 1](#_Toc467257136)

[1.2. PROBLEM STATEMENT 3](#_Toc467257137)

[1.3. OBJECTIVES 3](#_Toc467257138)

[1.3.1. Primary Objectives 3](#_Toc467257139)

[1.3.2. Secondary Objectives 4](#_Toc467257140)

[1.4. IMPLICATION 5](#_Toc467257141)

[Chapter 2 LITERATURE REVIEW 6](#_Toc467257142)

[Chapter 3 TOOLS AND METHODOLOGY 10](#_Toc467257143)

[3.1. REQUIRED TOOLS 10](#_Toc467257144)

[3.1.1 Language 10](#_Toc467257145)

[3.1.2 Framework 11](#_Toc467257146)

[3.1.3 Software 11](#_Toc467257147)

[3.1.4 Version Control and Project Management 13](#_Toc467257148)

[3.2. ALGORITHM OR SOLUTION APPROACH USED 13](#_Toc467257149)

[3.2.1 Role base access control 13](#_Toc467257150)

[3.2.2 Business to business model 15](#_Toc467257151)

[3.2.3 Business to consumer model 16](#_Toc467257152)

[Chapter 4 SYSTEM ANALYSIS AND DESIGN 18](#_Toc467257153)

[4.1 Use Case and Operation Contracts 18](#_Toc467257154)

[4.2 Activity Diagram 30](#_Toc467257155)

[4.3 Domain Model 37](#_Toc467257156)

[4.4 Entity-Relation (ER) Diagram 38](#_Toc467257157)

[4.5 System Sequence Diagrams 39](#_Toc467257158)

[4.6 Design Class Diagram 42](#_Toc467257159)

[4.7 Interaction Diagram (Sequence Diagrams) 43](#_Toc467257160)

[4.8 UI Designs 51](#_Toc467257161)

[Chapter 5 IMPLEMENTATION 57](#_Toc467257162)

[5.1 Environment Configuration 62](#_Toc467257163)

[5.2 Components of Laravel 63](#_Toc467257164)

[5.3 File Structure 64](#_Toc467257165)

[Chapter 6 TESTING 66](#_Toc467257166)

[6.1 Scope 66](#_Toc467257167)

[6.2 Test Objectives 66](#_Toc467257168)

[6.3 Testing Goals 67](#_Toc467257169)

[6.4 What were tested 67](#_Toc467257170)

[6.5 Entrance Criteria 68](#_Toc467257171)

[6.6 Exit Criteria 68](#_Toc467257172)

[6.7 Test Execution 68](#_Toc467257173)

[6.8 Test Result 71](#_Toc467257174)

[Chapter 7 RESULT AND DISCUSSIONS 76](#_Toc467257175)

[Chapter 8 FUTURE IMPROVEMENTS 77](#_Toc467257176)

[Chapter 9 CONCLUSION 78](#_Toc467257177)

[Bibliography 79](#_Toc467257178)

[APPENDIX I USER INTERFACE 81](#_Toc467257179)

[APPENDIX II SAMPLE CODE SNIPPETS 94](#_Toc467257180)

# LIST OF FIGURES

[Figure 2- 1: Ecommerce Trends in Nepal. 8](#_Toc467253536)

[Figure 2- 2: E-commerce Market growth in Nepal by Year. 9](#_Toc467253537)

[Figure 4- 1: use case diagram for the user and system interaction. 18](#_Toc467253538)

[Figure 4- 2: use case diagram for Super Admin. 21](#_Toc467253539)

[Figure 4- 3: use case diagram for Merchant. 24](#_Toc467253540)

[Figure 4- 4: use case diagram for Registered User 27](#_Toc467253541)

[Figure 4- 5: Activity Diagram for User Registration. 30](#_Toc467253542)

[Figure 4- 6: Activity Diagram for User Login. 31](#_Toc467253543)

[Figure 4- 7: Activity Diagram for Shopping. 32](#_Toc467253544)

[Figure 4- 8: Activity Diagram for Merchant. 33](#_Toc467253546)

[Figure 4- 9: Activity Diagram for Super Admin User Management 34](#_Toc467253547)

[Figure 4- 10: Activity Diagram for Super Admin User Management. 35](#_Toc467253549)

[Figure 4- 11: Activity Diagram for Super Admin User Roles Management. 36](#_Toc467253550)

[Figure 4- 12: Domain Model. 37](#_Toc467253551)

[Figure 4- 13: ER Diagram. 38](#_Toc467253552)

[Figure 4- 14: SSD for Super Admin. 39](#_Toc467253553)

[Figure 4- 15: SSD for Merchant. 40](#_Toc467253554)

[Figure 4- 16: SSD User(Customer). 41](#_Toc467253555)

[Figure 4- 17: Design Class Diagram. 42](#_Toc467253556)

[Figure 4- 18: Sequence Diagram for User Registration. 43](#_Toc467253557)

[Figure 4- 19: Sequence Diagram for User Login. 44](#_Toc467253558)

[Figure 4- 20: Sequence Diagram for Shopping. 45](#_Toc467253559)

[Figure 4- 21: Sequence Diagram of Merchant to add Product. 46](#_Toc467253560)

[Figure 4- 22: Sequence Diagram of Merchant to Update Product. 47](#_Toc467253561)

[Figure 4- 23: Sequence Diagram of Merchant to Delete Product. 48](#_Toc467253562)

[Figure 4- 24: Sequence Diagram of Super Admin to view or delete user. 49](#_Toc467253563)

[Figure 4- 25: Sequence Diagram of Super Admin to assign/Revoke Roles. 50](#_Toc467253565)

[Figure 4- 26: UI Wireframe of Home Page 51](#_Toc467253566)

[Figure 4- 27: UI Wireframe of User Registration Page 52](#_Toc467253567)

[Figure 4- 28: UI Wireframe of Login page. 53](#_Toc467253568)

[Figure 4- 29: UI Wireframe Product Detail Page 54](#_Toc467253569)

[Figure 4- 30: UI Wireframe of Shopping-Cart Page. 55](#_Toc467253571)

[Figure 4- 31: UI Wireframe of Checkout Page. 56](#_Toc467253572)

[Figure 5- 1: Code Snippets of. env file. 62](#_Toc467253573)

[Figure 5- 2: Components of Laravel. 63](#_Toc467253574)

# LIST OF TABLES

TABLE 5-1 Laravel File Structure........................................................…65

TABLE 6-1 Test Results.......................................................................…75

# LIST OF ABBREVIBIATION

CRUD = Create Read Update Delete

RDBMS = Relational Database Management System

RGB = Red Blue Green

CMYK = Cyan Magenta Yellow Key

ER = Entity Relationship

SSD = System Sequence Diagram

UML = Unified Modeling Language

ACL = Access Control List

RBAC = Role Based Access Control

B2B = Business to Business

B2C = Business to Customers

JCB = Japan Credit Bureau

CoD = Cash on Delivery

IoC = Inversion of Control

# Chapter 1 INTRODUCTION

# BACKGROUND

Nepalese handicraft history can be traced back to the Stone Age when human beings were inadequate of tools of any kind. Hence, we have lot of religious and traditional handicrafts in Nepal. In Nepal, the production of handicraft is an age-old practice. For the last 25–30 years, export of handicrafts has been growing. The development of handicraft helps the conservation of national heritage and culture of country. The handicrafts of Nepal are produced in a traditional way, from generations to generations leading the footpath of ancestors or from forefather to grandfather to father and to son and this continuity has given the survival to Nepalese handicrafts, preserving their heritage, cultural values, aspects and tradition. More recently, these arts and crafts is one of the major exporting industry of Nepal, earning foreign exchange and providing employment to thousands of Nepalese craftsmen, artisans, promoters and businessmen generating revenue to government.

In spite of such a revenue generating source, all we lack is proper promotion of those handicrafts and tools among the mass. We are unaware of the possible handicrafts, their beauty and importance of those materials that have been produced locally in our remote villages. Even we Nepalese people are unaware of all the products and tools that have been locally prepared. So, proper promotion and managed way to get those handicrafts would lead to not only promotion of Nepalese traditional tools and handicrafts but also conservation of our national heritage and culture. This can be done with proper promotion and a better way to get those products. Smart Hotel might be the medium to provide proper promotions and easy access to get those products.

The world now relies on internet. E-commerce is now a major part of internet. Major countries and companies have started buying and selling goods through internet. In our context, internet is the best possible method to sell our traditional Nepalese handicrafts to foreigner.

There are many e-commerce websites in Nepal like muncha.com, rojeko.com, metrotarkari.com, foodmandu.com and many more. Muncha.com is popular for buying birthday cakes, flowers, gifts etc. rojeko.com is mostly focused on electronic items. Metrotarkari.com is popular for delivery of vegetables at your doorstep and foodmandu.com provides service, which delivers food from different restaurants in Kathmandu and Lalitpur to your doorstep. But there is hardly a website that will help in buying Nepali traditional handicrafts and tools. Smart Hotel is the one that help one to get the handmade tools from remote villages that reflects the heritage and culture of Nepal on one click.

“SMART HOTEL” is an E-commerce site for browsing, searching and buying Nepali local goods and materials. This Project aims to promote traditional and modern goods and materials that are produced locally in Nepal. The basic objective of this project is to ease the process of selling and buying local goods through an online portal. The project itself acts as a middle-man by creating a link of business between the local sellers and buyers.

# PROBLEM STATEMENT

As Nepal is an under developed country, people in some parts of Nepal are even deprived of facilities such as electricity, telephone communication. Some places like Dolpa and Rolpa don’t have good transportation systems and don’t have direct routers to main cities of Nepal. Products of rural areas and villages who don’t even have reach to the communication facilities remains undiscovered. There are lots of high quality goods that are produced even in those rural areas which are of great value. They just remain undiscovered due to the lack communication facilities.

For a developing country like Nepal, this whole process might seem a little complex and difficult to implement. There might be some conditions where some people might not want to get their products exposed to the public. The process of collecting all these local stuffs from a versatile community of ours might add some challenges to our project. But someone has to start and why not us. It might take some time to aware the remote people about the e-commerce practice and the whole phenomena.

# OBJECTIVES

## Primary Objectives

* To build a feature rich e-commerce Website for buying especially Nepali Goods.
* To provide efficient management of users divided in 3 general categories as Logged-in User, Merchant and Super Admin.
* To enable proper management and authentication of users, their roles and permissions using ACL and RBAC.
* To provide simple search of products to increase the availability and discoverability of different varieties of Nepalese goods.
* To provide efficient and proper products sorting for better products recommendations to user/customers.
* To provide an easy and hassle free buy/sell process to enhance ecommerce B2B and B2C methodologies.
* To enable email verification of user account to get started to avoid any spams and wastages in the system.
* To enable better and secure account information recovery options to the users through their own email used during registration.
* To provide easy and secure payment options using Credit cards accepting Visa, MasterCard, American Express, JCB, Discover, Diners Club, and even CoD.
* To enable customers to report to low quality or random unrelated products for better user experience and quality improvement of the system.

## Secondary Objectives

* To enable local people and Nepalese people to sell and promote their locally made products through our merchant-ship program.
* To provide a platform itself being a middle-man by creating a link of business between the local sellers and buyers.
* To enable users/customers to select from a wide selection range of products.

# IMPLICATION

This Project aims to promote traditional and modern goods and materials that are produced locally in Nepal. There are already a couple of e-commerce site for buying and selling goods and services in Nepal. But Smart Hotel would be a first in its kind and a unique portal to mainly focus on the sale and promotion of Nepali goods and products. For a developing country like ours where the literacy rate is below the average, this project might help the local farmers, villagers and the local goods producer to better get their product onto a platform that really is built for one and only them.

There is a great necessity of such portals that would collaborate to reduce the import ratio and promote our own goods and products. This project aims to provide a platform where people can view, search and buy locally made products at a very effective and easy manner without any hard processes and tedious payment gateways. So, it must be a great concern to build such a product like this project.

# Chapter 2 LITERATURE REVIEW

Ecommerce is still immature in Nepal. Even though the website like Kaymu.com.np, Meroshopping.com etc. are doing well, they still lack the public exposure. Buying something from Internet is myth in Nepal. People still afraid to do it. In the past 10 years e-commerce have totally changed. Different payment gateways have start evolving. Companies have started investing millions of dollars in security and user’s experience. Nepal is still immature in all these kinds of stuff. Majority of Ecommerce website provides service to only big cities like Kathmandu and Pokhara. Nepalese people still don’t’ trust most of those companies.

Some of the above-mentioned website deals with the selling of all the goods which are easily available in many of the store in the cities and their targeted users are only of the big cities.

Nepali Product Exporter is an online order portal with a difference which is a manufacturer and exporter of different kinds of Nepali products at wholesale price like; Statue, Thankas, Pashmina, Herbal, Carpet (Nepali and Tibetan), jewelry, Handmade Paper Items, Wooden Crafts, Nepali Tea, Metal Crafts, Stone Crafts, Garments product and Buddhist item including prayer flag, stupas and incense. This site provides the payment gateway which is immature in country like Nepal and most people are far away from this payment gateway.

Nepal Art Shop Export & Import (P) Ltd. is an online shopping portal with showroom located in the heart of the Thamel Kathmandu, is full with various handicraft products? This also deals with selling of products same as of Nepali Product Exporter. This site claims to deliver goods to you no matter where you reside on any corner of the world. (Shop, 2007)

Meroshopping.com is the super digital shopping mall that is owned and managed by highly expert team of professionals which are committed to give you best shopping experience. Their prime objective is to start online shopping trend in Nepal and increase online shopping habit in Nepal. They are known in online shopping platform for providing quality products in cheap price. The goods that they deliver will be on time every time. Shopping online at meroshopping.com is fast and easy. They use latest technology to provide you with the secured shopping. They also provide you with the best after sales service which makes them the fastest growing online shopping website in Nepal.

According a research done by a student of Irish American University, showed that there are various benefits from ecommerce as 100% of the respondents mentioned global coverage, 88.2% respondent mentioned new market entry, 44.4% of the respondents have mentioned instant services which are the benefits of ecommerce. The findings of the research also showed that there are various challenges to adopt ecommerce in travel and tourism as 59.3% of the respondents agreed lack of trust & resistance to change as well as lack of skilled human resources are major challenge. Similarly, 51.9% of respondents have mentioned that consumers are less loyal these days and need better services and rates to attract them. The findings of the research showed that traditional way of doing business is still accepted in context of Nepalese tourism industry however 47.5% of respondent mentioned that ecommerce will surpass traditional business in future. (Khan)

Also, an article from a group of students, stated that a business-consumer site for marketing some handicraft item or items such as Buddhist thangka paintings via the Internet would be a great option in context of development of Nepalese Information and technology sector. (Larry Press)

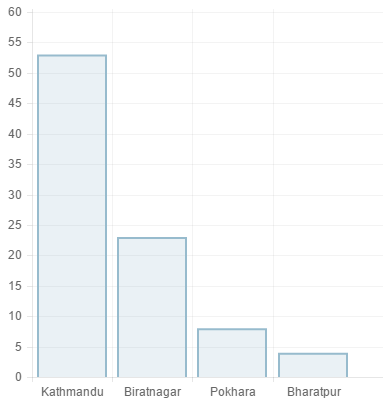
Similarly, a study from one of the popular International Ecommerce site in Nepal Kaymu.com showed that Kathmandu, being the capital city and the largest municipality of Nepal, prevails the list of yearly e-commerce activity in the country with 53%, closely followed by sub-metropolitan municipality and third largest city of Nepal i.e. Biratnagar with 23%. Pokhara, the sub-metropolitan municipality and the second largest city of Nepal stands at third place with 8% of the country’s total e-commerce activity each year. The city of Bharatpur, located in Chitwan Valley with 4% of the country's total e-commerce activity, remains the fourth city of Nepal with highest inclination towards e-commerce. (Kaymu.com.np, 2015)

Figure 2- 1: Ecommerce Trends in Nepal.

Also, the research showed that, Nepali E-Commerce market is expected to achieve more than 80% growth in e-commerce transactions in the near future. This consensus is based on 10% exponential growth in e-commerce transactions for each passing quarter. (Larry Press)

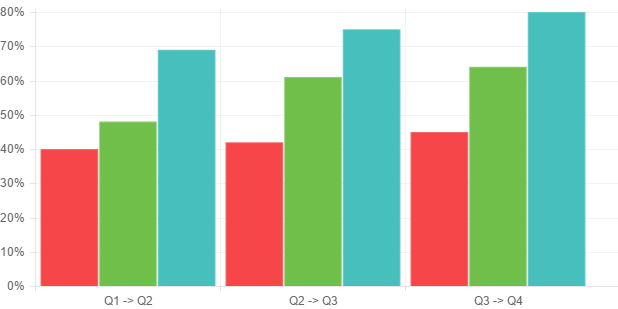
****

Figure 2- 2: E-commerce Market growth in Nepal by Year.

The various above studies and reviews from different sites and papers can depict challenges of E-commerce in Nepal. Still they are in the favor ofa positive growth of Ecommerce sector in Nepal and the emerging opportunities gifted by the ecommerce practices in Nepal.

# Chapter 3 TOOLS AND METHODOLOGY

# 3.1. REQUIRED TOOLS

The system is developed using the following tools:

## Language

* **JavaScript**

JavaScript is a high-level, dynamic, untyped, and interpreted programming language which is used alongside HTML and CSS. JavaScript is easy to learn easy to edit and prototyping language easy to debug object oriented scripting language which allows you to create highly responsive interfaces that improve the user experience and provide dynamic functionality, without having to wait for the server to react and show another page. (wikipedia, https://en.wikipedia.org/wiki/JavaScript, 2016)

* **MySQL (Database)**

MySQL is an open-source relational database management system and the most widely used open-source client–server model RDBMS. MySQL has one major advantage, since it is free, it is usually available on shared hosting packages and can be easily set up in a Linux, UNIX or Windows environment.

* **PHP**

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages. PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP. It is a great option for many reasons like Fast Load Time, Less Expensive Software, Less Expensive Hosting, Database Flexibility, and Increased Available Programming Talent.

## Framework

* **Laravel**

Laravel is current best web application PHP framework with advanced query syntax that makes web development simple and rapid by enabling general tasks that will be used in the majority of web projects such as route, queue, sessions, caching and authentication.

* **Bootstrap**

Bootstrap is a free and open-source front-end library for creating websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. It aims to ease the development of dynamic websites and web applications.

## Software

* **WAMP**

WAMP is an archetypal model of web service solution stacks, named as an acronym of the names of its original four components: The **W**indows operating system, the **A**pache HTTP Server, the **M**ySQL relational database management system (RDBMS), and the **P**HP programming language. The WAMP components are largely interchangeable and not limited to the original selection. As a solution stack, WAMP is suitable for building dynamic web sites and web applications.

* **JetBrains PhpStorm**

JetBrains PhpStorm is a commercial, cross-platform IDE for PHP is built on JetBrains' IntelliJ IDEA platform.

PhpStorm provides an editor for PHP, HTML and JavaScript with on-the-fly code analysis, error prevention and automated refactorings for PHP and JavaScript code. PhpStorm's code completion supports PHP 5.3, 5.4, 5.5, 5.6 & 7.0 (modern and legacy projects), including generators, coroutines, the finally keyword, list in foreach, namespaces, closures, traits and short array syntax. It includes a full-fledged SQL editor with editable query results.

PhpStorm is built on IntelliJ IDEA, which is written in Java. Users can extend the IDE by installing plugins created for the IntelliJ Platform or write their own plugins. (JetBrains, 2016)

* **Adobe Photoshop**

Adobe Photoshop is a raster graphics editor for Windows and OS X. It can edit and compose raster images in multiple layers and supports masks, alpha compositing and several color models including RGB, CMYK, Lab color space, spot color and duotone.

## Version Control and Project Management

* **Git and Gitlab**

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows

# 3.2. ALGORITHM OR SOLUTION APPROACH USED

## Role base access control

 Role-based access control (RBAC) is an approach to restricting system access to authorized users. It is used by the majority of enterprises and can implement [mandatory access control](https://en.wikipedia.org/wiki/Mandatory_access_control) (MAC) or [discretionary access control](https://en.wikipedia.org/wiki/Discretionary_access_control) (DAC). RBAC is sometimes referred to as role-based security.

Role-based-access-control (RBAC) is a policy neutral access control mechanism defined around roles and privileges. The components of RBAC such as role-permissions, user-role and role-role relationships make it simple to perform user assignments. RBAC can be used to facilitate administration of security in large organizations with hundreds of users and thousands of permissions. Although RBAC is different from MAC and DAC access control frameworks, it can enforce these policies without any complication. Its popularity is evident from the fact that many products and businesses are using it directly or indirectly.

Within an organization, [roles](https://en.wikipedia.org/wiki/Role_(computer_science)) are created for various job functions. The permissions to perform certain operations are assigned to specific roles. System users are assigned particular roles, and through those role assignments acquire the computer permissions to perform particular computer-system functions. Since users are not assigned permissions directly, but only acquire them through their role (or roles), management of individual user rights becomes a matter of simply assigning appropriate roles to the user's account; this simplifies common operations, such as adding a user, or changing a user's department. (Wikipedia, 2016)

Three primary rules are defined for RBAC:

Role assignment: A subject can exercise a permission only if the subject has selected or been assigned a role.

Role authorization: A subject's active role must be authorized for the subject. With rule 1 above, this rule ensures that users can take on only roles for which they are authorized.

Permission authorization: A subject can exercise a permission only if the permission is authorized for the subject's active role. With rules 1 and 2, this rule ensures that users can exercise only permissions for which they are authorized.

Additional constraints may be applied as well, and roles can be combined in a [hierarchy](https://en.wikipedia.org/wiki/Hierarchy) where higher-level roles subsume permissions owned by sub-roles.

## Business to business model

Business to business, also called B to B or B2B, is a type of transaction that exists between businesses, such as one involving a manufacturer and wholesaler, or a wholesaler and a retailer. Business to business refers to business that is conducted between companies, rather than between a company and individual consumers. A typical [supply chain](http://www.investopedia.com/terms/s/supplychain.asp) involves multiple business to business transactions, as companies purchase components and products such as other [raw materials](http://www.investopedia.com/terms/r/rawmaterials.asp) for use in the manufacturing processes. Finished products can then be sold to individuals via business to consumer transactions. In the context of communication, business to business refers to methods by which employees from different companies can connect with one another, such as through [social media](http://www.investopedia.com/terms/s/social-media.asp). This type of communication between the employees of two or more companies is called B2B communication. (investopedia.com, http://www.investopedia.com/terms/b/btob.asp, 2015)

The internet provides a robust environment in which businesses can find out about products and services and lay the groundwork for future business to business transactions. Company websites allow interested parties to learn about a business's products and services and initiate contact. Online product and supply exchange websites allow businesses to search for products and services and initiate procurement through e-procurement interfaces. Specialized online directories providing information about particular industries, companies and the products and services they provide also facilitate business to business transactions.

## Business to consumer model

B2C, or business-to-consumer, is the type of commerce transaction in which businesses sell products or services to consumers. Traditionally, this could refer to individuals shopping for clothes for themselves at the mall, diners eating in a restaurant, or subscribing to pay-per-view TV at home. More recently, the term B2C refers to the online selling of products, or e-tailing, in which manufacturers or retailers sell their products to consumers over the Internet.

The mid-1990s to the 2000s saw the rise of e-commerce through sites like Amazon, Zappos and Victoria's Secret. Now, it's rare to see a consumer-based business not sell their products online. Consumers enjoy the convenience of online shopping in their own homes, while businesses thrive on the low overhead. With a virtual storefront, a business doesn't need a storefront or a large inventory stocked at all times. This is ideal for small businesses, like a jewelry company or a bakery.

There are challenges for businesses in B2C, however. As websites continue to become flashier and more user-friendly, it's up to the business to keep their site intuitive and easy to navigate. The site must also be optimized to get consumer traffic — search engine marketing (SEM) is a necessity. Most consumers use search engines like Google, Bing and Yahoo! to find the products that they are looking to purchase. Customers generally choose websites on the first few pages of results after they've searched a specific keyword or phrase. If a site does not have a site with good SEM, they could get buried in the mix, lose site traffic, and thus lose potential customers.

Another challenge is the payment processing. SSL encryption lets people know that the site isn't compromised, but many people are hesitant to submit their credit cards to companies. Even if the site is safe, the place where the credit card numbers are stored is not. In 2004, the Payment Card Industry Security Standards Council (PCI) formed to create compliance standards for any company processing credit cards. Services like PayPal can perform the payment processing for online vendors, and has proven to be popular with online shoppers and businesses.

# Chapter 4 SYSTEM ANALYSIS AND DESIGN

## Use Case and Operation Contracts

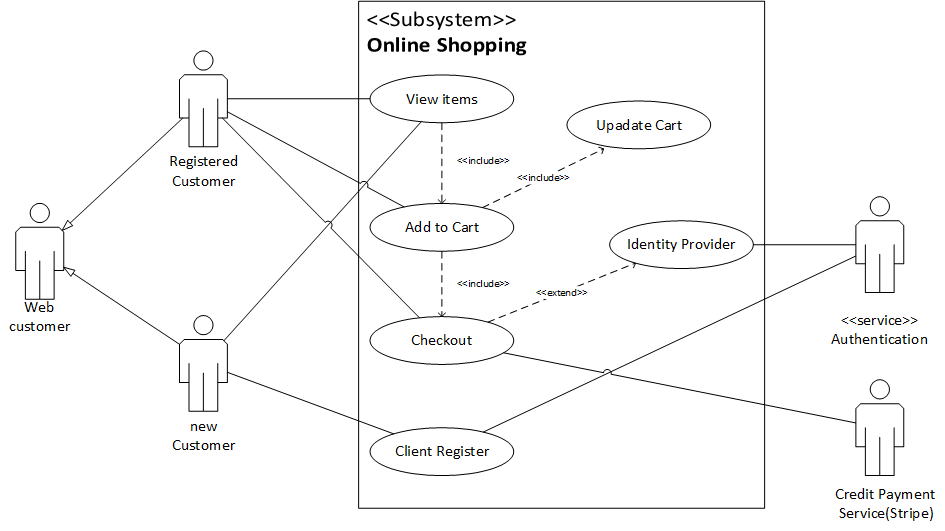
The overall system consists of a Super Admin, any number of Merchants and the general customers. The customer interacts with the system in two ways; either as a web customer or a registered customer. The web customer can view and browse the system and can register to get the other access to the system. The registered customer has a couple more access and permissions than the general web customer/user. The registered customer can view items, make purchase and check out to confirm their purchase. The system authenticates the registered users and provide them with access and permissions. The overall process can be depicted in the use case diagram below.

Figure 4- 1: use case diagram for the user and system interaction.

The System Operation Contracts for above use case are depicted below:

**Operation:** viewItems ()

**Cross references:** use-case diagram for the user and system interaction

**Preconditions:** user browse the Page

**Post conditions:** list of items are displayed to the user

**Contract CO1: viewItems**

**Operation:** addToCart()

**Cross references:** use-case diagram for the user and system interaction

**Preconditions:** user visits the store

**Post conditions: -**user adds item to be purchased to the cart

-cart gets updated

**Contract CO2: addToCart**

**Operation:** checkout ()

**Cross references:** use-case diagram for the user and system interaction, use-case diagram for user

**Preconditions:** user finishes purchase

**Post conditions:** none

**Contract CO3: checkOut**

**Operation:** clientRegister ()

**Cross references:** use-case diagram for the user and system interaction

**Preconditions:** user visits the web page

**Post conditions:** -user inputs their information into the system

-new id for client is created

**Contract CO4: clientRegister**

* **Super Admin**

Super Admin owes the overall control and responsibility to the system. The super admin has the prime right and access to the system. A super admin can add or remove features to the project or even has the complete right to discontinue or destroy the project. A super admin can add new merchants or remove existing merchants. He/she can permit necessary rights and authorities control to the merchants. Super admin can even add, remove, verify or modify products, categories and users. The overall super Admin panel can be represented in the use case shown below.

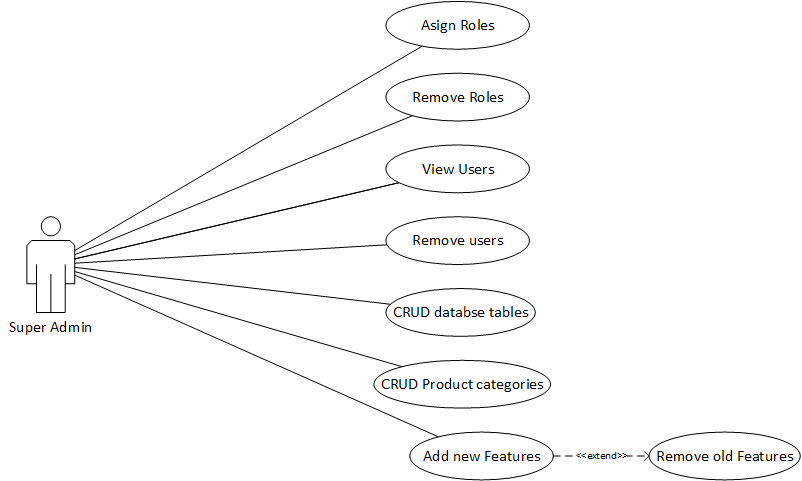
****

Figure 4- 2: use case diagram for Super Admin.

The System Operation contracts for Super Admin Use case are shown below:

**Operation:** assignRoles ()

**Cross references:** use-case diagram Super Admin

**Preconditions:** must be super admin of the system

**Post conditions:** -assigns or revoke roles and permissions to or form users

**Contract CO5: assignRoles**

**Operation:** viewUsers ()

**Cross references:** use-case diagram Super Admin

**Preconditions:** must be super admin of the system

**Post conditions:** -search users from database view their details

-removes user

**Contract CO6: viewUsers**

**Operation:** crudDbTables ()

**Cross references:** use-case diagram Super Admin

**Preconditions:** must be super admin of the system

**Post conditions:** -Create/Read/Update/Delete table to and from database

**Contract CO7: CRUD database tables**

**Operation:** crudProductCategories ()

**Cross references:** use-case diagram Super Admin

**Preconditions:** must be super admin of the system

**Post conditions:** - Create/Read/Update/Delete product categories to and from the product table

**Contract CO8: CRUD product Categories**

**Operation:** addNewFeatures ()

**Cross references:** use-case diagram Super Admin

**Preconditions:** must be super admin of the system

**Post conditions:** - add new features in the system

- remove existing features from the system

**Contract CO9: addNewFeatures**

* **The Merchant**

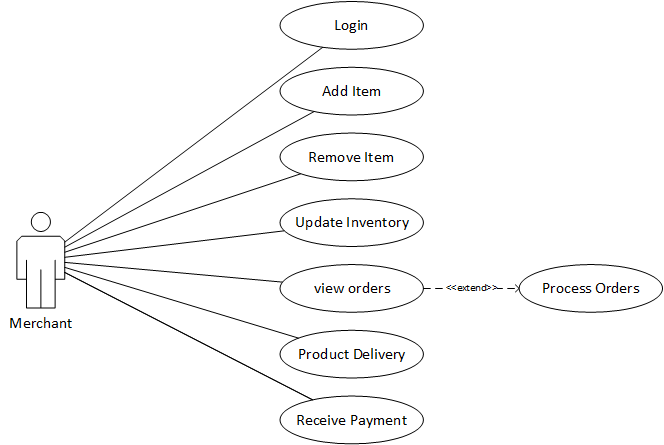
The Merchant is another actor of this project. Whenever the super admin creates a unique Merchant, he/she is also assigned a set of authorities and access to the system. The merchants are the people who runs small shops or firm with the goods that met the requirements of our site. A merchant can also be an individual person who locally produces and build Nepali goods and products. A merchant has the access to the system by logging in. The merchant can add item along with item details, remove items or modify products. Merchants also delivers the product ordered by the customer and receive payment from them and update the inventory item.

Figure 4- 3: use case diagram for Merchant.

The System Operation Contracts for Merchant Use case are shown below:

**Operation:** addItem ()

**Cross references:** use-case diagram Merchant

**Preconditions:** merchant logged into their account

**Post conditions:** item was added

**Contract CO10: addItem**

**Operation:** removeItem ()

**Cross references:** use-case diagram Merchant

**Preconditions:** item was added

**Post conditions:** item was removed

**Contract CO11: removeItem**

**Operation:** updateInventory ()

**Cross references:** use-case diagram Merchant

**Preconditions:** -product was sold

-product was expired

**Post conditions:** changes were made in the inventory

**Contract CO12: updateInventory**

**Operation:** viewOrder ()

**Cross references:** use-case diagram Merchant

**Preconditions:** -product(s) was ordered by customer

**Post conditions:** orders were viewed and processed

**Contract CO13: viewOrders**

**Contract CO14: deliverProducts**

**Operation:** deliverProducts ()

**Cross references:** use-case diagram Merchant

**Preconditions:** -order was processed

**Post conditions:** - customer order was placed for delivery

* **The Registered User**

Users can browse as guest on the site or register and login to get access to most of the features. A registered user can login into the system, search for items or categories. The logged in user can view a particular item and add to his/her cart or remove previously item added to the cart. He/she can checkout of the system to get the total billing amount and logout of the system. The user does not have any permissions and authority to add or change any feature the system apart from buying and browsing items and products.

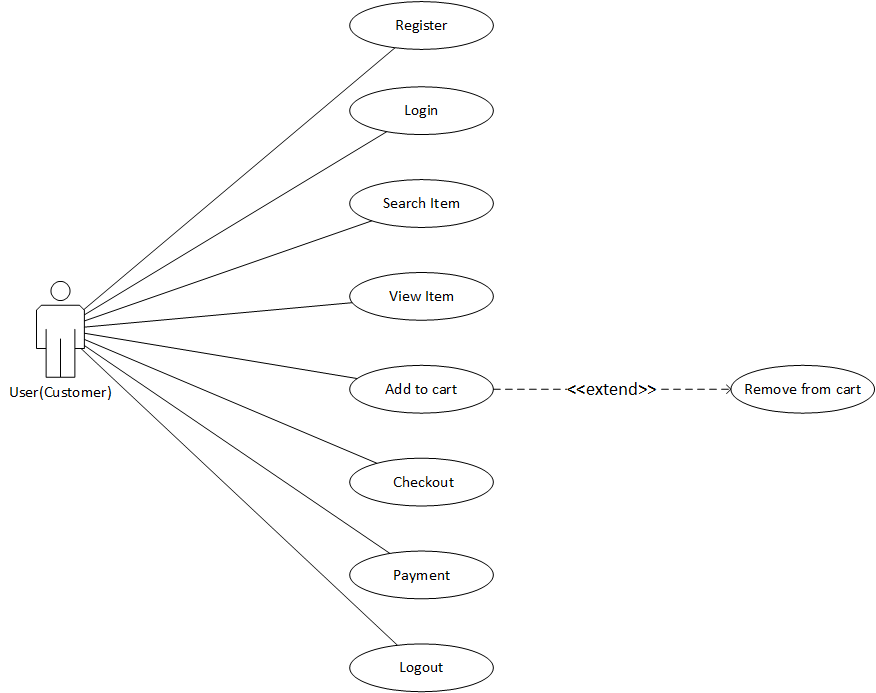
****

Figure 4- 4: use case diagram for Registered User

The System Operation Contracts for User use case are shown below

**Operation:** register ()

**Cross references:** use case diagram user

**Preconditions:** none

**Post conditions:** new user was created

**Contract CO15: register**

**Operation:** login ()

**Cross references:** use case diagram user

**Preconditions:** a new user was created

**Post conditions:** -user was given access to their account

-user was given access to add item to their cart

**Contract CO16: login**

**Operation:** searchItem ()

**Cross references:** use case diagram user

**Preconditions:** none

**Post conditions:** searched item was displayed

**Contract CO17: searchItem**

**Operation:** viewItem ()

**Cross references:** use case diagram user

**Preconditions:** item was searched

**Post conditions:** details of the particular item was displayed

**Contract CO18: viewItem**

**Operation:** addToCart ()

**Cross references:** use case diagram user

**Preconditions: -**item was searched

-item was viewed

**Post conditions:** -item was added to the cart

-user proceeded to checkout

**Contract CO19: addToCart**

**Operation:** logout ()

**Cross references:** use case diagram user

**Preconditions:** user logged in to his/her account

**Post conditions:** none

**Contract CO20: logout**

## Activity Diagram

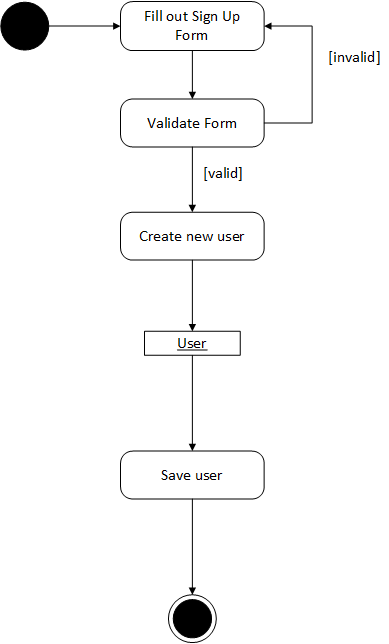


Figure 4- 5: Activity Diagram for User Registration.

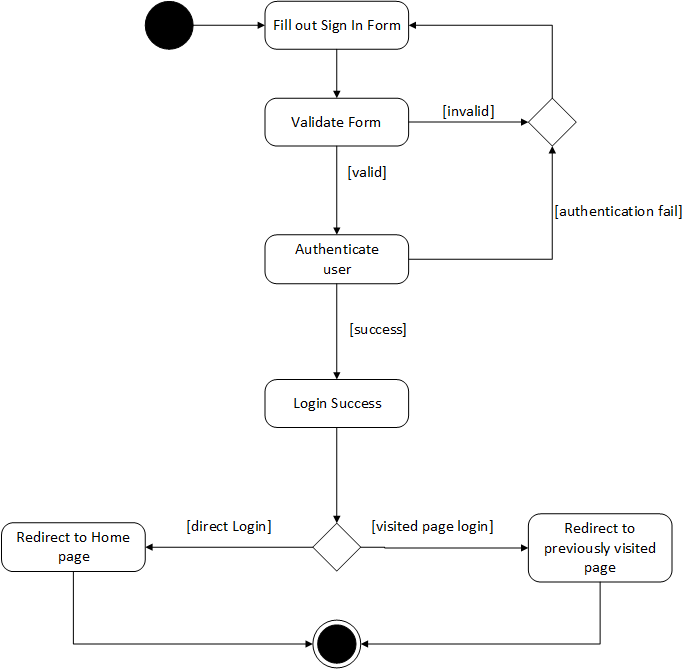
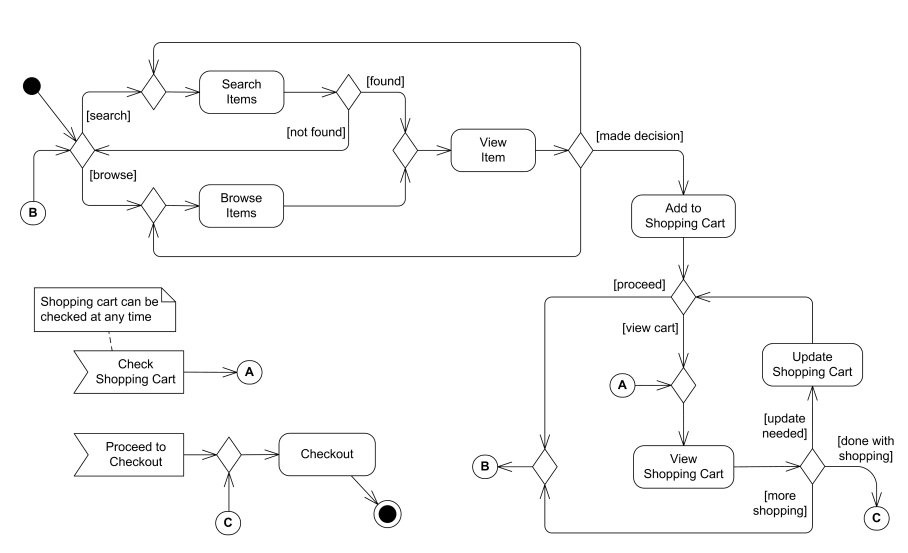


Figure 4- 6: Activity Diagram for User Login.

  
Figure 4- 7: Activity Diagram for Shopping.

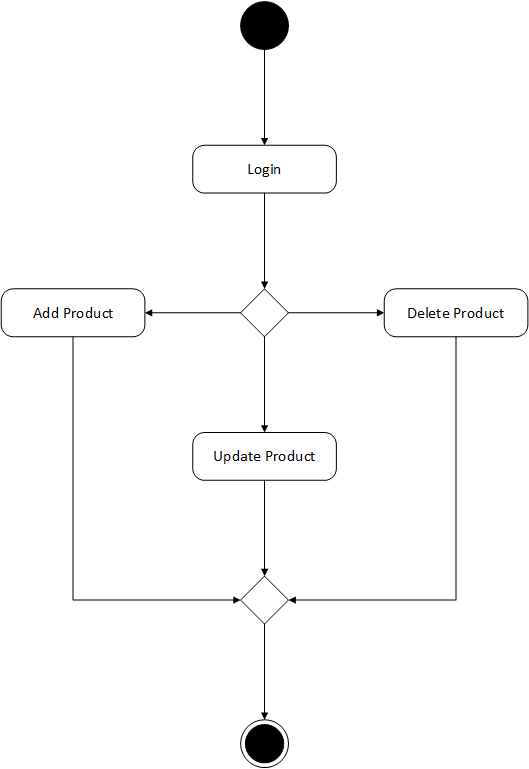
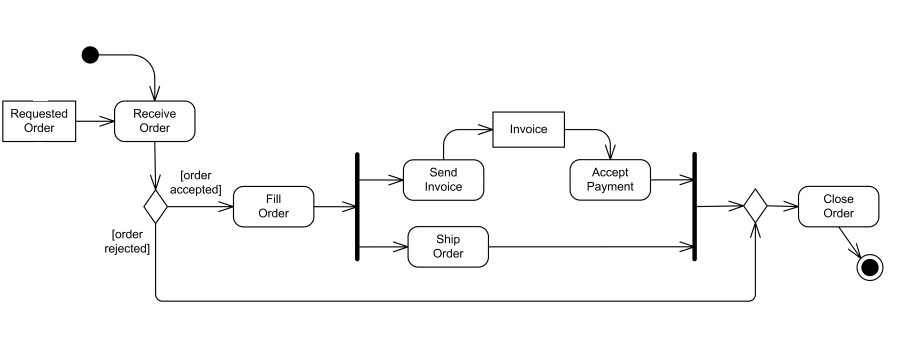
****

Figure 4- 8: Activity Diagram for Merchant.

  
  
Figure 4- 9: Activity Diagram for Super Admin User Management

.

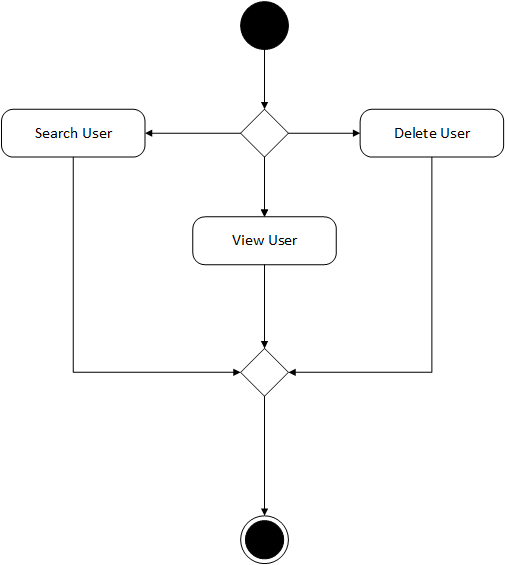


Figure 4- 10: Activity Diagram for Super Admin User Management.

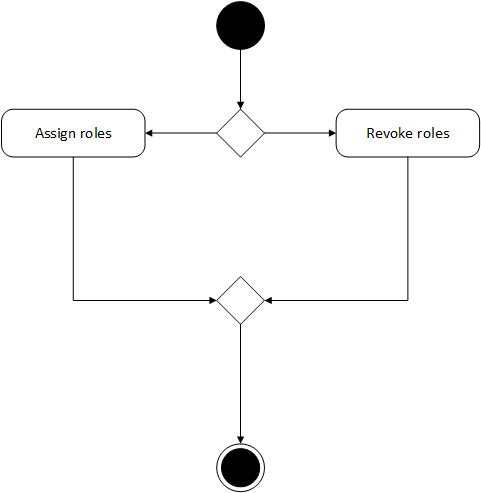
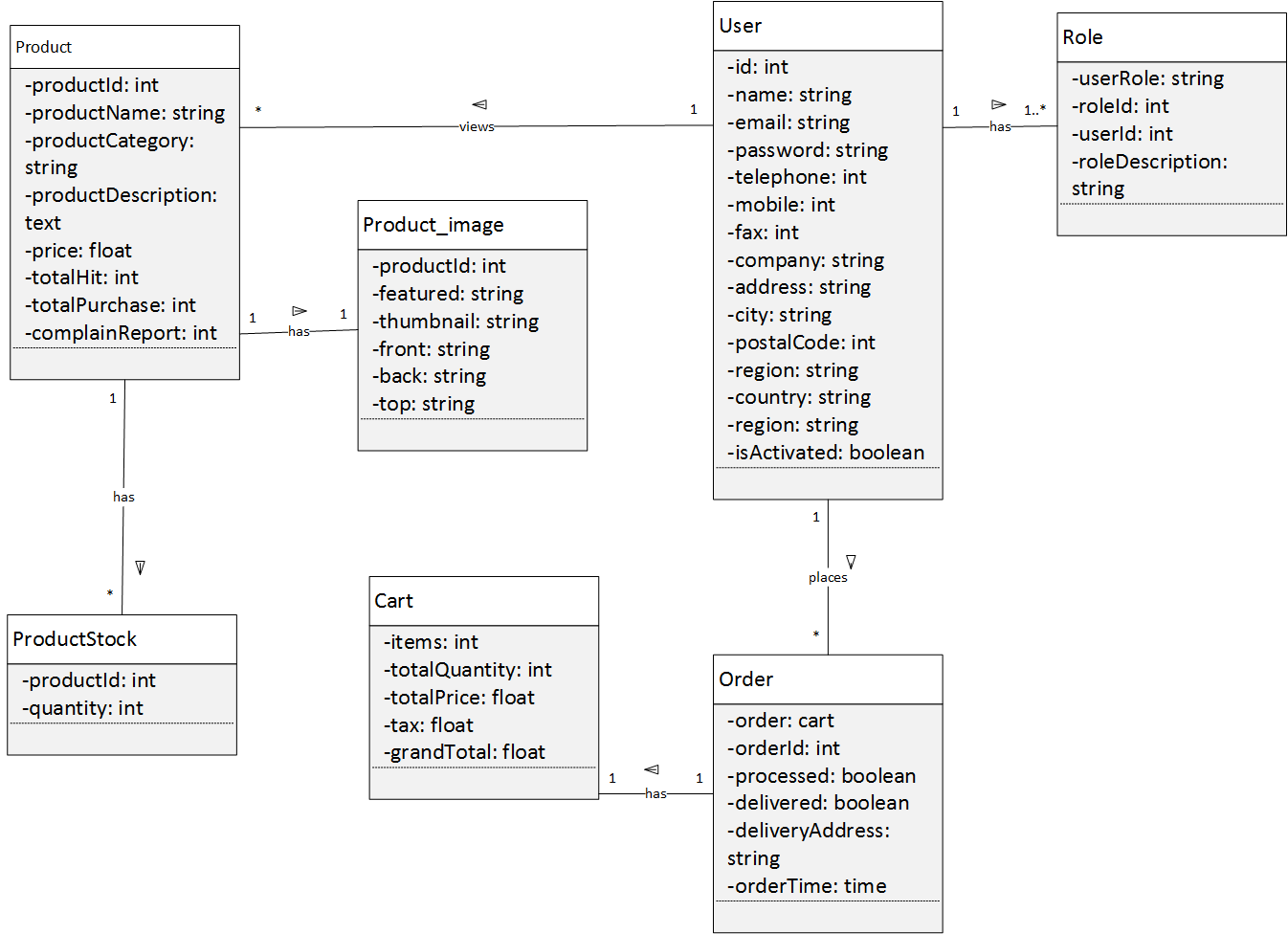


Figure 4- 11: Activity Diagram for Super Admin User Roles Management.

## Domain Model

  
  
Figure 4- 12: Domain Model.

## Entity-Relation (ER) Diagram

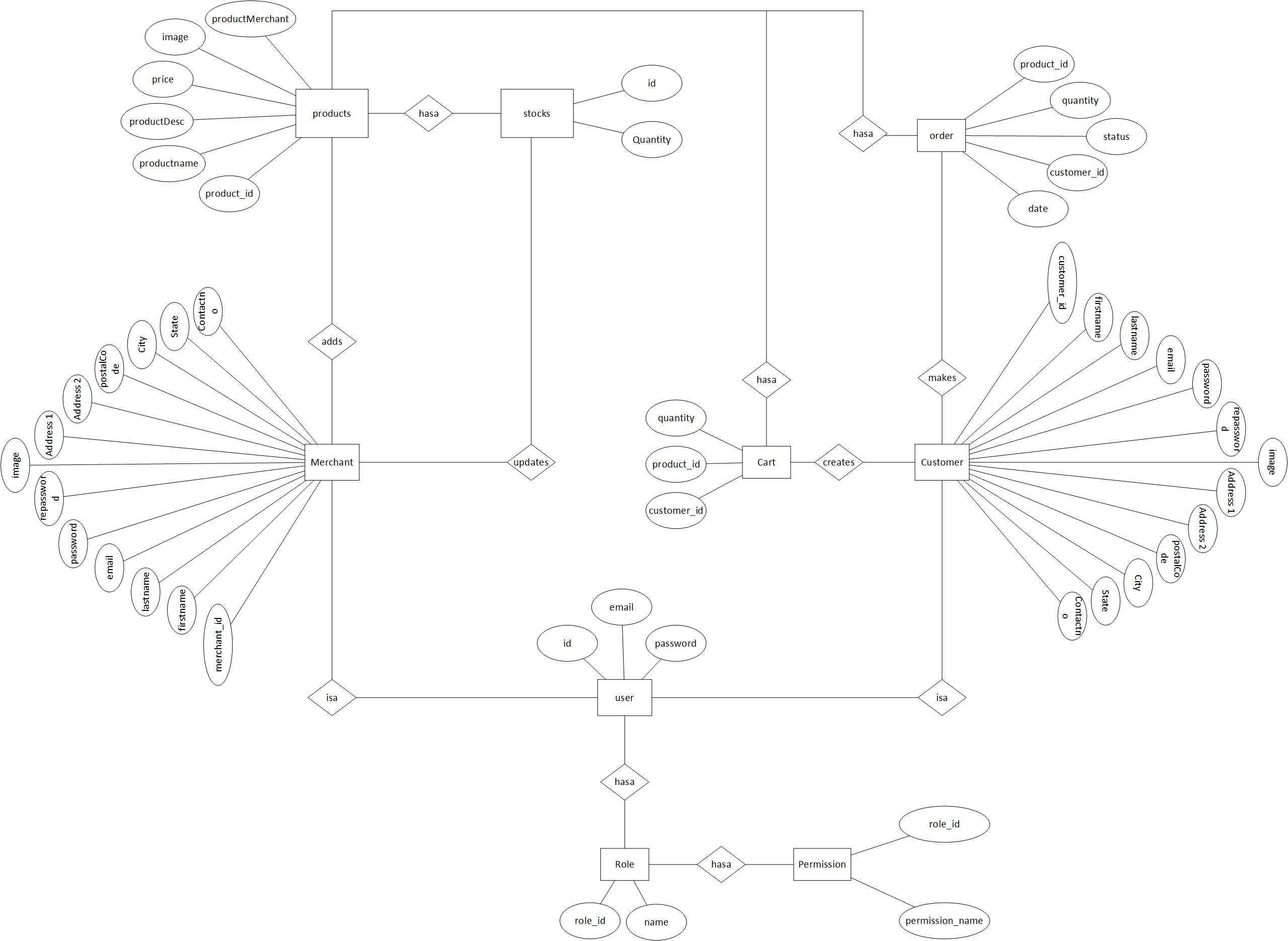


Figure 4- 13: ER Diagram.

## System Sequence Diagrams

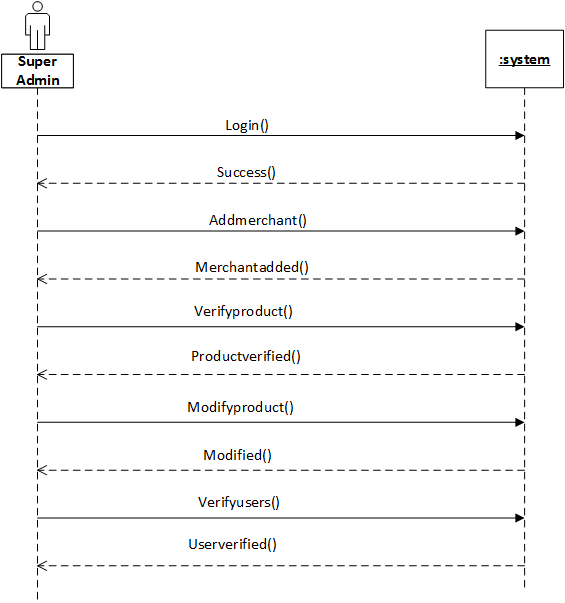
****

Figure 4- 14: SSD for Super Admin.

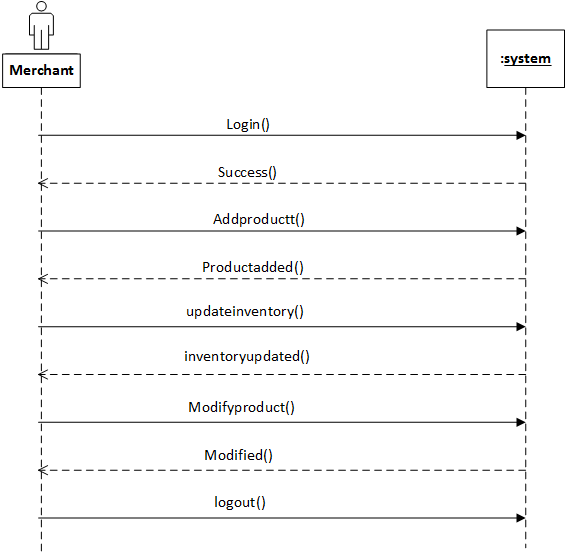
****

Figure 4- 15: SSD for Merchant.

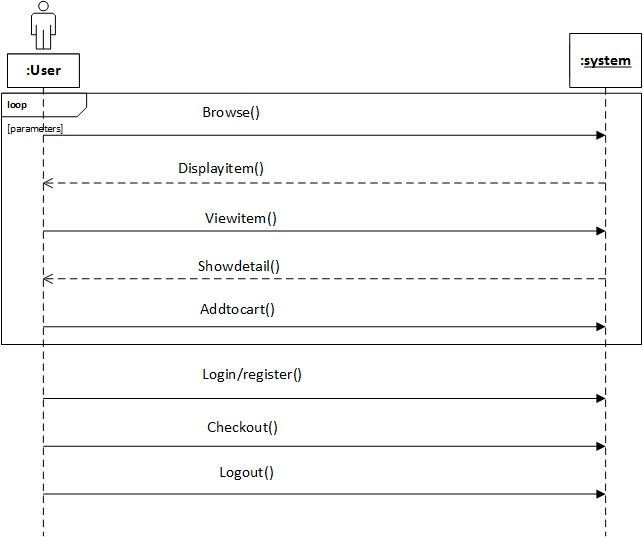
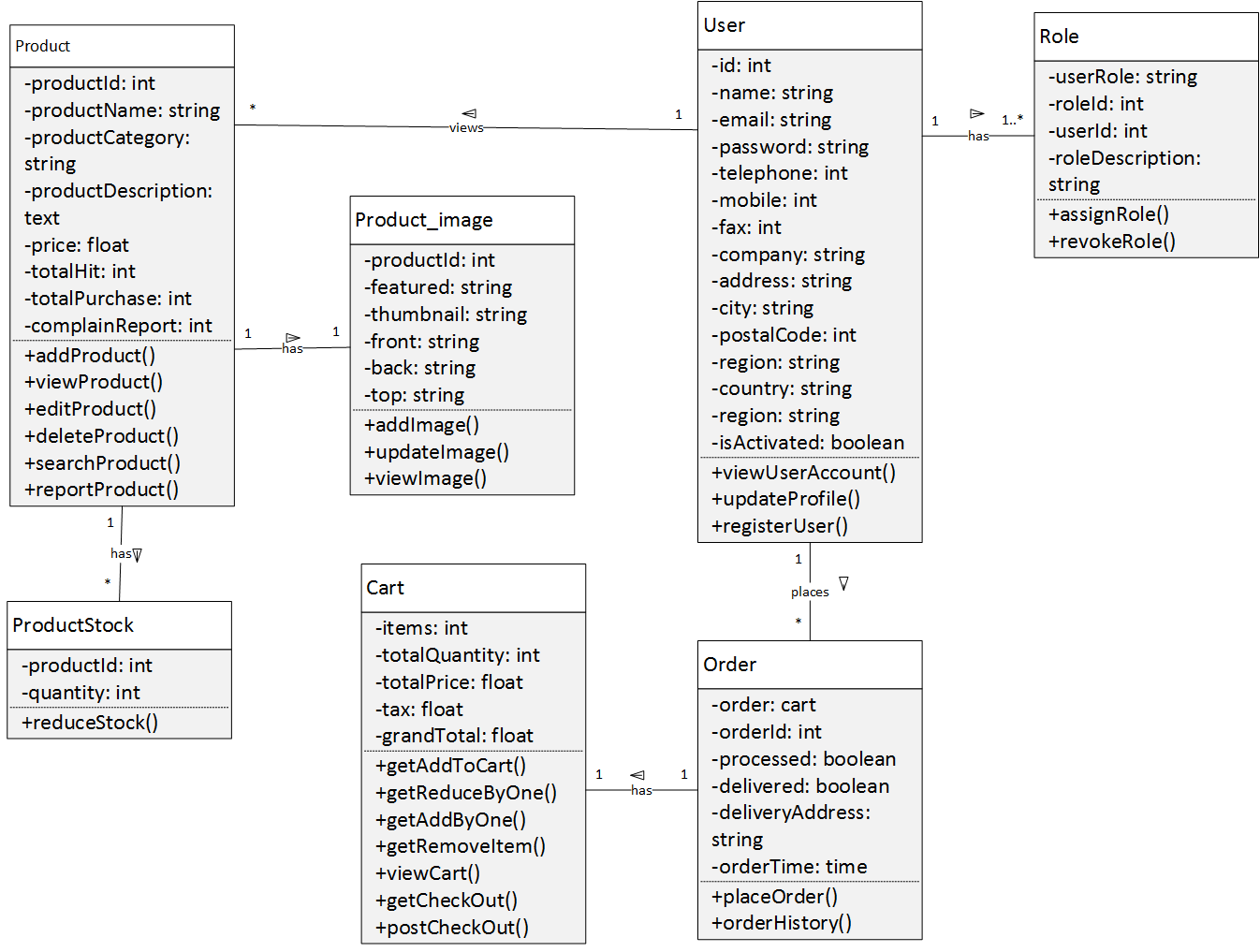
****

Figure 4- 16: SSD User(Customer).

## Design Class Diagram

Figure 4- 17: Design Class Diagram.

## Interaction Diagram (Sequence Diagrams)

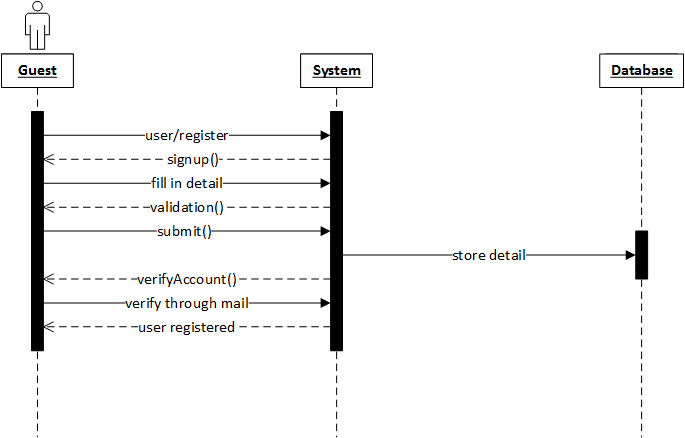


Figure 4- 18: Sequence Diagram for User Registration.

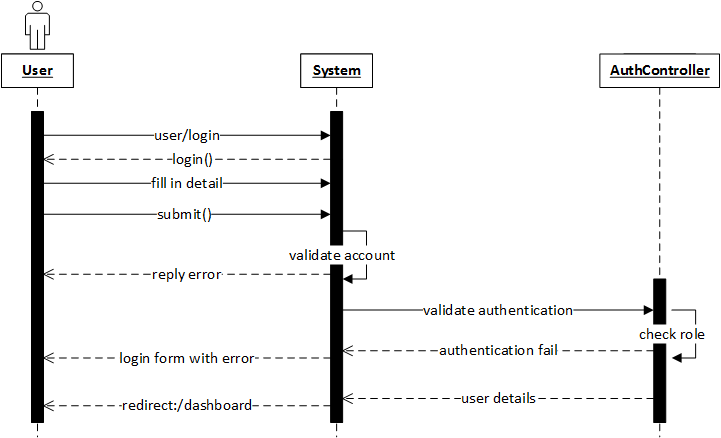
****

Figure 4- 19: Sequence Diagram for User Login.

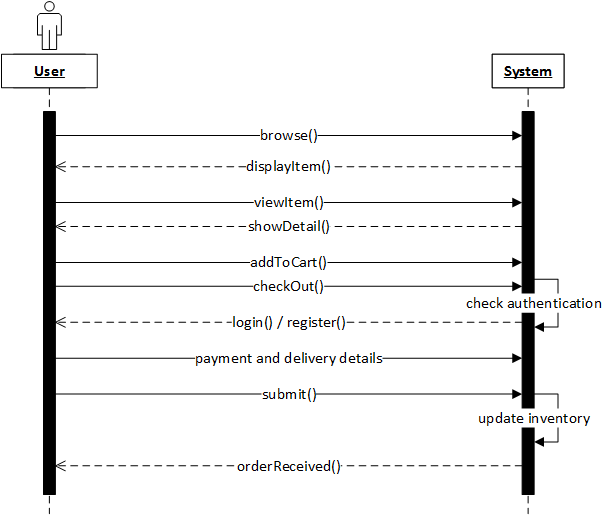
****

Figure 4- 20: Sequence Diagram for Shopping.

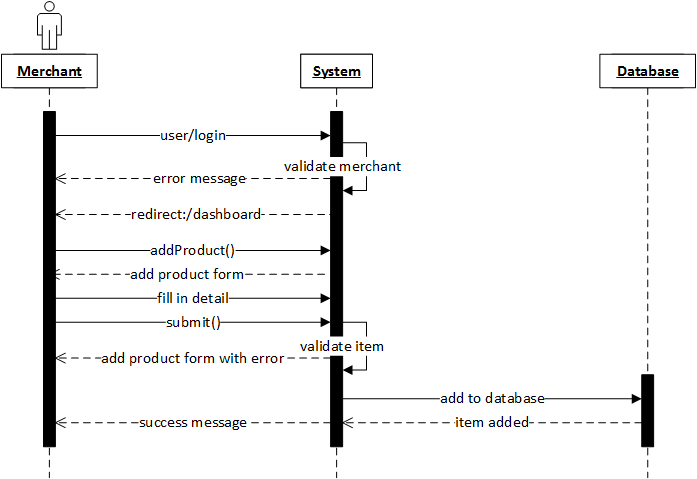
****

Figure 4- 21: Sequence Diagram of Merchant to add Product.

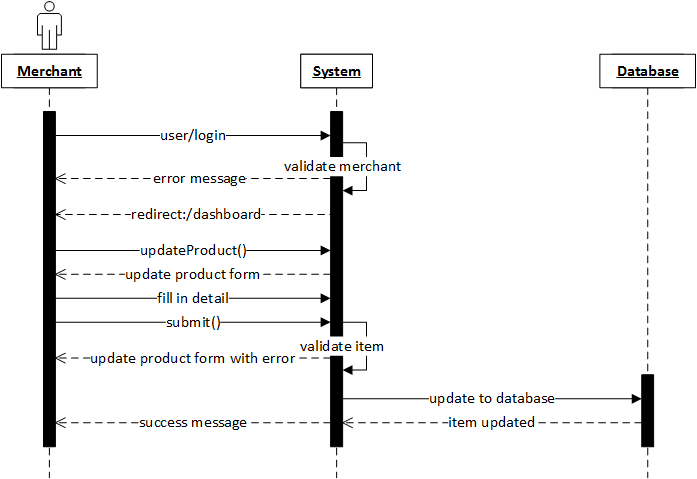
****

Figure 4- 22: Sequence Diagram of Merchant to Update Product.

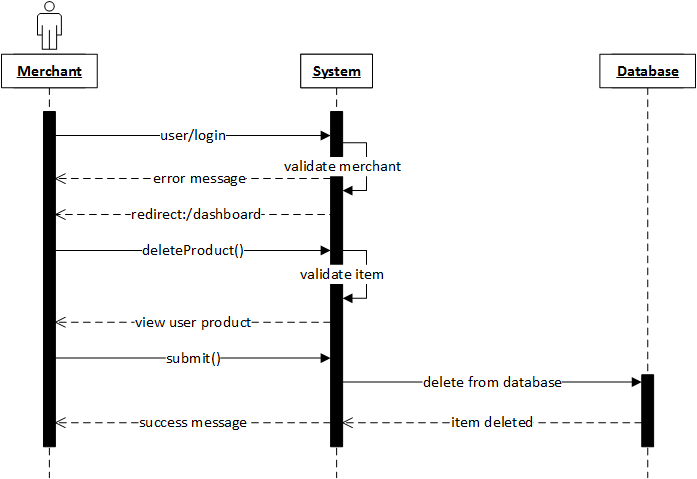
****

Figure 4- 23: Sequence Diagram of Merchant to Delete Product.

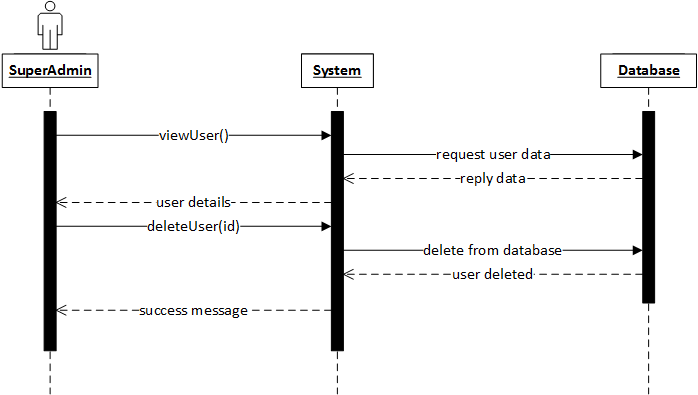
****

Figure 4- 24: Sequence Diagram of Super Admin to view or delete user.

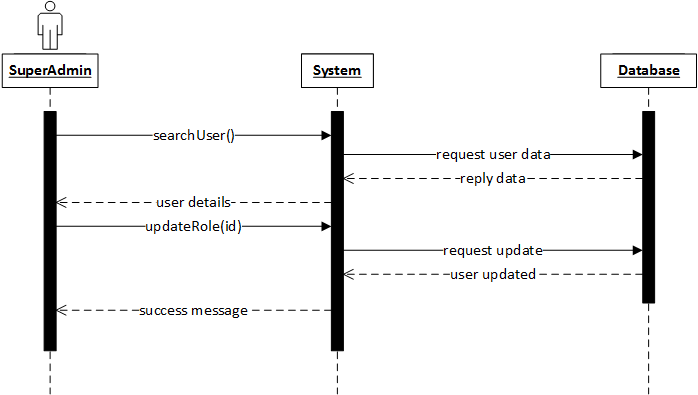
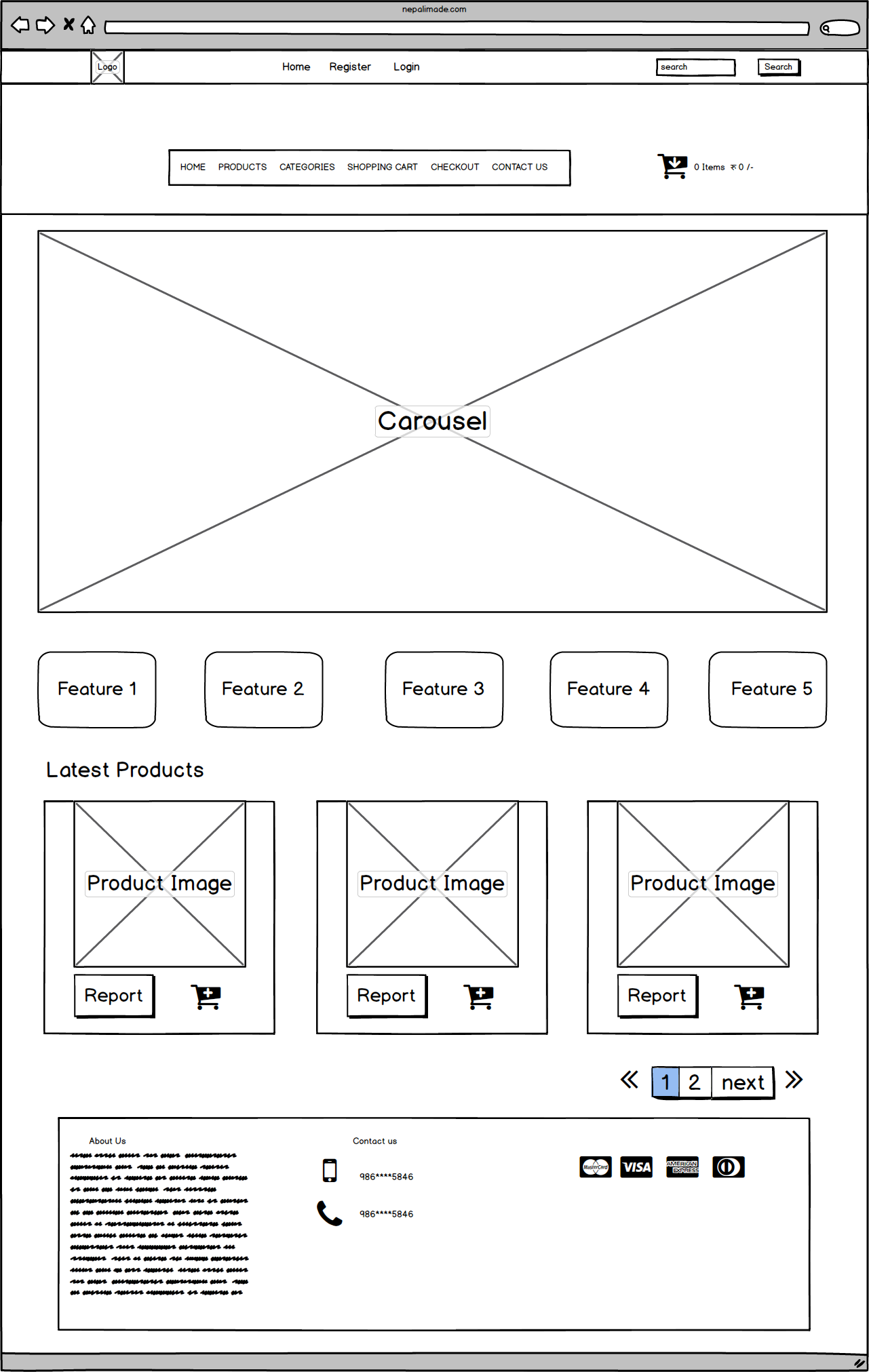
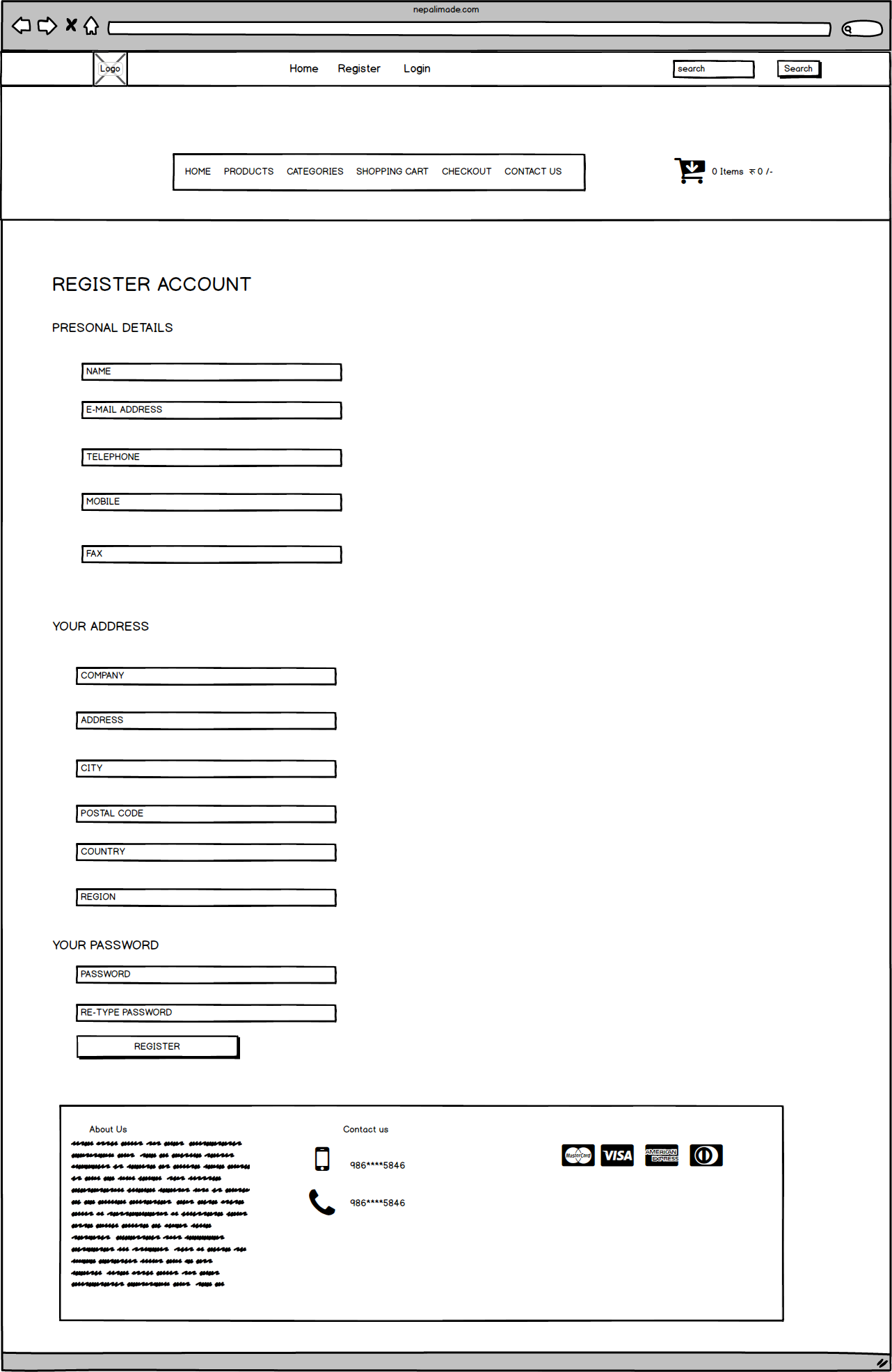
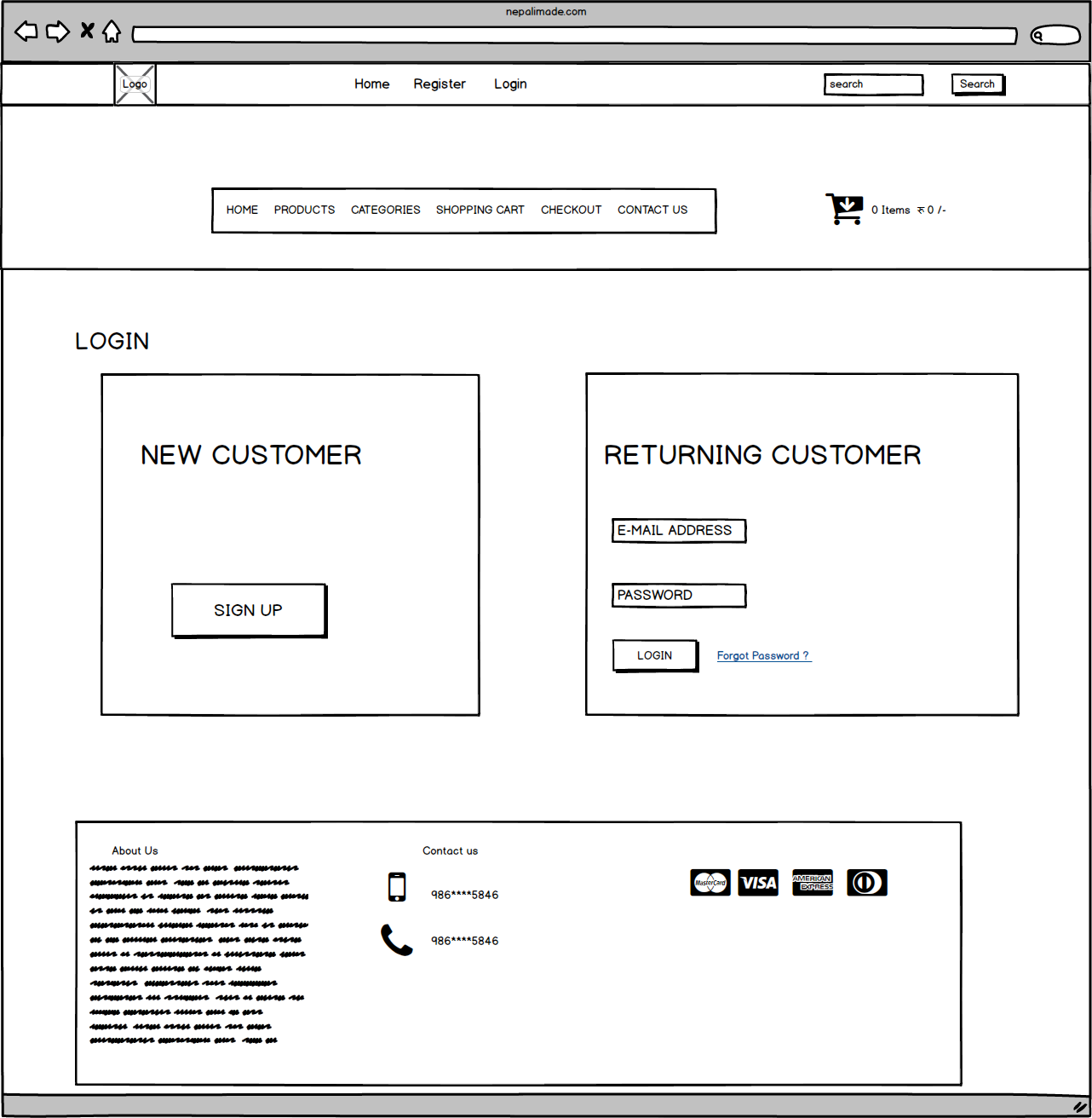
****

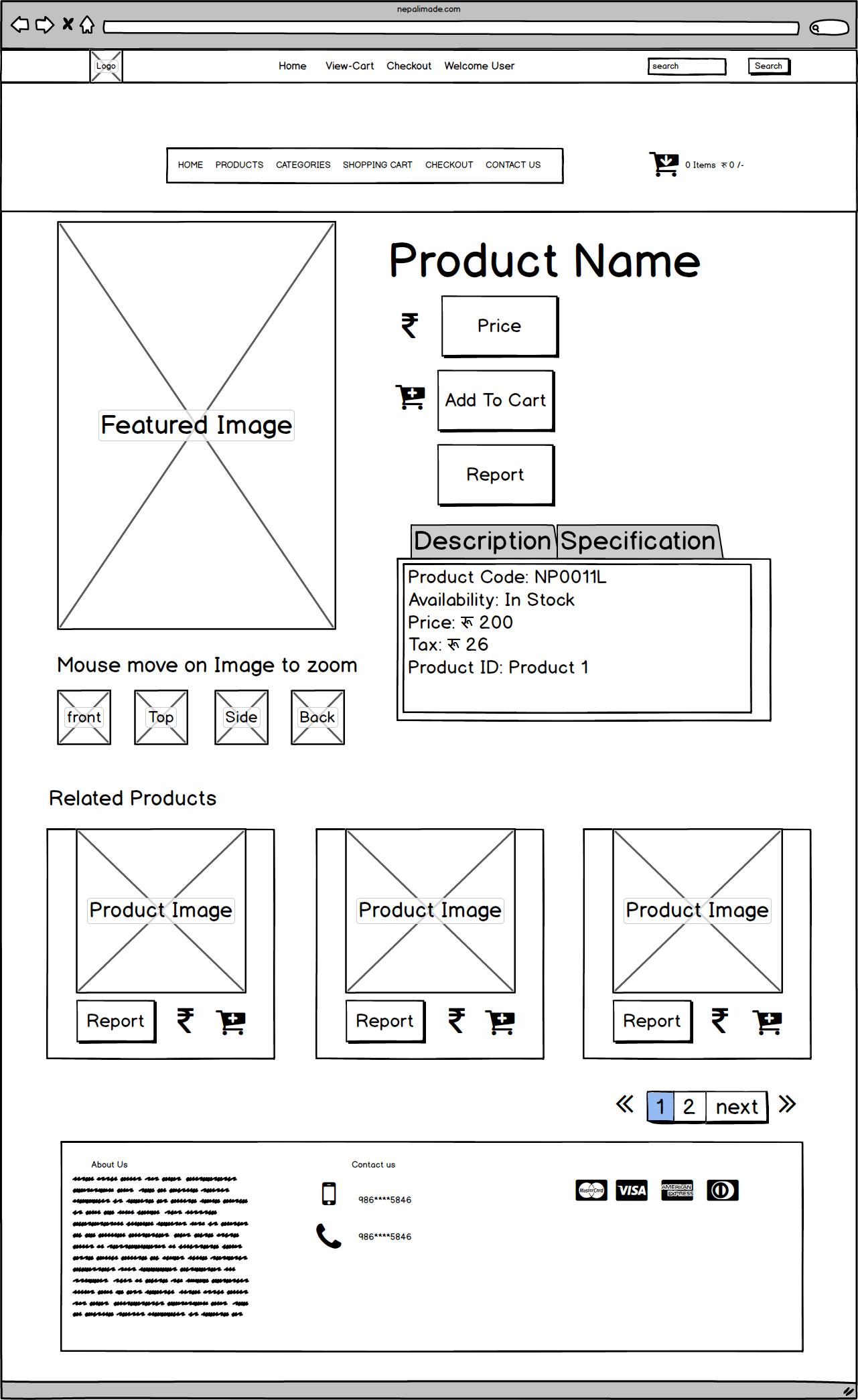
Figure 4- 25: Sequence Diagram of Super Admin to assign/Revoke Roles.

## UI Designs

  
Figure 4- 26: UI Wireframe of Home Page

****Figure 4- 27: UI Wireframe of User Registration Page

****Figure 4- 28: UI Wireframe of Login page.

****  
Figure 4- 29: UI Wireframe Product Detail Page

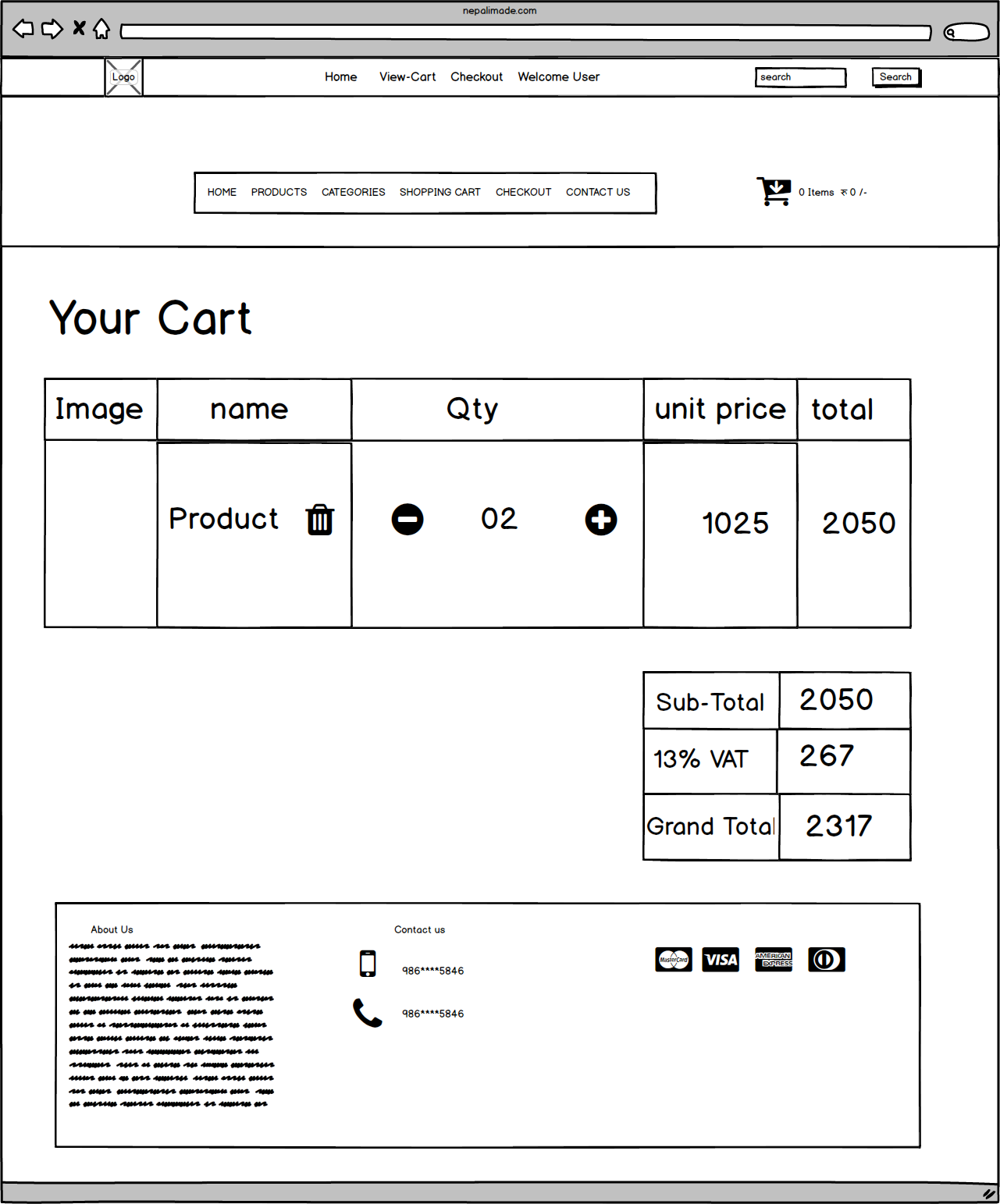
****

Figure 4- 30: UI Wireframe of Shopping-Cart Page.

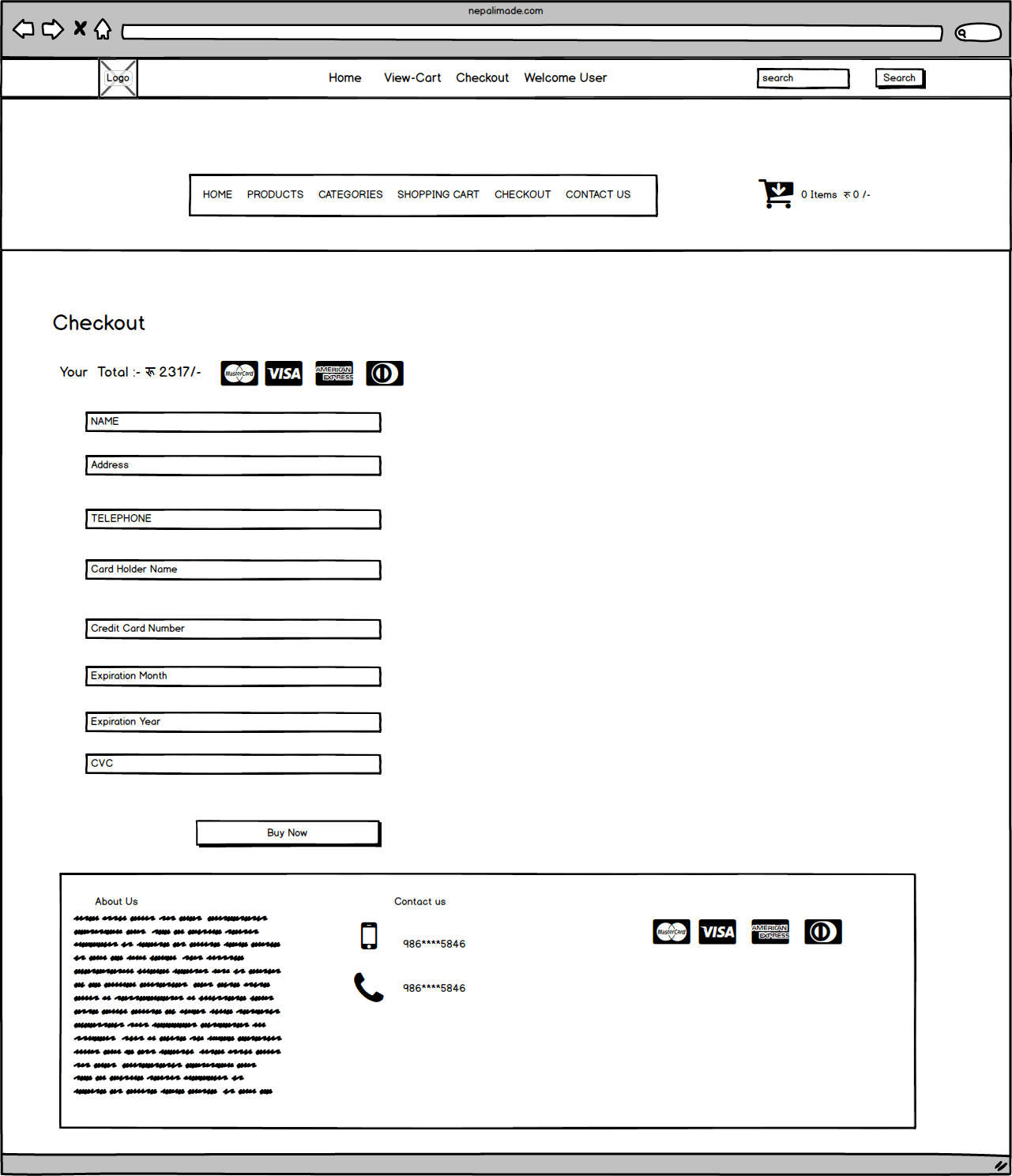
****

Figure 4- 31: UI Wireframe of Checkout Page.

# Chapter 5 IMPLEMENTATION

The Project NepaliMade is developed on top of PHP web framework **Laravel** ver. 5.2, created by Taylor Otwell and intended for the development of web applications following the **model–view–controller** (MVC) **architectural** **pattern**. Laravel is referred to as a “full stack” framework because it handles everything from web serving to database management right down to HTML generation. A vertically integrated web development environment can provide a better experience for the developer. Laravel’s templating engine “Blade” did made our development much easier and hassle free.

The typical developer interacts with Laravel through a command-line utility that generates and manages the Laravel project environment. Laravel comes with an excellent command-line tool named **Artisan** that can be used to generate skeleton code and database schema stubs. Artisan handles everything from database schema migration to asset and configuration management. Below are the major reasons that made us choose Laravel for our project:

* **Bundles** provide a modular packaging system since the release of Laravel 3, with bundled features already available for easy addition to applications. Furthermore, Laravel 4 uses Composer as a dependency manager to add framework-agnostic and Laravel-specific PHP packages available from the Packagist repository.
* **Eloquent ORM** (object-relational mapping) is an advanced PHP implementation of the active record pattern, providing at the same time internal methods for enforcing constraints on the relationships between database objects. Following the active record pattern, Eloquent ORM presents database tables as classes, with their object instances tied to single table rows.
* **Query builder**, available since Laravel 4, provides a more direct database access alternative to the Eloquent ORM. Instead of requiring SQL queries to be written directly, Laravel's query builder provides a set of classes and methods capable of building queries programmatically. It also allows selectable caching of the results of executed queries.
* **Application logic** is an integral part of developed applications, implemented either by using controllers or as part of the route declarations. The syntax used to define application logic is similar to the one used by Sinatra framework.
* **Reverse routing** defines a relationship between the links and routes, making it possible for later changes to routes to be automatically propagated into relevant links. When the links are created by using names of existing routes, the appropriate uniform resource identifiers (URIs) are automatically created by Laravel.
* **Restful controllers** provide an optional way for separating the logic behind serving HTTP GET and POST requests.
* **Class auto loading** provides automated loading of PHP classes without the need for manual maintenance of inclusion paths. On-demand loading prevents inclusion of unnecessary components, so only the actually used components are loaded.
* **View composers** serve as customizable logical code units that can be executed when a view is loaded.
* **Blade templating engine** combines one or more templates with a data model to produce resulting views, doing that by transpiring the templates into cached PHP code for improved performance. Blade also provides a set of its own control structures such as conditional statements and loops, which are internally mapped to their PHP counterparts. Furthermore, Laravel services may be called from Blade templates, and the templating engine itself can be extended with custom directives.
* **IoC containers** make it possible for new objects to be generated by following the inversion of control (IoC) principle, in which the framework calls into the application- or task-specific code, with optional instantiating and referencing of new objects as singletons.
* **Migrations** provide a version control system for database schemas, making it possible to associate changes in the application's codebase and required changes in the database layout. As a result, this feature simplifies the deployment and updating of Laravel-based applications.
* **Database seeding** provides a way to populate database tables with selected default data that can be used for application testing or be performed as part of the initial application setup.
* **Unit testing** is provided as an integral part of Laravel, which itself contains unit tests that detect and prevent regressions in the framework. Unit tests can be run through the provided artisan command-line utility.
* **Automatic pagination** simplifies the task of implementing pagination, replacing the usual manual implementation approaches with automated methods integrated into Laravel.
* **Form request** is a feature of Laravel 5 that serves as the base for form input validation by internally binding event listeners, resulting in automated invoking of the form validation methods and generation of the actual form.
* **Homestead** - a Vagrant virtual machine that provides Laravel developers with all the tools necessary to develop Laravel straight out of the box, including, Ubuntu, Gulp, Bower and other development tools that are useful in developing full scale web applications.
* **Featured packages**

Ready-to-use bundles provided by Laravel through Composer and Packagist include the following:

* **Cashier**, introduced in Laravel 4.2, provides an interface for managing subscription billing services provided by Stripe, such as handling coupons and generating invoices.
* **SSH**, introduced in Laravel 4.1, allows programmatic execution of CLI commands on remote servers using the Secure Shell (SSH) as an encrypted network protocol.
* **Scheduler**, introduced in Laravel 5.0, is an addition to the Artisan that allows programmatic scheduling of periodically executed tasks. Internally, Scheduler relies on the crondaemon to run a single Artisan job that, in turn, executes the configured tasks.
* **Flysystem**, introduced in Laravel 5.0, is a file system abstraction layer that allows local file systems and cloud-based storage services provided by Amazon S3 and Rackspace Cloud to be used transparently and in the same way.
* **Socialite**, introduced in Laravel 5.0 as an optional package, provides simplified mechanisms for authentication with different OAuth providers, including Facebook, Twitter, Google, GitHub and Bitbucket.
* **Artisan CLI**

Laravel's command-line interface (CLI), called Artisan, was initially introduced in Laravel 3 with a limited set of capabilities. Laravel's later migration to a Composer-based architecture allowed Artisan to incorporate different components from the Symfony framework, resulting in the availability of additional Artisan features in Laravel 4.

The features of Artisan are mapped to different subcommands of the artisan command-line utility, providing functionality that aids in managing and building Laravel-based applications. Common uses of Artisan include managing database migrations and seeding, publishing package assets, and generating boilerplate code for new controllers and migrations; the latter frees the developer from creating proper code skeletons. The functionality and capabilities of Artisan can also be expanded by implementing new custom commands, which, for example, may be used to automate application-specific recurring tasks.

# Environment Configuration

We have tried to write self-documented code as much as possible which provides readability and understandability among team members and other. The following code snippet shows Environment configuration of our Laravel Application which is stored in .env file:

|  |
| --- |
| APP\_ENV=local APP\_DEBUG=true APP\_KEY=base64:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\*\*\*\*\*\*\*\*\*\*+\*\*\*\*\*\*\*\*\*\*\*\*= APP\_URL=http://localhost  DB\_CONNECTION=mysql DB\_HOST=127.0.0.1 DB\_PORT=3306 DB\_DATABASE=nepalimade DB\_USERNAME=\*\*\*\*\*\*\* DB\_PASSWORD=\*\*\*\*\*\*\*  CACHE\_DRIVER=file SESSION\_DRIVER=file QUEUE\_DRIVER=sync  REDIS\_HOST=127.0.0.1 REDIS\_PASSWORD=null REDIS\_PORT=6379  MAIL\_DRIVER=smtp MAIL\_HOST=smtp.gmail.com MAIL\_PORT=587 MAIL\_USERNAME=\*\*\*\*\*\*\*@gmail.com MAIL\_PASSWORD=\*\*\*\*\*\*\*\*\*\*\* MAIL\_ENCRYPTION=tls |

Figure 5- 1: Code Snippets of. env file.

As shown in the above code snippets, configuration file includes Environment configuration of Laravel application, MySQL database configuration and Mail configuration.

# Components of Laravel

Since Laravel follows model–view–controller (MVC) architectural pattern, the whole project application is divided as shown in the diagram below:



Figure 5- 2: Components of Laravel.

# File Structure

| **Root folder** | **Purpose** |
| --- | --- |
| app | It contains the Models and Controllers of the application. It also contains all of the logic for performing any operation. Models are placed directly inside this folder. |
| bootstrap | This folder contains the basic setting for starting the application |
| config | This folder contains all configuration settings of the application like database connection, including core classes, email settings, etc. |
| database | This folder contains the code for database transactions like creating table, modifying columns, adding default rows in database, etc. |
| public | This folder can be seen by outer world. This is the directory that is to be pointed to web server. All static assets (like css, js, less, etc.) are placed here. The index.php file in this folder calls the bootstrap folder files and other core files and starts the application. This index.php file receives all of the requests and after complex process responses the content to the visitor. In other words, this is the file which transactions with browser. |
| resources | This folder contains the resources of the application like views, raw assets, language for localization, etc. |
| storage | This folder stores the local data like sessions, caches, compiled views files, etc. |
| tests | This folder contains all tests of application |
| vendor | This folder contains all third-party files (dependencies and additional prepackages for plugins) and code files of the Laravel frameworks. |

**Table 5-1. Laravel File Structure**

# Chapter 6 TESTING

In this strategic testing, road map was drawn and testing was conducted.

## Scope

The overall purpose of testing was to ensure that **NepaliMade** web application meets all of its technical, functional and business requirements. The purpose of this document is to describe the overall testing done for our website application. The approach described in this document provides the framework for all testing related to this application. Individual test cases were written for each version of the application that was released.

## Test Objectives

The quality objectives of testing the NepaliMade web application were to ensure complete validation of the business and software requirements:

* + Verify that software requirements are complete and accurate.
  + Perform detailed test planning.
  + Identify testing standards and procedures that will be used on the project.
  + Prepare and document test scenarios and test cases.
  + Regression testing to validate that unchanged functionality has not been affected by changes.
  + Manage defect tracking process.
  + Provide test metrics/testing summary reports.

## Testing Goals

The goals in testing this application included validating the quality, usability, reliability and performance of the application. Testing was performed from a black-box approach, not based on any knowledge of internal design or code. Testing was done around requirements and functionality.

Another goal was to make the tests repeatable for use in regression testing during the project lifecycle, and for future application upgrades. A part of the approach in testing was to initially perform a ‘Smoke Test’ upon delivery of the application for testing. Smoke Testing is typically an initial testing effort to determine if a new software version is performing well enough to accept it for a major testing effort. For example, if the new software is crashing frequently, or corrupting databases, the software is not in a stable enough condition to warrant further testing in its current state. This testing was performed first. After acceptance of the build delivered for system testing, functions were tested based upon the designated priority (critical, high, medium, low).

## What were tested

The following features of the website were tested for accuracy

* + Home page
  + Categories
  + Items page
  + Cart menu
  + Checkout Process
  + Login page

## Entrance Criteria

* All design specifications were reviewed and approved.
* Unit testing was completed by the development team, including vendors.
* All hardware needed for the test environment were available.
* The application delivered to the test environment were of reliable quality.
* Initial smoke test of the delivered functionality was approved by the testing team.
* Code changes made to the test site went through a change control process.

## Exit Criteria

* All test scenarios were completed successfully.
* All issues prioritized and priority issues resolved.
* All outstanding defects were documented in a test summary with a priority and severity status.
* Go/No-go meeting were held to determine acceptability of product.

## Test Execution

The test execution phase was the process of running test cases against the software build to verify that the actual results meet the expected results. Defects discovered during the testing cycle were introduced to the developers. Once a defect was fixed by a developer, the fixed code was incorporated into the application and regression tested.

These following testing phases was completed:

* **Unit Testing**

Unit testing was performed by the **NepaliMade** development team in their development environment. The developers knew and tested the internal logical structure of each software component.

* **Functional Testing**

Functional testing focused on the functional requirements of the software and was performed to confirm that the application operates accurately according to the documented specifications and requirements, and to ensure that interfaces to external systems are properly working.

* **Regression Testing**

Regression testing was performed to verify that previously tested features and functions do not have any new defects introduced, while correcting other problems or adding and modifying other features.

* **Integration Testing**

Integration testing was the phase of software testing in which individual software modules were combined and tested as a group. In its simplest form, two units that have already been tested were combined into a component and the interface between them was tested. In a realistic scenario, many units were combined into components, which were in turn aggregated into even larger parts of the program. The idea was to test combinations of pieces and eventually expand the process to test your modules with those of other groups. Eventually all the modules making up a process were tested together.

* **Interface Testing**

This testing followed a transaction through all of the product processes that interact with it and tests the product in its entirety. Interface testing were performed to ensure that the product actually works in the way a typical user would interact with it.

* **Destructive Testing**

Destructive testing focused on the error detection and error prevention areas of the product. This testing was exercised in an attempt to anticipate conditions where a user may encounter errors. Destructive testing was less structured than other testing phases and was determined by individual testers.

* **User acceptance testing**

User acceptance testing activities were performed by the business users. The purpose of this testing was to ensure the application meets the users’ expectations. This also included focuses on usability and included appearance, consistency of controls, consistency of field naming, accuracy of drop down field information lists, spelling of all field name/data values, accuracy of default field values, tab sequence, and error/help messaging.

* **Browser Testing**

Functional and Regression as defined in this test strategy were executed using following Browsers:

* Google Chrome
* Microsoft Edge

## Test Result

The test result of each unit test and integration test are done while developing the system and is reviewed to identify and remove errors. The Following table consists of the test results of Black Box Testing which are performed to validate the system with respect to the requirement.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Test** | **Expected Behavior** | **Precondition** | | **Actual Behavior** | | **Result** | |
| 1. 1 | Register | The user should register | None | | user registers and email is sent for account activation | | Pass | |
| 1. 2 | Test register with email already taken | Email already taken error should be generated | None | | As Expected | | Pass | |
| 1. 3 | Register with invalid email | Invalid Email error should be generated | None | | As Expected | | Pass | |
| 1. 1 | Login to the system | The user should be logged in | 1 PASS | | User is logged in to the system | | Pass | |
| 1. 2 | Test login form with missing inputs | Suitable form error should be generated | None | | It shows the form returning invalid username or password | | Pass, need to improve input validation error message | |
| 1. 3 | Login into system providing incorrect email or password | The user should not be logged in and error should be generated indicating incorrect email or password | None | | As Expected | | PASS | |
| 1. 4 | Logout form system | The user should be logged out and login page should be displayed | | 4 PASS | | As Expected | | PASS |
| 1. 5 | Forget Password | After providing correct email the password should be reset and sent to user email in case of forgotten. | | None | | As Expected | | PASS |
| 1. 6 | Do not provide the mandatory details and submit form | The error should be generated indicating missing field | | 4PASS | | As Expected | | PASS |
| 1. 8 | Adding product to cart | Product with that product Id should be added to the cart using Session | | 4PASS | | As Expected | | PASS |
| 1. 9 | Adding same product to cart | Adding same product in cart should only increase the quantity of the product not the product row. | | 10PASS | | As Expected | | PASS |
| 1. 10 | Reducing product from cart | reducing a product from cart should decrease the quantity of that certain product by 1 or by the amount deducted by the user along with the tax and grand total. | | 11PASS | | As Expected | | PASS |
| 1. 11 | Reduce product form cart when Quantity = 1 | When quantity of a product becomes 1, thereafter deducting any quantity should ultimately delete the row of that product and update the tax and the grand total | | 12PASS | | Product row with quantity < 1 is deleted from the cart | | PASS |
| 1. 12 | Remove all products from cart | All products rows should be removed and no carts should be shown and “No items in cart” should be shown. | | 13 PASS | | “No items in the cart” message is displayed | | PASS |
| 1. 13 | Checkout without any items in the cart | Error message “no items in the car” should be displayed | | None | | As Expected | | PASS |
| 1. 14 | Checkout with items in cart but without login | User should be redirected to the login page where he should login or registers (if new user) and should be thrown back to the checkout page after login successful | | 10PASS | | As Expected | | PASS |
| 1. 15 | Checkout with grand total less than 50 cents. | Error message should be generated with suitable error | | 10PASS | | As Expected | | PASS |
| 1. 16 | Checkout without internet connection | Suitable error message should be generated | | 10PASS | | “not connected to the internet” error message displayed | | PASS |
| 1. 17 | Checkout with invalid credit card credentials | Suitable error message should be generated | | 10PASS | | Stripe error message displayed | | PASS |

**Table 6-1. Test Results**

# Chapter 7 RESULT AND DISCUSSIONS

We started to develop this system as our Final Year Major Project for the Bachelor of Engineering Software Engineering. Within the time frame we were proud to complete this project with some additional features like Role Based Access Control and Credit Card Payment gateway integration which were not the part of our project when we submitted the proposal report for commencing this particular project.

The project was completed within the given time frame with active involvement and coordination of all of the four team members and excellent support from the supervisor.

With the goal, which we started this project, we have come far ahead now where we can now start deploying our project in the market with a very little cost and effort.

Smart Hotel is focused on providing a user-friendly and easy to use web portal for people to buy/sell Nepali products. People can easily search thousands of Nepali products that remained undiscovered till now. With a single user sign in people can even apply for Merchant ship program for selling their own products through our web portal. No more tedious payment gateways as we have integrated support for large number of credit card payment services along with a cash on delivery procedure to attract more and more users towards our system.

# Chapter 8 FUTURE IMPROVEMENTS

In the very first version of this project we have integrated a number of features and options. Being an ecommerce system, the need to improve and introduce new additional features and removal of some existing features is a must. So, for our next version we have planned and made plenty of research tasks to make our system much more advance. Some of the future improvements for this project can be listed below:

* Advanced rating and review of individual products.
* Advanced reporting of unrelated bad products for the users to filter the products more precisely.
* Better recommendation systems to show related products based on user’s products views, purchase etc.
* Enhanced recommendation system for users to recommend particular products to their relative and friends.
* Advanced compare sub system integration for users to go for the best choice on the spot.
* Social Media integration in the system for registration/login and sharing.

# Chapter 9 CONCLUSION

Hence, our project will successfully promote the local products as well as helps in increasing the opportunities in employment and uplifting the economic status of rural areas. It will not only help to promote the products in our local market but also help product reach it to the international market which will help us to collect good revenue from international market.

Smart Hotel as a whole will make the process of online shopping in Nepal lot easier than traditional shopping. And the best part of all of this is that people will be able to search for the products manufactured in their own backyard. This will certainly help little or more for uplifting the national economy and national identity.

# Bibliography

Fakhroutdinov, K. (2010). *http://www.uml-diagrams.org/examples/online-shopping-use-case-diagram-example.html*. Retrieved from http://www.uml-diagrams.org.

foodmandu.com. (2016). *http://foodmandu.com/*. Retrieved from http://foodmandu.com/.

investopedia.com. (2015, 10). *http://www.investopedia.com/terms/b/btob.asp*. Retrieved from http://www.investopedia.com.

investopedia.com. (2015, 03). *http://www.investopedia.com/terms/b/btoc.asp*. Retrieved from http://www.investopedia.com.

JetBrains. (2016). *https://www.jetbrains.com/phpstorm/*. Retrieved from https://www.jetbrains.com/phpstorm/.

Kaymu.com.np. (2015). *http://Kaymu.com.np.* Retrieved from http://Kaymu.com.np.

Khan, H. (n.d.). *A Report on eCommerce Trends in Nepal.* http://www.kaymu.com.np/.

Laravel. (2016). *https://laravel.com/docs/5.2.* Retrieved from https://laravel.com.

Larry Press, S. G. (n.d.). *Electronic commerce in Nepal.* Retrieved from https://www.isoc.org/: https://www.isoc.org/oti/articles/0401/press.html

Products, N. (2015). *http://www.nepaliproducts.com/about\_us.php*. Retrieved from http://www.nepaliproducts.com/.

Shop, N. A. (2007). *http://www.nepalartshop.com/*. Retrieved from http://www.nepalartshop.com/.

wikipedia. (2016, 11 16). Retrieved from https://en.wikipedia.org/wiki/Laravel.

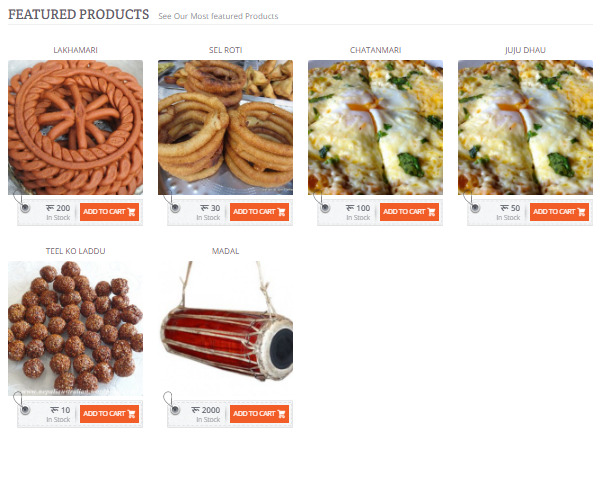
wikipedia. (2016, 11 18). *https://en.wikipedia.org/wiki/JavaScript*. Retrieved from https://en.wikipedia.org: https://en.wikipedia.org/wiki/JavaScript

Wikipedia. (2016, 11 03). *https://en.wikipedia.org/wiki/Role-based\_access\_control*. Retrieved from https://en.wikipedia.org/wiki/Role-based\_access\_control.

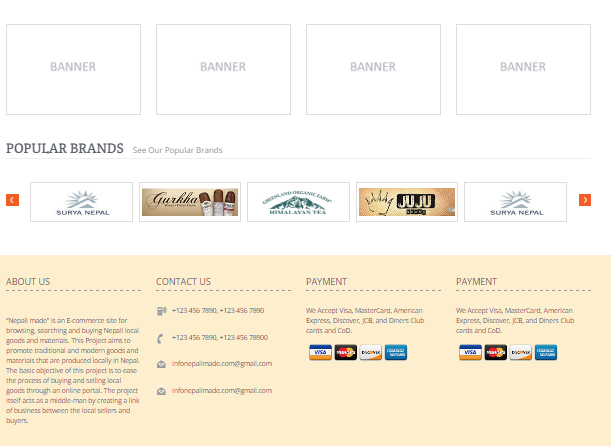
# APPENDIX I USER INTERFACE

****

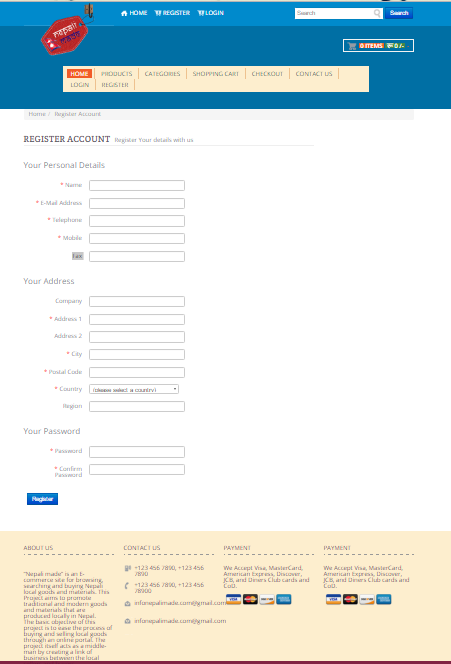
**Figure i. Home Page upper section**



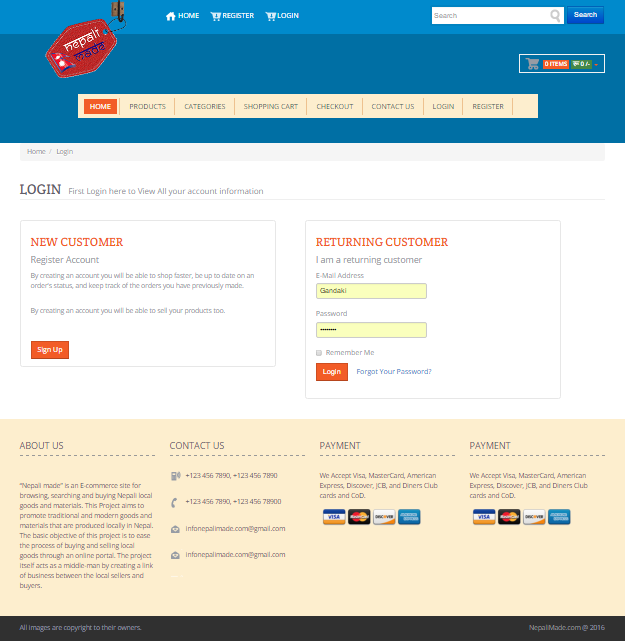
**Figure ii. Home Page middle section**

****

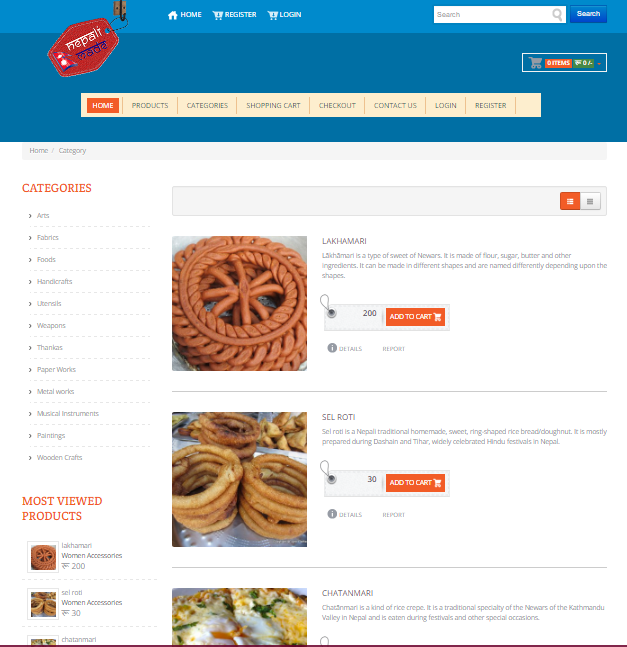
**Figure iii. Home Page lower section**

****

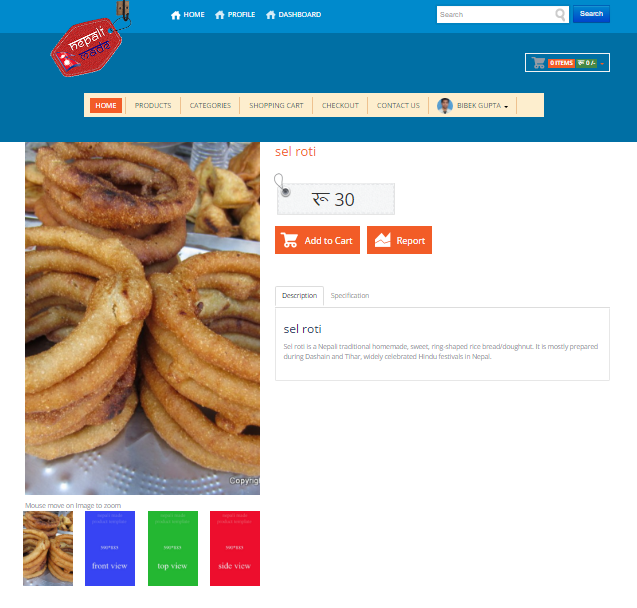
**Figure iv. User Registration Page**

****

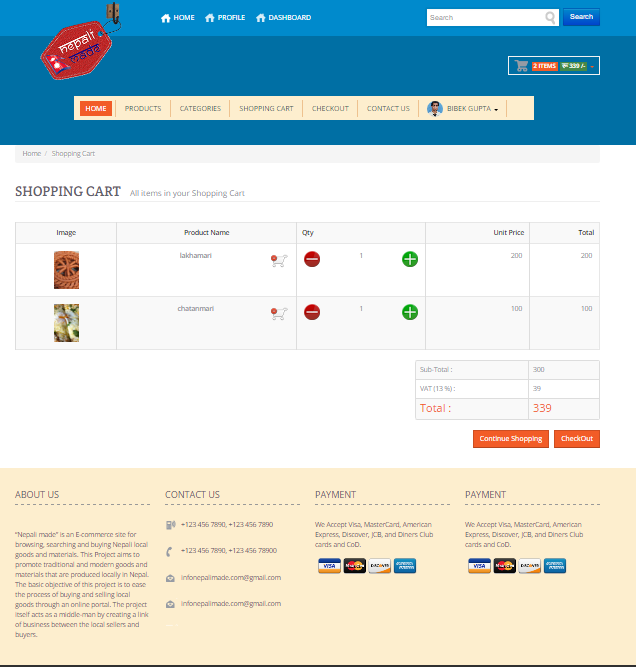
**Figure v. User Login Page**

****

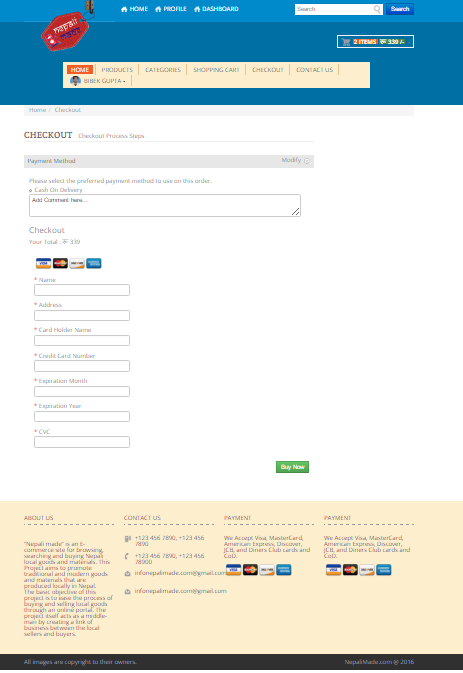
**Figure vi. Products Category Page**



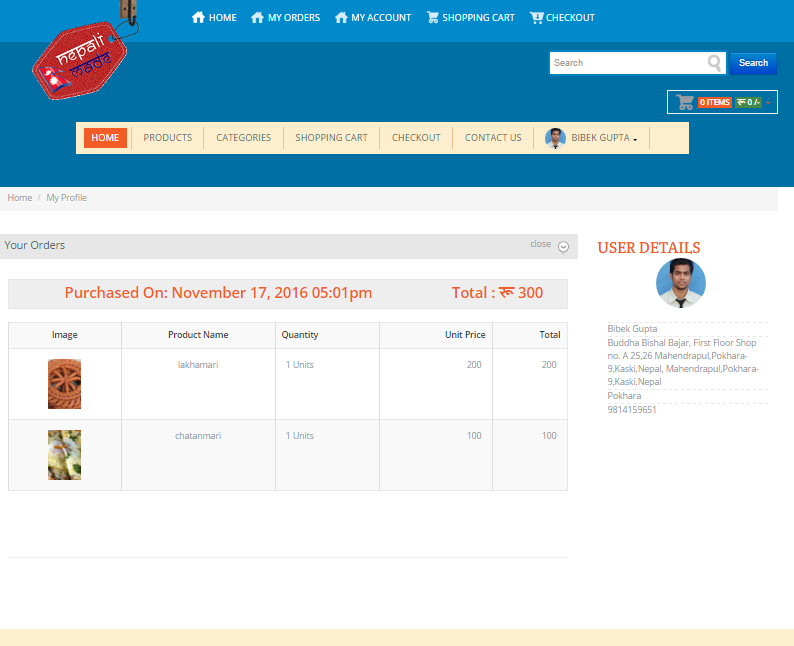
**Figure vii. Product Details page**

****

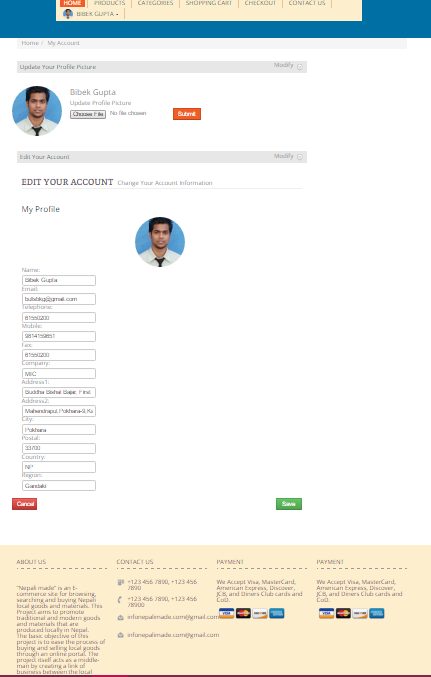
**Figure viii. Shopping-Cart Page**

****

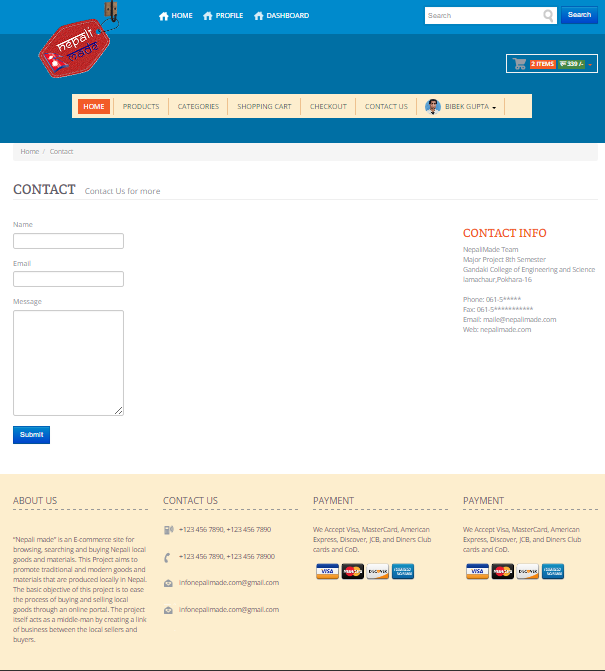
**Figure ix. Checkout Page**

****

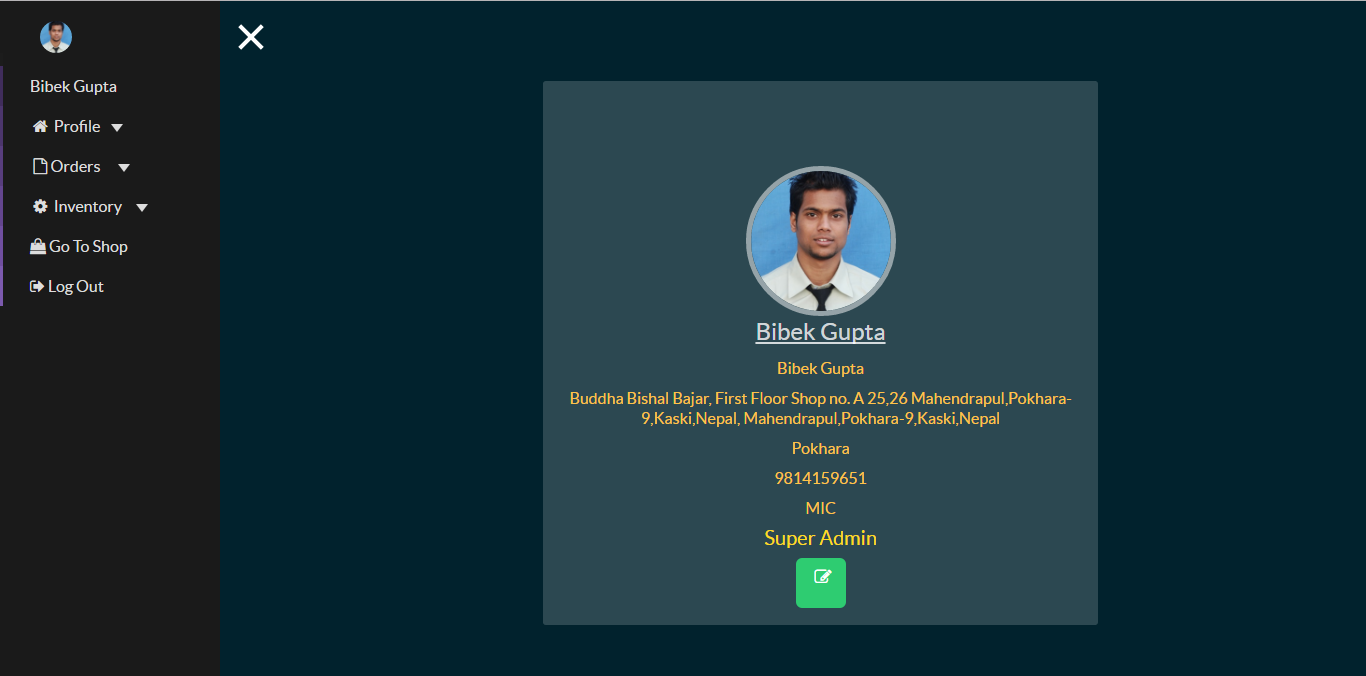
**Figure x. Orders history Page**



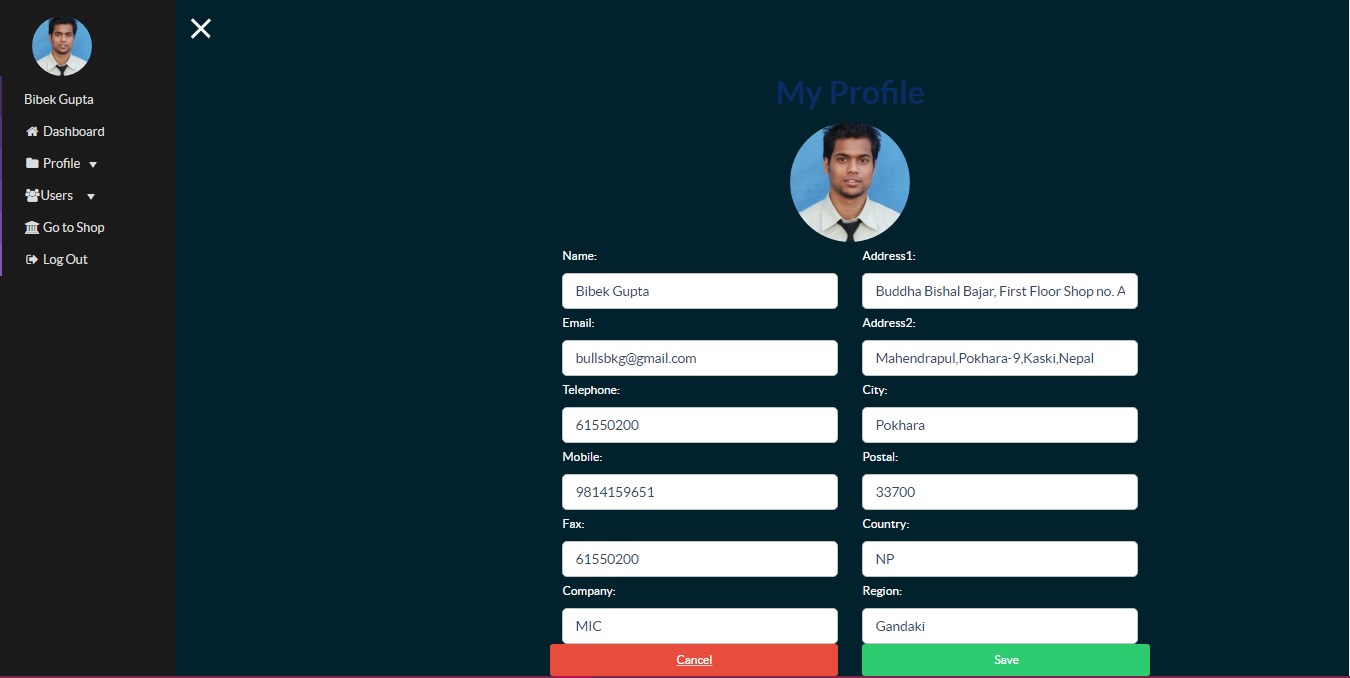
**Figure xi. user Edit Account and Change Avatar Page**

****

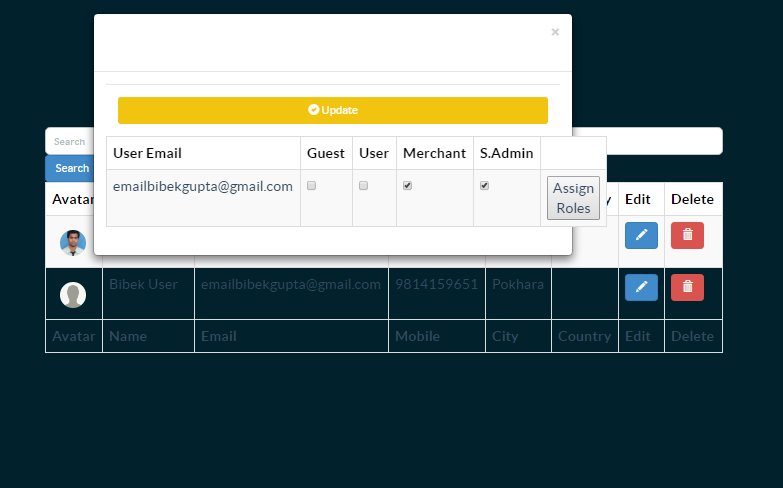
**Figure xii. Contact-us Page**

****

**Figure xiii. Merchant Dashboard**

****

**Figure xiv. Merchant/Super admin update account page**

****

**Figure xv. Super Admin view users and assign roles page**

# APPENDIX II SAMPLE CODE SNIPPETS

1. Routes
2. Routes for guest/users

Route::get('/', [  
 'uses' => 'PageController@index',  
 'as' => 'pages.master'  
]);

Route::Post('/contact-message', [  
 'uses' => 'ContactController@store',  
 'as' => 'postContactMessage'  
]);  
  
Route::auth();  
Route::resource('/product', 'ProductController');  
  
  
Route::get('/product/report/{id}', [  
 'uses' => 'ProductController@reportProduct',  
 'as' => 'product.report'  
]);  
  
Route::get('/product-search-result', [  
 'uses' => 'ProductController@searchProduct',  
 'as' => 'product.search'  
]);  
  
Route::get('/add-to-cart/{id}', [  
 'uses' => 'ProductController@getAddToCart',  
 'as' => 'product.addToCart'  
]);  
  
Route::get('/reduce/{id}', [  
 'uses' => 'ProductController@getReduceByOne',  
 'as' => 'product.reduceByOne'  
]);  
  
Route::get('/add/{id}', [  
 'uses' => 'ProductController@getAddByOne',  
 'as' => 'product.addByOne'  
]);  
  
Route::get('/remove/{id}', [  
 'uses' => 'ProductController@getRemoveItem',  
 'as' => 'product.remove'  
]);  
  
Route::get('/shopping-cart', [  
 'uses' => 'ProductController@getCart',  
 'as' => 'product.shoppingCart'  
]);  
  
Route::get('/checkout', [  
 'uses' => 'ProductController@getCheckout',  
 'as' => 'checkout',  
 'middleware' => 'auth'  
  
]);  
  
Route::post('/checkout', [  
 'uses' => 'ProductController@postCheckout',  
 'as' => 'checkout',  
 'middleware' => 'auth'  
]);  
  
  
Route::get('/profile', [  
 'uses' => 'UserController@getProfile',  
 'as' => 'user.profile',  
 'middleware' => 'auth'  
]);  
  
Route::get('/update-profile', [  
 'uses' => 'UserController@getUpdateProfile',  
 'as' => 'user.updateProfile',  
 'middleware' => 'auth'  
]);  
Route::post('/update-profile/{id}', [  
 'uses' => 'UserController@postUpdateProfile',  
 'as' => 'user.updateProfile',  
 'middleware' => 'auth'  
]);  
  
  
Route::get('/invoice', [  
 'uses' => 'ProductController@getInvoicePrinted',  
 'as' => 'print.invoice',  
  
]);

**Figure a. Code Snippets of Routes.php file without any middleware**

1. Routes for Merchant

*/\* Routes for Merchants \*/*Route::group(['prefix' => 'admin'], function () {  
 Route::group(['middleware' => 'roles'], function () {  
 Route::group(['roles' => 'Merchant'], function () {  
 Route::get('/', [  
 'uses' => 'AdminController@getViewAccount',  
 'as' => 'admin.dashboard',  
 ]);  
 Route::get('/dashboard', [  
 'uses' => 'AdminController@getViewAccount',  
 'as' => 'Admin.dashboard',  
 ]);  
 Route::get('/neworders', [  
 'uses' => 'AdminController@newOrders',  
 'as' => 'Admin.neworders',  
 ]);  
 Route::get('/viewmyproducts', [  
 'uses' => 'AdminController@viewMyProducts',  
 'as' => 'Admin.viewmyproducts',  
 ]);  
 Route::get('addproduct', [  
 'uses' => 'AdminController@addProductForm',  
 'as' => 'admin.addproduct',  
 ]);  
 Route::get('/edit-Profilepic', [  
 'uses' => 'AdminController@getEditProfile',  
 'as' => 'Admin.editProfilePic',  
 ]);  
 Route::Post('/edit-Profilepic', [  
 'uses' => 'AdminController@postEditProfile',  
 'as' => 'Admin.editProfilePic',  
 ]);  
 Route::get('/viewupdate-Product', [  
 'uses' => 'AdminController@getViewProduct',  
 'as' => 'Admin.viewProduct',  
 ]);  
  
 *//Admin.viewProduct* Route::get('/update-Product/{id}', [  
 'uses' => 'AdminController@getUpdateProduct',  
 'as' => 'Admin.getupdateproduct',  
 ]);  
  
 Route::post('/update-Product/{id}', [  
 'uses' => 'AdminController@postUpdateProduct',  
 'as' => 'Admin.postupdateproduct',  
 ]);  
 *// to do Admin.updateProduct* Route::get('/delete-product/{id}', [  
 'uses' => 'AdminController@destroy',  
 'as' => 'Admin.deleteproduct',  
 ]);  
  
  
 Route::get('/view-orders', [  
 'uses' => 'AdminController@viewOrders',  
 'as' => 'Admin.vieworders',  
 ]);  
 Route::get('/order-details/{id}', [  
 'uses' => 'AdminController@orderDetails',  
 'as' => 'Admin.orderdetails',  
 ]);  
 Route::get('/process-order/{id}', [  
 'uses' => 'AdminController@processOrder',  
 'as' => 'Admin.processorder',  
 ]);  
 Route::get('/deliver-order/{id}', [  
 'uses' => 'AdminController@deliverOrder',  
 'as' => 'Admin.deliverOrder',  
 ]);  
 *//Admin.deliverOrder* Route::Post('/edit-product', [  
 'uses' => 'AdminController@postProductUpdate',  
 'as' => 'Admin.getupdateproduct',  
 ]);  
  
  
 Route::get('updateProduct', [

'uses' => 'AdminController@getProductUpdate',  
 'as' => 'Admin.updateProduct'  
  
 ]);  
  
 Route::get('/processed-orders', [  
 'uses' => 'AdminController@processedOrders',  
 'as' => 'Admin.processedorders',  
 ]);  
 Route::get('/delivered-orders', [  
 'uses' => 'AdminController@deliveredOrders',  
 'as' => 'Admin.deliveredorders',  
 ]);  
 */\*route for admin profile\*/* Route::get('/edit-account', [  
 'uses' => 'AdminController@getEditAccount',  
 'as' => 'Admin.editAccount',  
 ]);  
  
 Route::Post('/edit-account/{id}', [  
 'uses' => 'AdminController@postEditAccount',  
 'as' => 'Admin.updateProfile',  
 ]);  
 Route::get('/view-account', [  
 'uses' => 'AdminController@getViewAccount',  
 'as' => 'Admin.viewAccount',  
 ]);  
 });  
  
 });  
});

**Figure b. Code Snippets of Routes.php file with middleware merchant roles**

1. Routes for Super Admin

Route::group(['prefix' => 'super-admin'], function () {  
 Route::group(['middleware' => 'roles'], function () {  
 Route::group(['roles' => 'Super Admin'], function () {  
 Route::get('/', [  
 'uses' => 'SuperAdminController@getSuperAdminPage',  
 'as' => 'superAdmin',  
 ]);  
 Route::post('/assign-roles', [  
 'uses' => 'SuperAdminController@postAdminAssignRoles',  
 'as' => 'superAdmin.assign',  
 ]);  
 Route::get('/edit-account/', [  
 'uses' => 'SuperAdminController@getEditAccount',  
 'as' => 'superAdmin.editAccount',  
 ]);  
 Route::Post('/edit-account/{id}', [  
 'uses' => 'SuperAdminController@postEditAccount',  
 'as' => 'superAdmin.updateProfile',  
 ]);  
 Route::get('/view-account', [  
 'uses' => 'SuperAdminController@getViewAccount',  
 'as' => 'superAdmin.viewAccount',  
 ]);  
 Route::get('/updateAvatar', [  
 'uses' => 'SuperAdminController@getUpdateAvatar',  
 'as' => 'superAdmin.Avatar',  
 ]);  
 Route::Post('/updateAvatar', [  
 'uses' => 'SuperAdminController@PostUpdateAvatar',  
 'as' => 'superAdmin.Avatar',  
 ]);  
  
 Route::get('/all-users', [  
 'uses' => 'SuperAdminController@viewUsers',  
 'as' => 'superAdmin.viewUsers',  
 ]);  
  
 Route::get('/search-users', [  
 'uses' => 'SuperAdminController@searchUsers',  
 'as' => 'superAdmin.SearchUsers',  
 ]);  
  
 Route::get('/delete-users/{id}', [  
 'uses' => 'SuperAdminController@deleteUser',  
 'as' => 'superAdmin.deleteUser',  
 ]);  
 Route::get('/contact-messages', [  
 'uses' => 'ContactController@showMessage',  
 'as' => 'superAdmin.showMessage',  
  
 ]);  
 });  
 });  
});

**Figure c. Code Snippets of Routes.php file with middleware super admin roles**

1. Controllers

<?php  
  
namespace App\Http\Controllers;  
  
use App\Product;  
use Illuminate\Http\Request;  
  
use App\Http\Requests;  
  
class SearchController extends Controller  
{  
  
public function autocomplete(Request $request)  
{  
  
 *//prevent this method called by non ajax* if ($request->ajax()) {  
 $products = Product::where(function ($query) use ($request) {  
  
 *//Filer by keyword entered* if (($term = $request->get('term'))) {  
 $query->orWhere('product\_name', 'like', '%' . $term . '%');  
 $query->orWhere('product\_description', 'like', '%' . $term . '%');  
 $query->orWhere('product\_category', 'like', '%' . $term . '%');  
 $query->orWhere('price', 'like', '%' . $term . '%');  
 }  
 })  
 ->orderBy("id", "desc")  
 ->take(2)  
 ->get();  
  
 *//Convert to json* $results = [];  
 foreach ($products as $product) {  
 $results[] = ['id' => $product->id, 'value' => $product->product\_name];  
 }  
 return response()->json($results);  
 }  
}  
}

**Figure d. Code Snippets of SearchController.php file**

public function getAddToCart(Request $request, $id){  
 $products = Product::find($id);  
 $stock\_qty=$products->stock->quantity;  
 $oldCart = Session::has('cart') ? Session::get('cart') : null;  
 $cart = new Cart($oldCart);  
 $cart -> add($products, $products->id);  
 $request->session()->put('cart', $cart);return redirect()->route('pages.master');  
}

**Figure e. Code Snippets of ProductController.php file to add product to cart**

public function register(Request $request)  
{  
 $input = $request->all();  
  
 $validator = $this->validator($input);  
  
 if ($validator->passes()) {  
  
 $user = $this->create($input)->toArray();  
  
 $user['link'] = str\_random(30);  
  
 DB::table('user\_activations')->insert(['id\_user'=>$user['id'],'token'=>$user['link']]);  
  
 Mail::send('emails.activation', $user, function($message) use ($user) {  
 $message->to($user['email']);  
 $message->subject('Site - Activation Code');  
 });  
  
 return redirect()->to('login')  
 ->with('success',"We sent activation code. Please check your mail.");  
 }  
  
 return back()->with('errors',$validator->errors());  
}

**Figure f. Code Snippets of AuthController.php file for user registration**

public function userActivation($token)  
{  
 $check = DB::table('user\_activations')->where('token',$token)->first();  
 if(!is\_null($check)){  
 $user = User::find($check->id\_user);  
 if($user->is\_activated == 1){  
 return redirect()->to('login')  
 ->with('success',"user are already activated.");  
 }  
 $user->update(['is\_activated' => 1]);  
 DB::table('user\_activations')->where('token',$token)->delete();  
 if (Session::has('oldUrl')){  
 $oldUrl = Session::get('oldUrl');  
 Session::forget('oldUrl');  
 return redirect()->to($oldUrl);  
 }  
 return redirect()->to('login')  
 ->with('success',"user active successfully.");  
 }  
 return redirect()->to('login')  
 ->with('warning',"your token is invalid.");  
}

**Figure g. Code Snippets of AuthController.php file for user activation through email verification**

protected function login(Request $request)  
 {  
 $this->validate($request, [  
 'email' => 'required|email',  
 'password' => 'required',  
 ]);  
 if (auth()->attempt(array('email' => $request->input('email'), 'password' => $request->input('password'))))  
 {  
 if(auth()->user()->is\_activated == '0'){  
 $this->logout();  
 return back()->with('warning',"First please active your account.");  
 }  
 if(auth()->user()->hasRole('User')) {return redirect()->intended('/myaccount');  
 }  
 elseif(auth()->user()->hasRole('Merchant')){return redirect()->intended('/admin');  
 }  
 elseif(auth()->user()->hasRole('Super Admin')){return redirect('super-admin');}

**Figure h. Code Snippets of AuthController.php file for user activation login and redirect**

public function postCheckout(Request $request)  
 {  
 if (!Session::has('cart'))  
 {  
 return redirect()->route('shop.shopping-cart');  
 }  
 $oldCart = Session::get('cart');  
 $cart = new Cart($oldCart);  
  
 Stripe::setApiKey('sk\_test\_YI4KNxY7onplBwQgjjyQqVfr');  
 try{  
 $charge = Charge::create(array(  
 "amount" => $cart->totalPrice \* 100,  
 "currency" => "NPR",  
 "source" => $request->input('stripeToken'), *// obtained with Stripe.js* "description" => "Test Charge"  
 ));  
 $order = new Order();  
 $order->cart = serialize($cart);$order->address = $request->input('address');  
   
 $order->name = $request->input('name');  
 $order->payment\_id = $charge->id;  
  
 Auth::user()->orders()->save($order);  
  
 }  
 catch(\Exception $e){  
 return redirect()->route('checkout')->with('error', $e->getMessage());  
 }foreach($cart->items as $item){  
   
  
 $product\_id=$item['item']->id;  
 $product=Product::findOrFail($product\_id);  
 $stock\_id=$product->stock->id;  
 $stock =Stock::findorfail($stock\_id);*// $arr->item->stock;* $stock->quantity = $stock->quantity - $item['qty'];  
 $stock->save();  
  
 $product->total\_purchased++;  
 $product->save();  
  
 }Session::forget('cart');  
 return redirect()->route('pages.master')->with('success', 'Successfully Purchased Products !');  
 }

**Figure i. Code Snippets of ProductController.php file for Checkout Process**

1. Models

<?php  
namespace App;  
use Illuminate\Foundation\Auth\User as Authenticatable;  
class User extends Authenticatable  
{protected $fillable = [  
 'name', 'email', 'password', 'telephone', 'mobile', 'fax', 'company', 'address1', 'address2', 'city', 'postalcode', 'country', 'region', 'is\_activated',  
 ];protected $hidden = [  
 'password', 'remember\_token',  
 ];  
 public function product(){  
 return $this->hasMany('App\Product');  
 }  
 public function orders(){  
 return $this ->hasMany('App\Order');  
 }  
 public function roles()  
 {  
 return $this->belongsToMany('App\Role', 'user\_role', 'user\_id', 'role\_id');  
 }  
 public function hasAnyRole($roles)  
 {  
 if (is\_array($roles)) {  
 foreach ($roles as $role) {  
 if ($this->hasRole($role)) {  
 return true;  
 }  
 }  
 } else {  
 if ($this->hasRole($roles)) {  
 return true;  
 }  
 }  
 return false;  
 }  
 public function hasRole($role)  
 {  
 if ($this->roles()->where('name', $role)->first()) {  
 return true;  
 }  
 return false;  
 }  
 public function scopeSearchByKeyword($query, $keyword)  
 {  
 if ($keyword!='') {  
 $query->where(function ($query) use ($keyword) {  
 $query->where("name", "LIKE","%$keyword%");  
 });  
 }  
 return $query;  
 }  
}

**Figure j. Code Snippets of User Model**

<?php  
  
namespace App;  
  
use Illuminate\Database\Eloquent\Model;  
  
class Product extends Model  
{  
   
 protected $fillable = [  
 'product\_name',  
 'product\_category',  
 'product\_description',  
 'price',  
 'user\_id',  
 'total\_hit',  
 'total\_purchased',  
 'complain\_reports',  
 ];  
 public function user(){  
 return $this->belongsTo('App\User');  
 }  
  
 public function images(){  
 return $this->hasOne('App\Product\_image', 'product\_id');  
 }  
  
 public function stock(){  
 return $this->hasOne('App\Stock');  
 }  
  
 public function scopeSearchByKeyword($query, $keyword)  
 {  
 if ($keyword!='') {  
 $query->where(function ($query) use ($keyword) {  
 $query->where("product\_name", "LIKE","%$keyword%")  
 ->orWhere("product\_category", "LIKE", "%$keyword%")  
 ->orWhere("product\_description", "LIKE", "%$keyword%")  
 ->orWhere("price", "LIKE", "%$keyword%");  
 });  
 }  
 return $query;  
 }  
}

**Figure k. Code Snippets of Product Model**

1. Views

<!DOCTYPE html>  
<html lang=**"en"**>  
<head>  
 <meta charset=**"utf-8"**>  
  
 <title>**NepaliMade-All kinds of Nepali Goods**</title>  
 <meta name=**"viewport"** content=**"width=device-width, initial-scale=1.0"**>  
 <meta name=**"description"** content=**""**>  
 <meta name=**"author"** content=**""**>  
 <link href=**'http://fonts.googleapis.com/css?family=Open+Sans:400,300italic,400italic,600,600italic'** rel=**'stylesheet'** type=**'text/css'**>  
 <link href=**'http://fonts.googleapis.com/css?family=Crete+Round'** rel=**'stylesheet'** type=**'text/css'**>  
 <link href=**'http://fonts.googleapis.com/css?family=Crete+Round'** rel=**'stylesheet'** type=**'text/css'**><link href=**"css/bootstrap.css"** rel=**"stylesheet"**>  
 <link rel=**"stylesheet"** href=**"**{{asset('assets/css/bootstrap.css')}}**"** media=**"all"**>  
 <link href=**"**{{asset('assets/css/bootstrap-responsive.css')}}**"** rel=**"stylesheet"**>  
 <link href=**"**{{ asset('jqueryui/jquery-ui.min.css') }}**"** rel=**"stylesheet"**>  
 <link href=**"**{{ asset('jqueryui/jquery-ui.theme.min.css') }}**"** rel=**"stylesheet"**>  
 <link href=**"**{{asset('assets/css/style.css')}}**"** rel=**"stylesheet"**>  
  
  
 <link href=**"**{{asset('assets/css/flexslider.css')}}**"** type=**"text/css"** media=**"screen"** rel=**"stylesheet"**/>  
 <link href=**"**{{asset('assets/css/jquery.fancybox.css')}}**"** rel=**"stylesheet"**>  
 <link href=**"**{{asset('assets/css/cloud-zoom.css')}}**"** rel=**"stylesheet"**><link rel=**"shortcut icon"** href=**"assets/ico/favicon.html"**>  
  
</head>  
<body>  
  
  
<div id=**"maincontainer"**>  
 @include('partials.header')  
 @yield('content')  
 @include('partials.footer')  
</div>  
  
*<!-- javascript  
 ================================================== -->  
<!-- Placed at the end of the document so the pages load faster -->*<script src=**"**{{asset('assets/js/jquery.js')}}**"**></script>  
<script src=**"**{{asset('assets/js/bootstrap.js')}}**"**></script>  
<script src=**"**{{asset('assets/js/respond.min.js')}}**"**></script>  
<script src=**"**{{asset('assets/js/application.js')}}**"**></script>  
<script src=**"**{{asset('assets/js/bootstrap-tooltip.js')}}**"**></script>  
<script defer src=**"**{{asset('assets/js/jquery.fancybox.js')}}**"**></script>  
<script defer src=**"**{{asset('assets/js/jquery.flexslider.js')}}**"**></script>  
  
<script type=**"text/javascript"** src=**"**{{asset('assets/js/jquery.tweet.js')}}**"**></script>  
<script src=**"**{{asset('assets/js/cloud-zoom.1.0.2.js')}}**"**></script>  
<script type=**"text/javascript"** src=**"**{{asset('assets/js/jquery.validate.js')}}**"**></script>  
<script type=**"text/javascript"** src=**"**{{asset('assets/js/jquery.carouFredSel-6.1.0-packed.js')}}**"**></script>  
<script type=**"text/javascript"** src=**"**{{asset('assets/js/jquery.mousewheel.min.js')}}**"**></script>  
<script type=**"text/javascript"** src=**"**{{asset('assets/js/jquery.touchSwipe.min.js')}}**"**></script>  
<script type=**"text/javascript"** src=**"**{{asset('assets/js/jquery.ba-throttle-debounce.min.js')}}**"**></script>  
<script defer src=**"**{{asset('assets/js/custom.js')}}**"**></script>  
  
  
@yield('scripts')  
</body>  
</html>

**Figure l. Code Snippets Master Layout for generating views**