



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Babatunde Julius Adewumi

AN E-COMMERCE WEB APPLICATION
FOR A SMALL RETAIL STORE

Technology and Communication
2017

ACKNOWLEDGEMENT

My unreserved gratitude goes to God, who gave me the opportunity to complete this journey. To my family, I say thank you for your support. To all my friends I say am grateful that you all came my way.

Special thanks to my supervisor, Ms. Pirjo Prosi for her support and patience throughout this thesis project.

Thanks to Dr. Ghodrat Moghadampour for the knowledge I gained from him and for pushing me to the extreme during his programming classes. I actually learnt a lot from him. Also, thanks to Dr. Seppo Mäkinen and Dr. Jarmo Mäkelä for their support.

ABSTRACT

Author	Babatunde Julius Adewumi
Title	An E-commerce Web Application for a Small Retail Store
Year	2017
Language	English
Pages	65 + 5 Appendices
Name of Supervisor	Pirjo Prosi

In recent times, it has become necessary for any business to have an online presence in order to remain relevant and competitive. As a result of this necessity many businesses, including small enterprises, now operate an e-commerce web store so as to increase sales and attract new customers. Also, business owners do not have to worry about finding a place to erect their stores and customers can have unhindered access to a wide range of products at any time and anywhere in the world.

The objective of this thesis project was to develop an e-commerce Java web application for a small retail store where the store owner sells his/her products online. The application allows the owner to manage products, customers, and orders. Also, with the application customers make orders and pay for the ordered products. The application uses PayPal Express Checkout as its payment solution. In addition, the web store offers customers and visitors to the site an opportunity to subscribe to an email list in order to get news about new products and special offers. Lastly, the application sends an automatic email confirmation after completing an order or subscribing to an email list.

The development of this application was carried out on Eclipse IDE using the Java programming language. The database communication of the application was implemented by using JPA and JPQL, and MySQL database was used to store the application data. The application was structured according to the Model-View-Controller (MVC) pattern. The model, the view and the controller layers were implemented by using JavaBeans, JSPs, and Servlet API respectively. The payment transaction of the application was carried out on PayPal Sandbox (testing environment) with different NVP API operations.

CONTENTS

ABSTRACT

1	INTRODUCTION	10
1.1	Background	10
1.1.1	Brief History of E-commerce	10
1.1.2	Types of E-commerce	11
1.1.3	Why E-commerce Website for Businesses?	11
1.2	Objective	11
1.3	Thesis Overview	12
2	RELEVANT TOOLS AND TECHNOLOGIES	13
2.1	Java Programming Language	13
2.2	JDK and JRE	13
2.3	Java EE	13
2.4	HTML and CSS	14
2.5	Servlet API	14
2.6	JSP Technology	14
2.7	JavaMail API	15
2.8	MySQL	16
2.9	JPA and JPQL	16
2.10	Eclipse IDE	17
2.11	Apache Tomcat	17
2.12	Apache POI	17
2.13	PayPal Express Checkout	18
3	ANALYSIS AND REQUIREMENTS	19
3.1	Application Description	19
3.2	Quality Function Deployment (QFD)	19
3.3	Analysis Models	21
3.3.1	Use Case Diagram	21
3.3.2	Class Diagram	23
3.3.3	Sequence Diagram	24
3.3.4	Component Diagram	26
3.3.5	Deployment Diagram	27

4	DATABASE AND GUI DESIGN	29
4.1	Database Design.....	29
4.2	GUI Design	30
4.2.1	Home Page	30
4.2.2	Shopping Cart Page.....	31
4.2.3	Email Subscription Page	32
4.2.4	Order Review Page	33
4.2.5	Order Confirmation Page	34
4.2.6	Admin Login Page	35
4.2.7	Admin Menu Page.....	36
4.2.8	Categories Page	37
4.2.9	Add Category Page	38
4.2.10	Products Page	39
4.2.11	Add Product Page.....	40
4.2.12	Email List Page	41
4.2.13	Report Download Form.....	42
4.2.14	Orders List Page.....	43
4.2.15	Payments List Page	44
5	IMPLEMENTATION	46
5.1	Home Page	46
5.2	Subscription to Email List	47
5.3	Add Product to the Shopping Cart	48
5.4	Update Item in the Shopping Cart	49
5.5	Remove Item from the Shopping Cart	50
5.6	Remove all Items from the Shopping Cart	50
5.7	Payment with PayPal Express Checkout	51
5.7.1	SetExpressCheckout API Call.....	51
5.7.2	GetExpressCheckoutDetails API Call.....	52
5.7.3	DoExpressCheckoutPayment API Call.....	53
5.8	Sending Email.....	54
5.9	Admin Login.....	55
5.10	Add Category	56

5.11 Add Product	57
5.12 Download Reports in Excel Format.....	58
6 TESTING	60
7 CONCLUSION	63
REFERENCES.....	64
APPENDICES.....	66

LIST OF FIGURES, TABLES AND CODE SNIPPETS

Figure 1. How email works /12/	16
Figure 2. Admin use case diagram.	22
Figure 3. Customer use case diagram.	23
Figure 4. Class diagram.	24
Figure 5. Admin login sequence diagram.	25
Figure 6. Order sequence diagram.	26
Figure 7. Component diagram.	27
Figure 8. Deployment diagram.	28
Figure 9. ER diagram.	30
Figure 10. Home page.	31
Figure 11. Shopping cart page.	32
Figure 12. Email subscription page.	33
Figure 13. Order review page.	34
Figure 14. Order confirmation page.	35
Figure 15. Admin login page.	36
Figure 16. Admin menu page.	37
Figure 17. Categories page.	38
Figure 18. Add category page.	39
Figure 19. Products page.	40
Figure 20. Add product page.	41
Figure 21. Email list page.	42
Figure 22. Report download form.	43
Figure 23. Orders list page.	44
Figure 24. Payments list page.	45
Figure 25. Add category sequence diagram.	66
Figure 26. Edit category sequence diagram.	66
Figure 27. Delete category sequence diagram.	67
Figure 28. Add product sequence diagram.	67
Figure 29. Edit product sequence diagram.	68
Figure 30. Delete product sequence diagram.	68
Figure 31. Display email subscribers' list diagram.	69
Figure 32. Display orders list sequence diagram.	69

Figure 33. Download orders report sequence diagram.....	70
Figure 34. Download email subscribers list sequence diagram.....	70
Table 1. Requirements prioritized according to QFD.....	20
Table 2. Software testing template and results.	60
Code Snippet 1. Retrieving available products from the database.....	47
Code Snippet 2. Method for subscribing to the email list.....	48
Code Snippet 3. Method for adding product to the shopping cart.	49
Code Snippet 4. Method for updating item quantity in the shopping cart.	50
Code Snippet 5. Method for removing item from the shopping cart.	50
Code Snippet 6. Method for removing all items from the shopping cart.....	51
Code Snippet 7. Method for SetExpressCheckout API call.....	52
Code Snippet 8. Method for GetExpressCheckoutDetails API call.....	53
Code Snippet 9. Method for DoExpressCheckoutPayment API call.	54
Code Snippet 10. Method for sending an email.....	55
Code Snippet 11. Form-based authentication configuration for admin login.	56
Code Snippet 12. Method for adding category data into the database.....	57
Code Snippet 13. Method for adding product data into the database.	58
Code Snippet 14. Method for downloading reports in excel format.....	59

LIST OF ABBREVIATIONS

API	Application Programming Interface
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
JPA	Java Persistence API
JPQL	Java Persistence Query Language
JSP	JavaServer Pages
EL	Expression Language
JSTL	JSP Standard Tag Library
JDBC	Java Database Connectivity
SMTP	Simple Mail Transfer Protocol
IMAP	Internet Message Access Protocol
POP	Post Office Protocol
RDBMS	Relational Database Management System
SQL	Structured Query Language
GUI	Graphical User Interface
UML	Unified Modeling Language
MVC	Model-View-Controller
NVP API	Name-Value Pair API
QFD	Quality Function Deployment

1 INTRODUCTION

Before the advent of e-commerce and the internet, consumers had to visit the traditional brick and mortar stores to purchase goods or services, and the sellers had to find a space where they could sell their products, but due to the arrival of e-commerce and the internet some decades ago shoppers do not have to visit these stores to make a purchase, neither do the sellers have to find a place to locate their stores. In fact, buying and selling without any form of e-commerce is unthinkable, complicated and cumbersome to many these days /1/.

E-commerce, which is now an integral part of many businesses, is used primarily to boost sales revenue, to attract new customers and to survive in today's competitive business environment /2/. Also, it has benefitted the customers as they now have easy access to a wide range of goods and services at any time and anywhere in the world /3/. Well-known examples of e-commerce companies are Amazon, eBay, and Zalando.

1.1 Background

E-commerce, also known as electronic commerce or e-business, is simply the buying and selling of goods and services via an electronic medium, mainly the internet. The usage of electronic commerce has been increasing rapidly in the last decades since its inception, prompting the majority of businesses to have an online platform. It is now essential for companies to do their business online, as virtually any kind of goods and services can be sold or purchased through the internet. /3/

1.1.1 Brief History of E-commerce

Electronic commerce started in the 1960s when Electronic Data Interchange (EDI) was used by companies to carry out their daily business transactions electronically. In 1979, Michael Aldrich invented online shopping from which the term *teleshopping* was coined. In 1990, Tim Berners-Lee invented the World Wide Web, and thereafter he was able to establish communication between a Hypertext Transfer Protocol (HTTP) client and a server through the internet, leading to the advent of Amazon and eBay in the 1990s. These two prominent online stores have revolutionized the e-commerce market since their inception as more and more online shops spring up every day. /4/

1.1.2 Types of E-commerce

Basically, there are four types of electronic commerce. They are:

1. Business to Business (B2B) – A situation where transactions take place between companies. For example, a computer manufacturing company selling computers to another company. /2/
2. Business to Consumer (B2C) – This takes place when a business sells directly to consumers. An example is when a customer buys a product from Amazon web store. /2/
3. Consumer to Business (C2B) – This happens when an individual or end user sells goods or services to companies /3/. This is reverse B2C. An example is when a paid Amazon advert is hosted on a consumer's website.
4. Consumer to Consumer (C2C) – Involves business transactions between consumers. An example is when a consumer wants to sell a used product to another consumer on eBay. /3/

1.1.3 Why E-commerce Website for Businesses?

In today's business world, it has become inevitable for any small, medium or large enterprise to have an e-business store. The following are some of the reasons a business should have an online presence.

1. To break the barrier posed by physical limitations.
2. To reach more shoppers in order to increase revenue.
3. To make products available to customers 24/7 globally.
4. To allow shoppers purchase goods at their own convenience, with just some mouse clicks.
5. To reduce the operational cost of running a business.
6. To provide better customer relations.

1.2 Objective

The aim of this thesis is to develop an e-commerce Java web application for a small retail store, where the store owner (also called the administrator or admin) can sell goods over the internet. In the application, the admin will be able to manage products,

customers, and orders, while the customers will be able to order and pay for products. The payment transaction will be carried out on PayPal testing environment (PayPal Sandbox). Also, the buyers will have the opportunity to subscribe to an email list in order to get announcements about new arrivals and sales promotions. Furthermore, there will be an email notification after completing an order or subscribing to an email list.

1.3 Thesis Overview

This thesis report contains seven chapters. Chapter 1 introduces the thesis work by giving some background knowledge of what e-commerce is, a brief history of e-commerce, reasons for operating an e-commerce web store and the objective of the thesis work. Chapter 2 presents some discussions about the relevant tools and technologies used to develop the e-commerce application. Chapter 3 deals with the analysis and requirements carried out during the development process of the application. Then chapter 4 deals with the database and Graphical User Interface (GUI) design for the application. Chapter 5 focuses on the implementation of the application, while chapter 6 presents the various tests carried out on the application, including the results of these tests. Lastly, the conclusion part of the thesis is given in chapter 7.

2 RELEVANT TOOLS AND TECHNOLOGIES

This chapter presents some discussions about the relevant tools and technologies used to develop the e-commerce web application. Some of the tools and technologies are Java programming language, JDK and JRE, Java EE, HTML, CSS, JSP technology (EL and JSTL), JavaMail API and MySQL. Others are JPA and JPQL, Eclipse IDE, Apache Tomcat, Apache POI and PayPal Express Checkout.

2.1 Java Programming Language

This is the main programming language used to develop the application. The Java Programming Language was formerly developed by Sun Microsystems as proposed by James Gosling. It was first released in 1995. It runs on Mac OS, Windows, the different versions of UNIX, and other platforms. Java is considered to be secure and robust. Also, it is multithreading and platform independent (unlike C and C++). /5/

2.2 JDK and JRE

The Java Development Kit (JDK) is a software development environment for developing applets and applications written in Java. It consists of an interpreter, a Java compiler, a documentation generator (JavaDoc) and several other tools for building Java applications. The Java Runtime Environment is a component of JDK, and it consists of the Java Virtual Machine (JVM), libraries, files and other components for running applications written in Java. JVM is an implementation of JRE for running Java bytecode. /6/

2.3 Java EE

A Java platform is a distinct software environment for running applications written in Java. The four Java programming language platforms are Java Standard Edition (Java SE), Java Enterprise Edition (Java EE), Java Micro Edition (Java ME) and JavaFX. Each of these platforms contains a JVM and an application programming interface (API). An API is a group of software components used for building software applications. /7/

The Java EE platform is a superset of the Java SE platform, and it provides a runtime environment, technologies, and APIs for building and running enterprise web

applications. The Java SE platform contains the core APIs of the Java programming language. All of the APIs in Java SE are also contained in Java EE. Some of the core technologies and APIs provided only by the Java EE platform are Servlet APIs, JavaMail API, JDBC, JSP, and JPA. /7/

2.4 HTML and CSS

HTML stands for Hyper Text Markup Language. It is used as the standard markup language for building web applications and web pages. It was originally developed by Tim Berners-Lee in 1991. HTML5 is the latest version of HTML. /8/

CSS stands for Cascading Style Sheet. It is used to style web pages for different kinds of devices and screen sizes. It also saves a lot of time and stress since it can be used to style multiple web pages simultaneously. Its latest version is CSS3. /9/

2.5 Servlet API

A servlet is a Java class used for extending the functionality of web servers by providing dynamic contents for web applications. These servers are used for hosting web applications and are accessed through a request-response model. The servlet API consists of classes and interfaces for developing Java servlets and is a component of the Java EE platform. /10/

2.6 JSP Technology

JavaServer Pages (JSP) technology is used to build server-independent and platform-independent dynamic web applications /11/. It is one of the core technologies provided by the Java EE platform, and it has access to all the APIs contained in Java EE. Also, it stands as a better alternative to Microsoft's Active Server Page (ASP) and PHP as it is more powerful, secure and portable.

The JSP tags have been in use prior to JSP 2.0 specification. These JSP tags are used to insert Java code into a JSP. Using these tags in a JSP results in an awkward combination of Java code and HTML, and, therefore, efforts should be taken to avoid this practice unless when maintaining legacy web applications. Some of these tags are JSP scriptlet, JSP expression, and JSP declaration. /12/

Expression Language (EL) and JSP Standard Tag Library (JSTL) were introduced with the JSP 2.0 specification. EL is used to easily get access to data stored in JavaBeans components, while JSTL consists of tags used to accomplish common tasks in JSP. Some of the tasks supported by JSTL are iteration and conditionals, manipulation of XML documents and internalization tasks. It is usually a best practice to use EL and JSTL instead of the JSP tags in a JSP. EL has a more fancy and simple syntax than JSP tags. With EL it is possible to access collections like lists and maps, and nested properties. Also, EL has more functionality and can handle null values better than JSP tags. /12/

2.7 JavaMail API

This high-level API provides the possibility to automatically send an email using Java. It is a platform-independent and protocol-independent framework, which depends on the JavaBeans Activation Framework (JAF) API. The diagram in Figure 1 shows the mechanism of how email works. When sending an e-mail, the message is first sent from the mail client software to the mail server software using Simple Mail Transfer Protocol (SMTP), which is the most commonly used protocol to send an e-mail. Thereafter, the sender mail server utilizes SMTP to send the message to the recipient's mail server software. Lastly, the recipient's mail client utilizes Post Office Protocol (POP) or Internet Message Access Protocol (IMAP) to retrieve the email message from the recipient's mail server. /12/

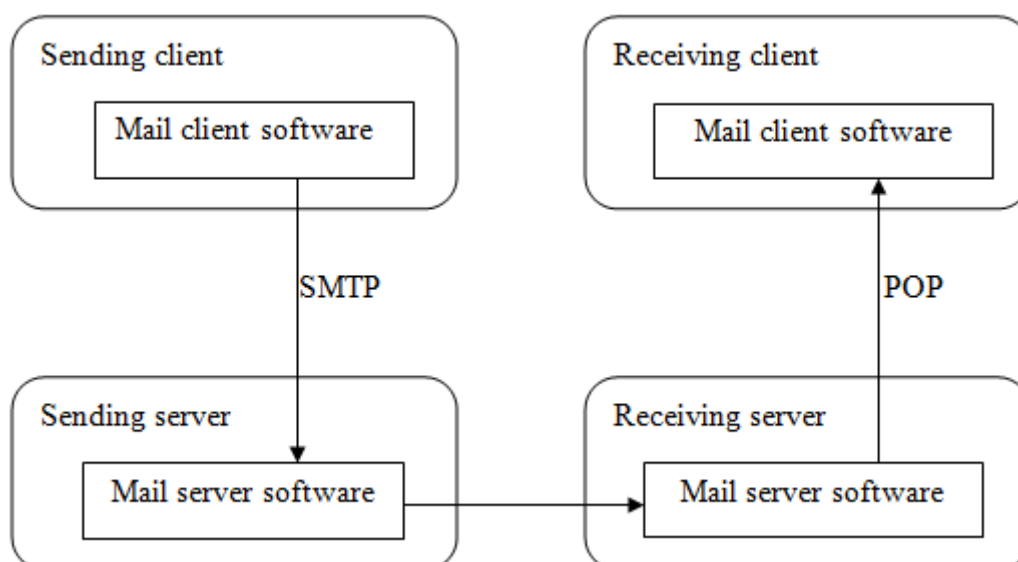


Figure 1. How email works /12/.

2.8 MySQL

MySQL is a free, open source relational database management system (RDBMS) that supports *Structured Query Language (SQL)*. An RDBMS is a system used to manage databases, and it is made up of tables containing columns and rows, where the tables are related by keys. MySQL DBMS is the world's most popular open source database system and one of the most commonly used database systems for Java web applications. It is one of the fastest RDBMS and it is easy to use. Also, it runs on almost all platforms, such as Windows, Linux and OS X. /12/

2.9 JPA and JPQL

Java Persistence API (JPA) provides an *object-relational (O/R) mapping* technique for transforming between business objects and relational database. The three most commonly used implementations of JPA are EclipseLink, Hibernate, and TopLink, with EclipseLink as the reference implementation. As a result, EclipseLink was used to implement JPA for this application. JPA runs on top of Java Database Connectivity (JDBC) and saves the programmer time and stress of writing SQL queries and JDBC code. In JPA, the business objects are referred to as entities. A JPA entity is a plain old Java object (POJO) with JPA annotations. /12/

The following are reasons for using JPA instead of JDBC:

1. JPA has the option to automatically create database tables according to the relationships between the business objects. /12/
2. No need to write JDBC and SQL code when using JPA. /12/
3. Conversion between objects and rows in a relational database is automatic when using JPA. /12/
4. Based on the relationships between business objects, JPA can perform automatic join operations. /12/

Java Persistence Query Language (JPQL) is an object-oriented query language defined by JPA. It is similar to SQL but operates on objects, attributes, and relationships instead of columns and rows. JPQL can be used to perform insert, select, update and delete operations just like SQL. /12/

2.10 Eclipse IDE

Eclipse is an open source integrated development environment (IDE) for developing applications in Java and other languages like C++, JavaScript, PHP and C. It is written in Java, and it is the most commonly used IDE for Java applications. Examples of Eclipse development environment are Eclipse IDE for Java, Eclipse PDT for PHP and Eclipse CDT for C/C++. The latest release of Eclipse is Eclipse Oxygen, which was released in June 2017. /13/

2.11 Apache Tomcat

The Apache Tomcat is an open source web server and servlet container developed and released by the Apache Software Foundation (ASF). It is an implementation of Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. Its latest version is Tomcat 8.0.45. /14/

2.12 Apache POI

Apache POI (Poor Obfuscation Implementation) is a Java API for reading and writing Microsoft Excel files using Java programs. Also, it is utilized for reading and writing Microsoft Word and Microsoft PowerPoint files using Java. It supports different file formats, such as xsl, xlsx and docx. It is an open source API developed and released by ASF. /15/

2.13 PayPal Express Checkout

PayPal is an online payment service owned by PayPal Holdings Inc, an American company that was established in 1998. PayPal provides any business or individual with an email address a secure, easier and worthwhile way to pay and get paid online. In addition, it gives a worldwide, instant payment solution by using existing infrastructure of bank accounts and credit cards. /16/

PayPal Express Checkout streamlines the checkout experience for buyers and still keeps them on the seller's website after completing a purchase. It allows buyers to make use of their PayPal balance, bank account or credit card to make payment without any need to enter delicate information. It is readily available in any country where PayPal is accepted. /17/

3 ANALYSIS AND REQUIREMENTS

Requirements analysis is one of the major tasks in software engineering, which is vital to the success of a software development project. It involves the determination of the requirements or functions of a software project /18/. The main task to perform before analyzing requirements is *requirements elicitation* /19/.

Requirements elicitation is the gathering of the requirements or needs of a software system from the client and other stakeholders involved in the software project. Some of the activities involved in requirements elicitation are interviews, meetings, and surveys. Three requirements elicitation techniques are Initiating the Process, Facilitated Application Specification Techniques (FAST), and Quality Function Deployment (QFD). /19/

The requirements for this application were gathered based on QFD. This is because QFD prioritizes both explicit and implicit requirements for the software. Also, it focuses on client satisfaction all through the development process /20/.

3.1 Application Description

This application is divided into two parts – the home page and the admin page. The home page is where customers (buyers) can order and pay for products, and optionally subscribe to an email list while the admin page is where the admin can carry out administrative tasks. The admin page is restricted and can only be accessed through authentication provided by the Apache Tomcat servlet container. This means that all the web resources in the admin page can only be accessed by an authorized user.

3.2 Quality Function Deployment (QFD)

QFD is a requirements elicitation technique used to convert client's requirements and expectations into technical requirements for the software product. It aims at building a software system that fulfills client satisfaction by focusing on what is relevant to the client. More so, it utilizes different methods, such as interviews, surveys, and review of historical data to achieve its objectives. /19/

According to user needs and expectations, QFD prioritizes requirements into three types – they are:

1. Normal requirements – These are must have requirements with priority level 1. They are requirements that fulfill client satisfaction if present. /19/
2. Expected requirements – These are should have requirements with priority level 2. They are requirements that are not explicitly declared by the client but could be a reason for customer dissatisfaction if not accomplished. /19/
3. Exciting requirements – These are nice to have requirements with priority level 3. They are needs that are beyond the scope of the project but could result in client satisfaction when present. /19/

The requirements for this application as prioritized according to QFD are shown in Table 1.

Table 1. Requirements prioritized according to QFD.

Normal Requirements
<ol style="list-style-type: none"> 1. The home page of the application should display available products whose quantity is not less than one. 2. Customers should be able to add products to a shopping cart. 3. Customers should be able to view products in the shopping cart. 4. Customers should be able to update product quantity in the cart. 5. Customers should be able to remove any product from the cart. 6. Customer should be able to empty all the products in the cart. 7. Customers should have an option to subscribe to an email list. 8. Customers should be able to checkout through PayPal Express Checkout payment solution on Paypal Sandbox. 9. Customers should be able to view order confirmation after a successful order completion. 10. The admin should be authenticated in order to have access to the admin page of the application to perform any administrative task. 11. The admin should be able to manage (add, update and delete) products and their categories. 12. The admin should be able to view the lists of products and categories. 13. The admin should be able to view the email subscribers' list. 14. The admin should be able to view payment and order details. 15. The admin should be able to update order status. 16. The application should save all customer, product, order, payment and admin data on MySQL database.
Expected Requirements
<ol style="list-style-type: none"> 1. The application should display a friendly error message after a failed login by the admin.

- | |
|--|
| <ol style="list-style-type: none"> 2. The application should display a friendly error message after a failed or canceled payment process. 3. The application should implement interactive graphical user interfaces (GUI). 4. An email confirmation should be sent to customers who subscribe to the email list. 5. An email confirmation should be sent to customers after a successful order completion. 6. The admin should be able to download email subscribers' list, payments and orders reports in MS excel format. |
|--|

Exciting Requirements

- | |
|---|
| <ol style="list-style-type: none"> 1. The application should implement other payment methods, such as debit/credit card and google checkout. 2. The application should also be available on mobile devices. |
|---|

3.3 Analysis Models

Modeling involves the designing of software systems before coding takes place. Modeling plays an important role in any software development project. It guarantees the completeness and correctness of a software system and the fulfillment of end-users' expectations. In addition, modeling serves as the only reference point to cross-check requirements before coding. /19/

A Unified Modeling Language (UML) based tool was used to model this application. UML diagrams give both static and dynamic views of an application and it is well suited for object-oriented languages like Java and C# /19/. The following sub-sections present the UML diagrams used to model this application.

3.3.1 Use Case Diagram

The use case diagrams for this application illustrate the interactions that exist between users (actors) and use cases (actions) within the application. There are two actors identified for this application – administrator (admin) and customer actors. As a result, there are two use case diagrams for the software application – admin use case diagram and customer use case diagram. The admin is the owner of the e-commerce store who performs various administrative tasks such as add products, view orders, and update order status while the customer is any individual who buys a product or products from the online store.

Figure 2 shows the admin use case diagram. The diagram depicts how the admin communicates with the application. More so, it shows all the actions that the admin can perform on the application. As can be seen in the diagram, before any of these actions could be executed the admin will have to login in order to be authenticated.

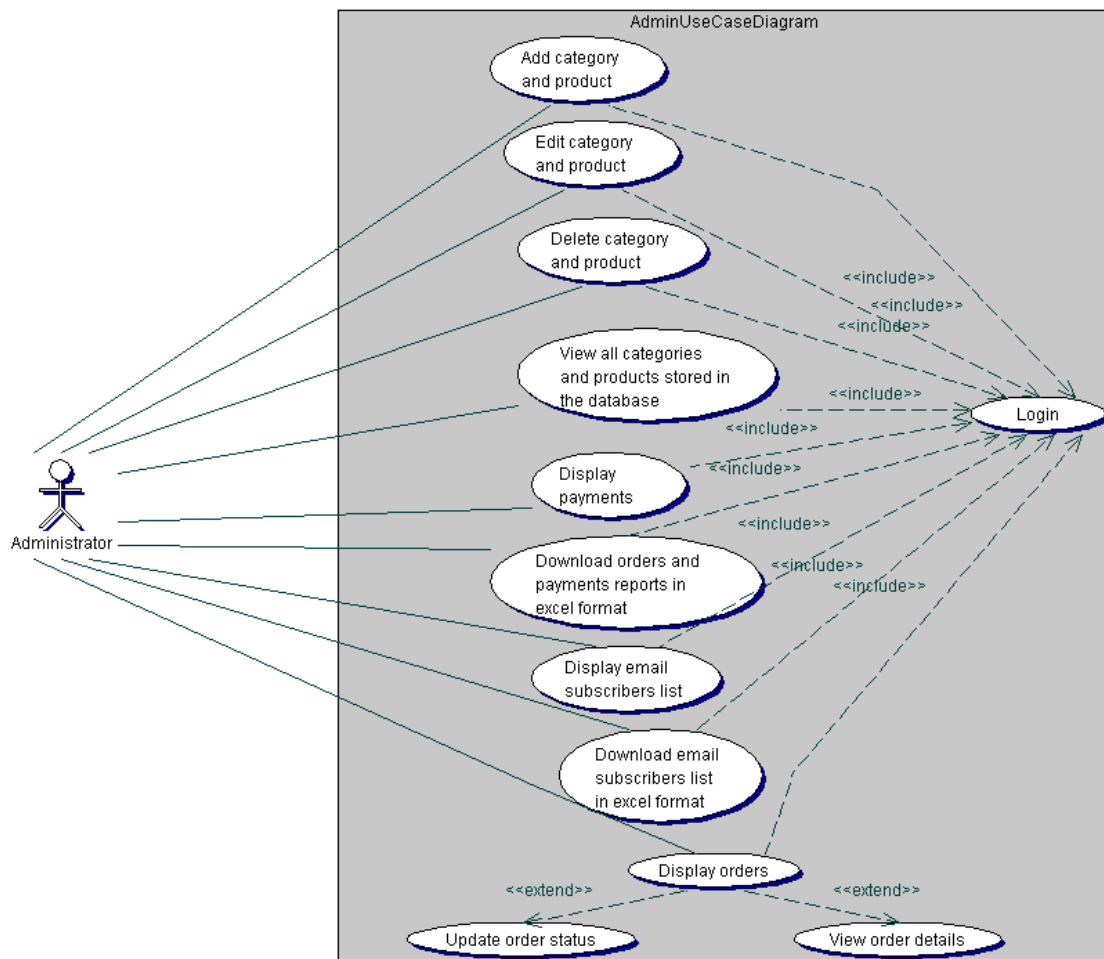


Figure 2. Admin use case diagram.

Figure 3 shows the customer use case diagram. It describes the different use cases that can be executed by the customer on the e-commerce application. For the checkout process using PayPal Express Checkout, the buyer will have to be authenticated on a secured PayPal website.

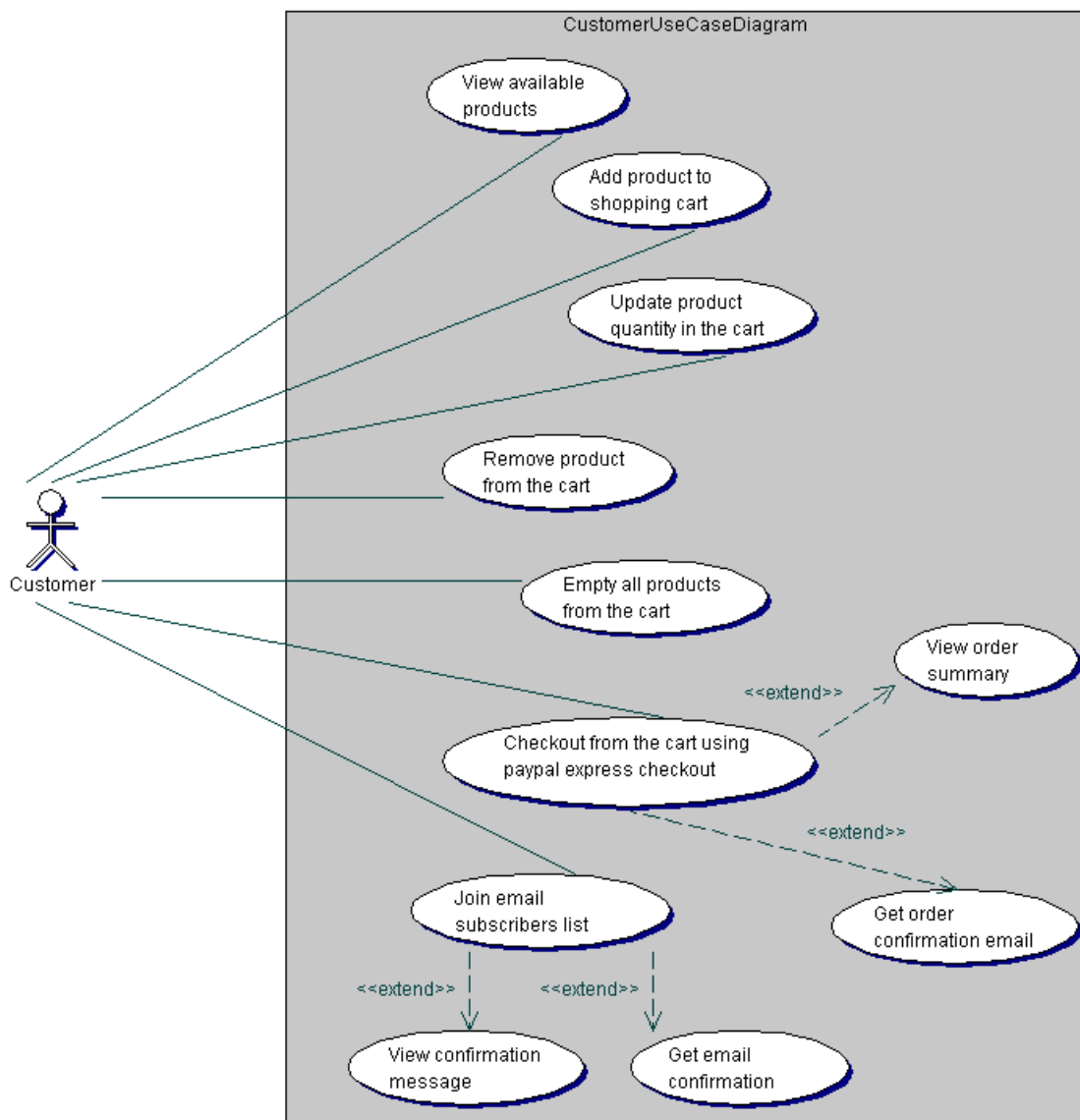


Figure 3. Customer use case diagram.

3.3.2 Class Diagram

A class diagram depicts the classes in a software system and how they interact with each other. Also, the class attributes and functions are illustrated in a class diagram. Figure 4 shows the class diagram for this application. It shows the relationships between classes in the application and constraints applied to these relationships.

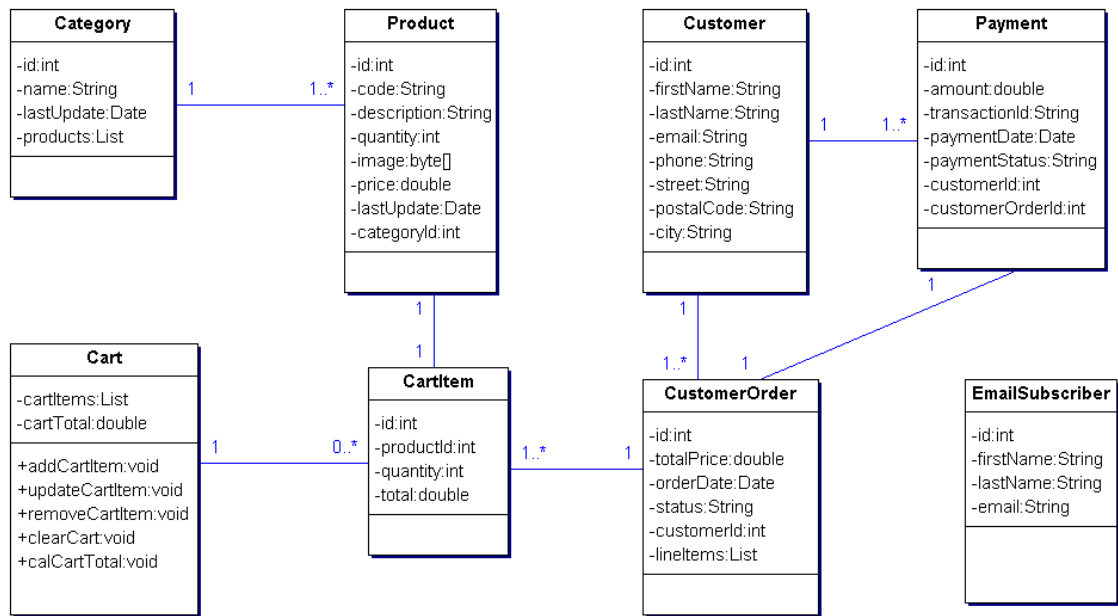


Figure 4. Class diagram.

3.3.3 Sequence Diagram

A sequence diagram gives a detailed visual description of how the various classes in a system interact with each other. Also, it depicts the order in which different objects exchange messages with one another in a system. The sequence diagrams for this application are presented in the following sub-sections.

3.3.3.1 Admin Login Sequence Diagram

Figure 5 gives a detailed sequence of events needed for the admin to login to the admin page of the application. After providing the login credentials on the admin login page, the admin is authenticated through a form-based authentication method provided by the Apache Tomcat web server. After a successful authentication process, the admin is forwarded to the admin menu page, which contains links to several administrative functions. However, if the authentication process fails, the admin is redirected to the admin login page with displayed error message. For the remaining sequence diagrams required for the administrative tasks, see Appendices.

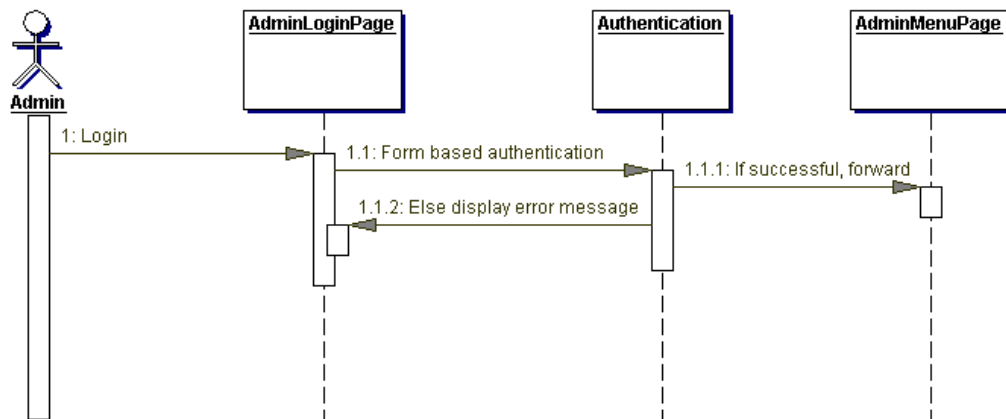


Figure 5. Admin login sequence diagram.

3.3.3.2 Order Sequence Diagram

The order sequence diagram is shown in Figure 6 below. On the home page of the application where all available products are displayed, the customer clicks the add product to cart link in order to add a product to the shopping cart. Then, the application returns the customer to the home page either to add more products to the cart or to view the shopping cart page. If the customer clicks the display shopping cart link, the shopping cart page is displayed. The shopping cart page contains all the line items in the cart and it is where the buyer can increase a line item quantity, remove a line item from the cart, and empty the entire cart line items. If the buyer performs any of these activities in the shopping cart, the application returns to the shopping cart to show the necessary updates in the shopping cart. If the customer clicks the continue shopping link on the shopping cart page, he/she is redirected to the home page of the application.

Furthermore, if the shopping cart is not empty and the buyer clicks the checkout with PayPal link, he/she is forwarded to the PayPal login page. On the PayPal login page, the customer login with his/her PayPal account credentials. If the login was successful, the customer is forwarded to PayPal order summary page, which shows summary information of all the ordered products, including the total price of the order. Otherwise, the buyer is redirected to PayPal login page to show an appropriate error message. On the PayPal order summary page, the customer clicks the continue link and is forwarded to the application's order review page where he/she can access the complete order link. Now, when the buyer clicks the complete order link, the database handler retrieves

customer, order and payment data. The handler updates the customer data on the database if the customer data already exists otherwise, it writes the data to the database. The handler also writes the order and payment data to the database.

Lastly, after the handler performs the necessary operations on the database, the customer is redirected to the order confirmation page to display order confirmation message. Also, an e-mail confirmation is sent to the customer.

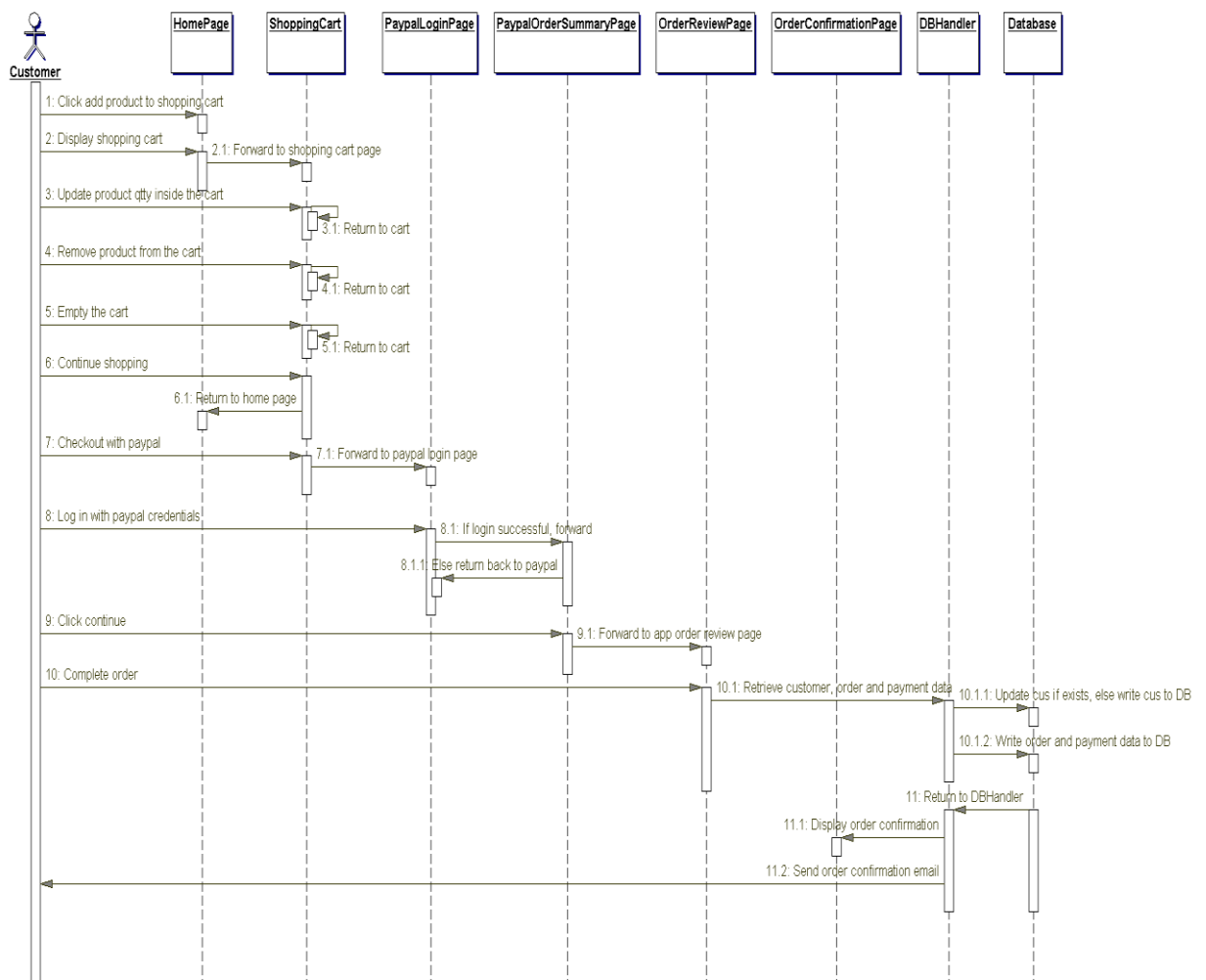


Figure 6. Order sequence diagram.

3.3.4 Component Diagram

A component diagram is used to depict the organizations of software components and the relationships that exist among them. Figure 7 illustrates the component diagram for this web application. It was modeled according to the Model-View-Controller (MVC)

pattern used for structuring web applications. The MVC pattern makes coding, testing and maintenance of an application easier and it is usually considered as a best practice.

As can be seen in Figure 7, the MVC pattern divides this application into three distinct layers: the model, the view, and the controller. The model is the business layer of the application, which contains the JavaBeans for the application. A JavaBean is simply a plain old Java object (POJO) used to encapsulate data. The view represents the presentation layer, which contains JSP files for displaying the various pages of the application. The controller controls the flow of data between the model and the view. It contains servlets for updating the model object and saving it to the database through the database handler. The servlets also update the view for presentation when necessary. In addition, the database handler consists of data access classes, which provide methods for storing data in the database since the JavaBeans do not provide these methods.

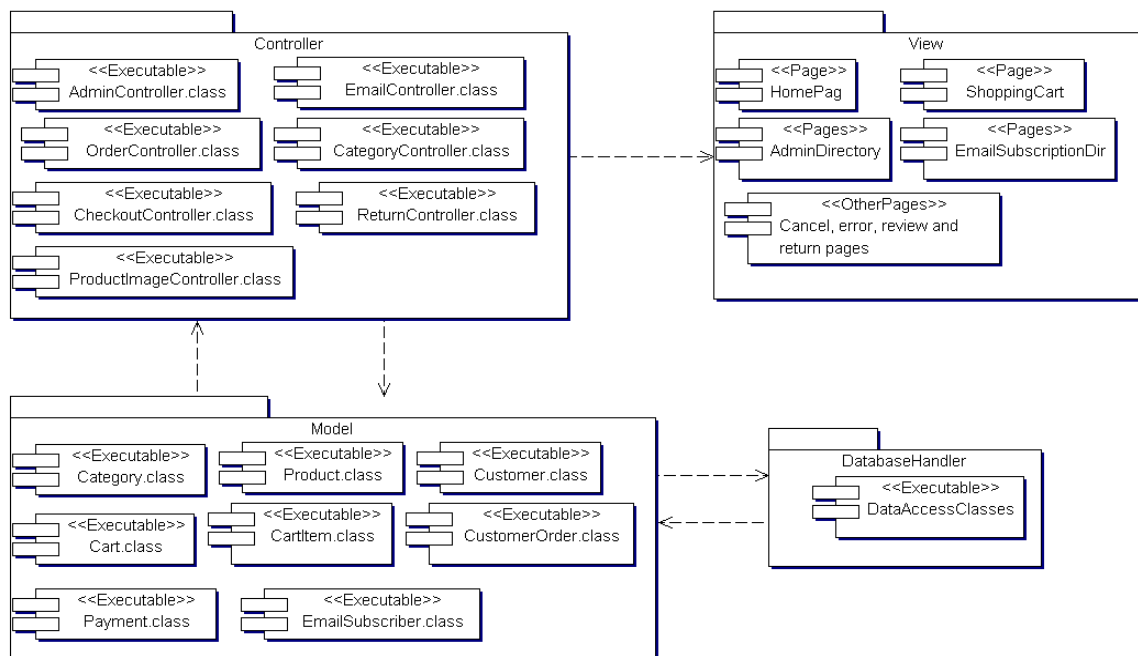


Figure 7. Component diagram.

3.3.5 Deployment Diagram

The deployment diagram for this application is illustrated in Figure 8. The diagram shows the configuration of the run-time hardware components (nodes) and the software components running on those nodes. As can be seen in Figure 8, to deploy this web

application a database server, an application server, and computers with internet access are needed. Also, backup servers are provided for the database and application servers.

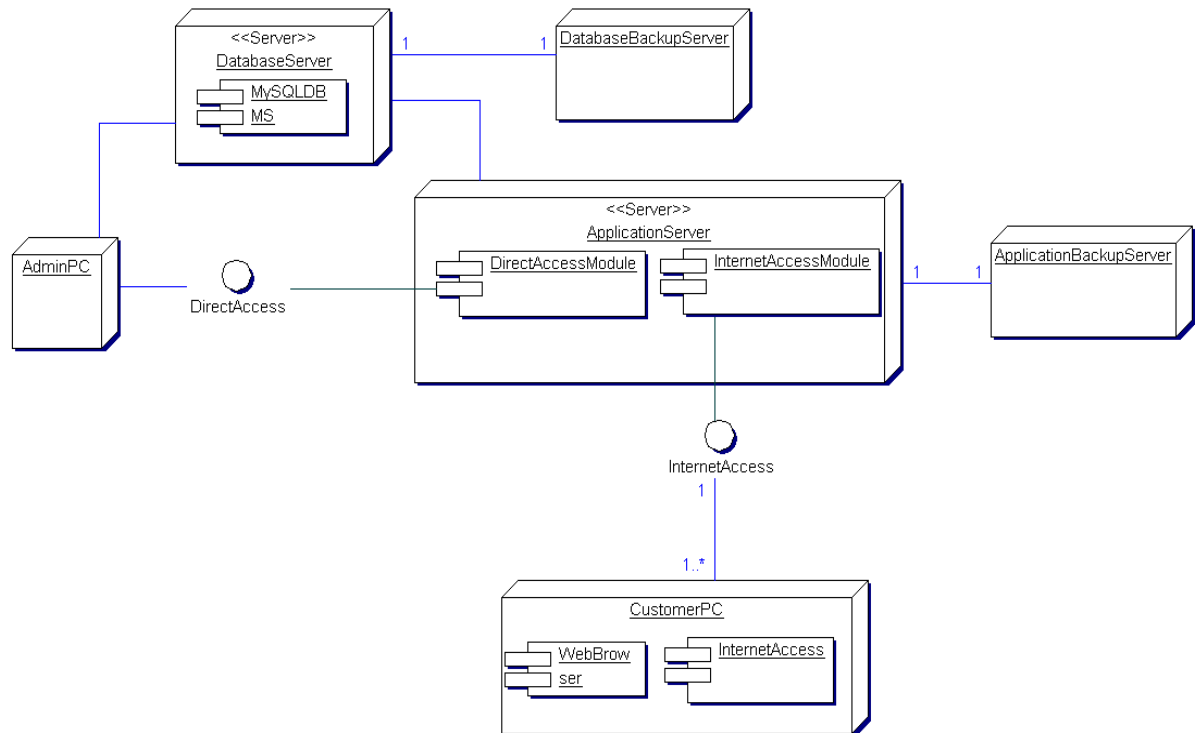


Figure 8. Deployment diagram.

4 DATABASE AND GUI DESIGN

Every web application needs to have an operational database to store its generated data. More so, a simple and interactive Graphical User Interface (GUI) is one of the success indicators of any web application project. This chapter presents the database and GUI design for this e-commerce application.

4.1 Database Design

MySQL database management system is the most popular database system for Java web applications because of its speed, reliability, and flexibility. All the data generated by this application are managed on MySQL database system. Figure 9 shows the Entity Relationship Diagram of the application database. All the tables (except admin and admin_roles) were generated by JPA based on the relationships that exist among the JPA entities defined for the application, and these relationships are clearly evidenced in the ER diagram. The admin and admin_roles tables were defined for authentication purpose. The following numbered list gives a brief explanation of these tables.

1. Category – Contains category data for the products.
2. Product – Contains product data.
3. Customer – Contains customer data.
4. Customer_order – Contains customer order information.
5. Order_line_item – Contains order line items information.
6. Customer_order_order_line_item – This is a bridge table automatically generated as a result of the one-to-many relationship that exists between customer_order and order_line_item tables.
7. Payment – This contains order payment data.
8. Sequence – Contains data used to automatically generate primary key values.
9. Email_subscriber – Contains email subscribers information.
10. Admin – Contains the administrator credentials for login (authentication) purpose.
11. Admin_roles – Contains roles data of admin.

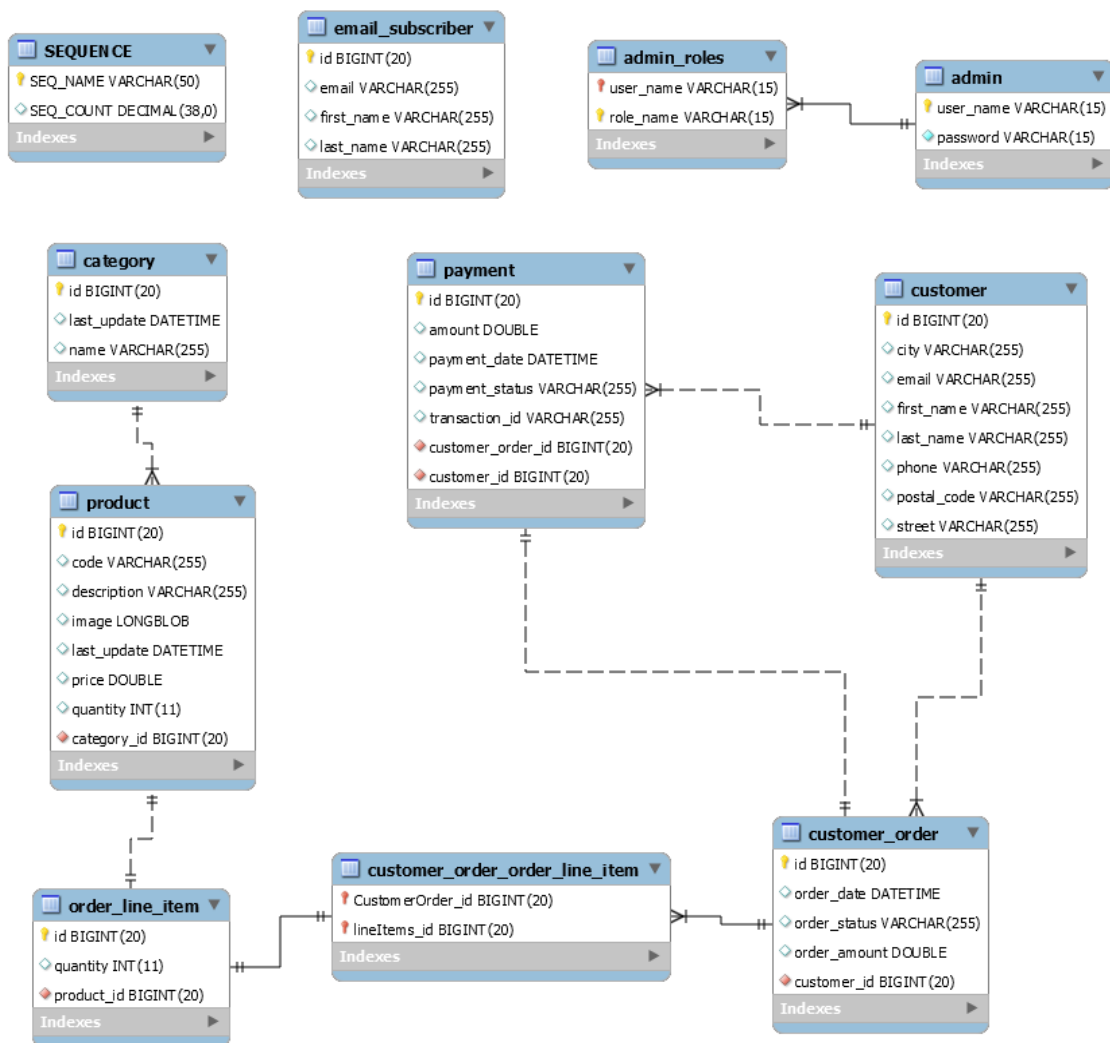


Figure 9. ER diagram.

4.2 GUI Design

The user interfaces for this e-commerce web application were designed using HTML and CSS. Also, some elements of HTML5 were used in designing the interfaces. The following subsections present the descriptions of these interfaces.

4.2.1 Home Page

The home page of this application is shown in Figure 10. It is the start page of the application where all available products with a quantity greater than zero are displayed. Also, customers or visitors to the e-commerce store can browse available products by category. On this page, customers can add products to the shopping cart, and there is a link provided to view products added to the shopping cart. More so, a link is provided

for customers and visitors to display and fill the email subscription form in order to subscribe to an email list for special announcements and offers.

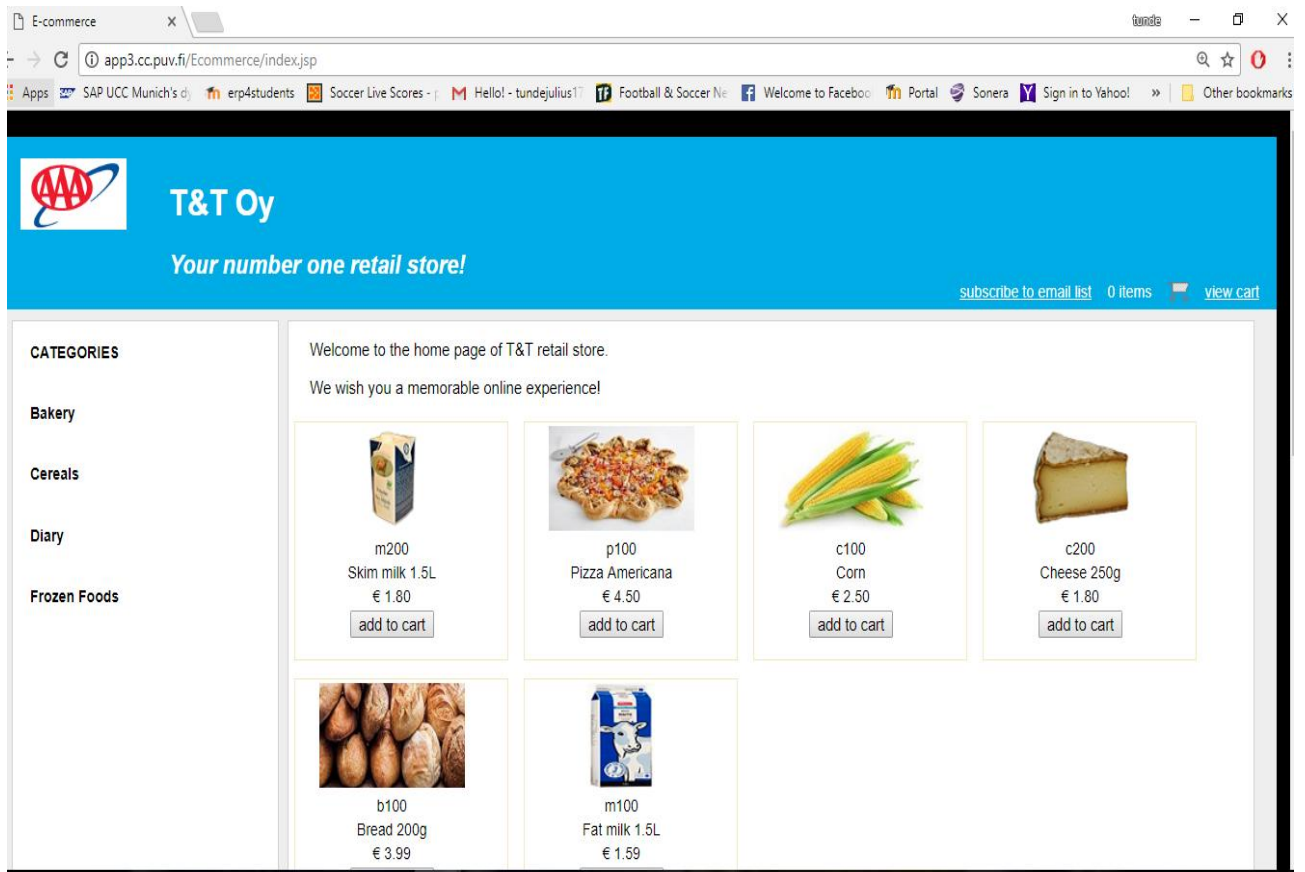


Figure 10. Home page.

4.2.2 Shopping Cart Page

The shopping cart page of this online store application is as shown in Figure 11. This page is accessed when a user clicks the view cart link to show the information of all the products added to the shopping cart. On this page, the quantity of any product in the cart can be changed and updated. Any product can be removed from the cart and the cart can be emptied as well. There are links provided for browsing available products by category and also to return to the home page of the application. In addition, the PayPal check out button is displayed on this page if the shopping cart is not empty.

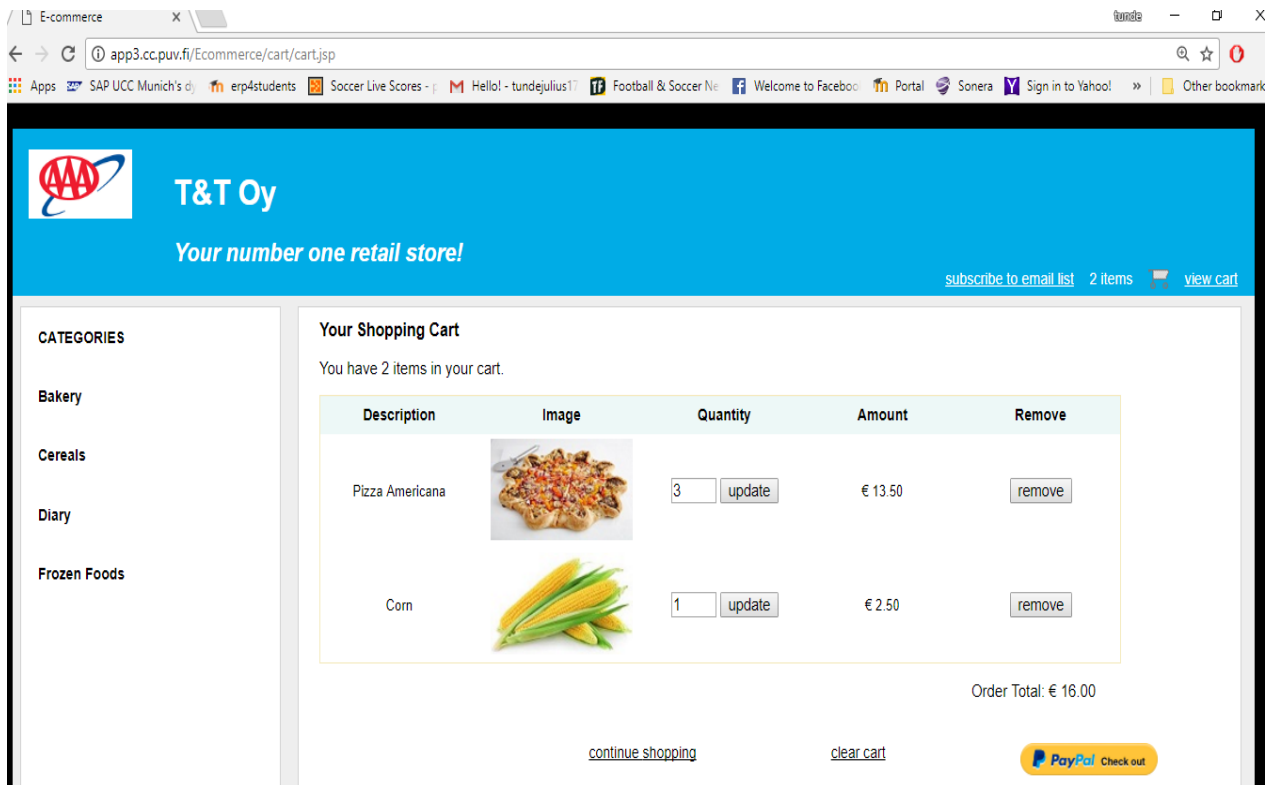


Figure 11. Shopping cart page.

4.2.3 Email Subscription Page

Figure 12 shows the email subscription page. This page is accessed when a user clicks the subscribe to email list link to display the email subscription form where a user can enter and submit the subscription information. The form does not accept null or empty values. After a successful subscription process, the user is redirected to a confirmation page which displays a confirmation message.

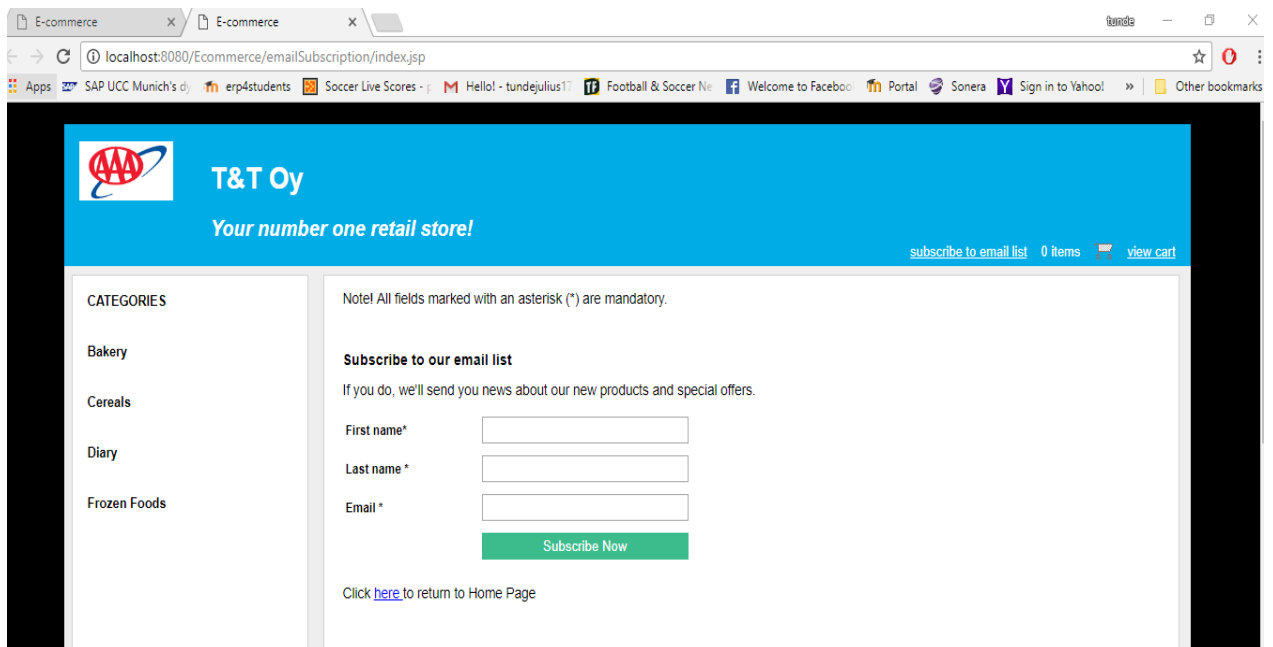


Figure 12. Email subscription page.

4.2.4 Order Review Page

The order review page of this application is as shown in Figure 13. This page shows an order review just before confirmation. It displays the customer's shipping address, order amount, currency and payment status of the order. A button is provided to confirm the order.

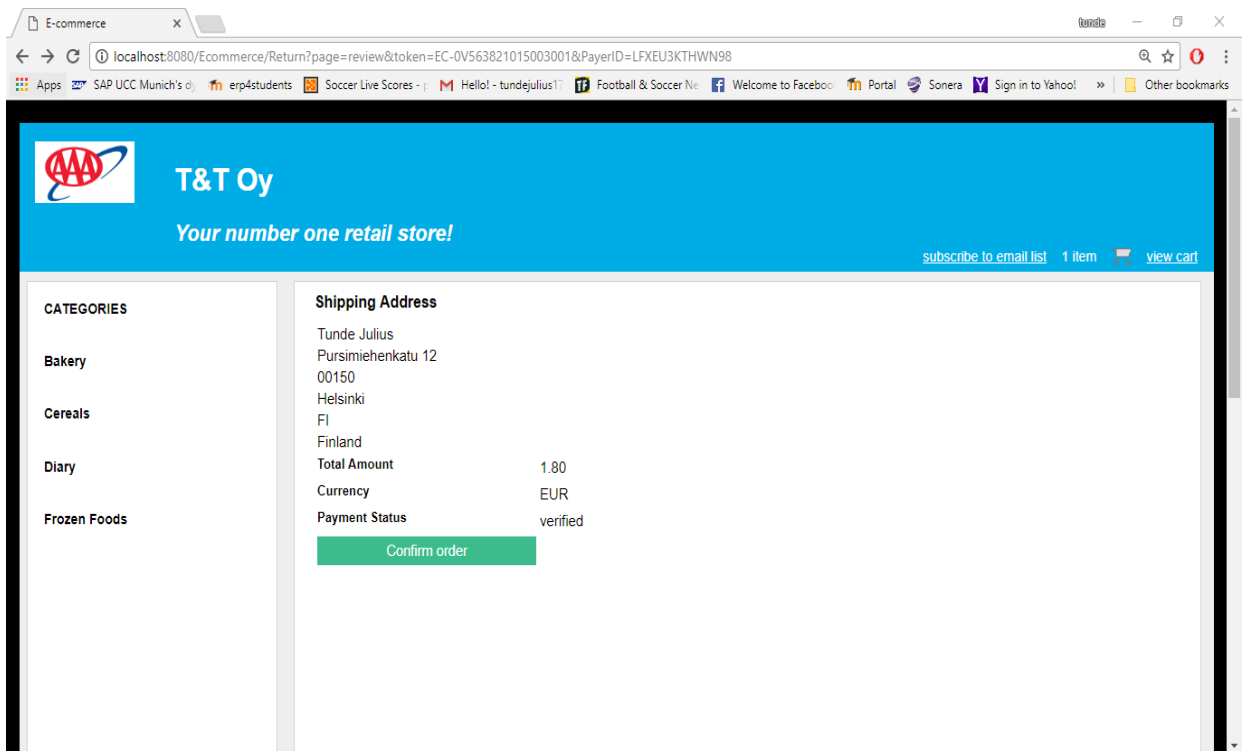


Figure 13. Order review page.

4.2.5 Order Confirmation Page

Shown in Figure 14 is the order confirmation page of this application. This page displays a confirmation message after a successful order process. It shows the customer's shipping information and the details of the payment.

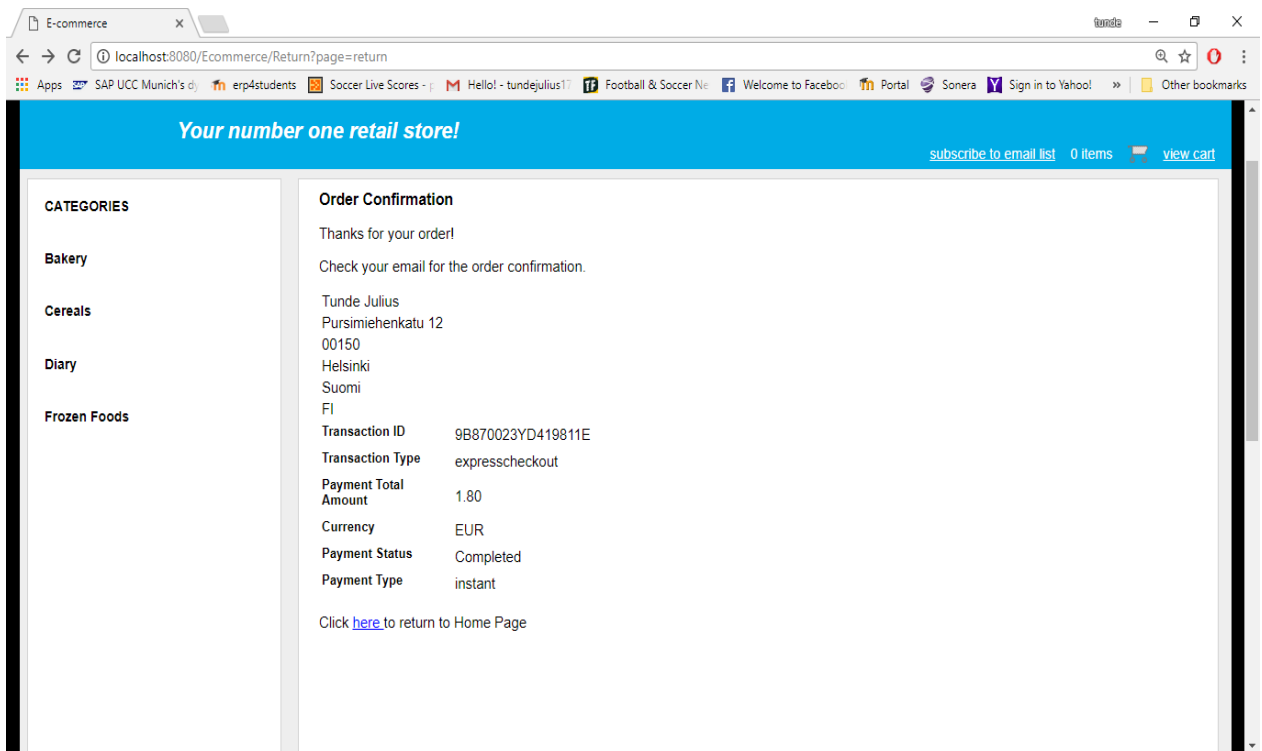


Figure 14. Order confirmation page.

4.2.6 Admin Login Page

Figure 15 shows the admin login page. This page presents the admin login form where a user can enter and submit login credentials for authentication purpose. It is actually a customized form for form-based authentication method provided by Apache Tomcat servlet container. More so, it is not possible to enter null or empty string values on the form. The page is displayed whenever a user tries to access the admin page of the application which is restricted and can only be accessed by authorized users. If the authentication fails the same page is displayed again but with an error message.

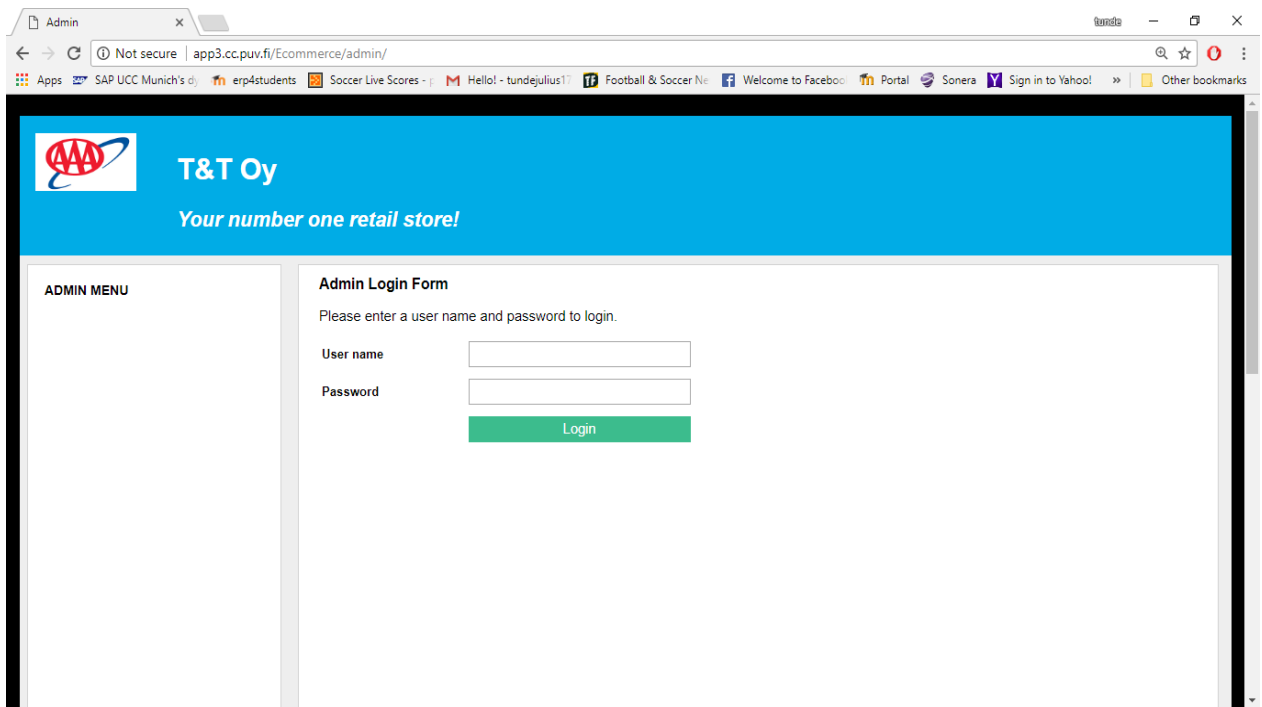


Figure 15. Admin login page.

4.2.7 Admin Menu Page

The admin menu page is shown in Figure 16 and it is displayed when a user has been successfully authenticated. The page is the gateway to the administrative tasks for the admin. The links for these tasks have been categorized so that they can be easily accessible.

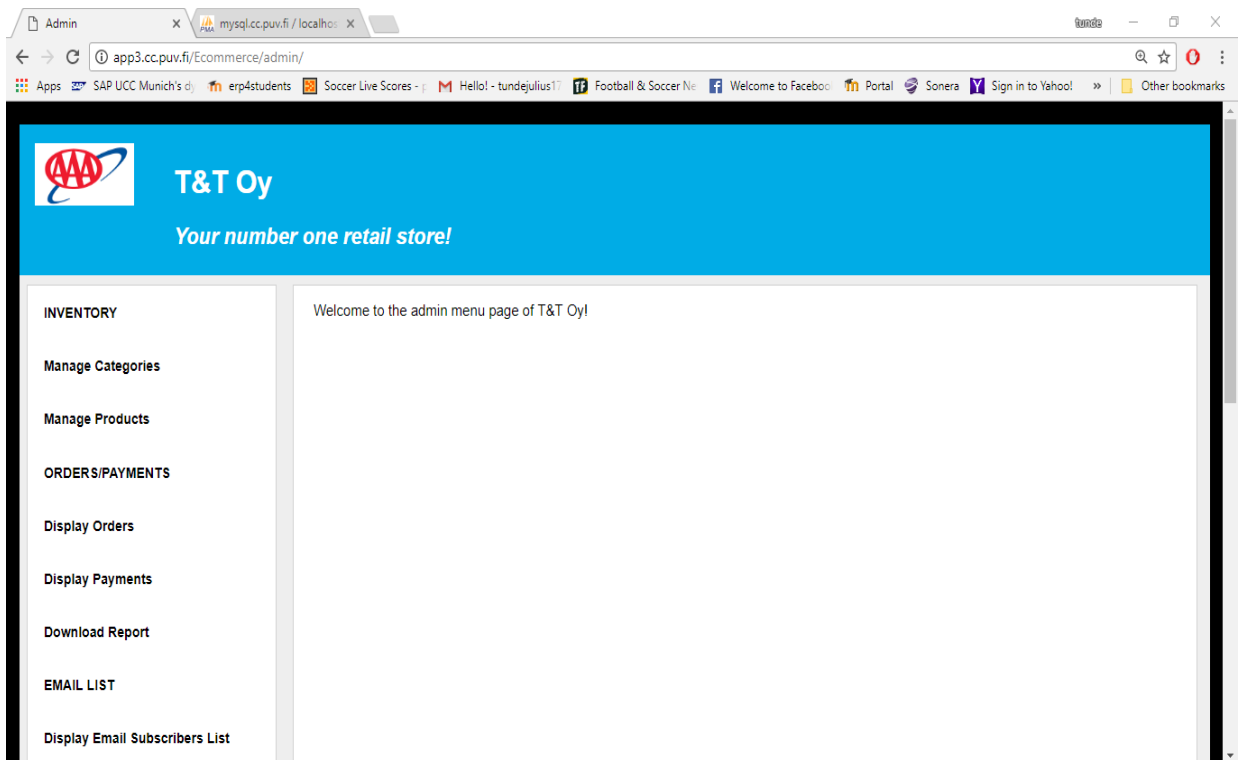


Figure 16. Admin menu page.

4.2.8 Categories Page

The categories page is the page for managing product categories for the e-commerce store, and it is accessed by clicking the manage categories link on the admin menu page. It is shown in Figure 17 below. All the product categories stored in the database are displayed on this page, and each of these categories has links for displaying all its products, editing its data and deleting it from the database. Also, at the top of the page there is a link for displaying a form for adding categories.

T&T Oy
Your number one retail store!

INVENTORY

- Manage Categories
- Manage Products

ORDERS/PAYMENTS

- Display Orders
- Display Payments
- Download Report

EMAIL LIST

- Display Email Subscribers List

Product Categories

To add a category, click [here](#).

ID	Name	Last Update	Action
2	Frozen Foods	01.06.2017 07:53	Display products Edit Delete
3	Diary	01.06.2017 07:53	Display products Edit Delete
7	Bakery	01.06.2017 07:57	Display products Edit Delete
8	Cereals	01.06.2017 08:00	Display products Edit Delete

Figure 17. Categories page.

4.2.9 Add Category Page

The page for adding product categories is shown in Figure 18. The page displays the add category form when an admin user clicks the link for adding a category on the categories page. The form checks for null and empty string values, thus, it is not possible to enter null or empty string values. The edit category form is basically the same as the add category form, except that it contains editable data in its field.

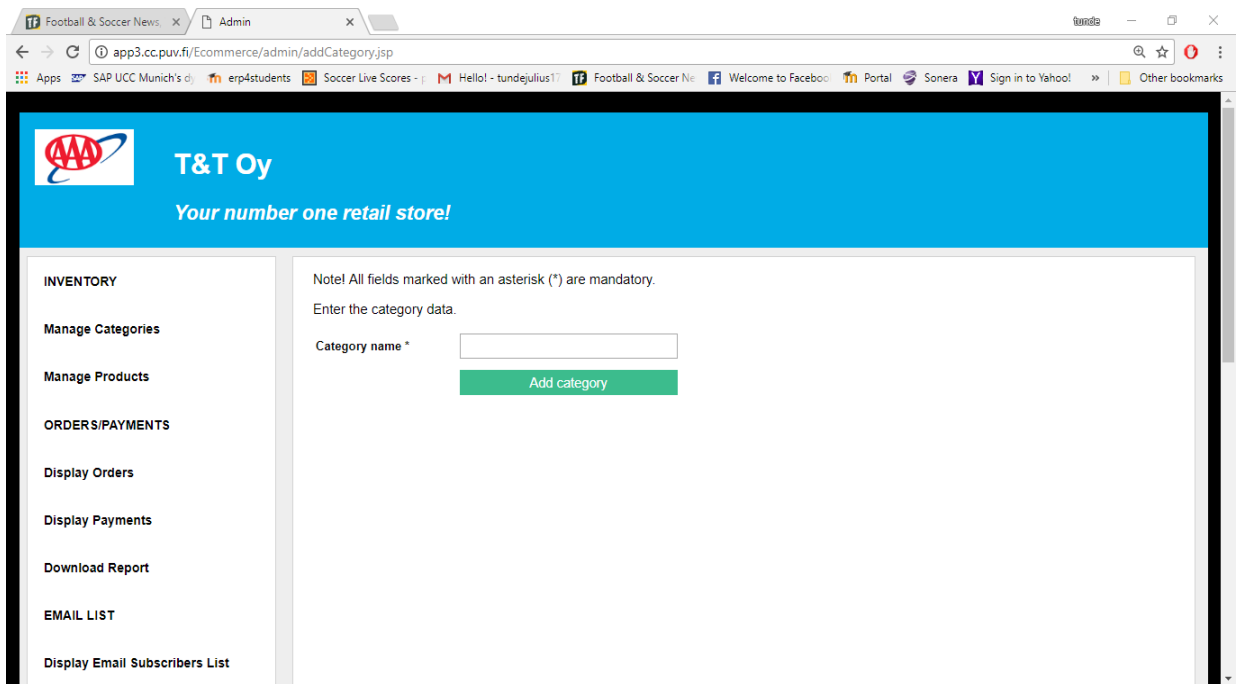
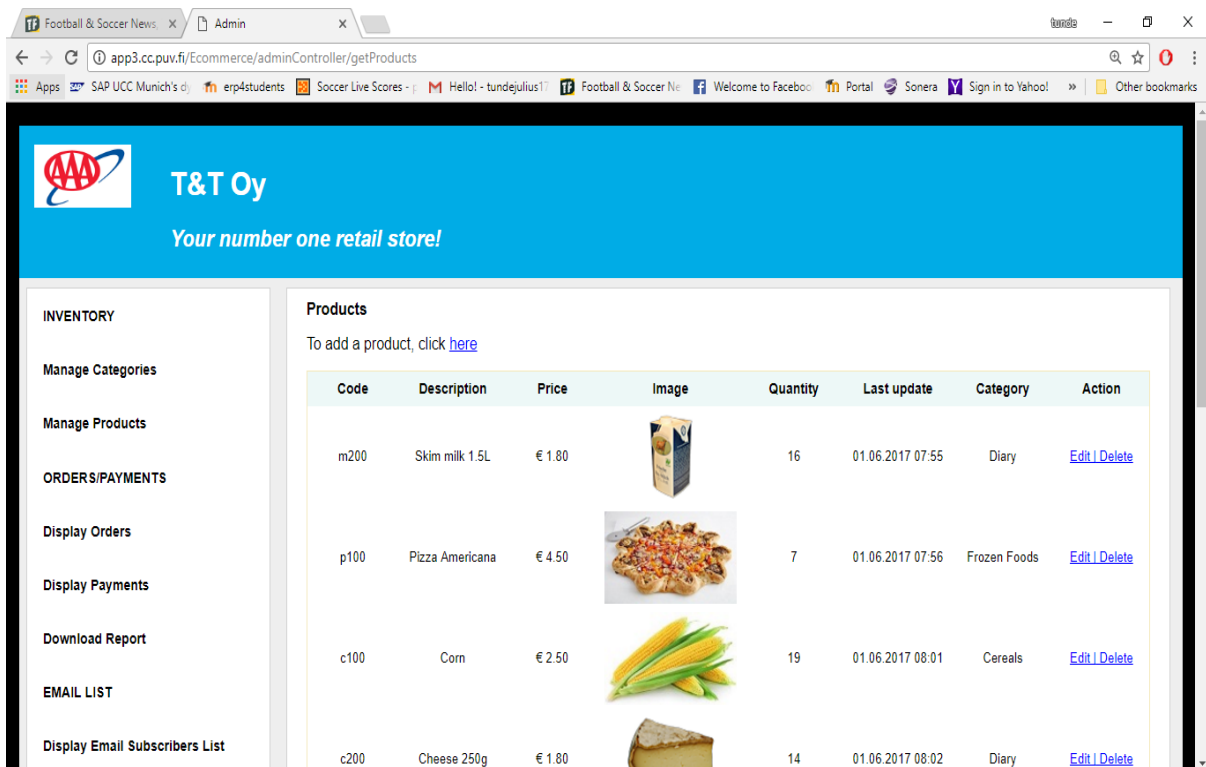


Figure 18. Add category page.

4.2.10 Products Page

Figure 19 shows the products page of this application. It is the page for managing products of the online store and is accessed by clicking the manage products link on the admin menu page. All products of the store are displayed on this page, and each of these products has links for editing its data and deleting it from the database. Also, at the top of the page there is a link for adding products.



T&T Oy
Your number one retail store!

INVENTORY

- Manage Categories
- Manage Products

ORDERS/PAYMENTS

- Display Orders
- Display Payments
- Download Report

EMAIL LIST

- Display Email Subscribers List

Products

To add a product, click [here](#)





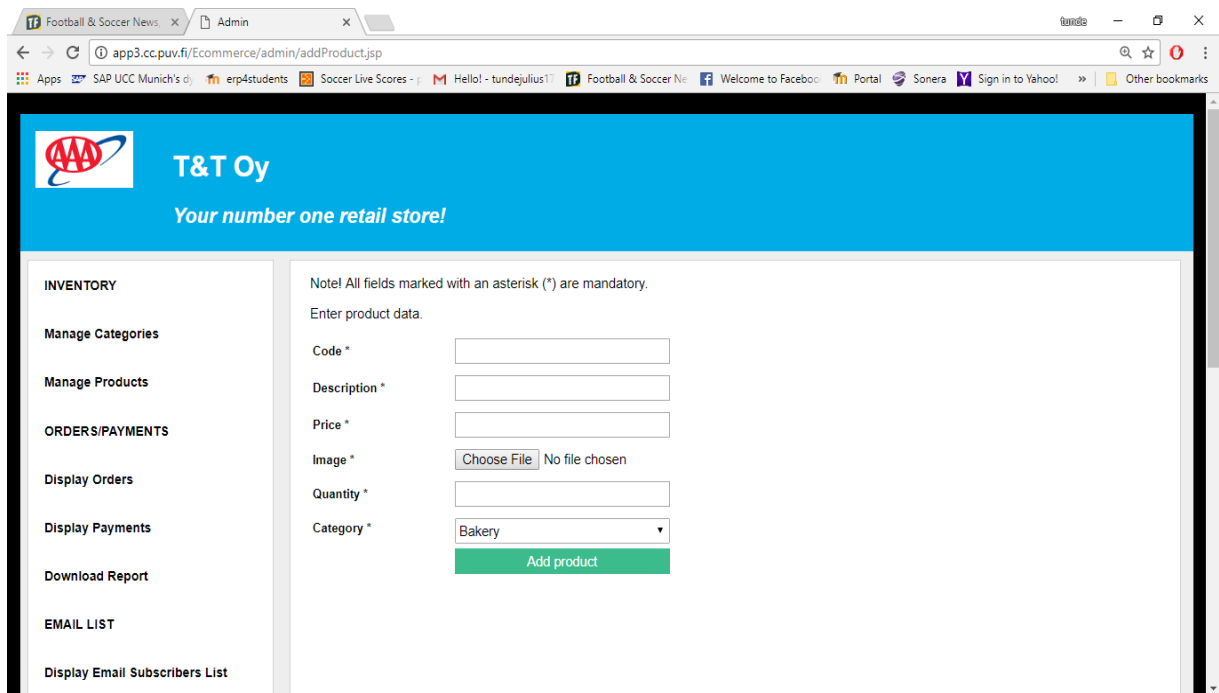
Code	Description	Price	Image	Quantity	Last update	Category	Action
m200	Skim milk 1.5L	€ 1.80		16	01.06.2017 07:55	Diary	Edit Delete
p100	Pizza Americana	€ 4.50		7	01.06.2017 07:56	Frozen Foods	Edit Delete
c100	Corn	€ 2.50		19	01.06.2017 08:01	Cereals	Edit Delete
c200	Cheese 250g	€ 1.80		14	01.06.2017 08:02	Diary	Edit Delete

Figure 19. Products page.

4.2.11 Add Product Page

The page for adding products is shown in Figure 20. The form on this page is displayed when the link for adding products on the products page is clicked. Every input on this form is checked for null and empty values. The edit product form is basically the same as the add product form, except that it contains editable data in its fields.

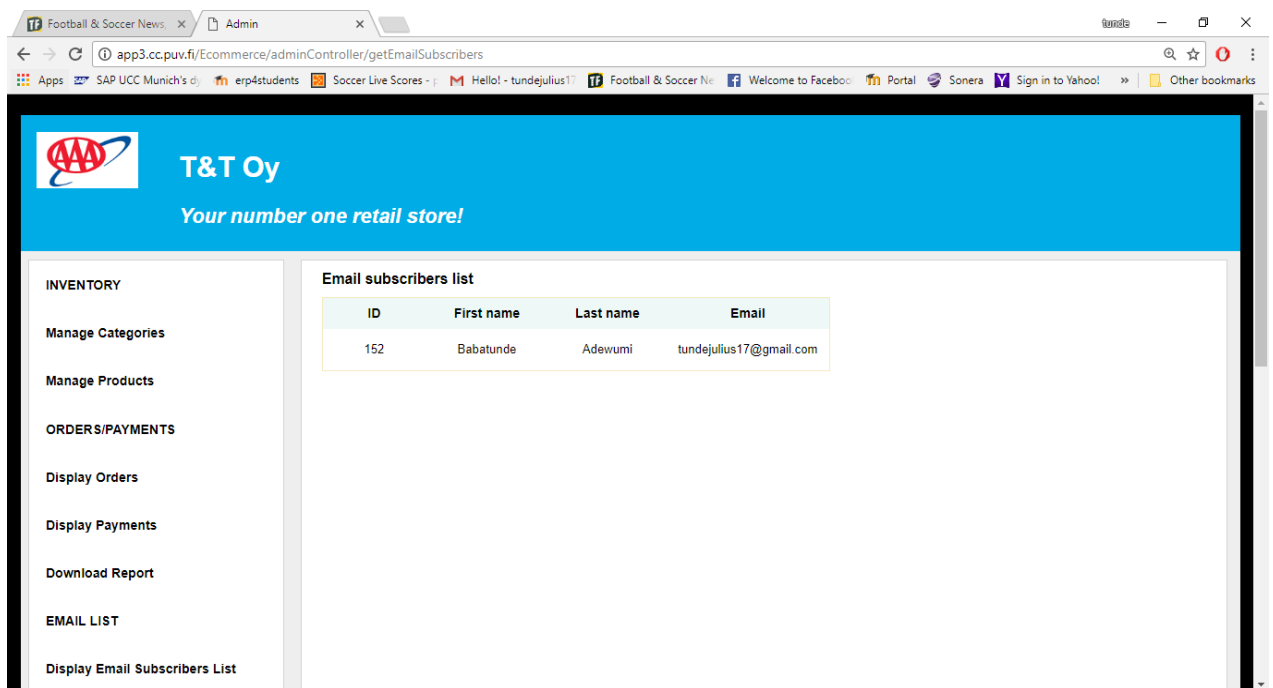


The screenshot shows a web browser window with the URL `app3.cc.puv.fi/Ecommerce/admin/addProduct.jsp`. The page header features the T&T Oy logo and the slogan "Your number one retail store!". A left-hand navigation menu lists various administrative functions under categories like INVENTORY, ORDERS/PAYMENTS, and EMAIL LIST. The main content area is titled "Add product" and contains a form with the following fields: Code *, Description *, Price *, Image * (with a "Choose File" button and "No file chosen" text), Quantity *, and Category * (with a dropdown menu currently set to "Bakery"). A green "Add product" button is located at the bottom of the form. A note at the top of the form states: "Note! All fields marked with an asterisk (*) are mandatory." The browser's address bar and bookmark bar are also visible at the top of the window.

Figure 20. Add product page.

4.2.12 Email List Page

The email list page is shown in Figure 21. The page is accessed by clicking the display email subscribers list link on the admin menu page. It displays a list of all customers and visitors to the web store who have subscribed to the email list for special announcements and offers.



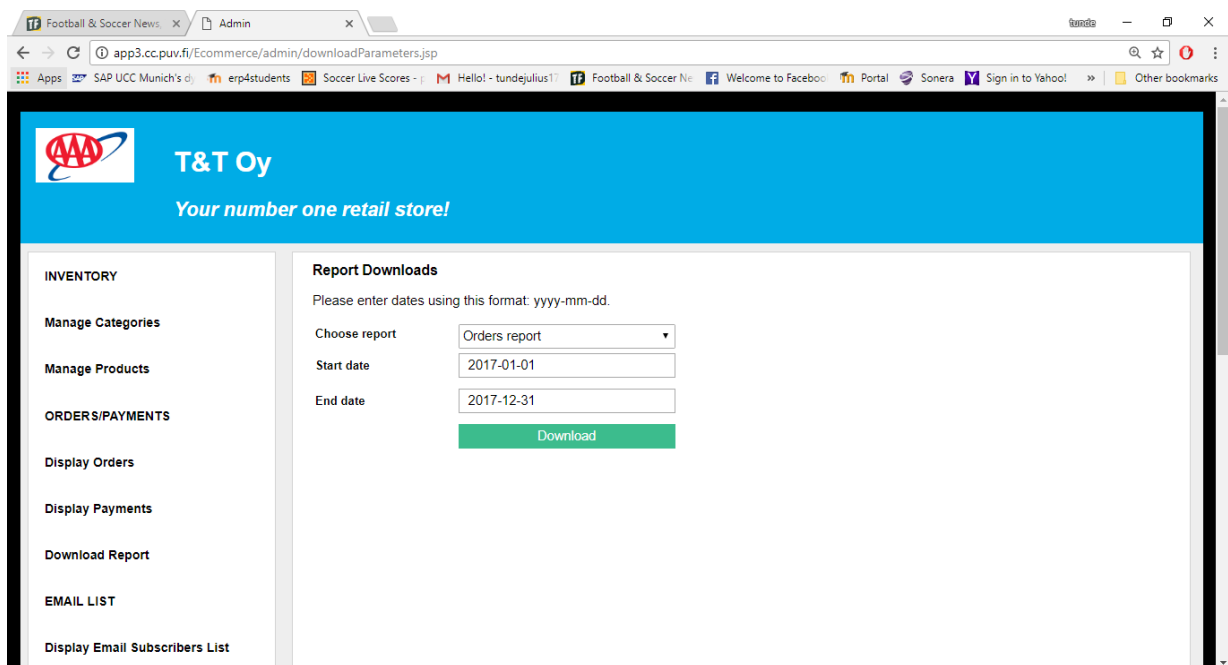
The screenshot shows a web browser window with the URL `app3.cc.puv.fi/Ecommerce/adminController/getEmailSubscribers`. The page header features the T&T Oy logo and the tagline "Your number one retail store!". A left-hand navigation menu includes sections for INVENTORY, ORDERS/PAYMENTS, and EMAIL LIST. The main content area displays a table titled "Email subscribers list" with the following data:

ID	First name	Last name	Email
152	Babatunde	Adeyemi	tundejulius17@gmail.com

Figure 21. Email list page.

4.2.13 Report Download Form

Figure 22 shows the report download form, which is accessed by clicking the download report link on the admin menu page. The user chooses the report to be downloaded, enters start and end dates in the given format, and clicks the download button to download the chosen report in excel format. An appropriate error message is displayed if a wrong date format is entered.



The screenshot shows a web browser window with the URL `app3.cc.puv.fi/Ecommerce/admin/downloadParameters.jsp`. The page header features the T&T Oy logo and the slogan "Your number one retail store!". A left-hand navigation menu is visible, with categories including INVENTORY, ORDERS/PAYMENTS, and EMAIL LIST. The main content area is titled "Report Downloads" and contains a form with the following fields:

- A note: "Please enter dates using this format: yyyy-mm-dd."
- "Choose report": A dropdown menu with "Orders report" selected.
- "Start date": A text input field containing "2017-01-01".
- "End date": A text input field containing "2017-12-31".
- A green "Download" button.

Figure 22. Report download form.

4.2.14 Orders List Page

The orders list page is shown in Figure 23. The page is displayed when an admin user fills and successfully submits the orders report form. The page displays a list of all orders requested in accordance with the start and end dates entered on the orders report form. Also, it is possible to view details of an order by clicking the view details button. Lastly, the status of the order can be updated on the order details page.

The screenshot shows the 'Orders list' page in the T&T Oy admin interface. The page has a blue header with the T&T Oy logo and the tagline 'Your number one retail store!'. On the left, there is a sidebar with navigation options under three main categories: INVENTORY, ORDERS/PAYMENTS, and EMAIL LIST. The main content area displays a table of orders with the following data:

ID	Order Date	Amount	Customer ID	Order Status	Details
253	11.08.2017 23:48	€ 1.80	107	open	view details
206	03.07.2017 17:46	€ 8.49	52	open	view details
202	03.07.2017 17:22	€ 6.10	52	closed	view details
108	12.06.2017 23:58	€ 4.50	107	open	view details
102	12.06.2017 23:49	€ 6.78	52	open	view details
57	01.06.2017 13:35	€ 7.68	52	closed	view details
53	01.06.2017 08:19	€ 5.58	52	open	view details

Figure 23. Orders list page.

4.2.15 Payments List Page

The diagram in Figure 24 is the payments list page of this online retail store application. The page is accessed by filling and submitting the payment report form correctly. The page displays a list of all payment transactions in accordance with the start and end dates entered.

The screenshot shows a web browser window with the URL `app3.cc.puv.fi/Ecommerce/adminController/getPayments`. The page header for T&T Oy features the company logo and the slogan "Your number one retail store!". A left-hand navigation menu includes sections for INVENTORY, ORDERS/PAYMENTS, and EMAIL LIST. The main content area displays a "Payments list" table with the following data:

ID	Amount	Payment Date	Payment Status	Transaction ID	Customer ID	Order ID
255	€ 1.80	11.08.2017 23:48	Completed	9B870023YD419811E	107	253
209	€ 8.49	03.07.2017 17:46	Completed	8TA47550JB634135S	52	206
205	€ 6.10	03.07.2017 17:22	Completed	1BC02669W2545302K	52	202
110	€ 4.50	12.06.2017 23:58	Completed	3VA76442NT986151L	107	108
106	€ 6.78	12.06.2017 23:49	Completed	3FL15144PD689435C	52	102
60	€ 7.68	01.06.2017 13:35	Completed	22M640977S362245S	52	57
56	€ 5.58	01.06.2017 08:19	Completed	7HS6416662966853R	52	53

Figure 24. Payments list page.

5 IMPLEMENTATION

Implementation in software development is the process of realizing an application's requirements and design. It mainly involves mapping the design into coding in order to achieve the specifications stated for the application. This section will describe the key implementation processes and some code snippets of this e-commerce web application.

This application was implemented as a JPA application, thus, the database tables used by the application were automatically generated by JPA. As earlier shown in the component diagram of Figure 7 of chapter 3, the application was structured according to the MVC pattern. This pattern aims to separate the user interface logic from the business logic. Also, this pattern helps to create well defined and organized web applications with efficient code reuse and multiple views.

5.1 Home Page

The home page of the application displays all the available products in the store. In Code Snippet 1, the `getProducts` helper method is called to retrieve all product data stored in the database. Also, shown in the code snippet is the implementation of the `getProducts` method, which calls the `getEntityManagerFactory` static method in order to establish a connection to the database. Then a Java Persistence Query Language (JPQL) is written to query the database to return a list of all available products in the database.

```

List<Product> products = DBProduct.getProducts();
session.setAttribute("custProducts", products);

// This retrieves product objects from the database.
public static List<Product> getProducts() {
    EntityManager em = DBUtil.getEntityManagerFactory()
        .createEntityManager();
    String query = "SELECT p FROM Product p ORDER BY p.lastUpdate";
    TypedQuery<Product> q = em.createQuery(query, Product.class);
    List<Product> products;

    try {
        products = q.getResultList();
        if (products == null || products.isEmpty())
            products = null;
    } finally {
        em.close();
    }
    return products;
}

```

Code Snippet 1. Retrieving available products from the database.

5.2 Subscription to Email List

Code Snippet 2 shows the implementation of the `subscribeToEmail` method. This method retrieves the submitted input data on the subscribe to email form. It then checks that no input parameter is null or empty by calling the `checkParam` method. It also calls the `checkEmail` method to ensure that the user enters a valid email address. In order to avoid duplicate email address in the database, the `emailExists` method is called to check if the entered email address already exists in the database. The `addEmailSubscriber` method is called to add the entered data into the database if all the input parameters are entered correctly. Lastly, the `sendEmail` method is called to send an email confirmation to the provided email address and the user is redirected to the confirmation page.

```

// Insert EmailSubscriber obj into the database.
private String subscribeToEmail(HttpServletRequest request,
    HttpServletResponse response) throws IOException {
    String firstName = request.getParameter("firstName").trim();
    String lastName = request.getParameter("lastName").trim();
    String email = request.getParameter("email").trim();
    String message = "";
    String url = "";

    EmailSubscriber emailSubscriber = new EmailSubscriber();
    emailSubscriber.setFirstName(firstName);
    emailSubscriber.setLastName(lastName);
    emailSubscriber.setEmail(email);
    request.setAttribute("emailSubscriber", emailSubscriber);

    if (!ValidatorUtil.checkParam(firstName)
        || !ValidatorUtil.checkParam(lastName)
        || !ValidatorUtil.checkParam(email)) {
        message = "No field can be empty";
        request.setAttribute("message", message);
        url = "/emailSubscription/index.jsp";
    } else if (!ValidatorUtil.checkEmail(email)) {
        request.setAttribute("emailError", "email not in the right format");
        url = "/emailSubscription/index.jsp";
    } else { // No two email address can be the same in the database.
        if (DBEmail.emailExists(email)) {
            String emailMessage = "This email address already exist. "
                + "<br> Kindly provide another email address.";
            request.setAttribute("emailMessage", emailMessage);
            url = "/emailSubscription/index.jsp";
        } else {
            DBEmail.addEmailSubscriber(emailSubscriber);
            // Prepare the confirmation email message
            String recipient = email;
            String sender =
                this.getServletContext().getInitParameter("custServEmailAddress");
            String subject = "Email Subscription Confirmation";
            String body = "<p>Hello "
                + emailSubscriber.getFirstName()
                + ",</p><n\n"
                + "<p>Thank you for subscribing to our email list."
                + " We will send you news about our new prod-
ucts and special offers.</p><n\n"
                + "<p>Sincerely, </p><n" + "<p>T&T Team</p><n\n";

            boolean bodyIsHTML = true;

            try {
                // Send email message.
                EmailUtil.sendEmail(recipient, sender, subject, body,
                    bodyIsHTML);
            } catch (MessagingException e) {
                e.printStackTrace();
            }

            url = "/emailSubscription/confirmation.jsp";
        }
    }

    return url;
}

```

Code Snippet 2. Method for subscribing to the email list.

5.3 Add Product to the Shopping Cart

In Code Snippet 3, the addCartItem method is called when a user clicks the add to cart button on the home page of the application. In this method, a new instance of the Cart

class is created if the retrieved Cart object does not exist in the Session object. Then the addCartItem method of the Cart class is called to add the product to the shopping cart after all the necessary operations and checks. Lastly, the Cart object is saved into the Session object and the user is redirected back to the home page of the application.

```
// Add product to the shopping cart.
private String addCartItem(HttpServletRequest request,
    HttpServletResponse response) throws IOException {
    HttpSession session = request.getSession();
    Cart cart = (Cart) session.getAttribute("cart");
    if (cart == null) {
        cart = new Cart();
    }
    String code = request.getParameter("code");

    if (!code.isEmpty()) {
        Product product = DBProduct.getProductByCode(code);
        cart.addCartItem(product);
    }
    session.setAttribute("cart", cart);
    return "/index.jsp";
}
}
```

Code Snippet 3. Method for adding product to the shopping cart.

5.4 Update Item in the Shopping Cart

The updateCartItem method in Code Snippet 4 is responsible for updating the quantities of items in the shopping cart. The code and quantity parameters are first retrieved from the Request object, and the Cart object retrieved from the Session object. After all the necessary operations and checks, the updateCartItem method of the Cart class is called to perform the necessary updates in the shopping cart, including item quantity.

```

// Update cart item quantity in the shopping cart.
private String updateCartItem(HttpServletRequest request,
    HttpServletResponse response) throws IOException {
    String code = request.getParameter("code");
    String qtyString = request.getParameter("quantity");
    HttpSession session = request.getSession();
    Cart cart = (Cart) session.getAttribute("cart");

    int quantity;
    Product product = DBProduct.getProductByCode(code);

    try {
        quantity = Integer.parseInt(qtyString);
        if (quantity < 0 || quantity == 0) {
            quantity = 1;
        }

        if (quantity > product.getQuantity()) {
            quantity = 1;
            request.setAttribute("insufficientQuantity",
                "insufficient quantity");
        }
    } catch (NumberFormatException ex) {
        quantity = 1;
    }

    if (quantity < product.getQuantity()
        || quantity == product.getQuantity()) {
        cart.updateCartItem(product, quantity);
    }
    return "/cart/cart.jsp";
}
}

```

Code Snippet 4. Method for updating item quantity in the shopping cart.

5.5 Remove Item from the Shopping Cart

The `removeCartItem` method in Code Snippet 5 is responsible for removing an item from the shopping cart. After performing all the necessary operations and checks it calls the `removeCartItem` method of the `Cart` class to remove the item from the cart.

```

// Remove cart item from the shopping cart
private String removeCartItem(HttpServletRequest request,
    HttpServletResponse response) throws IOException {
    String code = request.getParameter("code");
    HttpSession session = request.getSession();
    Cart cart = (Cart) session.getAttribute("cart");
    Product product = DBProduct.getProductByCode(code);
    if (product != null && cart != null) {
        cart.removeCartItem(product);
    }
    return "/cart/cart.jsp";
}
}

```

Code Snippet 5. Method for removing item from the shopping cart.

5.6 Remove all Items from the Shopping Cart

Code Snippet 6 shows the implementation of the `clearCart` method. It is responsible for removing all items from the shopping cart. It calls the `clearCart` method of the `Cart` class to delete all items from the cart after all the necessary operations and checks.

```

// Remove all items in the cart.
private String clearCart(HttpServletRequest request,
    HttpServletResponse response) throws IOException {
    HttpSession session = request.getSession();
    Cart cart = (Cart) session.getAttribute("cart");
    if (cart != null) {
        cart.clearCart();
    }
    return "/cart/cart.jsp";
}

```

Code Snippet 6. Method for removing all items from the shopping cart.

5.7 Payment with PayPal Express Checkout

This e-commerce web store uses PayPal Express Checkout as its payment solution. With this payment solution, the buyer does not enter any information as the buyer's details can be obtained from PayPal. For the purpose of testing, this application uses a virtual testing environment called PayPal Sandbox, which mimics most of the features of the PayPal production environment. In order to use the Sandbox server, a seller (merchant) and a buyer test accounts were created. Three PayPal Name-Value Pair (NVP) API operations were created to implement the Express Checkout payment solution. The following subsections describe these API operations.

5.7.1 SetExpressCheckout API Call

Code Snippet 7 shows the method used to implement the SetExpressCheckout API call. This API call is used to initiate the payment transaction. When a buyer clicks the PayPal check out button on the shopping cart page of the application, the callShortcutExpressCheckout method is called. The checkoutDetails parameter of the method contains the request-specific fields retrieved from the clicked button. The returnUrl parameter is the page on the e-commerce web store that PayPal redirects to after a successful payment authorization, and the cancelURL is the page on the e-commerce web store that PayPal redirects to if a buyer cancels the payment. From the method implementation, the fields in the checkoutDetails parameter are used to construct an NVP string, and then an HTTP POST request is sent to PayPal API server by calling the httpCall method, which makes the SetExpressCheckout API call. Part of the response to the API call is an acknowledgment status, which indicates a success or failure with or without warning messages. Also, a token is returned as part of the

response. The token is a unique string used for identifying each transaction and in making other API calls.

```
//Method for implementing the SetExpressCheckout API call.
public Map<String, String> callShortcutExpressCheckout(
    Map<String, String> checkoutDetails, String returnUrl,
    String cancelURL) {

    // Construct the parameter string that describes the
    // SetExpressCheckout API call
    StringBuilder nvpstr = new StringBuilder("");

    //Append the line item parameters to the nvpstr variable.
    for (String key : checkoutDetails.keySet()) {
        if (key.startsWith("L")) {
            nvpstr.append("&" + key +
"=").append(checkoutDetails.get(key));
        }
        if (isSet(checkoutDetails.get("PAYMENTREQUEST_0_AMT"))) {
            nvpstr.append("&PAYMENTREQUEST_0_AMT=").append(
                checkoutDetails.get("PAYMENTREQUEST_0_AMT"));
        }
        if (isSet(checkoutDetails.get("paymentType"))) {
            nvpstr.append("&PAYMENTREQUEST_0_PAYMENTACTION=").append(
                checkoutDetails.get("paymentType"));
        }
        if (isSet(returnURL))
            nvpstr.append("&RETURNURL=").append(returnURL);
        if (isSet(cancelURL))
            nvpstr.append("&CANCELURL=").append(cancelURL);
        if (isSet(checkoutDetails.get("currencyCodeType"))) {
            nvpstr.append("&PAYMENTREQUEST_0_CURRENCYCODE=").append(
                checkoutDetails.get("currencyCodeType"));
        }
        if (isSet(checkoutDetails.get("PAYMENTREQUEST_0_ITEMAMT"))) {
            nvpstr.append("&PAYMENTREQUEST_0_ITEMAMT=").append(
                checkoutDetails.get("PAYMENTREQUEST_0_ITEMAMT"));
        }
        if(isSet(checkoutDetails.get("LOGOIMG")))
            nvpstr.append(" &LOGOIMG="+ checkoutDetails.get("LOGOIMG"));
    //Make the API call.
    return httpCall("SetExpressCheckout", nvpstr.toString());
}
}
```

Code Snippet 7. Method for SetExpressCheckout API call.

5.7.2 GetExpressCheckoutDetails API Call

In Code Snippet 8, the getShippingDetails implements the GetExpressCheckoutDetails API call. In the method, an NVP string is constructed by using the token returned from the SetExpressCheckout API call. Then, a call is made to the GetExpressCheckoutDetails API by calling the httpCall method. The response of this API call contains the buyer's details and an acknowledgment status, which indicates a success or a failure with or without warning messages.

```
//Method for implementing GetExpressCheckoutDetails API call.
public Map<String, String> getShippingDetails(String token) {
    //Construct the nvp string
    String nvpstr = "&TOKEN=" + token;
    //Make the API call.
    return httpCall("GetExpressCheckoutDetails", nvpstr);
}
```

Code Snippet 8. Method for GetExpressCheckoutDetails API call.

5.7.3 DoExpressCheckoutPayment API Call

As shown in Code Snippet 9, the `confirmPayment` method implements the `DoExpressCheckoutPayment` API call, which completes the payment transaction. In this method, the NVP string is constructed with the necessary request-specific fields in the `checkoutDetails` parameter of the method. Then the API call is made by calling the `httpCall` method. The response from this API contains a success or a failure with or without warning messages, and other transaction specific information like transaction id.

```

//Method implementation for DoExpressCheckoutPayment API call.
public HashMap<String, String> confirmPayment(
    Map<String, String> checkoutDetails, String serverName) {
    // Construct the NVP string
    String finalPaymentAmount = encode(checkoutDetails
        .get("PAYMENTREQUEST_0_AMT"));
    StringBuilder nvpstr = new StringBuilder("");
    // mandatory parameters in DoExpressCheckoutPayment call
    if (isSet(checkoutDetails.get("TOKEN")))
        nvpstr.append("&TOKEN=").append(
            encode(checkoutDetails.get("TOKEN")));
    if (isSet(checkoutDetails.get("payer_id")))
        nvpstr.append("&PAYERID=").append(
            encode(checkoutDetails.get("payer_id")));
    nvpstr.append("&PAYMENTREQUEST_0_SELLERPAYPALACCOUNTID=").append(
        this.getSellerEmail());
    if (isSet(checkoutDetails.get("paymentType")))
        nvpstr.append("&PAYMENTREQUEST_0_PAYMENTACTION=").append(
            encode(checkoutDetails.get("paymentType")));
    if (isSet(serverName))
        nvpstr.append("&IPADDRESS=").append(encode(serverName));
    nvpstr.append("&PAYMENTREQUEST_0_AMT=").append(finalPaymentAmount);
    // Check for additional parameters that can be passed in
    // DoExpressCheckoutPayment API call
    if (isSet(checkoutDetails.get("currencyCodeType")))
        nvpstr.append("&PAYMENTREQUEST_0_CURRENCYCODE=").append(
            encode(checkoutDetails.get("currencyCodeType").toString()));
    if (isSet(checkoutDetails.get("PAYMENTREQUEST_0_ITEMAMT")))
        nvpstr.append("&PAYMENTREQUEST_0_ITEMAMT=").append(
            encode(checkoutDetails.get("PAYMENTREQUEST_0_ITEMAMT")
                .toString()));
    if (isSet(checkoutDetails.get("PAYMENTREQUEST_0_TAXAMT")))
        nvpstr.append("&PAYMENTREQUEST_0_TAXAMT=").append(
            encode(checkoutDetails.get("PAYMENTREQUEST_0_TAXAMT")
                .toString()));
    if (isSet(checkoutDetails.get("shippingAmt")))
        nvpstr.append("&PAYMENTREQUEST_0_SHIPPINGAMT=").append(
            encode(checkoutDetails.get("PAYMENTREQUEST_0_SHIPPINGAMT")
                .toString()));
    if (isSet(checkoutDetails.get("handlingAmt")))
        nvpstr.append("&PAYMENTREQUEST_0_HANDLINGAMT=").append(
            encode(checkoutDetails.get("PAYMENTREQUEST_0_HANDLINGAMT")
                .toString()));
    if (isSet(checkoutDetails.get("shippingDiscAmt")))
        nvpstr.append("&PAYMENTREQUEST_0_SHIPDISCAMT=").append(
            encode(checkoutDetails.get("PAYMENTREQUEST_0_SHIPDISCAMT")
                .toString()));
    if (isSet(checkoutDetails.get("insuranceAmt")))
        nvpstr.append("&PAYMENTREQUEST_0_INSURANCEAMT=").append(
            encode(checkoutDetails.get("PAYMENTREQUEST_0_INSURANCEAMT")
                .toString()));
    //Make the API call.
    return httpCall("DoExpressCheckoutPayment", nvpstr.toString());
}

```

Code Snippet 9. Method for DoExpressCheckoutPayment API call.

5.8 Sending Email

Code Snippet 10 shows the method implementation for sending an email confirmation to users after completing an order or subscribing to an email list. The method uses methods and classes from the JavaMail API to automatically send email notifications.

```
//Method for sending email messages.
public static void sendEmail(String recipient, String sender,
    String subject, String body, boolean bodyIsHTML)
    throws MessagingException {
    // Get a mail session
    Properties properties = new Properties();
    properties.put("mail.transport.protocol", "smtp");
    properties.put("mail.smtps.host", "localhost");
    properties.put("mail.smtps.port", 25);
    Session session = Session.getDefaultInstance(properties);
    session.setDebug(true);
    // Create a message
    Message message = new MimeMessage(session);
    message.setSubject(subject);
    if (bodyIsHTML) {
        message.setContent(body, "text/html");
    } else {
        message.setText(body);
    }
    // Here we set the addresses
    Address fromAddress = new InternetAddress(sender);
    Address toAddress = new InternetAddress(recipient);
    message.setRecipient(Message.RecipientType.TO, toAddress);
    message.setFrom(fromAddress);
    Transport.send(message);
}
```

Code Snippet 10. Method for sending an email.

5.9 Admin Login

The admin login process was implemented by the form-based authentication method provided by Apache Tomcat container. Code Snippet 11 shows the authentication configuration. As can be seen from the Code Snippet, all web resources in the admin directory of the application are restricted and a user with manager role is being granted authorization to access the restricted resources. Lastly, customized login and error pages are defined for the authentication.

```

<security-role>
  <description>owner</description>
  <role-name>manager</role-name>
</security-role>
<security-constraint>
  <!-- Restrict access to the URLs in the admin directory -->
  <web-resource-collection>
    <web-resource-name>Admin</web-resource-name>
    <url-pattern>/admin/*</url-pattern>
  </web-resource-collection>
  <!-- Authorize the service and programmer roles -->
  <auth-constraint>
    <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>
<!-- Use form-based authentication to provide access -->
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>/login.jsp</form-login-page>
    <form-error-page>/login_error.jsp</form-error-page>
  </form-login-config>
</login-config>

```

Code Snippet 11. Form-based authentication configuration for admin login.

5.10 Add Category

The method for adding category data into the database is shown in Code Snippet 12. The method responds to the request sent when an authorized user clicks the add category button on the add category form. The validity of the input data is checked by calling the `checkParm` method. An appropriate error message is sent whenever something goes wrong otherwise, the category data is inserted into the database by calling the `addCategory` method.


```

// Insert Category obj into the database.
private String addCategory(HttpServletRequest request,
    HttpServletResponse response) throws IOException {
    String name = (request.getParameter("name")).trim();
    Date date = new Date();
    Category addCategory = new Category();
    addCategory.setName(name);
    addCategory.setLastUpdate(date);
    request.setAttribute("category", addCategory);
    String message = "";
    String url = "";
    if (!ValidatorUtil.checkParam(name)) {
        message = "Field cannot be empty.";
        request.setAttribute("message", message);
        url = "/admin/addCategory.jsp";
    } else {
        // No two categories can have the same name in the database.
        if (DBCategory.categoryNameExists(name)) {
            message = "This name already exists for another category. "
                + "<br> Kindly provide another name.";
            request.setAttribute("message", message);
            url = "/admin/addCategory.jsp";
        } else {
            DBCategory.addCategory(addCategory);
            url = getCategories(request, response);
        }
    }
    return url;
}

```

Code Snippet 12. Method for adding category data into the database.

5.11 Add Product

Code Snippet 13 shows the method for adding product data into the database. The method responds to the request sent when an admin user clicks the add product button on the add product form. The method checks the validity of the entered data by calling the checkParam method. An appropriate error message is sent whenever something goes wrong otherwise, the product data is added into the database by calling the addProduct method.

```

// Insert Product obj into the database.
private String addProduct(HttpServletRequest request,
    HttpServletResponse response) throws IOException {
    String code = request.getParameter("code").trim();
    String description = request.getParameter("description").trim();
    double price = Double.parseDouble(request.getParameter("price"));
    byte[] image = uploadFile(request, response);
    int quantity = Integer.parseInt(request.getParameter("quantity"));
    String categoryName = request.getParameter("categoryName");
    Category category = DBCategory.getCategoryByName(categoryName);
    Product product = new Product();
    product.setCode(code);
    product.setDescription(description);
    product.setPrice(price);
    product.setImage(image);
    product.setQuantity(quantity);
    product.setCategory(category);
    product.setLastUpdate(new Date());
    request.setAttribute("product", product);
    String message = "";
    String url = "";
    if (!ValidatorUtil.checkParam(code)
        || !ValidatorUtil.checkParam(description)) {
        message = "No field can be empty";
        request.setAttribute("message", message);
        url = "/admin/addProduct.jsp";
    } else {
        // No two product can have the same code.
        if (DBProduct.productCodeExists(code)) {
            message = "This code already exist for another product. "
                + "<br> Kindly provide another code.";
            request.setAttribute("message", message);
            url = "/admin/addProduct.jsp";
        } // A product must belong to only one existing category in the
        // database.
        else if (category == null) {
            message = "The selected category no longer exist. "
                + "<br> Kindly select another category.";
            request.setAttribute("message", message);
            url = "/admin/addProduct.jsp";
        } else {
            DBProduct.addProduct(product);
            url = getProducts(request, response);
        }
    }
    return url;
}

```

Code Snippet 13. Method for adding product data into the database.

5.12 Download Reports in Excel Format

One of the methods for downloading reports in Microsoft excel format is shown in Code Snippet 14. The method uses classes and methods from the Apache POI API to create the email subscribers report in excel format.

```
// This creates an excel workbook of all email subscribers.
public static Workbook getEmailSubscribersReport() {
    List<EmailSubscriber> emailSubscribers = DBEmail.getEmailSubscribers();
    // create a workbook, sheet and its title
    XSSFWorkbook workbook = new XSSFWorkbook();
    XSSFSheet sheet = workbook.createSheet("Email Subscribers Report");
    XSSFRow row = sheet.createRow(0);
    row.createCell(0).setCellValue("The Email Subscribers Report");
    // create row headers
    row = sheet.createRow(2);
    row.createCell(0).setCellValue("LastName");
    row.createCell(1).setCellValue("FirstName");
    row.createCell(2).setCellValue("Email");
    // create data rows
    int i = 3;
    for (EmailSubscriber e : emailSubscribers) {
        row = sheet.createRow(i);
        row.createCell(0).setCellValue(e.getLastName());
        row.createCell(1).setCellValue(e.getFirstName());
        row.createCell(2).setCellValue(e.getEmail());
        i++;
    }
    return workbook;
}
```

Code Snippet 14. Method for downloading reports in excel format.

6 TESTING

Software testing is carried out on a software application mainly to detect software bugs or missing requirements of the application. It involves the process of investigating and evaluating a software product or application in order to make sure that its business and technical requirements are fulfilled. For the purpose of testing, this application was deployed and run on Apache Tomcat servlet container and accessed on a web browser. For simplicity, the testing template in Table 2 was used for the testing.

Table 2. Software testing template and results.

S/N	Test Description	Steps	Expected System Response	Status (Pass/Fail)
1.	Access home page of the application to view available products.	Enter the home page URL of the application on a web browser.	The home page of the application is displayed with available products.	Pass.
2.	View available products by category.	Click any of the category links on the home page.	The products for the clicked category are displayed.	Pass.
3.	Check for empty data input on the email subscription form.	Click the subscribe to email list on home page. Enter the required data on the form with one or more empty inputs and click the subscribe now button.	The application displays a message prompt telling the user to enter data in the empty field(s).	Pass.
4.	Check for an existing email address in the email subscription list.	Enter an existing email address together with the other inputs data. Click the subscribe now button.	The application displays an error message indicating that the provided email address already exists in the database.	Pass.
5.	Check for correct data inputs on the email subscription form.	Enter correct data inputs on the email subscription form and click subscribe now.	The application displays a confirmation message with the entered data. Also, an email confirmation is sent to the provided email address.	Pass.
6.	Add product to the shopping cart.	Click the add to cart button on the displayed product on the home page.	The application remains on the home page with the number of items in the cart updated and shown on the top of the home	Pass.

			page.	
7.	View the shopping cart.	Click the view cart link on the home page of the application.	The shopping cart of the application is displayed with its items.	Pass.
8.	Update an item quantity in the shopping cart.	Edit the quantity of an item in the cart and click the update button.	The item quantity, its price and the total price of all items in the cart are updated accordingly.	Pass.
9.	Remove an item from the shopping cart.	Click the remove button of the item to be removed.	The item is removed from the cart and the total price of all items in the cart is updated accordingly.	Pass.
10.	Clear all items from the shopping cart.	Click the clear cart button.	The cart is empty.	Pass.
11.	Continue shopping from the shopping cart page.	Click the continue shopping link.	The user is redirected to the home page of the application to continue shopping.	Pass.
12.	Check out with PayPal.	Click the PayPal check out button on the shopping cart page.	The application redirects to the login page of PayPal.	Pass.
13.	Confirm order after a successful payment authorization on PayPal.	Click the confirm order button on the order review page of the application.	The order confirmation page is displayed with details of the transaction. Also, an email order confirmation is sent to the email address of the buyer.	Pass.
14.	Access the admin page of the application to carry administrative tasks.	Enter the admin page URL of the application on a web browser.	The admin login page of the application is displayed.	Pass.
15.	Check for empty login data on the admin login form.	Click the login button with one empty field.	The application displays a prompt message telling the user to enter data in the empty field.	Pass.
16.	Check for incorrect login credentials on the admin	Enter the incorrect login details. Click the login button.	The login form with an error	Pass.

	login form.		message is displayed.	
17.	Check for correct login credentials on the admin login form.	Enter the correct login details. Click the login button.	The admin menu page is displayed.	Pass.
18.	Access the admin menu items on the admin menu page.	Click any of the menu links on the page.	The appropriate page is displayed.	Pass.
19.	Check for empty data inputs on forms accessible by the admin.	Enter the required data on the form with one or more empty inputs and submit the form.	The application displays a message prompt telling the user to enter data in the empty field(s).	Pass.
20.	Add/edit category/product.	Enter the required data on the appropriate form and submit.	An appropriate page is displayed to show the added/edited category or product.	Pass.
21.	Delete category/product.	Click the appropriate delete button.	The application displays an appropriate page to confirm the deleted category or product.	Pass.
22.	Download report in excel format.	Access the appropriate link to download the report.	The application presents the report in an excel document.	Pass.

7 CONCLUSION

The main objective of this thesis work was to develop an e-commerce Java web application for a small retail store where the store owner manages products, customers, and orders, while the customers make orders and pay for products. The application was developed with the above-mentioned features.

One of the biggest challenges faced during the development of this software project was how to implement JPA for the application. A lot of time and effort were invested in learning and implementing JPA for this e-commerce application. Another challenge faced was how to integrate PayPal Express Checkout NVP API operations for the project. PayPal has a poor API integration documentation, especially API integration for the Java programming language. This actually affected the flow of the application development process as much time was used to learn and understand the NVP API integration for Java. With these challenges and others not mentioned here, a lot of new experience has been gained during the development process of this application.

Although all the requirements set out for the e-commerce web application have been met, there are still areas to improve on. A mobile version can be developed for the application so that users can have a better access to the application. Also, other online payment methods like credit/debit card and bank payment methods can be implemented for the application.

REFERENCES

- /1/ Miva 2011. The History Of Ecommerce: How Did It All Begin? Accessed 15.07.2017. <http://www.miva.com/blog/the-history-of-ecommerce-how-did-it-all-begin/>
- /2/ Arline, K. 2015. What is E-commerce? Accessed 12.07.2017. <http://www.businessnewsdaily.com/4872-what-is-e-commerce.html>
- /3/ Coleman, S. 2017. What is Electronic Commerce? – Definition, Types, Advantages & Disadvantages. Accessed 12.07.2017. <http://study.com/academy/lesson/what-is-electronic-commerce-definition-types-advantages-disadvantages.html>
- /4/ Henson, J. 2012. A Brief History of the eCommerce World. Accessed 15.07.2017. <https://www.unleashed-technologies.com/blog/2012/08/14/brief-history-ecommerce-world>
- /5/ Tutorialspoint 2017. Java Tutorial. Accessed 17.07.2017. <https://www.tutorialspoint.com/java/index.htm>
- /6/ Stackoverflow 2017. What is the difference between JDK and JRE? Accessed 17.07.2017. <https://stackoverflow.com/questions/1906445/what-is-the-difference-between-jdk-and-jre>
- /7/ Evans, I. 2012. Differences between Java EE and Java SE - Your First Cup: An Introduction to the Java EE Platform. Accessed 17.07.2017. <http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html>
- /8/ Tutorialspoint 2017. HTML Tutorial. Accessed 17.07.2017 <https://www.tutorialspoint.com/html/>
- /9/ W3Schools 2017. CSS Tutorial. Accessed 17.07.2017 <https://www.w3schools.com/css/>
- 10/ Oracle. What is a Servlet – The Java EE 5 Tutorial. Accessed 18.07.2017. <http://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html>
- /11/ Oracle. JavaServer Pages Technology. Accessed 19.07.2017. <http://www.oracle.com/technetwork/java/javaee/jsp/index.html>
- /12/ Murach, J. & Urban, M. 2014. Murach's Java Servlets and JSP. 3rd ed. Fresno California. Mike Murach & Associates, Inc.
- /13/ Tutorialspoint 2017. Eclipse Tutorial. Accessed 21.07.2017 <https://www.tutorialspoint.com/eclipse/>
- /14/ The Apache Software Foundation 2017. Apache Tomcat. Accessed 22.07.2017. <https://tomcat.apache.org/>
- /15/ The Apache Software Foundation 2017. Apache POI – the Java API for Microsoft Documents. Accessed 23.07.2017. <https://poi.apache.org/>

- /16/ PayPal 2017. About Us. Accessed 23.07.2017.
<https://www.sandbox.paypal.com/fi/webapps/mpp/about>
- /17/ PayPal 2017. PayPal Express Checkout Integration. Accessed 23.07.2017.
<https://developer.paypal.com/docs/integration/direct/express-checkout/integration-jsv4/>
- /18/ Margret, R. 2007. What is requirements analysis (requirements engineering)?
Accessed 25.07.2017.
<http://searchsoftwarequality.techtarget.com/definition/requirements-analysis>
- /19/ VAMK University of Applied Sciences Software Engineering. Accessed
25.07.2017. <https://portal.vamk.fi/course/view.php?id=4166>
- /20/ Charene, C. 2017. Quality Function Deployment for Competitive Advantage.
Accessed 26.07.2017. <https://www.isixsigma.com/tools-templates/qfd-house-of-quality/quality-function-deployment-competitive-advantage/>

APPENDICES

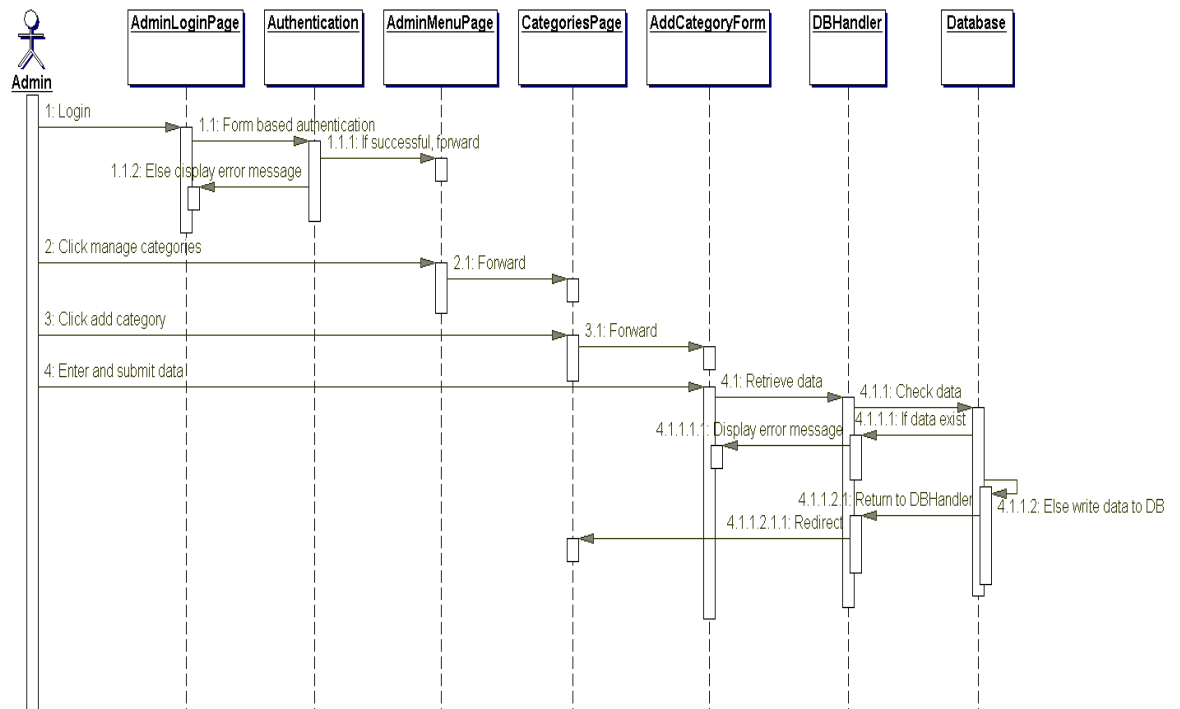


Figure 25. Add category sequence diagram.

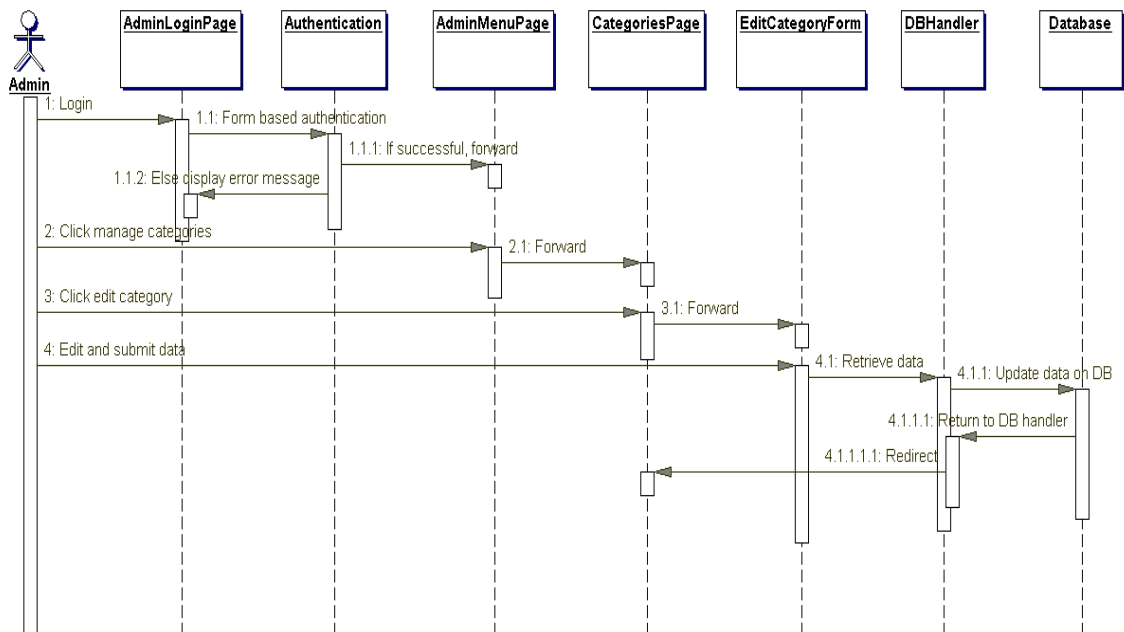


Figure 26. Edit category sequence diagram.

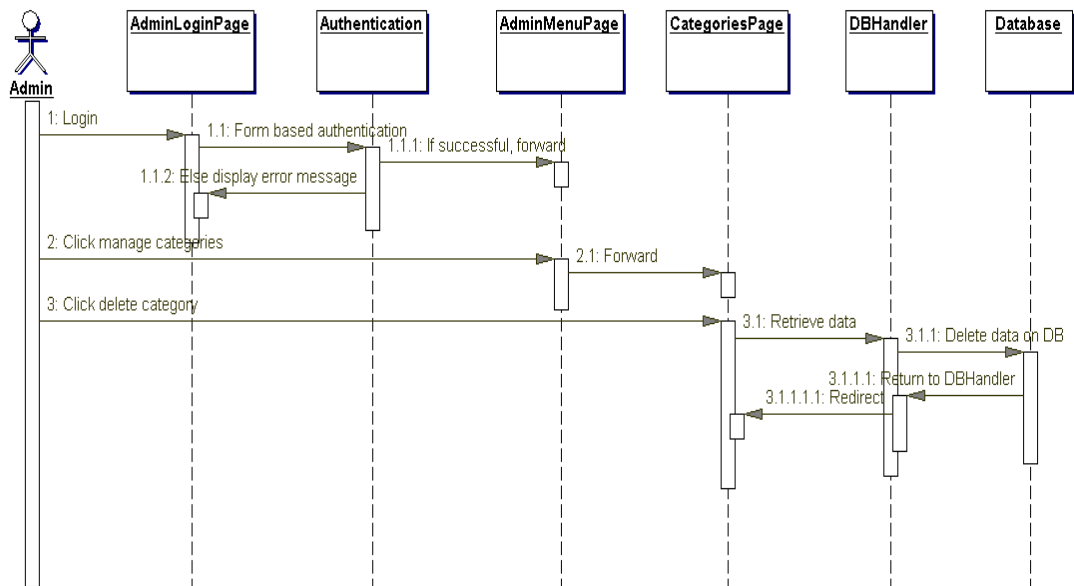


Figure 27. Delete category sequence diagram.

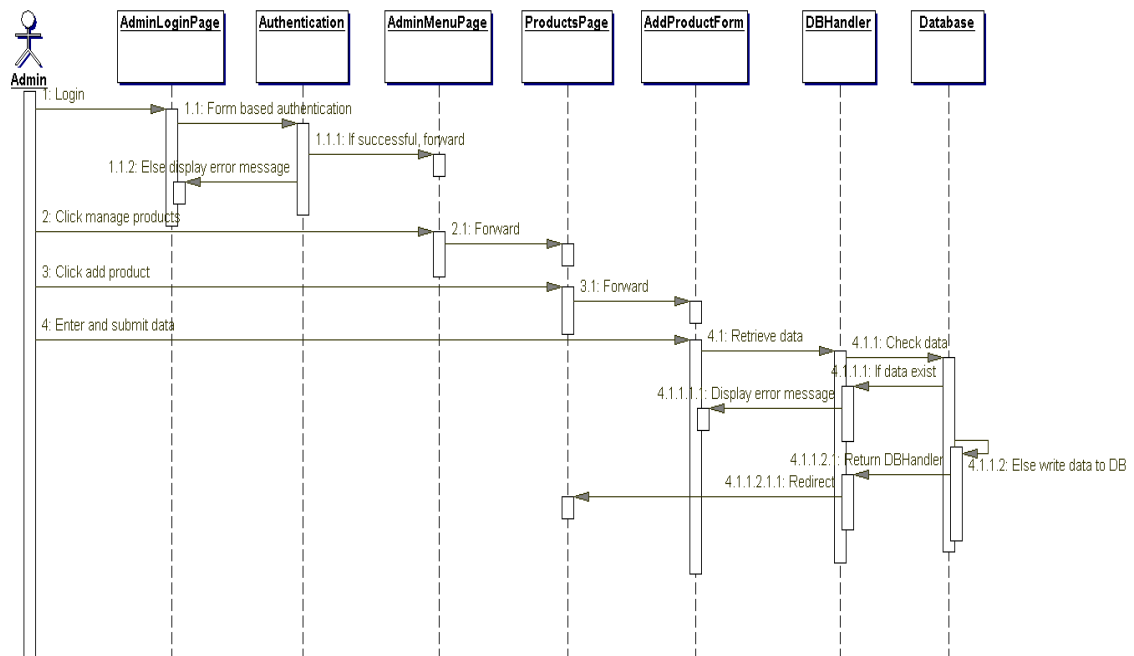


Figure 28. Add product sequence diagram.

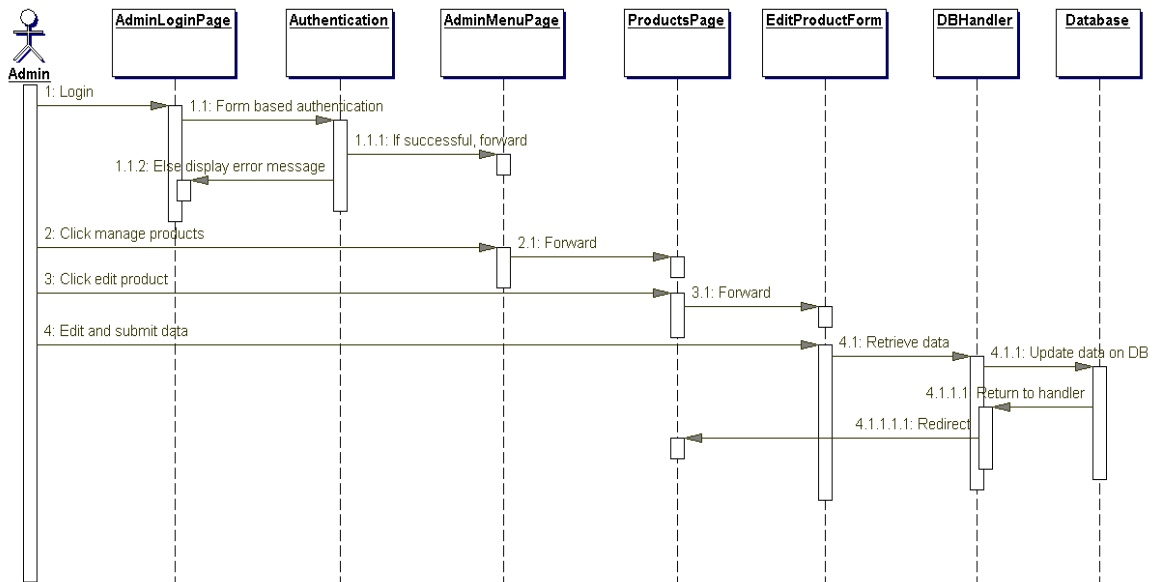


Figure 29. Edit product sequence diagram.

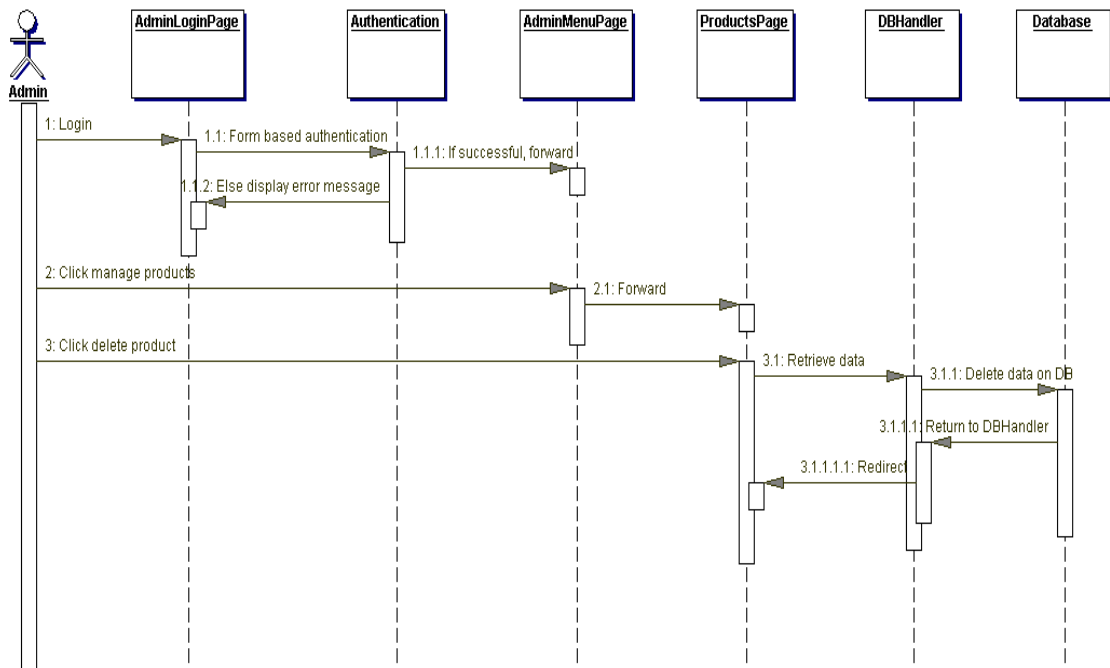


Figure 30. Delete product sequence diagram.

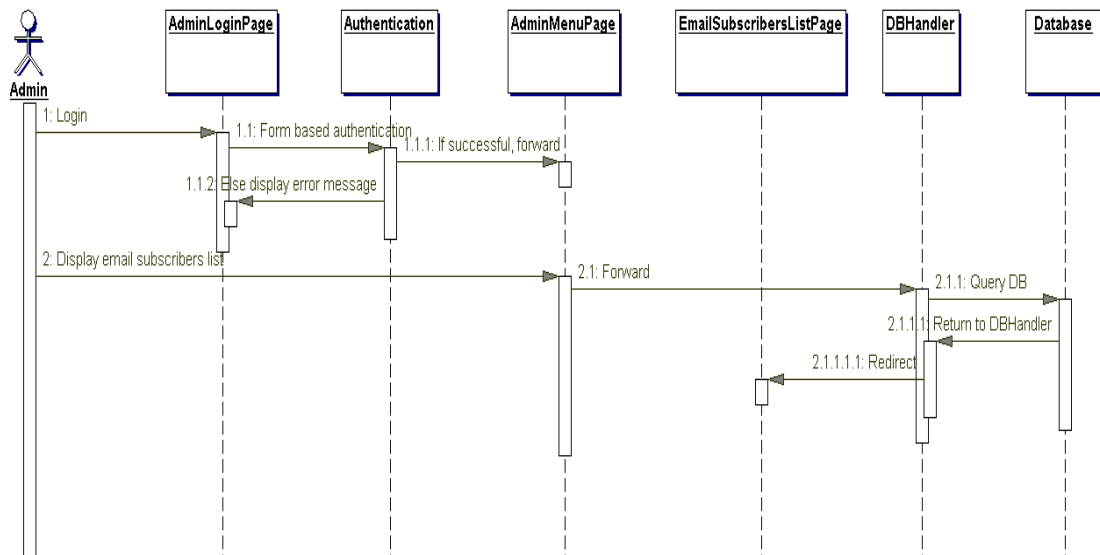


Figure 31. Display email subscribers' list diagram.

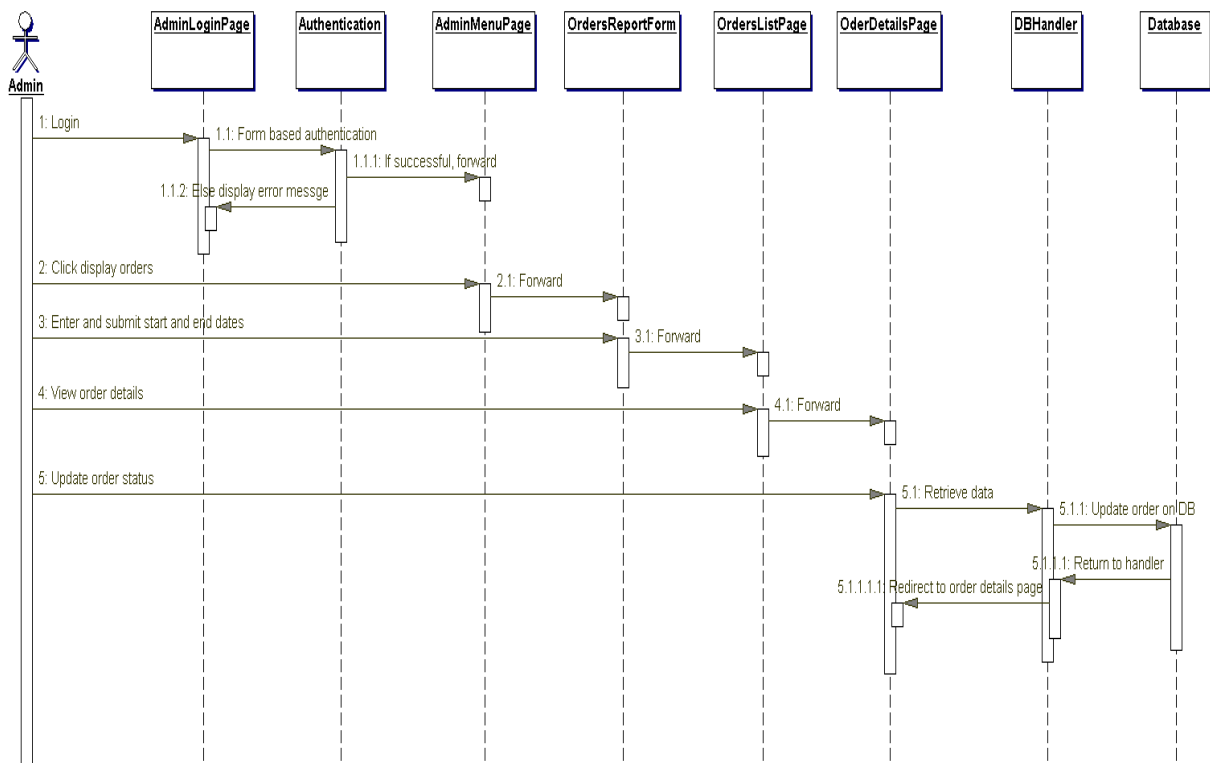


Figure 32. Display orders list sequence diagram.

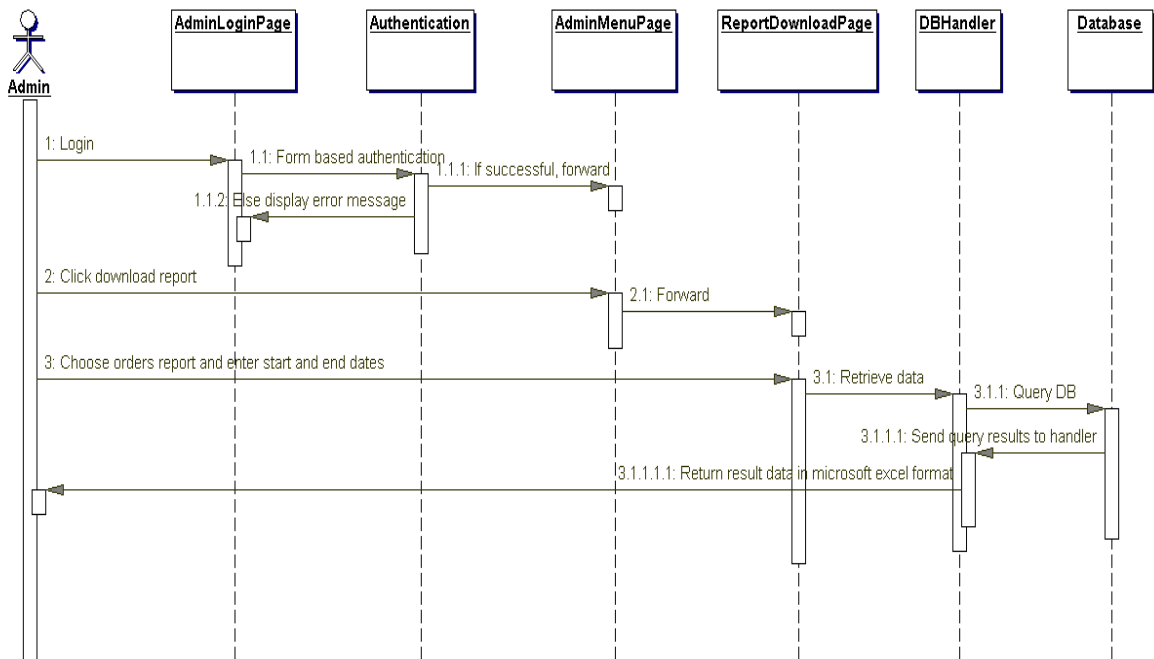


Figure 33. Download orders report sequence diagram.

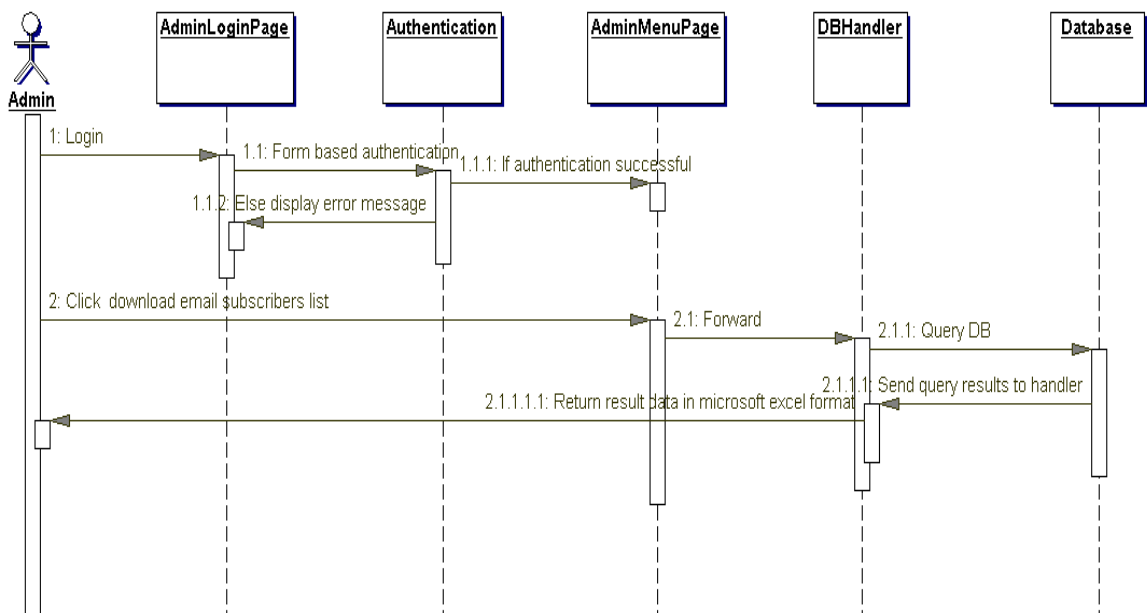


Figure 34. Download email subscribers list sequence diagram.