Web Component Based Ecommerce Application Development Framework for Hosted Software Solutions (Pvt) Ltd.

A dissertation submitted for the Degree of Master of Computer Science

W.T.K Dilhara
University of Colombo School of Computing
2018

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: W.T. K Dilhara

Registration Number: 2015/MCS/022

Index Number: 15440225

_____

Signature: Date:

This is to certify that this thesis is based on the work of

Ms. W.T.K Dilhara

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Dr. K.L Jayaratne

_____

Signature: Date:

Abstract

With the growth of software size and complexity, the traditional approach to building software from zero, it becomes ineffective in terms of productivity and cost. The guarantee of product software quality makes it almost impossible, discouraging the introduction of new technologies. To meet the demands of quality, modernized software on a large scale, there are new paradigms of development that facilitate the creation of evolvability systems, flexible, reliable and reusable. One of these paradigms is the software component-based development (CBSD) (Engineering, Component Based Software) and is based on the concept of building elements of an application that must be the independent, reusable pieces of code.
Research problem is built on that same CBSD scenario, but it is for web-based ecommerce applications. Leading ecommerce solution providers in Sri Lanka and USA need proper solution for mentioned problem.

Web component-based E-Commerce Application Framework with Component should be able to plug and play with other components and/or frameworks so that component can be composed at run-time without compilation. When Web Application Live Run/Host, add new module to existing web, no need to down the system and want to upload relevant module to existing base web application.

Acknowledgements

I would like to extend my sincere gratitude to everyone who has helped me to make this research a success.

It is my responsibility to be thankful to the staffs who are working at Hosted Software Solutions (Pvt) Ltd., without their support my Research Project and Documentation would not possible thing. The special thank is go to owner of the business. Without his permission, were unable to do this.

Especially I give my gratitude to my project supervisor Dr. K.L Jayaratne, for the guidance and support provided me during the project, even during his busy schedules. The guidance I got from him during those hard times which I came across was more than very helpful to drive the research in the correct direction and to choose best approaches and techniques.

I wish to express my sincere gratitude to my colleagues and my friends who participated in the research for giving valuable information and suggestions spending their valuable time. Most of the information and knowledge was extracted from the Internet, and I am thankful for those individuals who shared information and ideas through the World Wide Web.

I would like to thank all my colleagues at MCS batch who gave me valuable ideas and support. And special thanks should go to my work place, Hosted Software Solutions (Pvt) Ltd. for the support given to me during my MSC studies in last two years.

Finally, I would like to mention my parents who were encouraging and supporting me, during the period of research study as well as during the postgraduate studies for nearly two years.

Thank you

Table of Contents

Abstract i

Acknowledgements ii

List of Figures v

List of Tables vi

List of Abbreviations vii

Chapter 1 : Introduction 1

1.1 General Introduction to the Problem 1

1.1.1 What is Component Based Software Engineering? 1

1.1.2 Definition and characteristic of components 1

1.1.3 Differences from object-oriented programming 2

1.2 Motivation 3

1.3 Aims and Objectives of the Project 3

1.4 Scope 4

1.5 Structure of the Thesis 4

Chapter 2 : Background / Literature Review 5

2.1 Selection of Related Course Works 5

2.2 Analysis of the related research work 14

2.3 Identification of Research Gap / Problem 19

Chapter 3 : Methodology 20

3.1 Component Based Software Development vs Traditional Software Development 20

3.2 Component Based Software Development vs Conventional Software Reuse 20

3.3 Architecture and Methodology 21

Chapter 4 : Proposed Solution Details 23

4.1 Characteristics of proposed software component 23

4.2 Proposed High Level Architecture of CBSD for eCommerce Solution 23

4.2.1 Main Components 24

4.2.2 Sub Components 25

4.3 Used Technologies 25

4.4 Implementation 28

4.4.1 Typical MVC Project Architecture 28

4.4.2 Proposed System Design Architecture 29

4.5 Proposed System Screen Designs 31

4.5.1 Home Page 31

4.5.2 Promotion Area 31

4.5.3 Store Front Area 32

4.5.4 Top Rated Items 32
4.5.5 Shopping Cart 33
4.5.6 Checkout 33
4.5.7 My Account Area 34
4.5.8 Admin Panel - Categories 34
4.5.9 Admin Panel – Add / Edit Items 35
Chapter 5 : Evaluation and Results 36
5.1 Evaluation Plan 36
5.2 Evaluation Approach 36
5.3 Evaluation Methodologies 37
5.4 Results 38
5.4.1 Questionnaire Evaluation 39
5.4.2 Analysis of Designation 42
5.4.3 Analysis of Higher Educational Level 42
5.4.4 Analysis of Age Category 43
5.4.5 Analysis of Experience in working Hosted Software Solutions (Pvt) 43
5.4.6 System Evaluation 44
Chapter 6 : Conclusion 46
6.1 Introduction 46
6.2 Findings and Limitations 46
6.3 Future Work 47
References 48
Appendices 51
Appendix A: Questionnaire 51
Appendix B: Interview Questions 55

List of Figures

Figure 1 A simple example of several software components - pictured within a hypothetical holiday-reservation system represented in UML 2.0 2
Figure 2 Citation level of papers by years 7
Figure 3 Component models 8
Figure 4 Programming languages 9
Figure 5 Framework types 10
Figure 6 Visual example of how a component-based development style streamlines your processes 17
Figure 7 CBSE methodologies 22
Figure 8 Proposed High-Level Architecture of CBSD for eCommerce Solution 24
Figure 9 Areas in ASP.Net MVC 26
Figure 10 The Agile Process and Lifestyle 27
Figure 11 Typical MVC Projects with Multiple Areas 28
Figure 12 Modules will build as separate MVC Projects 29
Figure 13 Module Manager Screen 29
Figure 14 Modules after plugged in to base project 30
Figure 15 Home Page 31
Figure 16 Promotion Area 31
Figure 17 Store Front Area 32
Figure 18 Top Rated Items 32
Figure 19 Shopping Cart 33
Figure 20 Checkout 33
Figure 21 My Account Area 34
Figure 22 Admin Panel - Categories 34
Figure 23 Admin Panel – Add / Edit Items 35
Figure 24 Analysis of Designation 42
Figure 25 Analysis of Higher Educational Level 42
Figure 26 Analysis of Age Category 43
Figure 27 Analysis of Experience in working Hosted Software Solutions (Pvt) 43
Figure 28 Questionnaire Evaluation - Hosted Software 52
Figure 29 Evaluation Sheet of Questionnaire 53
Figure 30 Criteria of Evaluation Sheet 54

List of Tables

Table 1 Categorized - Citation Based 5
Table 2 Categorized - Conference Proceedings 6
Table 3 Overview component model & programing language 8
Table 4 Usage of CBD and connection to architecture 12
Table 5 Component Based Software Development vs Traditional Software Development 20
Table 6 Component Based Software Development vs Conventional Software Reuse 21
Table 7 Questionnaire Evaluation - Hosted Software 41
Table 8 System Evaluation Data Sample 45
Table 9 System Evaluation Diagram 45

List of Abbreviations

CBSE	Component Based Software Engineering
CBD	Component Based Development
SoC	Separation of Concerns
UML	Unified Modelling Language
OOP	Object-Oriented Programming
MVC	Model View Controller
CMS	Content Management System
SSE2	Streaming SIMD Extensions 2
AVX2	Advanced Vector Extensions 2
WPF	Windows Presentation Foundation
API	Application Programming Interface

Chapter 1 : Introduction

1.1 General Introduction to the Problem

1.1.1 What is Component Based Software Engineering?

Component Based Software Engineering (CBSE) is also popular as component development (CBD), concept has emerged in mid of 1990s. This concept is a development of separation of concerns (SoC) [1], where we divide the software development project based on its functionality or its architecture. It is a re-use-based approach-to define, implement and deploy isolated components and plug & play with a main system. This practice is aimed at creating the same level of benefits for both the short and long term for the software itself and the sponsoring organization. Components may issue or retrieve events and may be used for EDA-driven event architecture [2].

1.1.2 Definition and characteristic of components

An individual software component is a software package, a web service, a web resource, or a module that encapsulates a set of related functions (or data). All system processes are placed into separate components so that all of the data and functions inside each component are semantically related (just as with the contents of classes). Because of this principle, it is often said that components are modular and cohesive. With regard to system-wide co-ordination, components communicate with each other via interfaces. When a component offers services to the rest of the system, it adopts a provided interface that specifies the services that other components can utilize, and how they can do so. This interface can be seen as a signature of the component - the client does not need to know about the inner workings of the component (implementation) in order to make use of it. This principle results in components referred to as encapsulated. The UML illustrations within this article represent provided interfaces by a lollipop-symbol attached to the outer edge of the component. However, when a component needs to use another component in order to function, it adopts a user interface that specifies the services that it needs. In the UML illustrations used interfaces are represented by an open socket symbol attached to the outer edge of the component as illustrated in Figure 1.

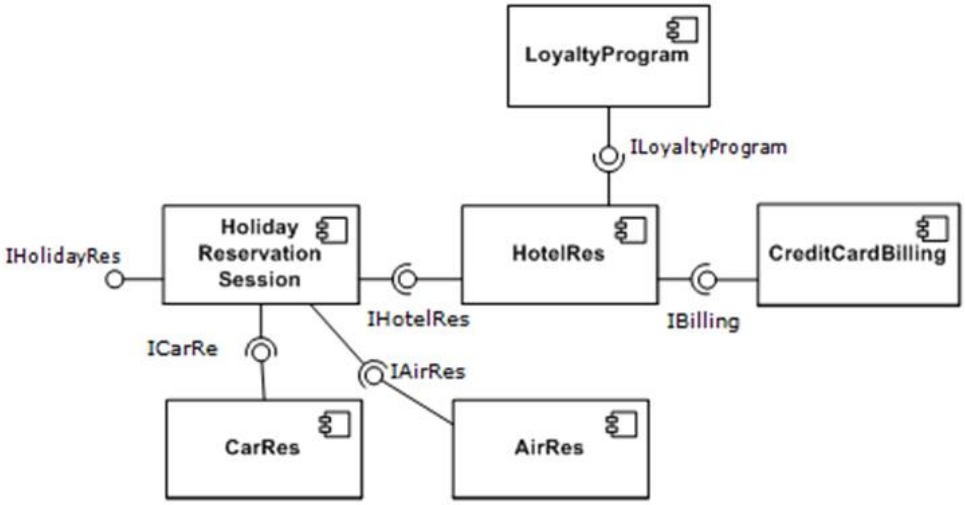


Figure 1 A simple example of several software components - pictured within a hypothetical holiday-reservation system represented in UML 2.0

1.1.3 Differences from object-oriented programming

Proponents of object-oriented programming (OOP) maintain that software should be written according to a mental model of the actual or imagined objects it represents. OOP and the related disciplines of object-oriented analysis and object-oriented design focus on modeling real-world interactions and attempting to create "nouns" and "verbs" that can be used in more human-readable ways, ideally by end users as well as by programmers coding for those end users.

Component-based software engineering, by contrast, makes no such assumptions, and instead states that developers should construct software by gluing together prefabricated components - much like in the fields of electronics or mechanics. Some will even talk of modularizing systems as software components as a new programming paradigm. Example for possible paradigm: many experts feel adaptability to evolving needs is more important than reuse, since 80% of software engineering deals with maintaining or releasing new versions. So it is desirable to build complex system by assembling highly cohesive loosely coupled large components, where cost of redesigning each of such adoptable components (or replacing by a better component) must be minimized.

Some argue that earlier computer scientists made this distinction, with Donald Knuth's theory of "literate programming" optimistically assuming there was convergence between intuitive and formal models, and Edsger Dijkstra's [3] theory in the article The Cruelty of Really Teaching Computer Science, which stated that programming was simply, and only, a branch of mathematics.

In both forms, this notion has led to many academic debates about the pros and cons of the two approaches and possible strategies for uniting the two. Some consider the different strategies not as competitors, but as descriptions of the same problem from different points of view.

One approach to creating component-based software using object-oriented programming is interface-based programming. However, interface-based programming does not inherently support distributed systems, and many computer systems are inherently distributed in the 21st century. Interface-based programming in the OOP Invalid source specified. sense may be extended to distributed systems with distributed component object models; however, many have argued in recent years that REST APIs or the actor model are more suitable approaches.

1.2 Motivation

My research problem is built on same CBSD scenario but it is for web based e-commerce applications. This is an actual issue in Hosted Software Solutions (Pvt) Ltd. This company is a leading ecommerce solution provider in Sri Lanka and USA. Main products in this company are e-Commerce and online food order solutions. Current Web Solutions are rich in features but it is hard to add a new feature to it and decouple an existing feature from it. Because of that even though customer do not want to buy fully featured online store (customer do not want item rating and facility), customer should buy it. In my research tries to find a solution to above problems and build a framework for web-based ecommerce solution development.

1.3 Aims and Objectives of the Project

Intention is this research is to develop a common framework which follow component based loosely coupled but highly integrated software component framework for e-commerce application development. Also, this research will search for suitable technologies to follow 5 for the CBSD in web-based e-commerce solutions in UI, Backend and Database level. Proposed solution will feature to develop web-based e-commerce features in module basis and those modules will able to plug-in to the main web solution when it needs and plug out form the main web solution when do not need.

1.4 Scope

Scope of the research is limited to Software Developments specially web-based online solutions and services. Specially this framework will be proposed for e-commerce solutions which mainly has its business scope in Sri Lanka, USA and UK. So those business requirements will consider on this research. Big picture of this research is to build a conceptual model framework and couple of sample modules which can plug in and plug out to the main web solutions.

1.5 Structure of the Thesis

The rest of this thesis is structured as follows:

Chapter 2 discusses about the background of this implementation with related publication on literature. Selection of Related Course Works, Analysis of the related research work and Identification of Research Gap / Problem also cover with this chapter.

Chapter 3 describes methodology. Aspects relating to the proof of concept specification which includes design assumptions relating to the scope of the proof of concept, prototype architecture contains in this chapter.

Chapter 4 discusses about Proposed Solution Details. chapter includes all the details about the proposed solution. Also describe Form of the solution and how it is formed, rationale behind the development of such solution.

Chapter 5 presents the findings and the evaluation of the research. Includes aspects such as designed experiments, results obtained and critical evaluation of the research work.

Chapter 6 summarizes the work, discusses its findings and contributions, points out limitations of the current work, and outlines directions for future research.

Chapter 2 : Background / Literature Review

2.1 Selection of Related Course Works

Following are related works which relevant to CBSD and eCommerce frameworks. Researcher has categorized the relevant articles, website documents, journals, conference proceedings and research papers, review papers and white papers as shown in Table 1.

Citation Based	Year	Cited by
Component-based frameworks for E-commerce	2000	74
Open MVC: A non-proprietary component-based framework for web applications	2014	9
Efficiently Distributing Component-Based Applications Across Wide-Area Environments	2003	4
The pataphysics of creativity: Developing a tool for creative search	2013	2
A Web Application Framework for End-User-Initiative Development with a Visual Tool	2012	2
An approach to formally modeling the component-based e-commerce system	2005	2
Blending E-Commerce Theory and Application	2005	1
Composing user-specific web applications from distributed plug-ins	2013	1
Nested web application components framework: A comparison to competing software component models	2013	0
Support for development and test of web application: A tree-oriented model	2011	0
Nested web application components framework: A comparison to competing software component models	2013	0
Component based Framework to Create Mobile Cross-platform Applications	2013	-

Table 1 Categorized - Citation Based

Conference Proceedings	Publisher
Thirteenth International World Wide Web Conference Proceedings, WWW2004	ACM
Lecture Notes in Computer Science	Springer
Proceedings - International Conference on Next Generation Web Services Practices, NWeSP 2005	IEEE
Web3D Symposium Proceedings	ACM
Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining	ACM
Proceedings of the ACM Symposium on Applied Computing	ACM
Lecture Notes in Computer Science	Springer
Lecture Notes in Engineering and Computer Science	Springer
Thirteenth International World Wide Web Conference Proceedings, WWW2004	ACM
Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference	IEEE
Lecture Notes in Computer Science	Springer
ENASE 2013 - Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering	Springer
2008 34th Euromicro Conference Software Engineering and Advanced Applications	IEEE
Lecture Notes in Computer Science	Springer
5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering,Software Architecture and Reuse (ICIS-COMSAR'06)	IEEE
Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Applications	IASTED
ICEIS 2005 - Proceedings of the 7th International Conference on Enterprise Information Systems	ICEIS

Table 2 Categorized - Conference Proceedings

In Figure 2, which presents number of cited papers by year, it can be noticed that most of the citations are from between 2005 and 2006, more than 50 %. But surprisingly the most cited paper is from 2007, and the more recent one, from 2014 is cited 9 times. It will be interesting to see if the rising trend as seen from 2011 up to 2014 will continue.

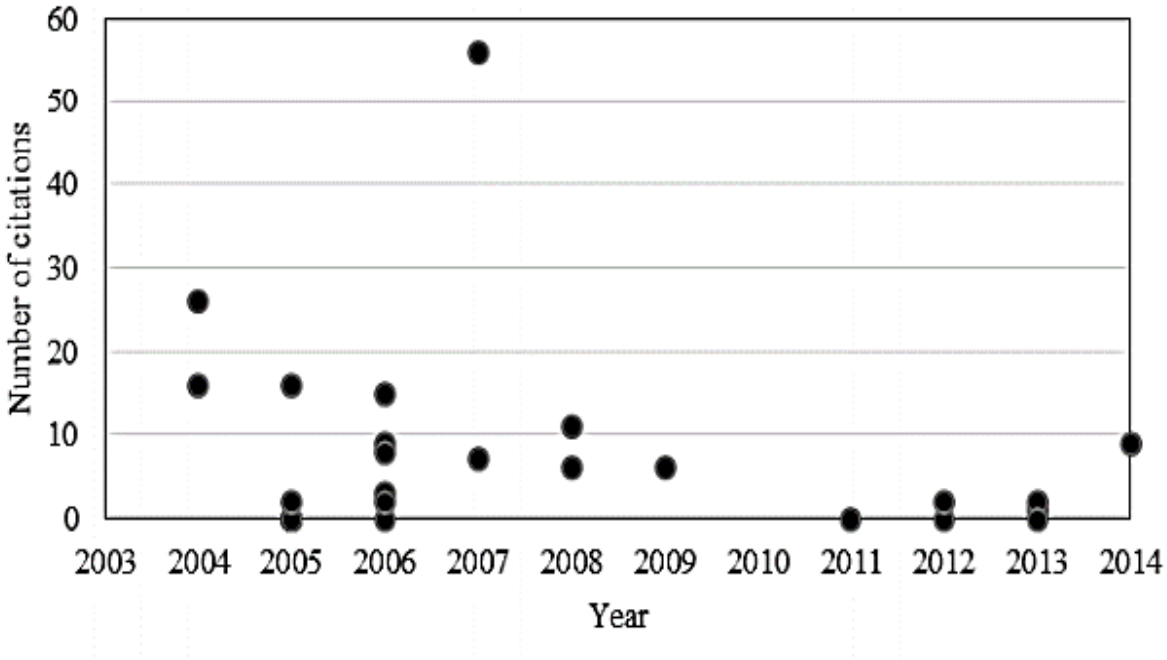


Figure 2 Citation level of papers by years

In Table 3 and Figure 3 one can notice that lot of papers have not defined a component model. The most used component model is some variation of JavaBeans consequently making Java the most popular development language (usually J2EE).

Component model	Programing language
not defined	.net, J2EE
Fractal	Java
CBTOWADM	note defined
JavaBeans, EJB	J2EE
Contigra	XML
not defined	J2SE
not defined	XML
not defined	not defined
Plux (plug and play like OSGi or SOFA 2.0)	.net (can be implemented in Java)
EJB	Java, EJB
not defined	Java Servlets with JSON
XVM	Java for build XVM Framework, XML for building Web applications
JavaBeans, EJB	J2EE, EJB
JavaBeans, EJB	J2EE, EJB
not defined	PHP + Smarty
not defined	ASP .NET
not defined	MATLAB 6.5 and a discrete-event simulator
not defined	not defined, (any language is possible to use)
COM	HTML, ASP, ActiveX, COM
JavaBeans	J2ME, JavaBeans
Corba, (possible to use DCOM, EJB)	Corba
CWBDM	not defined, but proposed architecture is based on Java
not defined	not defined, (any language is possible to use)
not defined	not defined, (any language is possible to use)

Table 3 Overview component model & programing language

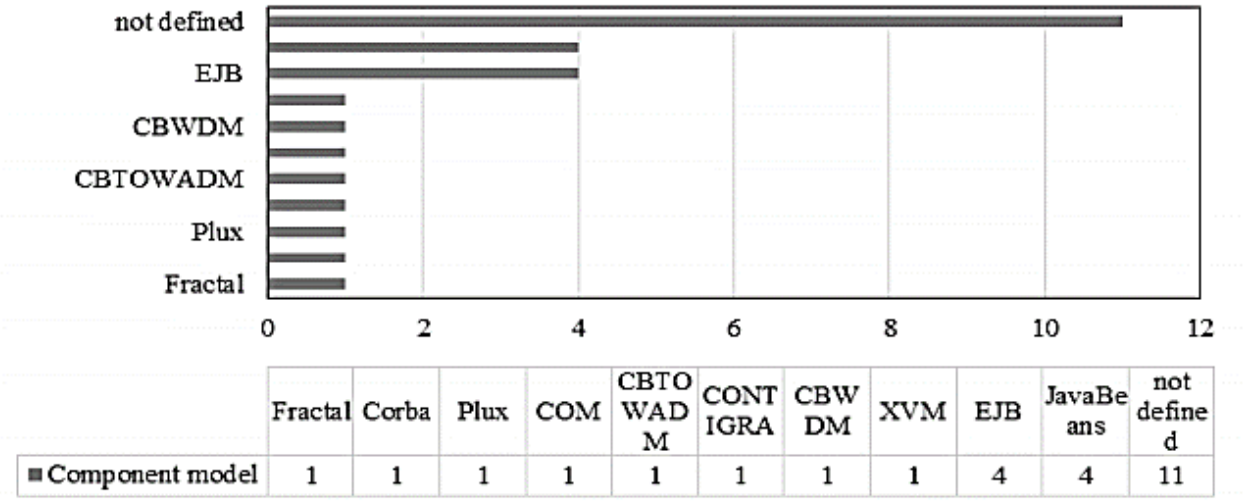
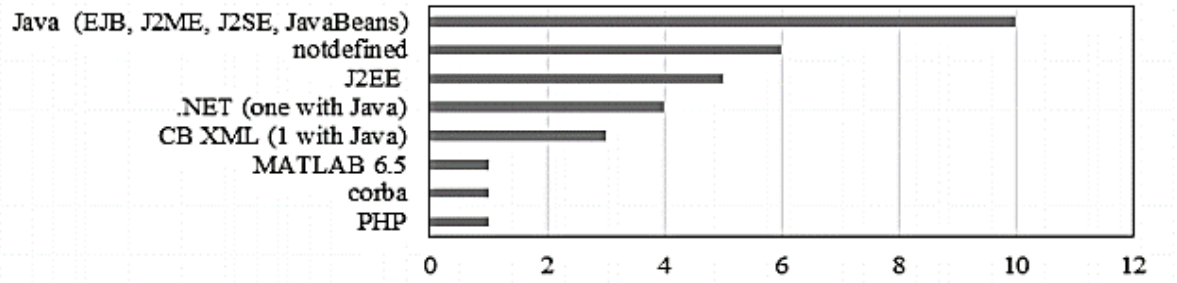


Figure 3 Component models



	PHP	corba	MATLAB 6.5	CB XML (1 with Java)	.NET (one with Java)	J2EE	notdefined	Java (EJB, J2ME, J2SE, JavaBeans)
Programing language	1	1	1	3	4	5	6	10

Figure 4 Programming languages

Both JavaBeans and EJB are used in several times. In lot of them, JavaBeans and EJB are used simultaneously, some papers use COM and Corba component models which are well known, and the remaining ones use custom component models. Considering programming languages (Figure 4), Java is most widely used, with 12 out of 21 papers using it. Among these, 10 of which using explicitly Java and in two paper Java is an option (papers [4] and [5]). Paper [6] also uses Java, but only to build XML which is then used to develop web applications, therefor it is not counted since this XML can be generated in many programming languages.

Most papers do not explicitly report on any major problems while using the component approach, but rather they report suggestions for future researchers and practitioners concerned with CBD. All the suggestions are aggregated and presented below.

- Components should capture domain knowledge of web application development and hide complexities from End User [7].
- Components should not capture application domain specific knowledge into Components. Rather those specific application needs should be abstracted and generic Components (Tools and Engines) should be created that can be used across many application domains [7].
- Components should be easy to use by End Users, yet they need to be complete so that it aids full capture of all the necessary ‘Components parts’ of the application such as front-end pages, back end processing logic and database information [7].

- It is not an easy task to develop an in-house component framework or to integrate available preexisting COTS in enterprise applications. It needed far more efforts and investments than it was foreseen in the beginning (approximately 50% more work than expected) [8].
- It is not one-time effort but continuous process, which needs considerable investment in time and resources [8].
- The percentage of reusability changes from application to application and often needs component modification and reconfiguration [8].
- The major benefit of an in-house Component framework development surprisingly is not the project cost and time reduction based on business logic and business functions reusability (our but the company knowledge sharing and the creation of business function components [8].
- Component composition: Each component is designed to achieve some special task; several components can be composed together in a dependent series to achieve a larger task [9].
- Problem in distributed systems is distributed component management [9].
- Problem is redesign of components to be more generic, simple and fast integration procedure with arbitrary Web applications [10].

All the selected papers authors use some type of a web development framework, which can be divided in two groups; general and specific. General frameworks are used to develop any kind of web application, i.e. they can be used in many domains, while specific frameworks are specialized or limited to only a certain type of web application, i.e. a certain domain. As it can be seen in Figure 5, authors tend to use general frameworks, however the number of specific ones is significant.

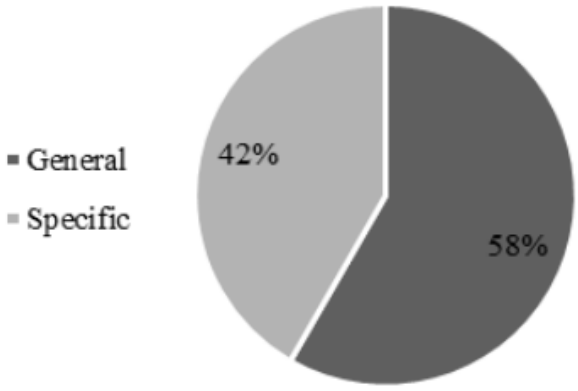


Figure 5 Framework types

Usage of Components	Connection to architecture
"Components are responsible for realizing the application logic related to the associated element and providing services to other components."	XVM architecture is component-based architecture and enables dynamic composition of components. XVM manager is responsible for handling all components. "In the XVM architecture, the key idea is a mapping between XML elements and software components, which associates XML elements with software components."
Two types of Components: " (a) Tools that allow End Users to create and assemble applications and (b) Engines that could be used to run these applications."	Architecture not particularly described. Framework is built so that end-users can use existing components to build new applications, or developers can create new component include into Framework and this can be further used again by end-users.
Components contain business logic or presentation logic, with connectors to HTTP, JDBC, JNDI, CORBA, RMI	Architecture is multi-tier (client, web (presentation), business, and database). Components are used at web and business tier.
Components have specialized functionalities to client modules that require server-based functionality (e.g. data analysis or computation of visualizations that require large data set)	Client-server architecture is used. Components are used on server side.
"Using component-based programming, we developed a highly maintainable system, which contains three components packages: Monitoring Controls Package based on ActiveX, Analysis Controls Package based on ActiveX, and Diagnosis Algorithms Package based on COM."	"A four-tier model based on the Microsoft's tier concept is adopted in the WRMFDS, which consists of the Presentation Services Tier, the Application Services Tier, the Data Access Services Tier and the Database Services Tier."
Layered structure of components is used. The top layer consists of	Three-tired-architecture is used (MVC pattern). The tires are: server, client and databases.

components which are used to build final applications (exp. welcome component, login components …)	Components are used on server side. “A request-response pair contains three parts (Model, View, controller) and forms a unit. Each unit is implemented by reusing component libraries in the layered component structure and each unit can be plug-and-play into the system.”
Component implement backend functionalities, which end-users can use when building web applications (exp. search component for searching some data table). Two types of components: Components that implement functionalities that are not domain dependent and components that are domain-dependent.	Three-tired-architecture (client, application server, DB server) with MVC pattern is used. Components are used in application server.
Component can be anything. Every user can add his own components. Components can be server-side that are installed and executed on server. Client side installed and executed on client and use local resources. And sandbox components installed on server and downloaded to client on demand and executed on sandbox on the client.	Component based plug-in architecture. Different components are combined by end users and web applications created.
Framework itself is a component that can be integrated into another web applications, but also consist of components which consist of components "nested components".	MVC architecture, components are used on all MVC layers.

Table 4 Usage of CBD and connection to architecture

Table 4 shows for which purpose authors used component-based development and how did it affect the software architecture of their web applications. There are three ways of component approach usage which one can distinguish:

- Components are used for creating web development frameworks – in this approach authors create component-based frameworks which are used to create web applications, which can, but don't need be component based.
- Components used as application building blocks – in this approach components are used to create component-oriented web applications without the underlying framework.
- Mixed approach – both framework and web application developed with this framework are component oriented. In all the above cases the architectural decision is made solely by the end user, and all papers report only on developing prototypes (weather it is a framework or a web application). While most of the authors use component approach on the server side to implement various services, on the client side.

2.2 Analysis of the related research work

Component-based software engineering (CBSE) [11] is an approach to develop software that relies on software reuse. The success of the final system is completely based on the component-based software engineering that sometimes depends on the previous successful or failed case experience, previous decision and helps us to select the component that leads to the final system. It may reduce time for software development. In software development, Component-Based Software Development (CBSD) is a new phase that helps in development of complex software from the integration of pre-build components instead of developing everything from scratch. There are several problems (For example: Integration, Maintenance, Testing, etc.) occurred in the selection of components for integration.

A component is the unit of a system that offers predefined service and must be able to communicate with the other components. Component based software engineering (CBSE) mainly focuses on building large software systems with integrating previously-existing software components/modules [12].To improve software reusability and maintainability so many techniques have explored by software engineers for easing distress and pains that are given by complexity of software system, i.e. Modularization of software, object oriented techniques. Object-oriented technique is an approach to designing modular, reusable software systems. The goal of Component based software development is to reuse components based on previous conclusion and experience. But, the main goal of Component based software development is that it reduces the time for software development [13]. By using previous experience and the knowledge of failure can reduce the overall development time. Difficulties faced by the developers are not only to overcome the common properties are sufficient but enable a software developer to make such confidence to select suitable components to be used. The main problem find was that how to select the component from the available list of components which satisfy the requirements of the system.

Component-based UI development is not just the future of the web. It is a technique that digital application owners need to implement right now. Developing with a component-based user interface creates a sustainable technical architecture, saving time and costs. It also ensures a consistent experience across a portfolio of applications. “component” as an independent piece of software. This standalone, discrete piece of software has a clear boundary that is accessible via an API and contains all the application dependencies. This enables teams to build the user interface quickly, leveraging the library of components.

There are lot of benefits to Using a Component-Based approach. Briefly can describe some of the benefits of Component-Based approach.

- It Allows for Reuse

  Components are atomic units and building with components allows for their reuse in future development cycles. Since technologies come and go, this is invaluable. If you build your application in a componentized format, you're able to swap the best components in and out.

  One of the challenges of reuse with other development types is that they are not internally built or that they include many dependencies. A component-based UI approach allows your application architecture to stay up to date over time instead of rebuilding it from scratch. You can build multiple applications that adhere to the intended design principles.

- A Component-based UI Approach Accelerates Development

  Using a component-based UI approach supports iterative, agile development. Components are hosted in a library from which teams can access, integrate and modify them throughout the development process.

  In the design process, instead of designing new components, the designer focuses time on extending the existing components and designing new components where required. This optimizes the design process without designing a new grid, layout, or navigation. Ultimately, this expedites the design and development process because of the level of reuse.

- It Ensures User Experience Consistency Across a Portfolio

  One of the major challenges for an organization is ensuring that a portfolio of applications provides consistent user experiences and interactions. The component library acts as a point of governance for the business, designers, and quality assurance teams. In the case of Quality Assurance (QA), teams often have challenges validating the user interface due to a lack of an approved set of user interface standards. The component-based approach enables the creation of a library that provides that approved reference point. This enables the QA team to govern the compliance to UX

standards across a portfolio of applications. It acts as a dynamic repository that the QA team can use to validate their tests.

- It Easily Integrates into the Development Process

  As components are created, production quality user interface code is managed within a source code repository such as GitHub. Application development teams are well versed in using source code repositories, and so they are able to extract the code as needed and incorporate it into the application. Leveraging the initial component as a starting point, development teams can extend it to meet their needs. Then they can submit it into the code repository for review and approval for inclusion.

  The component library can be versioned in the repository, enabling tracking of which applications are on which version of the approved UX. This also will facilitate the governance and update process.

- Component-based UI Development Optimizes the Requirements & Design Process

  Using the component-based library as a reference, product managers, business analysts and user experience designers can spend less time defining the detailed application functionality and user experience. As they work through the definition process and requirements elaboration, they can reference a component as the baseline for the requirement, and then only spend time defining the required extensions and business logic.

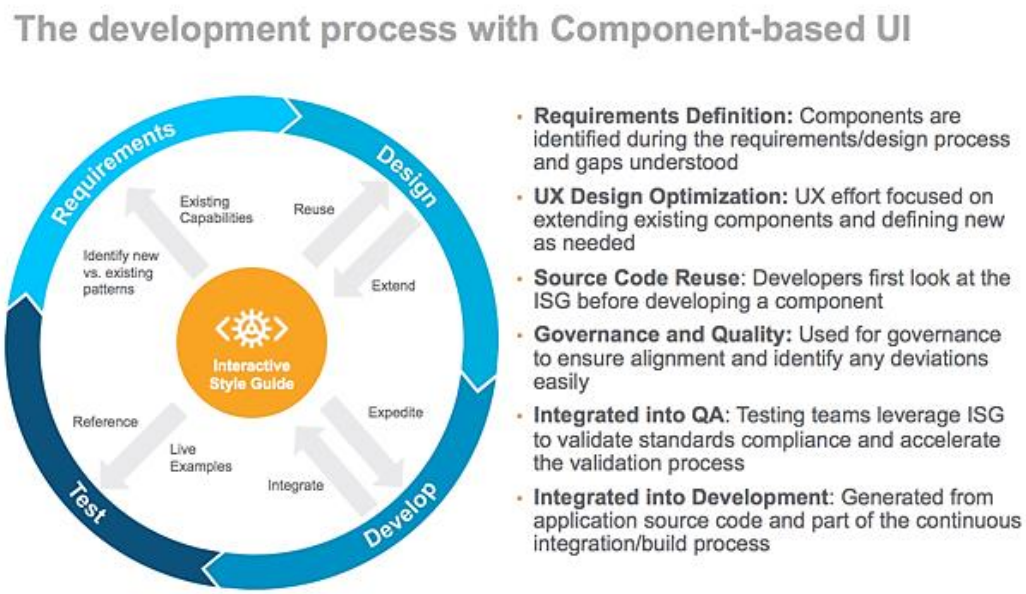


Figure 6 Visual example of how a component-based development style streamlines your processes

Component-based e-commerce technology is a recent trend towards resolving the ecommerce challenge at both system and application levels. Instead of delivering a system as a prepacked monolith system containing any conceivable feature, component-based systems consist of a lightweight kernel to which new features can be added in the form of components. In order to identify the central problems in component-based e-commerce and ways to deal with them, we investigate prototypes, technologies, and frameworks that will transcend the current state of the practice in Internet commerce. [14] Component implement backend functionalities; which end-users can use when building web applications. There are two types of components: Components that implement functionalities that are not domain dependent and components that are domain-dependent. [15] As it can be seen, in last four years general frameworks are preferred. Although there is one exception, one can notice that there seems to be the stabilization of the research domain. Initially, there was a lot of specific frameworks but due to growing complexity of web applications, researchers seem to use existing and already proven frameworks. [16] But there are limitations and competitive disadvantage in using general frameworks.

Components are content created form three layers. “A Web application can usually be described in three layers. Presentation layer, business logic layer, and database layer. Each layer can be partitioned and distributed among the CDN’s replica servers. [17]
In a modern economy, computer systems development determines to some extent the value added to the company's true profit and to a large extent the possibility of achieving business

success. Since the time when recording and storing data in computer memory became possible, there have been constant efforts to exploit the analysis of this data. The most advanced data analysis techniques involve data warehousing (to store large amounts of varied data) and business intelligence technology (to mine useful information and discover knowledge from data). Both areas have acquired new meaning thanks to e-commerce's systematic progress and evolution, which depend on and give impetus to the development of data acquisition, storage, and analysis technologies. So, we need knowledge about the models, technologies, and so forth, that can be applied to e-commerce systems development. Data Warehousing and Business Intelligence for E-Commerce attempts to meet this demand. [18] Another trend is the use of component frameworks for building network services. Their component-based nature makes such applications natural candidates for distributed deployment, but it is unclear if the design patterns underlying component frameworks also enable efficient service distribution. In this paper, we investigate the application design rules and accompanying system-level support essential to a beneficial and efficient service distribution process [19].

Another very important area of eCommerce CBD is mobile devices. In modern world, smartphones provide a set of native functionalities and another set of functionalities available through third-party applications. The emergence of more and more actors, without standards to provide their devices or OS, stops the cross-platform development. Indeed, a developer would have to learn many programmatic languages and create many user interfaces for many devices. To resolve this problem, several solutions often consist in the creation of a common SDK to only write the application once [20] . Then, same business logic uses for every deice or platform. In this paper, we propose a solution based on a component model.
The lack of standardized approaches in the development of web-based systems is an ongoing issue for the developers of commercial software. To address this issue, we propose a hybrid development framework for web-based solutions that combines much of the best attributes of existing frameworks but utilizes open, standardized W3C technologies where possible. This framework called open MVC is an evolution of the Model-View Controller (MVC) pattern [21].

2.3 Identification of Research Gap / Problem

There are large number of ecommerce frameworks available. 60% of eCommerce web-based frameworks are in open source form, these lead developers to choose features from various frameworks, but they are not easy to integrate, and plug & play feature is not there, even though it has plug and play feature in same framework components most of those have not met the required level of satisfaction for developers as well as clients. Following are the main questions which are not directly answered by related researches.

- In which way is CBD used for web application development
- What is the relation between CBD and web application development?
- Which component models are used for web application development?
- In which web application development domains is CBD used?

Chapter 3 : Methodology

3.1 Component Based Software Development vs Traditional Software Development

Traditional software development approach to the functionality of the system and mainly follow the sequential models like waterfall, which are mostly overridden by the Iterative and Evolutionary models like increment, prototyping, Boehm's Spiral Model. Component-Based Software Development is to address the development of systems as an assembly of parts (components), the development of parts as reusable entities, maintenance and upgrading of systems by customizing and replacing such parts.

Component-Based Software Development	**Traditional Software Development**
Building system from pre-existing components.	Building system from scratch.
Components and systems integrated from those components are developed.	Software system is developed.
Component selection and evaluation are special lifecycle phases.	In the lifecycle, there is no special phase like that.
Much effort is required in the selection of components, testing and verification phase.	Much effort is required for system development.
Reusability is the main theme.	Reusability usually not considered.

Table 5 Component Based Software Development vs Traditional Software Development

3.2 Component Based Software Development vs Conventional Software Reuse

Object-oriented technologies have produced software reuse, there is a big gap between the total systems and classes libraries. OOP include software design patterns, frameworks and architecture of reusable elements. Component-Based Software Engineering is an elite form of software engineering that offers the feature of reusability. Reuse of software artefacts and the process of reusability make CBSE a specialized paradigm of software development.

Characteristics	CBSE	Conventional
Architecture	Modular	Monolithic
Components	Interface and Black-Box	Implementation and White-Box
Methodology	Composition	Build from scratch
Process	Evolutional and Concurrent	Big-bang and Waterfall
Organization	Specialized: component Vendor, Broker and Integrator	Monolithic

Table 6 Component Based Software Development vs Conventional Software Reuse

As shown in Table 6 CBSE and Conventional comparison important point is an Architecture. Modular is more minimal and only provides the barebones functionality and structure for our application. Generally, only has one "responsibility". The code is more loosely couple where each part of code communicates in a more or less standard interface. Monolithic typically provides a tightly coupled codebase that makes a lot of assumptions about how the code interacts with each other. It usually includes everything we would need to get a web application up and running quickly.

3.3 Architecture and Methodology

CBSD leads to build a modular architecture. It helps to partially develop a system and incrementally enhance the processes, functions by adding and/or replacing components. Common component-based systems underlying software architecture such as MFC (Microsoft Foundation Class) and CORBA. Latest technologies are developed for web-based solutions such as .Net core, MVC, Angular and jQuery / JSON based libraries. Proposed solution will use these technologies in appropriate areas.

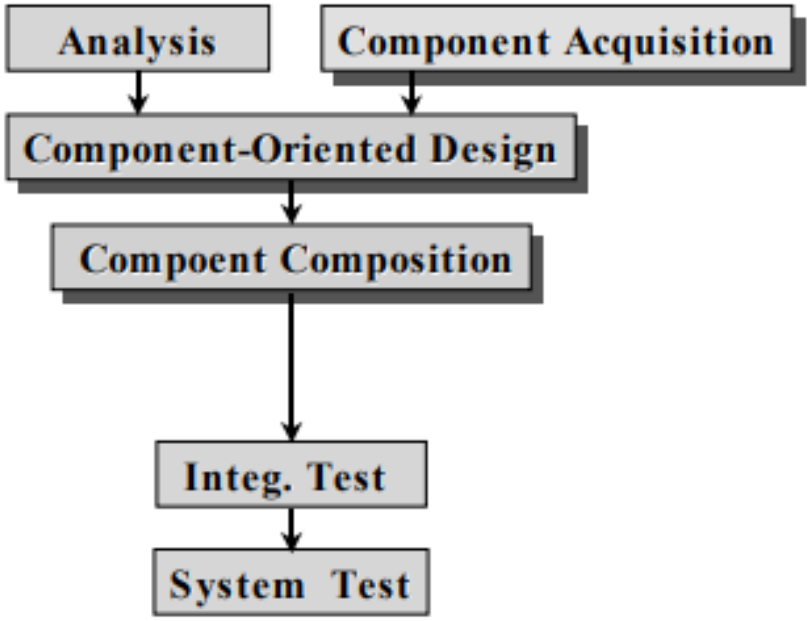


Figure 7 CBSE methodologies

CBSE focuses on connection of components through their interface. This connection also requires designing and develop collaborative behavior of multiple components. CBSE methodologies need to help interface-centric and behavior-oriented design such as connection-oriented programming and catalysis.

Chapter 4 : Proposed Solution Details

4.1 Characteristics of proposed software component

By looking at above two traditional development approaches following are the identified characteristics of single component.

1) Plug & Play: Component should be able to plug and play with other components and/or frameworks so that component can be composed at run-time without compilation
2) Interface-centric: Component should separate the interface from the implementation and hide the implementation details so that they can be composed without knowing their implementation.
3) Architecture-centric: Components are designed on a pre-defined architecture so that they can interoperate with other components and/or frameworks.
4) Standardization: Component interface should be standardized so that they can be manufactured by multiple developers or teams and widely reused across the company.
5) Distribution through Market: Components can be acquired and improved though competition market and provide incentives to the vendors.

4.2 Proposed High Level Architecture of CBSD for eCommerce Solution

As writer analyzed the existing eCommerce web-based solution following are the main processes identified.

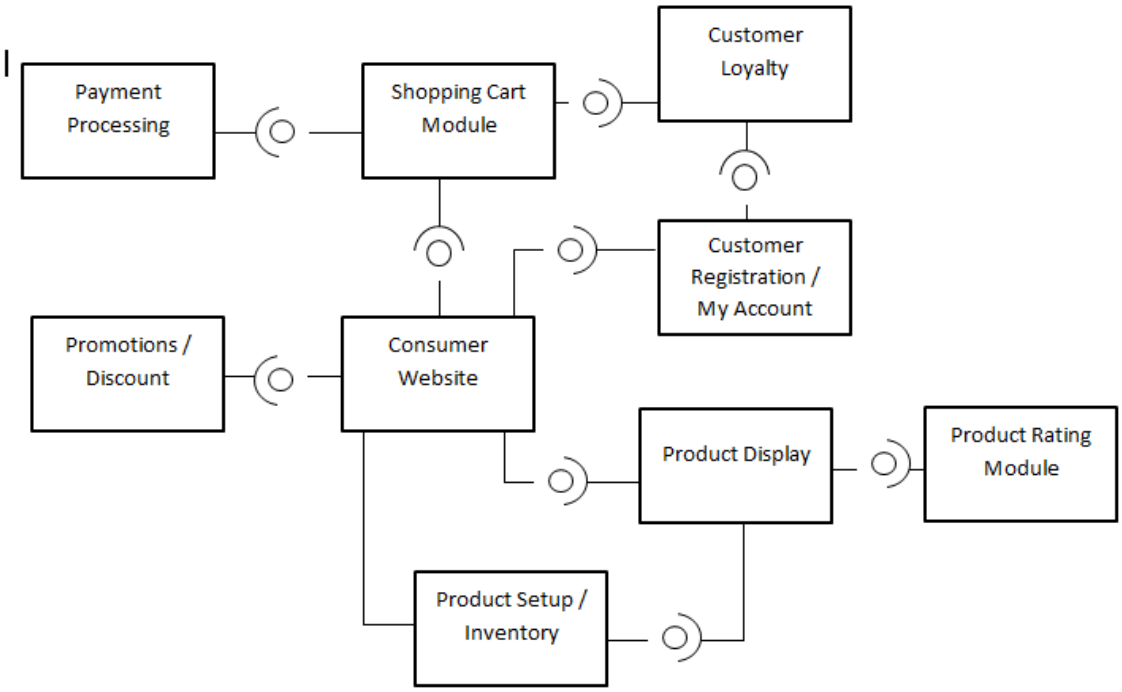


Figure 8 Proposed High-Level Architecture of CBSD for eCommerce Solution

These modules will be developed as separate independent software components which will able to plug into the base module. (Here writer has mainly focus on eCommerce process, Because, Hosted Software Solutions (Pvt) Ltd. is a leading ecommerce solution provider in Sri Lanka and USA. Main products in this company are e-Commerce and online food order solutions.)

4.2.1 Main Components

1) Consumer Website (Base)

Main website which directly accessed by customers. Design layout is fixed. Color changes, Images changes and text changes can be done via CMS Module.

2) CMS Module

Content Management Module of the website. Images, Text, Colors can be changed using this module.

3) Product Setup

Administration area of item display module, inventory, price management, etc. done here.

4) Product Display

5) Shopping Cart Module

6) Customer Registration / My Account

7) Product Rating Module
8) Promotions / Discount
9) Customer Loyalty
10) Payment Processing
11) Shipping / Delivery
12) Order Processing

4.2.2 Sub Components

Above components can have sub components which can be plugged in to its main component based on its behavior or process.
As an example, Payment Processing component has;

1) Credit card processing
2) Debit Card Processing
3) Pay on Delivery Processing
4) Bank Payment Processing

4.3 Used Technologies

1) Operating System – Microsoft Windows 10

Windows 10 is a personal computer operating system developed and released by Microsoft as part of the Windows NT family of operating systems. It was released on July 29, 2015. [22] It is the first version of Windows that receives ongoing feature updates. Devices in enterprise environments can receive these updates at a slower pace or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

2) Framework - Microsoft .NET Framework 4.6

.NET Framework 4.6 was announced on 12 November 2014. It was released on 20 July 2015. It supports a new just-in-time compiler (JIT) for 64-bit systems called RyuJIT [23], which features higher performance and support for SSE2 and AVX2 [24] instruction sets. WPF and Windows Forms both have received updates for high DPI scenarios.

3) Language - ASP.NET MVC 6

The ASP.NET MVC is a web application framework developed by Microsoft, which implements the model–view–controller (MVC) pattern. It is open-source software, apart from the ASP.NET Web Forms component which is proprietary.

4) Database Management System - Microsoft SQL Server 2014

SQL Server 2014 is relational database management system (RDBMS) designed for the enterprise environment. Released on April 1, 2014, SQL Server 2014 runs on the Structured Query Language (SQL), but has several notable differences from its immediate predecessor SQL Server 2012.

5) Areas Concept in ASP.NET MVC

Areas are some of the most important components of ASP.NET MVC projects. The main use of Areas is to physically partition web project in separate units. If we look into an ASP.NET MVC project, logical components like Model, Controller, and the View are kept physically in different folders, and ASP.NET MVC uses naming conventions to create the relationship between these components. Problems start when you have a relatively big application to implement. For instance, if we are implementing an E-Commerce application with multiple business units, such as Checkout, Billing, and Search etc. Each of these units have their own logical components views, controllers, and models. In this scenario, we can use ASP.NET MVC Areas to physically partition the business components in the same project. As illustrated in Figure 9 , Also, an area can be defined as: Smaller functional units in an ASP.NET MVC project with its own set of controllers, views, and models.

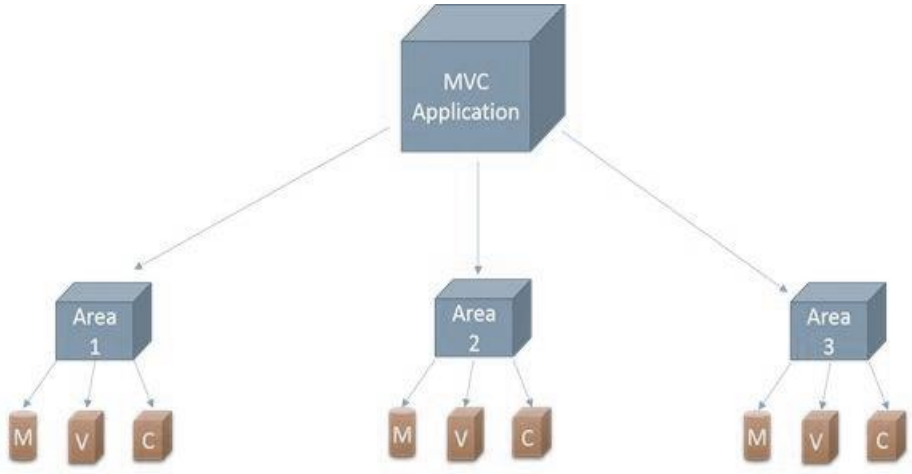


Figure 9 Areas in ASP.Net MVC

A single MVC application may have any number of Areas. Some of the characteristics of Areas are:

- An MVC application can have any number of Areas.
- Each Areas has its own routing configuration to define the request pattern inside that area.
- Each Area has its own controllers, models, and views.
- Physically, Areas are put under separate folders.
- Areas are useful in managing big web applications.
- A web application project can also use Areas from different projects.
- Using Areas, multiple developers can work on the same web application project.

6) Agile Methodology

Agile is a software development methodology that is becoming more popular every day. It defines the mind set of many software development teams working across the globe. Agile means ability to move quickly. Value Driven (Agile approach) is a new way of building software (Figure 10).

Figure 10 The Agile Process and Lifestyle

4.4 Implementation

4.4.1 Typical MVC Project Architecture

Even though MVC has a concept called areas they are not meant to build and deploy separately. All Modals, as illustrated in Figure 11, Controllers and Routing Configuration (AreaRegistration.cs) in a specific Area in MVC project will build in to one ".dll" file and deployed under main project bin folder. All the Views and other contents will have deployed as files in to the server.

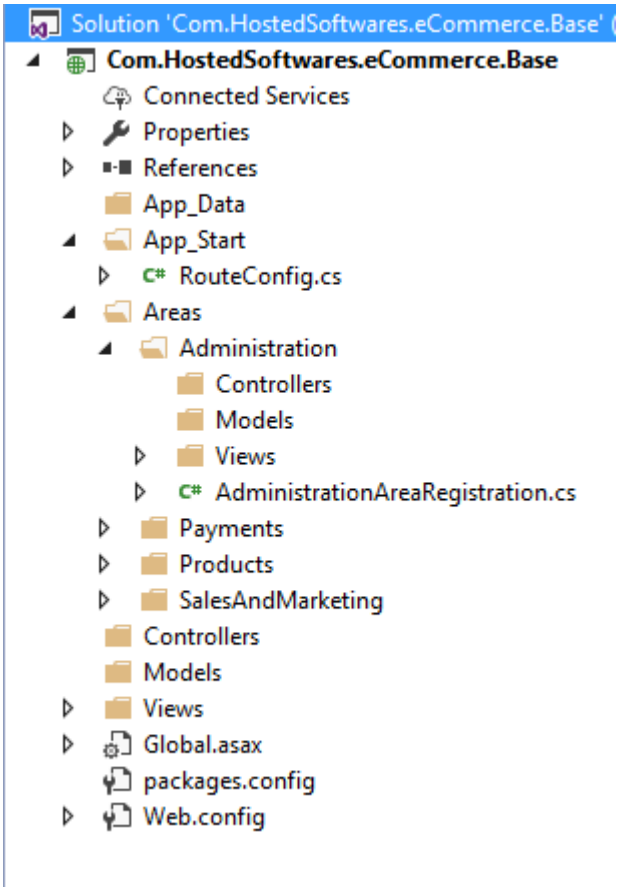


Figure 11 Typical MVC Projects with Multiple Areas

This typical MVC project with Areas needs to develop all the modules at once or it needs to develop new module again in the same project, deploy a full publish in to the server again and re-test total application because of this full deployment. Also, this architecture is not supporting to run and test Areas separately. Because of this, typical MVC project architecture wills not suite for module-based development and deployment. But this area concept can be used as a starting point to develop module-based web component development.

4.4.2 Proposed System Design Architecture

According to Module based software development (IEEE), software component can be developed separately, run separately, test separately and plug and unplug from a base module in minimum steps. Writer tries to develop architecture with above features using ASP.Net MVC to overcome the problems in 4.1.

In proposed architecture main project and modules will develop as separate MVC projects (Figure 12), run and test separately, publish separately and zip the project and upload it to main project. Codes which need to run that module will deploy as a ".dll" with this publish.

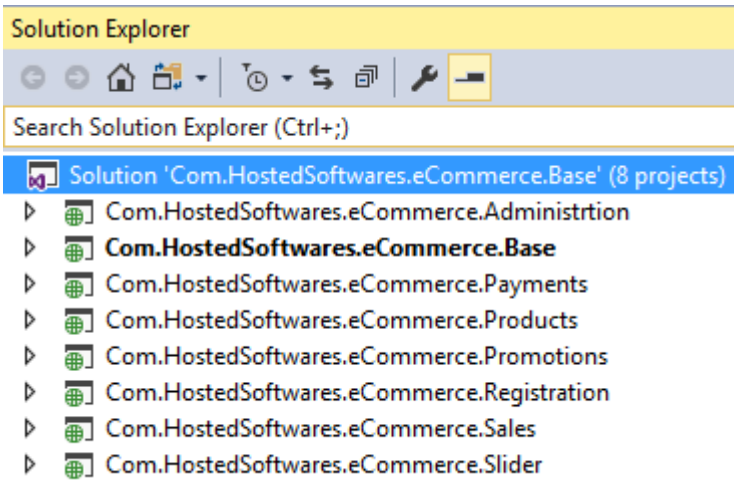


Figure 12 Modules will build as separate MVC Projects

There will be a module manager in main project and it will unzip the uploaded module publish in to main project Area folder and place the ".dll" file to main project bin folder. All other relevant files and contents will use same process in new module file placement. Also, this module manager will help to remove (plug-out) as component easily.

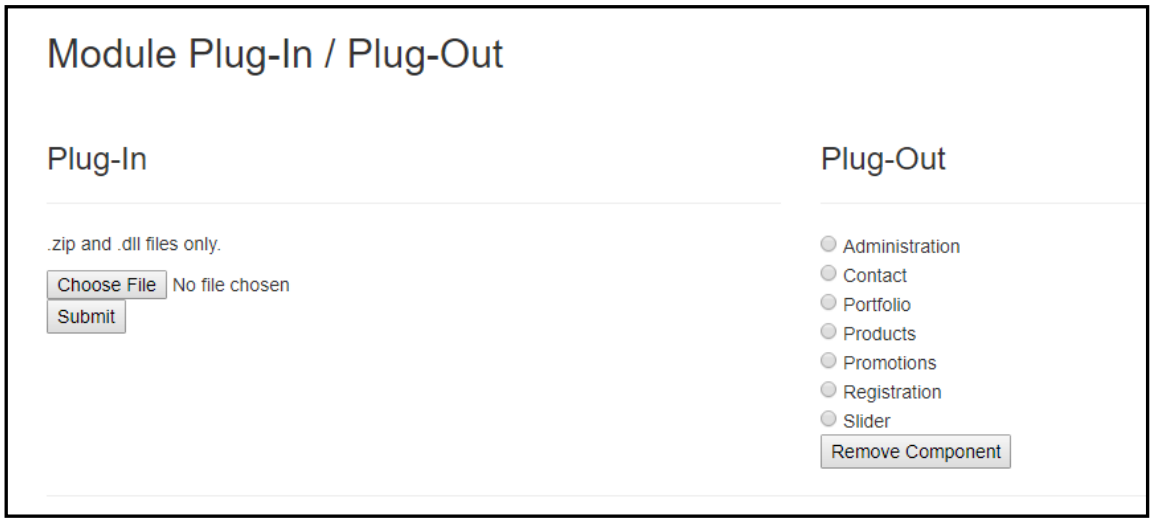


Figure 13 Module Manager Screen

After this process this module will automatically visible in the base project in relevant section. To do this we have define sections in the main project. A separate module can be plug as complete new page or else as a new section in in a specific page.

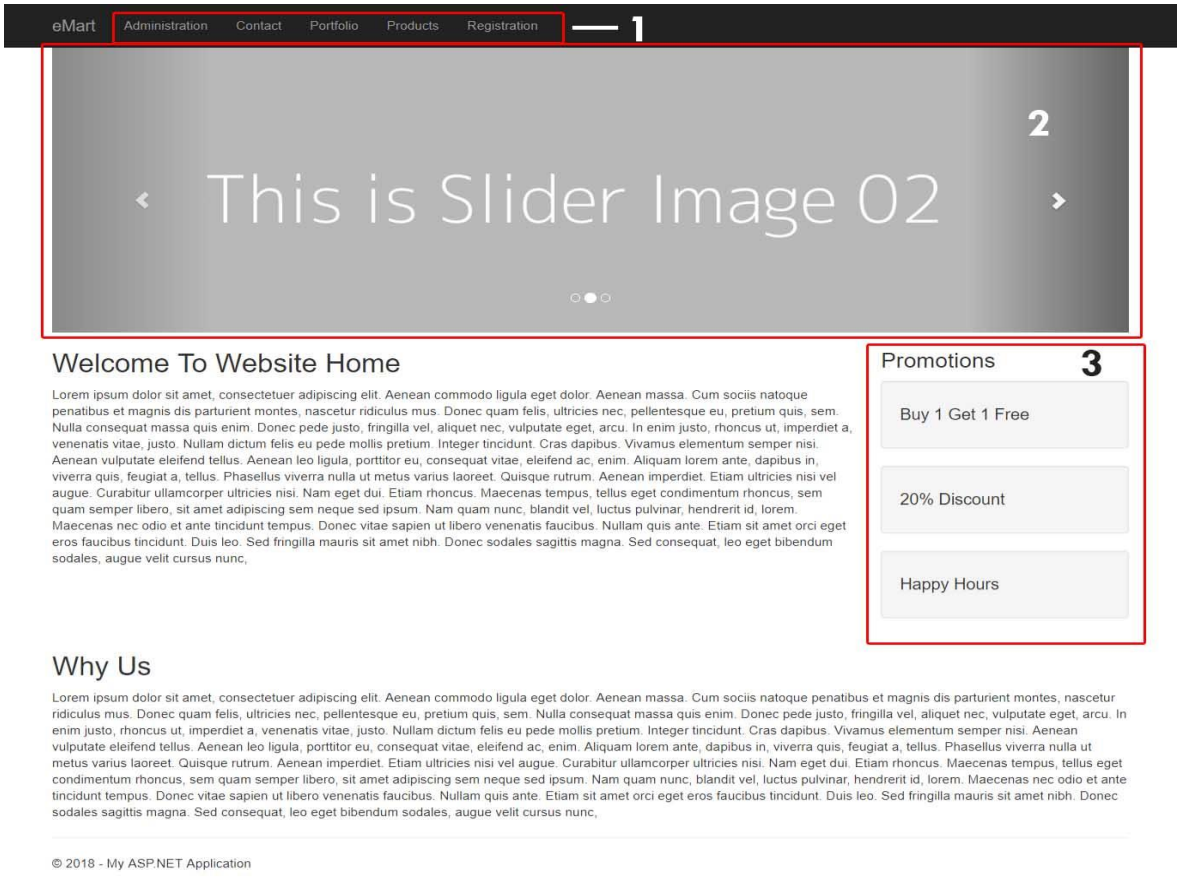


Figure 14 Modules after plugged in to base project

1. – As a complete new page in main menu
2. – As a section in an existing page
3. – As a sub section in an existing section

4.5 Proposed System Screen Designs

4.5.1 Home Page

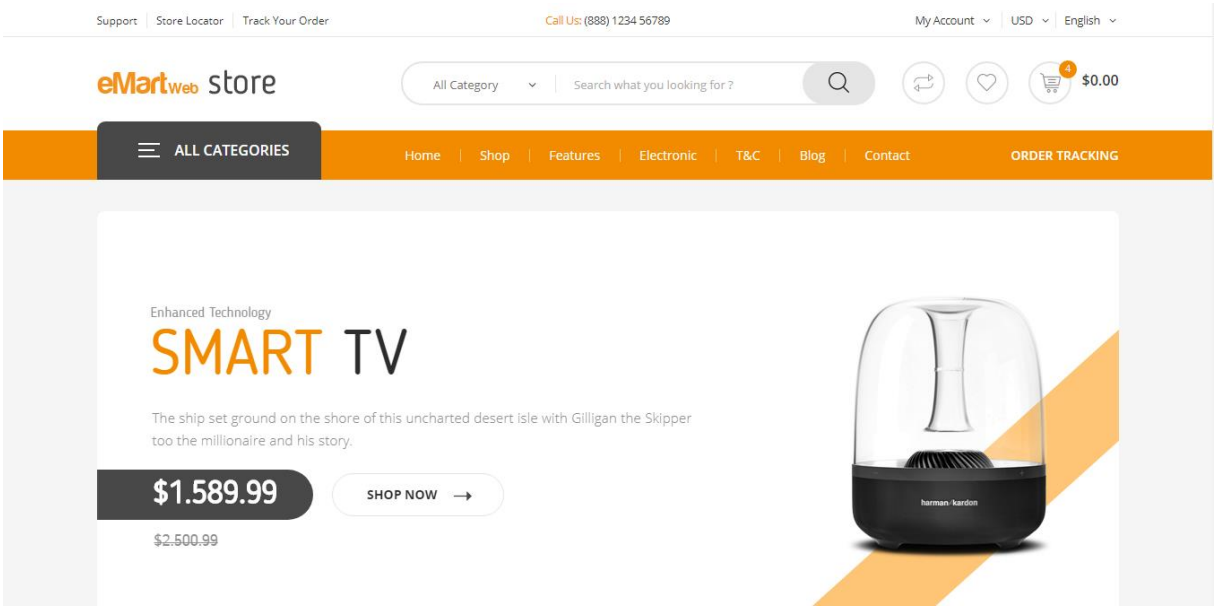


Figure 15 Home Page

4.5.2 Promotion Area

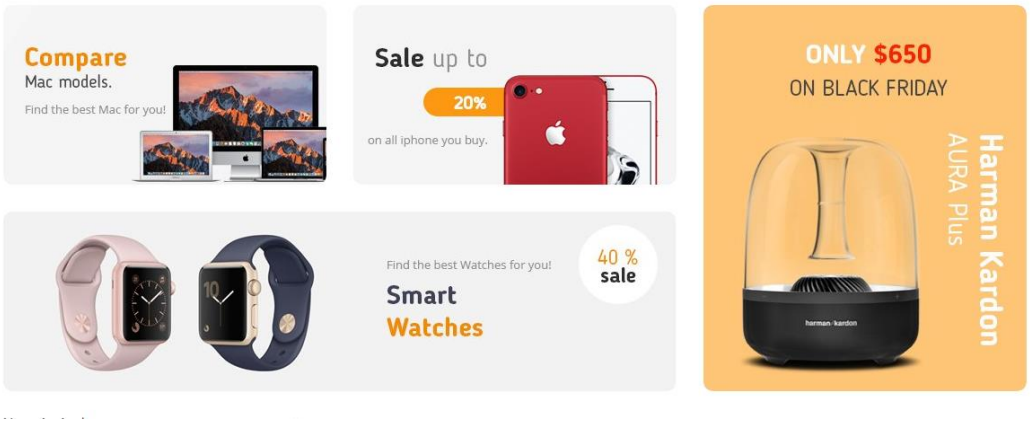


Figure 16 Promotion Area

4.5.3 Store Front Area

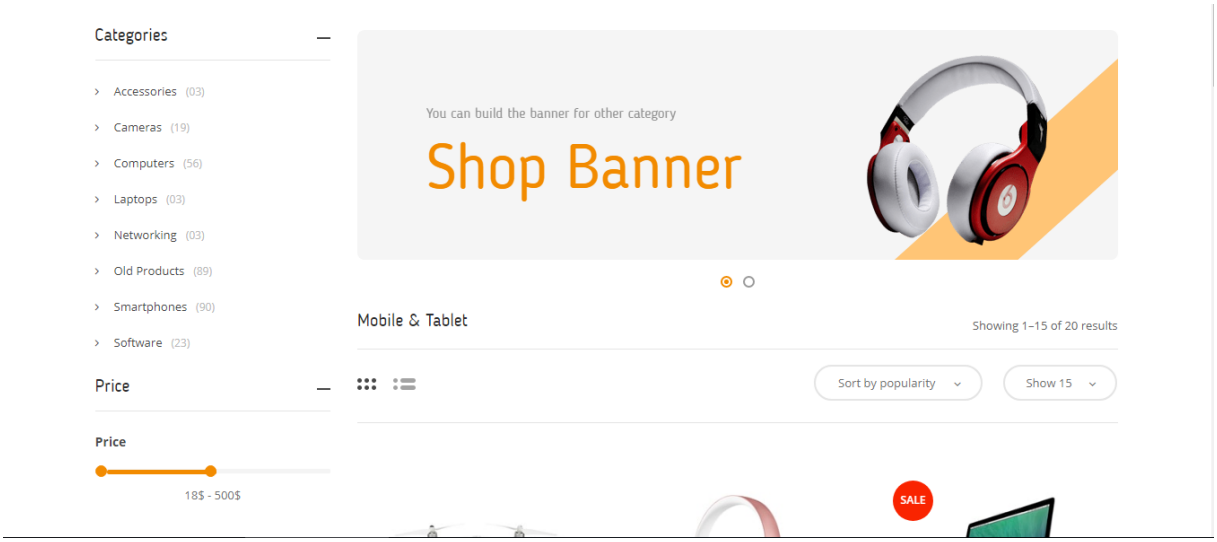


Figure 17 Store Front Area

4.5.4 Top Rated Items

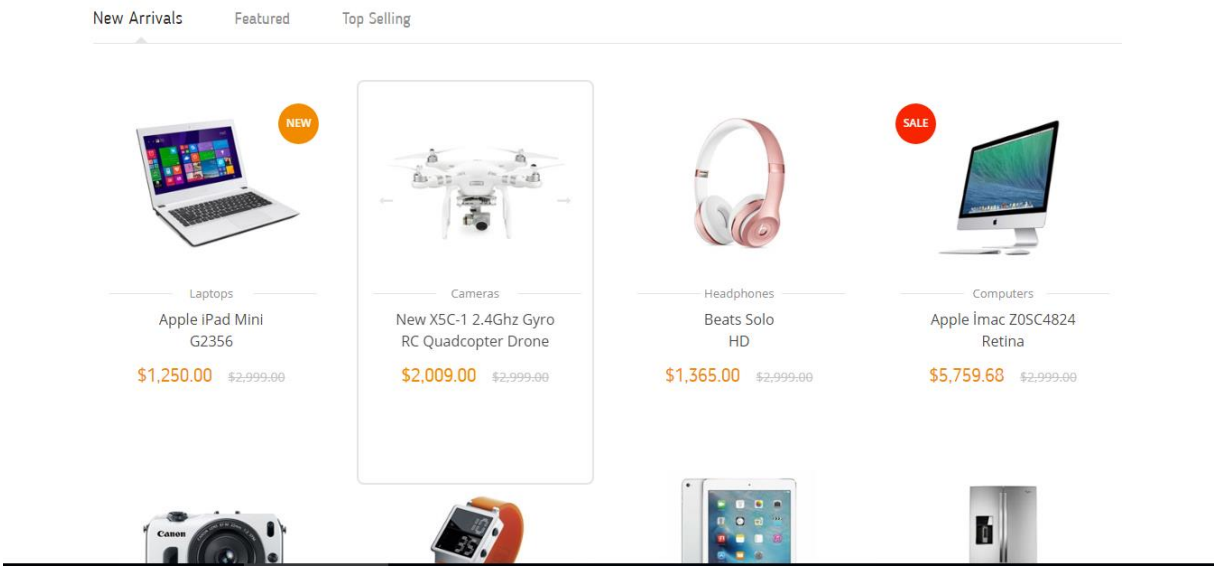


Figure 18 Top Rated Items

4.5.5 Shopping Cart

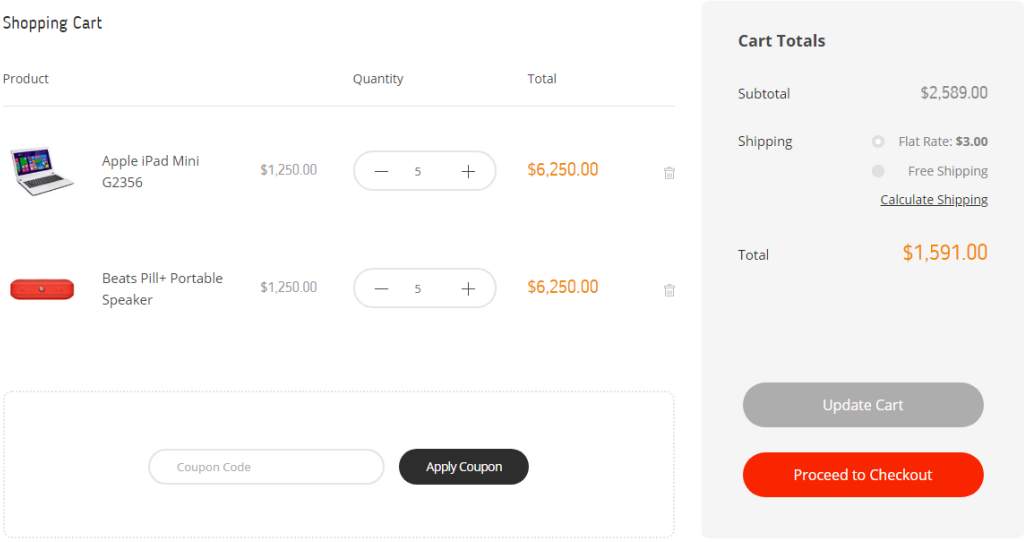


Figure 19 Shopping Cart

4.5.6 Checkout

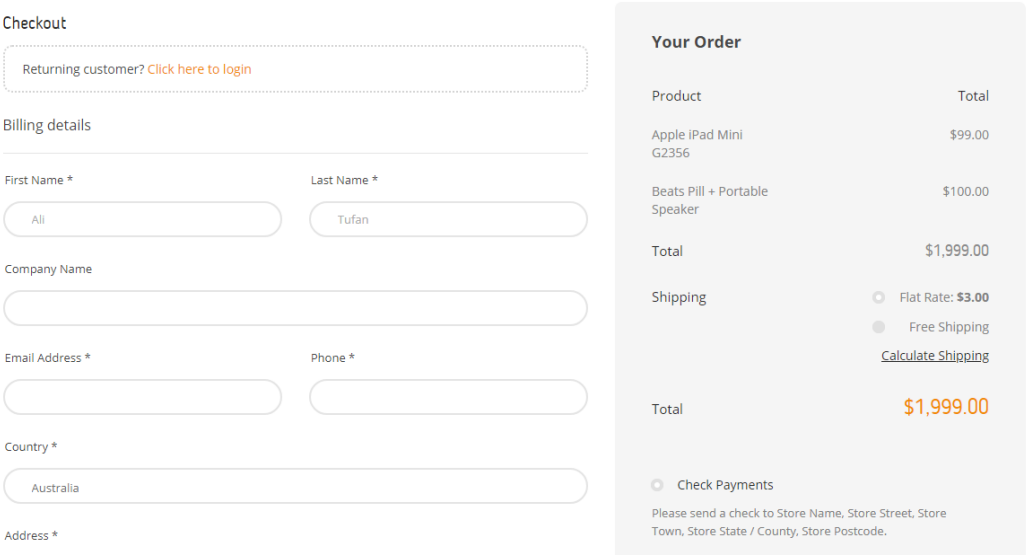


Figure 20 Checkout

4.5.7 My Account Area

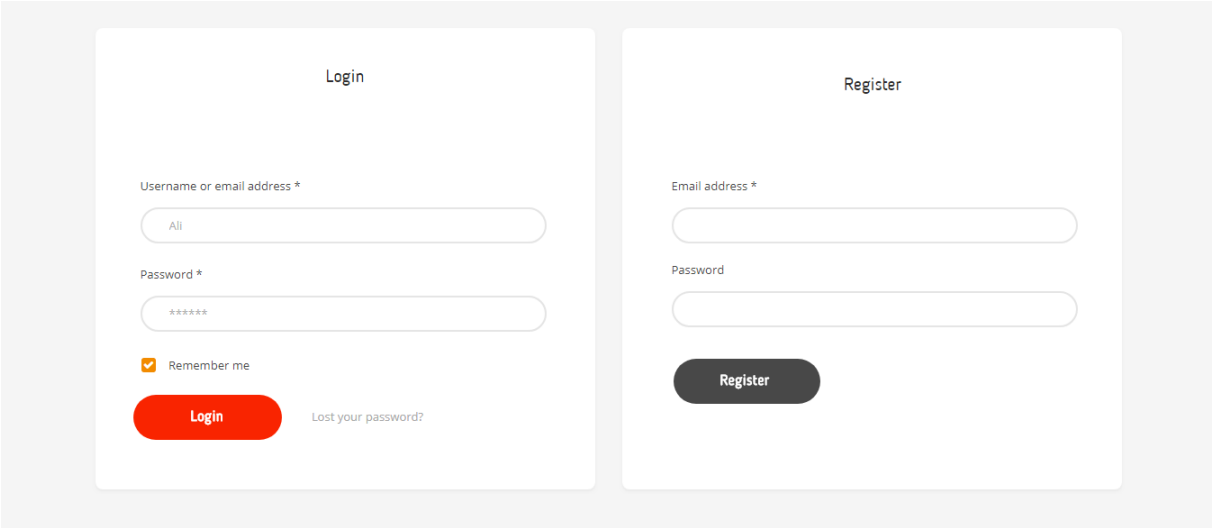


Figure 21 My Account Area

4.5.8 Admin Panel - Categories

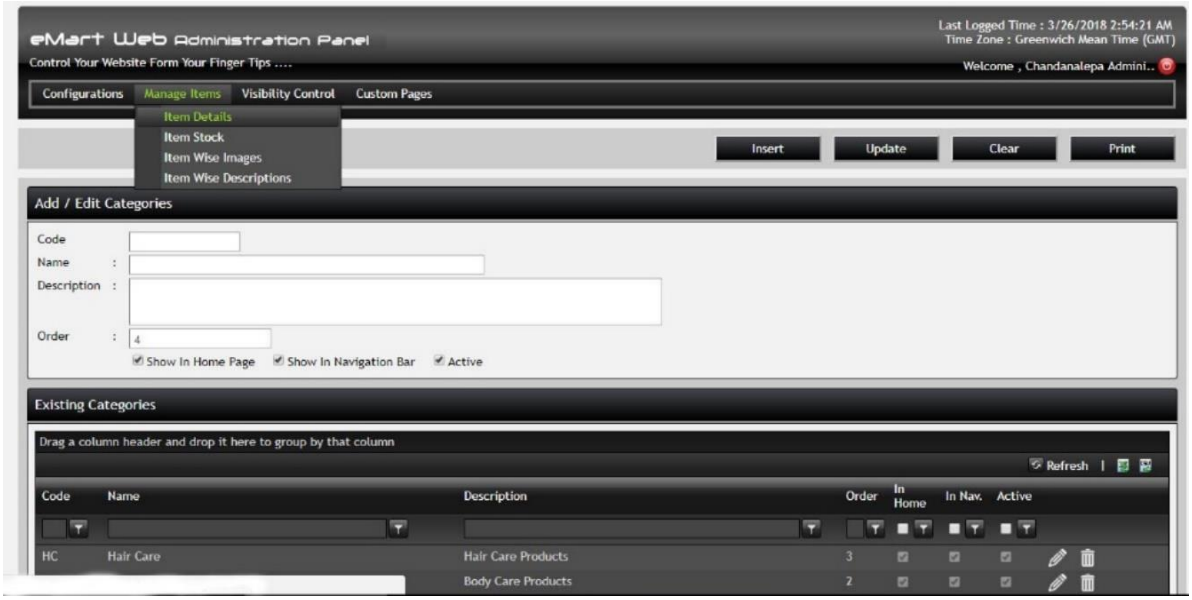


Figure 22 Admin Panel - Categories

4.5.9 Admin Panel – Add / Edit Items

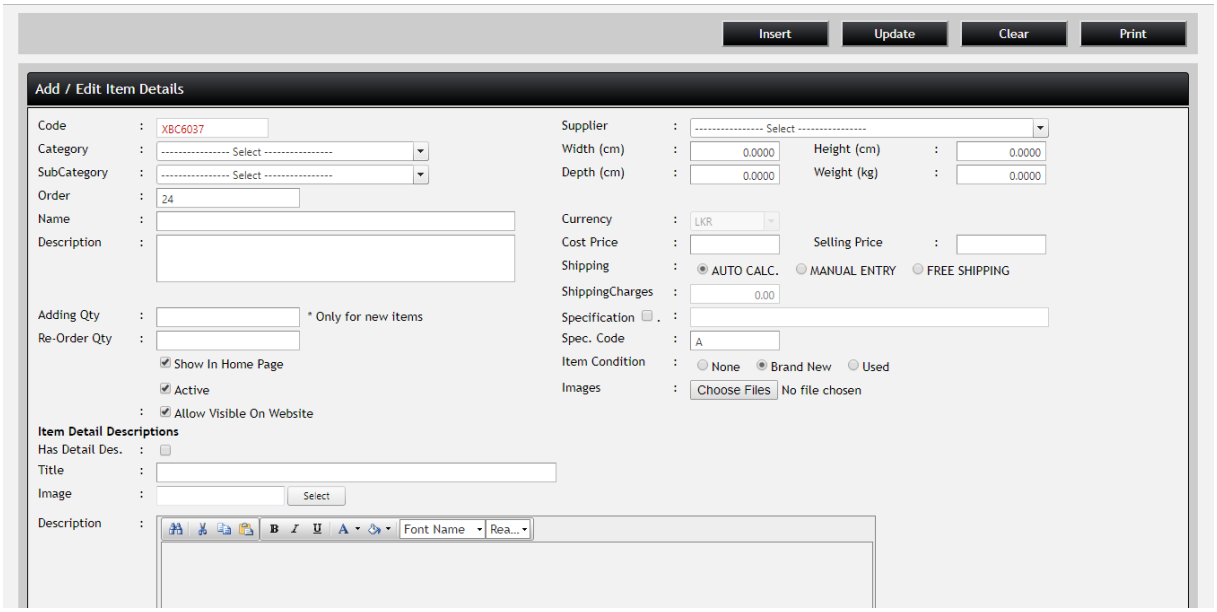


Figure 23 Admin Panel – Add / Edit Items

Chapter 5 : Evaluation and Results

5.1 Evaluation Plan

Evaluation Plan is all about testing / evaluating functional as well as non-functional requirements of the research. Testing require to be carried out for quantitative and qualitative aspects of each functional and non-functional requirements.

- Purpose of preparing an Evaluation Plan
  - Validate the appropriateness of a software product.
  - Help the people outside the test group to understand „why“ and „how“ of product validation.

- Scope
  - Every area of the system testing was done by the QA team. - Specify the areas which are out of scope (screens, database, mainframe processes etc).

- Evaluation Approach
  - Include Details on how the testing is to be performed.
  - Include specific strategy is to be followed for testing.

5.2 Evaluation Approach

I have chosen Opinion & Interview based evaluation approach to evaluate my research project.

Evaluation is a systematic process to understand what a program does and how well the program does it. Evaluation results can be used to maintain or improve program quality and to ensure that future planning can be more evidence-based. Evaluation constitutes part of an ongoing cycle of program planning, implementation, and improvement.

Evaluation falls into one of two broad categories: formative and summative. Formative evaluations are conducted during program development and implementation and are useful if we want direction on how to best achieve our goals or improve our program. Summative evaluations should be completed once our programs are well established and will tell us to what extent the program is achieving its goals.

The purpose of the project evaluation is to assess the software development methodology that was used throughout the development of the framework, assess the usefulness of the technologies and tools, the accuracy of the estimations and the usefulness of the reviews. The solution will be reviewed and evaluated to decide whether it accomplishes the ideas processed in the initial overview and for the quality of the project.

5.3 Evaluation Methodologies

Choosing an appropriate evaluation methodology is a key factor when conducting proper evaluation. The objective of this project is aimed to develop a common framework which follow component based loosely coupled but highly integrated software component framework for e-commerce application development. Also, this research will search for suitable technologies to follow 5 for the CBSD in web-based e-commerce solutions in UI, Backend and Database level. Therefore, conducting a general user evaluation would be of most importance to ensure that the product is accepted by the user. Used several methodologies evaluate. These are the most importance once.

- Demonstration

  Demonstration is one of main evaluation method of this research.After the demo has been completed, our job isn't finished. Each person from our software selection team attending the on-site demo should now participate in scoring. This will give you a quantitative method of evaluating the software. Determining an appropriate scoring or rating system and breaking down the demonstration into different sections (each of which represents a percentage of the overall evaluation score) will facilitate the scoring process.

  - Functionality or Performance
  - Ease of use
  - Process and flow
  - Flexibility

  These are the main areas to focus on during the demo, and it's important that each of these is covered to ensure potential solutions meet our company's specific needs.

- Interview

  Interviews were held with domain expert, developers and clients in a formal manner to evaluate the system.

- Questionnaire

  A questionnaire was circulated among member of target audience along with using the system the member could fill in a feedback from and give in their evaluation of the system. Used for mention methodologies to evaluate the system using expert as well normal users.

5.4 Results

Based on company opinions and practical experimentation, a web application evaluation makes it possible to check whether the project objectives were satisfied. I will collect feedback after deployed final Project to our company. At the early stage I have deploy prototype in our company and allow to access that system to Developers and Implementation Teams. After getting an idea about my proposed system, I have deployed another Prototype based on feedbacks and evaluate again. Then based on the evaluation results, we will decide to change our existing E-commerce applications on client companies with my New Research Project. I will collect feedbacks using verbal interview, written interview and observations. Evaluation Results can be measure in several areas. Such as Appearance, Usability, Functionality, Performance, simplicity etc...

Proposed system contains Main Modules and Sub Module. These modules will be developed as separate independent software components such as Main Component and Sub Component which will able to plug into the base module. When we consider about single component, we can do static testing or dynamic testing without any doubt. Web component-based E-Commerce Application Framework has lot of individual components which are connect to base web application. At the initial stage, I have done unit testing for each component. Then time to time have done Integration Testing, System Testing and Acceptance Testing.

5.4.1 Questionnaire Evaluation

	Questionnaire Evaluation - Hosted Software Solutions (Pvt)										
Participant	**Designation**	**Department you work for**	**Highest Educational Level**	**Age Category**	**Experience in working Hosted Software Solutions (Pvt)**	**System Evaluation**					
						Ability to add new features to the existing system immediately	**Ability to remove unnecessary features from the existing system immediately**	**Add or Remove features to existing system without system down**	**Ability to add new features to the proposed system immediately**	**Ability to remove unnecessary features from the proposed system immediately**	**Easiness of users in using proposed application**
1	**Software Engineer /Senior Software Engineer**	Software Department	Master/PhD	18-27 years	10< years	[2=Poor]	[1=Very Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[5=Excellent]
2	**Implementation Engineer / Senior Implementation Engineer**	Implementation Department	Degree	18-27 years	3-5 years	[1=Very Poor]	[2=Poor]	[2=Poor]	[3=Satisfactory]	[4=Good]	[4=Good]
3	**Implementation Engineer / Senior Implementation Engineer**	Implementation Department	Diploma/ Higher Diploma	28-37 years	1> years	[2=Poor]	[2=Poor]	[2=Poor]	[4=Good]	[4=Good]	[3=Satisfactory]
4	**QA Engineer / Senior QA Engineer**	QA Department	Diploma/ Higher Diploma	18-27 years	3-5 years	[1=Very Poor]	[2=Poor]	[2=Poor]	[4=Good]	[5=Excellent]	[5=Excellent]
5	**QA Engineer / Senior QA Engineer**	QA Department	Degree	28-37 years	3-5 years	[1=Very Poor]	[2=Poor]	[2=Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
6	**Software Engineer /Senior Software Engineer**	Software Department	Degree	18-27 years	3-5 years	[3=Satisfactory]	[2=Poor]	[1=Very Poor]	[5=Excellent]	[4=Good]	[5=Excellent]

7	**Software Engineer /Senior Software Engineer**	Software Departm ent	Degree	18-27 years	3-5 years	[2=Poor]	[1=Very Poor]	[2=Po or]	[3=Satisf actory]	[3=Satisf actory]	[5=Excell ent]
8	**Software Engineer /Senior Software Engineer**	Software Departm ent	Degree	18-27 years	3-5 years	[3=Satisf actory]	[2=Poor]	[2=Po or]	[5=Excell ent]	[4=Good]	[5=Excell ent]
9	**Impleme ntation Engineer / Senior Impleme ntation Engineer**	Impleme ntation Departm ent	Degree	18-27 years	3-5 years	[2=Poor]	[1=Very Poor]	[2=Po or]	[5=Excell ent]	[4=Good]	[5=Excell ent]
10	**Software Engineer /Senior Software Engineer**	Software Departm ent	Master/P hD	18-27 years	3-5 years	[2=Poor]	[1=Very Poor]	[2=Po or]	[5=Excell ent]	[5=Excell ent]	[5=Excell ent]
11	**QA Engineer / Senior QA Engineer**	QA Departm ent	Master/P hD	18-27 years	1-2 years	[2=Poor]	[3=Satisf actory]	[2=Po or]	[5=Excell ent]	[5=Excell ent]	[4=Good]
12	**Software Engineer /Senior Software Engineer**	Software Departm ent	Degree	18-27 years	1-2 years	[2=Poor]	[2=Poor]	[1=V ery Poor]	[5=Excell ent]	[5=Excell ent]	[4=Good]
13	**Software Engineer /Senior Software Engineer**	Software Departm ent	Degree	18-27 years	3-5 years	[2=Poor]	[2=Poor]	[2=Po or]	[5=Excell ent]	[5=Excell ent]	[3=Satisf actory]
14	**Software Engineer /Senior Software Engineer**	Software Departm ent	Degree	18-27 years	3-5 years	[1=Very Poor]	[3=Satisf actory]	[2=Po or]	[3=Satisf actory]	[5=Excell ent]	[3=Satisf actory]
15	**Software Engineer /Senior Software Engineer**	Software Departm ent	Degree	18-27 years	1> years	[2=Poor]	[3=Satisf actory]	[2=Po or]	[5=Excell ent]	[5=Excell ent]	[4=Good]
16	**Software Engineer /Senior Software Engineer**	Software Departm ent	Degree	18-27 years	3-5 years	[2=Poor]	[2=Poor]	[2=Po or]	[5=Excell ent]	[5=Excell ent]	[4=Good]
17	**Impleme ntation Engineer / Senior Impleme ntation Engineer**	Impleme ntation Departm ent	Degree	18-27 years	10< years	[1=Very Poor]	[3=Satisf actory]	[2=Po or]	[5=Excell ent]	[5=Excell ent]	[4=Good]

18	**Software Engineer /Senior Software Engineer**	Software Department	Degree	18-27 years	3-5 years	[2=Poor]	[2=Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[4=Good]
19	**Software Engineer /Senior Software Engineer**	Software Department	Degree	18-27 years	3-5 years	[1=Very Poor]	[2=Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[5=Excellent]
20	**Implementation Engineer / Senior Implementation Engineer**	Implementation Department	Degree	18-27 years	1-2 years	[1=Very Poor]	[1=Very Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[4=Good]
21	**Software Engineer /Senior Software Engineer**	Software Department	Degree	28-37 years	3-5 years	[2=Poor]	[2=Poor]	[2=Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
22	**Software Engineer /Senior Software Engineer**	Software Department	Degree	18-27 years	3-5 years	[2=Poor]	[2=Poor]	[1=Very Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
23	**Software Engineer /Senior Software Engineer**	Software Department	Degree	28-37 years	3-5 years	[2=Poor]	[1=Very Poor]	[2=Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
24	**Software Engineer /Senior Software Engineer**	Software Department	Diploma/ Higher Diploma	28-37 years	3-5 years	[2=Poor]	[2=Poor]	[2=Poor]	[4=Good]	[4=Good]	[4=Good]
25	**Implementation Engineer / Senior Implementation Engineer**	Implementation Department	Degree	18-27 years	3-5 years	[1=Very Poor]	[3=Satisfactory]	[2=Poor]	[3=Satisfactory]	[3=Satisfactory]	[5=Excellent]

Table 7 Questionnaire Evaluation - Hosted Software

5.4.2 Analysis of Designation

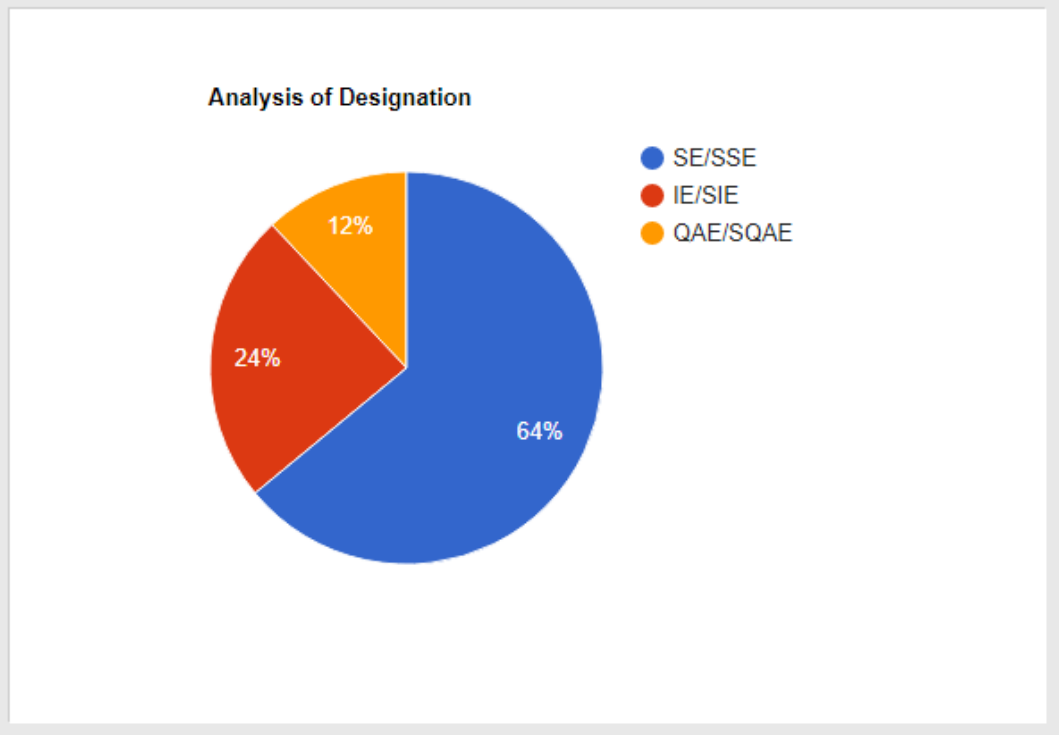


Figure 24 Analysis of Designation

5.4.3 Analysis of Higher Educational Level

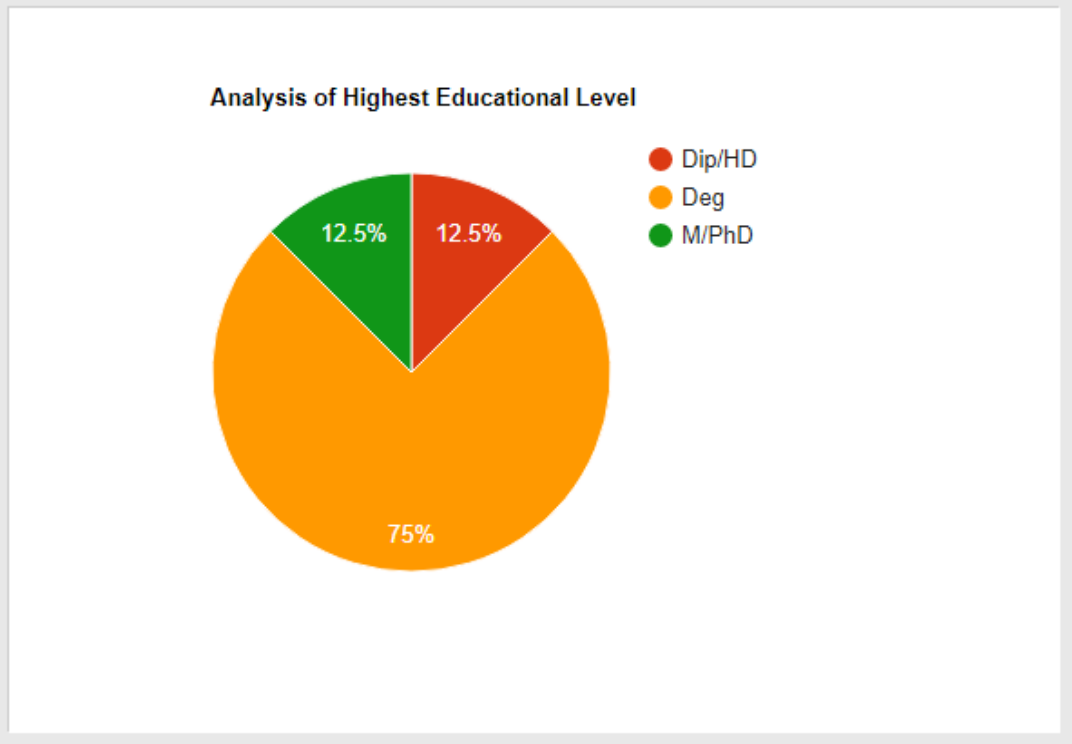


Figure 25 Analysis of Higher Educational Level

5.4.4 Analysis of Age Category

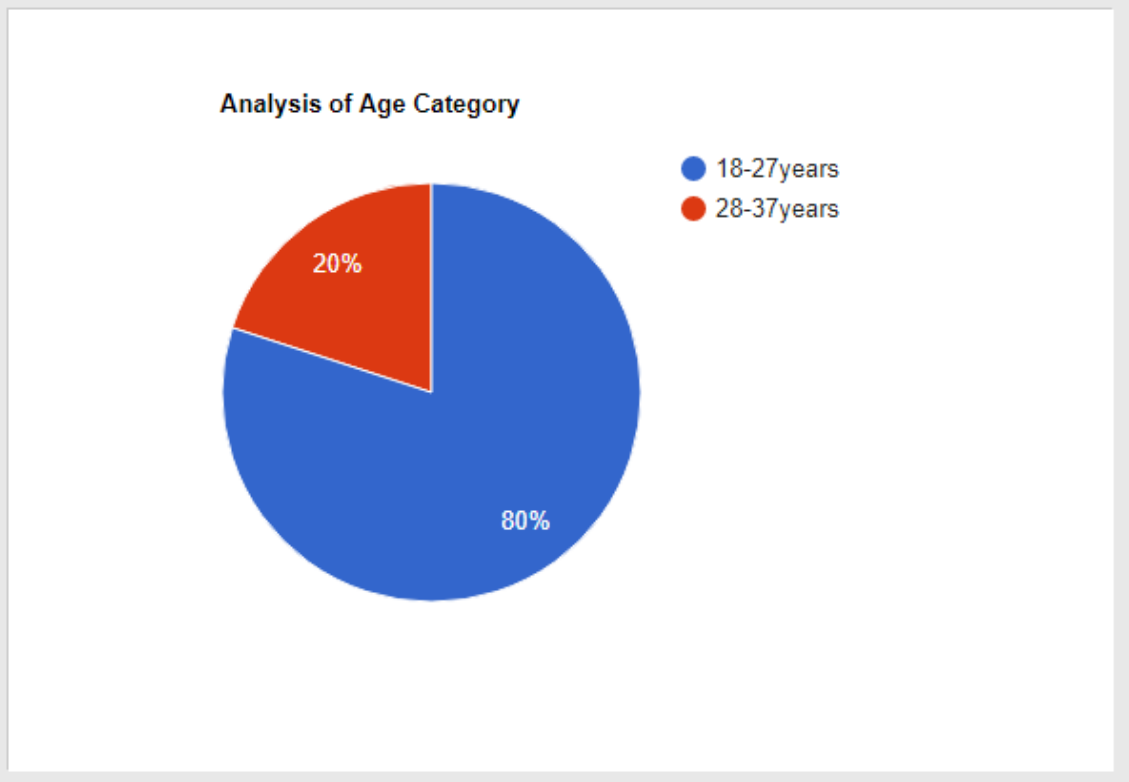


Figure 26 Analysis of Age Category

5.4.5 Analysis of Experience in working Hosted Software Solutions (Pvt)

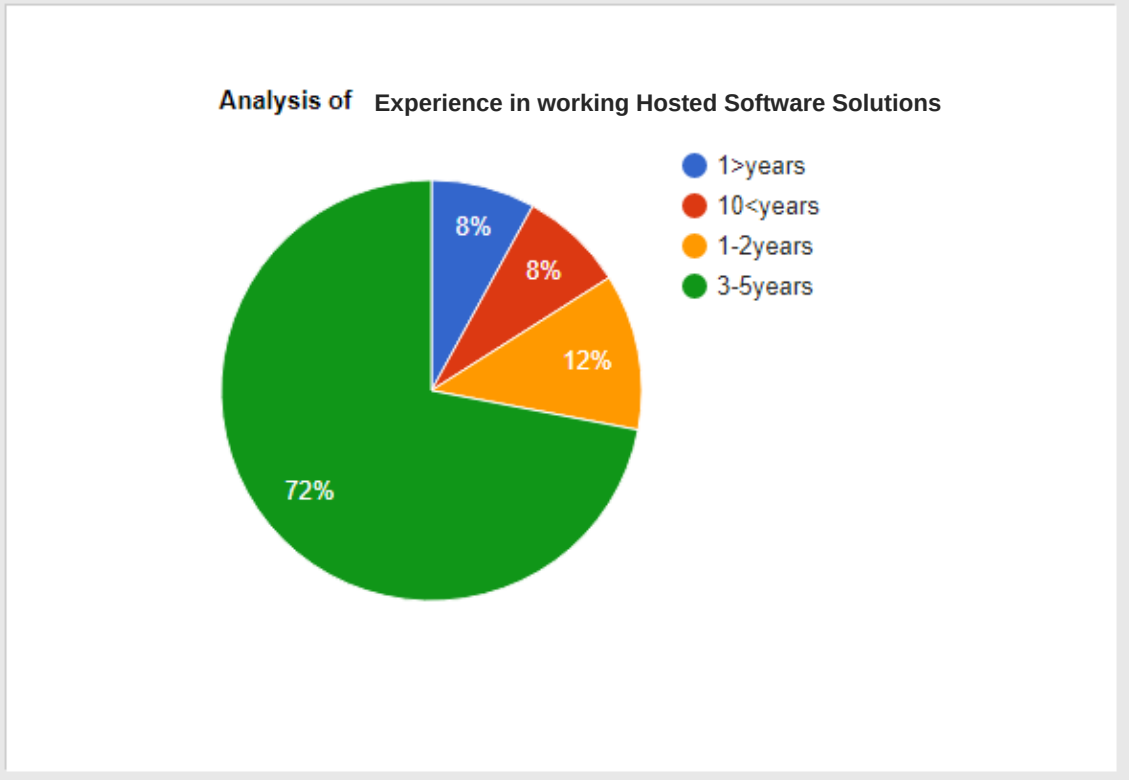


Figure 27 Analysis of Experience in working Hosted Software Solutions (Pvt)

5.4.6 System Evaluation

A - Ability to add new features to the existing system immediately

B - Ability to remove unnecessary features from the existing system immediately

C - Add or Remove features to existing system without system down

D - Ability to add new features to the proposed system immediately

E - Ability to remove unnecessary features from the proposed system immediately

F - Easiness of users in using proposed application

1 - [1=Very Poor]

2 - [2=Poor]

3 - [3=Satisfactory]

4 - [4=Good]

5 - [5=Excellent]

No	**A**	**B**	**C**	**D**	**E**	**F**
1	2	1	2	5	5	5
2	1	2	2	3	4	4
3	2	2	2	4	4	3
4	1	2	2	4	5	5
5	1	2	2	5	4	5
6	3	2	1	5	4	5
7	2	1	2	3	3	5
8	3	2	2	5	4	5
9	2	1	2	5	4	5
10	2	1	2	5	5	5
11	2	3	2	5	5	4
12	2	2	1	5	5	4
13	2	2	2	5	5	3
14	1	3	2	3	5	3
15	2	3	2	5	5	4
16	2	2	2	5	5	4

17	1	3	2	5	5	4
18	2	2	2	5	5	4
19	1	2	2	5	5	5
20	1	1	2	5	5	4
21	2	2	2	5	4	5
22	2	2	1	5	4	5
23	2	1	2	5	4	5
24	2	2	2	4	4	4
25	1	3	2	3	3	5

Table 8 System Evaluation Data Sample

Based on the evaluated data sample, system evaluation diagram can be generated as Table 9.

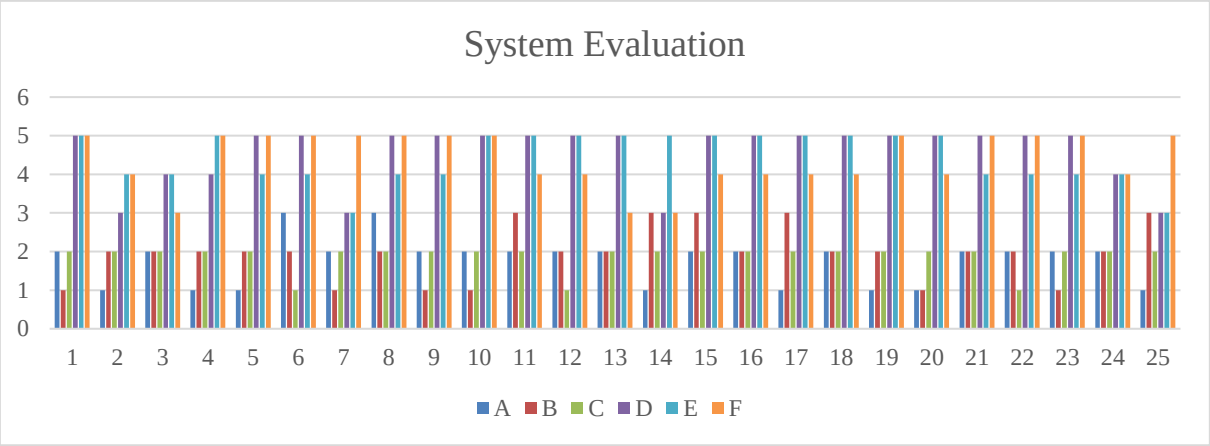


Table 9 System Evaluation Diagram

Chapter 6 : Conclusion

6.1 Introduction

Finally, I deliver complete web component-based E-Commerce Application Framework with Component should be able to plug and play with other components and/or frameworks so that component can be composed at run-time without compilation. Special thing is when Web Application Live Run/Host, we want to add new module to existing web. There is no any issue with this solution. Only we want to upload relevant module to existing base web application. No need to down the system according to add new features.

6.2 Findings and Limitations

Finally, the answer to the overall research questions. How much of the current web application development frameworks explicitly refer to application of component-based approach is hardly intuitive.

1) In which way is CBD used for web application development? - There are three main approaches:
   a) component approach is used for creating component-based frameworks which are then used for creating web application (not necessarily component oriented).
   b) component approach is used for building components which are the building blocks of web applications.
   c) a mix of two previous approaches.
   In approach a) and c) the end user decides whether to use component approach for web application development while in b) component approach is imposed to the end users.

2) What is the relation between CBD and web application development? – Component approach is used mostly for server-side applications. Using it on the client side is less common, but there are cases and end-users aren't constrained to use it. Most widely used architecture is n-tired with components used inside different layers. For any future researchers and practitioners, it is strongly suggested to plan component approach right from the start of the application design process. Although it requires more time, true benefits (separation of concerns, better maintainability, scalability, replaceability, single point of edit, etc.) are apparent later.

3) Which component models are used for web application development? –EJB and Java beans are most preferable component models rather than ASP.NET. But ASP.NET MVC the most popular programming language for this purpose. Although, it should be noted that there are a lot of custom models also. Offers new possibilities independent of a single technology.

4) In which web application development domains is CBD used? – It is hard to recognize distinct domains however there are two types of web application development frameworks presented in the selected publications:
   a) general; used for any kind of web applications and
   b) specific; for developing special purpose web applications (e.g. eCommerce, eLearning, etc.).

6.3 Future Work

Future work will concentrate on improving the performance of the existing prototype. The evaluation results provided some useful information to help us to find out how this approach works and how this approach fine tune.

If one is interested component approach and web development frameworks the most relevant scientific databases are Scopus, Springer and IEEE which will cover most of the related publications. Currently, the most relevant publications (2/3 of them being conference proceedings) were published between 2005 and 2006 which is most likely due the popularization of Web 2.0.

It is apparent that component-based approach is becoming a serious architectural direction and there is a very recent working group focused solely on component-based development for web, including the one from the W3C.

Existing prototype component plug-in to predefined and specific location or plug-out from predefined and specific location. There will be a proper method to define a component with location in the Main web application. In future, Module Manager Screen modified to define location and map existing locations of the components using simple way. Component plug-in and plug-out with locations define option will be the best solution rather than current prototype.

References

[1] Wikipedia, "Senaration Of Concerns," 14 07 2017. [Online]. Available: https://en.wikipedia.org/wiki/Separation_of_concerns. [Accessed 25 07 2017].

[2] Wikipedia, "Event-driven architecture," 02 02 2018. [Online]. Available: https://en.wikipedia.org/wiki/Event-driven_architecture. [Accessed 04 02 2018].

[3] IEEE, "Edsger W. Dijkstra," [Online]. Available: https://www.computer.org/web/awards/goode-edsger-dijkstra. [Accessed 25 07 2017].

[4] M. T. G. G. a. P. B. M. Vaitis, "Structural engineering:Processes and tools for developing component-based open hypermedia systems," *Lecture Notes in Computer Science,* vol. 3511, p. 113–128, 2005.

[5] R. W. M. L. a. H. M. M. Jahn, "Composing user-specific web applications from distributed plug-ins," *Comput. Sci. - Res. Dev,* vol. 28, p. 85–105, 2013.

[6] M. Y. K. E. S. a. S. W. Q. Li, "XVM: A bridge between XML data and its," *Thirteenth International World Wide Web Conference Proceedings, WWW2004,* p. 155–163, 2004.

[7] B. D. S. a. A. G. J. A. Ginige, "Towards end user development of Web applications for SMEs: A component based approach," *Lecture Notes in Computer Science, 2005,* vol. 3579, p. 489–499, 2005.

[8] V. L. a. S. Ilieva, "Towards development and use of in-house component framework: Results and expectations in Software Engineering and Advanced Applications," in *31st EUROMICRO Conference, 2005*, 2005.

[9] J. L. a. T. Chusho, "A web application framework for end-user-initiative development with," *Lecture Notes in Engineering and Computer Science, 2012,* vol. 1, p. 816–822, 2012.

[10] S. P. a. M. Žagar, "Nested web application components framework: A comparison to competing software component models," in *ENASE 2013 - Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering*,

2013.

[11] t. f. e. Wikipedia, "Component-based software engineering," Wikipedia, 18 02 2018. [Online]. Available: https://en.wikipedia.org/wiki/Component-based_software_engineering. [Accessed 02 02 2018].

[12] W. J. Lloyd, "A Common Criteria Based Approach for COTS Component," *Journal of Object Technology*, vol. 4, no. 2004, 2005.

[13] Syed Ahsan Fahmi, Ho-Jin Choi, "A Study on Software Component Selection," in *ICACT 2009*, 2009.

[14] A. S. J. L. Z. M. Bichler, "Component-based e-commerce: assessment of current practices and future directions," *ACM Digital Library*, vol. 27, pp. 7-14, 1998.

[15] T. C. J. Li, "A Web Application Framework for End-User-Initiative Development with a Visual Tool," *Research Gate*, 2012.

[16] I. Š. M. Novak, CURRENT USAGE OF COMPONENT BASED PRINCIPLES FOR DEVELOPING WEB APPLICATIONS WITH FRAMEWORKS, Faculty of Organization and Informatics, University of Zagreb, 2016.

[17] R. S. R. R. J. Wang, "Shared Content Management in Replicated Web Systems: A Design Framework Using Problem Decomposition, Controlled Simulation, and Feedback Learning," *IEEE*, vol. 38, p. 1, 2008.

[18] V. Galant, "Blending E-Commerce Theory and Application," *IEEE Distributed Systems Online*, vol. 6, p. 5, 2005.

[19] A. T. V. K. D. Llambiri, "Efficiently distributing component-based applications across wide-area environments," in *23rd International Conference*, Rhode Island, USA, 2003.

[20] M. S. JoachimPerchat, "Component based Framework to Create Mobile Cross-platform Applications," *Procedia Computer Science*, vol. 9, pp. 1004- 1011, 2013.

[21] S. J. D. R. Barrett, openMVC: A Non-proprietary Component-based, New York: Dublin Institute of Technology, 2004.

[22] T. Myerson, "Hello World: Windows 10 Available on July 29," Microsoft, 01 06 2015. [Online]. Available: https://blogs.windows.com/windowsexperience/2015/06/01/hello-world-windows-10-available-on-july-29/. [Accessed 02 02 2018].

[23] R. L. [MSFT], "Announcing .NET Framework 4.6," Microsoft, 20 07 2015. [Online]. Available: https://blogs.msdn.microsoft.com/dotnet/2015/07/20/announcing-net-framework-4-6/Announcing .NET Framework 4.6. [Accessed 31 01 2018].

[24] Wikipedia, "Advanced Vector Extensions," 30 01 2018. [Online]. Available: https://en.wikipedia.org/wiki/Advanced_Vector_Extensions. [Accessed 31 01 2018].

Appendices

Appendix A: Questionnaire

- Questionnaire

Questionnaire - Hosted Software Solutions (Pvt)

Dear Staff Members, this Questionnaire is designed to collect information on perception of yours about eCommerce existing and Proposed Applications in your company.All the information you give will be used only for academic purpose, and not for any other purpose.Please be sincere with your responses and be kind enough to answer all the questions in this questionnaire.

* Required

Designation *

Your answer

Department you work for *

Your answer

Highest Educational Level *

- ○ Advanced Level
- ○ Diploma/Higher Diploma
- ○ Degree
- ○ Master/PhD

Age Category *

- ○ 18-27 years
- ○ 28-37 years
- ○ 38-47 years
- ○ 48-57 years
- ○ More than 57 years

Experience in working Hosted Software Solutions (Pvt) *

- ○ 1> years
- ○ 1-2 years
- ○ 3-5 years
- ○ 10< years

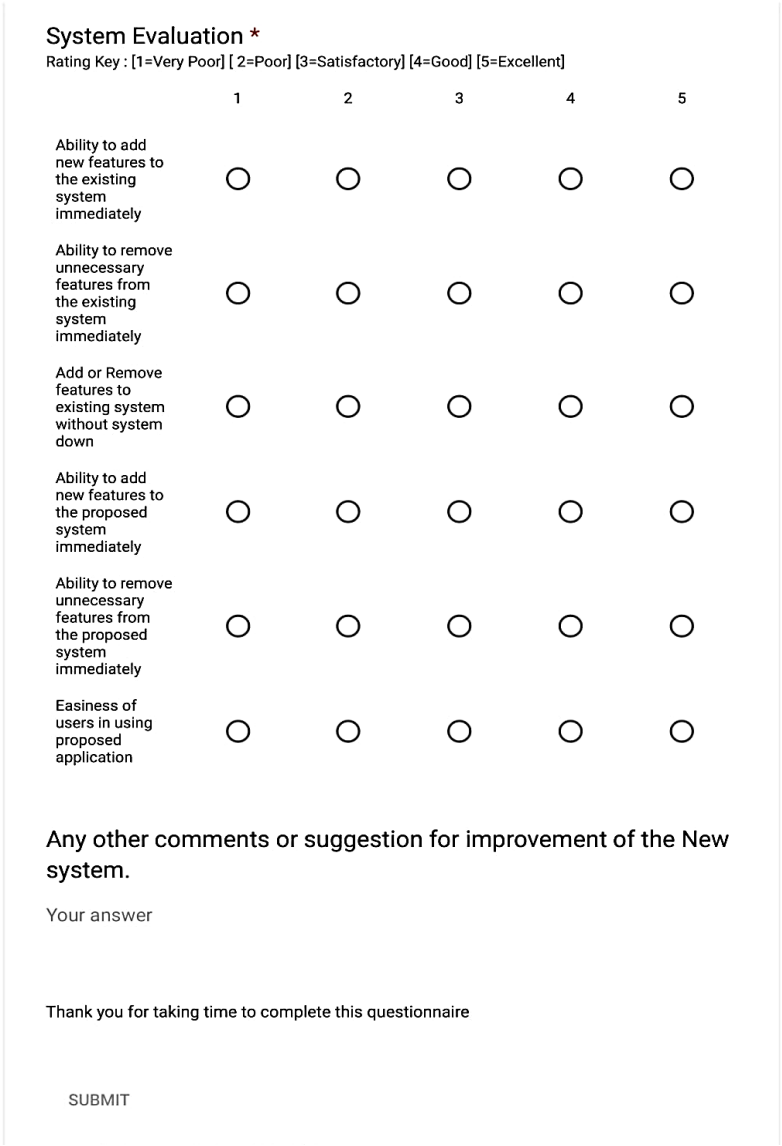


Figure 28 Questionnaire Evaluation - Hosted Software

- Evaluation Sheet of Questionnaire

	Questionnaire Evaluation - Hosted SoftwareSolutions (Pvt)										
Participant	Designation	Department you work for	Highest Educational Level	Age Category	Experience in working Hosted Software Solutions (Pvt)	System Evaluation					
						Ability to add new features to the existing system immediately	Ability to remove unnecessary features from the existing system immediately	Add or Remove features to existing system without system down	Ability to add new features to the proposed system immediately	Ability to remove unnecessary features from the proposed system immediately	Easiness of users in using proposed application
1	Software Engineer /Senior Software Engineer	Software Department	Master/PhD	18-27 years	10< years	[2=Poor]	[1=Very Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[5=Excellent]
2	Implementation Engineer / Senior Implementation Engineer	Impelementation Department	Degree	18-27 years	3-5 years	[1=Very Poor]	[2=Poor]	[2=Poor]	[3=Satisfactory]	[4=Good]	[4=Good]
3	Implementation Engineer / Senior Implementation Engineer	Impelementation Department	Diploma/Higher Diploma	28-37 years	1> years	[2=Poor]	[2=Poor]	[2=Poor]	[4=Good]	[4=Good]	[3=Satisfactory]
4	QA Engineer / Senior QA Engineer	QA Department	Diploma/Higher Diploma	18-27 years	3-5 years	[1=Very Poor]	[2=Poor]	[2=Poor]	[4=Good]	[5=Excellent]	[5=Excellent]
5	QA Engineer / Senior QA Engineer	QA Department	Degree	28-37 years	3-5 years	[1=Very Poor]	[2=Poor]	[2=Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
6	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	3-5 years	[3=Satisfactory]	[2=Poor]	[1=Very Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
7	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	3-5 years	[2=Poor]	[1=Very Poor]	[2=Poor]	[3=Satisfactory]	[3=Satisfactory]	[5=Excellent]
8	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	3-5 years	[3=Satisfactory]	[2=Poor]	[2=Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
9	Implementation Engineer / Senior Implementation Engineer	Impelementation Department	Degree	18-27 years	3-5 years	[2=Poor]	[1=Very Poor]	[2=Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
10	Software Engineer /Senior Software Engineer	Software Department	Master/PhD	18-27 years	3-5 years	[2=Poor]	[1=Very Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[5=Excellent]
11	QA Engineer / Senior QA Engineer	QA Department	Master/PhD	18-27 years	1-2 years	[2=Poor]	[3=Satisfactory]	[2=Poor]	[5=Excellent]	[5=Excellent]	[4=Good]
12	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	1-2 years	[2=Poor]	[2=Poor]	[1=Very Poor]	[5=Excellent]	[5=Excellent]	[4=Good]
13	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	3-5 years	[2=Poor]	[2=Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[3=Satisfactory]
14	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	3-5 years	[1=Very Poor]	[3=Satisfactory]	[2=Poor]	[3=Satisfactory]	[5=Excellent]	[3=Satisfactory]
15	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	1> years	[2=Poor]	[3=Satisfactory]	[2=Poor]	[5=Excellent]	[5=Excellent]	[4=Good]
16	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	3-5 years	[2=Poor]	[2=Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[4=Good]
17	Implementation Engineer / Senior Implementation Engineer	Impelementation Department	Degree	18-27 years	10< years	[1=Very Poor]	[3=Satisfactory]	[2=Poor]	[5=Excellent]	[5=Excellent]	[4=Good]
18	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	3-5 years	[2=Poor]	[2=Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[4=Good]
19	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	3-5 years	[1=Very Poor]	[2=Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[5=Excellent]
20	Implementation Engineer / Senior Implementation Engineer	Impelementation Department	Degree	18-27 years	1-2 years	[1=Very Poor]	[1=Very Poor]	[2=Poor]	[5=Excellent]	[5=Excellent]	[4=Good]
21	Software Engineer /Senior Software Engineer	Software Department	Degree	28-37 years	3-5 years	[2=Poor]	[2=Poor]	[2=Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
22	Software Engineer /Senior Software Engineer	Software Department	Degree	18-27 years	3-5 years	[2=Poor]	[2=Poor]	[1=Very Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
23	Software Engineer /Senior Software Engineer	Software Department	Degree	28-37 years	3-5 years	[2=Poor]	[1=Very Poor]	[2=Poor]	[5=Excellent]	[4=Good]	[5=Excellent]
24	Software Engineer /Senior Software Engineer	Software Department	Diploma/Higher Diploma	28-37 years	3-5 years	[2=Poor]	[2=Poor]	[2=Poor]	[4=Good]	[4=Good]	[4=Good]
25	Implementation Engineer / Senior Implementation Engineer	Impelementation Department	Degree	18-27 years	3-5 years	[1=Very Poor]	[3=Satisfactory]	[2=Poor]	[3=Satisfactory]	[3=Satisfactory]	[5=Excellent]

Figure 29 Evaluation Sheet of Questionnaire

- Criteria of Evaluation Sheet

Software Engineer /Senior Software Engineer
Implementation Engineer / Senior Implementation Engineer
QA Engineer / Senior QA Engineer
Software Department
Impelementation Department
QA Department
Advanced Level
Diploma/Higher Diploma
Degree
Master/PhD
18-27 years
28-37 years
38-47 years
48-57 years
More than 57 years
1> years
1-2 years
3-5 years
10< years
[1=Very Poor]
[2=Poor]
[3=Satisfactory]
[4=Good]
[5=Excellent]

Figure 30 Criteria of Evaluation Sheet

Appendix B: Interview Questions

01. Satisfaction level of allowing to create web modules separately.

- ☐ Extremely satisfied
- ☐ Satisfied
- ☐ Need to improve
- ☐ Not so satisfied
- ☐ Not satisfied

02. Satisfaction level of allowing to separate business process to each module.

- ☐ Extremely satisfied
- ☐ Satisfied
- ☐ Need to improve
- ☐ Not so satisfied
- ☐ Not satisfied

03. Satisfaction level of allowing to design screens separately.

- ☐ Extremely satisfied
- ☐ Satisfied
- ☐ Need to improve
- ☐ Not so satisfied
- ☐ Not satisfied

04. Satisfaction level of allowing to add new modules.

- ☐ Extremely satisfied
- ☐ Satisfied
- ☐ Need to improve

- [ ] Not so satisfied
- [ ] Not satisfied

05. Satisfaction level of allowing to modify existing module.

- [ ] Extremely satisfied
- [ ] Satisfied
- [ ] Need to improve
- [ ] Not so satisfied
- [ ] Not satisfied

06. Satisfaction level of allowing to configure new modules to base module.

- [ ] Extremely satisfied
- [ ] Satisfied
- [ ] Need to improve
- [ ] Not so satisfied
- [ ] Not satisfied

07. Satisfaction level of allowing to publish base module.

- [ ] Extremely satisfied
- [ ] Satisfied
- [ ] Need to improve
- [ ] Not so satisfied
- [ ] Not satisfied

08. Satisfaction level of allowing to publish sub modules to base module.

- [ ] Extremely satisfied
- [ ] Satisfied

☐ Need to improve

☐ Not so satisfied

☐ Not satisfied

09. Satisfaction level of detaching modules from base module.

☐ Extremely satisfied

☐ Satisfied

☐ Need to improve

☐ Not so satisfied

☐ Not satisfied

10. Satisfaction level of overall progress of suggested architecture.

☐ Extremely satisfied

☐ Satisfied

☐ Need to improve

☐ Not so satisfied

☐ Not satisfied