ASSIGNMENT-I SUBMISSION REPORT

Enterprise Application Development (425)

Gunathilaka D. D. T. M. (IT13011130)

AUGUST 14, 2016

SLIIT

The following report describes the implementation of “Order Management” Application which provides the solution for the first assignment of Enterprise Application Development module

Declaration

I hear by declare that following task is my own creation and it does not violate any constraints of the assignment.

Gunathilaka D. D. T. M. (IT13011130)

Contents

List of Figures

Introduction

Java Enterprise Application is created using Java Enterprise Edition (JAVAEE). This application is capable of managing Customers and Orders within the system. The implemented solution is named as “Order Management” and below use case diagram illustrates the functional requirements of the solution.

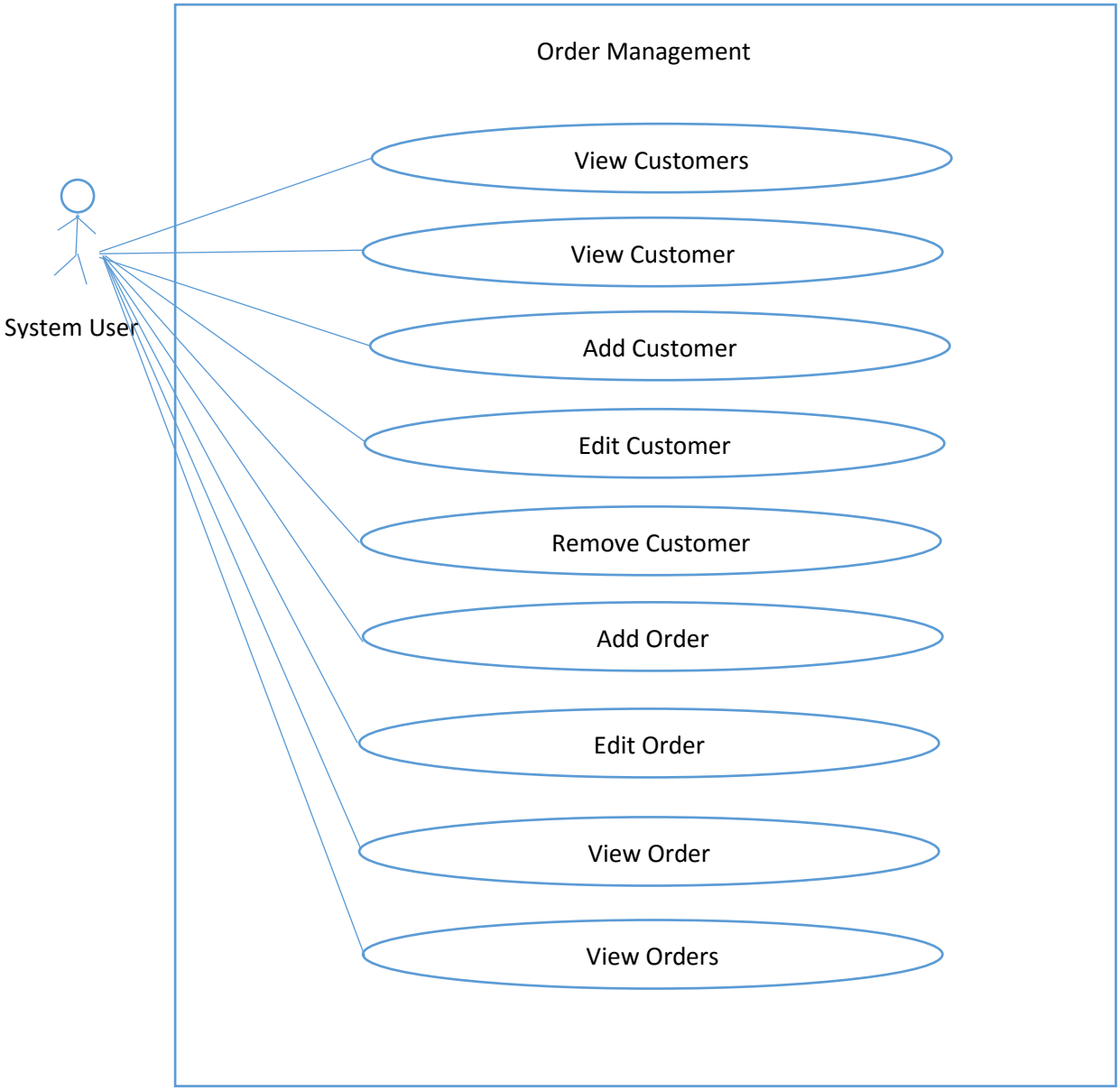

Figure 1 Use case Diagram of Order Management

Methodology

Design & Technologies

The application developed following three tier architecture, where it consists with data tire, business & logic tire and presentation tiers. Furthermore solution is deployed as two components namely "EJB" module and "Web" module. In "EJB" module it contains the data-access and business logics while presentation layer resides inside the "Web" module.

In order to facilitate client interaction "Java Servlets" are used in the presentation layer. These servlets are hosted in a Glass Fish server. The business logics are called within these servlets to invoke data access methods. Furthermore data access methods are implemented using Java persistence while Java DB is used as the data storage.

Layer	Classes
Data Access Tier	CustomerManagementEntity.Customer OrderManagementEntity.CustomerOrder
Business Logic Tier	SessionFacadeDAO. AbstractFacade SessionFacadeDAO. CustomerFacade SessionFacadeDAO. CustomerOrderFacade Asynchronous. CreateOrderMDB
Presentation Tier	Servlet. AllCustomers Servlet. AllOrders Servlet. CreateCustomer Servlet. CreateOrder Servlet. DeleteCustomer Servlet. HomePage Servlet. ManageCustomer Servlet. ManageOrder Servlet. ViewCustomer Servlet. ViewOrder HTMLCreation. CheckAndValidate HTMLCreation. HTML HTMLCreation. POPUP

Figure 2 Tiers and respective classes

The Following UML Class Diagram describes the organization of classes in the implemented solution.

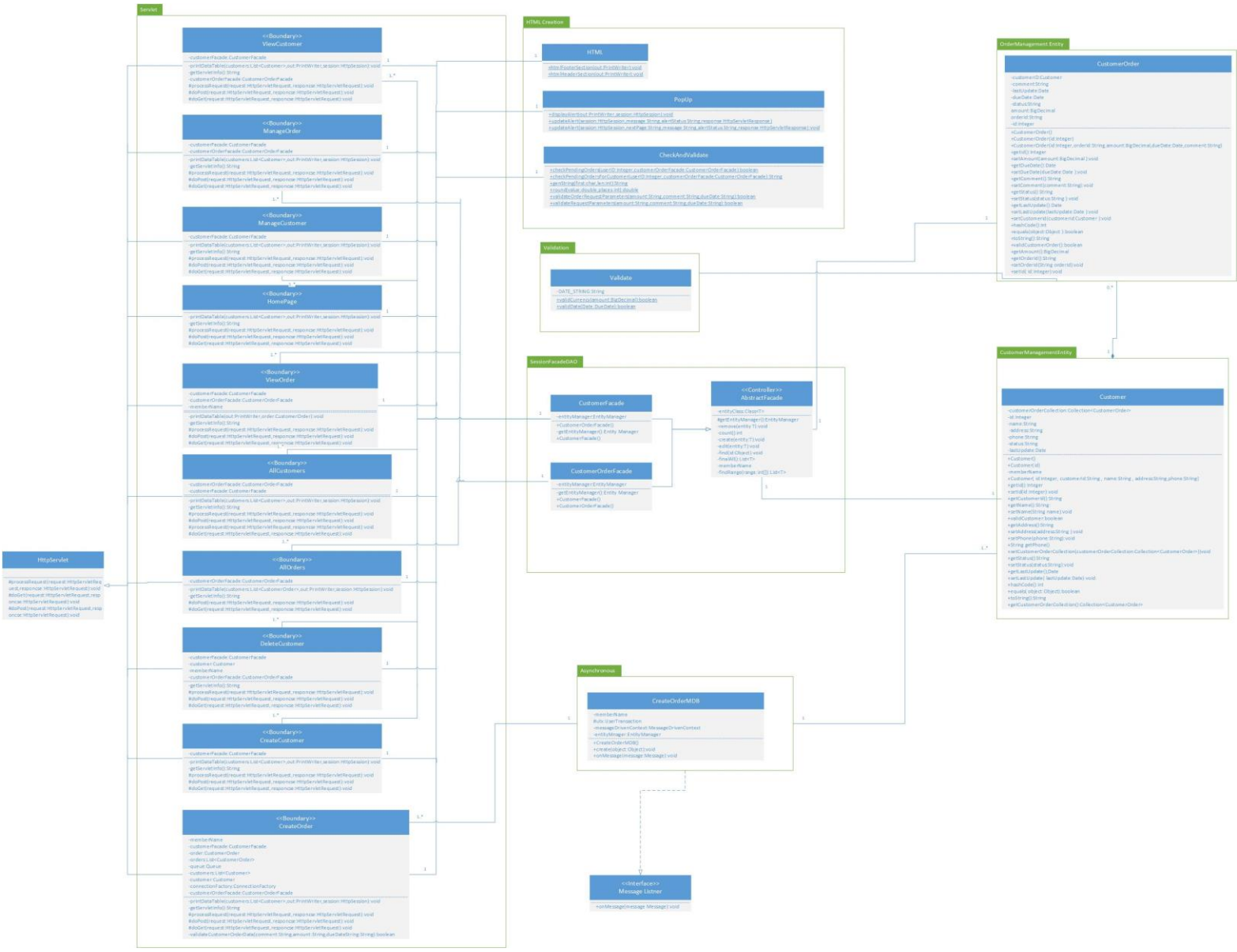

Figure 3 Class diagram of Order Management

Implementation

Synchronous & asynchronous call usage

The new orders are handled asynchronously using Message Driven Beans (MDB).Object Messages are used to pass “CustomerOrder “Entity Bean from presentation tier to business logic tier. The created Object Messages which contain “CustomerOrder” Entity Beans are pushed into a Session Queue which carries respective Object Message to the business logic tier.

In business logic tier the implemented Message Driven Bean instance listens to incoming Object Messages via Session Queue. Once a new Object Message arrives it executes “onMessage (…)“event where it grabs the content of it. This process happens asynchronously due to default behavior of Message Driven Beans.

In this solution the requirement is to save incoming “Order” Entity Beans which arrive via Session Queue as Object Messages. In order to save Entity Bean to database,” Entity Manager” instance is created. This instance consists of” Persist ()” method which allows to save incoming “Order” Entity Bean in database.

As the conclusion Message Driven Bean listens & captures incoming “Order” Entity Beans and save those in database using “Persist” method. This entire process executes asynchronously due to default behavior of Message Driven Beans.

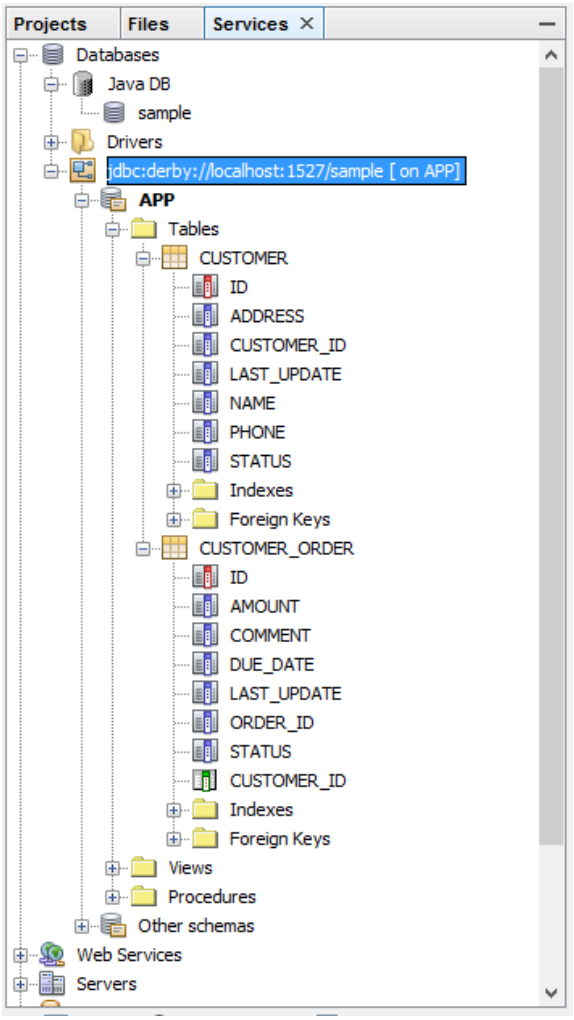

Figure 4 Order Management Table Structure

All the other data access logics except creating new order are invoked synchronously by invoking methods in Façade classes. These Façade classes are the Session Beans which manipulate “Customer” and “CustomerOrder “Entity Beans at runtime.

Data access and table structure

In the database there are 2 tables “CUSTOMER” and “CUSTOMER _ORDER” to represent Entity Beans. The “CUSTOMER _ORDER” table refers “id” from “Customer” table.

Tables are mapped with entity classes and also javax.persistance annotations are used to indicate primary and foreign keys. The @id annotation allows creating primary keys in tables. Foreign keys and cardinality ratio are annotated using @JoinColum and @ManyToOne respectively.

Discussion

Technical

- Application is implemented using JavaEE version 6
- The solution compiles and deployed using JDK 1.7.0_79
- GlassFish server version 3.1.0.2 is used to deploy the application

Standards

- The “Google's Java coding standards” has followed as the coding standard throughout the solution implementation

Sample Screens

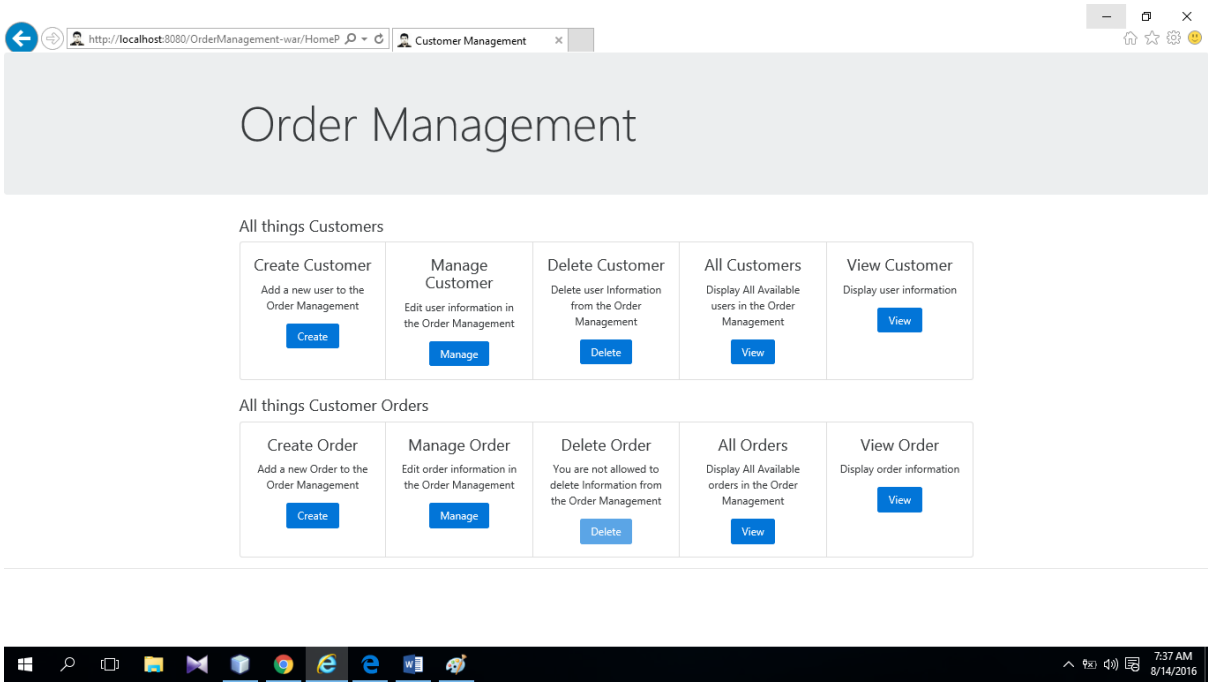

Figure 5 Order Management Home Page

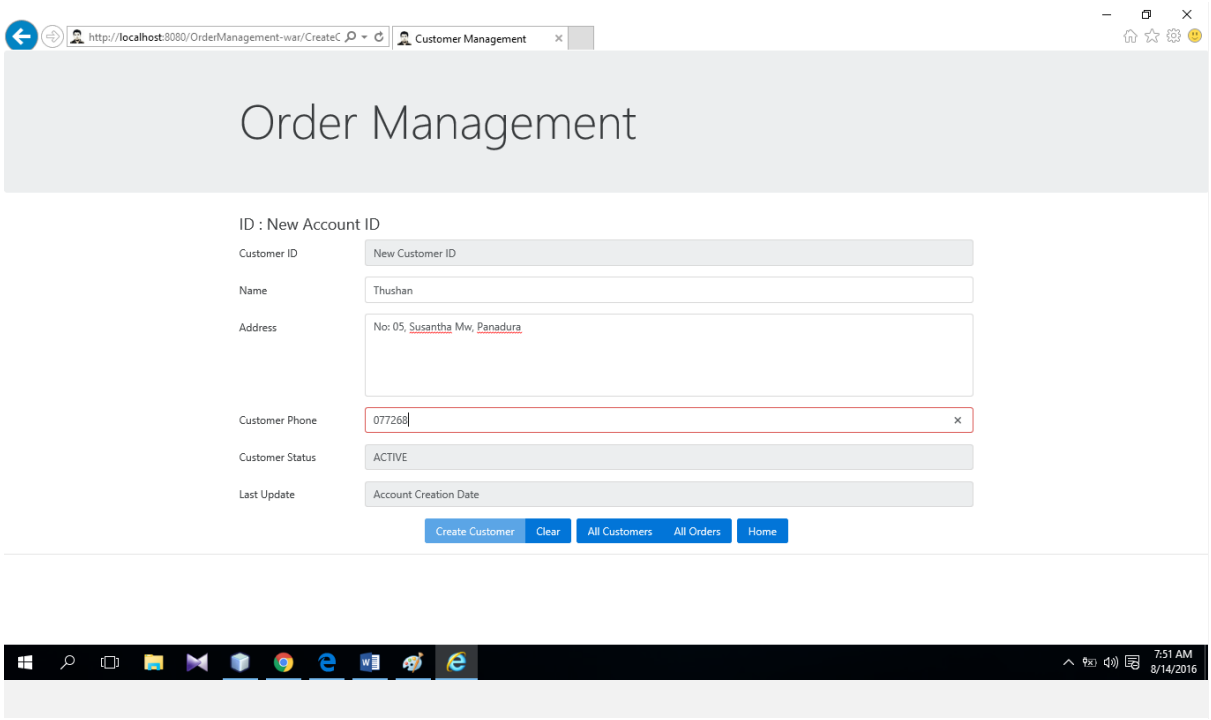

Figure 7 Create a Customer

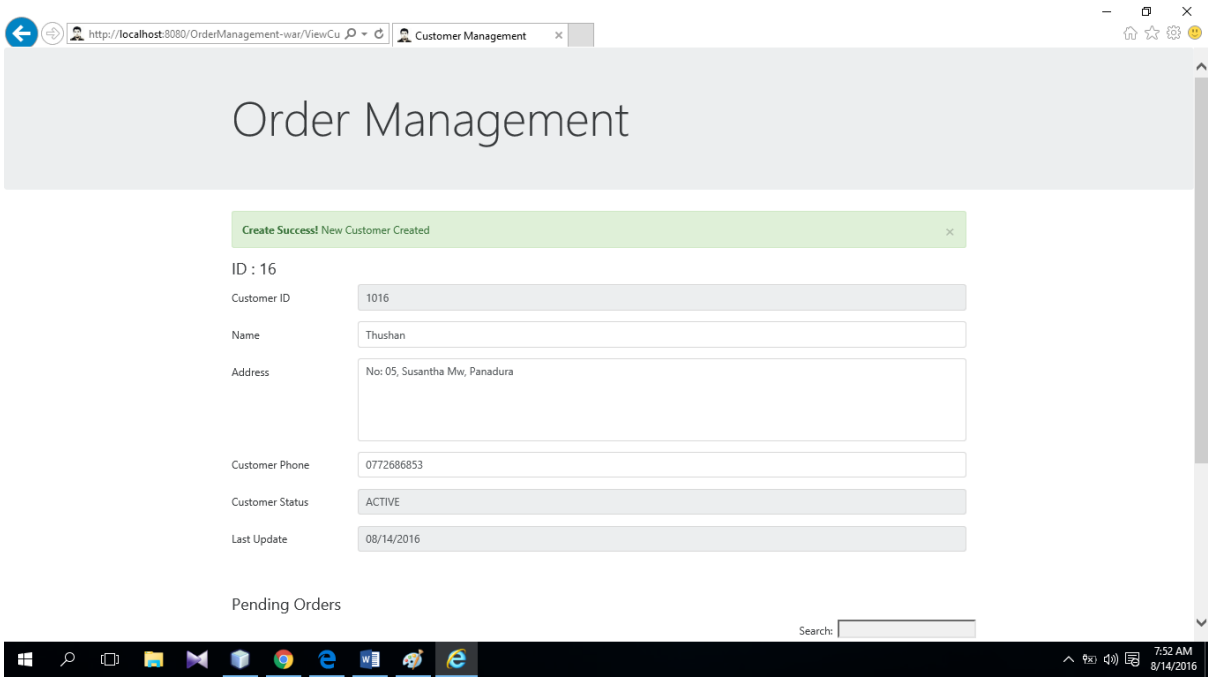

Figure 6 View Customer I

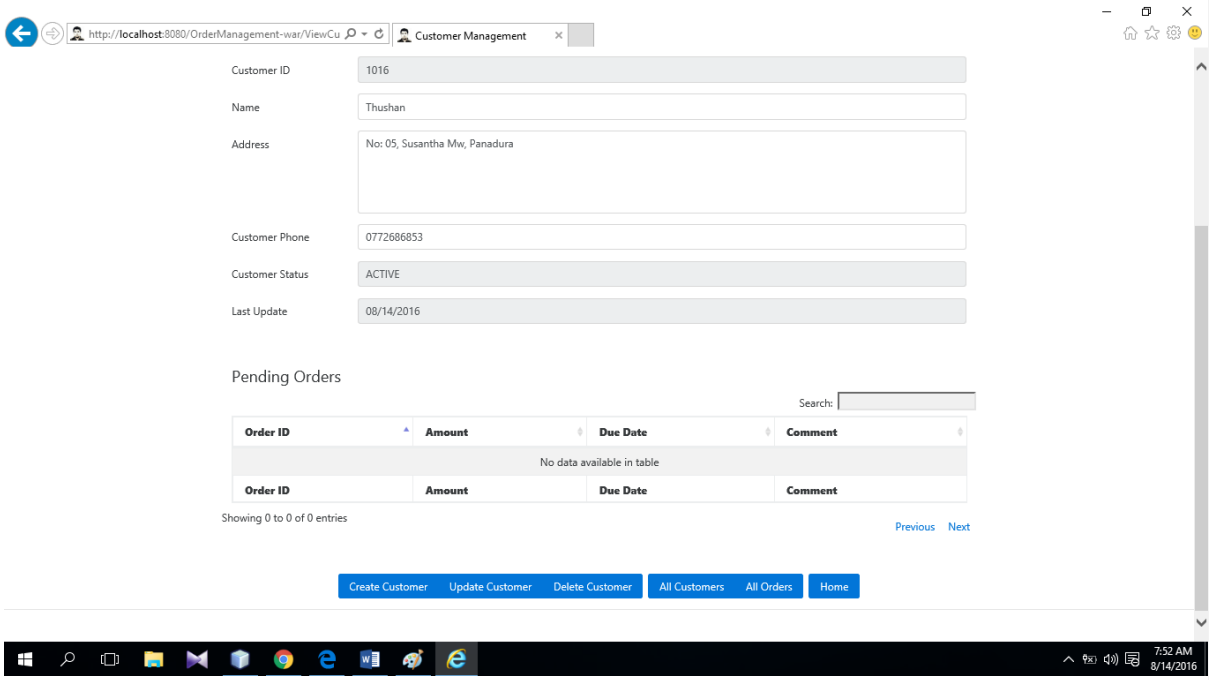

Figure 8 View Customer II

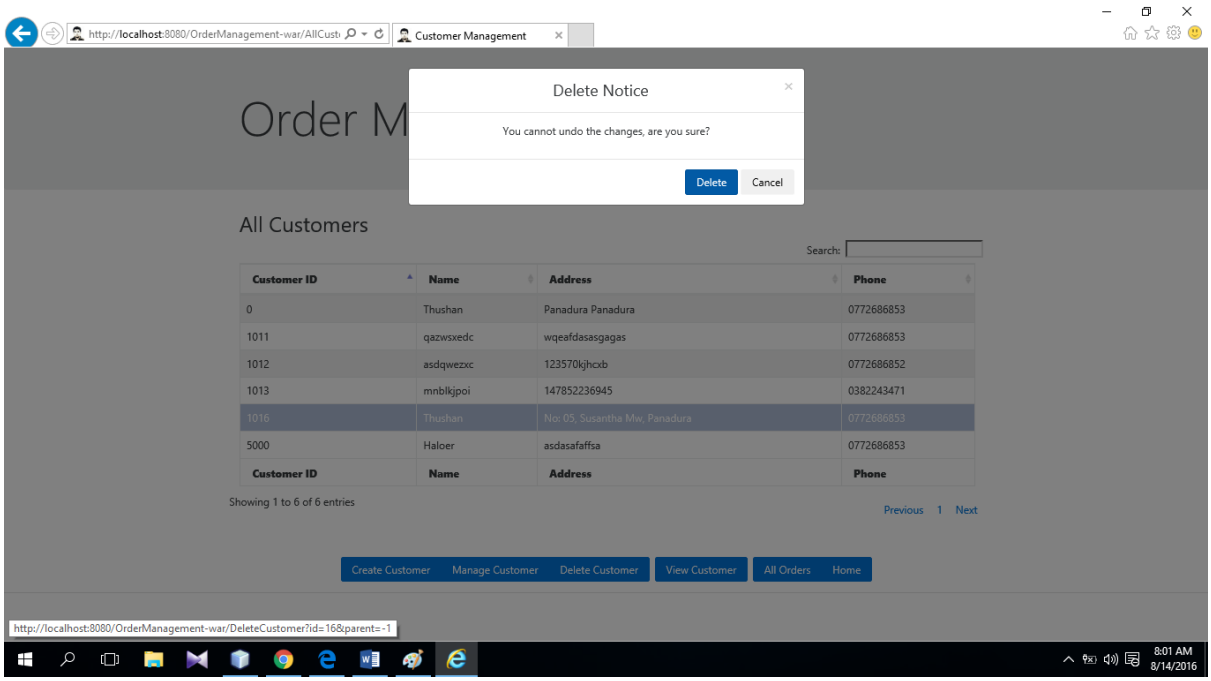

Figure 9 Delete a Customer

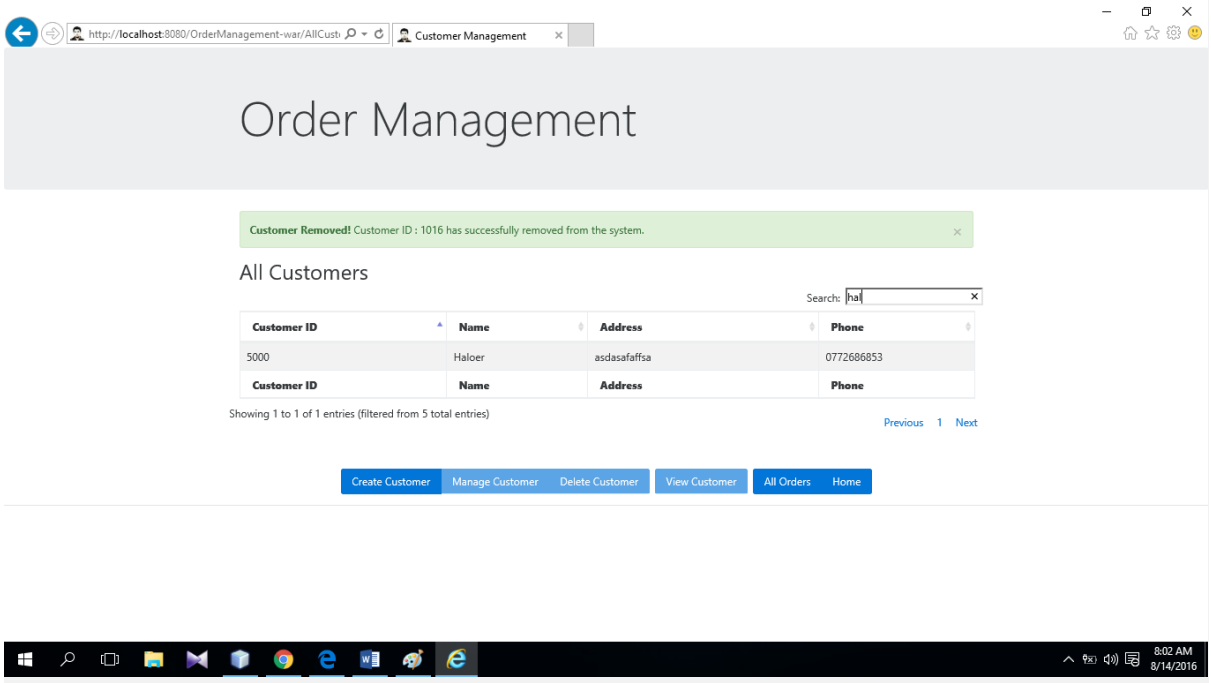

Figure 10 All Customers I

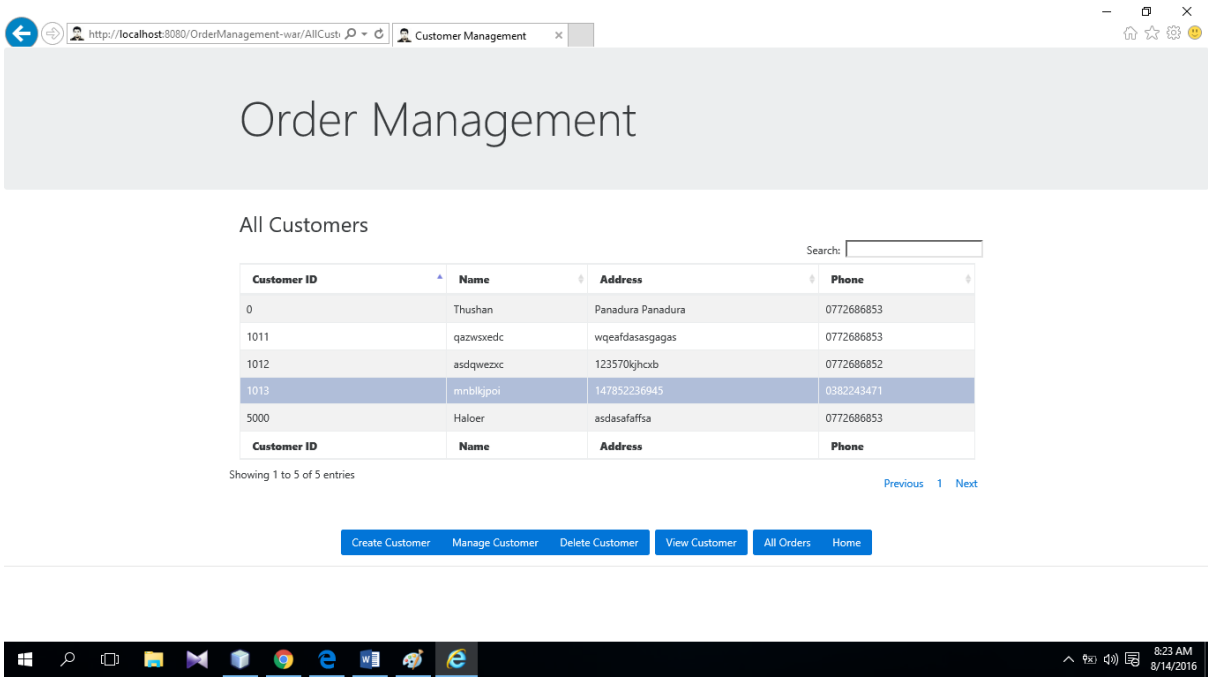

Figure 11 All Customers II

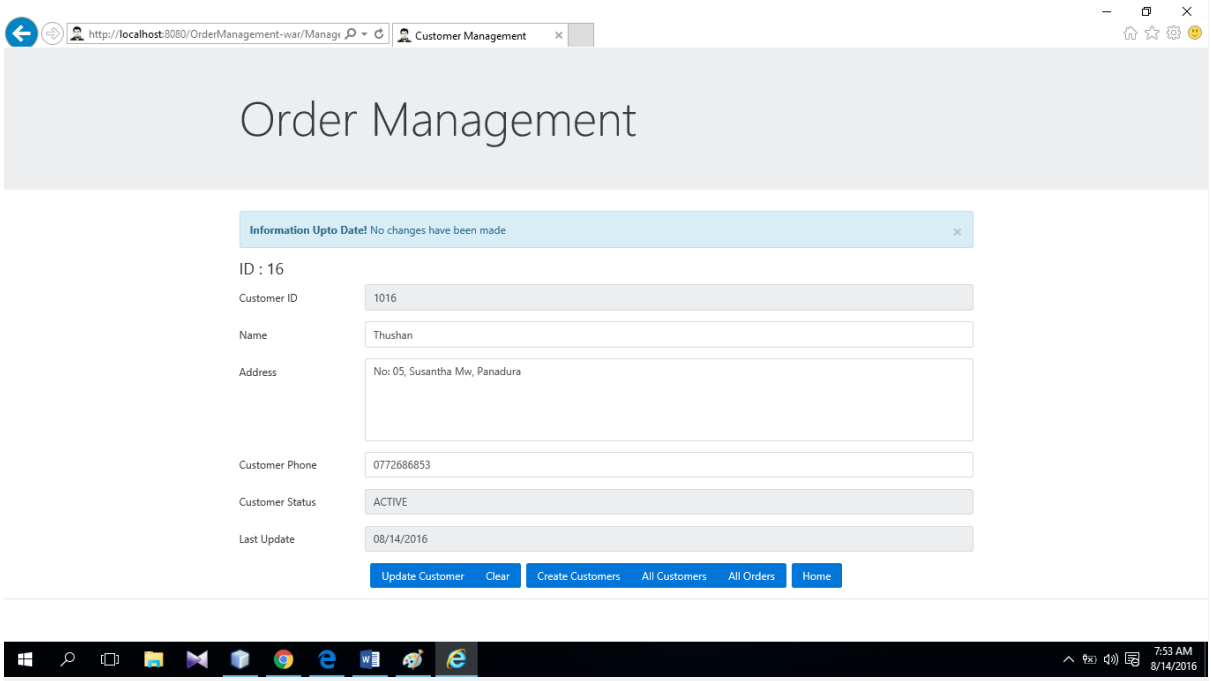

Figure 12 Edit Customer

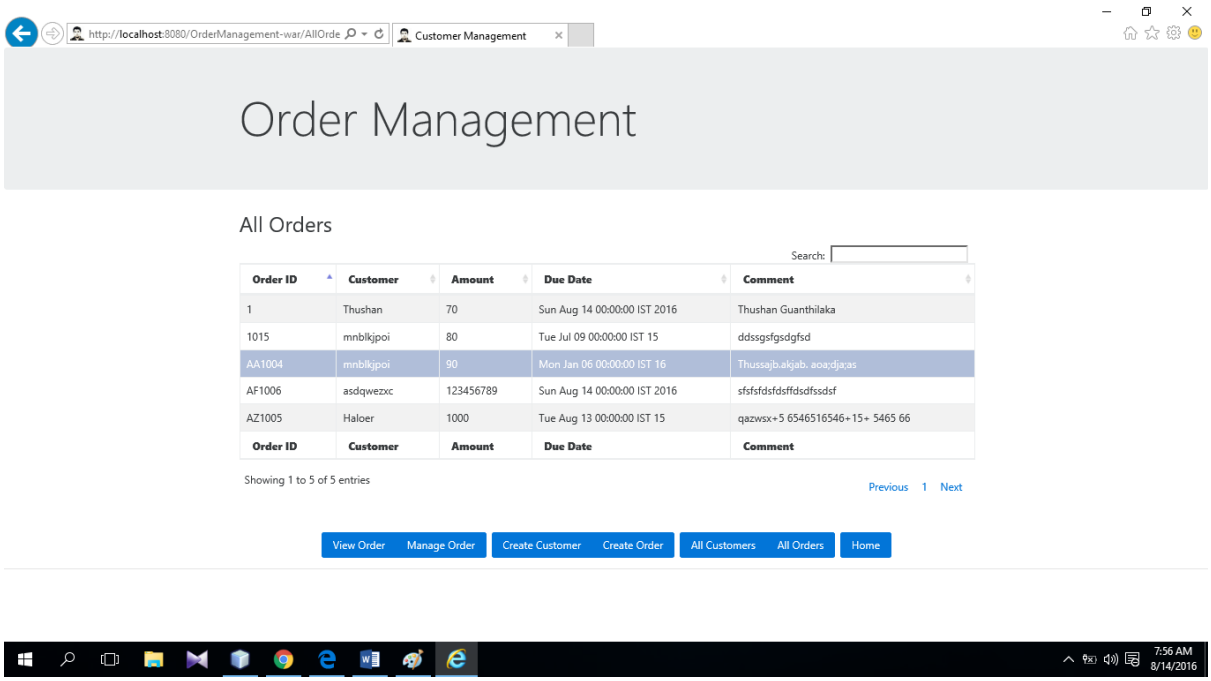

Figure 13 All Orders

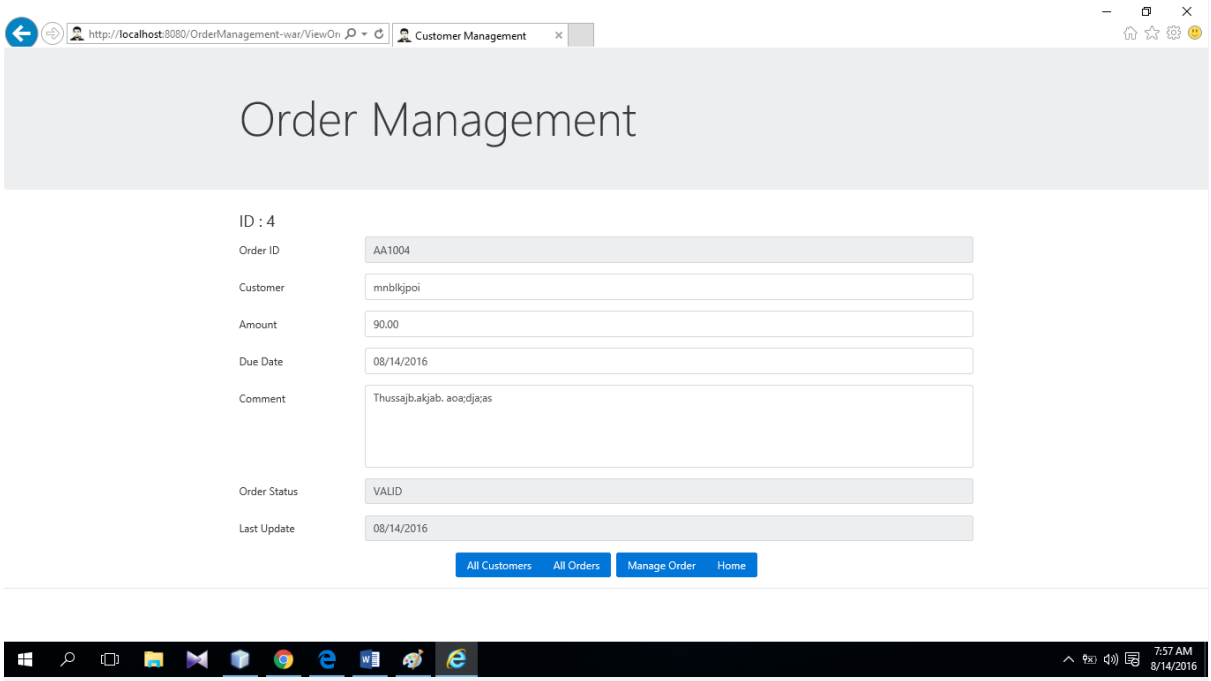

Figure 14 View Order

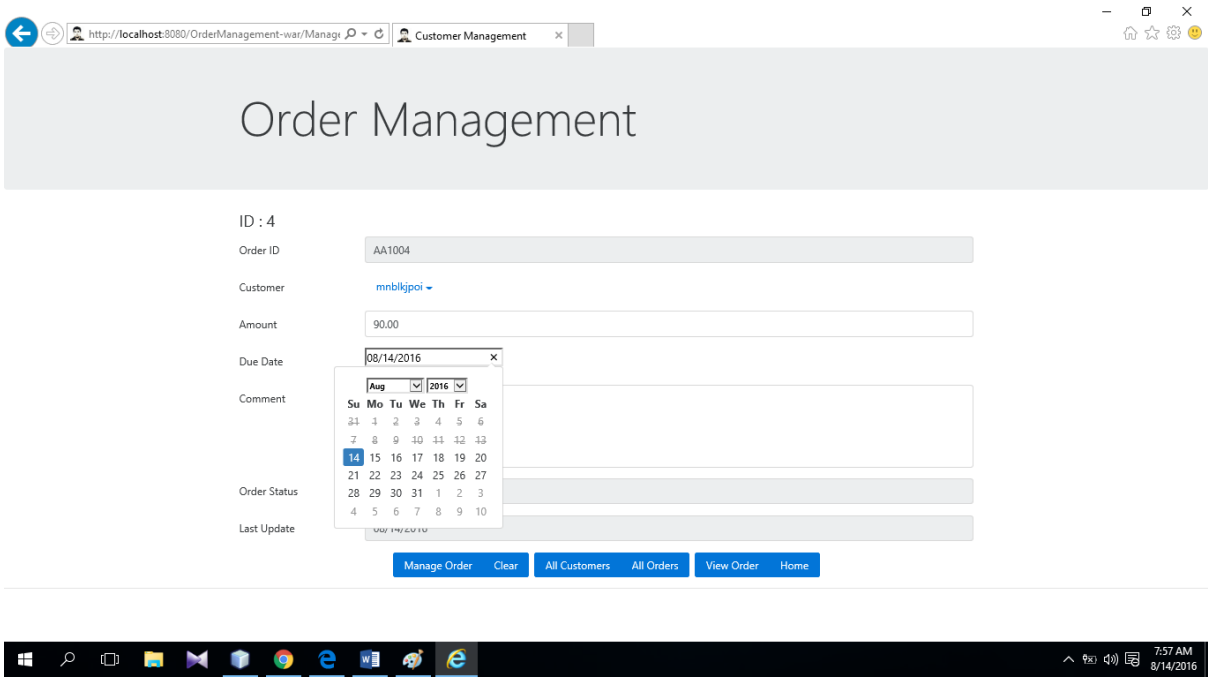

Figure 15 Edit Order

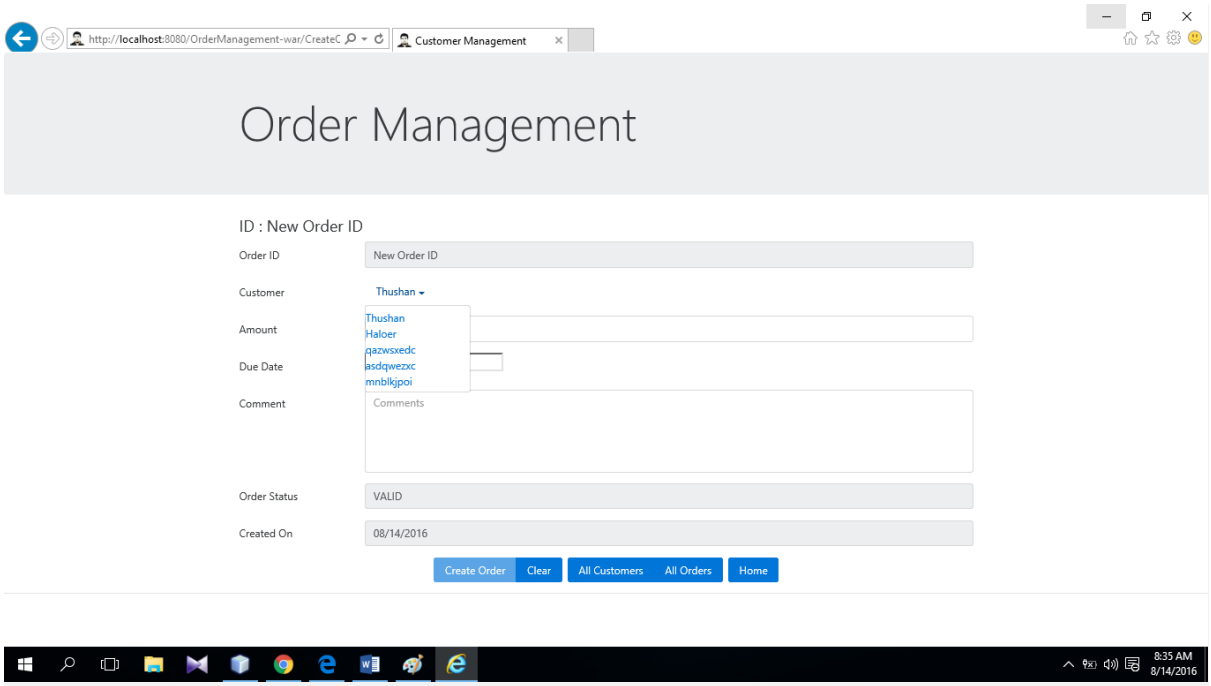

Figure 16 Create Order