Faculty of Science, Engineering and Technology

COS30041 Creating Secure and Scalable Software

Research Report for High Distinction (HD)

Prepared by: S M Ragib Rezwan 103172423

[Optional Feedback, timeline and schedule]

Discussion with Tutor for Feedback: **Week 10 – 12 Lab classes**

[Final Submission]

Submission for Portfolio: **Week 14, Monday, 9:00am**

Instructions - This document is for students aiming to achieve High Distinction (HD).

For **HD**, a student needs to complete the software for D grade as well as a research report. Possible options are

R1 Implement the same functionality of the software for D grade using two different technologies of the same language (e.g comparing JSF with PrimeFaces) and compare the two in terms of some criteria nominated by the student (e.g. performance or ease of development).

R2 Implement the same functionality twice (one using Java EE technologies and the other using .NET technologies), and compare the two in terms of some criteria nominated by the student (e.g. performance or ease of development)

The work in this option involves integrating Java EE applications with .NET technologies or vice versa.

R3 Other please specify (to be detailed in the research proposal)

In the research report, the student must (1) collect useful and relevant data, (2) perform their own analysis (quantitative comparison, NO qualitative comparison) and (3) draw conclusion based on their comparison.

Your research report is a free-form report. You can decide on your own sectioning. The one below is just an example. It may not suit your needs. Please feel free to customize it. However, you must present your research results in a concise and precise manner that the interview panel could understand.

Formatting guidelines: 10pt font size, single line spacing, 6 – 8 pages including diagrams, tables, figures and references.

Research Report: Comparison between JavaEE and .NET framework on data submission via webpage

Research Topic: Comparison between JavaEE and .NET framework on data submission in terms of webpage forms

I wish to test the functionality of "a form on a webpage inserting data into a database" for webpages created using the two different frameworks. In order to accomplish this, I am going to set up the same webpage (from backend to frontend) using both JavaEE and also .Net Framework, before comparing them using the following matrixes: number of files and lines of code (and languages) needed to set up the functionality, development time and performance of the webpage.

Research Option: "R2" [Implimenting the same functionality using different technologies (JavaEE vs .NET)]

Introduction

In this report, I am going to compare the two frameworks JavaEE and .Net on basis of development of a website for Create operation (ie inserting records into a database) from the CRUD (create, read, update and delete) operations.

I had gotten this idea while doing my D level project where I needed to setup the JavaEE framework in order to allow customers to only add records and to allow authorised admins to only view them. In the weeks I spent setting up the framework completely for these functionalities there, I experienced and faced though lots of strange and unusual errors (like missing certain jar files, sun security hello error, etc) whenever I tried to set up something differently from the way that had been taught in class. Thus, in the end, I had been forced to follow the patterns taught in class in order to ensure everything worked as intended in the website.

This made me feel irritated as although the frontend was different, the backend had to follow a certain restrictive and lengthy pattern. So, I started to research on alternative ways to develop web application, leading me to .NET framework. It was then that I realised that .NET framework could be run on visual studio community (an editor I used to use in my previous course for C# coding) and by installing different modules (**[1],[2], [3]**), I could develop and run SQL databases and websites on it!

Thus, a question rose in my mind, "since .NET can also be used to develop websites, is it better than JavaEE (ie the framework we had used in this course)"? So, I began my research in order to answer it.

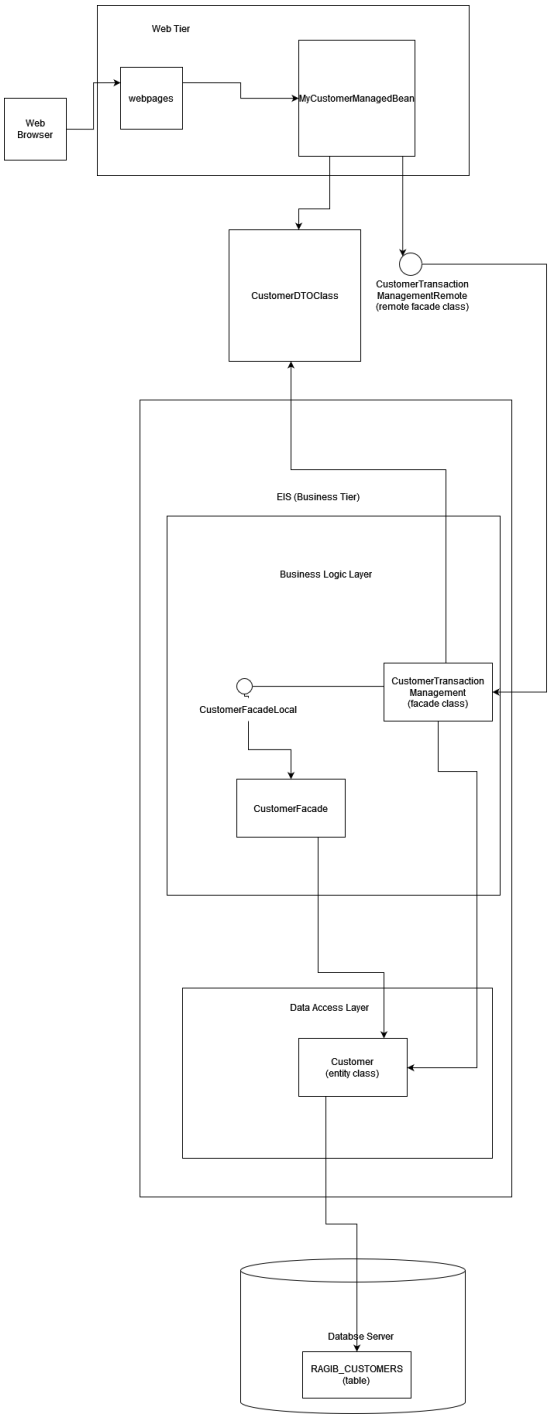
Research Methodology

In order to compare the two technologies, I needed to develop the same webpage using those JavaEE and .NET. But while looking up different ways to develop the websites, I found out that there were actually multiple ways in .NET to develop websites, each with their own uniqueness (which you can guess when you look at the version number shown for the .Net files). But unfortunately, there hadn't been enough time to go through and analyse each and every variation of website development in .NET (like Blazor, RAZOR, Web API, MVC, etc.). So, instead, I chose to set up the webpage in the simplest way, so as to ensure the data collected is only for the functionality developed and nothing else (ensuring valid comparison)

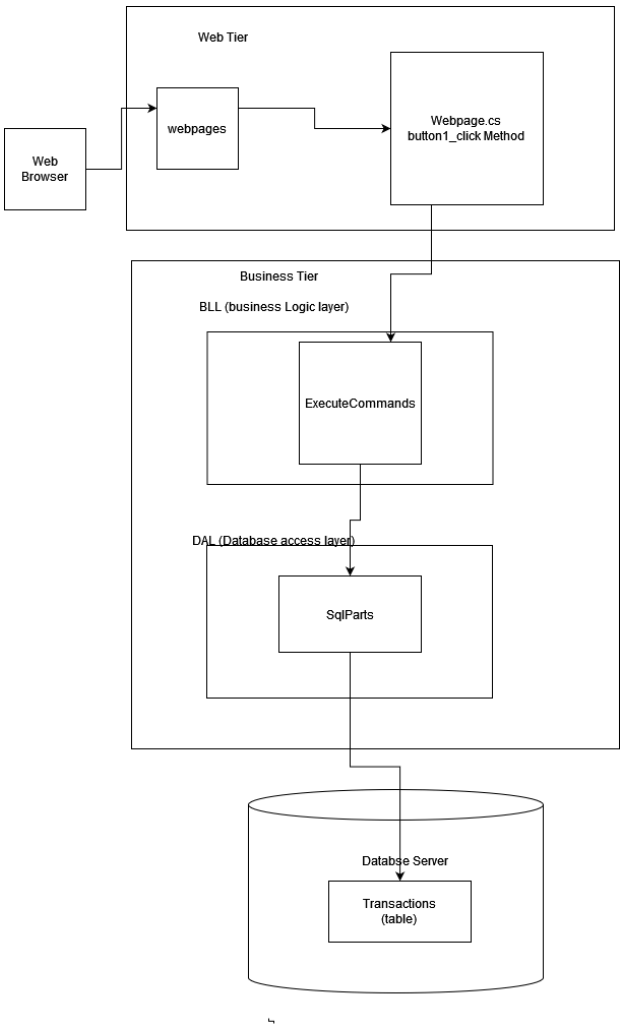
[**Note:** *No security feature or validation has been implemented in any of the websites here as I wanted to check the bare-bones version of the websites' create option, to ensure valid comparison*]

Research Plan

At first I developed a website using JavaEE where only a single webpage with a form to capture the data in the front end, relevant methods, managedBeans, façade classes and data transfer objects to carry and process data, and sql database to store the data in the backend. This can be seen in the architecture diagram below:



After that, I developed the same webpage using .NET technology where I used ASP.NET Web application to create a web form with a single webpage with a from to capture data at front end, executeCommands and SqlParts to process data before connecting to sql database to storing the data in the backend. This can be seen in the architecture diagram below:



*[**Note**: All the codes for all the files needed to develop the respective website has been added in the "Setup" before the reference paragraph]*

After setting them up, I ran the two websites to ensure they were both working as intended before starting my analysis on the following matrixes:

A) Number of files needed in order to set up same functionality (alongside no of lines in code and languages it used)

B) Time taken to develop both features from back to front

C) Performance of the two websites

Observations noted in terms of the mentioned matrixes:

A) <u>Number of files needed in order to set up same functionality (alongside no of lines in code and languages used):</u>

In order to get a good idea on the effort spent by developer in developing the system in both JavaEE and .NET, I have noted down the number of files required by the respective systems, number of lines of codes needed to be written and also the languages in the respective systems.

Here, I have counted the number of files by hand as it is unchanging and also of small number for both cases. Furthermore, I have only counted the files which had either been created from scratch or modified by me and have not included the files automatically generated by the respective systems as they will auto created by the systems whenever any developer uses them to build a web application. I have noted the results in the table below:

No. of files needed by JavaEE	No. of files needed by .NET
13	5

Furthermore, the numbers of lines of codes were also static. Thus I counted those using the line numbers written on each file and added them together to get the total.

No. of line of code needed by JavaEE	No. of line of code needed by .NET
1569	358

*[**Note**: values of lines of codes for JavaEE: 255,76,25,127,120,236,48,104,20,296,221,19,22 which gave the cumulative 1569 and values of lines of codes for .Net: 115,29,162,23,29 which gave the cumulative 358]*

Moreover, the languages used in the respective systems were the following:

Languages used to develop the system in JavaEE	Languages used to develop the system in .NET
Java, xhtml, SQL,xml	C#,html, SQL,

B) Time taken to develop both features from back to front:

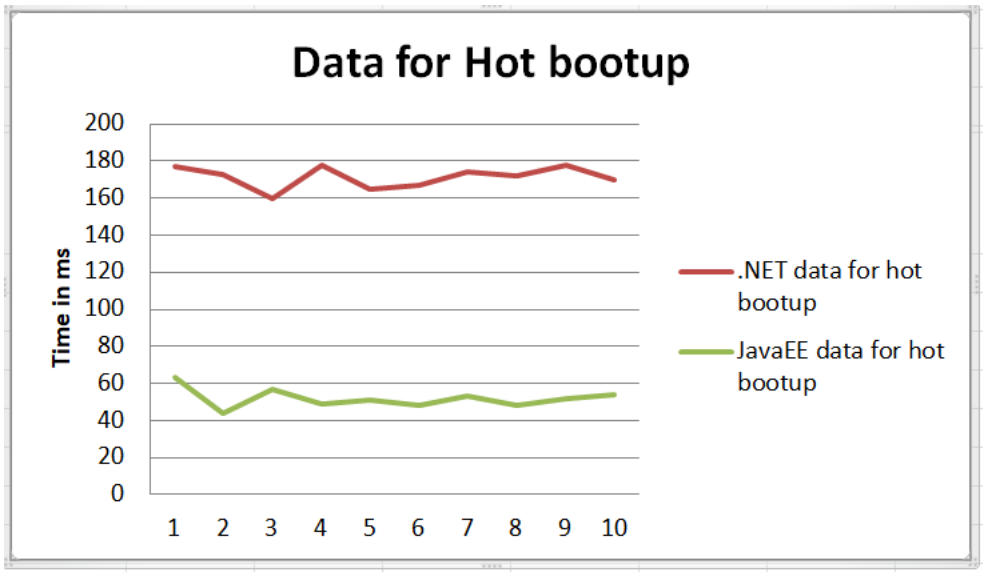
In order to further ensure the effort spent by developer in developing the system in both JavaEE and .NET, I have also noted down the time it had taken to develop the two systems. In this case, I had used the stopwatch on my android device to measure the time it had taken to develop the two systems in their respective technologies from scratch. This had resulted in the following values:

Time taken to develop the system in JavaEE	Time taken to develop the system in .NET
55 min 35 sec 66 ms	27 min 19 sec 41 ms

C) Performance of the two websites:

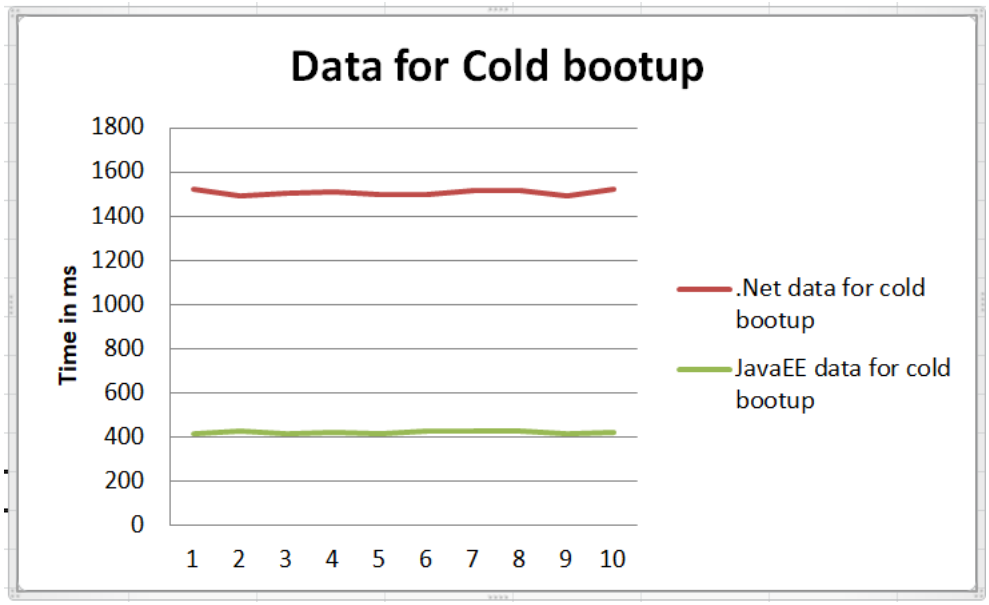
In order to measure the performance of the two systems in performing create or post operations from webpage to database, I decided to use “Network monitor” tool provided by Firefox browser (**[4]**). This tool showed all the http requests made on the page opened on Firefox browser, with the time noted in milliseconds (ms). Then I ran the two software to open the website in Firefox browser for the following scenarios:

a) For Hot bootup, where the device had been restarted between each respective reading to ensure no other software had impacted the readings:



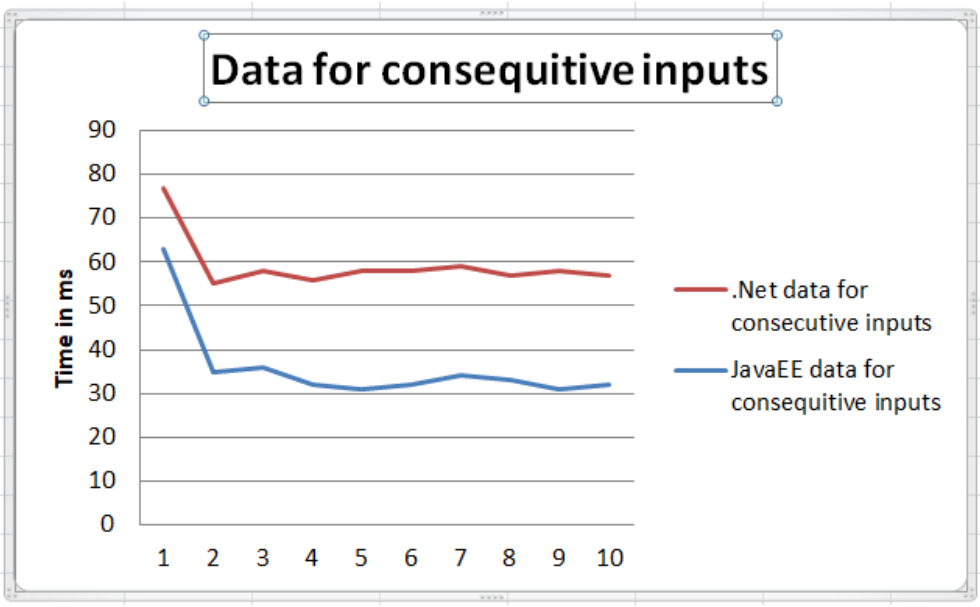
Here we can see that overall, .Net had taken an average time of **171.4 ms** which is ***69.72%*** more time than JavaEE which took an average of **51.9 ms** for the hot bootups.

b) For Cold bootup, where the device had been shutdown between each respective reading, waited for 10 minutes, before restarting and taking the readings. This had been done to further ensure no other software had impacted the readings and also to ensure cache memory had been completely cleared:



Here we can see that overall, .Net had taken an average time of **1507 ms** which is ***71.99%*** more time than JavaEE which took an average of **422.1 ms** for the hot bootups.

c) For Consecutive Inputs, where the same website had been used to perform multiple, back to back create operations from their respective webpages to the database. This had been done to take note in the reduction in time taken to perform the operation after the webpage had already been used once (see difference in time for 1st reading and the remaining readings)



Here we can see the following:

	1st reading	Average of the other 9 readings	Percentage difference in the readings
Time taken by .NET	77	57.3333	25.54
Time taken by JavaEE	63	32.8889	47.80

Overall, there was also a difference of **42.64%** between the average consecutive reading time of .NET and JavaEE.

Results, Observation summary and Analysis:

A) <u>Number of files needed in order to set up same functionality (alongside no of lines in code and languages it used)</u>

In this case, I had noticed that the no of files required to set up the website in JavaEE had been **8** more than that required by .NET. Furthermore, it also JavaEE **1211** extra lines of codes to implement the feature when compared with .NET. Last but not the least, JavaEE needed the developer to be have knowledge regarding **Java, Xhtml, Sql** and **Xml languages** in order to use the framework to develop the website whilst .Net only needed **C#, Html** and **Sql languages** knowledge.

Thus, in this case, .Net is better than JavaEE in terms of effort required for development as it required less number of files to be edited, required less number of lines of codes to be written and required proficiency in fewer languages for the developer to create the functionality.

B) Time taken to develop both features from back to front

In this case, JavaEE had taken 50.87% more time in setup when compared with .NET.

Since, I had taken the readings on the stopwatch on my android phone, the average human reactions time of **273 ms** (**[5]**) could have come into effect in the readings. But I hadn't reduced that time from the readings taken as:

a) it affected both readings equally,
b) it was insignificant compared to the difference of **28.2708 mins** between the readings (*where 1 min = 60,000 ms*)

Furthermore, it should also be noted that these times had been achieved as I had been using JavaEE for the last 3 months and had a clear idea about the pattern to follow (and files to create) while I hadn't used .NET before to develop web application or database and thus had to research online to trouble-shoot issues whilst coding. So, if I had the same proficiency in .NET as I have had for JavaEE, it would have been possible to further reduce the time taken for .NET and hence further increase the time difference between them.

Thus, in this case, .Net is better than JavaEE in terms of effort required for development as it required less time to develop the same functionality there when compared to that in JavaEE.

C) Performance of the two websites

Here, we can see that for all cases, .Net took significantly more time than JavaEE in performing the Post/Create operation from website UI to the database for hot bootup, cold bootup and even in consecutive inputs. This showed that .NET was performing slower than JavaEE for the functionality chosen. Furthermore, it showed that other factors (like other software running in background and using memory and cache) did not affect it as the similar difference was also present during hot and cold bootup where only that software were running (in order to maintain proper test condition).

This had been quite surprizing to me, as when I looked up information online, each and every websites had claimed that .Net in fact had better performance than JavaEE, which was exactly opposite of what I had detected in my data!

So, I decided to read those articles in details and find out what criteria they had used to measure the performance. There I noticed that most websites says that both have similar performance (**[6]**) while websites that claim that .Net have better performance are ones which also include compiling and deploying time (**[7]**,**[8]**) alongside the fact that code is further optimized and consumes less memory. So, although it can be noticed that NetBeans (JavaEE) usually takes more time in deploying and loading up webpage when compared with time taken by .NET (proving the claim made by the websites), this time difference has not been recorded during the investigation of performance and thus cannot be confirmed. Thus it would be better to repeat this investigation, noting down those values as well for better performance comparison.

[**Note:** *Here time had been measured for "transfer of data from webpage to database" only as I believed it was adequate to measure performance of data transfer for both systems. Thus I had not measured deployment and webpage load up timings, as I believed these were "out of scope" as they measured performance of the system as a while and not just performance of the specific functionality I had focused on*]

Moreover, they also note the fact that .NET has more upgrades and enhancements (**[8]**) which give it a further edge over JavaEE. But here in the setup, I had used the simplest .NET setup to implement the

functionality on the webpage, without using any of the enhancements (like ones present Razor, MVC, etc.) which may have further reduced the time it took to send the data from web UI to the database.

Conclusion

Overall, two different conclusions can be drawn from investigation in terms of which technology is better. In terms of development effort in implementing the functionality, .NET is better than JavaEE as it takes less time to develop it, with less codes to write and less files to modify. But in terms of performance, JavaEE is better, as it takes less time for the functionality to perform its task (ie transfer data from webpage UI to the database) compared to .NET.

But can this be used for a general purpose comparison between all JavaEE and .NET frameworks regarding the functionalities developed? Not really.

That's because:

a) .NET has multiple ways to develop webapplications and here only the simpliest one had been used in comparison with the JavaEE framework. Thus it would be better to repeat the experiment with other variations of .NET framework (Like MVC, Razor, etc.)

b) Here only Create functionality had been discussed and not all the CRUD functionalities that most webpages tend to do in practical situations. Thus it would be better to test those functionalities as well in order to properly compare between the two technologies

c) In this investigation, performance had been tested using "the time it took for webpage to transfer data to database", and so did not include deployment or page load up time. Thus although it is accurate in terms of performance regarding functionality itself, it cannot be used to say about the overall performance of the two technologies *(noted in the observation paragraph)*

Thus, it would be better to repeat the experiment using other variations of the .NET framework's web development pattern before coming to the conclusion on which technology is better to develop websites with, both in terms of the functionality chosen and also for a general overall web development.

SETUP for the two websites:

[Note: *Here I have only added for pages where I had to edit/ modify data and have not included the files that were automatically generated by either systems***.]**

a)Using JavaEE:

For making the database tables:

Customer.java:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ragib.jdbc;
```

```
import java.io.Serializable;

/**
 *
 * @author Ragib
 */
public class Customer implements Serializable {

    private final String transaction_id;
    private final String firstname;
    private final String lastname;
    private final String email;
    private final String address;
    private final String suburb;
    private final String postcode;
    private final String phone;

    private final String product_name;
    private final String quantity;
    private final String comment;
    private final String credit_card_name;
    private final String credit_card_number;
    private final String credit_card_expiry_date;
    private final String credit_card_CVV;

    //keep it all as string as the frame work keeps messing up for int's case
    //specially when it tried to make the table later on

    public Customer(String transaction_id, String firstname, String lastname, String email, String address, String suburb,
```

```
        String postcode,
        String phone,
        String product_name, String quantity,
        String comment, String credit_card_name, String credit_card_number, String credit_card_expiry_date, String credit_card_CVV) {
    this.transaction_id = transaction_id;
    this.firstname = firstname;
    this.lastname = lastname;
    this.email = email;
    this.address = address;
    this.suburb = suburb;

    this.postcode = postcode;
    this.phone = phone;

    this.product_name = product_name;
    this.quantity = quantity;
    this.comment = comment;
    this.credit_card_name = credit_card_name;
    this.credit_card_number = credit_card_number;
    this.credit_card_expiry_date = credit_card_expiry_date;
    this.credit_card_CVV = credit_card_CVV;
  }

  public String getTransaction_id() {
    return transaction_id;
  }

  public String getFirstname() {
    return firstname;
  }
```

```
public String getLastname() {
    return lastname;
}

public String getEmail() {
    return email;
}

public String getAddress() {
    return address;
}

public String getSuburb() {
    return suburb;
}

public String getPostcode() {
    return postcode;
}

public String getPhone() {
    return phone;
}

public String getProduct_name() {
    return product_name;
}

public String getQuantity() {
```

```
        return quantity;
    }

    public String getComment() {
        return comment;
    }

    public String getCredit_card_name() {
        return credit_card_name;
    }

    public String getCredit_card_number() {
        return credit_card_number;
    }

    public String getCredit_card_expiry_date() {
        return credit_card_expiry_date;
    }

    public String getCredit_card_CVV() {
        return credit_card_CVV;
    }

}
```

CustomerDB.java:

```
package ragib.jdbc;

import java.io.IOException;
import java.util.ArrayList;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;

/**
 *
 * @author Ragib
 */
public class CustomerDB {

   // Database parameters for connection - url, username, password
   static String url;
   static String username;
   static String password;

   static final String DB_TABLE = "Ragib_Customers";
   static final String DB_PK_CONSTRAINT = "PK_" + DB_TABLE;

   /**
    * constructor using default url, username and password
    */
   public CustomerDB() {
      // set default parameters for Derby and JavaDB
      url = "jdbc:derby://localhost/sun-appserv-samples;create=true";
      username = "APP";
      password = "APP";
   }
```

```
/**
 * getConnecion()
 *
 * @aim Get a connection to the database using the specified info
 */
public static Connection getConnection()
        throws SQLException, IOException {
    // first, need to set the driver for connection
    // for Derby
    System.setProperty("jdbc.drivers",
            "org.apache.derby.jdbc.ClientDriver");

    // next is to get the connection
    return DriverManager.getConnection(url, username, password);
}

/*
 * createDBTable
 *
 * @aim Use SQL commands to create the database table
 */
public void createDBTable() {
    Connection cnnct = null;    // declare a connection object
    Statement stmnt = null;     // declare a statement object

    try {
        // get connection
        cnnct = getConnection();
        // get statement
        stmnt = cnnct.createStatement();
```

```
        /**
         * execute query to create a data table with the required fields
         * keeping all of them as strings as the these will only be stored and retrieved from database in form of logs and will not be used in calculation purposes
         */
        stmnt.execute("CREATE TABLE " + DB_TABLE
            + " (Transaction_id VARCHAR(20) CONSTRAINT " + DB_PK_CONSTRAINT + " PRIMARY KEY,"
            + " Firstname VARCHAR(20),"
            + " Lastname VARCHAR(20), "
            + " Email VARCHAR(30), "
            + " Address VARCHAR(30), "
            + " Suburb VARCHAR(30), "
            + " Postcode VARCHAR(4), "
            + " Phone VARCHAR(10), "
            + " Product_name VARCHAR(40), "
            + " Quantity VARCHAR(10), "
            + " Comment VARCHAR(100), "
            + " Credit_card_name VARCHAR(40), "
            + " Credit_card_number VARCHAR(16), "
            + " Credit_card_expiry_date VARCHAR(4), "
            + " Credit_card_CVV VARCHAR(3))");
    } catch (SQLException ex) {
        // do nothing
    } catch (IOException ex) {
        // do nothing
    } finally {
        // close Statement object
        if (stmnt != null) {
            try {
                stmnt.close();
```

```
            } catch (SQLException e) {
                // do nothing
            }
        }

        // close Connection object
        if (cnnct != null) {
            try {
                /**
                 * cnnct.close() throws a SQLException, but we cannot
                 * recover at this point
                 */
                cnnct.close();
            } catch (SQLException sqlEx) {
                // do nothing
            }
        }
    }
}

/**
 * destroyDBTable()
 *
 * @aim Remove the database table
 */
public void destroyDBTable() {
    Connection cnnct = null;
    Statement stmnt = null;

    try {
```

```
        // get connection
        cnnct = getConnection();
        // get statement
        stmnt = cnnct.createStatement();

        // execute action query to destroy a data table
        stmnt.execute("DROP TABLE " + DB_TABLE);
    } catch (SQLException ex) {
        // do nothing
    } catch (IOException ex) {
        // do nothing
    } finally {
        // close Statement object
        if (stmnt != null) {
            try {
                stmnt.close();
            } catch (SQLException e) {
                // do nothing
            }
        }

        // close Connection object
        if (cnnct != null) {
            try {
                cnnct.close();
            } catch (SQLException sqlEx) {
                // do nothing
            }
        }
    }
```

```
}

/**
 * addRecord()
 *
 * @aim Add a record into the database table
 */
public void addRecords(ArrayList<Customer> custList) {

    Connection cnnct = null;

    // create a PreparedStatement object
    PreparedStatement pStmnt = null;

    try {
        // get connection
        cnnct = getConnection();

        // precompiled query statement
        String preQueryStatement = "INSERT INTO " + DB_TABLE
                + " VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

        for (Customer cust : custList) {

            // get statement
            pStmnt = cnnct.prepareStatement(preQueryStatement);

            // set individual parameters at corresponding positions
            pStmnt.setString(1, cust.getTransaction_id());
            pStmnt.setString(2, cust.getFirstname());
```

```
            pStmnt.setString(3, cust.getLastname());
            pStmnt.setString(4, cust.getEmail());
            pStmnt.setString(5, cust.getAddress());
            pStmnt.setString(6, cust.getSuburb());
            pStmnt.setString(7, cust.getPostcode());
            pStmnt.setString(8, cust.getPhone());
            pStmnt.setString(9, cust.getProduct_name());
            pStmnt.setString(10, cust.getQuantity());
            pStmnt.setString(11, cust.getComment());
            pStmnt.setString(12, cust.getCredit_card_name());
            pStmnt.setString(13, cust.getCredit_card_number());
            pStmnt.setString(14, cust.getCredit_card_expiry_date());
            pStmnt.setString(15, cust.getCredit_card_CVV());




            int rowCount = pStmnt.executeUpdate();

            /*
             * rowCount should be 1 because 1 record is added
             *
             * throws exception if not
             */
            if (rowCount == 0) {
               throw new SQLException("Cannot insert records!");
            }
         }
      } catch (SQLException ex) {
         // do nothing
      } catch (IOException ex) {
```

```
            // do nothing
        } finally {
            // close Prepared Statement object
            if (pStmnt != null) {
                try {
                    pStmnt.close();
                } catch (SQLException e) {
                    // do nothing
                }
            }

            // close Connection object
            if (cnnct != null) {
                try {
                    cnnct.close();
                } catch (SQLException sqlEx) {
                    // do nothing
                }
            }
        }
    }
}
```

MakeCustomerDB:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package ragib.jdbc;
```

```
import java.security.MessageDigest;
import java.util.ArrayList;

/**
 *
 * @author Ragib
 */
public class MakeCustomerDB {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      // the database object to access the actual database
      CustomerDB db = new CustomerDB();

      // make sure no name conflicts
      try {
         db.destroyDBTable();
      } catch (Exception ex) {
      }

      // create the database table
      System.out.println("Create an empty database table Customer");
      db.createDBTable();

      System.out.println("Add several static records in the database table");

      // prepare data
```

```
    Customer cust001 = new Customer("1", "Ragib", "Tester", "Tester@google.com", "378 Riversdale Road", "Hawthorn", "3123", "1234567890", "Stark", "5", "Tester Comment", "TesterCredit Card", "1234567890123456", "0320", "376");


    // prepare list
    ArrayList<Customer> custList = new ArrayList<>();
    custList.add(cust001);


    // add data to db
    db.addRecords(custList);


  }


}
```

Remote interface and DTO:

CustomerDTO.java:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package entity;


import java.io.Serializable;


public class CustomerDTO implements Serializable {


  String transaction_id;
  String firstname;
  String lastname;
  String email;
```

```
    String address;
    String suburb;
    String postcode;
    String phone;
    String product_name;
    String quantity;
    String comment;
    String credit_card_name;
    String credit_card_number;
    String credit_card_expiry_date;
    String credit_card_CVV;


    public CustomerDTO(String transaction_id, String firstname, String lastname, String email, String address, String
suburb, String postcode, String phone, String product_name, String quantity, String comment, String cred-
it_card_name, String credit_card_number, String credit_card_expiry_date, String credit_card_CVV) {
        this.transaction_id = transaction_id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.email = email;
        this.address = address;
        this.suburb = suburb;
        this.postcode = postcode;
        this.phone = phone;
        this.product_name = product_name;
        this.quantity = quantity;
        this.comment = comment;
        this.credit_card_name = credit_card_name;
        this.credit_card_number = credit_card_number;
        this.credit_card_expiry_date = credit_card_expiry_date;
        this.credit_card_CVV = credit_card_CVV;
    }
```

```
public String getTransaction_id() {
    return transaction_id;
}

public String getFirstname() {
    return firstname;
}

public String getLastname() {
    return lastname;
}

public String getEmail() {
    return email;
}

public String getAddress() {
    return address;
}

public String getSuburb() {
    return suburb;
}

public String getPostcode() {
    return postcode;
}

public String getPhone() {
```

```
    return phone;
  }

  public String getProduct_name() {
    return product_name;
  }

  public String getQuantity() {
    return quantity;
  }

  public String getComment() {
    return comment;
  }

  public String getCredit_card_name() {
    return credit_card_name;
  }

  public String getCredit_card_number() {
    return credit_card_number;
  }

  public String getCredit_card_expiry_date() {
    return credit_card_expiry_date;
  }

  public String getCredit_card_CVV() {
    return credit_card_CVV;
  }
```

```
}
```

CustomerTransactionManagementRemote.java:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package session;

import javax.ejb.Remote;
import entity.CustomerDTO;

@Remote
public interface CustomerTransactionManagementRemote {

    boolean hasCustomerTransaction(String transaction_id);

    boolean addCusomterTransaction(CustomerDTO customerDTO);

    CustomerDTO getCustomerTransactionDetails(String transaction_id);



}
```

Ejb files:

Customer.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
```

```
 */

package entity;


import java.io.Serializable;

import javax.persistence.Basic;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

import javax.xml.bind.annotation.XmlRootElement;


@Entity

@Table(name = "Ragib_Customers", catalog = "", schema = "APP")

@XmlRootElement

@NamedQueries({

   @NamedQuery(name = "Customer.findAll", query = "SELECT c FROM Customer c"),

   @NamedQuery(name = "Customer.findByTransaction_id", query = "SELECT c FROM Customer c WHERE
c.transaction_id = :transaction_id"),

   @NamedQuery(name = "Customer.findByFirstname", query = "SELECT c FROM Customer c WHERE c.firstname =
:firstname"),

   @NamedQuery(name = "Customer.findByLastname", query = "SELECT c FROM Customer c WHERE c.lastname =
:lastname"),

   @NamedQuery(name = "Customer.findByEmail", query = "SELECT c FROM Customer c WHERE c.email = :email"),

   @NamedQuery(name = "Customer.findByAddress", query = "SELECT c FROM Customer c WHERE c.address =
:address"),

   @NamedQuery(name = "Customer.findBySuburb", query = "SELECT c FROM Customer c WHERE c.suburb =
:suburb"),

   @NamedQuery(name = "Customer.findByPostcode", query = "SELECT c FROM Customer c WHERE c.postcode =
:postcode"),


   @NamedQuery(name = "Customer.findByPhone", query = "SELECT c FROM Customer c WHERE c.phone = :phone"),
```

```
    @NamedQuery(name = "Customer.findByProduct_Name", query = "SELECT c FROM Customer c WHERE
c.product_name = :product_name"),

    @NamedQuery(name = "Customer.findByQuantity", query = "SELECT c FROM Customer c WHERE c.quantity =
:quantity"),

    @NamedQuery(name = "Customer.findByComment", query = "SELECT c FROM Customer c WHERE c.comment =
:comment"),

    @NamedQuery(name = "Customer.findByCredit_Card_Name", query = "SELECT c FROM Customer c WHERE
c.credit_card_name = :credit_card_name"),

    @NamedQuery(name = "Customer.findByCredit_Card_Number", query = "SELECT c FROM Customer c WHERE
c.credit_card_number = :credit_card_number"),

    @NamedQuery(name = "Customer.findByCredit_Card_Expiry_Date", query = "SELECT c FROM Customer c WHERE
c.credit_card_expiry_date = :credit_card_expiry_date"),

    @NamedQuery(name = "Customer.findByCredit_Card_CVV", query = "SELECT c FROM Customer c WHERE
c.credit_card_CVV = :credit_card_CVV")})

public class Customer implements Serializable {


    private static final long serialVersionUID = 1L;

    @Id

    @Basic(optional = false)

    @Column(name = "TRANSACTION_ID")

    private String transaction_id;

    @Column(name = "FIRSTNAME")

    private String firstname;

    @Column(name = "LASTNAME")

    private String lastname;

    @Column(name = "EMAIL")

    private String email;

    @Column(name = "ADDRESS")

    private String address;

    @Column(name = "SUBURB")

    private String suburb;

    @Column(name = "POSTCODE")

    private String postcode;
```

```
    @Column(name = "PHONE")
    private String phone;
    @Column(name = "PRODUCT_NAME")
    private String product_name;
    @Column(name = "QUANTITY")
    private String quantity;
    @Column(name = "COMMENT")
    private String comment;
    @Column(name = "CREDIT_CARD_NAME")
    private String credit_card_name;
    @Column(name = "CREDIT_CARD_NUMBER")
    private String credit_card_number;
    @Column(name = "CREDIT_CARD_EXPIRY_DATE")
    private String credit_card_expiry_date;
    @Column(name = "CREDIT_CARD_CVV")
    private String credit_card_CVV;

    public Customer() {
    }

    public Customer(String transaction_id) {
        this.transaction_id = transaction_id;
    }

    public Customer(String transaction_id,
            String firstname,
            String lastname,
            String email,
            String address,
            String suburb,
```

```
        String postcode,
        String phone,
        String product_name,
        String quantity,
        String comment,
        String credit_card_name,
        String credit_card_number,
        String credit_card_expiry_date,
        String credit_card_CVV) {
    this.transaction_id = transaction_id;
    this.firstname = firstname;
    this.lastname = lastname;
    this.email = email;
    this.address = address;
    this.suburb = suburb;

    this.postcode = postcode;
    this.phone = phone;

    this.product_name = product_name;
    this.quantity = quantity;
    this.comment = comment;
    this.credit_card_name = credit_card_name;
    this.credit_card_number = credit_card_number;
    this.credit_card_expiry_date = credit_card_expiry_date;
    this.credit_card_CVV = credit_card_CVV;
  }

  public String getTransaction_id() {
    return transaction_id;
```

```
    }

    public void setTransaction_id(String transaction_id) {
        this.transaction_id = transaction_id;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
```

```
    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getSuburb() {
        return suburb;
    }

    public void setSuburb(String suburb) {
        this.suburb = suburb;
    }

    public String getPostcode() {
        return postcode;
    }

    public void setPostcode(String postcode) {
        this.postcode = postcode;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
```

```
    }

    public String getProduct_name() {
        return product_name;
    }

    public void setProduct_name(String product_name) {
        this.product_name = product_name;
    }

    public String getQuantity() {
        return quantity;
    }

    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }

    public String getComment() {
        return comment;
    }

    public void setComment(String comment) {
        this.comment = comment;
    }

    public String getCredit_card_name() {
        return credit_card_name;
```

```
    }

    public void setCredit_card_name(String credit_card_name) {
        this.credit_card_name = credit_card_name;
    }

    public String getCredit_card_number() {
        return credit_card_number;
    }

    public void setCredit_card_number(String credit_card_number) {
        this.credit_card_number = credit_card_number;
    }

    public String getCredit_card_expiry_date() {
        return credit_card_expiry_date;
    }

    public void setCredit_card_expiry_date(String credit_card_expiry_date) {
        this.credit_card_expiry_date = credit_card_expiry_date;
    }

    public String getCredit_card_CVV() {
        return credit_card_CVV;
    }

    public void setCredit_card_CVV(String credit_card_CVV) {
        this.credit_card_CVV = credit_card_CVV;
    }
```

```
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Customer)) {
            return false;
        }
        Customer other = (Customer) object;
        if ((this.transaction_id == null && other.transaction_id != null) || (this.transaction_id != null &&
!this.transaction_id.equals(other.transaction_id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "persistence.entity.Customer[ transaction_id=" + transaction_id + " ]";
    }

}
```

CustomerFacade:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package session;

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
```

```
import entity.Customer;



@Stateless
public class CustomerFacade implements CustomerFacadeLocal {

    @PersistenceContext(unitName = "Ragib-ejbPU")
    private EntityManager em;

    public CustomerFacade() {
    }

    private void create(Customer entity) {
        em.persist(entity);
    }

    private void edit(Customer entity) {
        em.merge(entity);
    }

    private void remove(Customer entity) {
        em.remove(em.merge(entity));
    }

    /**
     *
     * @param id
     * @return
     */
    @Override
```

```
    public Customer find(String id) {
        return em.find(Customer.class, id);
    }

    /**
     * checks whether an customer exist using transaction_id
     *
     * @param transaction_id
     * @return true if exist, false otherwise
     */
    @Override
    public boolean hasCustomerTransaction(String transaction_id) {
        return (find(transaction_id) != null);
    }

    /**
     * add a customer's transaction to the system
     *
     * @param customer
     * @return true if addition is successful, false otherwise
     */
    @Override
    public boolean addCustomerTransaction(Customer customer) {
        // check again - just to play it safe
        Customer c = find(customer.getTransaction_id());

        if (c != null) {
            // could not add one
            return false;
        }
```

```
        create(customer);

        return true;
    }

}
```

CustomerFacadeLocal:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package session;


import javax.ejb.Local;
import entity.Customer;


@Local
public interface CustomerFacadeLocal {

    Customer find(String id);

    boolean hasCustomerTransaction(String transaction_id);

    boolean addCustomerTransaction(Customer customer);
```

```
}
```

CustomerTranasactionManagementLocal.java:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package session;

import javax.ejb.EJB;
import javax.ejb.Stateless;
import entity.Customer;
import entity.CustomerDTO;
import javax.annotation.security.DeclareRoles;
import javax.annotation.security.PermitAll;
import javax.annotation.security.RolesAllowed;

@DeclareRoles({"Ragib-ADMIN"})
@Stateless
public class CustomerTransactionManagement implements CustomerTransactionManagementRemote {

    @EJB
    private CustomerFacadeLocal customerFacade;

    private Customer customerDTO2Entity(CustomerDTO customerDTO) {
        if (customerDTO == null) {
            // just in case
            return null;
```

```
        }

        String transaction_id = customerDTO.getTransaction_id();
        String firstname = customerDTO.getFirstname();
        String lastname = customerDTO.getLastname();
        String email = customerDTO.getEmail();
        String address = customerDTO.getAddress();
        String suburb = customerDTO.getSuburb();
        String postcode = customerDTO.getPostcode();
        String phone = customerDTO.getPhone();
        String product_name = customerDTO.getProduct_name();
        String quantity = customerDTO.getQuantity();
        String comment = customerDTO.getComment();
        String credit_card_name = customerDTO.getCredit_card_name();
        String credit_card_number = customerDTO.getCredit_card_number();
        String credit_card_expiry_date = customerDTO.getCredit_card_expiry_date();
        String credit_card_CVV = customerDTO.getCredit_card_CVV();

        Customer customer = new Customer(transaction_id, firstname, lastname, email, address, suburb, postcode,
phone, product_name, quantity, comment, credit_card_name, credit_card_number, credit_card_expiry_date, cred-
it_card_CVV);

        return customer;
    }

    private CustomerDTO customerEntity2DTO(Customer customer) {
        if (customer == null) {
            // just in case
            return null;
        }
```

```
    CustomerDTO customerDTO = new CustomerDTO(
            customer.getTransaction_id(),
            customer.getFirstname(),
            customer.getLastname(),
            customer.getEmail(),
            customer.getAddress(),
            customer.getSuburb(),
            customer.getPostcode(),
            customer.getPhone(),
            customer.getProduct_name(),
            customer.getQuantity(),
            customer.getComment(),
            customer.getCredit_card_name(),
            customer.getCredit_card_number(),
            customer.getCredit_card_expiry_date(),
            customer.getCredit_card_CVV()
    );

    return customerDTO;
}

/**
 * check whether the customer transaction is in the system
 *
 * @param transaction_id
 * @return true if the customer transaction is in the system, false
 * otherwise
 */
@Override
@PermitAll   //as customers need to check before adding thier transaction
```

```
public boolean hasCustomerTransaction(String transaction_id) {
    return customerFacade.hasCustomerTransaction(transaction_id);
}

/**
 * add a customer transaction to the system
 *
 * @param customerDTO
 * @return true if addition is successful, false otherwise
 */
@Override
@PermitAll  //as cstomers need to be able to add thier transaction
public boolean addCusomterTransaction(CustomerDTO customerDTO) {

    if (customerDTO == null) {
        // just in case
        return false;
    }

    // check customer exist?
    if (hasCustomerTransaction(customerDTO.getTransaction_id())) {
        // customer exists, cannot add one
        return false;
    }

    // customer not exist
    // convert to entity
    Customer customer = this.customerDTO2Entity(customerDTO);
    // add one
    return customerFacade.addCustomerTransaction(customer);
```

```
    }


    /**
     * get customer Transaction details and use a DTO to transmit the details
     *
     * @param transaction_id
     * @return a DTO containing the information of the customer if exists, null
     * otherwise
     */




}
```

Webpage:

EnquiryNew.xhtml:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <head>

        <title>Ragib Televisions (Enquiry)</title>
    </head>
    <body>
```

```
<h1 id="bring_down">Enquiry</h1>

<article id= "Faq_no_b">

  <section>
    <h:form id="payment">
      <table>
        <tbody>
          <tr>

            <td>
              <h:outputLabel value="Transaction ID:"/>
            </td>
            <td>
              <h:inputText id="transaction_id"
                    maxlength="5"

                    value="#{myCustomerManagedBean.transaction_id}"
                    >

              </h:inputText>
            </td>

          </tr>

          <tr>

            <td>
              <h:outputLabel value="First Name:"/>
            </td>
```

```
    <td>
      <h:inputText id="firstname"
                   maxlength="20"

                   value="#{myCustomerManagedBean.firstname}"
                   >

      </h:inputText>
    </td>

  </tr>

  <tr>

    <td>
      <h:outputLabel value="Last Name:"/>
    </td>
    <td>
      <h:inputText id="lastname"
                   maxlength="20"

                   value="#{myCustomerManagedBean.lastname}"
                   >

      </h:inputText>
    </td>

  </tr>

  <tr>
```

```
    <td>
      <h:outputLabel value="Email:"/>
    </td>
    <td>
      <h:inputText id="email"
                   maxlength="20"

                   value="#{myCustomerManagedBean.email}"
                   >


      </h:inputText>
    </td>

  </tr>

  <tr>

    <td>
      <h:outputLabel value="Address:"/>
    </td>
    <td>
      <h:inputText id="address"
                   maxlength="20"

                   value="#{myCustomerManagedBean.address}">

      </h:inputText>
      <!--no need regex for address as it can have special characters in them-->
```

```
    </td>

</tr>
<tr>

    <td>
        <h:outputLabel value="Suburb:"/>
    </td>
    <td>
        <h:inputText id="suburb"
                     maxlength="20"

                     value="#{myCustomerManagedBean.suburb}">

        </h:inputText>
        <!--no need regex for suburb as it can have special characters in them-->
    </td>

</tr>

<tr>

    <td>
        <h:outputLabel value="Postcode:"/>
    </td>
    <td>
        <h:inputText id="postcode"
                     maxlength="20"

                     value="#{myCustomerManagedBean.postcode}"
```

```
                >


        </h:inputText>
    </td>

</tr>

<tr>

    <td>
        <h:outputLabel value="Phone Number:"/>
    </td>
    <td>
        <h:inputText id="phone"
                maxlength="20"

                value="#{myCustomerManagedBean.phone}"
                >

        </h:inputText>
    </td>

</tr>

<tr>

    <td>
        <h:outputLabel value="Product Name:"/>
    </td>
```

```
    <td>
      <h:inputText id="product_name"
                   maxlength="20"

                   value="#{myCustomerManagedBean.product_name}"
                   >

      </h:inputText>
    </td>

  </tr>

  <tr>

    <td>
      <h:outputLabel value="Quantity:"/>
    </td>
    <td>
      <h:inputText id="quantity"
                   maxlength="20"

                   value="#{myCustomerManagedBean.quantity}"
                   >
      </h:inputText>
    </td>

  </tr>

  <tr>
```

```
    <td>
      <h:outputLabel value="Comment:"/>
    </td>
    <td>
      <h:inputText id="comment"
                   maxlength="20"
                   value="#{myCustomerManagedBean.comment}">

        <!--no need regex for comment as it can have special characters in them-->
      </h:inputText>
    </td>

  </tr>



  <tr>
    <td>
      <h:outputLabel value="Credit Card Name:"/>
    </td>
    <td>
      <h:inputText id="credit_card_name"
                   maxlength="40"

                   value="#{myCustomerManagedBean.credit_card_name}"
                   >

      </h:inputText>
    </td>

  </tr>
```

```
<tr>
  <td>
    <h:outputLabel value="Credit Card Number"/>
  </td>
  <td>
    <h:inputText id="credit_card_number"
          maxlength="16"

          value="#{myCustomerManagedBean.credit_card_number}"
          >

    </h:inputText>
  </td>

</tr>

<tr>
  <td>
    <h:outputLabel value="Credit Card Expiration Date"/>
  </td>
  <td>
    <h:inputText id="credit_card_expiry_date"
          maxlength="4"

          value="#{myCustomerManagedBean.credit_card_expiry_date}"
          >

    </h:inputText>
  </td>
```

```
        </tr>

        <tr>
          <td>
            <h:outputLabel value="Credit Card CVV"/>
          </td>
          <td>
            <h:inputText id="credit_card_CVV"
                    maxlength="3"

                    value="#{myCustomerManagedBean.credit_card_CVV}"
                    >

            </h:inputText>
          </td>

        </tr>



      </tbody>
    </table>

    <h:commandButton id="submit" value="Submit"
                action="#{myCustomerManagedBean.addCustomerTransaction()}"/>



  </h:form>
```

```
        </section>


      </article>


  </body>
</html>
```

MyCustomerManagedBean:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package web;

import java.io.Serializable;
import javax.ejb.EJB;
import javax.enterprise.context.Conversation;
import javax.enterprise.context.ConversationScoped;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.component.UIInput;
import javax.faces.context.FacesContext;
import javax.faces.validator.ValidatorException;
import javax.inject.Inject;
import javax.inject.Named;
import entity.CustomerDTO;
```

```
import session.CustomerTransactionManagementRemote;

@Named(value = "myCustomerManagedBean")
@ConversationScoped
public class MyCustomerManagedBean implements Serializable {

    @Inject
    private Conversation conversation;
    @EJB
    private CustomerTransactionManagementRemote customerTransactionManagement;

    private String transaction_id;
    private String firstname;
    private String lastname;
    private String email;
    private String address;
    private String suburb;
    private String postcode;
    private String phone;

    private String product_name;
    private String quantity;
    private String comment;
    private String credit_card_name;
    private String credit_card_number;
    private String credit_card_expiry_date;
    private String credit_card_CVV;

    /**
     * Creates a new instance of MyCustomerManagedBean
```

```
 */
public MyCustomerManagedBean() {
    transaction_id = null;
    firstname = null;
    lastname = null;
    email = null;
    address = null;
    suburb = null;
    postcode = null;
    phone = null;
    product_name = null;
    quantity = null;
    comment = null;
    credit_card_name = null;
    credit_card_number = null;
    credit_card_expiry_date = null;
    credit_card_CVV = null;
}

public String getFirstname() {
    return firstname;
}

public void setFirstname(String firstname) {
    this.firstname = firstname;
}

public String getLastname() {
    return lastname;
}
```

```
public void setLastname(String lastname) {
    this.lastname = lastname;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

public String getTransaction_id() {
    return transaction_id;
}

public void setTransaction_id(String transaction_id) {
    this.transaction_id = transaction_id;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getPhone() {
```

```
    return phone;
  }

  public void setPhone(String phone) {
    this.phone = phone;
  }

  public String getSuburb() {
    return suburb;
  }

  public void setSuburb(String suburb) {
    this.suburb = suburb;
  }

  public String getPostcode() {
    return postcode;
  }

  public void setPostcode(String postcode) {
    this.postcode = postcode;
  }

  public String getProduct_name() {
    return product_name;
  }

  public void setProduct_name(String product_name) {
    this.product_name = product_name;
  }
```

```
public String getQuantity() {
    return quantity;
}

public void setQuantity(String quantity) {
    this.quantity = quantity;
}

public String getComment() {
    return comment;
}

public void setComment(String comment) {
    this.comment = comment;
}

public String getCredit_card_name() {
    return credit_card_name;
}

public void setCredit_card_name(String credit_card_name) {
    this.credit_card_name = credit_card_name;
}

public String getCredit_card_number() {
    return credit_card_number;
}

public void setCredit_card_number(String credit_card_number) {
```

```
    this.credit_card_number = credit_card_number;
  }

  public String getCredit_card_expiry_date() {
    return credit_card_expiry_date;
  }

  public void setCredit_card_expiry_date(String credit_card_expiry_date) {
    this.credit_card_expiry_date = credit_card_expiry_date;
  }

  public String getCredit_card_CVV() {
    return credit_card_CVV;
  }

  public void setCredit_card_CVV(String credit_card_CVV) {
    this.credit_card_CVV = credit_card_CVV;
  }

  public void startConversation() {
    conversation.begin();
  }

  public void endConversation() {
    conversation.end();
  }

  public String addCustomerTransaction() {
    startConversation();
    // check transaction_id is null
```

```
        if (isNull(transaction_id)) {
            return "debug";
        }

        // all information seems to be valid
        // try add the customer transaction
        CustomerDTO customerDTO = new CustomerDTO(transaction_id, firstname, lastname, email, address, suburb, postcode, phone, product_name, quantity, comment, credit_card_name, credit_card_number, credit_card_expiry_date, credit_card_CVV);
        boolean result = customerTransactionManagement.addCusomterTransaction(customerDTO);
        if (result) {
            endConversation();
            return "success";

        } else {
            return "failure";
        }

    }

    private boolean isNull(String s) {
        return (s == null);
    }

}
```

Faces-config.xml:

```
<?xml version='1.0' encoding='UTF-8'?>

<!-- ========== FULL CONFIGURATION FILE ================================== -->
```

```
<faces-config version="2.1"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-facesconfig_2_1.xsd">
    <navigation-rule>
      <description>form submission new</description>
      <from-view-id>/EnquiryNew.xhtml</from-view-id>
      <navigation-case>
        <from-action>#{myCustomerManagedBean.addCustomerTransaction()}</from-action>
        <from-outcome>success</from-outcome>
        <to-view-id>/EnquiryNew.xhtml</to-view-id>
      </navigation-case>
   </navigation-rule>
</faces-config>
```

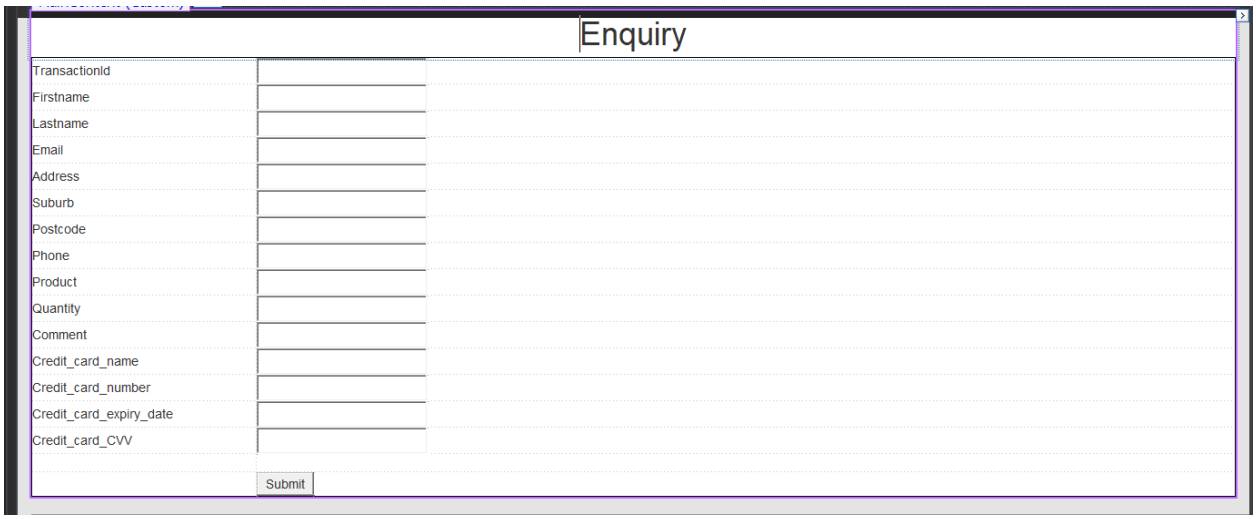
Web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
   <context-param>
      <param-name>javax.faces.PROJECT_STAGE</param-name>
      <param-value>Development</param-value>
   </context-param>
   <servlet>
      <servlet-name>Faces Servlet</servlet-name>
      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
      <servlet-name>Faces Servlet</servlet-name>
```

```
    <url-pattern>*.xhtml</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

b)Using .NET:

Webpage (design aspect)




```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master" Au-
toEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="Ragib_Television6._Default" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

    <div>
        <div style="font-size:x-large" align ="center"> Enquiry</div>

        <table cellpadding="2" style="width: 100%; border: 1px solid #000000">
            <tr>
                <td style="width: 247px">TransactionId</td>
                <td>
                    <asp:TextBox ID="TextBox1" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Firstname</td>
                <td>
                    <asp:TextBox ID="TextBox2" runat="server" Font-
Size="Small"></asp:TextBox>
```

```
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Lastname</td>
                <td>
                    <asp:TextBox ID="TextBox3" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Email</td>
                <td>
                    <asp:TextBox ID="TextBox4" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Address</td>
                <td>
                    <asp:TextBox ID="TextBox5" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Suburb</td>
                <td>
                    <asp:TextBox ID="TextBox6" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Postcode</td>
                <td>
                    <asp:TextBox ID="TextBox7" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Phone</td>
                <td>
                    <asp:TextBox ID="TextBox8" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="height: 20px; width: 247px">Product</td>
                <td style="height: 20px">
                    <asp:TextBox ID="TextBox9" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Quantity</td>
                <td>
                    <asp:TextBox ID="TextBox10" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Comment</td>
                <td>
                    <asp:TextBox ID="TextBox11" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
```

```
            <tr>
                <td style="width: 247px">Credit_card_name</td>
                <td>
                    <asp:TextBox ID="TextBox12" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Credit_card_number</td>
                <td>
                    <asp:TextBox ID="TextBox13" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Credit_card_expiry_date</td>
                <td>
                    <asp:TextBox ID="TextBox14" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px">Credit_card_CVV</td>
                <td>
                    <asp:TextBox ID="TextBox15" runat="server" Font-
Size="Small"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td style="width: 247px"> </td>
                <td> </td>
            </tr>
            <tr>
                <td style="width: 247px"> </td>
                <td>
                    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click"
Text="Submit" />
                </td>
            </tr>
        </table>


    </div>

</asp:Content>

//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace Ragib_Television6
{


    public partial class _Default
    {

        /// <summary>
        /// TextBox1 control.
        /// </summary>
```

```
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox1;

        /// <summary>
        /// TextBox2 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox2;

        /// <summary>
        /// TextBox3 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox3;

        /// <summary>
        /// TextBox4 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox4;

        /// <summary>
        /// TextBox5 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox5;

        /// <summary>
        /// TextBox6 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox6;

        /// <summary>
        /// TextBox7 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox7;

        /// <summary>
        /// TextBox8 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
```

```
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox8;

        /// <summary>
        /// TextBox9 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox9;

        /// <summary>
        /// TextBox10 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox10;

        /// <summary>
        /// TextBox11 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox11;

        /// <summary>
        /// TextBox12 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox12;

        /// <summary>
        /// TextBox13 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox13;

        /// <summary>
        /// TextBox14 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.TextBox TextBox14;

        /// <summary>
        /// TextBox15 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
```

```
        protected global::System.Web.UI.WebControls.TextBox TextBox15;

        /// <summary>
        /// Button1 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Button Button1;
    }
}
```

Webpage method call code: (default.cs)

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Ragib_Television6
{
    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            ExecuteCommands command = new ExecuteCommands();

            command.ExecuteInsert(TextBox1.Text, TextBox2.Text, TextBox3.Text, Text-
Box4.Text, TextBox5.Text, TextBox6.Text, TextBox7.Text, TextBox8.Text, TextBox9.Text, Text-
Box10.Text, TextBox11.Text, TextBox12.Text, TextBox13.Text, TextBox14.Text, TextBox15.Text);


            ScriptManager.RegisterStartupScript(this, this.GetType(), "script",
"alert('Successfully Inserted')", true);

        }
    }

}
```

ExecuteCommands.cs:

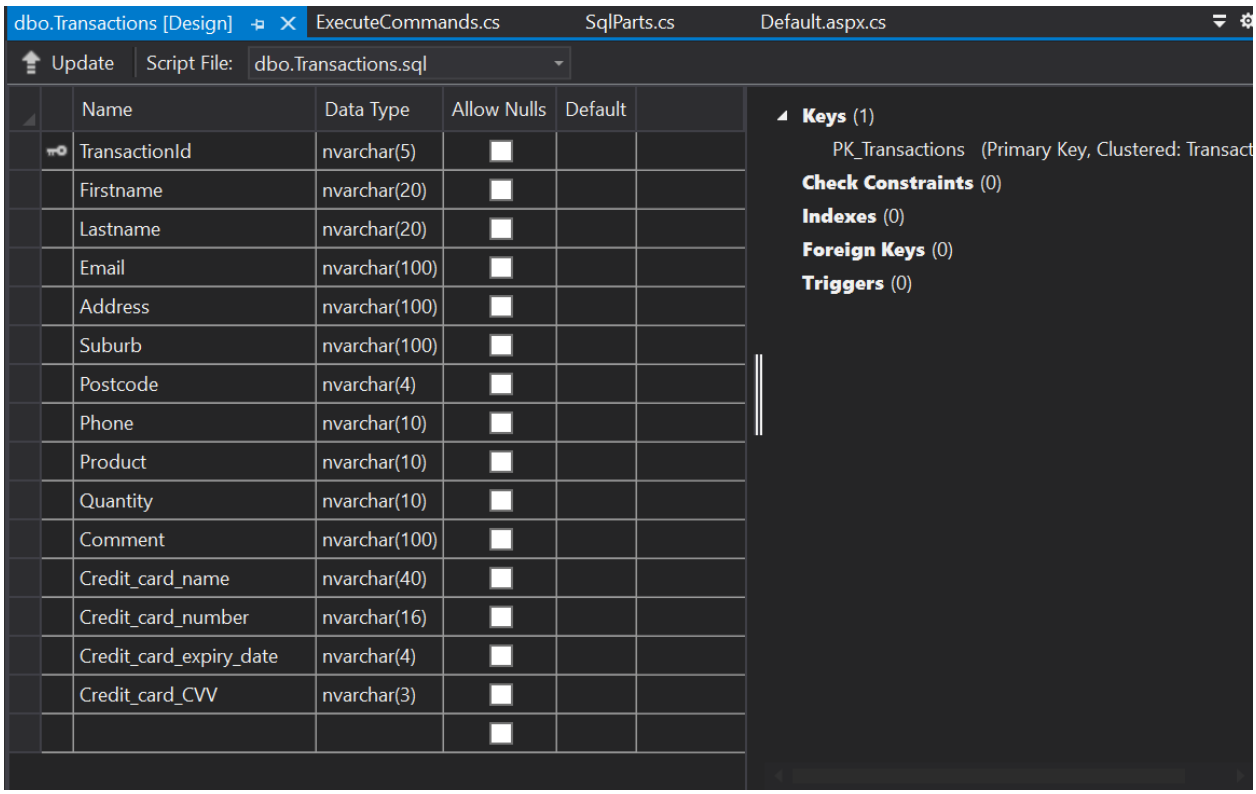
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Ragib_Television6
{
    public class ExecuteCommands
    {
        private SqlParts _sqlStuff;
```

```
        public ExecuteCommands()
        {
            _sqlStuff = new SqlParts();
        }

        public void ExecuteInsert(string a1, string a2, string a3, string a4, string a5,
string a6, string a7, string a8, string a9, string a10, string a11, string a12, string a13,
string a14, string a15)
        {
            _sqlStuff.Insert(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14,
a15);


        }
    }

}
```

SqlParts.cs:

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;

namespace Ragib_Television6
{
    public class SqlParts
    {
        SqlConnection _con = new SqlConnection("Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=TransactionDb2;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubne
tFailover=False");

        SqlCommand _comm;
        public SqlParts()
        {
            _con.Open();
        }

        public void Insert(string a1, string a2, string a3, string a4, string a5, string a6,
string a7, string a8, string a9, string a10, string a11, string a12, string a13, string a14,
string a15) {

            _comm = new SqlCommand("Insert into Transactions values ('" + a1 + "','" + a2 +
"',    '" + a3 + "', '" + a4 + "', '" + a5 + "', '" + a6 + "', '" + a7 + "', '" + a8 + "', '"
+ a9 + "',    '" + a10 + "',       '" + a11 + "',       '" + a12 + "',       '" + a13 + "',
       '" + a14 + "',       '" + a15 + "')", _con);
            _comm.ExecuteNonQuery();
            _con.Close();

        }


    }

}
```

Transactions table in transactionsDb2 database:




```
CREATE TABLE [dbo].[Transactions] (
    [TransactionId]           NVARCHAR (5)   NOT NULL,
    [Firstname]               NVARCHAR (20)  NOT NULL,
    [Lastname]                NVARCHAR (20)  NOT NULL,
    [Email]                   NVARCHAR (100) NOT NULL,
    [Address]                 NVARCHAR (100) NOT NULL,
    [Suburb]                  NVARCHAR (100) NOT NULL,
    [Postcode]                NVARCHAR (4)   NOT NULL,
    [Phone]                   NVARCHAR (10)  NOT NULL,
    [Product]                 NVARCHAR (10)  NOT NULL,
    [Quantity]                NVARCHAR (10)  NOT NULL,
    [Comment]                 NVARCHAR (100) NOT NULL,
    [Credit_card_name]        NVARCHAR (40)  NOT NULL,
    [Credit_card_number]      NVARCHAR (16)  NOT NULL,
    [Credit_card_expiry_date] NVARCHAR (4)   NOT NULL,
    [Credit_card_CVV]         NVARCHAR (3)   NOT NULL,
    CONSTRAINT [PK_Transactions] PRIMARY KEY CLUSTERED ([TransactionId] ASC)
);
```

References:

[1] Microsoft Visual Studio. "SQL Server Data Tools for Visual Studio". Microsoft Visual Studio. https://visualstudio.microsoft.com/vs/features/ssdt/ (Accessed 05/30/2022).

[2] Microsoft Visual Studio. "Mordern Web Tooling". Microsoft Visual Studio. https://visualstudio.microsoft.com/vs/features/web/ (Accessed 05/30/2022).

[3] Microsoft Documentation. ".NET Documentation". Microsoft Visual Studio. https://docs.microsoft.com/en-us/dotnet/ (Accessed 05/30/2022).

[4] Firefox Source Docs. "Network Monitor". Firefox Source Docs. https://firefox-source-docs.mozilla.org/devtools-user/network_monitor/ (Accessed 05/30/2022).

[5] Human Benchmark. "Reaction Time Test". Human Benchmark. https://humanbenchmark.com/tests/reactiontime (Accessed 05/30/2022).

[6] Quora. "How is .net better than Java". Quora. https://www.quora.com/How-is-net-better-than-Java (Accessed 05/30/2022).

[7] L. White. "Reasons why dot net is better than Java". Codersera. https://codersera.com/blog/reasons-why-dot-net-is-better-than-java/ (Accessed 05/30/2022).

[8] M. Kaczorowski . ".NET vs Java: Which Technology Is Better For Software Development?". Ideamotive. https://www.ideamotive.co/blog/dotnet-vs-java-which-technology-is-better-for-software-development (Accessed 05/30/2022).