Student ID: _108758____

Full Names: _____Semhal Redda____

Software Engineering (CS425)

(November 2018)

Professor: O. Kalu

Final Integration Exercise

1. The exam duration is 2 hours.
2. The exam is computer-based; so you may use a computer for both the coding and theory parts.
3. Make sure to switch-off your cell-phones or simply turn the ringer off.
4. **This exam is a copyrighted material and must not be copied or reproduced or transferred**.
5. You are expected to use an IDE or any Code Editor tool of your choice to implement your solutions for the questions in the Coding part. Upon completion, put your project(s), **<u>(source code only)</u>** in a single zip file named **FinalExam.zip**, including your completed/finished exam paper (i.e. this document – **in Microsoft Word or PDF format, only**), and submit to Sakai.

Type your answers to the theory questions in the following pages.

(CS425 - SWE)
(November 2018)
Final Examination (70 points)

Part I – Theory (True/False, Short answers, Multiple-choice questions):
(16 points)

1. (2 points) **Science of Consciousness**:

Answer the following question; giving 2 sentences in support of your answer:

State two (2) reasons why our twice daily practice of Transcendental Meditation can be considered to be an Agile technique for improving our brain performance?

Answer:

Yes, Transcendental meditation has a lot of similarities with Agile technique in terms of how to practice it on a daily basis or while someone wants to achieve a goal. One SCI point that can clearly express the outcome of Agile technique which is "Purification leads to progress." Because the Agile technique help us iteratively checking our project design progress and help us to refine the project and deliver it on time.

2. (2 points) Answer the following questions with True or False.

I. (1 point) The technique of "mocking" an object or component or service implies creating a "fake" one to serve as a placeholder for the actual object or component or service, for the purpose of implementing Unit tests, in isolation. True or False?

__________________True________________________

II. (1 point) In the RUP process model, during the Architecture Design activity, the class coded below will be globally visible to all other classes across the system. True or False?

_________False__________________________________

```
package edu.mum.cs.cs425.prodmgmt.global.publicc;

class GlobalPublicUtils {
        public static final long MAX_INVENTORY_COUNT = 10000L;

        // TODO
    }
```

3. (12 points) Give short answers to the following questions.

I. (2 points) Explain the difference between the following 2 JPA annotations. (You may give examples to illustrate your answer).

a. @Column(name="ProductNumber", **nullable=false**)
   private long productNumber;

b. **@NotNull**
   private long productNumber;

<u>Solution:</u>

a) It is the JPA way of declaring a column to be not null. This is intended for indicating database schema details.

b) It is a JSR 303, however, it intelligently picks up on these constraints and translates them into database constraints for you, so you get two for the price of one. It is intended for validation.

II. (2 points) With respect to relationship between two classes/entities, what do we mean by the term, Dependency. Give an example using code snippets and/or diagram.

Dependency can develop between objects while they are associated, aggregated or composed. This kind of relationship develops while one object invokes another object's functionality in order to accomplish some task. Any change in the called object may break the functionality of the caller.

<u>Example:</u>

```
import java.util.Date;

public class Customer {

public int custmoerId;

public  String name;
public int getCustmoerId() {

return custmoerId;

}

public void setCustmoerId(int custmoerId) {

this.custmoerId = custmoerId;

}

public String getName() {

return name;

}

public void setName(String name) {

his.name = name;

}         }

public class Order{

public int orderId;

public int OrderCustmerId;

public Date dateandTime;

public order(Customer customer) {

this.OrderCustmerId= customer.custmoerId;

}}
```

III. (2 points) With respect to relationship between classes/entities, what do we mean by the term, Composition? And how does it differ from an Aggregation? Give examples to illustrate your answer.

Solution:

Composition implies a relationship where the child cannot exist independent of the parent. Whereas, **Aggregation** implies a relationship where the child can exist independently of the parent.

Aggregation vs Composition

1. **Dependency:** Aggregation implies a relationship where the child **can exist independently** of the parent. For example, Bank and Employee, delete the Bank and the Employee still exist. whereas Composition implies a relationship where the child **cannot exist independent** of the parent. Example: Human and heart, heart don't exist separate to a Human
2. **Type of Relationship:** Aggregation relation is **"has-a"** and composition is **"part-of"** relation.
3. **Type of association:** Composition is a **strong** Association whereas Aggregation is a **weak** Association.

To see composition in details:

Composition is a restricted form of Aggregation in which two entities are highly dependent on each other.

- It represents **part-of** relationship.
- In composition, both the entities are dependent on each other.
- When there is a composition between two entities, the composed object **cannot exist** without the other entity.

Lets take example of Library.

```
// Java program to illustrate

// the concept of Composition

import java.io.*;
import java.util.*;


// class book
class Book
{


    public String title;
    public String author;


    Book(String title, String author)
    {


        this.title = title;
        this.author = author;
    }
}


// Libary class contains
// list of books.
class Library
{


    // reference to refer to list of books.
    private final List<Book> books;
    Library (List<Book> books)
    {
        this.books = books;
    }

    public List<Book> getTotalBooksInLibrary(){

       return books;
    }

}

// main method
class GFG
{
    public static void main (String[] args)
    {

        // Creating the Objects of Book class.
        Book b1 = new Book("EffectiveJ Java", "Joshua Bloch");
        Book b2 = new Book("Thinking in Java", "Bruce Eckel");
        Book b3 = new Book("Java: The Complete Reference", "Herbert Schildt");

        // Creating the list which contains the
        // no. of books.
        List<Book> books = new ArrayList<Book>();

        books.add(b1);

        books.add(b2);

        books.add(b3);


        Library library = new Library(books);


        List<Book> bks = library.getTotalBooksInLibrary();

        for(Book bk : bks){


            System.out.println("Title : " + bk.title + " and "

            +" Author : " + bk.author);

        }

    }

}
```

To see Aggregation in details:

It is a special form of Association where:

- It represents Has-A relationship.
- It is a unidirectional association i.e. a one way relationship. For example, department can have students but vice versa is not possible and thus unidirectional in nature.
- In Aggregation, both the entries can survive individually which means ending one entity will not effect the other entity

```
Java program to illustrate

//the concept of Aggregation.

import java.io.*;
import java.util.*;


// student class
class Student
{
   String name;
   int id ;
   String dept;


   Student(String name, int id, String dept)
   {


      this.name = name;
      this.id = id;
      this.dept = dept;


   }
}


/* Department class contains list of student
Objects. It is associated with student
class through its Object(s). */
class Department
{
    String name;

    private List<Student> students;

    Department(String name, List<Student> students)

    {



        this.name = name;

        this.students = students;



    }



    public List<Student> getStudents()

    {

        return students;

    }

}
```

IV. (6 points) For the MUMScheduling project, given below is Use-case specification for when a faculty who owns a Specialization track, reviews pending track enrollment requests.

Use-case: Faculty Specialization Track Owner Reviews Pending Track Enrollment Requests.

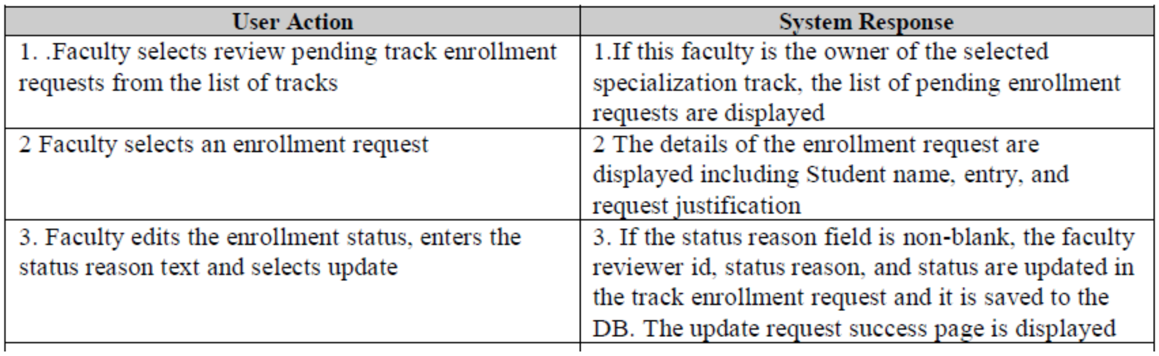
Brief Description:

This use case allows a faculty owner of a specialization track to review pending specialization track requests from students and to mark them as approved or rejected.

Actors: Faculty

Preconditions:
Faculty is logged in and at the view specialization track page.

Flow of Events:
Basic Flow



User Action	System Response
1. .Faculty selects review pending track enrollment requests from the list of tracks	1.If this faculty is the owner of the selected specialization track, the list of pending enrollment requests are displayed
2 Faculty selects an enrollment request	2 The details of the enrollment request are displayed including Student name, entry, and request justification
3. Faculty edits the enrollment status, enters the status reason text and selects update	3. If the status reason field is non-blank, the faculty reviewer id, status reason, and status are updated in the track enrollment request and it is saved to the DB. The update request success page is displayed

Post-Conditions:
For the success basic flow the track enrollment request is saved to the DB with updates.

Business Rules:

1. Only the faculty owner of the specialization track can review track enrollment requests.
2. The status reason entry is text to explain the reason for accepting or rejecting an enrollment request. It must be non-blank when setting the status to accepted or rejected.
3. The status of track enrollment requests can be pending, accepted, or rejected.

Using ATDD, write a User Story for the above use-case, representing the basic success scenario. In your user story, indicate (in brackets, at the end of each statement) what section of the RUP use-case description, as given above, corresponds to each of your User story statement.

Answer:

- As a faculty **(RUP Actor)**
- I want to review pending track enrollment requests **(RUP brief description)**
- Given I am at the specialization track page **(RUP precondition)**
- So that I can mark the as either accepted or rejected and they are saved in the DB **(RUP post condition)**
- When I select review pending enrollment requests for a track **(RUP user input)**

Then I see the pending enrollment requests for the track that I am the owner of **(RUP system output)**

Part II – Software Engineering Problem-solving, Coding skills: (54 points)

Note: *For these questions, for each of your solution, you are expected to take screenshot(s) of your result(s), save it into a .png or .jpg image file and include these in the FinalExam.zip file you submit.*

1. (12 points) **Implementing Unit Testing using the JUnit framework**:

Implement code for a component named, MyArrayUtils, as a Java command-line (console) application. In your component (class), implement a method named, **hasMultipleMaximum(…)**, which takes as input, any array of integers and it returns true, if the maximum integer value in the array, occurs more than once, otherwise it returns false. For example, when given an input, a_in = [-6, 2, 5, 6, -6, 5, 6], your hasMultipleMaximum method should return the value, true. And when given an input, a_in = [-6, 2, 5, -6, 5, 6], the method should return the value, false.
Also, when hasMultipleMaximum (...) is called and the input argument passed is null or an empty array, it should return, false.

Using JUnit, implement unit tests for your MyArrayUtils component and its hasMultipleMaximum () method; covering the following 3 test-cases:

1. When the input is a legit integer array, such as [-6, 2, 5, 6, -6, 5, 6] or [-6, 2, 5, -6, 5, 6].

2. When the input is a null.

3. When the input is an empty array.

Create a JUnit TestSuite containing your 3 test-cases defined above. And execute your TestSuite and take a screenshot of your result, as displayed by your IDE.

2. (42 points) **Implementing an Enterprise Web Application**

A local SuperStore, named WallyMarty, has hired you to design and develop a Customer Relationship Management (CRM) system for them, which they will be using to collect, maintain and manage data about their customers. They want you to implement a basic web application for this purpose. Especially important to the store is, their select group of PrimeCustomers, who they offer a special 10% discount on every purchase that customers belonging in this group, make at the store.

A PrimeCustomer is a customer who is of age, 40 or older.

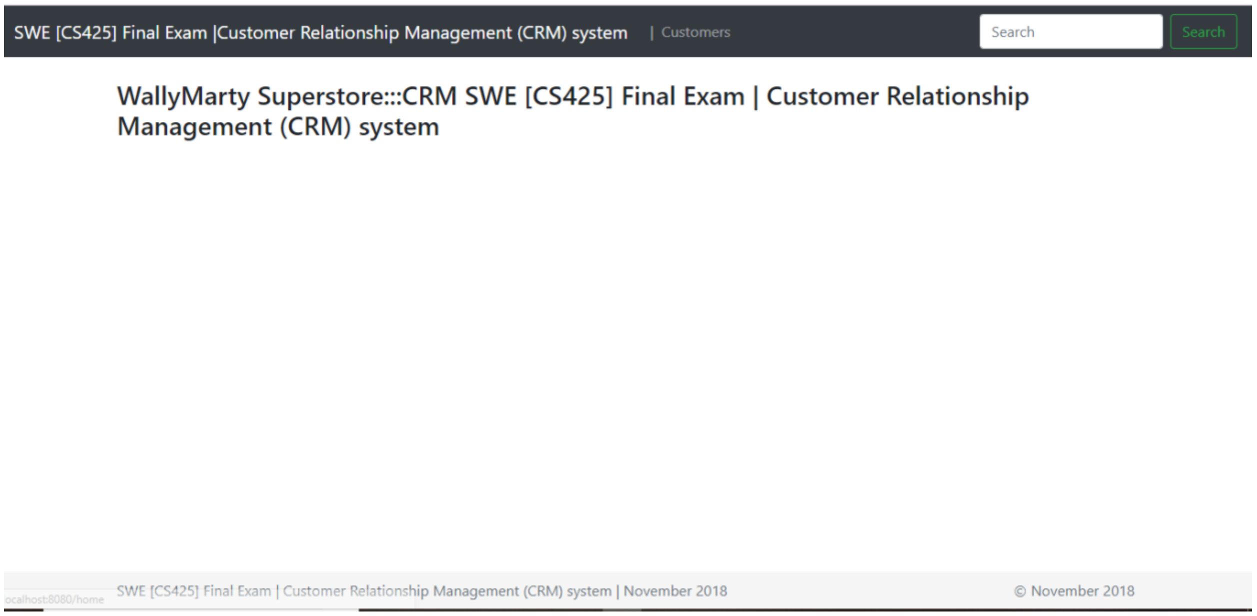
Here are the attributes for the Customer entity:
Customer: customerId:long, customerNumber:string, name:string, contactPhoneNumber:string, dateOfBirth:date

For this question, you are expected to do the following:

1. Using the set of tools, technologies and frameworks which you have learnt about in this CS425 course, including Spring Boot, Spring Web MVC, Spring Data JPA, etc., (or some other Enterprise Web application development platform/tool(s) that you prefer), implement a working web application for WallyMarty SuperStore. You may use any database of your choice.

You are expected to implement only the following use-cases:

1. Display the application's Homepage.
   Solution:



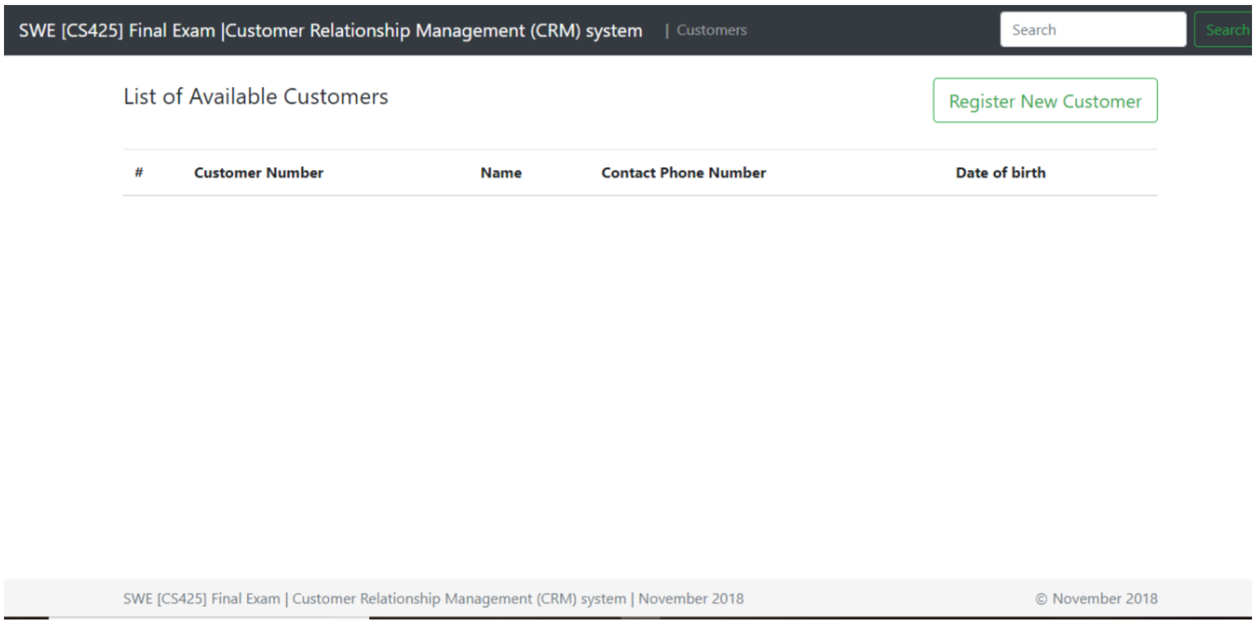
SWE [CS425] Final Exam |Customer Relationship Management (CRM) system
| Customers
Search
Search

WallyMarty Superstore:::CRM SWE [CS425] Final Exam | Customer Relationship Management (CRM) system

SWE [CS425] Final Exam | Customer Relationship Management (CRM) system | November 2018

© November 2018

2. Display list of Customers (Allows the store manager to view a list of all the Customers registered in the system). The store requires this list to be displayed sorted in ascending order of the Customers' names (see sample screen below).
   Solution:



SWE [CS425] Final Exam |Customer Relationship Management (CRM) system | Customers

Search

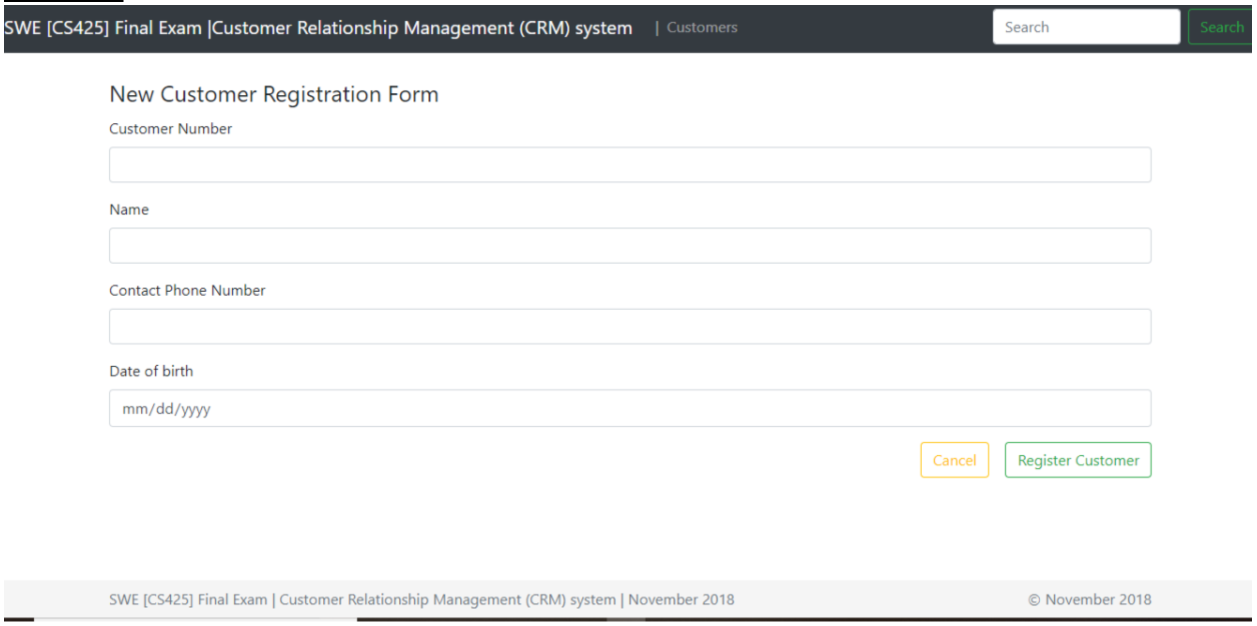
List of Available Customers

Register New Customer

#	Customer Number	Name	Contact Phone Number	Date of birth

SWE [CS425] Final Exam | Customer Relationship Management (CRM) system | November 2018 © November 2018

3. Register a new Customer (Allows the store manager to add a new Customer into the system).
   Solution:



SWE [CS425] Final Exam |Customer Relationship Management (CRM) system | Customers

Search

New Customer Registration Form

Customer Number

Name

Contact Phone Number

Date of birth

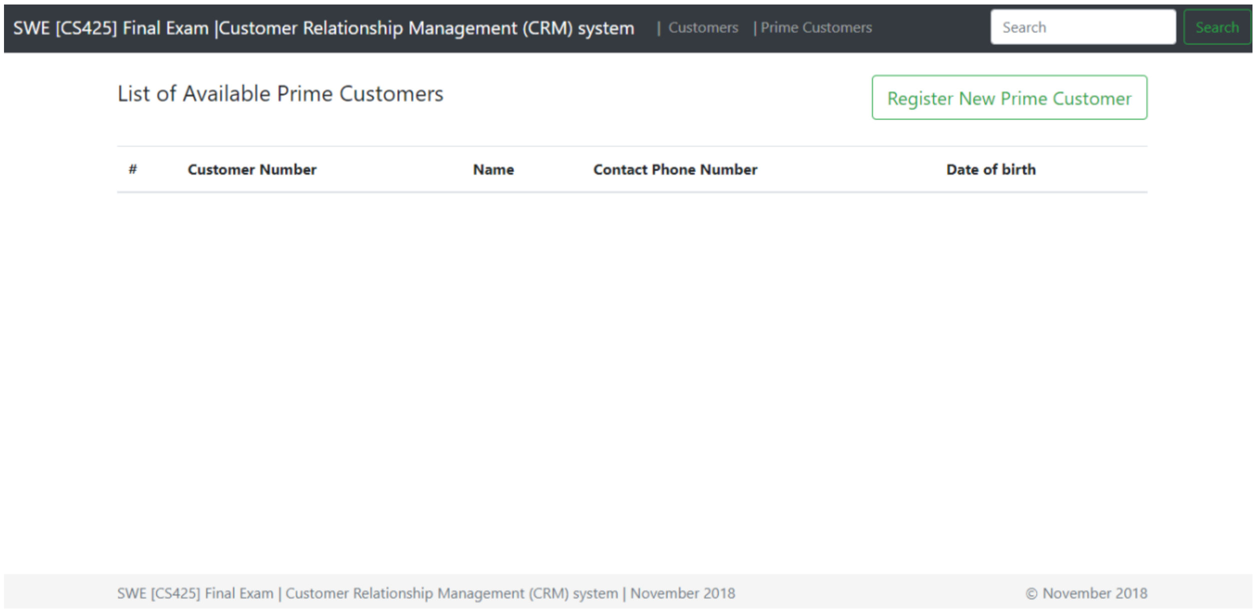
mm/dd/yyyy

Cancel Register Customer

SWE [CS425] Final Exam | Customer Relationship Management (CRM) system | November 2018 © November 2018

4. Display list of PrimeCustomers (Allows the store manager to view a list of all the PrimeCustomers in the system). The store requires this list to be displayed sorted in ascending order of the Customers' dates of birth (see sample screen below).

Solution:


SWE [CS425] Final Exam |Customer Relationship Management (CRM) system
| Customers
| Prime Customers
Search
Search

List of Available Prime Customers

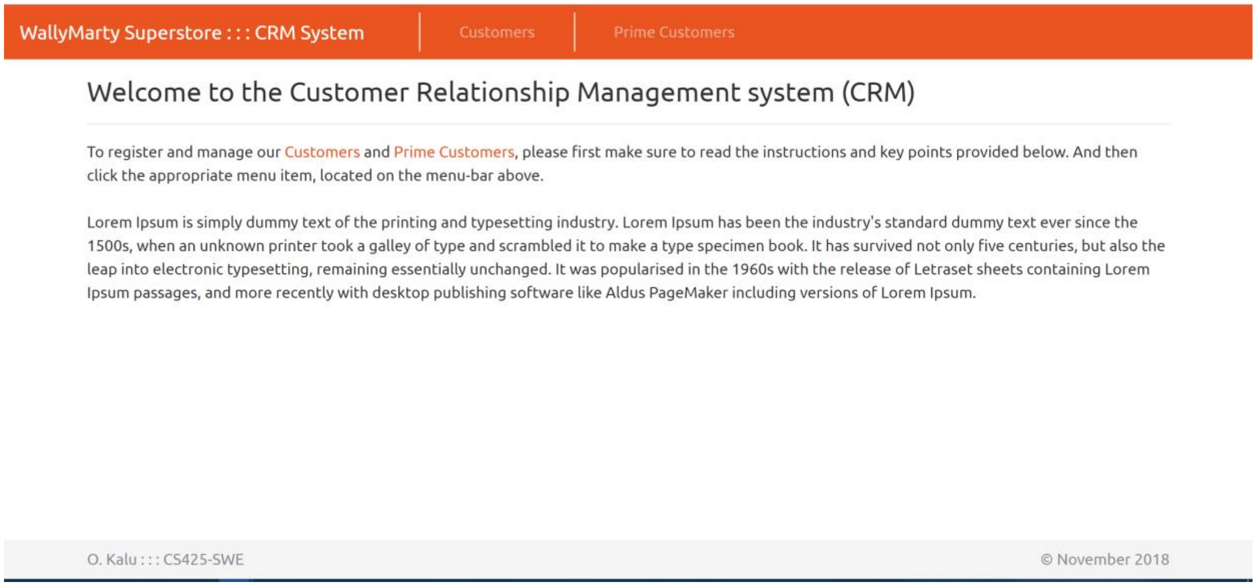
Register New Prime Customer

#	Customer Number	Name	Contact Phone Number	Date of birth

SWE [CS425] Final Exam | Customer Relationship Management (CRM) system | November 2018 © November 2018

Shown below are sample User Interfaces for the use-cases. Note: Your own UI design does NOT necessarily have to look exactly like these samples. But your UIs should contain all the necessary data and data fields, as shown.

Homepage:



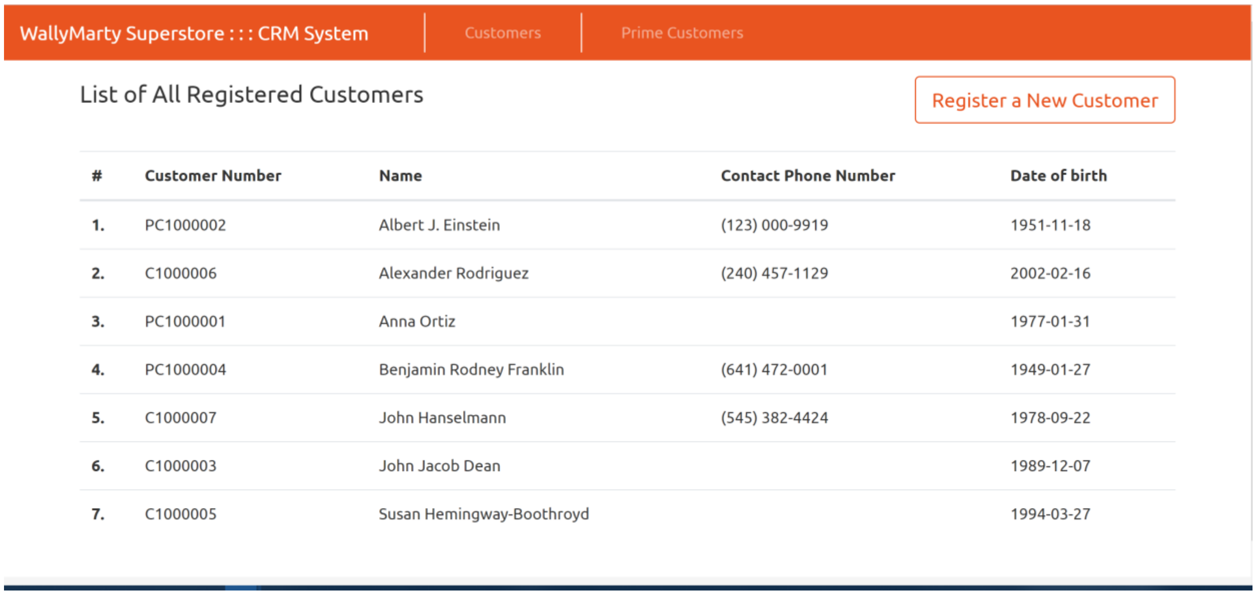
Welcome to the Customer Relationship Management system (CRM)

To register and manage our Customers and Prime Customers, please first make sure to read the instructions and key points provided below. And then click the appropriate menu item, located on the menu-bar above.

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

O. Kalu : : : CS425-SWE © November 2018

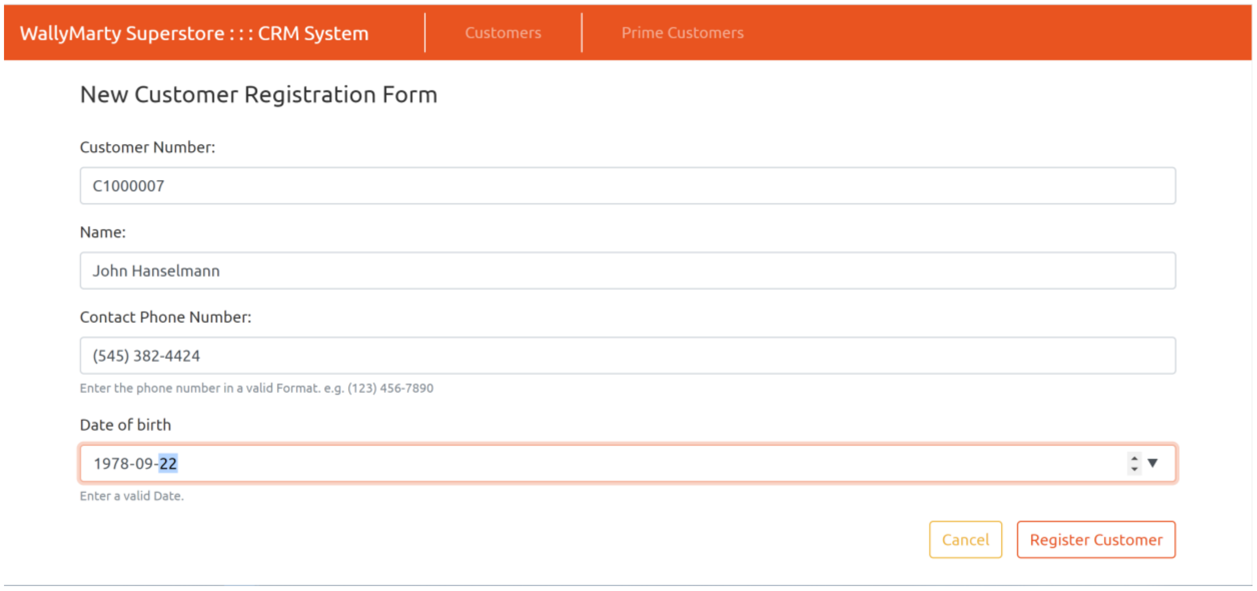
List of all Registered Customers:



List of All Registered Customers

#	Customer Number	Name	Contact Phone Number	Date of birth
1.	PC1000002	Albert J. Einstein	(123) 000-9919	1951-11-18
2.	C1000006	Alexander Rodriguez	(240) 457-1129	2002-02-16
3.	PC1000001	Anna Ortiz		1977-01-31
4.	PC1000004	Benjamin Rodney Franklin	(641) 472-0001	1949-01-27
5.	C1000007	John Hanselmann	(545) 382-4424	1978-09-22
6.	C1000003	John Jacob Dean		1989-12-07
7.	C1000005	Susan Hemingway-Boothroyd		1994-03-27

Register a new Customer:



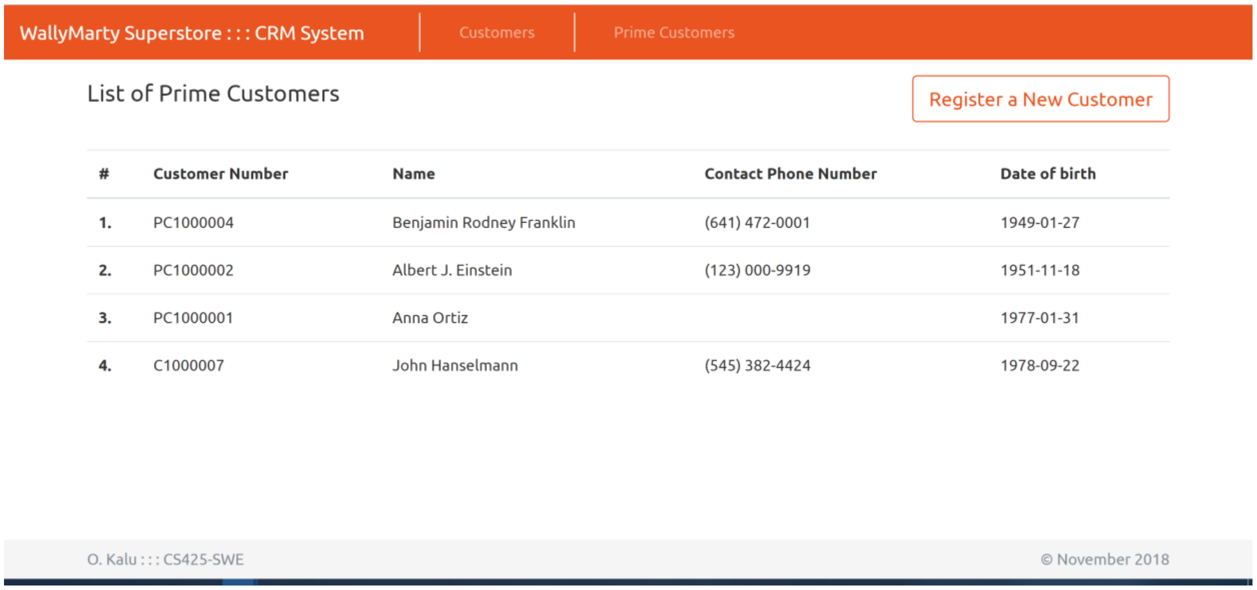
New Customer Registration Form

Customer Number:	C1000007
Name:	John Hanselmann
Contact Phone Number:	(545) 382-4424
Date of birth	1978-09-22

Enter the phone number in a valid Format. e.g. (123) 456-7890

Enter a valid Date.

List of Prime Customers:



List of Prime Customers

#	Customer Number	Name	Contact Phone Number	Date of birth
1.	PC1000004	Benjamin Rodney Franklin	(641) 472-0001	1949-01-27
2.	PC1000002	Albert J. Einstein	(123) 000-9919	1951-11-18
3.	PC1000001	Anna Ortiz		1977-01-31
4.	C1000007	John Hanselmann	(545) 382-4424	1978-09-22

O. Kalu : : : CS425-SWE © November 2018

//-- The End --//