# Table of Contents

I. Introduction .......................................................................................................................... 3
   I.I Description of the Problem ............................................................................................... 3
   I.II Overview of Solution ........................................................................................................ 4
      Platform Independent ........................................................................................................... 4
      Import/Export .................................................................................................................... 4
      Bulk Email .......................................................................................................................... 4
      Reminder Tool .................................................................................................................... 5
      Customer Management/ Group Management ..................................................................... 5
      Shipping Label Generator .................................................................................................. 5
      User Friendly Interface ....................................................................................................... 5
      Referral Tracking ............................................................................................................... 6

II. Scenario of System Use ......................................................................................................... 6

III. Technical Overview ............................................................................................................ 12
   III.I Database Design ............................................................................................................ 13
   III.II Package Structure ....................................................................................................... 14

IV. Testing Plan .......................................................................................................................... 15
   IV.I User Testing ................................................................................................................... 15
   IV.II Code Testing ................................................................................................................. 15

V. Conclusion ............................................................................................................................ 19

Bibliography ............................................................................................................................... 20
I Introduction

I.I Description of the Problem

In the field of non-retail sales, customer tracking and referral organization is key to success. Being organized allows sales reps to communicate effectively. With today’s technology, there are many outlets that sales reps organize and communicate with customers and referrals. As far as organization goes, sales reps may use programs with address books such as Microsoft Outlook and other email programs. For communication, there are emails, letters, and text messages.

Vector Marketing Sales reps are part of the marketing arm of Cutco Cutlery. Vector Marketing sales reps do in home demonstrations to potential customers. Also, sales reps maintain a customer list of their customers they routinely service. Vector Marketing sales reps manage anywhere from 100 to 10,000 customers. In order to manage customers, Vector Marketing sales reps use multiple systems. Historically, sales reps tend to use an email client to email customers, a spread sheet as a database, outside services to mail letters, and a calendar tool to track meetings/appointments.

There is customer relationship management (CRM) software available that encompasses some of the desired features above. Table 1 depicts some of the CRM software available:

<table>
<thead>
<tr>
<th>Features</th>
<th>Simply Contacts</th>
<th>Customer Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import/Export of Contacts</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Dependencies on other Software</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Bulk Email</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Reminder Tool</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Shipping Label Generator</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Issue Report</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Add/Remove Contact</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Add/Remove Contact Fields</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Contact Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>List contacts</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer Notes/Reminder</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>User Friendly</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Easy Navigation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform independent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Dependencies</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Intuitive UI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1

## I.II Overview of Solution

### Platform Independent

Given the matrix comparison above, the Customer Communicator is aimed at filling in the gaps between the two already existing database applications, and provides the features of existing general-purpose communication tools. The solution will consist of providing the user an application that sits on their desktop. To accommodate the fact that Simply Contacts and Customer Database are platform dependent, Customer Communicator will be written in java so that it may be run on any operating system that has a Java Virtual Machine.

### Import/Export

Simply Contacts requires that the user has a working copy of Microsoft Excel in order to import existing databases. Customer Database’s UI for importing customers is very confusing. Customer Communicator will be implemented such that there are no software dependencies and importing is as simple as uploading a CSV file.

### Bulk Email

Customer Communicator will also allow the user to send bulk email from their existing email account. This means that Customer Communicator will allow the user to configure settings such that they are able to set up their email address and password to connect to their existing email client to send emails. Furthermore, emails can be personalized so that bulk emails do not lose personal touch.
Reminder Tool

A reminder tool allows the user to associate reminders to customers. This is useful because it allows the user to be more effective in their communication. Also, combining a reminding tool in the same place as the customer database, allows the user to not have to go two places to perform these tasks.

Customer Management/ Group Management

The Customer Management system will be implemented such that the user interface is intuitive and easy to navigate. This is done by displaying customers in the database in a table format. Operations can then be done directly to the table or via buttons that exist in the window. Examples of operations that can be performed are adding, deleting, and updating customers and information. The Customer List window also allows the user to send emails from the table itself via the email buttons. To increase ease of navigation, users can filter customers to view subsets of their customer database. Users can save subsets of customers in groups that appear in tabs across the window.

Shipping Label Generator

Users also have the ability to create preformatted labels that they can use to attach to letters that they want to send to customers. Having the ability to create shipping labels exposes the user to another form of communication with their customers.

User Friendly Interface

In order to simplify navigation and use of the application, a main console with buttons that exhibit the program functionality is implemented. The main console window will include the following. Simple User Console – 7 buttons

- Import/Export
- Create Labels
- Manage Contacts
- Manage Referrals
- Reminders
- Email
- Settings
Referral Tracking

The application allows the user to track referrals. This is extra functionality that is not found in the other two applications. Although not all of the functionality applies to referrals in the database, users are still able to send emails and perform editing commands such as adding, deleting, and updating referral information. Also, users can filter referrals for easy navigation of the referral database.

II. Scenario of System Use

To give a better idea of how the Customer Communicator is used a brief scenario use will be shown. The diagrams will show use cases where the customer navigates from the main console, imports customers, emails them, filters and groups customers.

When users launch the program, the first thing they see is the console. It is from here that users can import/export customers, manage customers/referrals, and access other features.

When users first begin using the program, it is common that they start by importing their existing database. Customer Communicator supports importing of CSV files. To import, the user exports their CSV file then selects the import button on the main console window in Figure 1. The import dialogue is shown in Figure 3.
After, the import, the user then can observe the customers via clicking on the ‘Manage Contacts’ button on the customer console window. Figure 4 shows the customer list:
The Customer List window provides an interface where users can perform operations by clicking on the buttons on the bottom panel. For example, when users add customers, the ‘Add Customer’ dialogue shown in Figure 4 opens.

Users can also group their customers to better organize them. Placing users in group usually begins by filtering the customers into same city, number of times purchased, order totals, etc. In the ‘Customer List’ window, users click the ‘Filter Customers’ button. When clicked, the ‘Filter’ dialogue in Figure 5 opens.
After the customers are filtered, the user then clicks the ‘New Group’ button in the Customer List window. The ‘New Group’ dialogue shown in Figure 6 appears.

![New Group Dialogue](image)

Figure 6 - New Group Dialogue

When the user clicks ‘OK,’ the Customer List window now contains a new tab with the name of the group that was just created. When the user clicks on the tab, the Customer List window displays the customers in that group as shown in Figure 7.

![Group Tab](image)

Figure 7 - Group Tab
Another operation that users commonly do is email customers. Users can email customers by clicking and highlighting those who they wish to email. Users may choose to email individuals or an entire group. This is done by clicking the ‘Email Customer’ button or the ‘Email Group’ button in the Customer List window. The ‘Email Options’ dialogue show in Figure 8 then opens which allows users to include personal greetings and specify how recipients receive emails.

![Figure 8 - Email Setup](image)

After the email options are set, the email preview (shown in Figure 9) opens which allows users to create the message. Also, users can choose to attach files, and add other recipients in this window. When completed the users clicks ‘Send’ to send the email.

![Figure 9 - Email Dialogue](image)
All emails that are sent are archived for the user. To view the outbox, the user clicks on the ‘Email’ button in the Customer List Window.

![Outbox](image1.png)

**Figure 10 - Outbox**

To help users maintain information about customers, users can attach notes to customers in the Customer List Window. To attach notes, users highlight a customer and click the ‘Add Note’ button to bring up the dialogue depicted by Figure 11.

![Note Dialogue](image2.png)

**Figure 11 - Note Dialogue**
Users may also access the settings dialogue by clicking the ‘Settings’ button in the main console window. The settings dialogue lets the user set preferences for emailing, personal information, display options, and tool settings.

![Settings Dialogue](image)

Figure 12 - Settings Dialogue

### III. Technical Overview

To describe the technical nature of the solution, the database schema will be described along with the package structure. Figure 13 shows the ER diagram of the schema and Figure 14 shows the package structure.
The two main records in the database are the Customer and Referral entities.

**Customer Attributes:** CustomerNo (primary key), Last Name, First Name, Address1, Address2, Email, Send Code, City, State, Zip, Phone, Do Not Call, Lifetime $, Lifetime Orders, # Items Purchased, 1st Order Date, Last Order Date

**Referral Attributes:** First Name, Last Name, Phone, Email, City

**cNotes Attributes:** Text, custNo(foreign key)

**Email Attributes:** Text, Id (auto_increment, primary key), custNo (foreign key)

**rNotes Attributes:** Text, (references: name, phone)

**Group Attributes:** gName (primary key), custNo (primary key)

Customers have a one-to-one relationship with the cNotes. This means that for every customer, there is exactly one cNote that is associated with it. Customers also have relationship to groups meaning many customers can be associated to many groups. Also, customers are associated to many emails. The same relationships hold for the Referral entity.
III.II Package Structure

The package structure for CustComm is broken up into three sections: logic, UI, and util. The actual controller of the application exists on top of the three sets of package. It is called CustCommApp.java. This controller interface serves as the proxy between the logic and UI classes in the model, view controller design pattern. CustCommApp controls the major components of the application, namely the functions found on the main console.

![Figure 14 - Source Packages]

**Logic Packages:** The packages in the first column represent the logic packages to the corresponding UI packages. The fact that the application relies on a MySQL database, most of the backend functionality that would be found in the logic packages is taken care of by the DBMS.

**UI packages:** The UI packages include all the packages that perform any sort of display to the screen. This includes updating customer tables (in the customer package), displaying the outbox, importing/exporting, and the referral table.

**Util:** The Util packages hold the classes that deal directly with the database. The DAO package contains classes that pertain to tables found in the database. All MySQL statements such as SELECTS, INSERTS, UPDATES, and DELETES as they pertain to the records in the database are in the DAO package. The DTO package is the records of the database in java object form. This allows the application to perform
operations that are outside of the database. Lastly, the startup package serves as configuration setup when the application is first launched.

IV Testing Plan

Testing Customer Communicator consisted of gathering feedback from actual users of the application. Also, testing model and UI classes were implemented by using a JUnit framework for the model classes and UISpec4J[^3] for the UI classes. UISpec4j is a framework that is used to test Java Swing applications.

IV.I User Testing

During the course of the project, potential users of the application were allowed use of the application for testing purposes. The application was installed for them and they were encouraged to use the application as a part of their customer operations. Specifically, Vector Marketing representatives that were consulted were Joe Giannosa, Trevor Kelly, and Nikola Pang. Using these sales reps as a reference were beneficial in testing the UI layout and testing the requirements.

IV.II Code Testing

The testing packages mirror the source package layout in Figure 14. For purposes of this project, the classes that the tests were focused on were those associated with the customer list. Figure 15 highlights the packages where the tests were implemented.

Figure 16 - Testing Packages
The first phase of testing dealt with inserting customers into the database. The first test comprised of null customers and those with no customer number (primary key that is required).

```java
public void testphase1() {
    ArrayList<CustomerDTO> list = new ArrayList<CustomerDTO>();
    list.add(null);
    list.add(new CustomerDTO());
    for(int i=0; i<list.size(); i++) {
        if(i==0)
            assertFalse(validateCust(list.get(i)));
        else
            assertFalse(validateCust(list.get(i)));
    }
}
```

The second phase tested input where the same customer number had already been entered. Since the customer number is a primary key, this should result in an error.

```java
public void testphase2() {
    CustomerDTO c1 = new CustomerDTO();
    c1.setCno("A00128");
    assertFalse(validateCust(c1));
    c1.setCno("1");
    assertTrue(validateCust(c1));
}
```

Since most of the underlying object data is verified by the database most of the testing was UI based. UISpec4J is a testing framework built off of JUnit that allows for UI testing. The code for adding and deleting customers is shown below:

```java
public void testAddDeleteCustomer() throws Exception {
    // open "Add Customer" dialogue
    WindowInterceptor.init(cust.getButton("Add Customer").triggerClick()).process( new WindowHandler() {
        @Override
        public Trigger process(Window window) throws Exception {
            window.getTextBox("custNo").setText("1111111");
        }
    });
}
```
window.getTextBox("firstName").setText("Eddie");
window.getTextBox("lastName").setText("Tavarez");
window.getTextBox("a1").setText("123 Casa St.");
window.getTextBox("city1").setText("Visalia");
window.getTextBox("state1").setText("CA");
window.getTextBox("zip1").setText("93277");
window.getTextBox("email1").setText("etavarez@calpoly.edu");
window.getTextBox("p1").setText("559");
window.getTextBox("p2").setText("123");
window.getTextBox("p3").setText("4567");
window.getTextBox("total1").setText("356");
window.getTextBox("items1").setText("3");
return window.getButton("Add").triggerClick();
}

Table table = cust.getTable();

assertTrue(table.rowEquals(0,new Object[] {
    // Table expected values
    cNo, last, first, address1, "", email, "", city, state, "93277", p1+p2+p3, "0", "356", "1", "3", new Date().toString(), new Date().toString()
}));

// test delete
    table.selectRow(0);
    Button del = cust.getButton("Delete Customer");
    del.click();

    assertFalse((table.containsRow(new Object[] {
        // Table expected values
        cNo, last, first, address1, "", email, "", city, state, "93277", p1+p2+p3, "0", "356", "1", "3", new Date().toString(), new Date().toString()})
    ));
    DBUtil.commit();
}

Testing the email functionality from the customer list window presented a challenge. Testing the sending required that actual emails be sent. The problem is that the email addresses tested were actual customers. Also, sending emails to customers and knowing if they received them or not almost makes testing pointless if their arrival cannot be verified. To overcome this, an open source project called Antix that emulates an STMP server was used. Antix, captures all outgoing email messages sent from within the network. To configure the Customer Communicator to send emails to the Antix outbox, the username in the email settings was set to ‘eddie@localhost.’ After running the UISpec4J test class, ‘EmailTest,’ the email sent via clicking all the corresponding dialogue buttons shows up in the Antix outbox. The code used to test the email functionality is show below. The Antix outbox is also show below in Figure 16.
public void testEmail() throws Exception {

    Table table = cust.getTable();
    table.selectRow(1);
    WindowInterceptor.init(cust.getButton("Email Customer").triggerClick()).process(new WindowHandler() {
        @Override
        public Trigger process(Window window) throws Exception {
            // email setup
            return window.getButton("OK").triggerClick();
        }
    });
    .process(new WindowHandler() {
        @Override
        public Trigger process(Window window) throws Exception {
            // email send
            return window.getButton("Send").triggerClick();
        }
    }).run();

    // actual send verified in Antix outbox
    assertTrue(true);
}

Figure 17 - Antix Outbox
V. Conclusion

Although not all of the functionality has been completed during the school year, there exists opportunities that make continued work on the project potentially lucrative. Currently, the released version includes all the customer and group management and email capabilities. As noted in section IV.I, the released version is currently being used by Vector Marketing Sales Rep, Joe Giannosa. Using Joe as a direct customer as a means to gather user input and testing has provided valuable input in creating the released version. Also, word of mouth of the application has generated interest in the project indicating that a version with full functionality would be worth releasing.

Future work of the project will include:

- Automated email
- Full testing suite
- Template design for emails
- Calendar tool
- Billing system
- Statistics tracker
- Business Plan

It should be noted that adding the rest of the functionality will provide a full suite of functionality that is not found in the current market. Being able to market and sell the Customer Communicator would require creating a business plan that encompasses the amount of work, how the product would be advertised, who will work on the project itself, and billing/pricing functionality.
Bibliography