Ramon's Organic Grain Fed Organ Emporium

Design Document

University of British Columbia Okanagan
COSC 304 - Fall 2016

Team Members:
Nino Gonzales
Julius Wu
Emerson Kirby
James Rogers

Table of Contents

1.0 Introduction
    1.1 Mission Statement
    1.2 Executive Summary

2.0 Domain Assumptions
    2.1 Users
    2.2 Product
    2.3 Transport
    2.4 Payment

3.0 Entity Descriptions
    3.1 UML Diagram
    3.2 SQL DDL
    3.3 Relational Assumptions

4.0 Mockups
    4.1 Landing Page Mockup

Section 1: Introduction

Ramon's Organic Grain Fed Organ Emporium is a store that specializes in the sale of 100% organic, grain fed, premium grade organs. We have multiple brick and mortar stores located in Canada, India and the Philippines, with plans to expand into the UK and USA in 2018.

The purpose of this document is to lay out the framework and the design requirements for Ramon's Organic Grain Fed Organ Emporium's push into the e-commerce space via ramonsorganicgrainfedorganemporium.org.

1.1 Mission Statement

To pour your hearts into our business, of selling human hearts. And kidneys. And livers. And whatever other organ a customer may wish for.

1.2 Executive Summary

Ramon's Organic Grain Fed Organ Emporium currently has a hefty share of the international human organ market, and is poised to dominate the e-commerce sphere as well. It couldn't come at a better time either, as the world need for organs is expanding rapidly. Current estimates value the profits of organ trade at $600 million to $1.2 billion, with long term growth expected as far the eye can see. Currently, about 90,000 people in the world are reported to be waiting for a new organ, paying anywhere from $2,000 to $160,000, organ dependant. Ramon's Organic Grain Fed Organ Emporium intends to serve them all, with the highest quality organs available.

The website will act as a broker between Ramon's Organic Grain Fed Organ Emporium (hereinafter referred to as the The Company) and consumers. The customer will visit The Company's website when they are looking to purchase an organ. They are able to browse organs, and search for organs by keyword. The customer is able to browse and add organs to their cart without signing in, but must create an account to make a purchase.

The website will display what organs The Company has available, will display the organs history and it's bill of health, along with details on quality and pictures. The Company will facilitate the financial transaction and organize the shipping to the customer.

The Company will NOT be responsible for the transplant, however, there will exist our patented and copyrighted "Doctors Portal", for a customer's doctor to check out the history and details of the customer's purchased organ before performing the transplant (ie blood type, organ removal date).

The Doctor Portal is a special log-in with elevated credentials for certified doctors. It allows them to view customer purchase history, add/remove/update organs and organ

information, and to track inventory levels. It will also allow doctors to generate simple sales reports to track trends in the organ trade. IE most popular organs, where customers are located, total sales volume for each organ etc etc.

In this document, the site is detailed with Unified Modeling Language using Entity-Relationship diagrams. These can be viewed in Section 3. The site will be constructed using the following technologies on the cosc304.ok.ubc.ca server:

PHP, JavaScript, HTML, CSS, MySQL

Section 2: Domain Assumptions

2.1 Users:

The site will have three types of users:

1. Customer
    a. A customer profile will store the user address (for shipping), payment information, communication information (ie email), username and password, and purchase history.
    b. Customers will be able to view organs and add organs to their cart without logging in.
    c. In order to make a purchase, user must log into their profile using their username (which is a unique ID) and password.
    d. Customers do not have access to the Doctor Portal.

2. Doctor
    a. A doctor profile stores the doctor's practice and practice address, plus the same information as user.
    b. Doctors are able to log in to our proprietary "Doctor Portal" which is essentially an elevated log in that entails the ability to:
        i. Access a customer's order history
        ii. Add/remove/update organs and organ history
        iii. Add/remove donor information
        iv. Track inventory levels in our warehouses
        v. Generate sales reports (excluding fiscal information, so things like where organs are being shipped, most commonly bought organ etc.)
    c. Doctors also have all customer privileges when they are not logged into the Doctor Portal
    d. Doctors can query the Donor table, and the Customer table

3. Administrator
    a. An administrator has root access to the system.
    b. Administrators are able to add/remove warehouses to the system.

c. Administrators can query payment table.
d. Administrators are able to add/remove doctors to the system
e. An administrator has doctor and user privileges.
f. Administrator privilege is reserved for executive IT staff only.

2.2 Product (organs)

- All organs sold are sourced and vetted entirely by Ramons Organic Grain Fed Organ Emporium. In other words we are not an organ reseller
- We may accept organs donated from certified doctors who are in our Doctor Portal (ie once they have been certified, they have the ability to ship organs directly to our warehouse and put them into stock. They will be vetted at the facility though)
- All organs are grain fed and totally organic.
- We claim no responsibility for transplants nor for rejected organs

2.3 Transport

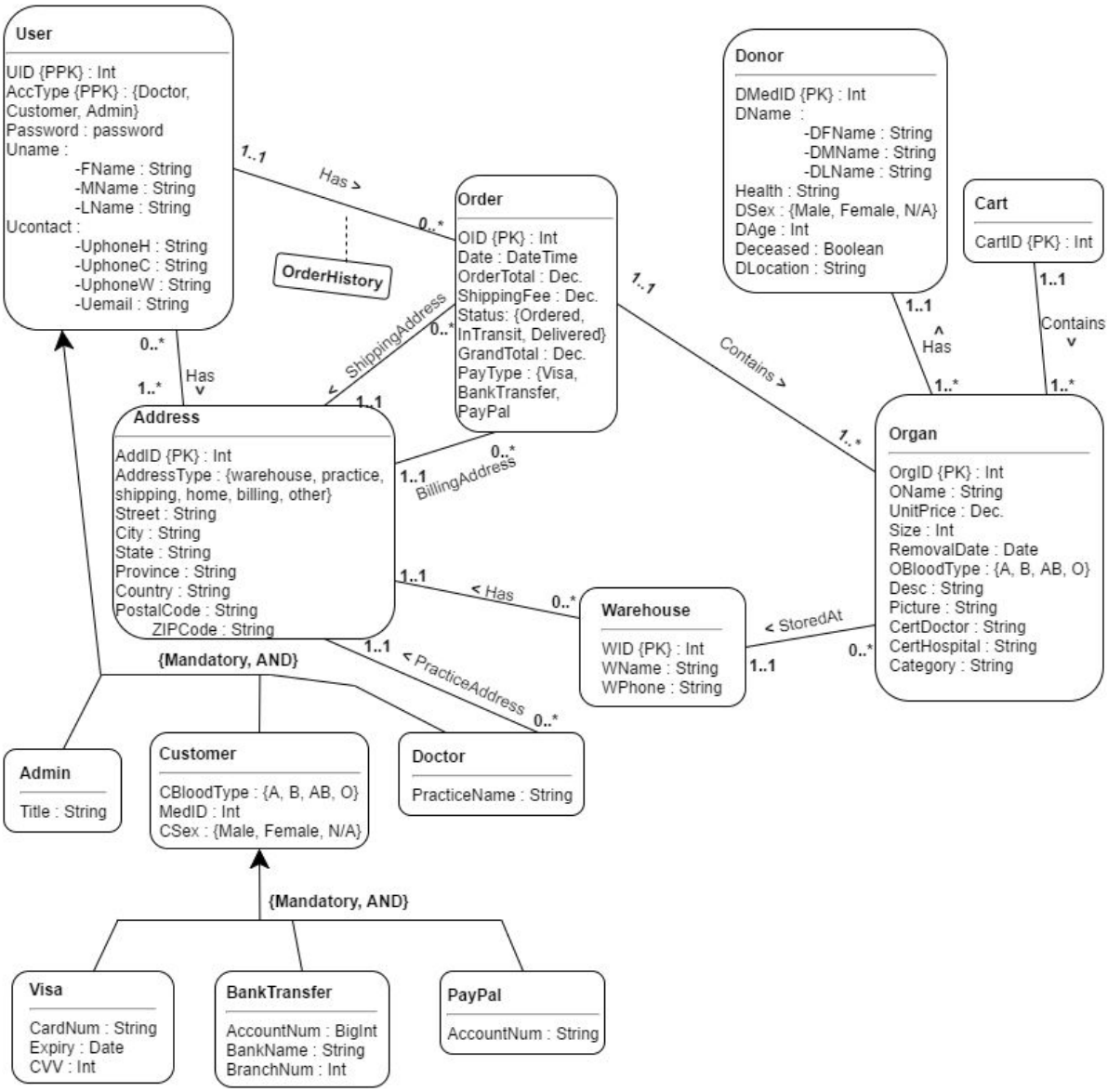
- Transport will only be done with certified medical couriers, we do not and will not ship by common courier
- Ramons Organic Grain Fed Organ Emporium is responsible for choosing the most suitable shipping company based on customer location
- Ultimately, we are not responsible for unanticipated delays with shipping.

2.4 Payment

- Payment is through either VISA, PayPal, or Bank Transfer
- Organs can not be reserved.

Section 3: Entity Descriptions

3.1 UML Diagram



3.2 SQL DDL

```
CREATE TABLE `User` (
        `UID` int NOT NULL AUTO_INCREMENT,
        `AccType` ENUM('Customer', 'Doctor',
'Admin') NOT NULL,
        `fName` varchar(50) NOT NULL,
        `mName` varchar(50),
        `lName` varchar(50) NOT NULL,
        `UphoneH` varchar(10) NOT NULL,
        `UphoneC` varchar(10),
        `UphoneW` varchar(10),
        `Uemail` varchar(10) NOT NULL,
        `Password` varchar(100) NOT NULL,
        PRIMARY KEY (`UID`,`AccType`)
);


CREATE TABLE `Address` (
        `AddID` int NOT NULL
AUTO_INCREMENT,
        `UID` int,
        `WID` int,
        `AddressType` ENUM('Warehouse',
'Practice', 'Shipping', 'Home', 'Billing', 'Other') NOT
NULL,
        `Street` varchar(50) NOT NULL,
        `City` varchar(50) NOT NULL,
        `Province` varchar(50),
        `Country` varchar(3) NOT NULL,
        `State` varchar(50),
        `PostalCode` char(6),
        `ZipCode` char(5),
PRIMARY KEY (`AddID`),
FOREIGN KEY (`UID`) REFERENCES `User`(`UID`)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (`WID`) REFERENCES
`Warehouse`(`WID`) ON DELETE CASCADE ON
UPDATE CASCADE
);


CREATE TABLE `Admin` (
        `UID` int NOT NULL AUTO_INCREMENT,
        `Title` varchar(40) NOT NULL,
PRIMARY KEY (`UID`),
FOREIGN KEY (`UID`) REFERENCES `User`(`UID`)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE `Order` (
        `OID` bigint NOT NULL
AUTO_INCREMENT,
        `ShipAddID` int NOT NULL,
        `BillAddID` int NOT NULL,
        `Date` DATE NOT NULL,
        `OrderTotal` DECIMAL(12,2) NOT NULL,
        `ShippingFee` DECIMAL(12,2),
        `Status` ENUM('Ordered', 'InTransit',
'Delivered') NOT NULL,
        `GrandTotal` DECIMAL(12,2) NOT NULL,
        `PayType` ENUM('Visa', 'BankTransfer',
'PayPal') NOT NULL,
        `UID` int NOT NULL,
PRIMARY KEY (`OID`),
FOREIGN KEY (`ShipAddID`) REFERENCES
`Address`(`AddID`) ON DELETE NO ACTION ON
UPDATE CASCADE,
FOREIGN KEY (`BillAddID`) REFERENCES
`Address`(`AddID`) ON DELETE NO ACTION ON
UPDATE CASCADE,
FOREIGN KEY (`UID`) REFERENCES `User`(`UID`)
ON DELETE CASCADE ON UPDATE CASCADE,
);


CREATE TABLE `Customer` (
        `MedID` int NOT NULL
AUTO_INCREMENT,
        `UID` int NOT NULL,
        `CBloodType` ENUM('A', 'B', 'AB', 'O') NOT
NULL,
        `CSex` ENUM('Male', 'Female'),
PRIMARY KEY (`MedID`),
FOREIGN KEY (`UID`) REFERENCES `User`(`UID`)
ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE `Doctor` (
        `UID` int NOT NULL AUTO_INCREMENT,
        `PracticeName` varchar(50) NOT NULL,
PRIMARY KEY (`UID`),
FOREIGN KEY (`UID`) REFERENCES `User`(`UID`)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE `Visa` (
	`CardNum` char(16) NOT NULL
AUTO_INCREMENT UNIQUE,
	`UID` int NOT NULL,
	`Expiry` DATE NOT NULL,
	`CVV` int(3) NOT NULL,
PRIMARY KEY (`CardNum`),
FOREIGN KEY (`UID`) REFERENCES `User`(`UID`)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE `PayPal` (
	`AccountNum` bigint NOT NULL
AUTO_INCREMENT,
	`UID` int NOT NULL,
PRIMARY KEY (`AccountNum`),
FOREIGN KEY (`UID`) REFERENCES `User`(`UID`)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE `Warehouse` (
	`WID` int NOT NULL AUTO_INCREMENT,
	`WName` varchar(50) NOT NULL,
	`WPhone` varchar(10) NOT NULL,
	PRIMARY KEY (`WID`)
);
```

```
CREATE TABLE `Cart` (
	`CartID` int NOT NULL
AUTO_INCREMENT,
	PRIMARY KEY (`CartID`)
);
```

```
CREATE TABLE `Donor` (
	`DMedID` int NOT NULL
AUTO_INCREMENT,
	`DFName` varchar(50) NOT NULL,
	`DMName` varchar(50) NOT NULL,
	`DLName` varchar(50) NOT NULL,
	`Health` varchar(50),
	`DSex` ENUM('Male', 'Female'),
	`DAge` int(3) NOT NULL,
	`Deceased` bool NOT NULL,
	PRIMARY KEY (`DMedID`)
);
```

```
CREATE TABLE `BankTransfer` (
	`AccountNum` bigint NOT NULL
AUTO_INCREMENT,
	`UID` int NOT NULL,
	`BankName` varchar(50) NOT NULL,
	`BranchNum` int NOT NULL,
PRIMARY KEY (`AccountNum`),
FOREIGN KEY (`UID`) REFERENCES `User`(`UID`)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE `Organ` (
	`OrgID` int NOT NULL AUTO_INCREMENT,
	`WID` int,
	`CartID` int,
	`DMedID` int NOT NULL,
	`OID` int,
	`OName` varchar(100) NOT NULL,
	`UnitPrice` DECIMAL(12,2) NOT NULL,
	`Size` int NOT NULL,
	`RemovalDate` DATE NOT NULL,
	`OBloodType` ENUM('A', 'B', 'AB', 'O') NOT
NULL,
	`Desc` varchar(500) NOT NULL,
	`Picture` varchar(500) NOT NULL,
	`CertDoctor` varchar(100) NOT NULL,
	`CertHospital` varchar(100) NOT NULL,
	`Category` varchar(100) NOT NULL,
PRIMARY KEY (`OrgID`),
FOREIGN KEY (`WID`) REFERENCES
`Warehouse`(`WID`) ON DELETE SET NULL ON
UPDATE CASCADE,
FOREIGN KEY (`CartID`) REFERENCES
`Cart`(`CartID`) ON DELETE SET NULL ON
UPDATE CASCADE,
FOREIGN KEY (`DMedID`) REFERENCES
`Donor`(`DMedID`) ON DELETE SET NULL ON
UPDATE CASCADE,
FOREIGN KEY (`OID`) REFERENCES
`Order`(`OID`) ON DELETE SET NULL ON UPDATE
CASCADE
);
```

3.3 Relational Assumptions

The following are assumptions to note about each entity, relating to its domain and how it will be constructed.

User:
The User entity contains personal information. All attributes are required when creating a profile except MName (middle name)UphoneC (cell phone), and UphoneW (work phone). A user can have multiple addresses, hence it's one-to-many relationship with the Address entity. We've assumed that a user will have a home address (required) and can also have multiple shipping addresses, or can ship to a specified address when checking out, both of which the system will remember. The system will suggest the users available addresses during checkout. A User logs in with email as their username, and can choose how they want to view the page via the mandatory subclasses they have available to them. There will be three subclasses of the User entity. A user must be of one type, and may be multiple:

- Customer
    - Customers additionally have their blood type, medical ID, and sex stored. If the user selects a sex value of N/A, that will be stored as a null value in the database. Customers will on top of that have 3 PayTypes. A customer must have one pay type and may have many:
        - Visa
            - The Visa entity additionally stores Card Number, Expiry, CVV.
        - BankTransfer
            - The BankTransfer entity additionally stores Account Number, Bank Name, and Branch Number
        - PayPal
            - The PayPal entity additionally stores the PayPal Account Number.
- Doctor
    - Doctors additionally have their practice name and their practice address stored. To get a doctor profile, a doctor must contact The Company who will verify authenticity, then an admin will create their profile for them.
- Admin
    - Admin's additionally have their title stored

Account:
The Account entity stores login information for users. All attributes are required when creating an account. There are three account types to match with the user types: Doctor, Customer, Admin.

Address:
The Address entity contains Address information. All attributes are required when adding an address. An address can have a state or a province, but not both. An address can have a PostalCode or ZIPCode There are 5 AddressTypes stored in the Address entity:

- Warehouse
    - Address for warehouses. Only used by the Warehouse entity.
- Practice
    - Address for a doctor's practice. A doctor may only have one practice.
- Shipping
    - Address to ship to, chosen / input during checkout. Only used by Order entity.
- Home
    - Address for each User. This is the required address of a user profile.
- Other
    - These are additional addresses for users, input during checkout or profile creation.

Order:
This is the table that stores orders. All attributes are required except ShippingFee which can be null. Orders are sent to the referenced warehouses as soon as they enter the system, which then immediately fills and ships the organ/s by special organ couriers. Because of the immediacy our system, once the order has been placed, the organ is automatically removed from its corresponding warehouse's inventory (by setting its Organ.WID to null), however they are left in the Organ table so they can queried by doctors. An order may contain organs from different warehouses, if this is the case, they are shipped individually.

Organ:
The Organ table stores all current organs and previously sold organs. All attributes are required except WID, the foreign key to the warehouse entity, which may be null. An organ is currently for sale if its WID is not null. The Location attribute allows the website to display the organs location without displaying the full address of the warehouse, ie displays the country, state/province, city. The Description attribute discusses the quality of the organ. The Picture attribute stores the filename and location in the system, so that it can be displayed. The CertDoctor attribute is the doctor who certified the organ to be up to ROGFOE standards, and the CertHospital is the hospital it was certified at.

Donor:
The Donor table stores the information of the organs donor. All attributes are required except DMName (donor's middle name). The Health attribute discusses the health of the donor, and is null if the donor is deceased. DSex is the donor's sex type and employees the same storage method as Customer sex type (if gender neutral, value is null). Information in the Donor entity is only available to Admin's and Doctor user types.

Warehouse:
The Warehouse entity stores The Company's warehouses. All attributes are required.

Cart:
The Cart entity stores organs while a user is shopping. When the user logs in to make a purchase, the CartID is stored in a session object. When the user checks out the cart, the cart is automatically turned into an Order to be completed and confirmed by the user. Once the user has checked out, the cart is deleted. If a user tries to add an organ to their cart that is currently in another cart, the user will be notified “the organ is in the process of being checked out, please come back later and try again”.

Payment:
The Payment entity stores billing information for an order. The GrandTotal attribute stores the Order total with tax in. The PayType attribute specifies how the customer paid for that order, as a customer may want to use different methods for different orders. There are three different PayTypes, which are subclasses of the Customer entity. A user must have at least one PayType, and may have more than one. They are:

Section 4: Mockups

4.1 Landing Page Mockup