**The Igloo**

*Semester Report*

Alex Indihar and Jonathan Shields

**Table of Contents**

**Section 1: Topic and Informal Description**

English Description of the Database

System Environment, Operating System, Programming Environment and Language

Project Plan

Responsibilities of each group member

**Section 2: Formal Description and Implementation**

Formal Description of the database

ER design

ER Mapping to relational database

Explanation of the mapping ER to relational database schema

Relationship database schema

-Provide the table information as a table of attributes

“Create Table” Statements

General description of implementation

-Show a diagram of the architecture of the system

-Explain the roles of each components and possible interactions

**Section 3: Group Details**

List of issues that has been considered and how they were resolved

Explanation of the Bonus

Description of the responsibilities of each group member

Major contributions of group members

Meeting information

**Section 1: Topic and Informal Design**

**Overview of Section:**

This section is going to be responsible for introducing the topic of the entire project as well as providing an English description of how the database application will relate to the topic. This section will also provide information on the system environment, operation system, and the programming environment and language. The project plan and the responsibilities of each group member will also be covered in this section.

**Topic of the Project:**

**What we are doing:** We are creating an online website that will allow people to create custom made smoothies that they can pick up at a local store or have delivered right to their door steps. This website will contain features that are common to other well known ordering sites such as Dominoes and Pizza Hut. From the website, users will be able to create an account, create their own smoothie, and decide rather or not they would like to pick the smoothie up or have it delivered.

**The Igloo Database:** In addition to creating the online website, we are also going to create a database in order to store all of the information generated from the people using the website. This database is going to be online and is called Igloo. The IGLOO database will keep track of all of the orders that are made online as well as store certain orders that are tailored to customers. In the end, the IGLOO database is going to allow people to order tasty cold treats anytime during the day.

**English Description of Database:**

* The Igloo database is only going to be accessible to three types of users: Admin, Registered Users, and Casual Users. Each user is going to live in a specific “paradigm” inside the Igloo Database.
  + - * Admin: The admin is going to live in the “Menu” paradigm of the database. The admin is going to have the ability to add items or remove items from the database. For example, an administrator can go on the database and add new items to the menu giving them a price or an administrator can remove certain item from the menu.
      * Registered Users: Registered Users are going to live in the “Account” and “Receipt” Paradigm. Registered users will have the ability to have their last order saved on the system as well being able to receive certain discounts. For example if a certain user love a specific type of smoothie, then he or she will be able to log on the application and click order without having to go to the process of making that order again.
      * Casual Users: Casual Users are going to live in the “Receipt” Paradigm. Every order that is made by a casual user is going to be stored in the Igloo Database
* The Igloo database is organized into three different paradigms: Receipt, Menu, and Account.
* Receipt Paradigm: In this paradigm the database is going to store all of the orders that are made online. Each order is going to contain a unique ticket number as well as the first name, last name, address, what was ordered, payment option, and the date and time the order was placed.
* Account Paradigm: In this paradigm the database is going to store all of the users who have made an account on the website. Each account is going to have a unique username. The account is going to store specific information (first name, last name, address, payment option) for the registered user.
* Menu Paradigm: In this paradigm the database is going to store all of the items that are included on the menu. Each item is going to have a unique ID associated with it as well as the price of the item.

**System Environment:**

The system environment is going to be Windows Desktop.

**Operating System:**

We are creating this application using Windows 10. Since our application is going to be web based, it should work on all operating systems.

**Programming Environment:**

We are going to use Notepad++ for our programming environment.

**Language:**

We plan on using multiple languages for this application. SQL is going to be used to interact with the online database. HTML is going to be used along with CSS for the formatting of the web site. PHP is going to be used as well and will serve as the main driver for the entire application connecting all of the modules together.

**Project Plan:**

**Planning Phase:** January 8, 2017 to January 31, 2017

Activities to Complete:

-Figuring out how we are going to approach this project.

-Deciding on the platform to create the project

-Deciding on the language to use for the project

Deliverables:

-Having a blueprint and game plan for creating this project

Milestone:

-Getting started on the creation of the application

**Creating Phase:** February 1, 2017 to February 28, 2017

Activities to Complete:

-Getting the foundation of the application working

-Making a program that can store registered users accou nts as well

as make orders. Also being able to change the items on the menu

Deliverables:

-A prototype version of the application

Milestone:

-Being able to implement the features

**Features/Testing Phase:** March 1, 2017 to April 10, 2017

Activities to Complete:

-Including additional features that will make our application unique

-Testing all of the additional features

-Fixing any bugs that may come up in the application

Deliverables:

-Finished Application

Milestone:

-Presenting the final product to the class

**Responsibilities of Each group member:**

This project is going to be developed by two developers Alex Indihar and Jonathan Shields. Since this is a two member project, responsibilities are going to be split evenly between the two members.

**Responsibilities for Alex Indihar:**

-Creating the website

-Formatting the layout of the website

-Connecting the Database Online

-Interacting the database with the website

-Contributing to the semester report

-Managing the online Database

**Responsibilities for Jonathan Shields:**

-Creating the website

-Formatting the layout of the website

-Connecting the Database Online

-Interacting the database with the website

-Contributing to the semester report

-Managing the online Database

**Section 2: Formal Description and Implementation**

**Overview of Section:**

This section is going to be responsible for describing the formal description of the IGLOO database as well how it is going to be implemented in terms of relations. An ER model is being used in order to describe the conceptual model of our database. The ER model is going to list the entity types and the attributes that are associated to each entity type. Towards the end of this section we are going to take the values from the ER diagram and map them into the relations for the database.

**Listing Entity Types for the Database**

1. tbAccount
2. tbOrder
3. tbMenu
4. Custom Menu

**Explanation of Entity Types**

**tbAccount:** tbAccount is an entity that represents registered users. Customers will have the option to apply for an account on the website. The moment the customer apply for an account they become a registered user and all of their information will be stored. Registered users will have an easier time placing an order online.

**tbOrder:** tbOrder is an entity that represents a receipt. This entity is going to be used to describe two things: what was ordered from the menu and who placed the order. Every order that is made online is going to be represented by this entity.

**tbMenu:** tbMenu is an entity that represents the menu of the restaurant that contains the specialty items. Specialty items are items that are going to change throughout the year depending on the seasons and special holidays.

**Custom Menu:** Custom menu is an entity that represents the menu of the restaurant that contains the standard items. Standard items are items that are going to stay the same throughout the year. Customers can always expect these items to be here. Custom Menu is a weak entity because it does not have any key attributes. Since Custom Menu is a weak entity every entity in the collection must be in total participation with the tbOrder entity. This will make tbOrder the owner of Custom Menu; therefore, every entity in the Custom Order type belongs to tbOrder.

**Listing Attributes for tbAccount**

* Email(Primary Key)
* Password
* Fname
* Lname
* Address
* Phone
* Card Number (unique)
* Security Code
* Expire Date
* Card Name

**Explanation of Attributes for tbAccount:**

**Email**(Primary Key): The username for each registered user is going to be their email address. By default this attribute will be unique because emails are unique and cannot share the same name. This will also provide a way to send special announcements and discounts to every user that has an account with the database. This will be an atomic, single-valued, and stored attribute.

**Password**: Registered users will have their own password that will be use to log in to their accounts. This will be an atomic, single-valued, and stored attribute.

**Fname**: This attribute will store the first name. Whenever a user log into the account his/her first name will be displayed on the screen. This will be an atomic, single-valued, and stored attribute.

**Lname**: This attribute will store the last name. Whenever a user log into the account his/her first name will be displayed on the screen. This will be an atomic, single-valued, and stored attribute.

**Address**: The location of where the user would like the smoothie to be sent to is going to be stored on the database as well. This will be an atomic, single-valued, and stored attribute.

**Phone**: The phone number is going to be used to let the user know whenever the order is ready. This will be an atomic, single-valued, and stored attribute.

**Card** **Number** (unique): Registered users will have the option to store their credit card information. If they decide to then their card number is going to be stored. This will be an atomic, single-valued, and stored attribute.

**Security** **Code**: The security code on the card is going to be stored. This will be an atomic, single-valued, and stored attribute.

**Expire** **Date**: The expiration date on the card is going to be stored. This will be an atomic, single-valued, and stored attribute.

**Card** **Name**: The name on the card is also going to be stored. This will be an atomic, single-valued, and stored attribute.

**Listing Attributes for tbOrder**

* Order Number (Primary Key)
* Order Time (Primary Key)
* Order Style
* Email
* Phone
* Order Price
* Pay Style
* Card Number

**Explanation of Attributes for tbOrder:**

**Order** **Number** (Primary Key): Every time an order is placed online the order number is going to increase. The Order Number is going to keep track of current total of how many times an order has been ordered from the website.

**Order** **Time** (Primary Key): The order time is going to document the exact time an order was place online.

**Explanation on the key** (Order Number, Order Time): Both the Order number and the Order time are going to serve as the primary keys for the tbOrder entity. Customers are able to select multiple items within one order. For example, a customer can select a strawberry smoothie and a blueberry smoothie and both items will share the same order number. However the times will be different for each item because of how the website is designed. Therefore both attributes will be used to uniquely identify every entity in the collection. And the Order Number attribute can be used to see all of the items that were included within the order.

**Order** **Style**: Order style is going to let us know if the user ordered takeout or delivery.

**Order** **Price**: This is going to present the total price of the time.

**Pay** **Style**: Pay style is going to let us know if the user decided to pay in cash or with a credit card.

**Card** **Number**: If the user decided to pay with a card then the credit card number will be stored in the database as well.

**Listing Attributes for tbMenu**

* Item Name (Primary Key)
* Item Price
* Item Description
* Item Size

**Explanation of Attributes for tbMenu:**

**Item** **Name** (Primary Key): This attribute is going to be the name of the specialty item that we have available for the customers. No specialty item will have the same name and because of that this can serve as the primary key. This will be an atomic, single-valued, and stored attribute.

**Item** **Price**: This will list the price of the specialty item. This will be an atomic, single-valued, and stored attribute.

**Item** **Description**: This will include a description of the specialty item describing all the ingredients that is included with it. This will be an atomic, single-valued, and stored attribute.

**Item Size:** This will represent the size of the item. All items on the menu can be purchased for small, medium, or large. This will be an atomic, single-valued, and stored attribute.

**Listing Attributes for Custom Menu**

* Item Name
* Item Price
* Item Size
* Flavors
* Toppings

**Explanation of Attributes for tbMenu:**

**Item Name:** This attribute is going to be name of the standard item on the menu. This will be an atomic, single-valued, and stored attribute.

**Item Price:** This will list the price of the standard item. This will be an atomic, single-valued, and stored attribute.

**Item Size:** This will represent the size of the item. All items on the menu can be purchased for small, medium, or large. This will be an atomic, single-valued, and stored attribute.

**Flavors:** This will represent the flavor of the smoothie. There will be many flavors to choose from and customers will have the ability to mix their flavors. As a result, this will be an atomic, multi-valued, and stored attribute.

**Toppings:** This will represent the toppings that the customers can put on their smoothie. There will be many toppings to choose from and customers will have to ability to add more than one topping on their smoothie. As a result, this will be an atomic, multi-valued, and stored attribute.

**Explanation of the Relationships between Entities**

**N**

**1**

orders

tbOrder

tbAccount

**Oders Relation:** Orders is a 1:N (one to many) relationship type between tbAccount and tbOrder. Both entities have a partial participation with the relationship. Registered users do not have to make an order online and every order that is made online does not have to be come from a registered User. Registered Users have the option to place as many orders as they like (one account can have many orders). However, each order is tied to one person (rather that person is a registered user or not).

1

N

Item Name

tbMenu

tbOrder

**Item Name Relationship:** Item name is a 1:N (one to many) relationship type between tbMenu and tbOrder. Both entities have a partial participation with the relationship. It is possible for an entity in the tbMenu collection to never appear in the tbOrder entity. For example, if none of the registered users select a specific item from the specialty menu then that particular item will never appear in the tbOrder entity. Specific items in the specialty menu may appear more than once in the tbOrder collection. However, each entity in tbOrder will contain one item from the specialty menu.

N

1

Belongs to

Custom Menu

tbOrder

**Belongs to Relationship:** Belongs to is a 1:N (one to many) relationship type between Custom Menu and tbOrder. Since Custom Menu is weak entity type, the “Belongs to” relationship the identifying relationship and tbOrder is the identifying entity type. Every entity in the Custom Menu collection must be in total participation with the relationship. An entity in the Custom Menu type cannot exist by itself and must be related to an order. A specific combination of items from the Custom Menu may appear twice in the tbOrder entity type. For example it is possible for two people to order the same custom smoothie (same flavor, same size, same toppings) and both of those orders will be stored. However, each entity in tbOrder will contain only one item from the custom menu.

tbAccount

**1**

Orders

**N**

tbOrder

**N**

**N**

Owns

Item Name

**1**

**1**

Custom Menu

tbMenu

**Mapping the ER diagram to relations**

**tbAccount(**Email, Password, First Name, Last Name, Address, Phone, Card Number, Security Code, Expiration Date, Card Name**)**

**tbOrder(**Order Number, Order Time, First Name, Last Name, Address, Order Style, Email, Phone, Item Name, Item Size, Flavors, Toppings, Order Price, Pay Style, Card Number**)**

**tbMenu(**item name, item Description, Item Price**)**

**Explanation of Mapping the ER to a relational schema:**

**tbAccount:** Creating the tbAccount relation from the ER diagram was pretty straightforward. The name of the relation reflects the name of the entity type in the diagram. All of the attributes that were included in the ER diagram are also included in this relation. The email attribute is going to serve as the primary key for the relation.

**tbOrder:** Creating the tbOrder relation from the ER diagram was also straightforward. The name of the relation reflects the name of the entity type in the diagram. All of the attributes that were included in the ER diagram are also included in this relation. The order number and order time attributes are going to serve as the primary key for the relation. The email attribute is a foreign key that is going to reference the email attribute from the tbAccount relation. The attributes first name, last name, address, and phone number are going to come from the user that placed the order (registered or unregistered). These attributes are included in order to provide detail on “who” placed the order. The item name attribute is a foreign key that is going to reference the item name attribute from either the tbMenu relation or the Custom Menu relation. The attributes item name, item size, flavors, and toppings are going to come from either the tbMenu relation or the Custom Menu relation. These attributes are included in order to provide detail on “what” was ordered.

**tbMenu:** Creating the tbMenu relation was very simple. The name of the relation reflects the name of the entity type in the diagram. All of the attributes that were included in the ER diagram are also included in this relation except the size. The size is an attribute that the user is going to pick directly from the website. For example, if the customer selects an item from the specialty menu then we know already know all of the toppings and flavors that are included with that particular item (that information is stored in the database). However, we do not know the size in which customers would like for that item to be ordered and therefore they will select the size from the website.

**Custom Menu:** There is not an actual relation for Custom Menu in our database. Now normally in most databases, there will be a relation for every entity that is presented in the ER diagram. Since we are designing a web base application, we are able to use the web server in such a way to account for the “custom menu” relation. Since the Custom menu will contain items that will remain the same throughout the year, this menu can be created on the website and essentially “live” on the server. The website will serve as the middle tier between taking information from the user (in regards to the smoothie) and storing it in the tbOrder relation. Customers will be able to select the item of their choosing along with the flavor, size and toppings directly from the website and all of that information will be stored in tbOrder. This is the reason why the Custom Menu entity was described as a weak entity in the ER diagram because every specific combination of a custom item belongs to a specific order.

**Relationship database schema**

**tbAccount**



**tbOrder**



**tbMenu**



**Create Table Statements:**

CREATE TABLE tbAccount(

email VARCHAR(64),

pass VARCHAR(32),

fname VARCHAR(32),

lname VARCHAR(32),

address VARCHAR(64),

phone CHAR(10),

cardNum CHAR(16),

securityCode CHAR(3),

expireDate CHAR(5),

cardName VARCHAR(21),

PRIMARY KEY (email),

UNIQUE (cardNum)

)";

CREATE TABLE tbOrder(

orderNum BIGINT,

orderTime TIMESTAMP(6),

fname VARCHAR(32),

lname VARCHAR(32),

orderStyle CHAR,

email VARCHAR(64),

phone CHAR(10),

item VARCHAR(8),

itemSize VARCHAR(4),

flavors VARCHAR(8000),

toppings VARCHAR(8000),

orderPrice FLOAT,

payStyle CHAR,

cardNum CHAR(16),

PRIMARY KEY (orderNum),

)";

CREATE TABLE tbMenu(

itemName VARCHAR(64),

itemPrice FLOAT,

itemDesc VARCHAR(8000),

PRIMARY KEY (itemName)

)";

**Diagram of the Architecture of the system**

Case 1: Customer Orders a specialty Item

tbAccount

tbMenu

tbOrder

Main Menu

Specialty Menu

Order Information

**Explanation of Case 1:**

**Step 1:** Customers are going to start with the main menu of the website. From the main menu customers will be given two options when it comes to ordering a smoothie. They will have the option to build their own smoothie or to pick a specialty smoothie. For this particular case, the customer has decided to pick a specialty smoothie.

**Step 2:** Selecting the specialty smoothie option from the main menu will direct the customer to a page that will contain a table listing all of the specialty items that are available. Everything that is presented on the table will come from the tbMenu table in the database. tbMenu will contain all of the specialty item and will have to ability to be adjusted by an administrator. So essentially, all of the information for the specialty item will be pulled from the tbMenu table and presented on the website. Customers will then have the ability to select which item they would like to order.

**Step 3:** After selecting the item to order, customers will be directed to the order information page. This page is going to require the customer to fill in the following: first name, last name, email, phone number, and payment method. If the customer is a **registered user** then he/she will not have to fill in those values because they will already be prefilled. However, if the customer is not a registered user then he/she will have to fill those values in every time an order is placed online. Once all of the information is provided then the customer can place the order.

**Step 4:** Once an order is placed, all of the information that was provided on the customer and the particular item ordered will be stored in the tbOrder table in the database. tbOrder will contain the receipt information of every item ordered from the website.

Case 2: Customer Orders a Custom Item

tbAccount

Main Menu

Custom Menu

tbOrder

Order Information

**Explanation of Case 2:**

**Step 1:** Customers are going to start with the main menu of the website. From the main menu customers will be given two options when it comes to ordering a smoothie. They will have the option to build their own smoothie or to pick a specialty smoothie. For this particular case, the customer has decided to build a custom smoothie.

**Step 2:** Selecting the custom smoothie option from the main menu will direct the customer to a page that will contain multiple tables that will be used in the process of creating a smoothie. From the tables listed, customers will have the option to build a smoothie exactly how they want it. Each smoothie is broken down into the following components: flavor, size, and toppings. Customers will have the option to select the flavor, the size, and the toppings that they would like on their smoothie. Once customers have created a smoothie to their liking they can place and order.

**Step 3:** After selecting the item to order, customers will be directed to the order information page. This page is going to require the customer to fill in the following: first name, last name, email, phone number, and payment method. If the customer is a **registered user** then he/she will not have to fill in those values because they will already be prefilled. However, if the customer is not a registered user then he/she will have to fill those values in every time an order is placed online. Once all of the information is provided then the customer can place the order.

**Step 4:** Once an order is placed, all of the information that was provided on the customer and the particular item ordered will be stored in the tbOrder table in the database. tbOrder will contain the receipt information of every item ordered from the website.

Case 3: Customer creating an Account

Main Menu

tbAccount

Account

Menu

**Explanation of Case 3:**

**Step 1:** Customers are going to start with the main menu of the website. From the main menu customers will be given two options when it comes to ordering a smoothie. They will have the option to build their own smoothie or to pick a specialty smoothie. For this particular case, the customer has decided to create an account

**Step 2:** Whenever a customer is creating an account they will be directed to the “Account Menu” page. On this page, customers are required to provide the following information: email, password, first name, last name, address, and phone number. They will also have the option to include their credit card information; however, it is not mandatory. Once all of the required information has been provided the customer can create the account.

**Step 3:** Once an account is created, all of the information is going to be stored in the tbAccount table in the databse. tbAccount is going to contain all of the customers that have created an account with the website. This account will be used to speed up the ordering process whenever a customer is making an order.

**Section 3: Group Details**

**Overview of Section:**