**Institute of Technology, Carlow**

**Project Documentation**

**for**

**Customer Invoice System**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Name:** Huu Chi Huynh

**Date:** 08/04/2022

**Tutor:** Jason Barron

# 

# Table of Contents

Section Page

Description 3

Requirements 4

Database tables 5

ER Diagram 6

Interesting source code snippets 8

Tests 15

## Description:

This project was commissioned by Jason Barron in the Object Orientated Software Development module in IT Carlow. // Write more

The document contains the following sections:

* Description
* Requirements
* Database tables
* ER Diagram
* Interesting source code snippets
* Tests

## Requirements:

A Customer Invoice Management system is a system that can be used to manage customer invoices. The customers can login and view their invoices and so as the companies as well. The companies which use the system can create invoices which they can hand out to customers.

Customers cannot login or signup into the system until a company has created a customer profile for them.

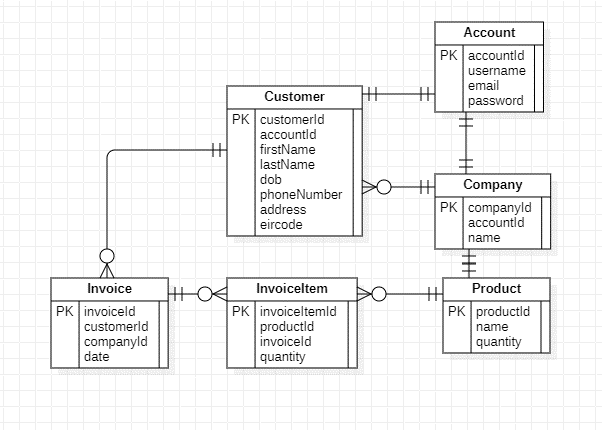
Products can also be added into the system with a quantity and cost defined by the company.

Invoices must contain the customer information and the products that company has added to the invoice.

The information of the product or invoice can be changed later by the company which created the item.

## Database tables:

## ER Diagram:



## Interesting source code snippets:

The following function is really interesting as it handles most of the login of the user, checking the username and password and creating the session id if the user has successfully logged in.

|  |
| --- |
| /\*\*       \* Checks both the username and password inputted       \* If the username and password are found then       \* Create / Update the session id and upload it into the database       \* @param username       \* @param password       \* @return (PacketResult.SUCCESS and sessionId) if successful       \*/      private static Packet requestLogin(String username, String password) {          Packet accountIdPacket = getAccountId(username); // Does username exist          if (accountIdPacket.getResult() != PacketResult.SUCCESS) { // Check if packet was successful              return accountIdPacket;          }          int accountId = (int) accountIdPacket.getInformation();          if (accountId > 0) { // Verify if account username exists              Packet canLoginPacket = canLogin(username, password); // Attempt login with username and password              if (canLoginPacket.getResult() == PacketResult.SUCCESS) {                  String sessionId = UUID.randomUUID().toString(); // Create session id                  DatabaseHandler handler = DatabaseHandler.getInstance();                  if (handler.isConnected()) {                      Object[] info = {username, password};                      // Attempt to get session id from username and password                      List<List<Object>> result = handler.get("Application.SessionId.sessionId FROM Application.SessionId JOIN Application.Account ON Application.Account.accountId = Application.SessionId.accountId AND Application.Account.username = ? AND Application.Account.password = ?", info, 1);                      if (result.isEmpty()) { // No session id exists                          if (handler.isConnected()) {                              Object[] args = {accountId, sessionId};                              boolean success = handler.insert("SessionId(accountId,sessionId) VALUES (?,?)", args);                              // Create session id                              if (success) {                                  return new Packet(PacketResult.SUCCESS, sessionId);                              } else {                                  return new Packet(PacketResult.DATABASE\_ERROR);                              }                          } else {                              return new Packet(PacketResult.CONNECTION\_ERROR);                          }                      } else { // Session id already exists                          Object[] args = {sessionId, accountId};                          if (handler.isConnected()) {                              // Update session id                              boolean success = handler.update("SessionId SET sessionId=? WHERE accountId=?", args);                              if (success) {                                  return new Packet(PacketResult.SUCCESS, sessionId);                              } else {                                  return new Packet(PacketResult.DATABASE\_ERROR);                              }                          } else {                              return new Packet(PacketResult.CONNECTION\_ERROR);                          }                      }                  } else {                      return new Packet(PacketResult.CONNECTION\_ERROR);                  }              } else {                  return canLoginPacket;              }          }          return new Packet(PacketResult.ERROR\_OCCURRED); // No account exists??? (This should never happen)      } |

The following function is interesting as it handles the session id checking of the user and is called in every login action

|  |
| --- |
| /\*\*       \* \*       \* Checks if account id and session id matches inside of the SessionId       \* database table. [USED TO CHECK IF USER IS SIGNED IN OR ACCOUNT EXISTS]       \*       \* @param accountId Account Id of user       \* @param sessionId Session Id of current session       \* @return (PacketResult.SUCCESS) if the session id exists       \*/      private static Packet checkSessionId(int accountId, String sessionId) {          Object[] info = {accountId, sessionId};          DatabaseHandler handler = DatabaseHandler.getInstance();          if (handler.isConnected()) {              // Check if session id with account id exists in Session id table              List<List<Object>> result = handler.get("accountId FROM Application.SessionId WHERE accountId = ? AND sessionId = ?", info, 1);              if (result.size() >= 1) {                  return new Packet(PacketResult.SUCCESS);              } else {                  return new Packet(PacketResult.BAD\_REQUEST);              }          }          return new Packet(PacketResult.CONNECTION\_ERROR);      } |

## Tests: // 3 more

**Test Case – Sign up Company**

|  |  |
| --- | --- |
| Name | TC-01 |
| Requirements | Secure |
| Precondition | User must be connected to database |
| Steps | User enters login screen  User enters the username and password  User submits the information  System brings user to company menu |
| Expected Results | User has logged in and is in the company menu |

**Test Case – Login Customer**

|  |  |
| --- | --- |
| Name | TC-02 |
| Requirements | Secure |
| Precondition | User must be connected to database |
| Steps | User enters login screen  User enters the username and password provided by the company  User submits the information  System brings user to customer menu |
| Expected Results | User has logged in and is in the customer menu |