PRODUCEPOS DEVELOPMENT REPORT

Overview

We would like to present you with ProducePOS v1.0. We have developed a functional Point of Sale (POS) system for use in a produce vendor. Upon opening ProducePOS you arrive at the main screen which allows you to choose whether to make a sale or perform a PLU lookup. After making your choice, ProducePOS employs an encrypted employee database which stores ID numbers and their associated passwords which are needed to get through the login screen that safeguards the functional areas of the system from unauthorized users. Once in the sale screen ProducePOS will allow a user to enter PLU numbers into the scanner area, search the connected produce database for pricing, keep a running total of all scanned items, display all scanned items, perform the HST calculation and determine possible change owed to the customer. The PLU lookup function can be used to find the price of any item in the database should a customer inquire.

Target Market

ProducePOS was developed with smaller grocery outlets like Farm Boy or Farmer's Market operatives in mind.

UML Diagram Description

Before development we drew four UML diagrams. The first of which is a Class Diagram that details how each class in the program interacts. It shows that Main inherits from Application and all events are handled by Controller which employs SystemLogic to perform the PLU scanning, database searching and all calculations.

The second is a Sequence Diagram which depicts how SystemLogic interacts with both the Produce and Employee databases. SystemLogic searches the employee database for a matching employee ID & password combination and the produce database for a price that corresponds to the PLU entered by the user. If the user makes an error ProducePOS informs them.

The third and fourth are State Diagrams which depict the different states that ProducePOS is in when performing either a sale or a PLU lookup. During a sale PLU numbers are entered. ProducePOS displays a list of the scanned items, after tax cost and change owed. During a PLU lookup, ProducePOS requests login credentials then allows a verified user to enter a PLU number and displays the price of that item.

RETROSPECTIVE

The easiest function of ProducePOS to implement was the login screen as it is made up entirely of two text boxes which are connected to the employee database and the only function it serves is to verify that employee ID numbers and passwords are entered correctly. It also only throws a single exception when an ID or password is incorrect which simply prompts the user to enter valid credentials.

The most difficult piece to implement was the sales screen as it required the most functionality; a textbox to enter PLU numbers in, a display box to list the scanned items, a display for the running total, a textbox to enter the amount a customer paid and a display box that showed the accurate change owed to that customer. This needed the most coding as well as required us to learn SQL queries to properly implement the database connectivity.

If we were to start this project over, we would ensure two things. First, we would decide upon a formal communication plan. Several times during development we did not accurately communicate where each team member was with their respective assignments which led to confusion, frustration and duplication of work. A formal agreement on how and how often to communicate could have mitigated this. The second thing would be to ensure full comprehension of how to properly use Git and Github to share work. A lack of understanding of Git led to some work being overwritten and therefore lost which led to us almost entirely relying on one team member handle the repository.

We feel there are three important things we learned from this development process. The first is the importance of frequent and honest communication as mentioned above. The second is the

value of the UML modelling process. As instructed, we started by creating our UML diagrams which made designing the UI and subsequently programming ProducePOS much easier as we maintained a vivid image of what our end-product would look like. The third, and possibly most valuable lesson we learned was the importance of iterative planning. When we first designed ProducePOS we envisioned a much bigger piece of software with more commercial functionality including; a sales tracker, an inventory management system, a back office wherein a manager could add or remove employees, a coupon database to allow for easy discounts to be applied during checkout and a return screen to name a few. Our initial UI designs included all these features but as due to time constraints and poor communication we decided to adopt an agile approach and focus on building the basic system first and add functionality in future updates.