Michigan State University

Team Meijer

Tablet-Based Point-of-Sale System

Project Plan

Fall 2011

Meijer Contacts:
Scott Pallas
Murali Rajagopalan

Team Members:
Riti Adhi
Peter Rifel
Andrew Rockwell
Mark Sun

TABLE OF CONTENTS

1. Executive Summary

The Tablet-Based Point-of-Sale System allows Meijer shoppers to purchase their items with a new intuitive interface on state of the art Windows-based tablet PCs. Our tablet-based system fulfills a very similar use case to that of the self-checkout stations currently used at Meijer. A customer begins their transaction, and begins scanning their item. The system receives input from Universal Product Code (UPC) barcode scanners to identify the shopper's items. Once the barcode has been scanned, it is sent to Meijer's Virtual Point-of-Sale (VPOS) system which returns information about the product such as name and price. The scanned item will then be displayed on the user interface on two tablet devices.

The objective of our system is to speed up the self-checkout process. Currently at Meijer, when a user uses the self-checkout stations many actions require the assistance of an employee. If the user scans an item which generates an exception, such as age verification on a bottle of wine, an employee must be called over to the station, blocking other shoppers from using the lane. In the tablet-based system, scanning items and finishing the transaction are two separate steps. When an exception occurs, a message is displayed to the user which they can simply confirm and then resume scanning. When they have finished scanning their items, they can then proceed to move down to a payment station to resolve the exceptions. This allows the next shopper in line to quickly start the checkout process. In addition, it resolves the issue of having employees monitor multiple self-checkout stations, which can cause long waits, and create a bad experience for the customer.

Our system is also a more flexible alternative to Meijer's current self-checkout stations. Since the system is only composed of off-the-shelf hardware such as a Windows PC and two Windows tablet PCs, if one of the systems stops working, it can easily be replaced with another. This means checkout lanes are closed for a shorter duration if there are issues and repairs are less costly.

2. Functional Specification

The main focus of the project was to build a brand new user interface. The user interface makes purchasing items a richer experience by providing complete visibility to items that are scanned in, displaying only the transaction totals the user cares about, and detailed item exception information. When the user scans an item, the item's barcode provides the system with the information it needs to determine what item is being scanned, the item's weight, and the current price. The tablet user interface provides the user with common functionality such as cancel transaction, end transaction, request for help, and remove item. The tablet also displays the current total as well as how much money the shopper has saved.

The system we have built communicates with Meijer's existing Virtual Point-of-Sale system for various information such as product name, price, and restrictions, for example a minimum age for purchase.

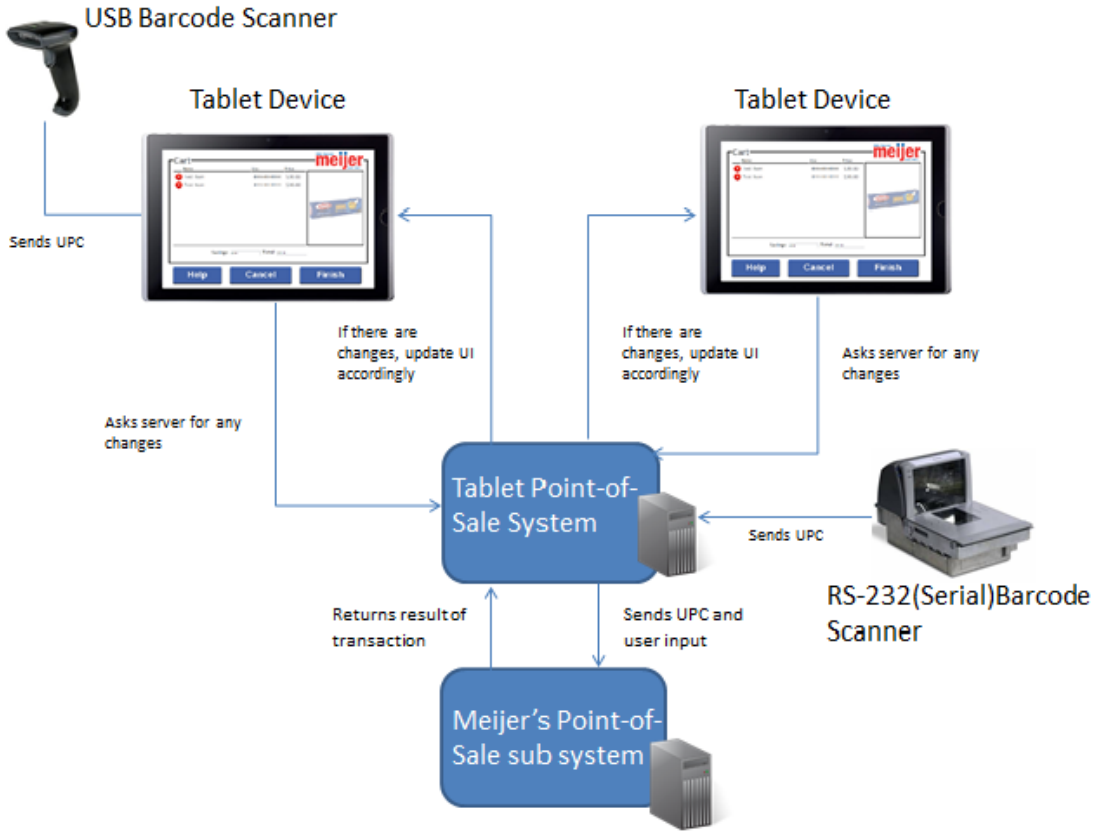

Figure 1: Meijer Table-Based Point-of-Sale System Diagram

3. Design Specification

3.1 Overview

The use case diagram in Figure 2 depicts the processes available to the user of the self-checkout system. The main components in the diagram are the customer and Meijer's Virtual Point of Sale System. The customer can login with their mPerks information, scan an item, request for help, cancel transaction, or finish transaction. If the user scans an item which triggers an item exception, then assistance from an employee will be required. The Meijer Virtual Point-of-Sale system updates transaction information to the cart and gets item details.

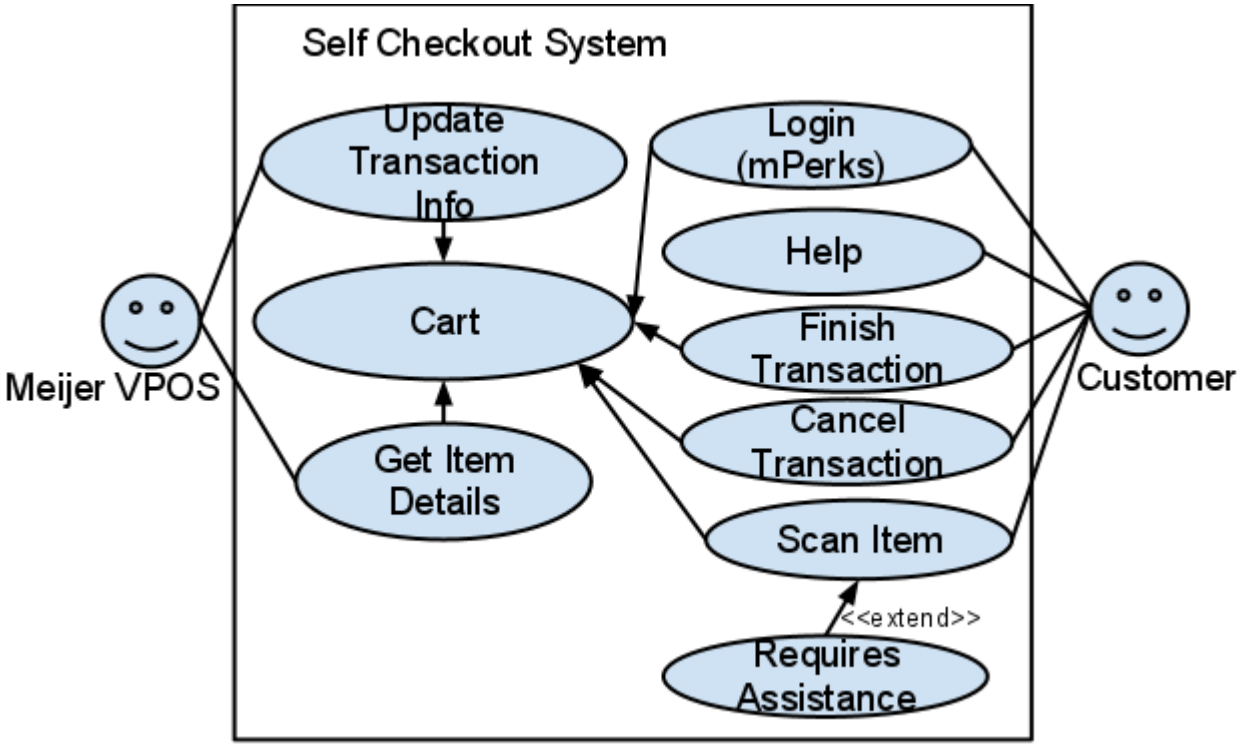

Figure 2: Use Case Diagram

3.2 User Interface

When a user arrives at the self-checkout station, they will see a new user interface. The user interface is simple and designed with the user in mind. It is meant to be very intuitive, so shoppers can effectively complete their transaction with ease.

3.2.1 mPerks Login

The user must login into their mPerks account in order to begin self-checkout. mPerks is currently a program designed so users can receive and redeem discounts at the checkout when they enter their 10-digit mPerks Number (mobile phone number) and 4-digit mPerks PIN (used to sign into their mPerks account). Our system uses the mPerks number and mPerks PIN as a way of pausing transactions to be continued later at a payment station. The user enters their mPerks number and mPerks PIN using the numeric buttons. The back arrow has the same functionality as backspace on a keyboard. The user presses the “Start” button to begin scanning items. If the user enters their mPerks information incorrectly, a message will appear and the user will be able to reenter their mPerks number and mPerks PIN.

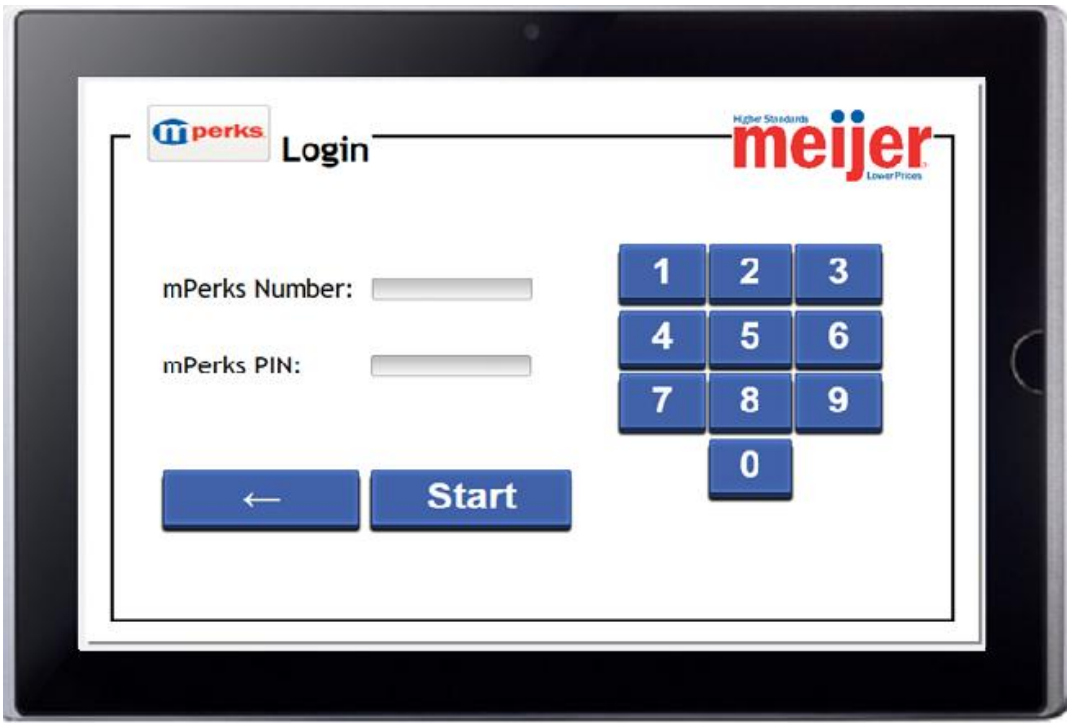

Figure 3: Start

3.2.2 Scan Items

This is the main screen. When the user scans the barcode of an item the item name, universal product code (UPC), and price are added to the cart. The image of the item that is scanned is shown in the box on the right. The savings and total amounts are also displayed and updated as items are scanned. Deposits for bottle returns and discounts on items appear below the item name. If the user would like to remove an item from their transaction, they can simply press the red “X” to the left of the item name and the item will be deleted from the cart.

Figure 4: Main

3.2.3 Item Exception

When a user scans an item, it may generate an item exception. An item exception is an item which does not get scanned correctly or requires verification such as an item which requires age verification, price verification, or product warranty information. When an item exception occurs, the user is redirected to a screen with a message and the image of the item that was scanned incorrectly. The user can press the “Help” button to request for assistance from an employee or press the “Continue” button to go back to the main screen. When the user resumes the transaction at a payment station, item exceptions will be displayed and can be resolved by an employee.

Figure 5: Item Exception

3.2.4 Request for Assistance

The user may request assistance in the case of an item exception or while the user is scanning in items. If the user is in need of assistance from an employee, they can press the “Help” button. When a user requests for help, a help message will be displayed to the user requesting them to wait for an employee. Once the user has been helped, they can press the “Continue” button to return to the main screen

Figure 6: Help

3.2.5 Cancel Transaction

If the user would like to cancel the transaction they can press the “Cancel” button. When the user confirms that they would like to cancel the transaction, they are redirected back to the mPerks login page. The transaction is terminated and no information about the transaction is saved.

Figure 7: Cancel Dialog

3.2.6 Finish Transaction

When the user would like to end the transaction, they can press the “Finish” button, which suspends the transaction and redirects them back to the mPerks login page. The transaction can be resumed at a payment station by entering the associated mPerks number and PIN.

Figure 8: Finish Dialog

4. Technical Specs

4.1. System Architecture

The three main components of the system are

1. an HTML 5 user interface run on tablets
2. a .Net web service communicating via SOAP to the existing Virtual Point-of-Sale system
3. a .Net interface for barcode scanners

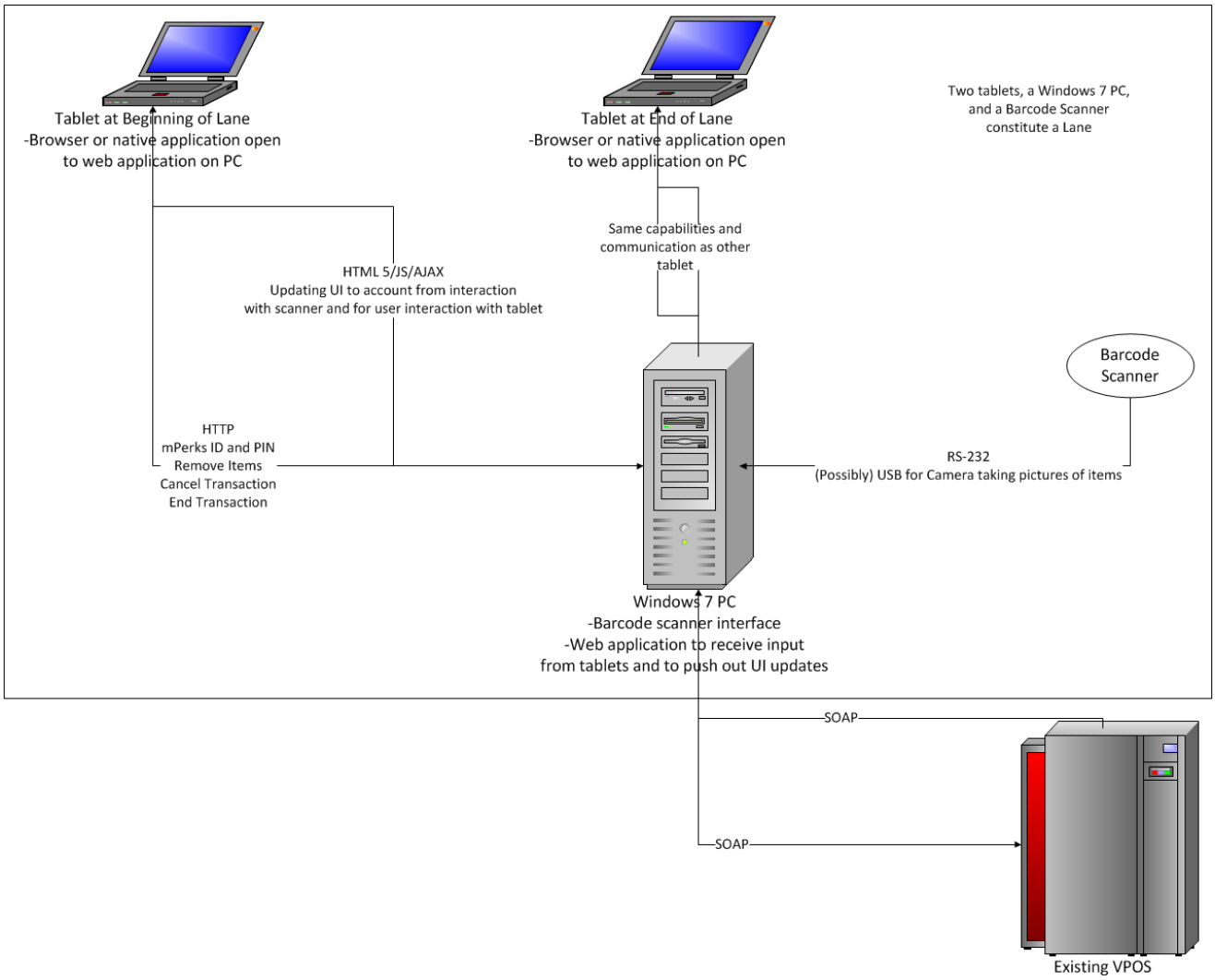

Figure 9: System Architecture Diagram

The user level component is the user interface run on the tablet. All interaction from the users and administrators come from the tablet. The tablet opens to the web application and communicates via form posts to the web app. Many calls are caused by the barcode scanner

interface triggering calls in the web app by scanning an item into the tablet. The web app then makes according requests to the existing Virtual Point-of-Sale system via SOAP calls. The Virtual Point-of-Sale responds to the SOAP calls, which then causes the web app to send back HTML to the tablet to update the user's interface.

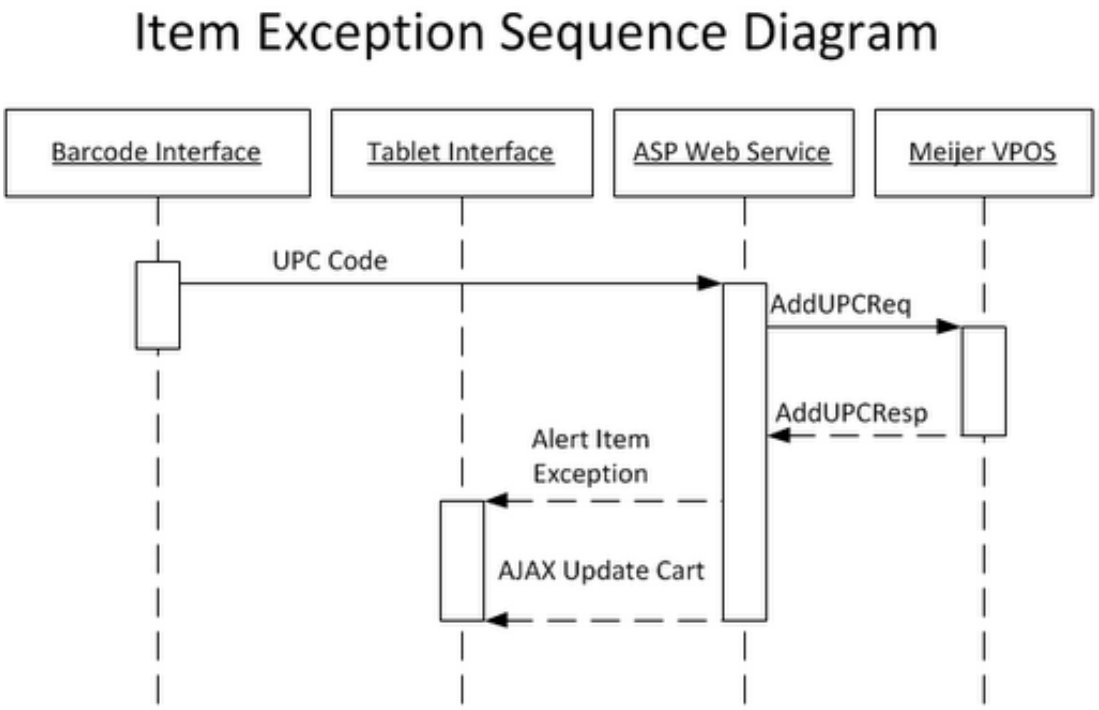

Figure 10: Item Exception Sequence Diagram

The sequence diagram above illustrates a scenario where an item exception occurs, for example a bottle of wine would trigger an exception for age verification. First, the barcode scanner interface reads the UPC and sends it to the tablet application, which then passes the UPC onto the web application. The web application then sends an AddUpc XML request to the Virtual Point-of-Sale system. The VPOS responds with an AddUpc XML response containing information such as the name and price of the item as well as item exception information. The web application parses the response and generates an alert to the tablet and the cart is then updated on the tablet via an AJAX response.

4.2. User Interface

The HTML 5 user interface is run on the tablet through a native .Net application that is simply a web browser control configured to point to our site. The application is configured to block the user to leave the application or have access to the address bar. The user interface from a design perspective is explored in detail in the Design Specs, section 3. The user interface is a

part of the MVC web app that comprises the web service. It is also comprised of ASP.Net components as needed.

4.3. Web Service

The web service is a .Net 4 MVC 3 application run inside of IIS 7 on our Windows 7 machine. The service communicates with Meijer's existing Virtual Point-of-Sale system via SOAP 1.1. The service requires hitting the Virtual Point-of-Sale system to validate the mPerks credentials and start the transaction, add items by UPC, remove items by UPC, cancel the transaction, and finish the transaction. The application's user interface is built in HTML5 and CSS3. Most actions return a full page through the normal HTTP model and the rest of the interaction is conducted via AJAX to update the scanned item list, transaction total, and total savings.

In an effort to keep code more readable and XML communication easier to use, requests are built up as objects and then serialized into XML and responses are received as text and deserialized into usable types. In production, these messages will be wrapped in a SOAP envelope with the proper HTTP headers by using a .Net 2.0 Web Reference to the existing Meijer VPOS system. Currently, the messages are sent to a server that is accessible when connected to the Virtual Private Network (VPN) which is running a web service written by the team that then forwards the messages on to the VPOS.

The web service logic is driven by 3 controllers: the User controller, the POS controller, and the Poll controller.

The User controller contains methods that render views to the user:

- Normal Views:
    - public ActionResult Start
    - public ActionResult Main
    - public ActionResult ItemException
    - public ActionResult Help
- Partial Views:
    - public ActionResult Cart
    - public ActionResult NumberPad

The actions visible from the POS controller are the actions that are triggered by scanning UPCs, pressing buttons, etc. The first five methods in the POS controller send an XML message to the VPOS system, parse the response and update the system's data accordingly. The actions it currently supports are:

- public ActionResult SendStart
    - Returns the /User/Start action if error occurs, otherwise returns the /User/Main action.
- public string AddUpc
- public string RemoveUpc

- public string Cancel
    - Terminates the transaction without saving information about it.
- public string End
    - Suspends the transaction, allowing it be resumed at another station with the associated mPerks number and PIN.
- public void ContinueFromItemException
    - Updates the database to confirm an exception has been shown.
- public void ContineFromHelp
    - Redirects the user back to the page they came from.
- public void Restart
    - This method clears the session and the database allowing a new transaction to begin. It exists in case of errors so the system can be reset.

The Poll controller contains all the actions that are used to give the site a real-time experience via AJAX calls:

- public string IsTransactionStarted
    - Redirects the user to Main if the valid mPerks information was entered.
- public ActionResult UpdateCart
    - Returns a partial view of the list of items, the current picture, savings, and total.
- public string HasExceptionOccured
    - Redirects to the ItemException page when exception occurs.
- public string HasExceptionBeenResolved
    - Redirect to the Main page when exception is handled.
- public string IsTransactionDone
    - Redirects to the Start page if the transaction has been cancelled or finished.

These actions are all called via JavaScript from the client. A timer is set to fire every second to call the actions appropriate for that page, a method called polling. The success handler then processes the response, and updates the user interface accordingly.

4.4. Barcode Scanner Interface

The Barcode Scanner Interface is a set of applications written running on the Windows 7 PC that allow the web service to receive barcode information of scanned items. A scanner emulator was written to simulate the physical barcode scanner which sends meta-data for the item as well as the barcode itself over a serial port. The emulator encodes the data into messages using the same specification as the physical barcode scanner. This program was only written for testing purposes and is not intended for use in a production environment. The emulator and physical barcode scanner can send the following message types regarding the scanning of an item:

- Good read - One item was successfully read from the scanner
- Multiple reads - More than one item was successfully read from the scanner
- No read - No barcode was detected from the scan
- Items too close - Item was too close to the camera but an item was scanned

- Security camera triggered - An item may have been hidden within a stack of other items and was not scanned.
- Barcode data not matched with object - A barcode was scanned but the camera did not detect an item

Depending on the message type, the message may also contain one or more barcodes for the images it scanned. The emulator then encodes these messages and sends them over a virtual serial port that is linked with a second virtual serial port on the machine. In production, the physical barcode scanner will be plugged directly into a single serial port on the PC.

A second application running on the same machine listens for messages over the second virtual serial port and decodes them into one or more barcodes. The application then forwards these barcodes as an HTTP request to the Web Service, instructing the service to add the items to the user's cart.

As an alternative to the physical barcode scanner, the User Interface supports a USB barcode scanner plugged directly into the tablet PC. The barcode is submitted in a form invisible to the user directly within the interface. For USB barcode scanners, only the barcode itself is submitted. There is no meta-data regarding the item scan or any encoding of messages.

5. Schedule

Development Legend
UI - User Interface
Web - Web Application
Bar - Barcode Scanner

Week 1 (8/31/11 - 9/3/11)
Installed Windows Server 2008 R2
Installed Visual Studio 2010
Installed Internet Information System (IIS)

Week 2 (9/4/11 - 9/10/11)
Initial Client Conference Call
Installed Team Foundation Server (TFS)

Week 3 (9/11/11 - 9/17/11)
9/14 - Team Status Report Presentations
Native Browser Application Prototyping

Week 4 (9/18/11 - 9/24/11)
9/21 - Project Plan Presentation and Project Plan Document Due
(Web) Finish work in prototype MVC solution

Week 5 (9/25/11 - 10/1/11)
(UI) Preliminary HTML code complete
(Web) Implement known XML requests/responses
(Bar) Successfully input data from USB scanners

Week 6 (10/2/11 - 10/8/11)
(UI) Translate raw HTML in ASP.Net in MVC solution
(Web) Implement state for items in cart
(Bar) Set up serial port emulation

Week 7 (10/9/11 - 10/15/11)
(Web) Attach business logic to Views
(Bar) Begin developing the serial scanner interface

Week 8 (10/16/11 - 10/22/11)
10/17 - Alpha Presentation Due
(UI) Code views for Item Exception interface
(Web) Continue attaching business logic to views
(Web) Implement handling for item exceptions in AddItem response
(Bar) Test serial scanner interface

Week 9 (10/23/11 - 10/29/11)
10/24 - Alpha Presentation
(Web) Develop business logic for Item Exception interface, attach to views
(Web) Continue handling for item exceptions in AddItem response

(Web) Implemented a basic item exception page
(Bar) Develop the barcode scanner receiver

Week 10 (10/30/11 - 11/5/11)
Begin creating the documentation/user manual
(Web) Begin end-to-end testing against Virtual Point-of-Sale System
(Web) Tweaking UI to improve usability
(Bar) Test scanner receiver sender against emulator and Web Application

Week 11 (11/6/11 - 11/12/11)
11/7 - Beta Presentation Due - Feature Complete
Resolve issues that arise with VPOS communication
Record audio for the project video
(Web) Continue tweaking UI to improve usability
(Bar) Clean up interfaces for scanner interface and scanner receiver

Week 12 (11/13/11 - 11/19/11)
11/16 - Beta Presentation
Record screen and external video for the project video

Week 13 (11/20/11 - 11/26/11)
Stretch Goals - “Nice to Have” cleanup
Complete End-to-end testing
Begin editing of project video

Week 14 (11/27/11 - 12/3/11)
Bug Fixing!
Finish editing of project video

Week 15 (12/4/11 - 12/10/11)
12/5 - Project Video Due
12/7 - All Project Deliverables Due
12/8 - Design Day Setup
12/9 - Design Day

6. Risks

Completely New User Interface

- Many components are likely to be re-used, but we will need to consult administrative users (self-checkout employees) to determine priority use cases so the user interface makes the most sense.
- Mitigation: Take a trip out to Meijer labs as soon as possible. Ask for access to talk to employees to determined administrative use cases.
- Status: Meijer made clear that the user interface was a very important component of this project. We were given creative freedom to make our own user interface to suit our customers' needs.

Browser v. Native App

- User interface can be viewed simply in a browser, which will take more work to lock down, or in native app with a browser frame, which would require more development, but will offer a better user experience.
- Mitigation: Talk to the client to determine if they have a preference. If they do not, prototype a native app to be sure we can create what we'd like.
- Status: The team decided to use a browser which offers Kiosk Mode, a full-screen experience with many functions disabled. Part of the system set up is disabling other options

Interfacing with existing Virtual Point-of-Sale system

- Currently waiting on documentation until a contact returns from vacation. Once we have received documentation, it will take some ramp up to learn the system and make sure we're following the Meijer coding standards for it.
- Mitigation: Ramp up .Net technologies in the meantime, not be afraid to ask plenty of questions when we get documentation.
- Status: The team received access to the VPOS and is sending and receiving data successfully through the web application.

Security

- Our application will likely be handling payment data and will need to ensure that only the intended clients can connect to our service.
- Mitigation: Make sure security is a priority throughout our development process.
- Status: We do not handle payment, which significantly lowers our security concerns. We still save data in the session which we have verified is adequately secure through the natural obfuscation in .Net.