Smart Refrigerator Proposal
Test Plan

Steven Strapp, Ben Reeves, Dustin Stroup
February 14, 2012

Testing of the Smart Refrigerator will be divided into unit testing of the various subsystem and then top-level integration testing once the sub-systems have been connected. Some components used within the system, such as the Angstrom operating system and SQL database implementation, have undergone extensive test prior to use in our system will only be tested to ensure proper configuration. The principle subsystems tested will be the base station user interface, mobile user interface and network connectivity, expiration date and shopping list prediction algorithms, and integration with the BeagleBoard.

1 Subsystem Tests

1.1 Base Station User Interface Testing

The main testing focus will be on the user application, both the software running on the base station as well as the web and Android interfaces. Unit testing will be performed during development of each component, as well as integration testing of the final application. Testing will focus on usability of the interface, accuracy of expiration date prediction and shopping list recommendations, and communication with the database.

The main testing focus will be on the user application, both the software running on the base station as well as the web and Android interfaces. Unit testing will be performed during development of each component, as well as integration testing of the final application. Testing will focus on usability of the interface, accuracy of expiration date prediction and shopping list recommendations, and communication with the database.

As a final test, someone not involved in the project will test the system for usability as an end-user.

1.2 Mobile User Interface and Network Interface Testing

The web and mobile interfaces will have their own set of tests, focused on basic functionality and interoperability on various platforms. The web interface will be tested on the most popular browsers (Google Chrome, Firefox, and Internet Explorer), as well as some of the most popular mobile platforms (Android, WebOS, and iOS). The Android interface will need to be tested on various versions of the operating system. At a minimum, major versions between 2.1 and 4.0 will be tested.

1.3 Shopping List and Expiration Prediction Test

Testing of the expiration prediction and shopping list prediction subsystems will be difficult if the system's timing cannot be accelerated; testing should occur over a few minutes not a series of days. For expiration date testing a special set of UPC codes can be added with a fabricated GS1 category so they expire very quickly. The intelligence of the system can then be tested by providing feedback that these imaginary products expired more or less quickly than expected and evaluating the updated predictions. Similarly, the recommendation system will normally discretize purchase dates into intervals of days. A special mode should be added to this subsystem which will consider purchase intervals in the range of seconds; with this accelerated mode new products can be purchased every few minutes and the prediction algorithm can be verified quickly. A test sets are shown for this subsystem in Tables 1 and 2, below.

Table 1: Expiration Date Prediction Test Cases

Test Writer:Steven Strapp						
Test Case Name:		Expiration Date Prediction System Unit Tests			Test ID #:	Pred-01
Description:		Verify that expiration data prediction system makes recommendations within an acceptable margin of true expiration. Simulates expiration of products.			Type:	White Box
Tester Information						
Name of Tester:					Date:	
Hardware Ver:					Time:	
Setup:		Develop fake product codes for quick testing of expiration. System should have no previous expiration date history				
Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Enter fake product code	Expiration date should be initialized with recommended "rule of thumb" value.				
2	Provide feedback the product expired before/after recommendation	Re-scan product and shelf-life estimate should decrease/increase.				
3	Enter fake product code and allow to nearly expire	Prior to expiration system indicates to the user product is nearing end of shelf-life.				

1.4 Integration with BeagleBoard

Preliminary testing will focus on the BeagleBoard itself and its ability to interact with the desired peripherals. The system will require an LCD screen, a USB barcode scanner, a network connection,

Table 2: Shopping List Prediction Test Cases

Test Writer:Steven Strapp						
Test Case Name:	Shopping List Prediction System Unit Tests				Test ID #:	Pred-02
Description:	Verify that shopping list recommendations are helpful, intuitive and reflect previous purchasing habits.				Type:	White Box
Tester Information						
Name of Tester:					Date:	
Hardware Ver:					Time:	
Setup:	System should be placed in time accelerated mode to facilitate quick testing. System should have no previous shopping history.					
Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Enter fake products indicative of uni-modal shopping habit	System should recommend purchase again on this mode.				
2	Add outlier shopping habits	System should continue to recommend purchase again after mode value.				
3	Enter various items with different buying habits	System should recommend products with highest probabilities.				
4	Begin with uni-modal habit only and add significant variation	System should attempt to track variation in habits.				
5	Enter fake products indicative of bi-modal shopping habits	System should recommend purchase again on each mode.				

and a keypad. Basic functionality of these components will be tested thoroughly during development, as well as during final system testing.

The SQL database used to store all data for the system will be tested once the core of the user application has been coded. Test scripts will be written to populate the databases with fake data in order to ensure that the database is configured as desired, and to verify that the user application is properly communicating with the database alongside the web interface.

Table 3: Finite State Machine

Test Writer: Sue L. Engineer						
Test Case Name:		Finite State Machine Path Test # 1			Test ID #:	FSM-Path-01
Description:		Simulate insertion of money with a mix of nickles and dimes. Verifies FSM, outputs candy in response to a total deposit of $ 0.30.			Type:	White Box
Tester Information						
Name of Tester:					Date:	
Hardware Ver:					Time:	
Setup:		Make sure that the system was reset sometime prior and is in state $0.00				
Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Strobe Nickel	State should go to $0.05				
2	Strobe Dime	State should go to $0.15				
3	Wait	State should remain $0.15				