Smart Refrigerator Proposal
Test Plan

Steven Strapp, Ben Reeves, Dustin Stroup
February 15, 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 interface, 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. This subsection will focus on top-level testing of the base station user interface as a module, with tests particularly directed at the engineering specifications and user requirements. Tests directly motivated by the requirements specification and engineering specifications are listed below and a test procedure is tabulated in Table 1.

- The user interface is required to be easy to use and intuitive; in order to verify this someone not involved in the project should contribute to top-level testing of this sub-system. This also can be tested quantitatively, tests should be performed to ensure the most used items are presented on the default tab and are the most frequently used controls are the most accessible.
- The user interface will provide access to the current inventory, which will be stored using an SQL database. The principle test effort at this step will be verifying integration of the display with the database, not verifying the storage of items themselves.
- The user interface will provide both read and write access to shopping lists, also stored using an SQL database. Testing of this feature will again focus on the ability of the interface to query and modify database entries, not on the database implementation itself.
- The user interface must provide a method to update expiration estimates. Testing of this subsystem will not verify that the update is reasonable or correct but simply verify that this user interface action triggers an update from the expiration prediction subsystem.
- To achieve the principle goal of the system, the user interface must provide a notification of items about to expire. Testing of this subsystem will not verify that the expiration estimate

is reasonable or correct, but simply that if triggered by the expiration prediction subsystem the user interface will display an indication.

Table 1: Base Station User Interface Test Cases

Test Writer:Steven Strapp						
Test Case Name:		Base Station Interface Top-Level Unit Tests			Test ID #:	Base-GUI-01
Description:		Verify that the base station user interface meets the requirement and engineering specifications. Some, such as usability will be evaluated qualitatively and are difficult to outline in this way.			Type:	White Box
Tester Information						
Name of Tester:					Date:	
Hardware Ver:					Time:	
Setup:		User interface subsystem should be entirely integrated with prediction subsystems and SQL databases. System should begin without shopping lists or inventory.				
Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Enter fake product code	Switch to inventory tab, entered product should be shown. Inventory should be otherwise empty.				
2	Wait for fake product to nearly expire	Interface should display a notification indicating expiring item.				
3	Use interface to indicate product has not yet expired	Verify that prediction sub-system is triggered to update its estimate.				
4	Create fake shopping list	Verify that list becomes accessible through base station and Android interface				
5	Modify items on fake shopping list	Verify that changes are retained and visible through base station or Android interface				

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.

Table 2: Mobile App Tests

Test Writer:Ben Reeves						
Test Case Name:		Downloading large database updates over an intermittent network connection			Test ID #:	MobApp-01
Description:		Ensure that the database is correctly downloaded even if the device's network connection is interrupted. This could be due to loss of service, a disabled network adapter, or the device powering down.			Type:	White Box
Tester Information						
Name of Tester:					Date:	
Hardware Ver:					Time:	
Setup:		System should have a fresh install of the application and no previous copies of the database downloaded.				
Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Initiate download update of the database	System should connect to the server and begin downloading.				
2	Sever device's network connection	System should pause the download upon sensing the interrupted connection.				
3	Reconnect device to the network	System should resume download of the database				
4	Allow update to complete	System should download the remaining portion of the database				

Table 3: UI Usability Test

Test Writer:Ben Reeves						
Test Case Name:		UI Usability Test			Test ID #:	UI-01
Description:		Ensure that the both the web and mobile versions of the User Interface are accessible and intuitive.			Type:	White Box
Tester Information						
Name of Tester:					Date:	
Hardware Ver:					Time:	
Setup:		System should be representative of one which is in active use; that is, its database should contain both shopping lists and grocery items associated with them.				
Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	System is given to a user unfamiliar with its operation and submitted to stress testing	User should experience little difficulty navigating the application and experience no bugs, freezes, or crashes.				

Table 4: UI Interoperability Test

Test Writer:Ben Reeves						
Test Case Name:	UI Interoperability Test				Test ID #:	UI-02
Description:	Ensure that the both the web and mobile versions of the User Interface are fully compatible with popular browsers.				Type:	White Box
Tester Information						
Name of Tester:					Date:	
Hardware Ver:					Time:	
Setup:	System should be representative of one which is in active use; that is, its database should contain both shopping lists and grocery items associated with them.					
Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Interface is accessed via Mozilla Firefox and subjected to stress testing	Interface is displayed properly, no artifacts or misplaced elements apparent.				
2	Interface is accessed via Google Chrome and subjected to stress testing	Interface is displayed properly, no artifacts or misplaced elements apparent.				
3	Interface is accessed via Microsoft Internet Explorer and subjected to stress testing	Interface is displayed properly, no artifacts or misplaced elements apparent.				
4	Interface is accessed via Android 2.1 and subjected to stress testing	Interface is displayed properly, no artifacts or misplaced elements apparent.				
5	Interface is accessed via Android 4.0 and subjected to stress testing	Interface is displayed properly, no artifacts or misplaced elements apparent.				

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 5 and 6, below.

Table 5: 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.				

Table 6: 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.				

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, a keypad, and temperature/humidity sensor. 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.

It is difficult to outline exactly what testing will be required for the processing platform, since it is unclear what compatibility issues will arise that would not be presented by a conventional platform, where ideally the system would be entirely "plug and play". However, listed below is a baseline sequence of tests.

- Verify that the BeagleBoard, with power adapter, can power all peripheral devices reliably. No sporadic failures occur, this will be performed as an endurance test.
- Verify that MAC address of Ethernet interface can be statically assigned and the BeagleBoard can be pinged reliably; this will be performed as an endurance test, cycling power or disconnecting the board multiple times.
- Verify that the BeagleBoard can reliably interface with the USB scanner and USB keypad, these tests should be performed by writing to a text editor or another program external to the user interface to isolate failures.
- Verify that the BeagleBoard's consistently receives accurate temperature and humidity measurements from the sensor, via the general purpose input/output pins. The measurements should be verified with an external sensor.
- Verify that the touchscreen display accurately records users clicks and controls the pointer; tested outside of the user interface to isolate failures.
- Verify that touchscreen accurately displays the graphical user interface without artifacts or distortion consistently, and ensure all controls on the display are accessible.