

Greenbin

By

Jessica Chao



Senior Project

Computer Engineering Department

California Polytechnic State University

San Luis Obispo

June 2017

Table Of Contents

<i>Section</i>	<i>Page</i>
Abstract	3
Acknowledgements	4
1. Introduction	5
2. Customer Needs, Requirements, and Specifications	10
3. Functional Decomposition	13
4. Project Planning	19
5. Project Design	20
6. Project Testing	26
7. Conclusion and Future Work	29
References	30
<i>Appendix</i>	
A. Senior Project Analysis	32
B. Hardware Decomposition	35
C. Software Flowchart	36
D. Source Code	37
a. Gui1.java	37
b. CheckoutWindow.java	41
c. DatabaseFunctions.java	46
d. HistoryWindow.java	51
e. Parse.java	63
f. ReturnWindow.java	68
g. SearchWindow.java	71
h. SystemFunctions.java	79
i. mySQL Database Create Table Statements	82

List of Tables and Figures

Tables

2.1 Greenbin Requirements and Specifications	11
2.2 Greenbin Deliverables	12
3.1 Greenbin Level 0 Functionality	13
6.1 Testing Specifications and Acceptance Criteria	27
6.2 Verification Test Results	28

Figures

3.1 Level 0 Functional Decomposition	13
3.2 Level 1 Functional Decomposition	14
3.3 Level 2 Functional Decomposition	15
3.4 Software Flowchart	16
3.5 Hardware Decomposition	17
3.6 Database Relational Diagram	18
4.1 Plan for Project Implementation Winter 2017	19
4.2 Plan for Project Implementation Spring 2017	19
5.1 Greenbin System	20
5.2 Raspberry Pi2, HAT board, and Li-ion Battery	21
5.3 ThingMagic M6E Board Underneath Divider	22
5.4 Host Computer IP Window	23
5.5 Greenbin User Interface	23
5.6 Container Checkout Form	24
5.7 Container Checkout Search	24
5.8 Query Search REsults	25

Abstract

Greenbin is an iteration of a larger project to implement a zero-waste container tracking system for use in Cal Poly's dining facilities. The system utilizes a database system as well as passive RFID technologies to track the checking-in and checking-out of plastic reusable food containers. These plastic food containers can be checked out by campus dining patrons, and returned autonomously to collection bins that contain these scanners, allowing the containers to be recollected, re-accounted for, and reused.

Acknowledgements

This project could not have happened without the support of my advisors and contributors. I would like to extend my thanks towards those who have helped me along the way. First, I would like to thank Dr. Tali Freed for allowing me an opportunity to work with RFID technology and applying it to my major. You have guided me in right direction for this project and provided so many resources for me to use. I would also like to thank Dr. Bridget Benson for being my senior project advisor, and keeping me on task. I would also like to thank my old senior capstone team, Zachary Ho and Steven Johnson, for working with me on this project in the beginning. I could not have done it without all of you.

Chapter I: Introduction

Currently, Cal Poly's on-campus dining facilities provide disposable paper and styrofoam products for food-service and take-home food containers. These containers are not reusable, and make up a percentage of Cal Poly's consumer waste stream which ends up in landfill. This project is a part of Cal Poly's 2013 CSU policy to become a zero waste campus.

A majority of the Cal Poly Dining system uses unsustainable disposable containers. Cal Poly's current container system consists of single-use containers. These containers are often made out of styrofoam or paper, with around 2,000 of these containers being used and throw away every single day. These containers represent a large contribution to the consumer waste stream, the majority of which ends up in landfill. The idea of implementing a reusable container system started with the announcement of the Cal Poly Zero Waste Program. In early 2015, Campus Facilities, University Housing, ASI, Campus Dining, the Green Campus Program, and the Zero Waste Club formed the Zero Waste Collaborative. This collaborative was formed to try and divert 80% of the landfill waste by 2020. Part of the solution is to implement the RFID reusable container system. Cal Poly needs a more sustainable food container system for its dining facilities if the university hopes to reach their goal of near-zero landfill waste by 2020.

The goal of this project was to create a MySQL database which utilizes RFID technology to keep inventory of the plastic containers for a reusable container system. In addition, the project goal consisted of networking the system to this database to update the inventory in real-time when containers are returned. The target users for this project are Cal Poly students who eat at on-campus dining facilities. This design of the system utilizes a Raspberry Pi with a ThingMagic M6E embedded reader board that reads in the tagged containers. As a result of six months of development, the system has the functions to check in and check out container. The system can send and query information to and from the database wirelessly via a cellular network. Multiple tags can be read at the same time.

Background Information

Radio Frequency Identification

RFID Tags

Radio-Frequency Identification (RFID) is the use of radio waves to read and capture information stored on a tag attached to an object. A tag can be read from up to several feet away and does not need to be within direct line-of-sight of the reader to be tracked. [16] Each tag consists of a GTIN and a Serial Number.

RFID Systems

RFID systems consist of a tag or label and a reader. RFID tags or labels are embedded with a transmitter and a receiver. The RFID component on the tags have two parts: a microchip that stores and processes information, and an antenna to receive and transmit a signal. [16]

Proprietary Software

The implemented system consists of proprietary reader software made by a developer by the name of Kyle Dodson. This system allows the reader to output to a .ie3 file that consists of the read tag data. This software was used to create the system's tag reading software via the press of a button.

Literature Review - Existing Implementations

University of Vermont Ecoware

System using cow tags to check out containers. Containers must be hand returned to the restaurant, as the system does not include a collection bin system. To participate in the program, students purchase their first container for 7.50. When students return a dirty container, they are given a cow tag to exchange for a new clean container. [1]

Eco2go System

This system was developed by the University of Texas. Students buy tokens that are exchanged for containers. Tokens are returned upon container return, so students always have either a container or token in their possession. Students initially join by signing up with \$5 at an "Eco2Go" station. For initiative, participators also get 5% discounts on their meals every time they use an Eco2Go container. [2]

Food Zoo Containers

University of Montana system where a container can be exchanged for a clean one. The Food Zoo system has an initial fee of \$5, which gives you a use of a container. Once used, dirty containers can be dropped off next to cash registers. Upon drop off, students may ask for a clean container. As long as there the student has possession of a clean container, they are allowed a clean container for free each time. One container must always be checked out to stay in the program. [3]

Ozzi Container System

Private purchased system. This system uses a barcode to scan in containers. Students receive a token upon return of a container. Ozzi dispenses tokens every time a container

is returned, and students exchange the token at the food service line for a clean container. [4]

Boston University To-Go Reusable Containers and Mugs

Allows a \$0.25 discount of the meal or drink every time you use a reusable container or mug. In addition, using the containers allows access to the “Green Line”, which is an express lane to pay for your meal, in comparison to waiting in line. The containers are purchased anywhere on campus for \$4. Containers are washed upon return and can be exchanged for a new one by a cashier. [8]

University of Houston To-go Program

An initial deposit of \$5 is paid for each customer’s first reusable container. After use, they can drop them off at any dining facility in exchange for a new container or key tag. The key tag can be traded in to get a new container. [9]

HUDS Reusable Container Program

Students get a token by showing their ID. From there, they can reuseable containers to pack their food. The containers are checked out at the cashier in exchange for a token while you pay for your food. After use, containers are returned for another token. [10]

Chews to Reuse

Students pay for their initial reusable container with \$5 or 5 meal points. After use, they can return their rinsed containers in exchange for a replacement. [11]

RecycleMania

RecycleMania is a friendly competition and benchmarking tool for college and university recycling programs to promote waste reduction activities to their campus communities. Over an 8-week period each spring, colleges across the United States and Canada report the amount of recycling and trash collected each week and are in turn ranked in various categories based on who recycles the most on a per capita basis, as well as which schools have the best recycling rate as a percentage of total waste and which schools generate the least amount of combined trash and recycling. With each week’s updated ranking, participating schools follow their performance against other colleges and use the results to rally their campus to reduce and recycle more. [12]

Literature Review - Relevant Studies

Modeling the Location of the Return Bins for a Reusable Container Program at Cal Poly

Cal Poly student project which studied the optimal location selection for the container collection bins after the reusable container system design is finalized. [5]

Zero Waste Campus Dining

Cal Poly students comparatively tested two methods of tagging reusable containers for tracking: a barcode system and RFID tags. The RFID tags were found to be far superior in readability consistency, and were tested before and after the container washing process. A ten-year cost analysis calculated savings of approximately \$50,000 if a reusable container system were adopted to replace the current disposable container system. [6]

Reason-to-Reuse A Sustainable To-go Food Storage Container System for Restaurants

Study shows that if there is 100% participation among restaurants within San Luis Obispo to use reusable containers, over 100,000 disposable food containers would be eliminated from entering the environment annually. [7]

Economic and Environmental Assessment of Reusable Plastic Containers

Studies suggest that using reusable plastic containers for food packaging has a positive environmental impact but can increase the cost of packaging. The adoption of a RPC system would result in a global cost increase of about 69,300€ a year in a given sample for farmers adopting this system for their produce, translating to a cost increase of 0.058€/kg for the delivered goods. However, this is an initial cost or for replacement costs only. [13]

Beyond Disposables: Eco-Clamshell Reusable To-go Program

Eckard College in Saint Petersburg, Florida created a reusable container called the “Eco-Clamshell”, which had heat resistance of up to 180 degrees Fahrenheit, stackability, durability, a hinged lid to minimize loss of parts, and BPA free plastic. The frequency of the use of these “eco-clamshells” were averaging about 2-5 days a week. Students were willing to use these containers mainly to help the environment. In addition, a fifth of the respondents replying for reasons of usage was because of the design of the container. The greenhouse gases emitted for disposing 360 medium foam containers via landfill was about 4.61 kgs of greenhouse gases versus one Eco-clamshell only releasing 0.32 kg. [14]

Optimal Pricing and Production Decisions in Utilizing Reusable Containers

There are different costs that need to be taken accounted for when implementing reusable container systems. The difference between the unit cost of production for reusable containers and the unit cost of reusing returned containers are different from each other. In this study, it is shown that the return of containers are correlated with the demand of

them. If demand is low, returns will be low. In addition, if there are more returns, there will be less spending on new containers. [15]

Chapter II: Customer Needs, Requirements, and Specifications

Customer Needs Assessment

Greenbin’s customer base primarily focuses on Cal Poly SLO’s population. This population includes students, faculty, and guests who eat at on-campus dining facilities. Cal Poly’s Chief of Sustainability Eric Veium and Cal Poly’s Zero Waste Recycling club wanted a system that reduces the campus waste. A product that tracks reusable containers, RFID inventory, and student accounts was introduced by Dr. Tali Freed.

Requirements, and Specifications

Greenbin’s requirements and specifications were given by Dr. Tali Freed, who works closely with Eric Veium on helping lower campus output of waste. Greenbin’s goal is to reduce the contribution of paper/plastic food containers to Cal Poly’s landfill waste. This is to be achieved by building a database system that keeps inventory for RFID tagged containers, connect container collection bins to the database system via cellular network, and having an inventory system that should record identifying user information for checked out containers. Additionally, the product should have a live database system that can be easily adjusted to administrative needs. The database would be a mySQL-database that can keep track of ~7,000 reusable containers. Having software to interface the RFID information with the database system was necessary as well. The accumulated charge of checked out containers should be shown per each account in the database as well. A list of marketing requirements and engineering specifications are shown below in **Table 2.1**.

Table 2.1 Greenbin Requirements and Specifications

Spec.	Parameter	Requirement or	Tolerance	Risk	Compliance
--------------	------------------	-----------------------	------------------	-------------	-------------------

Number	Description	Target with units			
1	# of Containers	7,000 containers	Min	L	A, T, I
2	RFID Reader Speed	902MHz-928MHz	Min	L	T, I
3	Availability	Campus-wide coverage (1,321 acres)	Min	M	T, I
4	Robustness	Data backed up every 5 minutes	Min	L	T
5	Environmental	0% - 5% expected landfill waste	Max	M	A, I
6	Maintenance	Quarterly (10 week)	Min	M	A, I
7	Usability	User friendly GUI	Min	L	A, T, I
8	Connectivity	Cellular	Wired	H	T
	Marketing Requirements <ol style="list-style-type: none"> 1. User Friendly 2. Durable 3. Secure 4. Environmentally Friendly 5. Fast 				

Risk Symbols: Low (L), Medium (M), High (H)

Compliance Symbols: Analysis (A), Test (T), Similarity to Existing Designs (S), Inspection (I)

Specified Project deliverables appear in **Table 2.2**.

Table 2.2 Greenbin Project Deliverables

Delivery Date	Deliverable Description
2/14/2017	CPE461 Senior Project Presentation

3/22/2017	Alpha Prototype Demo
5/8/2017	Beta Prototype Demo
5/8/2017	Draft of Final Report
6/2/2017	Senior Project Expo
6/13/2017	Final Project Demo
6/13/2017	Final Project Report

Chapter III: Functional Decomposition

Level 0 Decomposition

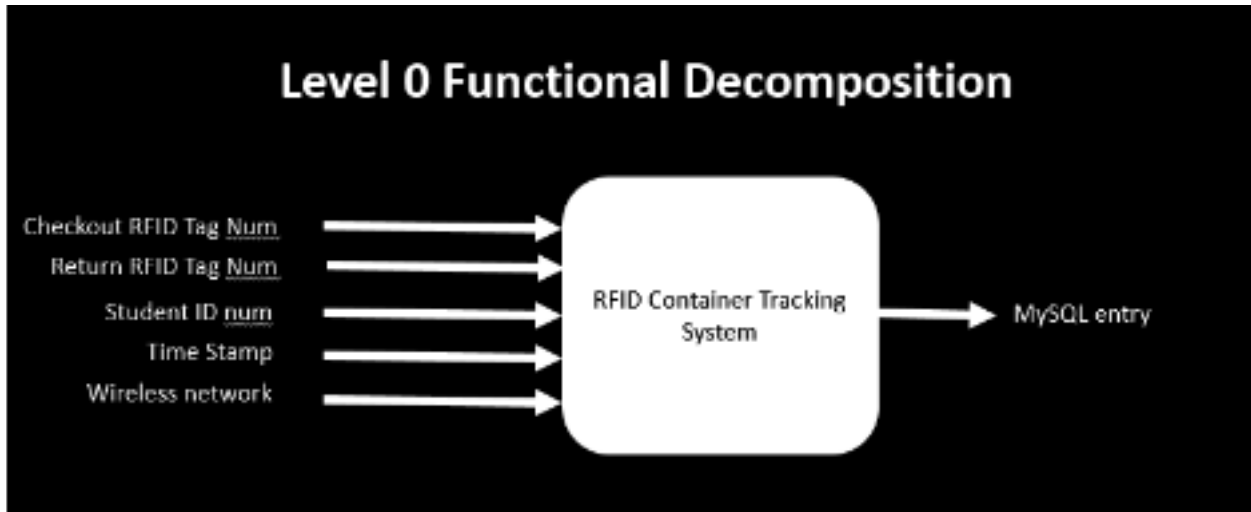


Figure 3.1 Level Zero Functional Decomposition

Level zero shows a black box of the system. On checkout the RFID container tracking system will receive a checkout RFID tag number, student ID number and timestamp. It will use the specified wireless network to then send a mySQL entry to the database. Table 3.1 explains the inputs and outputs within the diagram.

Table 3.1 Car Safe Level 0 Functionality

Module	Signal Description
Inputs	<p>Checkout RFID Tag Number - The tag number on the reusable container that is to be checked out.</p> <p>Return RFID Tag Number - The tag number on the reusable container that is to be checked back in</p> <p>Student ID Number - The Student ID number to be associated with the tagged container to be checked out</p> <p>Time Stamp - the date and time at which the container was checked out.</p> <p>Wireless network - The Cellular connection via mobile hotspot to the MySQL database</p>
Outputs	MySQL entry - the database entry to be sent to the database
Functionality	The Beta RFID Container Tracking System consists of cellularly networked Raspberry Pi 2's connected to embedded Mercury 6e ThingMagic passive RFID readers. The Pi's interpret data obtained from the readers (e.g. RFID tag info read from tagged food containers) and sends it to a remote MySQL database which holds records of checked out containers, as well as a history of returned containers. The Beta's key features are its checkout and return

	<p>functions which interact with the MySQL database. Upon checkout, the student user's information, a time/date-stamp, and the RFID tag number are inserted into the database's checkout table. Upon return, the associated row in the checkout table will be removed and copied over to the history table. The behavior model that best fits this system is a data flow diagram. The flow of the system is checkout, update database, return, update database.</p>
--	---

Level 1 Decomposition

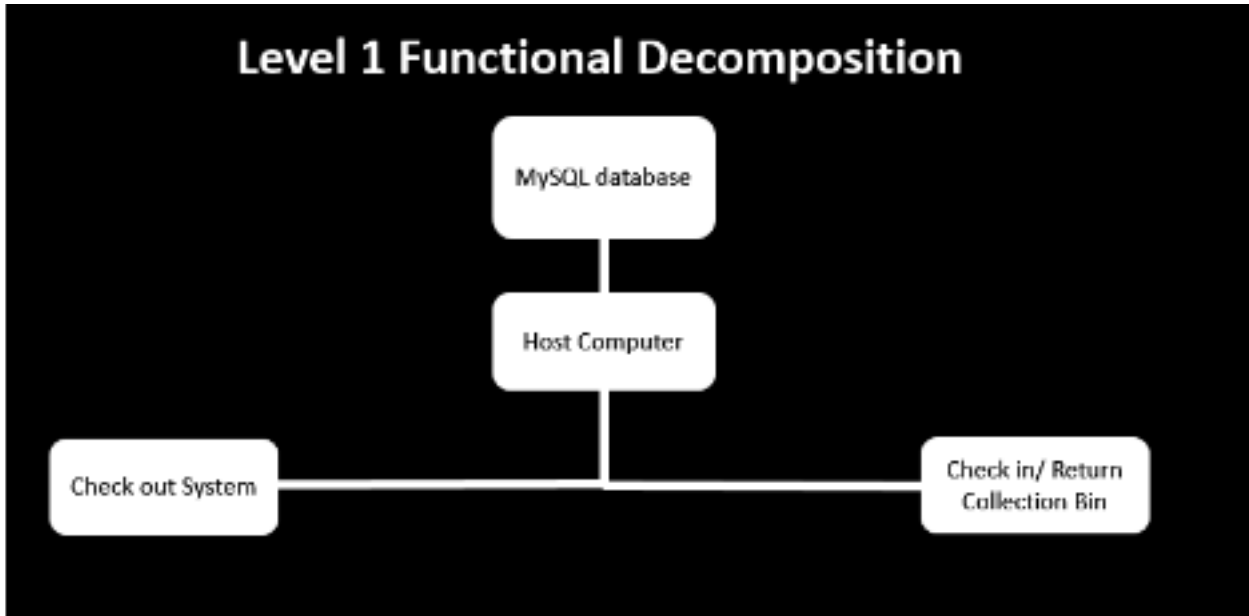


Figure 3.2 Level 1 Functional Decomposition

The level one system shows the main components of the system. It includes a host computer that will be running the collection container database 24/7. There is a check out system that will be used on campus dining facilities. Finally, there will be a return system through the a collection bin. The collection bin design will be designed by a Mechanical engineering student in the future.

Figure 3.3 Level 2 Functional Decomposition

Level two shows the entire system. Like in level one the host computer will be running the collection container database 24/7. The checkout system consists of a raspberry pi 2 and an embedded reader that identifies the RFID tag of the container checked out. The raspberry pi will then be able to send that information to the database on the host computer. The check in/return system will consist of a collection bin that will be designed by Kyle, a Cal Poly Mechanical Engineer. Inside the collection bin then group is responsible for the checkout process through the host computer database. It will use a raspberry pi 2 and embedded reader to identify the collection container returned and send the data to the database for removal. The process will be very similar to the checkout but a delete statement instead of an insert.

Level 3 Functional Decomposition

Software Flowchart

The flow of the software is shown in the flowchart below. The system initially reads in tags. The software then processes the output tag text file and the system stores it in a list. The system then waits for user input. The user can either check out a container, check in a container, or look up information on the database.

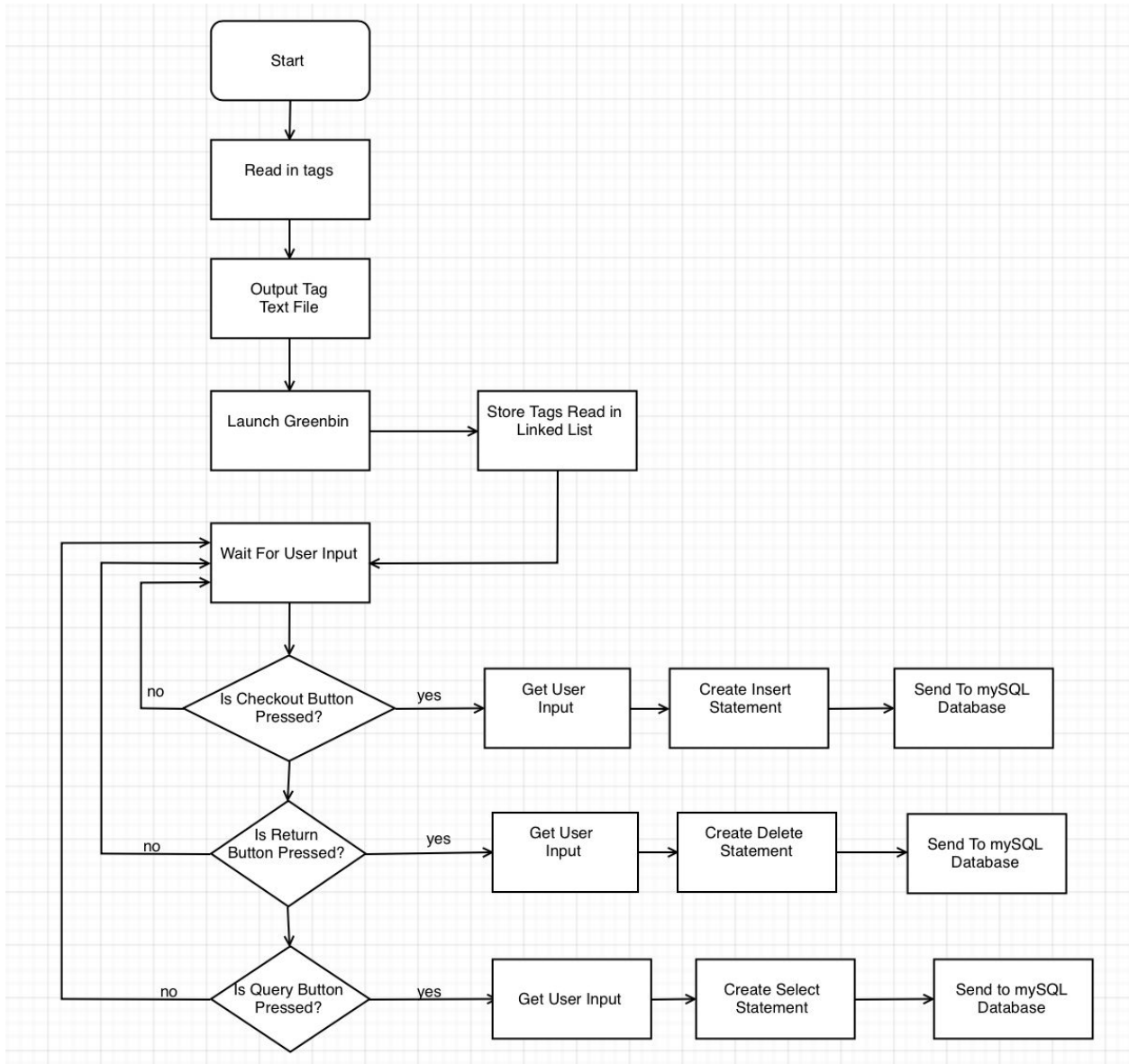


Figure 3.4 Software Flowchart

System Decomposition

The system decomposition is shown in **Figure 3.5**. The system takes in container RFID tags via a ThingMagic M6e reader. The M6E reader is connected to a Samsys Antennae that picks up tags. The raspberry pi is connected to a reader and a wifi dongle, that connects to a MySQL Database. Similar components are used in the checkout module, but the checkout module takes in student information as well.

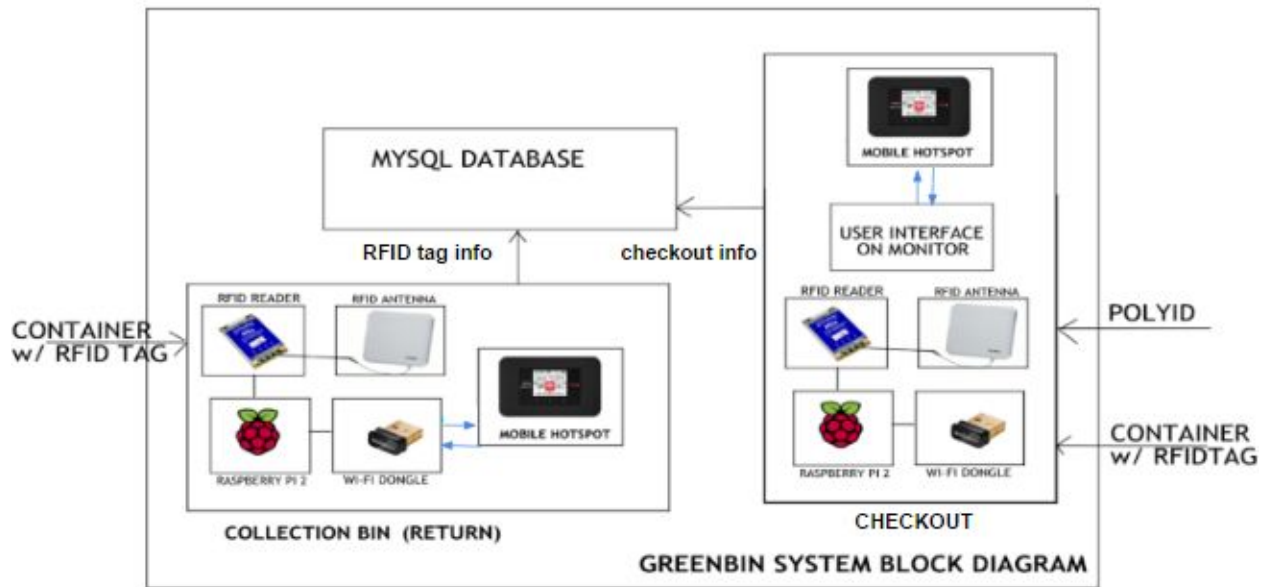


Figure 3.5 Hardware decomposition

Database Relational Diagram

The database structure of Greenbin is a key component in keeping track of checkins and checkouts. The database “greenbin” contains four tables: container, student, checkout, and history. The table “container” has a primary key of serialnumber and gtin. The table “student” has a primary key of polyid. The table “checkout” has foreign references to student and container’s poly id, gtin and serial number. The table history directly references from checkout. The database relational diagram is shown in **Figure 3.6**.

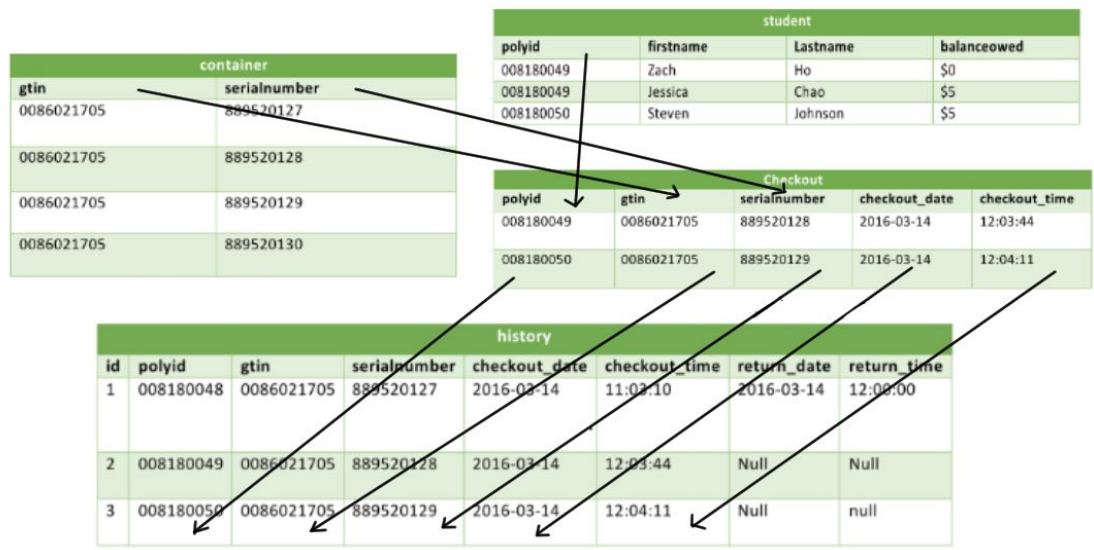


Figure 3.6 Database Relational Diagram

Chapter IV: Project Planning

The project plan for the Greenbin project shows organization and accountability of this project. The Gantt charts that outlines the work distribution for this project is shown in the following Gantt Charts below.

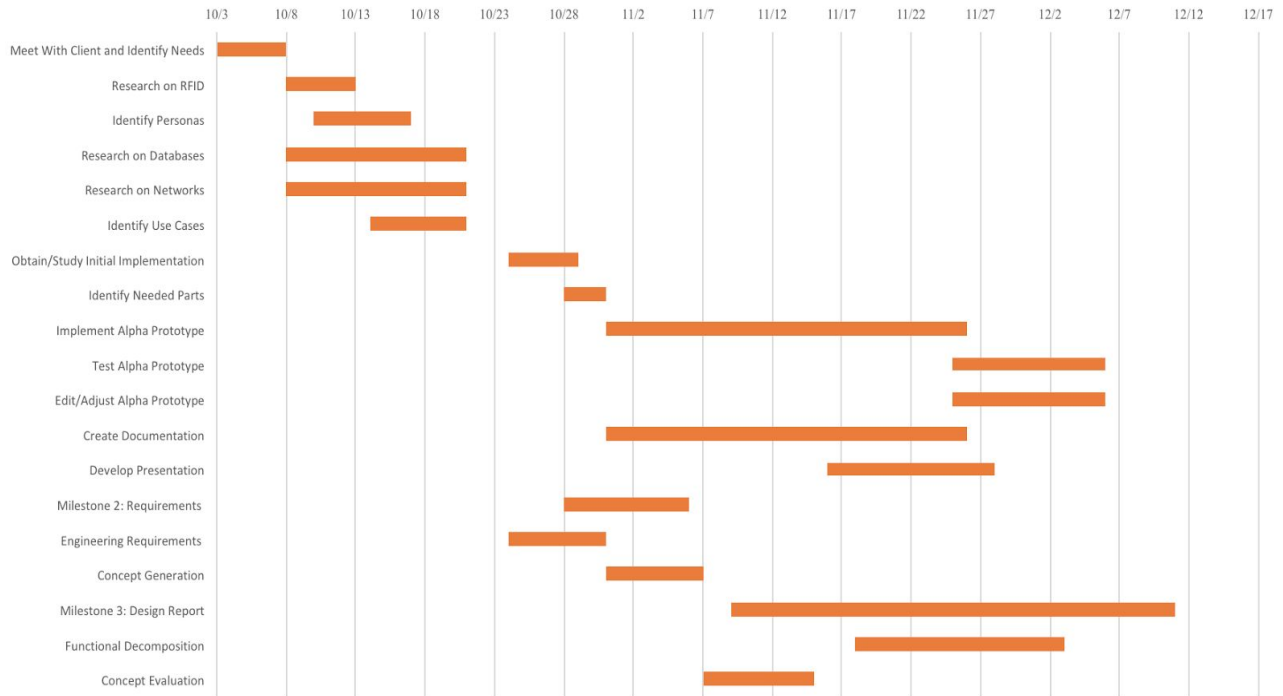


Figure 4.1 Plan for Project Implementation Winter 2017

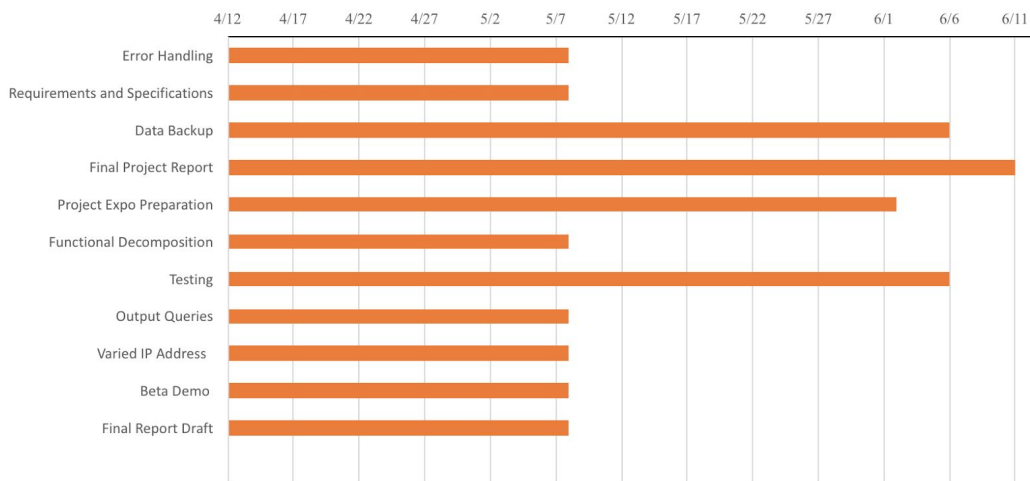


Figure 4.2 Plan for Project Implementation Spring 2017

Chapter V: Project Design

Greenbin uses hardware and software. With a Raspberry Pi 2 running the system, the program sends information about scanned RFID tags to a database. This information is transmitted via a cellular network to a database server. The tag information is gathered via a ThingMagic passive

RFID tag reader with a samsys antennae. An underlying proprietary software outputs the read tags to a textfile and software processes that data. An LED button is attached via a connector on a hat board, which notifies the user when the tags have finished reading. They whole system is powered by a Li-ion Battery. The monitor shows the user interface and a keyboard and mouse can be used to take in user input.



Figure 5.1 Greenbin System

Hardware Design
Internal Components



Figure 5.2 Raspberry Pi 2, HAT board, and Li-ion battery

Raspberry Pi 2

The Raspberry Pi is the component that holds and drives the whole system. It holds the software for the user interface and tag reading. Connected to it is a mouse, Anewish Wi-fi adaptor, and a HAT board. The Raspberry Pi 2 was chosen as the computer that the system so that components such as a keyboard, mouse and wifi capability can be interfaced to it. Due to its affordability, this computer was chosen so that the software could be replicated and reproduced into many systems to use on campus.

Power

Greenbin uses DC battery supplied by a Li-ion rechargeable battery pack. It powers the hat board as well as the raspberry pi and LEDs. A recharging battery is not optimal, but a direct supply can be used to power it. The advantage of using a recharging battery allows for portability.

HAT Board

The HAT board takes in the LED board connector and the ThingMagic M6E reader development board. It interfaces the reader to the whole system, which allows text files filled with read tag numbers to be stored into the Raspberry Pi 2.

ThingMagic M6E Development Reader Board

The ThingMagic M6E development kit was chosen as the reader so that the Mercury API can be used, since it provides a programmatic interface for development with all ThingMagic fixed and embedded reader products.

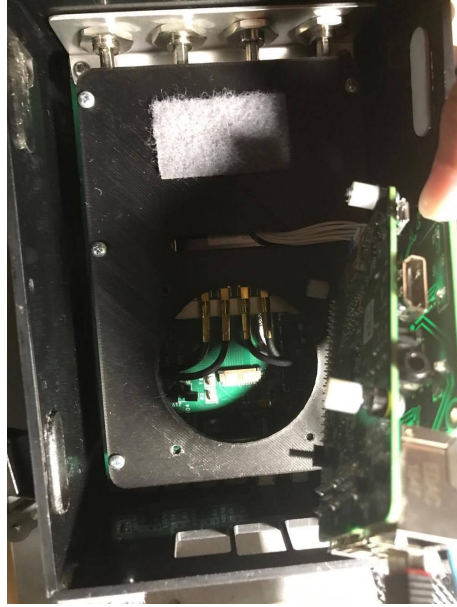


Figure 5.3 ThingMagic M6E Board Underneath Divider

Software Design

Greenbin's software design is divided into three functions: check in, checkout, and information search. The programming of the system was done in Eclipse IDE, which helped in previews of the graphical user interface in the JAVA swing library. The code to communicate to the MySQL database was written using the oracle java.sql.* library. The database was designed using Oracle's MySQL.

On power up, a window pops up where the user can input the IP of the host computer that is holding the database server. Once the IP has been set, the system can communicate to the database server, as long as the server is running, and a network connection has been established. The window screen is shown in Figure 5.4.



Figure 5.4 Host Computer IP Window

After the IP address of the host computer has been set, the software can now add, delete, checkout, return, or search for containers. The Add and Delete functions take in the tags read from the file and add or delete them from the database server. The Checkout and Return functions update the container and history tables of the database respectively. The interface for the user is shown below in Figure 5.5.

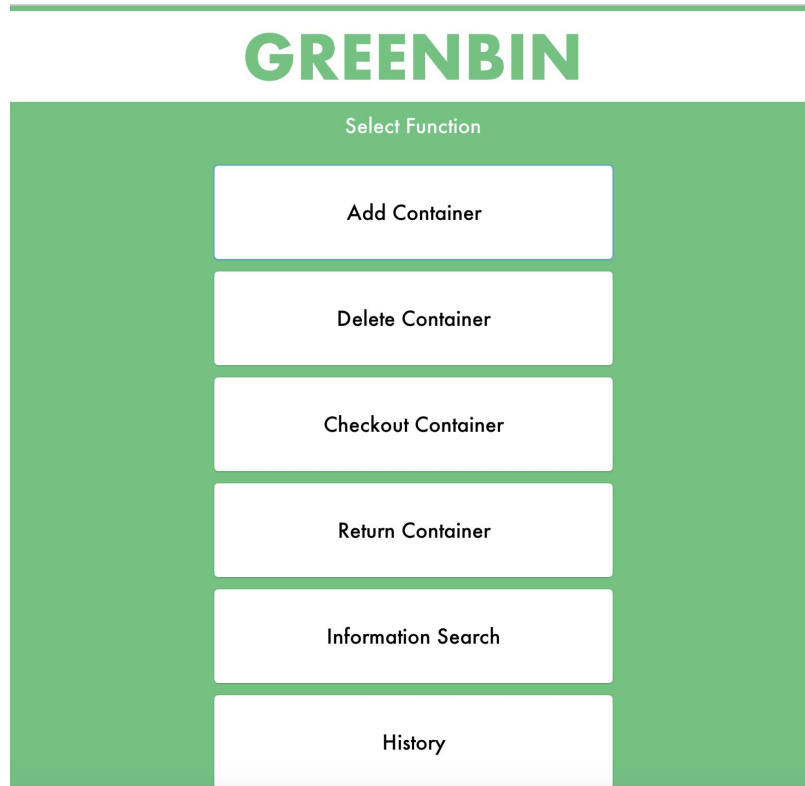


Figure 5.5 Greenbin User Interface

The container checkout form is shown below in Figure 5.6. This form takes in the customer's first name, last name, and poly ID, and constructs a mySQL insert statement that is sent to the container table. An entry is added upon success and a popup window notifies you. On failure, an error window pops up.

Checkout Window

CONTAINER CHECKOUT FORM

First Name

Last Name

PolyID #

Figure 5.6 Container Checkout Form

Finally, the container checkout search form is shown below in Figure 5.7. It takes in user input that allows for containers to be searched up in the container table. Searching by student, container, date, and time frame are options the user can use to search. Upon submitting the form, a mySQL select statement is created and if anything is found, a pop up window of the search results is shown in Figure 5.8.

search Window

CONTAINER CHECKOUT SEARCH

Search Student

PolyID #

Search Container

RFID Tag Gtin#

RFID Tag Serial#

Search By Date

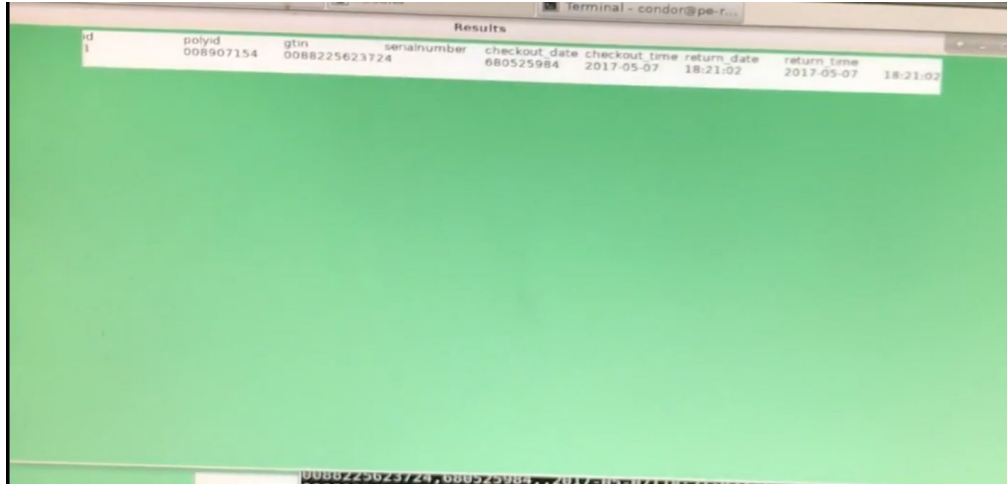
Date(YYYY-MM-DD)

Timeframe

Start(YYYY-MM-DD)

End(YYYY-MM-DD)

Figure 5.7 Container Checkout Search Form



The image shows a terminal window with a green background. At the top, there is a title bar that says "Terminal - condor@pe-r...". Below the title bar, the word "Results" is centered. A table of data is displayed, with columns for id, polyid, gtn, serialnumber, checkout_date, checkout_time, return_date, and return_time. The table contains one row of data.

id	polyid	gtn	serialnumber	checkout_date	checkout_time	return_date	return_time
1	008907154	0088225623724		680525984	2017-05-07 18:21:02	2017-05-07	18:21:02

Figure 5.8 Query Search Results

The program flowchart is in appendix C. The complete code for the program is found in appendix D.

Chapter VI: Testing

Testing

The testing of Greenbin involves on ensuring data integrity, network reliability, and user friendliness. To isolate any problems of data transmission, direct comparisons between the user interface's output and the mySQL output of select statements were made. When data integrity is breached, code was debugged to keep the data accurate and reflective of the database.

The transition of the system from a regular laptop to a Raspberry Pi was challenging initially. The initial software given was proprietary and there was not much access to the code. The Raspberry Pi initially ran with an alien reader. After switching to a different reader, the reader presented some problems with getting tags into a text file. It was found that the reader only read after 37 seconds of delay. This was found by incrementing the wait time until it was found that 37 seconds was needed to fully process a read command.

After the completion of the Greenbin design and build process, more tests were run to ensure data integrity, network reliability, and user friendliness.

Table 6.1 Testing Specifications and Acceptance Criteria

Item	Specification	Test Description	Acceptance Criteria
------	---------------	------------------	---------------------

No			
1	Read tags through embedded reader	Run software procedure to read RFID tags, save tag info on Raspberry Pi locally	Reader does not read unless with 37 second delay. Test was done in 1 second increments between 30 seconds and 45 seconds.
2	Delete Information on MySQL Server through Raspberry Pi	View MySQL Tables on Host Computer as information is modified through the Raspberry Pi	Correctly deletes information from server. Logs history if it is a deletion from checkout table.
3	Insert Information on MySQL Server through Raspberry Pi	View MySQL Tables on Host Computer as information is read through the Raspberry Pi	Correctly inserts information from system to server and can query from server side
4	Check database if returned container is registered	Query checkout table for returned container's tag (should return nothing) and make sure history table is updated	Outputs correct queried information onto system just as the query statement on the server.
5	Query containers by student id in checkout table	Query checkout table and verify correct student id associated with given rfid tag number	Outputs correct queried information onto system just as the query statement on the server.
6	Query checkout and checkout history information (sql tables) by date	Check output table after query	Outputs correct queried information onto system just as the query statement on the server.
7	Establish secure connection with cellular hotspot brick	Ping database host computer	Couldn't connect to Cal Poly Wifi, but had success with a phone hotspot
8	Establish error handling for incorrect inputs	Enter incorrect inputs	Handles errors of incorrect input by outputting a warning
9	Establish freedom of changing host computers.	Enter IP Address from remote location, away from host	IP address input from raspberry pi can be accessed from various computers other than original host computer
10	Keep data integrity between database and system for backups	Make sure that when a force quit happens, the most recent activity has been backed up to a file	File full of statements correctly outputs every time data is inputted.
11	The system is user friendly.	Make sure that the system is easy to use by a non technical user	User knows how to use the system with minimal directions.

Table 6.2 Verification Test Results

Item No.	SAMPLES TESTED		TIMING		TEST RESULTS			NOTES
	Qty	Type	Start date	Finish date	Result	Qty Pass	Qty Fail	
1	75	Hardware	3/2/17	3/9/2017	Success	40	35	Reader does not read unless with 37 second delay. Test was done in 1 second increments between 30 seconds and 45 seconds.
2	50+	Software	2/23/17	3/9/2017	Success	50+	0	Correctly deletes information from server. Logs history if it is a deletion from checkout table.
3	50+	Software	2/23/17	3/9/2017	Success	50+	0	Correctly inserts information from system to server and can query from server side
4	50+	Software	2/27/2017	3/9/2017	Success	50+	0	Outputs correct queried information onto system just as the query statement on the server.
5	15	Software	2/27/2017	3/9/2017	Success	15	0	Outputs correct queried information onto system just as the query statement on the server.
6	15	Software	2/27/2017	3/9/2017	Success	15	0	Outputs correct queried information onto system just as the query statement on the server.
7	10	Hardware	2/23/2017	2/23/2017	Success *SEE NOTES	-	-	Couldn't connect to Cal Poly Wifi, but had success with a phone hotspot. Approximately 5 hours of running time connected and executed
8	10	Software	4/23/17	5/5/17	Success	20	0	Handles errors of incorrect input output
9	10	Network	4/25/17	5/5/17	Success	10	0	IP address input from raspberry pi can be accessed from various computers other than original host computer
10	10	Data Integrity	4/28/17	5/28/17	Success	10	0	All files correctly output the data that needs to be there when a crash happens.

11	5	User Friendly	5/1/17	6/6/17	Success	5	0	All users seem to have no problem with the UI. there was no confusion.
----	---	---------------	--------	--------	---------	---	---	--

Chapter VII: Conclusion and Future Work

As of now, Greenbin works well as a first iteration that has the basic system functionality of what is meant to be eventually implemented into Campus Dining. Greenbin allows the tracking of RFID tagged reusable containers as well as actively backs up data in case of network failures and information searching. However, improvements must be made to consider it as usable by Cal Poly's dining system. While the device is fully functional, it is necessary to polish the system as well as test with the actual PolyCard database instead of mock data.

To achieve device practicality, the device must work with a collection bin that allows for no stray containers to be scanned. A smart collection bin that locks the return function when network failures happen is necessary to work well with this system. Additionally, a faster reading software must be implemented in order to ensure that long lines during meal times are not delayed longer than needed from RFID tags needing to be read.

References

- [1] University of Vermont's Eco-Ware Reuseable Takeout Container Program, *fesmag.com*, 2012. [Online]. [Accessed: 30-Oct-2016].
- [2] Eco2Go, *utexas.edu*, 2016. [Online]. [Accessed: 30-Oct-2016].
- [3] Food Zoo Reusable To-Go Container Program, *umt.edu*, 2016. [Online]. [Accessed: 30-Oct-2016].
- [4] OZZI, *agreenozzi.com*, 2016. [Online]. [Accessed: 30-Oct-2016].
- [5] Terry and Williams. Green Bin Final Presentation. [Powerpoint slides].
- [6] Caudillo, Dahel, Ghazalian. "Zero Waste Campus Dining". *Digitalcommons.calpoly.edu*. Cal Poly San Luis Obispo, June 2016. [Accessed 30 Oct. 2016].
- [7] LaBuda, Ryan Christopher. "REASON-TO-REUSE: A SUSTAINABLE TO-GO FOOD STORAGE CONTAINER SYSTEM FOR RESTAURANTS." *Digitalcommons.calpoly.edu*. Cal Poly San Luis Obispo, 4 June 2013. [Accessed: 30 Oct. 2016].
- [8] Carillo, Maria. "Improved Reusable To-go Program Helps Reduce University's Carbon Footprint". *Uh.edu*. University of Houston, August 2014. [Accessed 1 Dec 2016]. Available: <http://www.uh.edu/af-auxiliary-services/news/articles/2014/august/08202014-improved-reusable-to-go.php>
- [9] Reusable To- go program, *uh.edu*, 2016. [Online]. Available: <http://www.uh.edu/af-auxiliary-services/news/articles/2014/august/08202014-improved-reusable-to-go.php>
- [10] Reusable Container Program, *green.harvard.edu*, 2016. [Online]. Available: <https://green.harvard.edu/topics/waste/reuse>
- [11] Chews to Reuse, *caldining.berkeley.edu*, 2016. [Online]. Available: <http://caldining.berkeley.edu/sustainability/chews-reuse>
- [12] RecycleMania, *recyclemaniacs.org*, 2016. [Online] Available: <http://recyclemaniacs.org/about>
- [13] Accorsi, Cascini, Cholette, Manzini, Mora. "Economic and Environmental Assessment of Reusable Plastic Containers: A food catering supply chain case study". *Science Direct*. June

2014. [Online]. Available:

<http://www.sciencedirect.com.ezproxy.lib.calpoly.edu/science/article/pii/S0925527313005732>

[14] Copeland, Ormsby. “ Beyond Disposables: Eco-Clamshell Reusable To-Go Program”.

Eckerd College. December, 2009. [Online]. Available:

<https://erefdn.org/beyond-disposables-eco-clamshell-reusable-to-go-program/>

[15] Dang, Shuo. “Optimal pricing and production decisions in utilizing reusable containers.”

Science Direct. August, 2011. [Online]. Available:

<http://www.sciencedirect.com/science/article/pii/S0925527311003288>

[16]"What Is RFID?" *EPC-RFID*. N.p., n.d. Web. 19 Mar. 2017. [Online]. Available:

<http://www.epc-rfid.info/rfid>

Appendix A: Senior Project Analysis

1. Summary of Functional Requirements

Greenbin is an iteration of a larger project to implement a zero-waste container tracking system for use in Cal Poly's dining facilities. The system utilizes a database system as well as passive RFID technologies to track the checking-in and checking-out of plastic reusable food containers. These plastic food containers can be checked out by campus dining patrons, and returned autonomously to collection bins that contain these scanners, allowing the containers to be recollected, re-accounted for, and reused.

2. Primary Constraints

The constraints for the Greenbin's development includes economic and design constraints. It is necessary for large amounts of funding to go into this project in order to fully implement it into Cal Poly. Having a faster reader is expensive as well as working with multiple systems. The project is intended to have approximately ten systems running at the same time, but the expense was too high for a first iteration. In terms of development, it was hard to work with proprietary software from past students that worked on this project. The source code of the software was secure and unaccessible. The duration of the senior project course as well as materials for replicating the system was a constraint.

3. Economic

This project is funded by Dr. Tali Freed of the PolyGAIT lab at Cal Poly San Luis Obispo. The original estimated cost is approximately \$53,000. This estimate includes ten collection bins, ten standard passive readers with raspberry pi2s, one host computer, and approximately seven thousand RFID tagged containers. Additionally, wages for engineers that drive the system is approximated to be \$15,600 yearly and \$500 for maintenance.

The projected project savings is \$200,000 dollars for a CSU campus within a ten year period of this system being adopted[5]. The adoption of this system onto Cal Poly's campus as compared to the current industry return bin system is not as cost efficient.

The economic difficulties of adopting this system includes the large upfront costs, information security, and the uncertainty of having 100% participation from all vendors and students. Additionally, students must pay a \$5 deposit to initially gain a container to use.

For this current iteration, the list of components costs \$789. The bill of materials is listed below, which has been purchased by Dr. Tali Freed.

Raspberry Pi 2 - \$35.00
Micro SD Card 32gb - \$25.00
Mouse - \$15.00
Keyboard - \$15.00
ThingMagic M6E Development Kit - \$699.00
Total: \$789

Additional equipment include a host computer, and a monitor to output the user interface.

The estimated development time at the start of this project was 100 hours over the course of two quarters. The actual development time was approximately 80 hours.

4. Environmental

The environmental impact of using this system would hopefully lead to near zero consumer waste going to landfill. By adopting the collection bin system with one hundred percent participation by all students and dining facilities on campus, it would eliminate 1,000 disposable containers going to landfill daily.

5. Manufacturability

There are no issues or challenges associating with manufacturing this product, as all components of the system are on the market and can be replaced with different models. As long as the SD card can be replicated, there are no issues.

6. Sustainability

There is the issue of security concerns in maintaining the completed device. As the system should access the PolyCard database, it is important to ensure that a secure system is implemented. In the case of a blackout or network failure, devices must lock so that users will not return containers and update the system as data integrity is threatened. The project impacts the sustainable use of resources positively due to its use of reusable containers instead of disposable containers. Disposable containers would go to landfill, and have a bigger carbon footprint than the reusable containers in a ten year span[5].

Upgrades that would improve the design of the project would include the implementation of a smart bin that locks the opening of returning containers, as well as using a more portable version of the system that scans in containers. Challenges of upgrading the design may include development time of creating the smart bin as well as cost constraints of finding a more portable way to scan tags.

7. Ethical

The Greenbin project aims to follow ethical engineering practices by adhering through IEEE standards throughout the development and design of this system. Additionally, Greenbin aims to ethically enforce the use of the system through secure means. There is risk of security breaches in the current iteration, so the system is currently unusable with storing card information and student information. The developers worked to improve the current dining system by developing a way to have a positive impact on the environment.

8. Health and Safety

There are concerns of safety when revolving around the food containers used in this system. It is important to ensure that RFID tags are embedded safely into the container, without risk of going into food. The tags must remain embedded so that washing them is safe as well as eating out of containers.

10. Social and Political

Global warming is a huge concern worldwide. In order to combat this major concern, decreasing the carbon footprint is one solution. This system aims to decrease the carbon footprint of the university by first working to improve the school's dining system waste management. Greenbin helps decrease the landfill waste coming from the dining system as well as decrease the carbon footprint of the university.

11. Development

Throughout the development of this project, I learned how to utilize RFID technology, and how the concept of radio frequency works. I also acquired the knowledge of how to manage and create database servers and send information to them via JAVA. Additionally, I learned a lot about how send information via cellular data networks to remote servers. This project helped expand my knowledge in software design and communication between different systems.

Appendix B: Hardware Decomposition

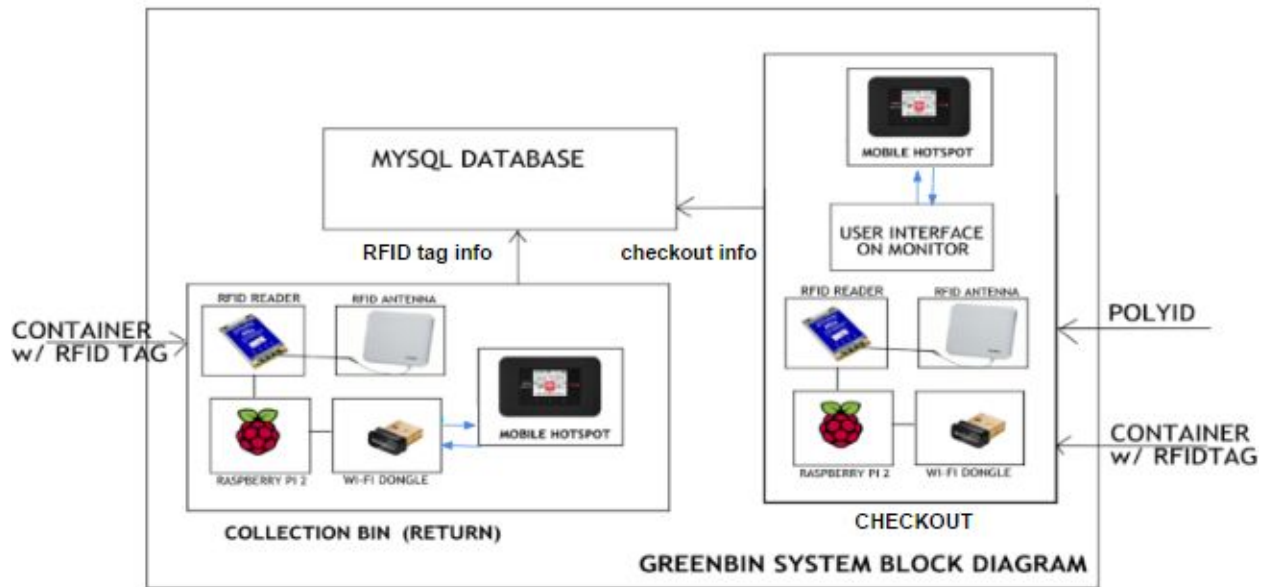


Figure B.1 Hardware decomposition

Appendix C: Software Flowchart

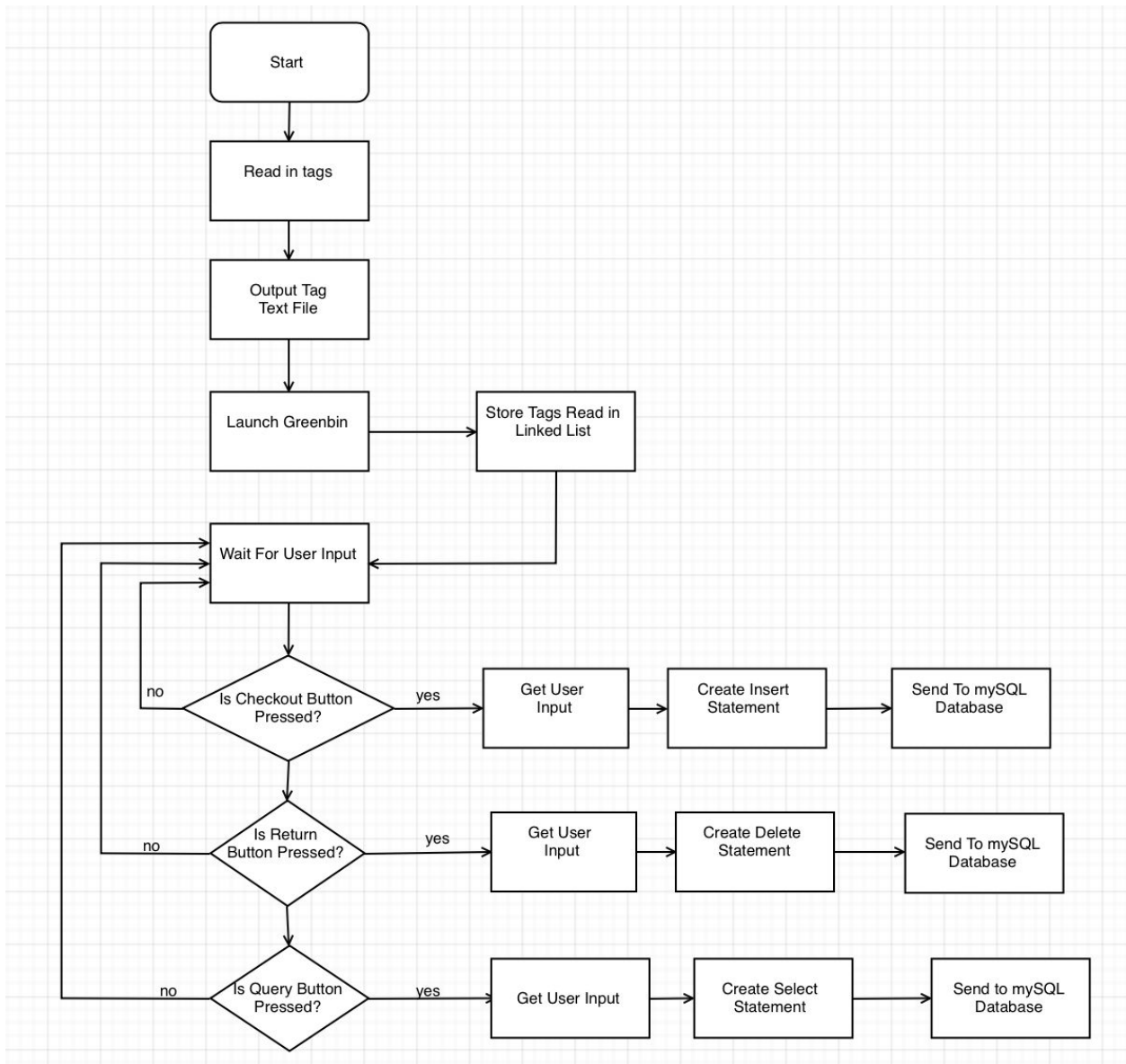


Figure C.1 System Software Flowchart

Appendix D: Software Code

Graphical User Interface (Gui1.java)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
//import java.sql.*;

/**
 * This class handles the creation of the GUI.
 *
 */
public class Gui1 {

    // Constants
    public static final Color GREEN_COLOR = new Color(116, 193, 130);
    public static final Font BANNER_FONT = new Font("Futura", Font.BOLD,
60);
    public static final Font PLAIN_FONT = new Font("Futura", Font.PLAIN,
20);

    private JFrame mainFrame;

    private JLabel titleLabel;
    private JLabel selectLabel;

    private JPanel bannerPanel;
    private JPanel selectPanel;

    private JButton checkoutBtn;
    private JButton returnBtn;
    //private JButton sysInfoBtn;
    private JButton addContainerBtn;
    private JButton delContainerBtn;
    private JButton searchBtn;
    private JButton historyBtn;

    public static void main(String[] args) {
        IPWindow ipwindow = new IPWindow();
        Gui1 window = new Gui1();
        window.run();
        ipwindow.run();
    }

    private void run() {
        initFrame();
        initButtons();
    }
}
```

```

        buildFrame();
        mainFrame.setVisible(true);
    }

    private void initFrame() {
        // Init main frame
        mainFrame = new JFrame("Greenbin");
        mainFrame.setSize(800, 800);
        mainFrame.setLayout(new FlowLayout());
        mainFrame.getContentPane().setBackground(GREEN_COLOR);
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Init GREENBIN title label
        titleLabel = new JLabel("GREENBIN");
        titleLabel.setFont(BANNER_FONT);
        titleLabel.setForeground(GREEN_COLOR);

        // Init "select function" label
        selectLabel = new JLabel("Select Function");
        selectLabel.setFont(PLAIN_FONT);
        selectLabel.setForeground(Color.WHITE);
    }

    private void initButtons() {
        Dimension btnDimension = new Dimension(400, 100);

        // Create checkout button
        checkoutBtn = new JButton("Checkout Container");
        checkoutBtn.setFont(PLAIN_FONT);
        checkoutBtn.setPreferredSize(btnDimension);
        checkoutBtn.setActionCommand("Checkout");
        checkoutBtn.addActionListener(new ButtonClickListener());

        // Create return button
        returnBtn = new JButton("Return Container");
        returnBtn.setFont(PLAIN_FONT);
        returnBtn.setPreferredSize(btnDimension);
        returnBtn.setActionCommand("Return");
        returnBtn.addActionListener(new ButtonClickListener());

        // Create add container button
        addContainerBtn = new JButton("Add Container");
        addContainerBtn.setFont(PLAIN_FONT);
        addContainerBtn.setPreferredSize(btnDimension);
        addContainerBtn.setActionCommand("AddContainer");
        addContainerBtn.addActionListener(new ButtonClickListener());

        // Create delete container button
        delContainerBtn = new JButton("Delete Container");
    }

```

```

delContainerBtn.setFont(PLAIN_FONT);
delContainerBtn.setPreferredSize(btnDimension);
delContainerBtn.setActionCommand("DelContainer");
delContainerBtn.addActionListener(new ButtonClickListener());

// Create Information Search (query) button
searchBtn = new JButton("Information Search");
searchBtn.setFont(PLAIN_FONT);
searchBtn.setPreferredSize(btnDimension);
searchBtn.setActionCommand("Search");
searchBtn.addActionListener(new ButtonClickListener());

// Create History Button
historyBtn = new JButton("History");
historyBtn.setFont(PLAIN_FONT);
historyBtn.setPreferredSize(btnDimension);
historyBtn.setActionCommand("History");
historyBtn.addActionListener(new ButtonClickListener());

}

public void buildFrame() {
    // Panel dimensions
    Dimension bannerDimension = new Dimension(800, 90);
    Dimension selectDimension = new Dimension(800, 50);

    // Build banner panel
    bannerPanel = new JPanel();
    bannerPanel.setBackground(Color.WHITE);
    bannerPanel.setPreferredSize(bannerDimension);
    bannerPanel.add(titleLabel);

    // Build select function panel
    selectPanel = new JPanel();
    selectPanel.setBackground(GREEN_COLOR);
    selectPanel.setPreferredSize(selectDimension);
    selectPanel.add(selectLabel);

    // Add panels to frame
    mainFrame.add(bannerPanel);
    mainFrame.add(selectPanel);

    // Add buttons to frame
    mainFrame.add(addContainerBtn);
    mainFrame.add(delContainerBtn);
    mainFrame.add(checkoutBtn);
    mainFrame.add(returnBtn);
    mainFrame.add(searchBtn);
    mainFrame.add(historyBtn);
}

```



```

/**
 * This private class handles the actions taken when buttons are
 clicked.
 *
 */
private class ButtonClickListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();

        switch(command) {
            case "AddContainer":
                SystemFunctions sysFunc = new
SystemFunctions();

                sysFunc.addContainer();
                break;
            case "DelContainer":
                SystemFunctions sysFunc2 = new
SystemFunctions();

                sysFunc2.deleteContainer();
                break;
            case "Checkout":
                CheckoutWindow checkoutWin = new
CheckoutWindow();

                checkoutWin.run();
                break;
            case "Return":
                ReturnWindow returnWin = new ReturnWindow();
                returnWin.run();
                break;
            case "Search":
                SearchWindow searchWin = new SearchWindow();
                searchWin.run();
                break;
            case "History":
                HistoryWindow historyWin = new
HistoryWindow();

                historyWin.run();
            default:
                break;
        }
    }
}
}
}

```

Checkout Window (CheckoutWindow.java)

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JTextField;

import java.awt.Dimension;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.sql.Connection;

/**
 * This class handles the checking out of a container.
 */
public class CheckoutWindow {

    private JFrame checkoutFrame;
    private JLabel checkoutLabel;
    private JPanel checkoutPanel;
    private String fields[] = {"First Name", "Last Name", "PolyID #"};
    private JButton submitBtn;
    private JButton cancelBtn;
    private JTextField textFields[] = new JTextField[fields.length];

    public void run() {
        initFrame();
        initButtons();
        buildFrame();
        checkoutFrame.setVisible(true);
    }

    private void initFrame() {
        checkoutFrame = new JFrame("Checkout Window");
        checkoutFrame.setSize(500, 250);
        checkoutFrame.setLayout(new FlowLayout());

        checkoutFrame.getContentPane().setBackground(Guil.GREEN_COLOR);
        checkoutFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Init container checkout form banner
        checkoutLabel = new JLabel("CONTAINER CHECKOUT FORM");
        checkoutLabel.setFont(Guil.PLAIN_FONT);
        checkoutLabel.setForeground(Guil.GREEN_COLOR);
    }
}
```

```

}

private void initButtons() {
    // Dimensions
    Dimension btnDimension = new Dimension(150, 25);

    // Create submit button
    submitBtn = new JButton("Submit");
    submitBtn.setFont(Guil.PLAIN_FONT);
    submitBtn.setPreferredSize(btnDimension);
    submitBtn.setActionCommand("Submit");
    submitBtn.addActionListener(new ButtonClickListener());

    // Create cancel button
    cancelBtn = new JButton("Cancel");
    cancelBtn.setFont(Guil.PLAIN_FONT);
    cancelBtn.setPreferredSize(btnDimension);
    cancelBtn.setActionCommand("Cancel");
    cancelBtn.addActionListener(new ButtonClickListener());
}

public void buildFrame() {
    // Panel dimensions
    Dimension bannerDimension = new Dimension(500, 50);
    Dimension panelDimensions = new Dimension(500, 25);
    Dimension labelDimension = new Dimension(200, 25);
    Dimension tfDimension = new Dimension(200, 25);

    // Build container checkout banner panel
    checkoutPanel = new JPanel();
    checkoutPanel.setBackground(Color.WHITE);
    checkoutPanel.setPreferredSize(bannerDimension);
    checkoutPanel.add(checkoutLabel);
    checkoutFrame.add(checkoutPanel);

    // Build form field panels and add them to the frame
    for (int i = 0; i < fields.length; i++) {
        JLabel label = new JLabel(fields[i]);
        label.setFont(Guil.PLAIN_FONT);
        label.setForeground(Color.WHITE);
        label.setPreferredSize(labelDimension);

        JTextField tf = new JTextField();
        tf.setPreferredSize(tfDimension);
        textFields[i] = tf;

        JPanel panel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 10, 0));
        panel.setBackground(Guil.GREEN_COLOR);
        panel.setPreferredSize(panelDimensions);
    }
}

```

```

        panel.add(label);
        panel.add(tf);

        checkoutFrame.add(panel);
    }

    // Add extra panel for spacing
    JPanel panel = new JPanel();
    panel.setBackground(Guil.GREEN_COLOR);
    panel.setPreferredSize(panelDimensions);
    checkoutFrame.add(panel);

    // Add buttons to frame
    checkoutFrame.add(cancelBtn);
    checkoutFrame.add(submitBtn);
}

/**
 * This method handles the actions taken when the submit button is
pressed.
 */
private void submit() {
    DatabaseFunctions dbf = new DatabaseFunctions();
    Parse tagGetter = new Parse();
    String firstname, lastname, polyid, gtin, serialnum, date,
time;

    Connection con = null;
    int parseIndex = 0;

    // Establish connection to database
    try {
        con = dbf.getConnection();
    } catch (Exception e) {
        System.out.println(e);
        return;
    }

    // Parse M6e output file to get RFID tag info
    tagGetter.parseReaderOutput();

    // Grab student and tag info
    firstname = textFields[0].getText();
    lastname = textFields[1].getText();
    polyid = textFields[2].getText();
    gtin = tagGetter.getGtin(parseIndex);
    serialnum = tagGetter.getSerialNum(parseIndex);
    date = tagGetter.getDate(parseIndex);
    time = tagGetter.getTime(parseIndex);
}

```

```

        // Add student and checkout entry only if container has a
        recognized gtin and serial number
        if (dbf.checkExists(con, "container", "gtin", gtin) &&
dbf.checkExists(con, "container", "serialnumber", serialnum)) {
            // Check if container is already checked out
            if (!dbf.checkExists(con, "checkout", "serialnumber",
serialnum)) {
                // Check if student exists in student table, if so
                increment existing balance
                if (dbf.checkExists(con, "student", "polyid",
polyid)) {
                    String update =
dbf.updateStatement("student", "balanceowed",
                    "balanceowed+5.00", "polyid",
polyid);
                    dbf.executeStatement(con, update);
                }
                // Otherwise make a new student entry
                else {
                    dbf.setNumVals(4);
                    String student_insert =
dbf.insertStatement("student", "firstname",
                    "lastname", "polyid",
"balanceowed", firstname, lastname, polyid, "5.00");
                    dbf.executeStatement(con, student_insert);
                }

                // Add entry to checkout table
                dbf.setNumVals(5);
                String checkout_insert =
dbf.insertStatement("checkout", "polyid", "gtin",
                    "serialnumber", "checkout_date",
"checkout_time", polyid, gtin,
                    serialnum, date, time);
                dbf.executeStatement(con, checkout_insert);
                JOptionPane.showMessageDialog(null, "Successful
checkout:\npolyid: "+polyid+"\ngtin: " +gtin+
                    "\nserialnumber: "+serialnum+"\ndate:
"+date+"\ntime: "+time, "SUCCESS", JOptionPane.INFORMATION_MESSAGE);
            }
            else {
                String error1 = "ERROR: Container with given gtin &
serial number is already checked out.";
                JOptionPane.showMessageDialog(null, error1, "ERROR",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    else {

```

```

        String error2 = "ERROR: Gtin and Serial Number not found
in system.";
        JOptionPane.showMessageDialog(null, error2, "ERROR",
JOptionPane.ERROR_MESSAGE);

    }

    checkoutFrame.dispose();
}

/**
 * This private class handles the actions taken when buttons are
clicked.
 */
private class ButtonClickListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();

        switch(command) {
            case "Submit":
                submit();
                break;
            case "Cancel":
                checkoutFrame.dispose();
                break;
            default:
                break;
        }
    }
}
}

```

Database Functions (DatabaseFunctions.java)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.swing.JOptionPane;
import java.io.*;
/**
 * This class provides methods to generate and execute different types of
 * SQL statements.
 *
 */
public class DatabaseFunctions {
    // Connection constants
    private final static String dbDriver = "com.mysql.jdbc.Driver";
    // Name of the database driver package
    private final static String schemaUrlPart1 = "jdbc:mysql://";
    private final static String schemaURLPart2 =
":3306/greenbin?autoReconnect=true&useSSL=false";
// jdbc:mysql://[ip address of database host]:[port num]/[Database name]
    private final static String dbUsername = "raspberry";
    // MySQL user name
    private final static String dbPassword = "Capstone350!";
// MySQL user password
    private int numVals = 0;    // Number of values in insert statement

    /**
     * Sets the number of values to append for insert statements.
     *
     */
    public void setNumVals(int num) {
        this.numVals = num;
    }
    /**
     * Establishes connection to mysql database schema.
     *
     */
    public Connection getConnection() throws Exception {
        Connection con;
        Class.forName(dbDriver);
        try {
            con =
DriverManager.getConnection(schemaUrlPart1+IPWindow.IPADDRESS+schemaURLPar
t2, dbUsername, dbPassword);
        } catch (Exception e) {
            String error1 = "Error connecting to database.";

            JOptionPane.showMessageDialog(null, error1, "ERROR",
JOptionPane.ERROR_MESSAGE);

```

```

        System.out.println(e);
        return null;
    }

    return con;
}

/**
 * This method executes an SQL DQL (Data Query Language) statement.
 */
public ResultSet executeQuery(Connection con, String s) {
    PreparedStatement statement;
    ResultSet result;

    System.out.println(s);

    try {
        statement = con.prepareStatement(s);
        result = statement.executeQuery();
    } catch (Exception e) {
        System.out.println(e);
        result = null;
    }

    return result;
}

/**
 * This method executes an SQL DML (Data Manipulation Language)
statement.
 */
public void executeStatement(Connection con, String s) {

    PreparedStatement statement;
    try {
        PrintWriter output= new PrintWriter(new
FileOutputStream("Backup.txt", true));
        String temp = s;
        temp += " ";
        output.println(temp);
        output.close();
    } catch (FileNotFoundException ex) {

        System.out.println("Cannot write to backup\n");
    }

    System.out.println(s);

    try {

```



```

        statement = con.prepareStatement(s);
        statement.executeUpdate();
    } catch (Exception e) {
        System.out.println(e);
    }
}

/**
 * This method constructs an SQL insert statement for the specified
table and attributes
 * Usage: method parameters should be in order [tablename],
[attributes],..., [values],...
 *
 * NOTE: Make sure to call setNumVals() and specify the number of
values beofre calling insertStatement()
 */
public String insertStatement(String...strings) {
    StringBuilder insertBuilder = new StringBuilder();
    int i;

    insertBuilder.append("INSERT INTO " + strings[0] + "(");

    for (i = 1; i < numVals+1; i++) {
        insertBuilder.append(strings[i]);
        if (i < numVals) { insertBuilder.append(", "); }
    }

    insertBuilder.append(") VALUES (");

    for (i = 1+numVals; i < (numVals*2)+1; i++) {
        insertBuilder.append("'" + strings[i] + "'");
        if (i < numVals*2) {insertBuilder.append(", "); }
    }
    insertBuilder.append(")");
    return insertBuilder.toString();
}

/**
 * This method constructs an SQL update statement for the specified
attribute within the given table
 */
public String updateStatement(String table, String set, String val,
String where, String whereVal) {
    StringBuilder updateBuilder = new StringBuilder();

    updateBuilder.append("UPDATE " + table);
    updateBuilder.append(" SET " + set + "=" + val + " ");
    updateBuilder.append(" WHERE " + where + "=" + whereVal +
    "'");
}

```

```

        return updateBuilder.toString();
    }

    /**
     * This method checks if there exists an entry (tuple) in the
     specified table with an attribute
     * equal to the given value.
     */
    public boolean checkExists(Connection con, String table, String
attribute, String value) {
        PreparedStatement select;
        Boolean found;

        try {
            select = con.prepareStatement("SELECT * FROM " + table +
" WHERE " + attribute + "=" + value + "");
            ResultSet result = select.executeQuery();
            found = result.next(); // next() returns false if the
ResultSet is empty
        } catch (Exception e) {
            System.out.println(e);
            found = null;
        }

        return found;
    }

    /**
     * This method constructs an SQL select statement from the specified
table
     */
    public String selectStatement(String table, String attribute, String
value) {
        StringBuilder selectBuilder = new StringBuilder();

        selectBuilder.append("SELECT * FROM " + table);
        selectBuilder.append(" WHERE " + attribute);
        selectBuilder.append("=" + value + "");

        return selectBuilder.toString();
    }

    /**
     * This method constructs an SQL select statement using two
attributes
     */
    public String twoAttributeSelect(String table, String attribute1,
String attribute2, String value1, String value2) {
        StringBuilder selectBuilder = new StringBuilder();

        selectBuilder.append("SELECT * FROM " + table);

```

```

        selectBuilder.append(" WHERE " + attribute1);
        selectBuilder.append("='" + value1 + "'");
        selectBuilder.append("and " + attribute2 +"='" +value2+"'");
        return selectBuilder.toString();
    }

    /**
     * This method constructs an SQL delete statement from the specified
table
     */
    public String deleteStatement(String table, String attribute, String
value) {
        StringBuilder selectBuilder = new StringBuilder();

        selectBuilder.append("DELETE FROM " + table);
        selectBuilder.append(" WHERE " + attribute);
        selectBuilder.append("='" + value + "'");
        return selectBuilder.toString();
    }
}

```

History Window (HistoryWindow.java)

```
import javax.swing.JFrame;
import javax.swing.JTextArea;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JTextField;

import java.awt.Dimension;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import javax.swing.WindowConstants;

/**
 * This class handles querying the database by various attributes (date,
 * student name, etc)
 */
public class HistoryWindow {

    private JFrame searchFrame;
    private JLabel searchLabel;
    private JPanel searchPanel;
    private String fields[] = {"PolyID #", "RFID Tag Gtin#", "RFID Tag
Serial#", "Date(YYYY-MM-DD)",
        "Start(YYYY-MM-DD)", "End(YYYY-MM-DD)"};
    private JButton searchByPolyIDBtn;
    private JButton searchByRFIDBtn;
    private JButton cancelBtn;
    private JButton searchByDateBtn;
    private JButton searchByTimeFrameBtn;
    private JButton searchAllBtn;

    private JTextField textFields[] = new JTextField[fields.length];

    public void run() {
        initFrame();
        initButtons();
        buildFrame();
        searchFrame.setVisible(true);
    }

    private void initFrame() {
```

```

searchFrame = new JFrame("History Window");
searchFrame.setSize(700, 600);
searchFrame.setLayout(new FlowLayout());
searchFrame.getContentPane().setBackground(Guil.GREEN_COLOR);
searchFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

// Init container search form banner
searchLabel = new JLabel("HISTORY SEARCH");
searchLabel.setFont(Guil.PLAIN_FONT);
searchLabel.setForeground(Guil.GREEN_COLOR);
}

private void initButtons() {
    // Dimensions
    Dimension btnDimension = new Dimension(300, 25);

    // Create cancel button
    cancelBtn = new JButton("Cancel");
    cancelBtn.setFont(Guil.PLAIN_FONT);
    cancelBtn.setPreferredSize(btnDimension);
    cancelBtn.setActionCommand("Cancel");
    cancelBtn.addActionListener(new ButtonClickListener());

    // Create search by polyID button
    searchByPolyIDBtn = new JButton("Search for Student");
    searchByPolyIDBtn.setFont(Guil.PLAIN_FONT);
    searchByPolyIDBtn.setPreferredSize(btnDimension);
    searchByPolyIDBtn.setActionCommand("searchByPolyID");
    searchByPolyIDBtn.addActionListener(new ButtonClickListener());

    // Create search by RFID tag button
    searchByRFIDBtn = new JButton("Search for Container");
    searchByRFIDBtn.setFont(Guil.PLAIN_FONT);
    searchByRFIDBtn.setPreferredSize(btnDimension);
    searchByRFIDBtn.setActionCommand("searchByRFID");
    searchByRFIDBtn.addActionListener(new ButtonClickListener());

    // Create search by date button
    searchByDateBtn = new JButton("Search by Date");
    searchByDateBtn.setFont(Guil.PLAIN_FONT);
    searchByDateBtn.setPreferredSize(btnDimension);
    searchByDateBtn.setActionCommand("searchByDate");
    searchByDateBtn.addActionListener(new ButtonClickListener());

    // Create search by timeframe button
    searchByTimeFrameBtn = new JButton("Search by Timeframe");
    searchByTimeFrameBtn.setFont(Guil.PLAIN_FONT);
    searchByTimeFrameBtn.setPreferredSize(btnDimension);
    searchByTimeFrameBtn.setActionCommand("searchByTimeFrame");
    searchByTimeFrameBtn.addActionListener(new ButtonClickListener());

    //Create search all button
    searchByTimeFrameBtn = new JButton("Search by Timeframe");
    searchByTimeFrameBtn.setFont(Guil.PLAIN_FONT);
}

```

```

        searchByTimeFrameBtn.setPreferredSize(btnDimension);
        searchByTimeFrameBtn.setActionCommand("searchByTimeFrame");
        searchByTimeFrameBtn.addActionListener(new ButtonClickListener());
    }

    private void print(ResultSet rs) {
        try {
            ResultSetMetaData rsmd = rs.getMetaData();
            int cNum = rsmd.getColumnCount();

            String tuple = "";
            for (int j = 1; j <= cNum; j++) {
                tuple += rsmd.getColumnName(j) + " ";
            }
            tuple += "\n";
            while (rs.next()) {

                for (int i = 1; i <= cNum; i++) {
                    if (i > 1) {
                        tuple += " ";
                    }
                    tuple += rs.getString(i);
                }
                tuple += "\n";
            }
            JTextArea result = new JTextArea(tuple);
            System.out.println(tuple);

            JFrame newFrame = new JFrame("Results");
            newFrame.setSize(800, 600);
            newFrame.setLayout(new FlowLayout());
            newFrame.getContentPane().setBackground(Guif.GREEN_COLOR);

            newFrame.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

            newFrame.add(result);
            newFrame.setVisible(true);

        }
        catch (SQLException se) {
            System.out.println(se);
        }
    }

    public void buildFrame() {
        // Panel dimensions
        Dimension bannerDimension = new Dimension(500, 50);
        Dimension panelDimensions = new Dimension(500, 25);
        Dimension labelDimension = new Dimension(200, 25);
        Dimension tfDimension = new Dimension(200, 25);
    }

```

```

// Build container search banner panel
searchPanel = new JPanel();
searchPanel.setBackground(Color.WHITE);
searchPanel.setPreferredSize(bannerDimension);
searchPanel.add(searchLabel);
searchFrame.add(searchPanel);

// Build form field panels and add them to the frame
for (int i = 0; i < fields.length; i++) {
    if(i == 0) {
        JLabel section = new JLabel("    Search Student");
        section.setFont(Guil.PLAIN_FONT);
        section.setForeground(Color.WHITE);
        section.setPreferredSize( new Dimension(200, 30));
        JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
        sectionpanel.setBackground(Guil.GREEN_COLOR);

        sectionpanel.add(section);
        searchFrame.add(sectionpanel);

    }
    JLabel label = new JLabel(fields[i]);
    label.setFont(Guil.PLAIN_FONT);
    label.setForeground(Color.WHITE);
    label.setPreferredSize(labelDimension);

    JTextField tf = new JTextField();
    tf.setPreferredSize(tfDimension);
    textFields[i] = tf;

    JPanel panel = new JPanel(new FlowLayout(FlowLayout.CENTER,
10, 0));

    panel.setBackground(Guil.GREEN_COLOR);
    panel.setPreferredSize(panelDimensions);
    panel.add(label);

    panel.add(tf);
    if(i == 1) {
        JLabel section = new JLabel("    Search Container");
        section.setFont(Guil.PLAIN_FONT);
        section.setForeground(Color.WHITE);
        section.setPreferredSize( new Dimension(200, 30));
        JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
        sectionpanel.setBackground(Guil.GREEN_COLOR);

        sectionpanel.add(section);
        searchFrame.add(sectionpanel);

    }
    else if(i == 3) {

```

```

        JLabel section = new JLabel("        Search By Date");
        section.setFont(Guil.PLAIN_FONT);
        section.setForeground(Color.WHITE);
        section.setPreferredSize( new Dimension(200, 30));
        JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
        sectionpanel.setBackground(Guil.GREEN_COLOR);

        sectionpanel.add(section);
        searchFrame.add(sectionpanel);

    }
    else if(i == 4) {
        JLabel section = new JLabel("        Timeframe");
        section.setFont(Guil.PLAIN_FONT);
        section.setForeground(Color.WHITE);
        section.setPreferredSize(labelDimension);
        JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
        sectionpanel.setBackground(Guil.GREEN_COLOR);

        sectionpanel.add(section);
        searchFrame.add(sectionpanel);

    }
    searchFrame.add(panel);

}

// Add extra panel for spacing
JPanel panel = new JPanel();
panel.setBackground(Guil.GREEN_COLOR);
panel.setPreferredSize(panelDimensions);
searchFrame.add(panel);

// Add buttons to frame
searchFrame.add(searchByPolyIDBtn);
searchFrame.add(searchByRFIDBtn);
searchFrame.add(searchByDateBtn);
searchFrame.add(searchByTimeFrameBtn);
searchFrame.add(cancelBtn);

}

/**
 * This method handles the actions taken when the search by poly ID
button is pressed.
 */
private void byID() {
    DatabaseFunctions dbf = new DatabaseFunctions();
    String polyid, select_id;
    Connection con = null;

    // Establish connection to database

```



```

        try {
            con = dbf.getConnection();
        } catch (Exception e) {
            System.out.println(e);
            System.out.printf("FAIL");
            return;
        }

        // Grab student and tag info
        polyid = textFields[0].getText();
        select_id = dbf.selectStatement( "history", "polyid", polyid);

        ResultSet rs = dbf.executeQuery(con, select_id);
        print(rs);

        searchFrame.dispose();
    }

    private void byTag() {
        DatabaseFunctions dbf = new DatabaseFunctions();
        String gtin, select_tag, serialnumber;
        Connection con = null;

        // Establish connection to database
        try {
            con = dbf.getConnection();
        } catch (Exception e) {
            System.out.println(e);
            return;
        }

        // Grab student and tag info
        gtin = textFields[1].getText();
        serialnumber = textFields[2].getText();

        select_tag = dbf.twoAttributeSelect("history", "gtin", "serialnumber",
        gtin, serialnumber);

        ResultSet rs = dbf.executeQuery(con, select_tag);
        print(rs);

        searchFrame.dispose();
    }

    private void byDate() {
        DatabaseFunctions dbf = new DatabaseFunctions();
        String date, select_date;
        Connection con = null;

        // Establish connection to database
        try {
            con = dbf.getConnection();

```

```

    } catch (Exception e) {
        System.out.println(e);
        return;
    }

    // Grab student and tag info
    date = textFields[3].getText();
    select_date = dbf.selectStatement( "history", "checkout_date", date);

    ResultSet rs = dbf.executeQuery(con, select_date);

    print(rs);

    searchFrame.dispose();
}

private void byTimeFrame() {
    DatabaseFunctions dbf = new DatabaseFunctions();
    String start, end, select_timeframe;
    Connection con = null;

    // Establish connection to database
    try {
        con = dbf.getConnection();
    } catch (Exception e) {
        System.out.println(e);
        return;
    }

    // Grab student and tag info
    start = textFields[4].getText();
    end = textFields[5].getText();

    select_timeframe = "SELECT * from history where checkout_date >='" +
start +
        "' and checkout_date <='" + end + "'";

    ResultSet rs = dbf.executeQuery(con, select_timeframe);

    print(rs);

    searchFrame.dispose();
}

private void byAll() {
    DatabaseFunctions dbf = new DatabaseFunctions();
    String start, end, select;
    Connection con = null;

    // Establish connection to database
    try {
        con = dbf.getConnection();
    } catch (Exception e) {
        System.out.println(e);

```

```

        return;
    }

    // Grab student and tag info

    select = "SELECT * from history";

    ResultSet rs = dbf.executeQuery(con, select);
    print(rs);

    searchFrame.dispose();
}

/**
 * This private class handles the actions taken when buttons are clicked.
 */
private class ButtonClickListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();

        switch(command) {
            case "searchByPolyID":
                byID();
                break;
            case "searchByRFID":
                byTag();
                break;
            case "searchByDate":
                byDate();
                break;
            case "searchByTimeFrame":
                byTimeFrame();
                break;
            case "searchAll":
                byAll();
                break;
            case "Cancel":
                searchFrame.dispose();
                break;
            default:
                break;
        }
    }
}
}
}

```

IP Window (IPWindow.java)

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JTextField;

import java.awt.Dimension;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.sql.Connection;

/**
 * This class handles querying the database by various attributes (date,
 * student name, etc)
 */
public class IPWindow{
    public static String IPADDRESS;

    private JFrame ipFrame;
    private JLabel ipLabel;
    private JPanel ipPanel;
    private String fields[] = {"IP Address"};
    private JButton ipBtn;
    private JTextField textFields[] = new JTextField[fields.length];

    public void run() {
        initFrame();
        initButtons();
        buildFrame();
        ipFrame.setVisible(true);
    }
    public static String getIPAddress() {
        return IPADDRESS;
    }
    private void initFrame() {
        ipFrame = new JFrame("ENTER IP ADDRESS");
        ipFrame.setSize(500, 200);
        ipFrame.setLayout(new FlowLayout());
        ipFrame.getContentPane().setBackground(Gui1.GREEN_COLOR);
        ipFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```

        // Init container search form banner
        ipLabel = new JLabel("HOST COMPUTER IP");
        ipLabel.setFont(Guil.PLAIN_FONT);
        ipLabel.setForeground(Guil.GREEN_COLOR);
    }

    private void initButtons() {
        // Dimensions
        Dimension btnDimension = new Dimension(300, 25);

        // Create cancel button
        ipBtn = new JButton("Enter");
        ipBtn.setFont(Guil.PLAIN_FONT);
        ipBtn.setPreferredSize(btnDimension);
        ipBtn.setActionCommand("Enter");
        ipBtn.addActionListener(new ButtonClickListener());
    }

    public void buildFrame() {
        // Panel dimensions
        Dimension bannerDimension = new Dimension(500, 50);
        Dimension panelDimensions = new Dimension(500, 25);
        Dimension labelDimension = new Dimension(200, 25);
        Dimension tfDimension = new Dimension(200, 25);

        // Build container search banner panel
        ipPanel = new JPanel();
        ipPanel.setBackground(Color.WHITE);
        ipPanel.setPreferredSize(bannerDimension);
        ipPanel.add(ipLabel);

        // Build form field panels and add them to the frame
        for (int i = 0; i < fields.length; i++) {
            if(i == 0) {
                ipLabel.setFont(Guil.PLAIN_FONT);
                ipLabel.setForeground(Color.WHITE);
                ipLabel.setPreferredSize( new Dimension(200, 30));
                JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
                sectionpanel.setBackground(Guil.GREEN_COLOR);

                sectionpanel.add(ipLabel);
                ipFrame.add(ipLabel);
            }
            JLabel label = new JLabel(fields[i]);
            label.setFont(Guil.PLAIN_FONT);
            label.setForeground(Color.WHITE);

```

```

        label.setPreferredSize(labelDimension);

        JTextField tf = new JTextField();
        tf.setPreferredSize(tfDimension);
        textFields[i] = tf;

        JPanel panel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 10, 0));
        panel.setBackground(Guil.GREEN_COLOR);
        panel.setPreferredSize(panelDimensions);
        panel.add(label);

        panel.add(tf);
        if(i == 1) {
            JLabel section = new JLabel("        Search
Container");

            section.setFont(Guil.PLAIN_FONT);
            section.setForeground(Color.WHITE);
            section.setPreferredSize(new Dimension(200, 30));
            JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
            sectionpanel.setBackground(Guil.GREEN_COLOR);

            sectionpanel.add(section);
            ipFrame.add(sectionpanel);

        }
        else if(i == 3) {
            JLabel section = new JLabel("        Search By Date");
            section.setFont(Guil.PLAIN_FONT);
            section.setForeground(Color.WHITE);
            section.setPreferredSize(new Dimension(200, 30));
            JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
            sectionpanel.setBackground(Guil.GREEN_COLOR);

            sectionpanel.add(section);
            ipFrame.add(sectionpanel);

        }
        else if(i == 4) {
            JLabel section = new JLabel("        Timeframe");
            section.setFont(Guil.PLAIN_FONT);
            section.setForeground(Color.WHITE);
            section.setPreferredSize(labelDimension);
            JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
            sectionpanel.setBackground(Guil.GREEN_COLOR);

```

```

        sectionpanel.add(section);
        ipFrame.add(sectionpanel);

        }
        ipFrame.add(panel);

    }

    // Add extra panel for spacing
    JPanel panel = new JPanel();
    panel.setBackground(Gui1.GREEN_COLOR);
    panel.setPreferredSize(panelDimensions);
    ipFrame.add(panel);

    // Add buttons to frame
    ipFrame.add(ipBtn);

}

private void getIP() {
    // Grab student and tag info
    IPADDRESS = textFields[0].getText();
    ipFrame.dispose();
}

/**
 * This private class handles the actions taken when buttons are
 * clicked.
 */
private class ButtonClickListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();

        switch(command) {
            case "Enter":
                getIP();
                break;
            default:
                break;
        }
    }
}
}
}

```

Parse(Parse.java)

```

import java.util.ArrayList;
import java.util.Scanner;
import java.util.TimeZone;

import javax.swing.JOptionPane;

import java.text.*;
import java.io.*;

/**
 * This class parses the output file given from the M6e reader to obtain
 tag info, as well as time and date of read.
 *
 */
public class Parse {
    private final String readerOutputFilename = "testBox.txt";
    private ArrayList<String> GtinList;
    private ArrayList<String> SerialNumberList;
    private ArrayList<String> LocationList;
    private ArrayList<String> DateList;
    private ArrayList<String> TimeList;
    private ArrayList<String> AntennaList;

    public String getGtin(int index) {
        return GtinList.get(index);
    }

    public String getSerialNum(int index) {
        return SerialNumberList.get(index);
    }

    public String getDate(int index) {
        return DateList.get(index);
    }

    public String getTime(int index) {
        return TimeList.get(index);
    }

    public int getCount() {
        File f = new File(readerOutputFilename);
        Scanner s;
        int count = 0;

        try {
            s = new Scanner(f);
            s.nextLine();

            while (s.hasNextLine()) {

```



```

        s.nextLine();
        count++;
    }
} catch (FileNotFoundException fnfe) {
    System.out.printf("Hi No such file\n");
}

return count;
}

public int getSize() {
    return GtinList.size();
}

public void parseReaderOutput() {
    File f = new File(readerOutputFilename);

    GtinList = new ArrayList<String>();
    SerialNumberList = new ArrayList<String>();
    LocationList = new ArrayList<String>();
    DateList = new ArrayList<String>();
    AntennaList = new ArrayList<String>();
    TimeList = new ArrayList<String>();

    Scanner s;
    int counter = getCount();
    try {
        s = new Scanner(f);
        String line = s.nextLine();

        while (counter-- != 0) {
            s.nextLine();
            line = s.nextLine();
            System.out.printf("%S\n", line);
            Scanner lineScan = new Scanner(line);
            lineScan.useDelimiter(",");
            int count = 0;
            int items = 4;
            while (items-- != 0) {
                String input = lineScan.next();

                if (count == 0) {
                    input = input.replaceAll("\\s+", "");
                    GtinList.add(input);
                }
                else if (count == 1) {
                    input = input.replaceAll("\\s+", "");
                    SerialNumberList.add(input);
                }
                else if (count == 2) {

```

```

        LocationList.add(input);
    }
    else if (count ==3) {
        String delims = "[TZ,]";
        String inputTime = convertTime(input);
        String[] tokens = inputTime.split(delims);

        for (int k = 0; k < tokens.length; k++) {
            if (k == 0) {
                DateList.add(tokens[k]);
            }
            else {
                TimeList.add(tokens[k]);
            }
        }
    }
    else {
        AntennaList.add(input);
    }
    count++;
}
lineScan.close();
count = 0;
}
} catch (FileNotFoundException fnfe) {
    JOptionPane.showMessageDialog(null, "Nothing was read.",
"ERROR", JOptionPane.ERROR_MESSAGE);

    System.out.printf("No such file\n");
}

}

public String convertTime(String utc) {
    DateFormat utcFormat = new
SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
    //utcFormat.setTimeZone(TimeZone.getTimeZone("UTC"));

    try {
        java.util.Date date = utcFormat.parse(utc);

        DateFormat pstFormat = new
SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
        //pstFormat.setTimeZone(TimeZone.getTimeZone("PST"));

        return pstFormat.format(date);
    } catch (ParseException pe) {
        System.out.printf("PARSEFAIL\n");
    }
}

```

```
        return "";  
    }  
}
```

Return Window(ReturnWindow.java)

```

import java.sql.Connection;
import java.sql.ResultSet;

import javax.swing.JOptionPane;

/**
 * This class handles the returning of a container.
 *
 */
public class ReturnWindow {
    public void run() {
        returnContainer();
    }

    private void returnContainer() {
        DatabaseFunctions dsg = new DatabaseFunctions();
        Parse tagGetter = new Parse();
        String polyid, gtin, serialnum, checkout_date, checkout_time,
return_date, return_time;
        String checkout_select, history_insert, student_update,
checkout_delete;
        Connection con = null;
        ResultSet result;
        int parseIndex = 0;

        // Establish connection to database
        try {
            con = dsg.getConnection();
        } catch (Exception e) {
            System.out.println(e);
            return;
        }

        // Read gtin (RFID tag number), date of return, and time of
return
tagGetter.parseReaderOutput();
        gtin = tagGetter.getGtin(parseIndex);
        serialnum = tagGetter.getSerialNum(parseIndex);
        return_date = tagGetter.getDate(parseIndex);
        return_time = tagGetter.getTime(parseIndex);

        // Copy checkout entry and remove from checkout table if gtin
is in checkout table
        if (dsg.checkExists(con, "checkout", "serialnumber",
serialnum)) {
            checkout_select = dsg.selectStatement("checkout",
"serialnumber", serialnum);
            result = dsg.executeQuery(con, checkout_select);

            try {

```

```

        if (result.next()) {
            polyid = result.getString("polyid");
            checkout_date =
result.getString("checkout_date");
            checkout_time =
result.getString("checkout_time");

            // Put checkout copy in history table with
return time and date
            dsg.setNumVals(7);
            history_insert =
dsg.insertStatement("history", "polyid", "gtin", "serialnumber",
                    "checkout_date", "checkout_time",
"return_date", "return_time", polyid,
                    gtin, serialnum, checkout_date,
checkout_time, return_date, return_time);
            dsg.executeStatement(con, history_insert);

            // Update student balance
            student_update =
dsg.updateStatement("student", "balanceowed",
                    "balanceowed-5.00", "polyid",
polyid);
            dsg.executeStatement(con, student_update);

            // Remove checkout entry
            checkout_delete =
dsg.deleteStatement("checkout", "serialnumber", serialnum);
            dsg.executeStatement(con, checkout_delete);
            String success1 = "Successful return:\n"+
"polyid: "+polyid+
"\nreturn_date:"+return_date+"\nreturn_time: "+return_time+"\ngtin: "+gtin
                    + "\nserialnumber: "+serialnum;
            JOptionPane.showMessageDialog(null, success1,
"SUCCESS", JOptionPane.INFORMATION_MESSAGE);

        }
    } catch (Exception e) {
        String error1 = "ERROR: Failed to get entry from
checkout.";

        JOptionPane.showMessageDialog(null, error1, "ERROR",
JOptionPane.ERROR_MESSAGE);
        System.out.println(e);
    }
}
else {

```

```
                String error2 = "ERROR: RFID tag number not found in
checkout.";
                JOptionPane.showMessageDialog(null, error2, "ERROR",
JOptionPane.ERROR_MESSAGE);

                System.out.println("ERROR: RFID tag number not found in
checkout.");
            }
        }
    }
}
```

Search Window (SearchWindow.java)

```
import javax.swing.JFrame;
import javax.swing.JTextArea;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JTextField;

import java.awt.Dimension;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import javax.swing.WindowConstants;

/**
 * This class handles querying the database by various attributes (date,
 * student name, etc)
 */
public class SearchWindow {

    private JFrame searchFrame;
    private JLabel searchLabel;
    private JPanel searchPanel;
    private String fields[] = {"PolyID #", "RFID Tag Gtin#", "RFID Tag
Serial#", "Date(YYYY-MM-DD)",
        "Start(YYYY-MM-DD)", "End(YYYY-MM-DD)"};
    private JButton searchByPolyIDBtn;
    private JButton searchByRFIDBtn;
    private JButton cancelBtn;
    private JButton searchByDateBtn;
    private JButton searchByTimeFrameBtn;
    private JTextField textFields[] = new JTextField[fields.length];

    public void run() {
        initFrame();
        initButtons();
        buildFrame();
        searchFrame.setVisible(true);
    }

    private void initFrame() {
        searchFrame = new JFrame("Search Window");
        searchFrame.setSize(700, 600);
        searchFrame.setLayout(new FlowLayout());
    }
}
```

```

searchFrame.getContentPane().setBackground(Guil.GREEN_COLOR);
searchFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

// Init container search form banner
searchLabel = new JLabel("CONTAINER CHECKOUT SEARCH");
searchLabel.setFont(Guil.PLAIN_FONT);
searchLabel.setForeground(Guil.GREEN_COLOR);
}

private void initButtons() {
    // Dimensions
    Dimension btnDimension = new Dimension(300, 25);

    // Create cancel button
    cancelBtn = new JButton("Cancel");
    cancelBtn.setFont(Guil.PLAIN_FONT);
    cancelBtn.setPreferredSize(btnDimension);
    cancelBtn.setActionCommand("Cancel");
    cancelBtn.addActionListener(new ButtonClickListener());

    // Create search by polyID button
    searchByPolyIDBtn = new JButton("Search for Student");
    searchByPolyIDBtn.setFont(Guil.PLAIN_FONT);
    searchByPolyIDBtn.setPreferredSize(btnDimension);
    searchByPolyIDBtn.setActionCommand("searchByPolyID");
    searchByPolyIDBtn.addActionListener(new ButtonClickListener());

    // Create search by RFID tag button
    searchByRFIDBtn = new JButton("Search for Container");
    searchByRFIDBtn.setFont(Guil.PLAIN_FONT);
    searchByRFIDBtn.setPreferredSize(btnDimension);
    searchByRFIDBtn.setActionCommand("searchByRFID");
    searchByRFIDBtn.addActionListener(new ButtonClickListener());

    // Create search by date button
    searchByDateBtn = new JButton("Search by Date");
    searchByDateBtn.setFont(Guil.PLAIN_FONT);
    searchByDateBtn.setPreferredSize(btnDimension);
    searchByDateBtn.setActionCommand("searchByDate");
    searchByDateBtn.addActionListener(new ButtonClickListener());

    // Create search by timeframe button
    searchByTimeFrameBtn = new JButton("Search by Timeframe");
    searchByTimeFrameBtn.setFont(Guil.PLAIN_FONT);
    searchByTimeFrameBtn.setPreferredSize(btnDimension);
    searchByTimeFrameBtn.setActionCommand("searchByTimeFrame");
    searchByTimeFrameBtn.addActionListener(new ButtonClickListener());
}

public void buildFrame() {
    // Panel dimensions
    Dimension bannerDimension = new Dimension(500, 50);
    Dimension panelDimensions = new Dimension(500, 25);
    Dimension labelDimension = new Dimension(200, 25);
}

```



```

Dimension tfDimension = new Dimension(200, 25);

// Build container search banner panel
searchPanel = new JPanel();
searchPanel.setBackground(Color.WHITE);
searchPanel.setPreferredSize(bannerDimension);
searchPanel.add(searchLabel);
searchFrame.add(searchPanel);

// Build form field panels and add them to the frame
for (int i = 0; i < fields.length; i++) {
    if(i == 0) {
        JLabel section = new JLabel("    Search Student");
        section.setFont(Guil.PLAIN_FONT);
        section.setForeground(Color.WHITE);
        section.setPreferredSize( new Dimension(200, 30));
        JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
        sectionpanel.setBackground(Guil.GREEN_COLOR);

        sectionpanel.add(section);
        searchFrame.add(sectionpanel);

    }
    JLabel label = new JLabel(fields[i]);
    label.setFont(Guil.PLAIN_FONT);
    label.setForeground(Color.WHITE);
    label.setPreferredSize(labelDimension);

    JTextField tf = new JTextField();
    tf.setPreferredSize(tfDimension);
    textFields[i] = tf;

    JPanel panel = new JPanel(new FlowLayout(FlowLayout.CENTER,
10, 0));
    panel.setBackground(Guil.GREEN_COLOR);
    panel.setPreferredSize(panelDimensions);
    panel.add(label);

    panel.add(tf);
    if(i == 1) {
        JLabel section = new JLabel("    Search Container");
        section.setFont(Guil.PLAIN_FONT);
        section.setForeground(Color.WHITE);
        section.setPreferredSize( new Dimension(200, 30));
        JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
        sectionpanel.setBackground(Guil.GREEN_COLOR);

        sectionpanel.add(section);
        searchFrame.add(sectionpanel);

    }
}

```

```

        else if(i == 3) {
            JLabel section = new JLabel("        Search By Date");
            section.setFont(Guil.PLAIN_FONT);
            section.setForeground(Color.WHITE);
            section.setPreferredSize( new Dimension(200, 30));
            JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
            sectionpanel.setBackground(Guil.GREEN_COLOR);

            sectionpanel.add(section);
            searchFrame.add(sectionpanel);

        }
        else if(i == 4) {
            JLabel section = new JLabel("        Timeframe");
            section.setFont(Guil.PLAIN_FONT);
            section.setForeground(Color.WHITE);
            section.setPreferredSize(labelDimension);
            JPanel sectionpanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 10, 0));
            sectionpanel.setBackground(Guil.GREEN_COLOR);

            sectionpanel.add(section);
            searchFrame.add(sectionpanel);

        }
        searchFrame.add(panel);

    }

    // Add extra panel for spacing
    JPanel panel = new JPanel();
    panel.setBackground(Guil.GREEN_COLOR);
    panel.setPreferredSize(panelDimensions);
    searchFrame.add(panel);

    // Add buttons to frame
    searchFrame.add(searchByPolyIDBtn);
    searchFrame.add(searchByRFIDBtn);
    searchFrame.add(searchByDateBtn);
    searchFrame.add(searchByTimeFrameBtn);
    searchFrame.add(cancelBtn);

}

private void print(ResultSet rs) {
    try {
        ResultSetMetaData rsmd = rs.getMetaData();
        int cNum = rsmd.getColumnCount();

        String tuple = "";
        for (int j = 1; j <= cNum; j++) {
            tuple += rsmd.getColumnName(j) + "    ";
        }
    }
}

```

```

    }
    tuple += "\n";
    while (rs.next()) {

        for (int i = 1; i <= cNum; i++) {
            if (i > 1) {
                tuple += "    ";
            }
            tuple += rs.getString(i);
        }
        tuple += "\n";
    }
    JTextArea result = new JTextArea(tuple);
    System.out.println(tuple);

    JFrame newFrame = new JFrame("Results");
    newFrame.setSize(800, 600);
    newFrame.setLayout(new FlowLayout());
    newFrame.getContentPane().setBackground(Guilib.GREEN_COLOR);

newFrame.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

    newFrame.add(result);
    newFrame.setVisible(true);

}
catch (SQLException se) {
    System.out.println(se);
}

}
/**
 * This method handles the actions taken when the search by poly ID
button is pressed.
 */
private void byID() {
    DatabaseFunctions dbf = new DatabaseFunctions();
    String polyid, select_id;
    Connection con = null;

    // Establish connection to database
    try {
        con = dbf.getConnection();
    } catch (Exception e) {
        System.out.println(e);
        System.out.printf("FAIL");
        return;
    }

    // Grab student and tag info
    polyid = textFields[0].getText();

```

```

select_id = dbf.selectStatement( "checkout", "polyid", polyid);

    ResultSet rs = dbf.executeQuery(con, select_id);

    print(rs);

    searchFrame.dispose();
}

private void byTag() {
    DatabaseFunctions dbf = new DatabaseFunctions();
    String gtin, select_tag, serialnumber;
    Connection con = null;

    // Establish connection to database
    try {
        con = dbf.getConnection();
    } catch (Exception e) {
        System.out.println(e);
        return;
    }

    // Grab student and tag info
    gtin = textFields[1].getText();
    serialnumber = textFields[2].getText();

    select_tag = dbf.twoAttributeSelect("checkout", "gtin", "serialnumber",
gtin, serialnumber);

    ResultSet rs = dbf.executeQuery(con, select_tag);
    print(rs);

    searchFrame.dispose();
}

private void byDate() {
    DatabaseFunctions dbf = new DatabaseFunctions();
    String date, select_date;
    Connection con = null;

    // Establish connection to database
    try {
        con = dbf.getConnection();
    } catch (Exception e) {
        System.out.println(e);
        return;
    }

    // Grab student and tag info
    date = textFields[3].getText();
    select_date = dbf.selectStatement( "checkout", "checkout_date", date);

    ResultSet rs = dbf.executeQuery(con, select_date);

```

```

        print(rs);

        searchFrame.dispose();
    }

    private void byTimeFrame() {
        DatabaseFunctions dbf = new DatabaseFunctions();
        String start, end, select_timeframe;
        Connection con = null;

        // Establish connection to database
        try {
            con = dbf.getConnection();
        } catch (Exception e) {
            System.out.println(e);
            return;
        }

        // Grab student and tag info
        start = textFields[4].getText();
        end = textFields[5].getText();

        select_timeframe = "SELECT * from checkout where checkout_date >='" +
start +
            "' and checkout_date <='" + end + "'";

        ResultSet rs = dbf.executeQuery(con, select_timeframe);
        print(rs);

        searchFrame.dispose();
    }

/**
 * This private class handles the actions taken when buttons are clicked.
 */
private class ButtonClickListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();

        switch(command) {
            case "searchByPolyID":
                byID();
                break;
            case "searchByRFID":
                byTag();
                break;
            case "searchByDate":
                byDate();
                break;
            case "searchByTimeFrame":
                byTimeFrame();
                break;
            case "Cancel":

```

```
        searchFrame.dispose();
        break;
    default:
        break;
    }
}
}
```

```

import java.sql.Connection;
//import java.sql.PreparedStatement;
//import java.sql.ResultSet;
import javax.swing.*;

/**
 * This class holds methods that perform system functions such as
 adding/removing containers from the system,
 * querying the database for specific information, etc.
 *
 */
public class SystemFunctions {

    /**
     * This method takes the first container from the parsed reader
 output and adds
     * its gtin (RFID tag number) to the container table.
     */
    public void addContainer() {
        DatabaseFunctions dbf = new DatabaseFunctions();
        Connection con;
        Parse tagGetter = new Parse();
        String gtin, serialnum, container_insert;

        // Parse reader output file to get gtin (RFID tag number)
        tagGetter.parseReaderOutput();

        try {
            con = dbf.getConnection();
            // for (int i = 0; i < tagGetter.getSize(); i++) {

                gtin = tagGetter.getGtin(0);
                serialnum = tagGetter.getSerialNum(0);

                // If the container's serialnumber doesn't already
 exist in container table, add it
                if (!dbf.checkExists(con, "container",
 "serialnumber", serialnum)) {
                    dbf.setNumVals(2);
                    container_insert =
 dbf.insertStatement("container", "gtin", "serialnumber", gtin, serialnum);
                    dbf.executeStatement(con, container_insert);
                    JOptionPane.showMessageDialog(null, "Successfully
 added container:\n gtin: " +gtin+
 "\nserialnumber: "+serialnum,
 "SUCCESS", JOptionPane.INFORMATION_MESSAGE);
                }
            else {

```

```

        String error = "ERROR: Container's gtin is
already in system.";
        JOptionPane.showMessageDialog(null, error,
"ERROR", JOptionPane.ERROR_MESSAGE);
    }

    //    }

    } catch (Exception e) {
        System.out.println("ERROR: addContainer() from
SystemFunctions.java : failed to connect to db");
    }
}

/**
 * This method takes the first container from the parsed reader
output and removes
 * the entry from checkout (if it exists) that has the same gtin.
 */
public void deleteContainer() {
    DatabaseFunctions dbf = new DatabaseFunctions();
    Connection con;
    Parse tagGetter = new Parse();
    String gtin, serialnum, container_delete;

    // Parse reader output file to get gtin (RFID tag number)
    tagGetter.parseReaderOutput();
    gtin = tagGetter.getGtin(0);
    serialnum = tagGetter.getSerialNum(0);

    try {
        con = dbf.getConnection();

        // If the container's gtin and serial number doesn't
already exist in container table, add it
        if (dbf.checkExists(con, "container", "gtin", gtin)
            && dbf.checkExists(con, "container",
"serialnumber", serialnum)) {
            container_delete = dbf.deleteStatement("container",
"serialnumber", serialnum);
            if (!dbf.checkExists(con, "checkout",
"serialnumber", serialnum)) {

                dbf.executeStatement(con, container_delete);
                JOptionPane.showMessageDialog(null,
"Successfully deleted container:\ngtin: " +gtin+
                "\nserialnumber: "+serialnum, "SUCCESS",
JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
}

```



```

        else {
            JOptionPane.showMessageDialog(null,
"Container is currently checked out.",
                                "ERROR", JOptionPane.ERROR);
        }
    }
    else {
        String error1 = "ERROR: Container is not in
system.";
        JOptionPane.showMessageDialog(null, error1, "ERROR",
JOptionPane.ERROR_MESSAGE);
    }
} catch (Exception e) {
    String error2 = "ERROR: Container's gtn is not in
system!!";
    JOptionPane.showMessageDialog(null, error2, "ERROR",
JOptionPane.ERROR_MESSAGE);
}
}
}

```

mysql Create Table Statements

```

CREATE TABLE container(
    gtin VARCHAR(13),
    serialnumber VARCHAR(9),
    PRIMARY KEY(gtin, serialnumber));

CREATE TABLE student(
    polyid VARCHAR(9),
    firstname VARCHAR(45),
    lastname VARCHAR(45),
    balanceowed DECIMAL(10,2),
    PRIMARY KEY(polyid));

CREATE TABLE checkout(
    polyid VARCHAR(9),
    gtin VARCHAR(13),
    serialnumber VARCHAR(9),
    checkout_date DATE,
    checkout_time TIME,
    PRIMARY KEY(polyid, gtin, serialnumber),
    FOREIGN KEY(polyid) REFERENCES student(polyid),
    FOREIGN KEY(gtin, serialnumber) REFERENCES container(gtin,
    serialnumber));

CREATE TABLE history(
    id INTEGER AUTO_INCREMENT,
    polyid VARCHAR(9),
    gtin VARCHAR(13),
    serialnumber VARCHAR(9),
    checkout_date DATE,
    checkout_time TIME,
    return_date DATE,
    return_time TIME,
    PRIMARY KEY(id));

```