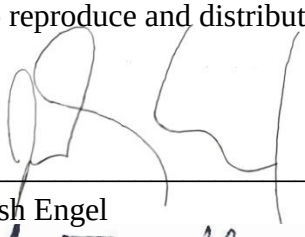
Grocery Navigation Application

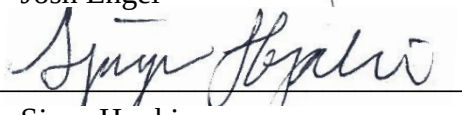
by

Josh Engel, Sjaya Hopkins, William Kohus

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

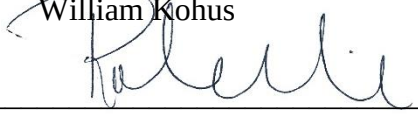


Josh Engel	4/17/2017 Date



Sjaya Hopkins	4/17/2017 Date

William Kohus	4/17/2017 Date



Robin Carew	4/17/2017 Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 17, 2017

Grocery Navigation Application

Prepared By:
Josh Engel
Sjaya Hopkins
William Kohus

Students of University of Cincinnati
College of Education, Criminal Justice, and Human Services School of Information Technology
www.cech.uc.edu/it

Contents

Figures

Tables

Acknowledgements

The Grocery Navigation Application team would first like to thank the members of the University of Cincinnati staff who have educated and motivated us throughout our arduous college journey. There have been many late nights, and long days during which we each individually felt as though success could not be achieved. With the guidance and continued assurance received by the instructors at the University of Cincinnati Main Campus, Clermont Campus and Blue Ash College we have successfully accomplished our goal of becoming college graduates. Graduating with a degree in IT from CECH will not only lead us into our future careers, but will put us on the path to become the next generation of leaders in the IT industry.

We also would like to thank all the help received by members of the Kroger Company who have provided us with the necessary data to take our vision and develop it into the application of which we now present.

Abstract

Kroger has been a leader in the grocery industry for over 130 years, now providing goods and services to families in 2,700 stores which span across 35 states. To manifest this level of success, Kroger has continually evolved to incorporate societal and technological changes. This application continues to build off of this formula for success, integrating the developing mobile computing technology market. This integration culminates in a more expeditious in-store experience and a new avenue to deliver marketing straight to the consumer. The Grocery Navigation Application uses indoor mapping technology to track consumers as they traverse the marketplace, and leads them efficiently to the items they have selected in their personalized shopping cart. This application is available for Android devices with an easy download from the Google Play Store. The innovation seen in this application continues to build on the foundation Kroger has long since set: service, selection and value.

1 Problem Statement

1.1 Introduction

The Kroger Company was founded in Cincinnati, Ohio, in 1883 as a family run business. During this long and rich history Kroger has evolved with the times, but never has left behind the original founding business motto, “Be particular. Never sell anything you would not want yourself” (Kroger, 2016.) Taking this into consideration, it is no stretch of the imagination to understand the commitment to providing a superior shopping experience to the Kroger consumer. The evidence of this has been seen in Kroger’s commitment to refining and creating flowing product shelves. Along with this, Kroger has been committed to the integration of modern technology. Evidenced in 1972 as the Kroger company became the first grocery chain retail store to successfully implement the use of an electronic product scanner. As is well known, product scanners transformed the in-store shopping experience. Presently, the state of technology is in the midst of another technological quantum leap as the budding wireless device market is revolutionizing nearly all aspects of modern life. Leading all businesses in the retail sector to the need with which we are now confronted.

1.2 Project Description

For this project, our team will be designing an application for mobile devices that will put many of the primary tasks tackled during a grocery shopping experience into the palm of a consumer's hand. This application will be designed to allow the creation and implementation of an individual’s personal electronic shopping list, which will be mapped in-store, leading the consumer to the products on the list. Along with the mapping technology, this application paves the way for customized advertising based specifically on a consumer’s list and route. All that is

just the beginning, as technology progresses all shopping experiences will continue to integrate technology and this mobile application creates a solid foundation for those future technologies to be integrated.

1.3 Problem

The immediate problems facing Kroger are keeping up with the modern trends of technology, creating continuity between cross-store shopping experiences, and continually providing growth in advertising opportunities. Currently, Kroger has not had success with in store interaction between a consumer and his/her personal mobile device. This limits the ability Kroger has to reach consumers during a shopping experience. Not only can this application reach a consumer in store, it also provides a seamless transition of a shopping list route to any Kroger store visited. That unique feature resolves the problem of visiting a different store and not knowing where to find needed grocery items. The final problem this application tackles is the need for continued advertising growth. Using mapping technology combined with proximity detectors, advertising will be in the palm of a consumer's hand, in real-time, as the consumer passes by a specific product.

1.4 User Profile

The potential users for the Grocery Navigation Application are consumers in the store, system administrators and customer technical support representatives who specialize with android devices. The consumer will be able to search for grocery products, add or delete them to a shopping list, find a desired store location, scan items for further insight and add them to a checkout list. The system administrators will be able to track the number of customers using the application, change the navigation layout (due to store changes), and adjust the product

throughout the database. Customer technical support representative will be an additional task for a Kroger store associate. The representative tasks include: navigation issues when finding a product, as well as any other general customer solutions a consumer may encounter. The user interface will be simplistic, in line with other Kroger mobile applications, which will create a seamless user experience (see Table 1).

Table 1 User Profile

Potential Users: • Kroger Customers (18 years and older) • System Administrator • Customer technical support
Software and Interface Experience: • Kroger customers who use this app will have at a minimum moderate experience with android devices • System administrators will have a high level of experience in store handling product placement, and have hands-on database knowledge. As well as expert experience with Android handheld devices. • Customer technical support will have a high level of product knowledge in his/her local Kroger store, and familiar with the android application software.
Experience with Similar Applications: • The average Kroger systems administrator and technical support representative are familiar with navigation applications such as google maps, and Kroger shopping technology such as ClickList and Scan-Bag-Go.
Task Experience: • Consumers using this application will be able to locate their local or nearby Kroger store, search for products they want, and add them to a desired shopping list. A consumer can scan items loading further product information (nutrition facts, recommended recipes, coupons, and other products.) A running price total and item list will be saved to the application. • System administrators will be able to update prices for products, analyze data on products being saved, scanned and bought and create more effective navigation routes when shelves and products move in stores. • Customer technical support representatives will help a customer with all in-store needs, as well as product location issues and other general grocery shopping solutions.
Frequency of Use: • The application will reach its highest peak of usage when Kroger stores are most active. During normal business hours, the app will be used moderately, and will see heavier demand during the weekend, and holiday seasons.
Key Interface Design Requirements that the Profile Suggest: • Simplistic and effective UI • Clear interactive navigation route(s) • Simple product searching

1.5 Application Title

Grocery Navigation Application

1.6 Potential Users

Members of the general public who are shopping consumers of grocery stores.

1.7 Software and Interface Experience

The app consists of multiple user interaction screens consisting of a Login, Home, Shopping List, Store List, Mapping Interface, Coupon, and User Profile Account. These options are accessible from a menu, which is located at the bottom of the screen. The login page will provide the user with a choice to either create an account or login with an existing account. Once the user logs in he or she can view his or her saved lists and view preferences. The home screen will offer a shortcut to three options, search for a store, create a shopping list, and view coupons. The shopping list screen will allow users to browse, select, and add items to their shopping list. The store list screen uses Google Maps to find a stores geographical location. The user will be able to select a store and easily locate it. The Mapping interface option once in a store, allows the user to navigate indoors. Coupon Screen, allows the user to view and select coupons saved and or received from previous visits. The user profile account, is where the user will be able to customize his or her personal settings.

1.8 Experience with Similar Applications

Google Maps provide similar mapping functions. The current Kroger Application provides shopping list functionality and a coupon viewer.

1.9 Task Experience

A user is required to have minimal knowledge on using a smartphone and using applications. A user will be able to perform all necessary tasks without complications that might arise from a lack of experience with Android applications.

1.10 Frequency of Use

Anytime the user wants to perform shopping tasks, whether it be in-store shopping or at home adding items to a personal list.

1.11 Key Interface Design Requirements

The design of this mobile application will be built upon mobile development standards to provide an easy to view and use application. The progression through options and menus will be intuitive and user friendly.

1.12 Use Case Diagram

The use case diagram, seen below in figure 1, illustrates how customers, store associates and system administrators interact with the grocery navigation application.

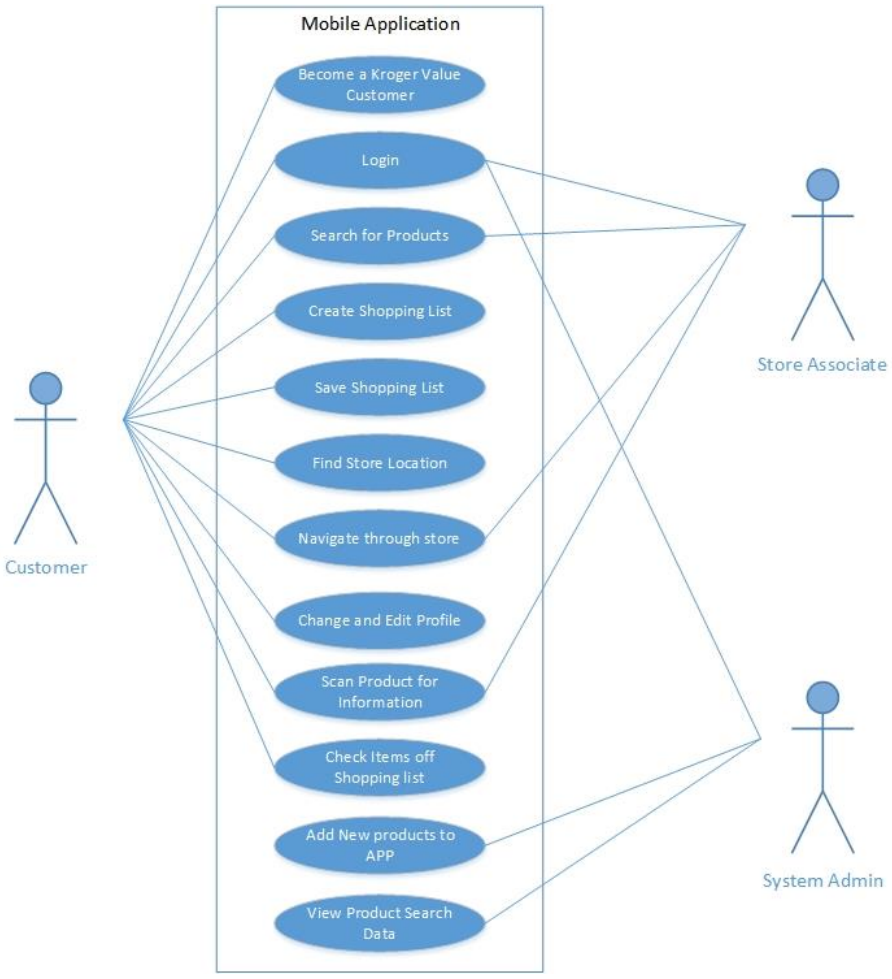


Figure 1 Use Case Diagram

2 Project Management

2.1 Objectives/Deliverables

Project deliverable deadlines (see Table 2)

Table 2 Project Objectives/Deliverables Due Dates

Major Project Milestones (Deliverables)				
Pre Planning Milestone	9/26/2016		**User Interface Milestone**	4/1/2017
Pre Setup Milestone	9/26/2016		**Shopping List Interface Milestone**	3/19/2017
Android Platform Milestone	4/1/2017		**Navigation Milestone**	3/1/2017

2.2 Budget

The cost of production is $31,349. Our prototype will be $0, as this is our senior project and all resources have been provided at no cost. (see Table 3).

Table 3 Project Budget

No.	Item	Unit Each	Unit Price	Line Total Item
		Development Environment		
1	Android Studio	3	$ 0	$ 0
		Labor		
2	Labor	150	$75	$11,250
		Software		
3	Indoor Atlas	< 1,000,000	$20,000/month	$20,000
		Distribution		
4	Google Play Store	1	$99/year	$99
			Total if in Production:	**$31,349**
			Total to Kroger:	**$0**

2.3 Project Schedule

Project schedule with the major milestones listed (see Figure 2).

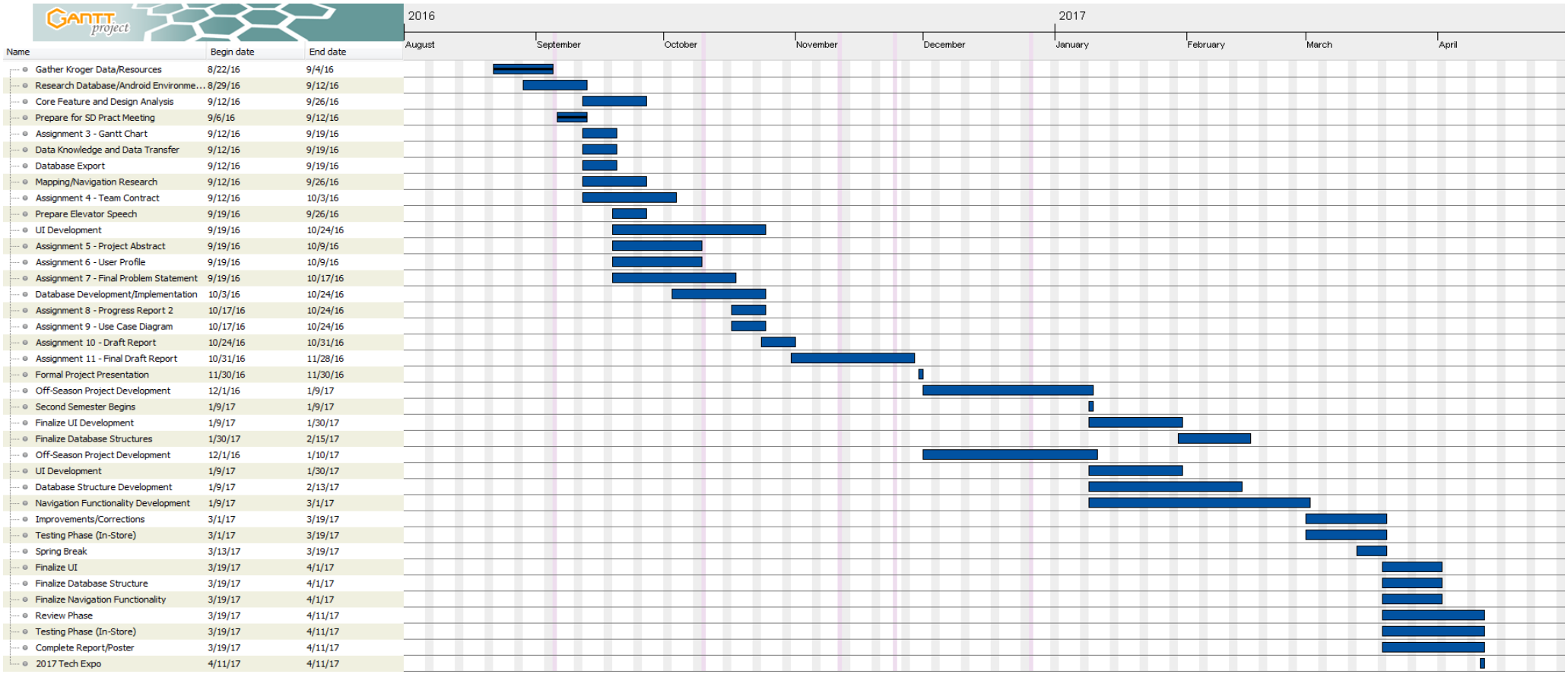


Figure 2 Project Schedule Gantt Chart

3 Technical Elements

3.1 Programing Language

The Grocery Navigation Application will be written in Java using the Android 7.0 Nougat SDK platform, and Android SDK Tools 25.2.2. Grade version for this app is 2.14.1 and the Android Plugin Version is 2.2.0. GitHub is being used for version control services, implementing the use of multiple branches to allow development of multiple features concurrently.

3.2 Platform

Google's Android Studio will be used to develop the application. The development environment is setup on Windows 10 environments, also using Mozilla Firefox teamed with SQLite Manager to view and develop the persistent storage.

3.3 API

The IndoorAtlas API was implemented into our application for indoor navigation.

3.4 Hardware

Mobile Android devices with 5.0 Lollipop or above will support the application.

3.5 Database

A SQLite database was used to persist the Grocery Navigation Application data.

3.6 Development Structure

Figure 3 seen below provides a visual representation of the Grocery Navigation Application development structure.

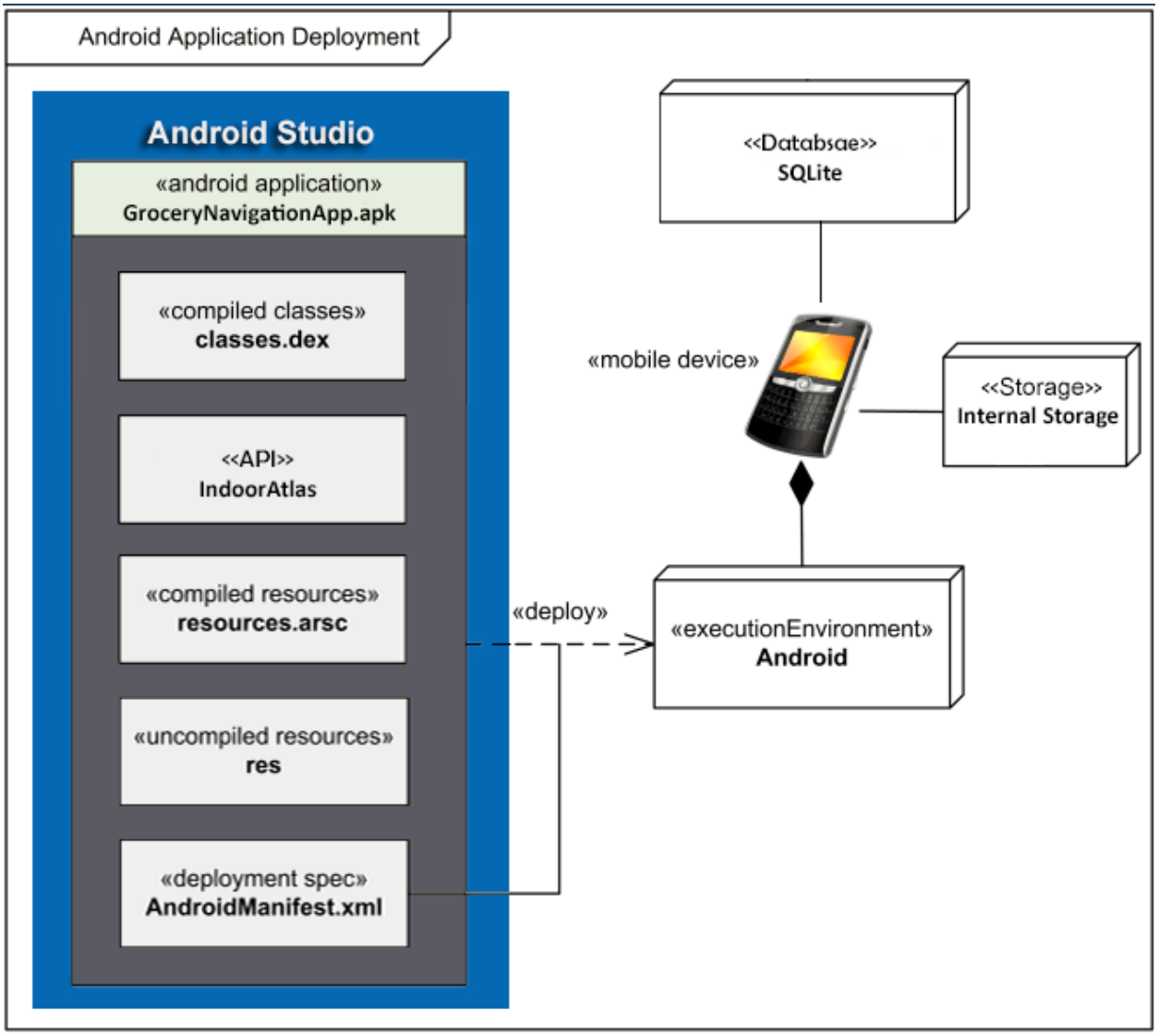


Figure 3 Project Structure

4 Security

In this section of our document, I am going to discuss the security considerations of the Grocery Navigation Application. The resource Grocery Navigation Application used to create a security platform is the Open Web Application Security Project (OWASP). To detail the Grocery Navigation Application's security platform, listed below are the OWASP top 10 mobile security concerns. For each category, a brief description is presented, followed by how it applies to the Grocery Navigation Application, as well as steps that have been taken to ensure the security of the application. Using this resource has better prepared the Grocery Navigation Application for its initial deployment, and has decreased safety concerns for all users and stakeholders.

4.1 Improper Platform Usage

The misuse of application features or the knowing disregard of available safety features. For an intrusion of this type to occur an assailant attacks the application through the user interface by inputting malicious content to reach a vulnerable endpoint. This typically involves uncovering an insecure API call, determining its vulnerability, and using it to infiltrate the application. To strengthen the application against this type of attack we have routinely tested for business logic flaws and ensured proper authentication of user roles.

4.2 Insecure Data Storage

The accessibility of confidential data housed within or available from the mobile device housing the application. Insecure data storage on a device can leave personal confidential data accessible to outside entities. This type of attack can result in the loss of data, or worse the dissemination of confidential personal or business data. The Grocery Navigation Application includes user

accounts and personal data. To ensure the safety of this functionality caching, application backgrounding, logging and data analytics have been considered to prohibit this type of attack.

4.3 Insecure Communication

The interception of streaming data between client – server connections. Assailants sharing internet connections, even from your own home, can perform this type of attack. To withstand this type of attack all sensitive data is encrypted before being sent between the application and application servers.

4.4 Insecure Authentication

The use of subversive techniques to improperly authenticate a user to the application. This can be done with brute force techniques or by exploiting poor password policies. To withstand this type of attack we have deployed a mobile application authentication design pattern ensuring only proper authentication of Grocery Navigation Application users.

4.5 Insufficient Cryptography

The access to data that is improperly encrypted or the use of malware to access data. A malicious app can be installed onto the applications device giving itself access to data within the applications schema. To remedy against this type of attack and persistent to the needs of the Grocery Navigation Application no sensitive data is being saved on an individual's device.

4.6 Insecure Authorization

Not to be confused with authentication, authorization insecurities deal with a logged in user that is able to reach forms and controls that they do not have the proper security access to reach, typically administrative features. The Grocery Navigation Application does include an

administrative set of features. To resist this type of attack a solid set of roles and permissions have been set to ensure application safety.

4.7 Client Code Quality

Poor code leading to vulnerabilities exploited through malicious user input, possible leading to exploitation through malware or phishing schemes. This attack is typically seen through very precisely crafted user inputs meant to break in and confiscate sensitive data. Although every adaptation of this type of attack cannot be thoroughly tested, the Grocery Navigation Application has adapted in depth quality assurance testing procedures to expose poor coding and send to development for repair.

4.8 Code Tampering

This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. These methods of attack are very precise and involve very sophisticate means. The scope of the Grocery Navigation Application does not incorporate specific protections against these types of attacks. However, proper error handling has been implemented where necessary to negate some attempts at these types of infiltrations.

4.9 Reverse Engineering

An attacker downloads your application and uses sophisticated tools to analyze and breakdown the inner workings of an application. To resist this type of attack an application can use an obfuscation tool. Obfuscation tools make the code unintelligible resisting the ability of a would-be attacker to determine the purpose of machine code. At this time the Grocery Navigation Application has not decided to implement obfuscation tools or technology.

4.10 Extraneous Functionality

The addition of code by developers during app production that creates a work around to the applications safety features. An example would be hardcoding in an administrative user and password during development, and then accidentally leaving a reference to it in the production code. The development team has taken extra care to review the Grocery Navigation Application's code base, ensuring all development and debugging code has been removed prior to the release of the product.

5 Testing

5.1 Overview

This section is an overview of the testing methodology for the Grocery Navigation Application. The initial release of this mobile application will only be available on the Android platform. This section is to be used as a reference for the following individuals:

- Project Managers
- Developers
- QA testers
- Business Analyst

5.2 Scope

The scope of testing the Grocery Navigation Application will be based on the requirements of the application and will be tested on an android mobile device.

5.3 Objective

The objective of testing the Grocery Navigation Application, is to test the functionally of all the application features. Quality Analysists will perform these tests in classified suites, allowing an agile workflow in testing. Software Engineers have coded unit tests and will continue to create further unit tests and run them in the through the developing environment.

5.4 Quality Risk

We are aware that the application is still in the beta stages, and is a prototype for a project. Testing will be based on functionality more over than productivity. In developing the applications, there were function that had to be recreated in order to appeal to the client. It is highly possible that many of the general function could change over the course of the project.

5.5 Testing Schedule:

DateSpan	Sprint Number	Sjaya	Bill	Josh
1/15/2017 - 1/29/2017	1	Test the UI for functionality and visual presentation. Create a log of resulting tasks/bugs	Develop a unit testing plan and begin writing coded tests	
1/30/2017 - 2/12/2017	2	Test the UI for functionality and visual presentation. Create a log of resulting tasks/bugs	Work on tasks/bugs created from qa & unit testing. Continue unit testing.	
2/13/2017 - 2/26/2017	3	Test the UI for functionality and visual presentation. Create a log of resulting tasks/bugs	Work on tasks/bugs created from qa & unit testing. Continue unit testing.	
2/27/2017 - 3/12/2017	4	Test the UI for functionality and visual presentation. Create a log of resulting tasks/bugs	Work on tasks/bugs created from qa & unit testing. Continue unit testing.	
3/13/2017 - 3/26/2017	5		Work on tasks/bugs created from qa & unit testing. Continue unit testing.	
3/27/2017 - 4/9/2017	6		Test mobile application onsite. Continue unit testing.	Test mobile application onsite. Work on application tasks/bugs.

Figure 4 Testing Schedule

5.6 Entry and Exit Criteria

The entry and exit criteria for this application is specified for each of the functions (suites) of the application. Each test will run through the entry step and must pass in order to be declared as completed.

- Entry
  - Build is complete
  - Development testing
  - Test Environment set-up (Application files are downloaded to mobile device)
- Exit
  - QA testing
  - Bugs are documents

All steps in test have passed

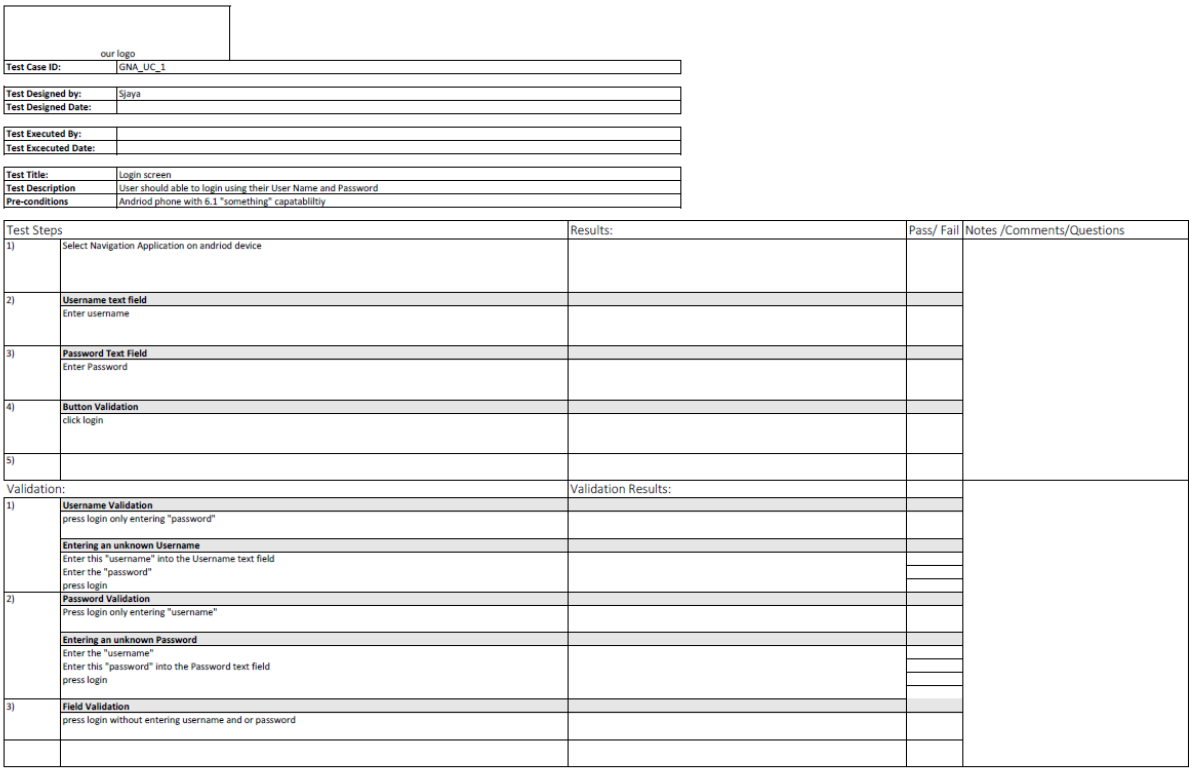
5.7 Testing Configuration and Environments

In order to test the application properly, the tester must have an android mobile device with the most updated firmware. The appropriate suite documentation (using excel) to execute the test, and be able to take screen shots of the application passing and or failing.

5.7.1 Test execution and resources

The testing will be split into suites ex. (GNA_UI_1) would stand for Grocery Navigation Application User Interface suite 1. This all suite 1 of the user interface might belong to the log in screen. If the login screen has more than one test, then the model number would then be GNA_UI_1.2.

Test are executed through the mobile device and an excel document. The tester will follow the steps listed on the excel documentations and can declare pass and or fail based on the test description and requirements listed in the xml sheet.



our logo

Test Case ID:	GNA_UC_1
Test Designed by:	Sjaya
Test Designed Date:	
Test Executed By:	
Test Executed Date:	
Test Title:	Login screen
Test Description	User should able to login using their User Name and Password
Pre-conditions	Andriod phone with 6.1 "something" capatablilty

Test Steps		Results:	Pass/ Fail	Notes /Comments/Questions
1)	Select Navigation Application on andriod device			
2)	**Username text field**			
	Enter username			
3)	**Password Text Field**			
	Enter Password			
4)	**Button Validation**			
	click login			
5)				
Validation:		Validation Results:		
1)	**Username Validation**			
	press login only entering "password"			
	Entering an unknown Username			
	Enter this "username" into the Username text field Enter the "password" press login			
2)	**Password Validation**			
	Press login only entering "username"			
	Entering an unknown Password			
	Enter the "username" Enter this "password" into the Password text field press login			
3)	**Field Validation**			
	press login without entering username and or password			

Figure 5 XML testing Document

5.7.2 Test Case and Bug Tracking

Organizing the application into suites allows us to centralize a bug if need be. This also give our developers a more precise location of where the bug in the code.

- Test Cycles: There will be a series of 3 test cycles for each feature between December 2016 and March 2017
- Change History: Having three test cycles, allows us to track our changes based on our testing, if a section of a feature as already passed, it will be necessary to test it again, once bugs are fixed. Changes and additions to the application will also be tested and documented the same way.

5.8 Unit Testing

Unit tests will be written by the developers where applicable to develop code in a Test Driven Development (TDD) environment. A TDD is a development environment in which a unit test is first created to pass only upon the section of code, method or function, behaving in such a manner as the desired result is received. This creates a blueprint of the applications functionality, allows individuals to code smaller portions of the code independently from the complexity of the totality and promotes direct and simple coding.

The following figure #2 is a flow diagram showing the process, or steps, taken to code a unit test, develop the feature and test its functionality. Figures 3 & 4 are snippets of the ItemDAOTest class. The ItemDAO class is the service that communicates with the database allowing updates, inserts, deletes and retrieval of data. This test class inserts a new record to the database and then retrieves it and ensures the data has been properly persisted.

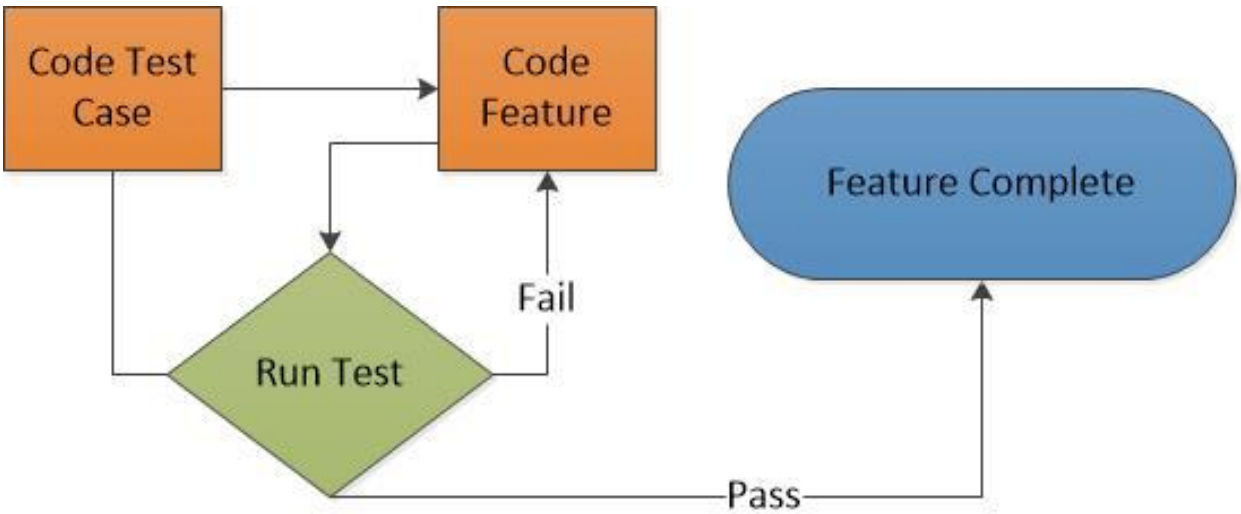


Figure 6 Unit Test Flow Diagram

```
20 public class ItemDAOTest extends TestCase {
21
22     private ItemDAO itemDAO;
23
24     @Override
25     @Before
26     protected void setUp() throws Exception {
27         super.setUp();
28         itemDAO = new ItemDAO(getTestContext());
29     }
30
31     @Test
32     public void itemTest(){
33      try {
34         //test inserting to the DB
35         int itemID = testInsertItem();
36         //check for result
37         assertNotNull(itemID);
38         //test fetching from the DB
39         Item item = testFetchItem(itemID);
40         //test that the item fetched matches the test item
41         assertEquals(item.getPrice(), 2.55);
42         assertEquals(item.getItemDsc(), "Tooth Paste");
43         assertEquals(item.getBrandNme(), "Colgate" );
44         assertEquals(item.getProductCategory(), "non-foods");
45         assertEquals(item.getSkuNumber(), "880-00125");
46     } catch (Exception e) {
47         fail("Should not throw Exception");
48     }
49     }
```

Figure 7 Coded Unit Test

```
51      private Item testFetchItem(int itemID) {
52
53          Item i = itemDAO.fetchOne(itemID);
54
55          return i;
56      }
57
58      /**
59       * This method inserts a new item and returns the new ID
60       */
61      private int testInsertItem(){
62          //grab an Item to insert
63          Item testItem = GetTestItem();
64          //persist to DB
65          long itemID = itemDAO.insert(testItem);
66
67          return (int)itemID;
68      }
69
70      /**
71       * Method to get a temp item for testing
72       * @return
73       */
74      private Item GetTestItem() {
75          Item i = new Item();
76
77          i.setPrice(2.55);
78          i.setItemDsc("Tooth Paste");
79          i.setBrandNme("Colgate");
80          i.setProductCategory("non-foods");
81          i.setSkuNumber("880-00125");
82
83          return i;
84      }
85
```

Figure 8 Coded Unit Test

6 Application Overview

6.1 DAO Package

Data Access Object (DAO) is used in to create and communicate with persistent storage and to form a model of that data to pass throughout the application. Within the DAO there is the Database Helper (DBHelper) class and the Unit of Work Builder (UowBuilder) class. The DBHelper creates the SQLite Database, populates the database with data for our beta application and handles Create, Read, Update and Delete (CRUD) methods. The UowBuilder class creates one object that can house all the data of the persistent storage. Uow is an idea borrowed from C#, which can keep track of change sets and handle the writing and implementing of T-SQL statements for a given application. The Grocery Navigation Application currently relies on the Uow to implement easy access to the objects of the SQLite database.

6.2 DAO Code Snippets

Method called on the first instantiation of the DBHelper class, used to begin the creation process of the SQLite database (see Figure 5).

```
@Override
public void onCreate(SQLiteDatabase db) {

    //create tables
    db.execSQL(createUserTable);
    db.execSQL(createEmailTable);
    db.execSQL(createPhoneTable);
    db.execSQL(createItemTable);
    db.execSQL(createItem_PositionTable);
    db.execSQL(createStoreTable);
    db.execSQL(createStore_AisleTable);
    db.execSQL(createMy_ListTable);
    db.execSQL(createListItemJunctionTable);

    //insert beta data
    db.execSQL(itemInsertScript);
    db.execSQL(item_positionInsertScript);
    db.execSQL(storeInsertScript);
    db.execSQL(store_aisleInsertScript);
}
```

Figure 9 DBHelper Code Snippet

Generic method called to retrieve the table column names for building insert statements of a specific table in the database. The method is passed any DTO class object and uses reflection to grab all the property names, which are a representation of the table columns (see Figure 6).

```
public <T> String getTableColumns(Class<T> dto) {
    StringBuilder sb = new StringBuilder();
    Field[] fields = dto.getFields();
    int i = 0;
    for (Field f : fields){
        if (i == 0) {
            sb.append(f.getName());
            i++;
            continue;
        }
        sb.append(", " + f.getName());
    }
    return sb.toString();
}
```

Figure 10 Table Database Code Snippet

6.3 DTO Package

The term DTO stands for Data Transfer Object and this package is used to store the framework of all the custom objects which are used throughout this application. Within the Grocery Navigation Application are the objects which represent tables of the SQLite database and the Uow object.

6.4 DTO Code Snippet

Class that forms the representation of all the data stored in the database, or any other form of persistent storage. Using this class in conjunction with the UowBuilder class the application has access to all the data as needed (see figure 7).

```
/**
 * This is a container to hold a representation of the data from the database
 *
 * Created by Josh on 9/17/2016.
 */
public class UnitOfWork {

    private List<Item> Item = new ArrayList<>();
    private List<Item_Position> Item_Position = new ArrayList<>();
    private List<Store> Store = new ArrayList<>();
    private List<Store_Aisle> Store_Aisle = new ArrayList<>();
    private List<User> User = new ArrayList<>();
    private List<Item_List> Item_List = new ArrayList<>();
    private List<My_List> My_List = new ArrayList<>();
```

Figure 11 Unit of Work Builder Code Snippet

6.5 Fragment & Indooratlasview Packages

These two packages house all the user interface classes. This handles the implementation of the buttons, text views, and visual elements of the application. Along with that this layer also includes the traditional service layer, or all the logic that handles the functioning of the application. The User Interaction (UI) layer is split into two packages, fragment and IndoorAtlas (indooratlasview). The indooratlasview package incorporates all the activities of the application. The fragments are small pieces which can be placed inside of the views to show a portion of functionality.

6.6 Fragment & Indooratlasview Code Snippets

The Coupon Fragment class implements the functionality that a user can use to view the newest weekly ad, giving them the ability to choose any items that are on sale for their list (see Figure 8).

```
public class CouponFragment extends Fragment {

    public CouponFragment() {
        // Required empty public constructor
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_weekly_ad, container, false);
        WebView wv = (WebView) view.findViewById(R.id.webView);
        wv.loadUrl("https://wklyads-krogercincinnati.kroger.com/flyers/krogercincinnati-weekly?type=1&locale=en&
        postal_code=45243&skip_postal_code=1&store_code=00915");

        // Enable Javascript
        WebSettings webSettings = wv.getSettings();
        webSettings.setJavaScriptEnabled(true);

        // Force links and redirects to open in the WebView instead of in a browser
        wv.setWebViewClient(new WebViewClient());

        return view;
    }
}
```

Figure 12 Coupon Fragment Code Snippet

The following snippet is taken from the MainActivity and highlights what all goes into loading the basic framework of the application and the implementation of a fragment (see Figure 9).

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    //create a new instance of the database
    myDB = new DBHelper(this);
    SQLiteDatabase db = myDB.getWritableDatabase();

    //Google Map
    mapFragment = SupportMapFragment.newInstance();
    viewPager = (ViewPager) findViewById(R.id.viewpager);
    setupViewPager(viewPager);
    tabLayout = (TabLayout) findViewById(R.id.tabs);
    tabLayout.setupWithViewPager(viewPager);
    setupTabIcons();

    pageHistory = new Stack<Integer>();
    viewPager.setOnPageChangeListener(new ViewPager.OnPageChangeListener() {

        @Override
        public void onPageSelected(int arg0) {
            if(saveToHistory)
                pageHistory.push(currentPage);
        }

        @Override
        public void onPageScrolled(int arg0, float arg1, int arg2) {
        }

        @Override
        public void onPageScrollStateChanged(int arg0) {
        }
    });
    saveToHistory = true;

    mapFragment.getMapAsync(this);
}
```

Figure 13 MainActivity Code Snippet

7 Style Guide

7.1 Logos

Standard Logo - Light Background Only

shoppingcartnew600.png

Inverted Logo - Navy Background Only

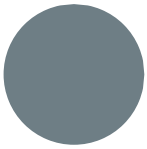
ShoppingCartInvert600.png

7.2 Logo Standards

rectangle.png

7.3 Brand Colors

Midnight Blue
#012D58
1, 45, 88



Rolling Stone
#6E7E85
110, 126, 133

White
#FFFFFF
255, 255, 255

7.4 Fonts

Aa

Circula-Medium
abcdefghijklmnopqrstuvwxyz
Weight: normal
Style: normal
1234567890(,.;:?!$&*)

Aa

Times New Roman
abcdefghijklmnopqrstuvwxyz
Weight: normal
Style: normal
1234567890(,.;:?!$&*)

7.5 Usage

CSS

```
font-family: 'Circula-Medium', sans-serif;
```

CSS

```
font-family: 'Times New Roman', sans-serif;
```

8 Conclusion

8.1 Fall Semester 2016

During the Fall Semester of 2016, the Grocery Navigation Application development learned many of the unique intricacies that correlate with the process of designing and building a mobile application. The creation of an application, especially that for a senior project, involves much more than the formation of an idea and its implementation. The process includes hours upon hours of rigorous paperwork. All this work manifests itself in a better designed application and a better final result. We have learned as a group how to handle disagreements and push through for a positive result. The work completed on the application included a basic framework, the menus and the views.

8.2 Spring Semester 2017

During the Spring Semester of 2017, the Grocery Navigation Application team completed the mapping of a grocery store, to an item and the integration of shopping lists to the application. At the conclusion of this semester, a viable beta application was delivered and presented.

9 Works Cited

8.1 Web Resources

Http://www.usdigitalpartners.com. "History of Kroger." The Kroger Co. -. Accessed November 08, 2016. http://www.thekrogerco.com/about-kroger/history-of-kroger.