**Roles and responsibilities.**

**Objective:** This project is for a List Organizer Android. A User will be able to make customer list and save them. They will also be able to add them to a calendar. (Similar to Color Notes)

## Features

* Create List that can be organized.
* Sort List by Title, Date Modified,
* Define Sort options.
* Save List
* Put list into another List
* Add item to List
* Remove item to list.
* Add multiple items to list.
* Delete multiple items to List
* Add List to Calendar.
* Prioritize List
* Export List to pdf,csv.
* Mail List to email.
* Like and Share List on Facebook and Twitter.
* Able to send to other phones/tablets, Cloud, and your computer.
* Flash card view?
* Different views
* Text to speech list ( add as an option)
* Support Different Languages.
* Save List Activity Screen (More Android Specific)
* Organize List to customer categories.
* Define customer categories for your list.
* Have set categories for grouping list.
* Enable or disable categories.
* Define time of days for your list.
* Options to buy pro version from google play
* Make easy to use.
* Make items draggable.
* Easy to view. Ensure thre is box to left to drag. (possibly to left)
* Ensure they are check box to add/remove from list. (possibly to right)
* Add List for reminders. (Handle with adding to calendar)
* Add Tips for beginners
* Add Notification.
* Add customization.(Ability to change text, font, and colors)
* Keep track of history. Able to go back in history.
* Have backup of all list feature.
* Allow landscape or portrait.
* Create Sample list
* Set Title Header of List.
* Add List to Favorites.
* Remove List to Favorites.
* Have Ads.
* Add About
* Add Rule
* Before or after a event at a specific time.
* Allow images
* Program whole day on list. Whole Day scheduler?
* Possible Serialization and De-serialization of lists for security. When saving and opening list.
* Add Alerts for your list.
* Set priorities
* Apply 80/20 rules.

## TO DO LIST

**Done**

**Not Done**

* Create Features List
* Create TO DO LIST
* Define class structure
* Create Simple UML Class Diagram.
* Flesh out design of program.
* Define business requirements.
* Define class structure
* Define classes and their responsibility
* Create Activity with Menu Layout with labels.
* Add GUI VISIO Diagram of program.
* Create Properties menu.
* Create menu toolbar(above, bottom, or both)
* Create GUI
* Implement Threads
* Main Activity
* Options Activity
* Change Background
* Change Color
* Change font,
* Create List.
* Prioritize list
* Have context menu for items selected.
* Create Use Case Document
* Technical document low level. Describes each method.
* Create UML Diagram.
* Create Program Flow Diagram.
* Create PseudoCode view for each method created for easily understanding for development or PseudoCode for a particular functionality.
* Add Unit Test
* Add Unit Testing Documents
* Create Unit Test Documentation for specific scenarios.
* Try/catch Exception Handling
* Document changes on TRAC type system.
* Upload project privately on GitHub
* Identify key issues in problem form similar to below.
* Create a Defect/problem document and recommended solution.
* Definite listed problem above in more detail.
* Identify in other useful framework that may be used.
* Like and share on facebook and twitter. Tweet list.
* How to layout features in GUI.
* Add Multi-language support
* Email List, Shark List, Tweet List

## Priorities

* Set priorities
* Add Title to Spinner
* Save List of Spinner
* Add to List View after spinner item is selected.
* Store List view by the title. In the spinner.
* Create Sample Broadcast Receiver
* Create Sample Intent
* Create Sample Content Provider
* Communicate between fragments.
* Create Features List
* Create TO DO LIST
* Define class structure
* Create Simple UML Class Diagram.
* Flesh out design of program.
* Define business requirements.
* Define class structure
* Define classes and their responsibility
* Create Activity with Menu Layout with labels.
* Add GUI VISIO Diagram of program.
* Create Properties menu.
* Create menu toolbar(above, bottom, or both)
* Create GUI
* Add Intro Screen
* Add events
* Add events for clicking an item and context menu for hold click an item.
* Add Text to speech
* Separate what can be done in Android framework and regular Job framework.
* Have own Java classes do the business logic.
* Pseudocode Category functionality. Each List will have a category. Define enum for category.
* PseudoCode AddList and RemoveList Functionality using Android.
* Pseudocode Priorities to list. List of Item Objects.
* Each Item will have category, priority.
* Pseudocode set and remove priority methods.
* Pseudocode Sort and Autosort. Sort by Date Modified, Category,etc.
* Pseudocode Save Method
* Pseudocode Remove methods.
* Pseudocode tips
* Pseudo code Favorite List/Unfavorite.
* Save and Load data through SQL Lite Database.
* Add Event Design, Long click,double click, Pinch, etcc…

## 80/20 Priorities

Get Spinner done. Able to save items in spinner. Add Click new item in spinner and add item select listener. Item select listen will open dialog box. Able to save a list to the view. Update view

What to do after you click add new Item? (Add item to list in grid update grid).

## Regular User (Client)

## Pro User (Client)

Text to speech

## Default Categories

## TODO

**Done Tasks**

* Create Use Case Document
* Technical document low level. Describes each method.
* Create UML Diagram.
* Create Program Flow Diagram.
* Create PseudoCode view for each method created for easily understanding for development or PseudoCode for a particular functionality.
* Precondition: User has entered the valid conditions
* User login: with right conditions
* Remove Order
* Remove Product
* Serialize Order list for persistent usage. Static Variable.
* Create Order Table. (Add Order object to database)
* Create Customer Table (Add combination of orders and Users to table)(Dates purchased as well). Country Origin, Address, Zip Code, First Name and Last Name
* Customer/User will have FirstName and Last name
* Extend User class to make Customer Class.
* Customer would contain FirstName, LastName, Last Purchase Date, Number of Orders, Number of Products purchased, Last Date logged in
* Add Customer method to populated these fields
* Add Customer to database to populate customer database that will have same fields.
* Add Customer to Orders
* Admin can also view all the customers.
* Have Customer class with static customer list.
* Fix adding datagrid to new column
* Remove Product and remove Order functionality;
* Move Place Order functionality in AddOrderMethod
* Make getcurrentProductID functionality for selectedCell
* Make RemoveProduct for selectedCell as well
* Make RemoveOrder for selectedCell as well
* Code when Placing an order with a blank quantity.
* Make ConfirmBox for Placing an Order.
* Add Customer to table.
* Add Order to table.
* Send Bug to Developer after Application Crash.

**Not Done Tasks**

* **Decide Columns to display for Orders.**
* **Implement Sort for Orders**
* **View Orders dataset in another form**
* **Add Key events for Tab and Enter for all Forms.**
* **Add status update when mouse hovers over Buttons**
* **Add Menu strip items.**
* **Design what menu items should be in the menus using Microsoft Visio 2013**
* **Finish About Form**
* **Finish Account Form**
* **Think about how a user can login and register to be a customer.**
* **Clean up code and re-factor code. Group UI functionality to more methods.**
* **Think how to implement current functionality to work on any computer. (Maybe have a sample hardcode admin and Current Admin )**
* **Add try/catch blocks. And exception handling**
* **Add Sample set of Users, Orders, and Products checkboxes for sample list.**
* **Add ability for admin user to change current database settings?**
* **Incorporate ASP.NET to project.**
* **Make ConfirmBox for Removing Order**
* **Make ConfirmBox for Add/Removing Product for Admin User.**
* **Maybe create check boxes to allow for sample data to fill datagrid instead of data from ProductID table. Or maybe create a database on this system.**
* **Add ASP.NET MVC and JQUERY/Javascript to project**
* **Maybe translate Customer and Orders to XML File.**
* **Generate Receipt class?**
* **Draw Database relationship diagram or create with MS SQL**
* **If Placeorder is clicked, display red message stayed customer must login.**
* **Create Customer Login and Register button.**
* **Can login as a guest.**
* **Can login as a customer. Check customer list for login username. Add Register button to Login Form. Click button goes to registration form. After valid information entered. Create a User for current customer. Assign User to Customer. Assign Customer to database.**
* **Add usability tips to Forms(click here to this, etc..)**
* **Upload project to GitHub**
* **Add validation for Username,password,phonenumber,email,address**
* **Use Thread Class**
* **Add Middle Name to Registration form.**
* **Add Unit Test**
* **Add Unit Testing Documents**
* **Create Unit Test Documentation for specific scenarios.**

SQL Lite Database

**Database**

**Table title**

Column Title

Column category

**Table Item**

Column Title

Column position

Column category

Column item content.

# Priority

## Sample Priority

**Unit Test (Sample Unit Test File)**

**Unit Test Documentation**

**Refactor code in Form Class**

**Add Key events for Tab and Enter for all Forms.**

**Key event for MouseHover**

**Menu bar layout**

**Thread Class**

**Try/catch Exception Handling**

**Sample Data list for User and Customers**

**Sample Defect and Code Fix with Unit Test Documentation.**

**Update Change Phone Number/Address/Email/Password to update customerlist with phone number; (not needed)**

**Customer can place multiple orders**

**Add Customer Unit Test**

**Incorporate ASP.NET to projects.**

**Identify defects**

**Login as Guest(Can Login but cannot place or view orders.) Disable buttons. Assign random guess ID. Can only view products.**

**Change Requirements – Move Username and Password Validation into User class instead of LoginUser Class.**

## ASP.NET

Convert Orders,Products, Receipt to HTML

Run project through ASP.NET. ASP.NET version of projects?

## Product

* Add Brands to products.
* Extend Product class for different product objects
* Add validation for adding a product.
* Change Category to Combo box in Product Form for admin and regular user.(Desktop, Laptop, Accessories, Software, Hardware, Other)
* Add Sorting of category for admin user and user.
* Add Sort by category for different categories. (Sort by category then sort by weight and other)
* Another combo box for sort by other categories.
* Define category as an Enum?
* Put products in datastore.
* Create Product Class and user as objects like User Class.
* Create Interface for Product Class.
* Create Update Product Form
* Create UpdateProduct Method In Product Class
* Create Method to obtain all products from product table and put in product objects then to List of Product objects. Call method before updating product
* Think of design of Update Product Form.
* Update Query method for Product class.
* Make method to get certain products based off of product name and category(?).
* **Search Product Forum. Auto-detect product from typing key words. (Need RegEx)**

## Orders

When a Customer buys a product. Product becomes an Order. Customer has a list of products ordered. Order has a quantity, the Product ordered, and date purchased. (gets current date), username, cost.

Create Add Order Method.

Create Add Multiple Orders.

Create Cancel Order Method.

Create Cancel all Orders.

View AllOrders (admin)

View Orders(username) – View Orders from a particular user.

Create option to update orders. (Have 2 days to update order)

Add Option to Cancel Order for Client Orders Class.

## 

## Client Orders

Buy single orders.

Buy multiple orders

Get your orders.

Update Orders Button go to Update Orders form

Create Update Order Form.

View Account Info

View Current Orders

Order will contain username, Hashtable for Products and last date purchase(For now list of products, quantity)

## Account

Username and email

View Orders Button

Change email

Change password

Contact support

About

**Issues**

**Invalid Phone number Issue 1:**

**How Issue was Found:** Found while testing Unit test for invalid phone numbers

**Description:** When registering a new customer, a valid phone number is required to be entered for registration. The phone number field doesn’t accept phone number without dashes.(3454433456)

Code needs to be change to accept phone number without dashes. Phone number digits should be 9 digits long.

**What type of issue? Code Issue:** Regular Expression was changed to optionally include dashes for phone number. A question mark was added after the dash in the regular expression as seen below. This makes it so dashes are optionally allowing the customer to register their phone number without dashes.

Before Code Change:

/// <summary>

/// Validate if an phone number is valid

/// </summary>

/// <param name="phonenumber">Phone number of customer</param>

public void ValidatePhonenumber(string phonenumber)

{

string regExPattern = "^[2-9]\\d{2}-\\d{3}-\\d{4}$";

if (!System.Text.RegularExpressions.Regex.IsMatch(phonenumber, regExPattern, System.Text.RegularExpressions.RegexOptions.IgnoreCase))

{

NotValidPhonenumber = true;

}

After Code Change:

/// <summary>

/// Validate if an phone number is valid

/// </summary>

/// <param name="phonenumber">Phone number of customer</param>

public void ValidatePhonenumber(string phonenumber)

{

string regExPattern = "^[2-9]\\d{2}-**?**\\d{3}-**?**\\d{4}$";

if (!System.Text.RegularExpressions.Regex.IsMatch(phonenumber, regExPattern, System.Text.RegularExpressions.RegexOptions.IgnoreCase))

{

NotValidPhonenumber = true;

}

Issue 2

**How Issue was Found:** Found while Unit testing for Orders Functionality

**Description:** When a valid customer is unable to view their current order unless they place an order. View current order functionality does not work properly

Code needs to be change to view the current order correctly

**What type of issue? Code Issue:**

Issue 3

**How Issue was Found:** Found while Unit testing for Orders Functionality

**Description:** When customer places an order, the customer username is not written to Orders table.

Code needs to be change to view the current order correctly

**What type of issue? Code Issue:**

Issue 3

**How Issue was Found:** Found while Unit testing for Change Phone Number Functionality

**Description:** When customer wants to change phone number, Database is updated corrected but customerlist is not updated

Code needs to be change to update the current customer list with the changed phone number method.

**What type of issue? Code Issue:**