# IT209: Final Project: Apps Cart

**Description:**
We will design a simple apps store. The purpose of this project is to demonstrate an acceptable level of expertise with the fundamental procedural and object-oriented concepts, Graphical User Interface (GUI) and data format (csv for this app) implementation techniques introduced and refined in lectures and labs during the course of the semester.

Recall that the acceptable resources for this assignment differs from those approved for lab assignments, and are limited to the class text, Python Library, Language and Tutorial references, lecture and lab slides/notes, lecture and lab instructors. **If you have questions, use virtual help from Instructor/GTA to obtain assistance.**

**Please note: This is an individual project and no collaboration is allowed other than accepted resources.**

**Project Requirements**:

- The source code for this lab must be submitted in files named using the accepted format.
- The source code files must contain a file header using the accepted format. The header information must be complete and accurate.
- The source code file should use self-documenting code and additional comments (as required) to improve code readability and should use appropriate method and function header comments.

**Project Submission**:
You need to submit the following files:
1. A class file containing all the class and methods (please see classes and methods section). The name of the file should be in the following format: **Firstname_lastname_Project_CLASS.py**
2. A GUI file containing the GUI applications and class implementation. The name of the file should be in the following format: **Firstname_lastname_Project_GUI.py**

**Program Structure:**

1. Program starts by reading from an input file containing an app information per line. (in CSV format). The information is separated by comma and saved in a .csv file.

2. Once read, an apps item will be created and saved to a dictionary ‘category_dict’ of apps per category. The key will be the category like: Store Design, Marketing etc. and the values will be the apps item object.
3. The program needs to create **two** more dictionaries ‘ratings_dict’ where the apps will be saved by ratings. Key will be ratings and the values in the dictionary will be the apps item object. If an apps has a rating of 4.5 then it will be saved under key 4, if an apps has a ratings of 2.1 , then it will be saved under key 2. The other dictionary is ‘price_dict’ similar as rating except key will be price. For example: all the apps between $1.00 - $1.99 will be under key 1.00$ and the list of App object will be the value.
4. A testing code is provided to check the class file. You need to comment the testing code once done checking the class file.
5. The GUI file imports the class file and use the App and Cart class information.
6. User has the option of ordering apps from different category or choose by apps ratings or by price. Once user selects a category, s/he will be directed to the list of apps available to purchase for that category.
7. Each list, loads the items from the corresponding category dict. If user selects Marketing, then all the apps under Marketing will be loaded from the category_dict[‘Markting’] from the class file.
8. For each apps, there will be some information that will be displayed to user.
    a. A check box to select the apps to purchase
    b. A label to show the name of the app
    c. Two Labels to display developer and description of the apps
    d. A label to display the price
    e. A label to display user ratings
    f. A label to display category of the apps
9. Once user selects item(s), and hits add to cart button then a subtotal label will appear.
10. User can then have the option of choosing more apps by other options (rating/pricing) or can checkout.

11. Checkout shows apps purchased, break-down of each apps and their information, tax and total.
12. A description of CLASS and GUI part is as follows:

**Cart Class:**

**Cart** is a list subclass. A Cart object contains all the apps object user purchased. It works like a list, where each item is an apps object.

| **Methods:** | **Description** | **Parameters** | **Return type** |
|---|---|---|---|
| subtotal: None | Returns subtotal from the **Cart** object. Iterate over all the **items** values to obtain subtotal (before applying tax) | None | float |

**App Class:**

**App** class contains three class variables to hold different types of apps per category or per ratings pr by price. These dictionaries are: category_dict, rating_dict and price_dict.

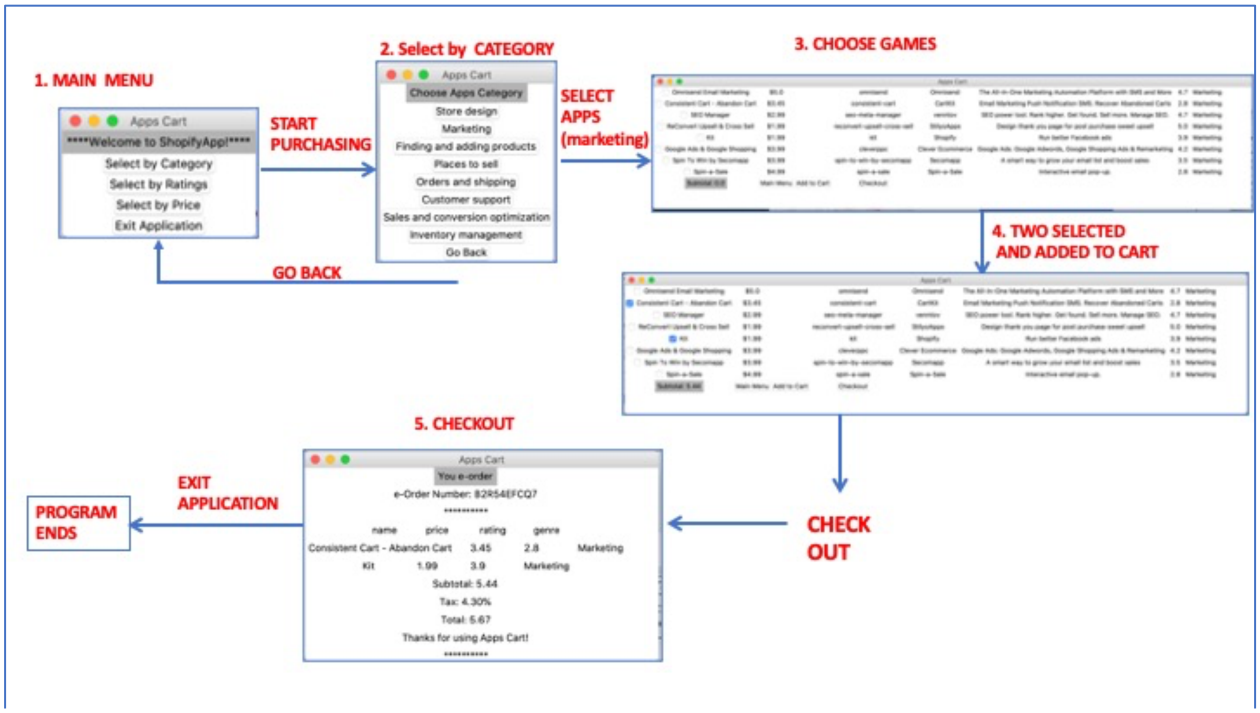
| **Methods:** | **Description** | **Parameters** | **Return type** |
|---|---|---|---|
| **__init__()** | Initialization method to initialize an app object. Each app has following information: ID, name, developer, description, price, rating, category. See below for details**@@**. Each apps will be added to category_dict, rating_dict and price_dict as described above (Program structure #2 and #3). | ID, name, developer, description, price, rating, category | None |
| **get_X()** | Returns X attribute of an app, where X = ID, name, developer, description, price, rating, category | None | string/float |

**@@**

| **Attribute name** | **Description** | **type** |
|---|---|---|
| id | Each app has a unique identifier. This is a private attribute | string |
| name | Private attribute. Name of an apps. | string |
| developer | Private attribute. Developer name of an app. | string |
| description | Private attribute. Description of an app. | string |
| price | Private attribute. Price of an apps | float |
| rating | Private attribute. User rating of an apps | float |
| category | Private attribute. Category of an apps | string |

**Graphical User Interface:**

1. **Welcome screen:** This screen will be the first screen to display. It will contain following widgets:
    a. Welcome message: Label
    b. Select by category: Button – will take user to different categories. Figure 1, #2
    c. Select by rating: Button – will take user to select different ratings Figure 2 #2
    d. Select by price: Button – will take user to select different price from 3 #2
    e. Exit Application: Button – exit the program



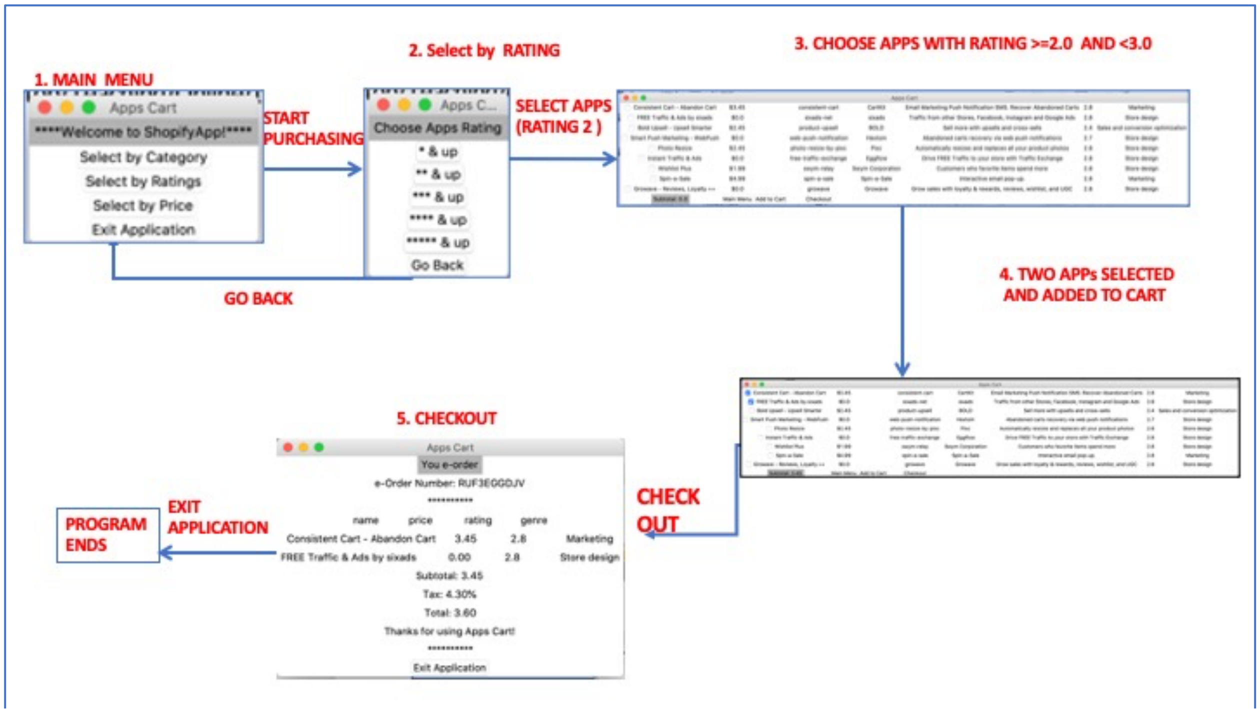
| | | | | | | |
|---|---|---|---|---|---|---|
| Omnisend Email Marketing | $0.0 | omnisend | Omnisend | The All-in-One Marketing Automation Platform with SMS and More | 4.7 | Marketing |
| Consistent Cart - Abandon Cart | $3.45 | consistent-cart | CartKit | Email Marketing Push Notification SMS. Recover Abandoned Carts | 2.8 | Marketing |
| SEO Manager | $2.99 | seo-meta-manager | venntov | SEO power tool. Rank higher. Get found. Sell more. Manage SEO. | 4.7 | Marketing |
| ReConvert Upsell & Cross Sell | $1.99 | reconvert-upsell-cross-sell | StilyoApps | Design thank you page for post purchase sweet upsell | 5.0 | Marketing |
| Kit | $1.99 | kit | Shopify | Run better Facebook ads | 3.9 | Marketing |
| Google Ads & Google Shopping | $3.99 | cleverppc | Clever Ecommerce | Google Ads: Google Adwords, Google Shopping Ads & Remarketing | 4.2 | Marketing |
| Spin To Win by Secomapp | $3.99 | spin-to-win-by-secomapp | Secomapp | A smart way to grow your email list and boost sales | 3.5 | Marketing |
| Spin-a-Sale | $4.99 | spin-a-sale | Spin-a-Sale | Interactive email pop-up | 2.8 | Marketing |
| Subtotal: 0.0 | Main Menu | Add to Cart | Checkout | | | |

```
e-Order Number: 82R54EFCQ7
**********
name                            price    rating    genre
Consistent Cart - Abandon Cart   3.45      2.8         Marketing
Kit                              1.99      3.9       Marketing
Subtotal: 5.44
Tax: 4.30%
Total: 5.67
Thanks for using Apps Cart!
**********
```

***Figure 1 Choose by Category****: App Cart –User needs to select a category #2, select an apps in the category(#3 ), hitting ‘Add to Cart’ button #4 will add the items in the cart and display the current subtotal. User can choose to select another category or by rating or by price by selecting button ‘Select by Category’/ ‘Select by Ratings’/ ‘Select by Price’ button or checkout for final receipt as shown in #5.*

2. **Choose By Category:** If user presses start by category, then the program will display another frame with different categories of apps (Figure -1 #2). This window has one button widget for each category of apps. For example: in figure -1 #2 user selects Marketing and then the program will display all the Marketing apps. Go Back: Button – will take to step 1 again. Selecting the category will take user to all the apps currently in the apps store. This window contains the following widgets:
    a. Name of the apps: check button – user selects which apps they want to buy
    b. Price of an apps: Label
    c. id: Label -id of an item
    d. developer: Label
    e. Description: label
    f. User rating: label of the apps rating
    g. Category: Category of the apps.
    h. Main Menu: Button will take again to main menu (Figure-1, #1)
    i. Add to Cart: Button – Add selected items to cart
        i. Will display a current subtotal as label after selecting an app and pressing this button (Figure-1, #4)

j. Checkout: Button – After adding to cart, user can directly go to checkout window (#5).



```
Apps Cart
You e-order
e-Order Number: RUF3EGGDJV
**********
name          price      rating     genre
Consistent Cart - Abandon Cart    3.45     2.8     Marketing
FREE Traffic & Ads by sixads      0.00     2.8     Store design
Subtotal: 3.45
Tax: 4.30%
Total: 3.60
Thanks for using Apps Cart!
**********
Exit Application
```

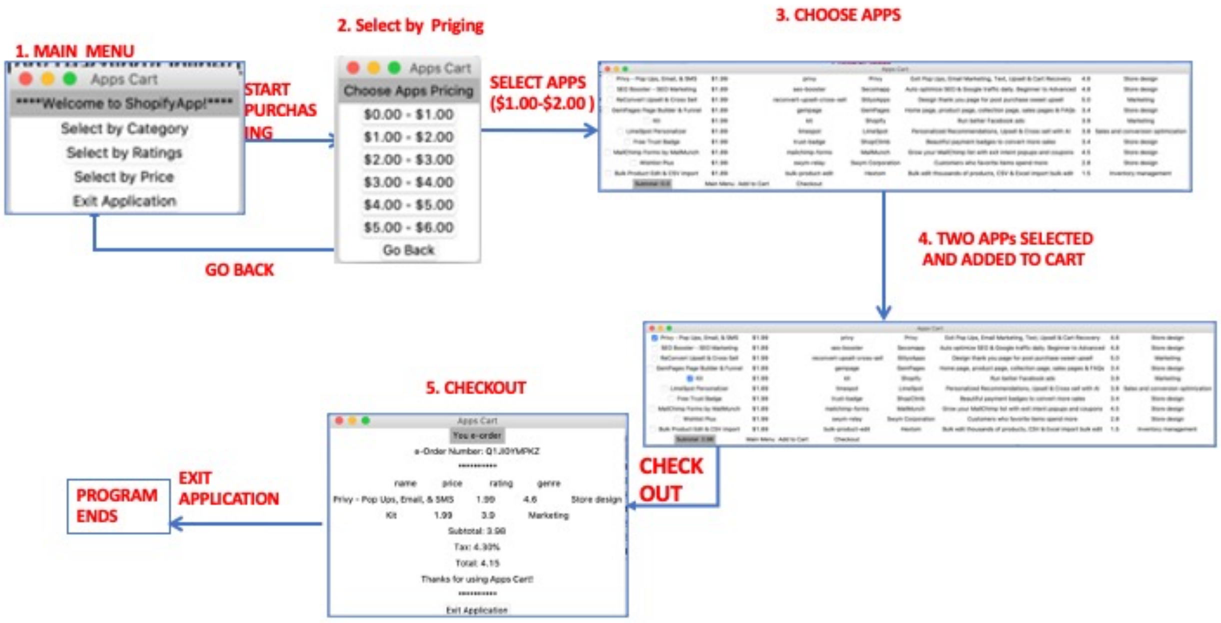
***Figure 2: Choose by Rating*** *Apps Cart –User can select apps by ratings #2, select an apps in that rating(#3 ), hitting 'Add to Cart' button #4 will add the items in the cart and display the current subtotal. User can choose to select another apps by category/rating/price selecting button 'Select by Category'/ 'Select by Ratings'/'Select by Price' button or checkout for final receipt as shown in #5.*

3. **Choose by Ratings:** If user presses start by ratings (Figure -2, #2) then the program will display another frame with different apps with same ratings(Figure -2 #3). This window has one button widget for each rating of apps. For example: in figure -2 #2 user selects rating 4 and then the program will display all the game apps with rating >=4.0 and < 5.0. For example: Figure- 2 #3 shows all the apps available. This window contains the following widgets:
    a. Name of the apps: check button – user selects which apps they want to buy
    b. Price of an apps: Label
    c. id: Label -id of an item
    d. developer: Label
    e. Description: label
    f. User rating: label of the apps rating
    g. Category: Category of the apps.
    h. Select by category/rating: Buttons will take again to main menu (Figure-1, #1)

i. Add to Cart: Button – Add selected items to cart
    i. Will display a current subtotal as label after selecting an app and pressing this button (Figure-1, #4)

j. Checkout: Button – After adding to cart, user can directly go to checkout window (#5).

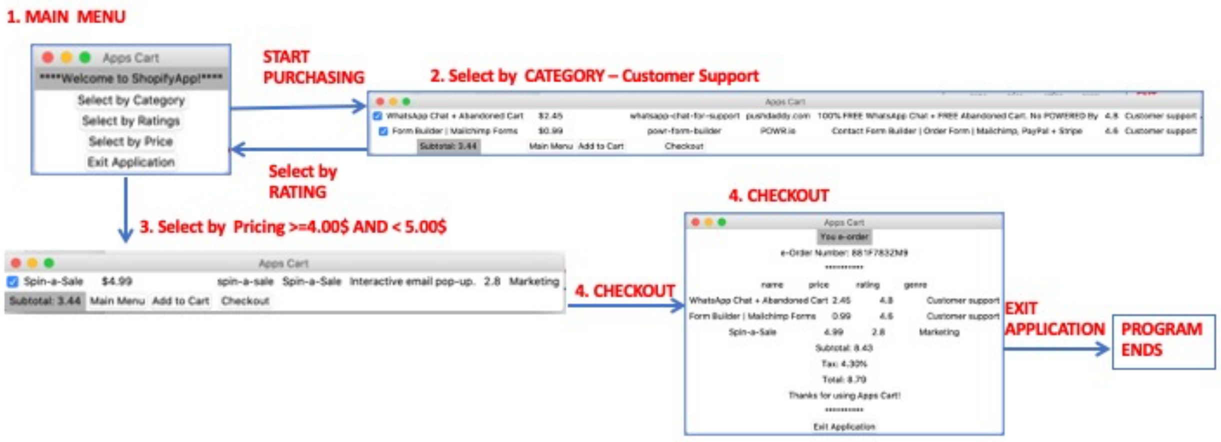
4. **Choose by Price:** If user presses start by pricing (Figure - 3, #2) then the program will display another frame with different apps with pricing for a range(Figure -2 #3). For example: if use selects $0.00 - $1.00 then all the apps with pricing of >= $0.00 and < 1.00$ will be displayed. The window has one button widget for each pricing range of apps. For example: in figure -2 #2 user selects rating $1.00-$2.00 and then the program will display all the apps with pricing >= $1.0 and < $2.0. This window contains the following widgets:
    a. Name of the apps: check button – user selects which apps they want to buy
    b. Price of an apps: Label
    c. id: Label -id of an item
    d. developer: Label
    e. Description: label
    f. User rating: label of the apps rating
    g. Category: Category of the apps.
    h. Select by category/rating: Buttons will take again to main menu (Figure-1, #1)
    i. Add to Cart: Button – Add selected items to cart
        i. Will display a current subtotal as label after selecting an app and pressing this button (Figure-1, #4)
    j. Checkout: Button – After adding to cart, user can directly go to checkout window (#5).



```
Apps Cart
You e-order
e-Order Number: Q1JI0YMPKZ
**********
name        price       rating      genre
Privy - Pop Ups, Email, & SMS     1.99     4.6     Store design
Kit        1.99       3.9       Marketing
Subtotal: 3.98
Tax: 4.30%
Total: 4.15
Thanks for using Apps Cart!
**********
Exit Application
```

***Figure3: Choose by Pricing*** *Apps Cart –User can select apps by pricing #2, select an apps in that pricing(#3 ), hitting 'Add to Cart' button #4 will add the items in the cart and display the current subtotal. User can choose to select another apps by category/rating/price selecting button 'Select by Category'/ 'Select by Ratings'/'Select by Price' button or checkout for final receipt as shown in #5.*

5. **Checkout:** Checkout window displays the summary of purchase with following widgets:
   a. Your e-receipt: Label
   b. Receipt Number: Label - Randomly generated by program
   c. Name Price Quantity Unit: Header Label
   d. Item purchased, price, rating, genre/category as Labels
   e. Subtotal: Label
   f. Tax: Label
   g. Total: Label
   h. 'Thank you' message: Label
   i. Exit application: Button – exit the program



```
****Welcome to ShopifyApp!****
```

```
WhatsApp Chat + Abandoned Cart   $2.45   whatsapp-chat-for-support  pushdaddy.com  100% FREE WhatsApp Chat + FREE Abandoned Cart. No POWERED By  4.8  Customer support
Form Builder | Mailchimp Forms   $0.99   powr-form-builder          POWR.io        Contact Form Builder | Order Form | Mailchimp, PayPal + Stripe  4.6  Customer support
Subtotal: 3.44
```

```
Spin-a-Sale   $4.99   spin-a-sale  Spin-a-Sale  Interactive email pop-up.  2.8  Marketing
Subtotal: 3.44
```

```
You e-order
e-Order Number: 881F7832M9
***********
name         price      rating      genre
WhatsApp Chat + Abandoned Cart  2.45   4.8   Customer support
Form Builder | Mailchimp Forms  0.99   4.6   Customer support
Spin-a-Sale                     4.99   2.8   Marketing
Subtotal: 8.43
Tax: 4.30%
Total: 8.79
Thanks for using Apps Cart!
***********
```

***Figure4**: Apps Cart –User can select apps in different ways in one purchase. Select by apps category #2 or select by pricing #3. The program runs until user wants to close or decide to checkout.*

**Grading Rubric:**

| | **Excellent** | **Average** | **Needs Improving** | **Points** |
|---|---|---|---|---|
| Submission Details | File names and file header meet stated spec. | Either file name is incorrect or file header is missing sections/details. | Both file name and file header are missing or are incorrectly implemented. | ___/ 5 |
| Comments, Self-Documenting Code and Method headers (docstring) | Comments, self-documenting code and method headers clearly convey intent of code. | Comments, self-documenting code and method headers generally convey intent of code. | Comments, self-documenting code and method headers are largely missing or trivially implemented. | ___/ 5 |
| Class File | Class completely implements the attributes and behaviors required for the application. | Class partially implements the attributes and behaviors required for the application. | Class is not implemented or is trivially implemented (to the point everything is in one module) | ___/ 20 |
| GUI Class | Contains all specified GUI components and event model operations. | Contains GUI components and event model operations that significantly implement intended operations. | Contains some GUI components and event model operations but cart operation is incomplete. | ___/ 30 |
| Driver code | Correctly initializes application (using the appropriate classes). | Initializes application with minor issues. | Driver module not implemented or dysfunctional. | ___/ 5 |
| I/O | Specified I/O fully implemented. | Specified I/O is mostly implemented, with minor deficiencies. | I/O is significantly dysfunctional. | ___/ 15 |
| **Overall Score** | | | | **___/80** |