Jameson Albers

CS 5001 Final Project: Move Helpr

Project Inspiration

Since I was 17, I've never lived in the same place for more than 3 years. In turn, I've been through quite a few moves! Whether it was driving a U-Haul across the Midwest, packing boxes and loading them on a truck, or letting a moving company do all the hard work for me, the common thread is that I could never find a good way to keep track of my belongings.

If you pulled the average person aside on the street and asked them what they used to inventory their belongings during their last move, odds are that they either used notebook paper or Excel spreadsheets. Unfortunately, these are both unwieldy and are not easy to use during the process of packing and moving. Due to these limitations, I was inspired to create a convenient solution that works at the point-of-action during your move to simplify the moving process.

Basic Design Principles

I used four basic principles while designing and creating this application: Object-Oriented Design, File Handling, Ease of Use, and Mobile Technology.

- **Object-Oriented Design**
  Every object we pack has certain characteristics that we care about with regards to moving: value, quantity, condition, serial number, etc. Since we care about these specific aspects of every item we are moving, it makes sense to think of them as objects and design classes to implement them. The same goes for boxes: we care about the room they were packed in as well as their value.
- **File Handling**
  We need to store this information about our items and boxes when the program is not running. We can use files to implement persistent storage for our item and box information.
- **Ease of Use**
  The reason paper notebooks and Excel sheets suck for moving is that they are extremely inconvenient. You have to enter tons of information manually and do a lot of calculations yourself. Why not keep track of things with an intuitive menu system?
- **Mobile Technology**
  Just about everyone these days uses a smartphone, so why not leverage it to help us move? The smartphone camera can be used to take pictures of items to archive their condition as well as scan barcodes and qr codes for easy information retrieval.

Project Scope

End of Semester Goal

By the end of the semester, the goal was to have the application's backend foundation developed and usable with a text interface. This was mostly achieved; the program runs and most desired features work. However, there are still some bugs that require fixes before the backend can be considered complete.

Final Product Goal

The eventual goal for this project is to develop a complete hybrid application that is able to run natively on mobile devices through an application wrapper as well as on a web browser. This enables access to functionality from any device at any location. It can also leverage the technologies built into mobile devices to simplify the user experience.

Course Concepts Used

- **Variables**
  Variables are used throughout the program to store information, pass to functions, and receive from functions.

- **Data Types**
  This program uses all basic data types, to include strings, integers, floating point numbers, and booleans. They are used to specify values or create conditional statements and loops.

- **Data Structures**
  Lists and sets are the main data structures used. Dictionaries are not used because the key/value recall system of the dictionary is replaced by file attributes and getter/setter methods using classes.

- **Loops** This program uses while and for loops to iterate functions and calculations through iterable data types and to repeat actions until certain conditions are met.

- **Functions** Functions are used extensively in this program to perform specific actions on specific information. Utilizing functions also makes the program easier to test and debug. Recursion is not used in this program because loops are more efficient at performing the actions in this program where recursion could be implemented.

- **Classes**
  Classes form the backbone of the program. The Item and Box classes are the blueprint around which all the data and functions operate.

- **Files**
  Files are used for persistent data storage when the program is not running. They also separate data by user.

Areas for Future Development

Moving Companies

A similar version of this application could be created for moving companies to simplify item tracking, asset value, document handling, and shipment coordination. Sharing data between the customer and moving company interfaces will keep both parties accountable and streamline the moving proccess.

Logistics and Inventory

This application could be modified to keep track of any items in an organization's inventory or help track shipment of goods. Using cheap, non-proprietary hardware (i.e. low-cost smartphones) would reduce costs and reduce the amount of training time needed for employees to manage inventory.

Download and Use

Refer to these instructions to properly download and use the program. The program requires the installation of Python 3 and the following external Python packages to function:

- qrcode
- reportlab
- PIL (installed with qrcode)

Download Instructions

To download the program, simply save the *driver, item,* and *box* Python files to the same directory. It is recommended to save them to C:\Users\ < username > \MoveHelpr to ensure access to PATH variables.

Running the Program

To start the program, simply run the *driver* file. You will be prompted to enter a username to create a profile; enter your preferred username and the program will create a new user folder in the MoveHelpr directory. This user directory will store your item file as well as any item pictures that you associate with an item. Please do not delete these fles unless you want all your info to be deleted!

Features

Move Helpr helps you keep track of your items and boxes while you're moving. Accordingly, the program revolves around the creation and management of Item objects and Box objects.

Items

Item objects store all the information for unique items. You can store the following item information:

1. **Name**
   The unique name of the item. To create a new item, you MUST give the item a unique name. When choosing an item name, it's best to be as descriptive as possible: for example, instead of naming something "shirt", you should use a more descriptive name like "Old Navy Flannel Gingham Blue Green Button Up Shirt". This will also help when searching for items: you can simply enter the search term "shirt" and all items with "shirt" in their name will be shown.

2. **Value**
   (Optional) The value of a single item. The value must be at least 0.00. The value is not associated with a particular currency; currency support and conversion is planned to be supported in a future release. Additionally, the value is for each indidvidual item: if the item quantity is more than 1, just enter the value for a single item.

3. **Quantity**
   (Optional) The quantity of the unique item. For example, if you are entering silverware, you could have the item "Dinner Fork" assigned a quantity of 8 if you have 8 forks.

4. **Serial**
   (Optional) The item's serial number. This item attribute is optional, as not every item is serialized. However, it can help you keep track of serialized or high-value items to ensure that they weren't damaged and replaced or switched in transit.

5. **Picture**
   (Optional) A picture of the item. It's stored as the path to the picture file in the user's folder. Pictures can be used to help identify items during unpacking or support a damaged goods claim.

6. **Box**
   (Optional) The number of the box in which the item is stored. If the item is not in a box, it's box number will be listed as 0.

Boxes

Box objects contain and organise items. They also aggregate the value of all the items they contain. Boxes contain the following information:

1. **Number**
   The number of the box. It is used to select and identify different boxes, as well as associate items with a particular box. Box numbers are always a natural number.

2. **Room**
   (Optional) The room where the box was packed and should be unpacked. This helps to group like items together: for example, a box from the kitchen would contain items like silverware, small appliances, etc. while a bedroom box might contain clothing, sheets, clocks, etc. If no room is entered, it will default to "Room TBD".

3. **Items**
   (Optional) A list of the items stored in the box. Each item must be an Item object. Items can be added and removed from the box. The list can also be empty if the box does not currently have any items in it.

4. **Value**
   The total value of all the items in the box. It is calculated automatically from the values of the box's items.

Actions

The program allows for several actions to enable the creation and management of your moving inventory.

1. **Creation**
   You can create new items and boxes from within the program. When creating an item, you must enter the required parameters. See the documentation for default values.

2. **Editing**
   Item and box attributes can be edited from the user menu. This can be used to correct mistakes or add additional description for an item after it has already been created.

3. **Deletion**
   Items and boxes can be deleted. Be aware that once something is deleted, it's gone for good!

4. **Search**
   Search for items by box room or by the item name. Keywords are allowed: i.e. searching for "bedroom" will return all items in a box where "bedroom" is in the box room, searching for the item "shirt" will return all items that have "shirt" in the item name, etc.

5. **Create Documents**
   The program can generate documents from your items and boxes, such as inventory sheets for an individual box or a master list of all items for your move. QR codes can also be generated to tape/print onto boxes that contain the box information and item list. Currently, only QR codes are implemented, but a full document suite is planned for a future release.

Known Issues and Bugs

1. **Item Importation** - If more than one item is stored in the items .csv file, only the first item will be automatically added to the box. The rest will be added to the unboxed items list despite having an assigned box.

2. **Box Creation** - New boxes will copy all items from all other boxes to their item list. Likewise, trying to remove an item from a box will remove it from all boxes.

3. **Edit Boxes** - This feature is unimplemented due to the box creation bug.

4. **Change Item Name** - Changing the item name will not change the filename of the picture. However, the filepath will be the same and the picture will still open.

5. **Delete Item** - Deleting an item will not delete the associated picture.

6. **Saving Information** - Exiting the program will not save any changes made to the item .csv file. This will be implemented after fixing other bugs.

Planned Features

- **Currency Settings/Conversion**
  Set values in a particular currency or switch between currencies for international moves.

- **Moving Documents**
  Generate inventory sheets on a box or move scale. Integrate other applications such as DocuSign to generate boilerplate forms and agreements for moving/transportation companies.

- **Different User Types**
  Add a user type for moving/transportation companies to allow them to upload custom forms, track multiple client moves, and keep a master list of schedules.

- **Cloud Storage**
  Store user's data in a cloud location to enable access from any device, any location.

- **Hybrid Application Front-End**
  GUI-based interface running as a web application as well as inside a native application wrapper for mobile devices.