Ivan Alejandre
CST338
4/26/19

Final Project Submittal – An Android application for my job

The main purpose of this application is to help me keep track of my truck's inventory. However, in the context of this project, that would not have been enough. So I added another feature – creating a work order.

The unique aspect of this when you add parts used in your work order, it pulls it from the inventory class. It will also update the stock in the inventory so when I go back and search for part and see how much I have, the inventory activity will show me the stock.

In the work order activity, it pulls the address, any codes, and the problem from another container class called AddressLog. AddressLog is a predefined class for this project only. The future intent is to create a web applet that uses the methods in AddressLog to add new addresses for the day.

Back to the WorkOrders class, after you fill in what you did, and used whatever parts, it will calculate for you the subtotal and tax of any parts used and update the total.

On the bottom of the WorkOrders activity, there are two buttons. One is the job clock which collects timestamps on when I start and when I end the job. When I stop the clock, it automatically fills out the labor charge and updates the total.

The complete job button takes what you have filled in the page and saves it all to a .txt file on the phone. For this project, that is all it does. In the future, instead of saving, it will export to a SQL database for the office ladies to a separate application to the work order data into an invoice.

Specs: see attached UML for attributes and methods
MainActivity:
Only two methods which lead to either Inventory or WorkOrders

Inventory:
the main class is a controller&view hybrid akin to the MVC model. Inventory is responsible for the creation of the event handlers for the activity and retrives or changes the information in a related class – InventoryData.

InventoryData:
contains all the getters and setters for the internal variables. All variables bar one are static.

WorkOrders:
Similar to Inventory, this class implements the event handlers for the activity and is responsible for all the information ferrying. WorkOrderData is a related container class which holds the information for *one* work order. WorkOrders contains an array of WorkOrderData classes, one for each work order.

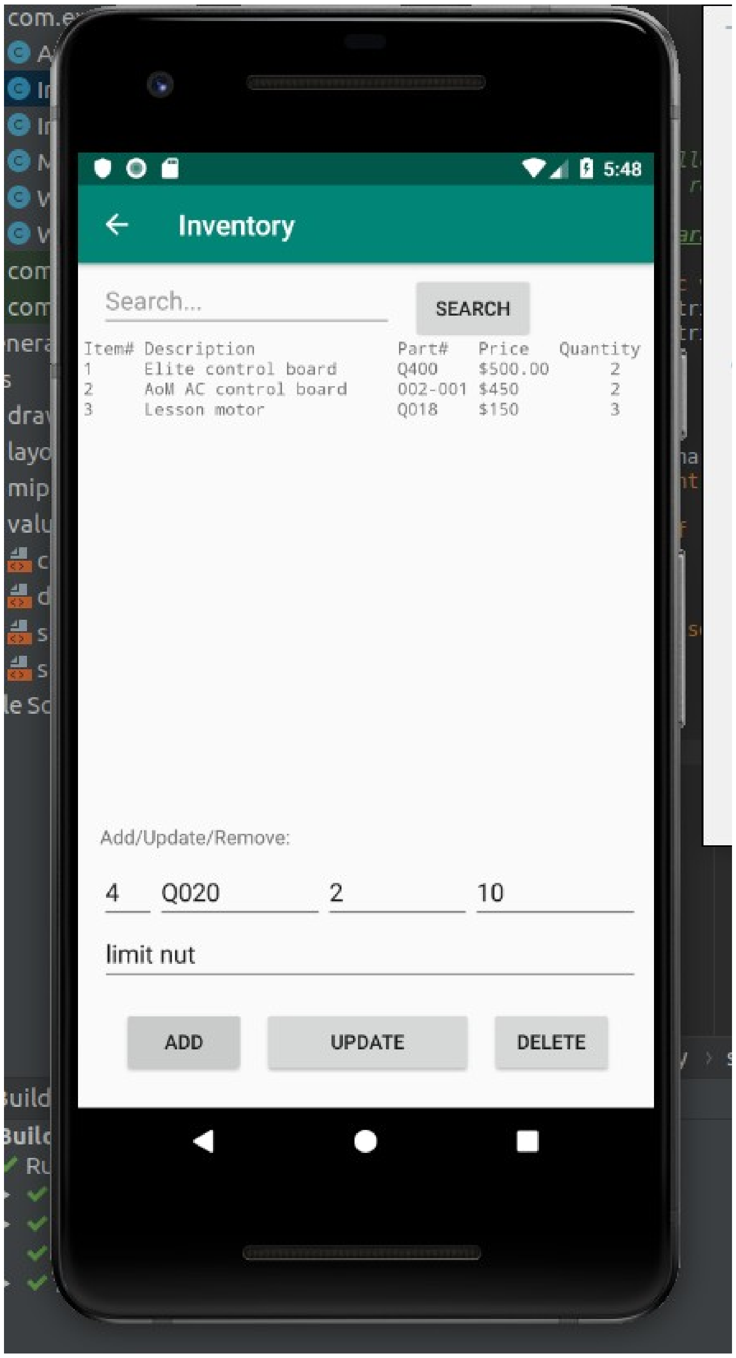
WorkOrderData:
contains all the related information to construct the work order

AddressLog:
a predefined class for the purpose of this project. It contains hardcoded address information. In future iterations of this application, there will be mutators in the class to change the internal variables.

Proof of function:

It’s a bit difficult to show what the application is capable of at this point in screen shots. However, I can show a bit

In the Inventory section, we can see I have three items already added. You can also see information already loaded, ready to be saved. This is information I inputted when I took the screenshot.



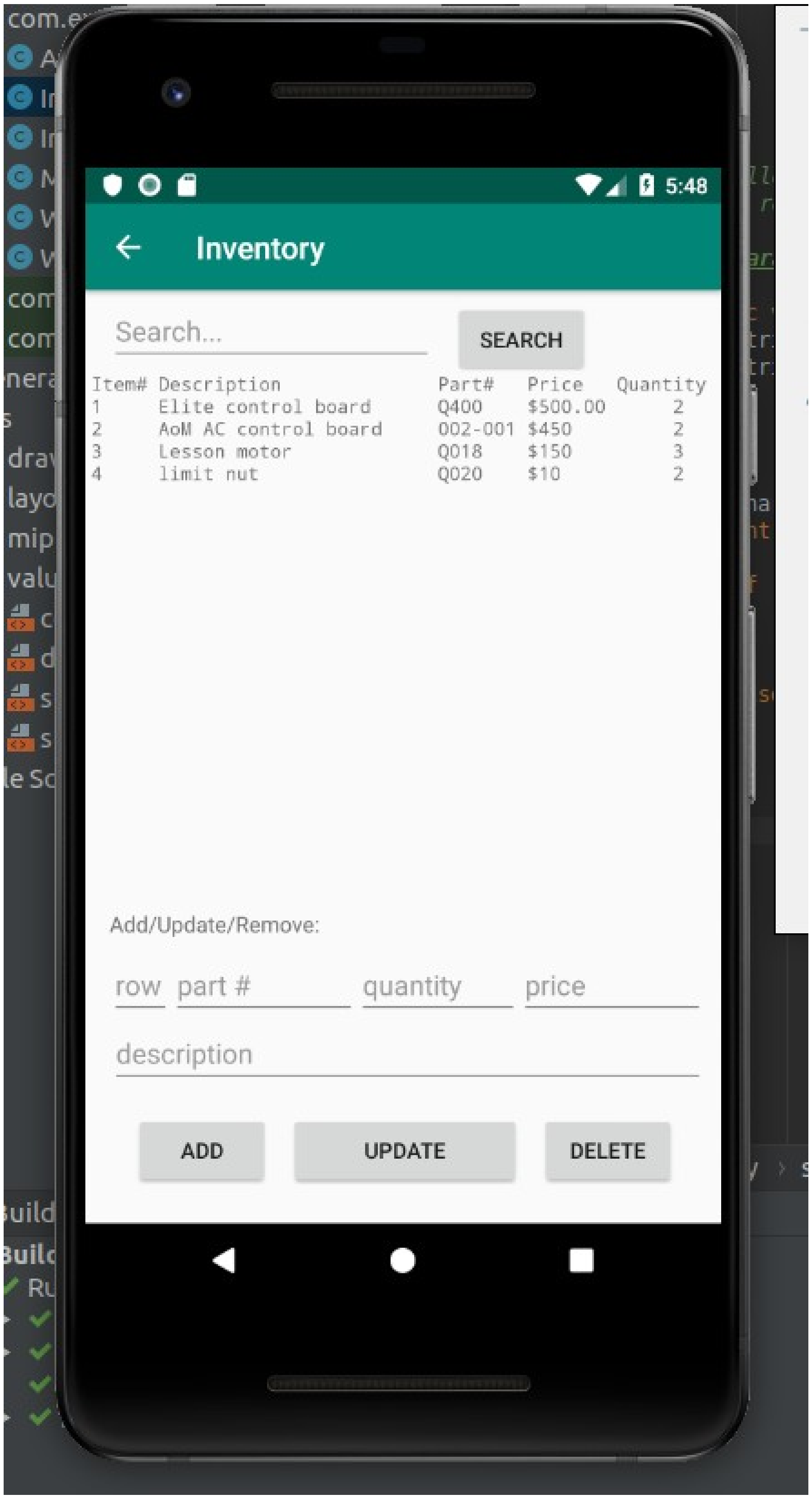
```
Item# Description          Part#   Price    Quantity
1     Elite control board  Q400    $500.00     2
2     AoM AC control board 002-001 $450        2
3     Lesson motor         Q018    $150        3
```

Add/Update/Remove:

4 Q020 2 10

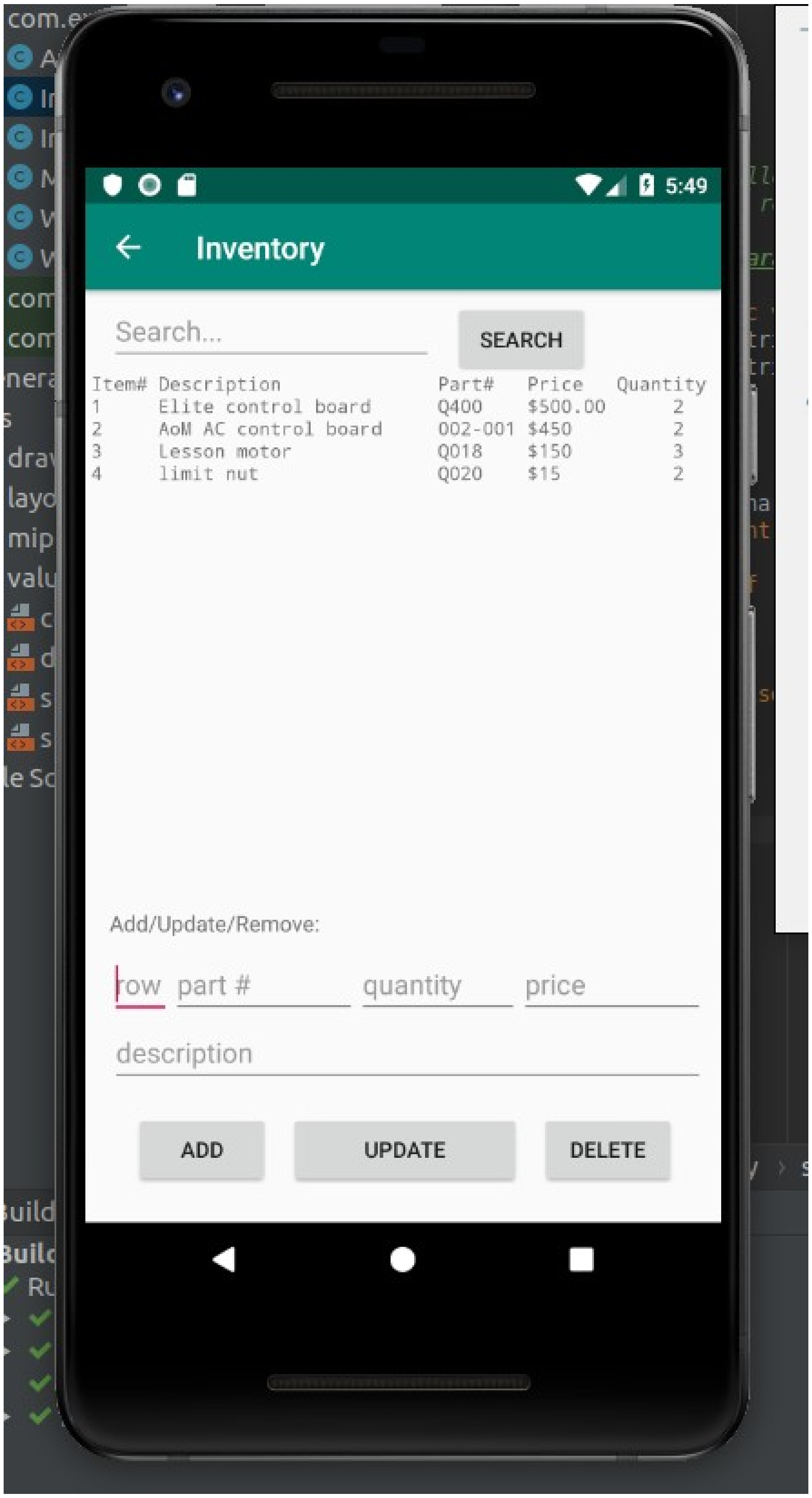
limit nut

After clicking on the ‘Add’ button, this is the change. You can see that the item has been added and the input fields have been cleared.



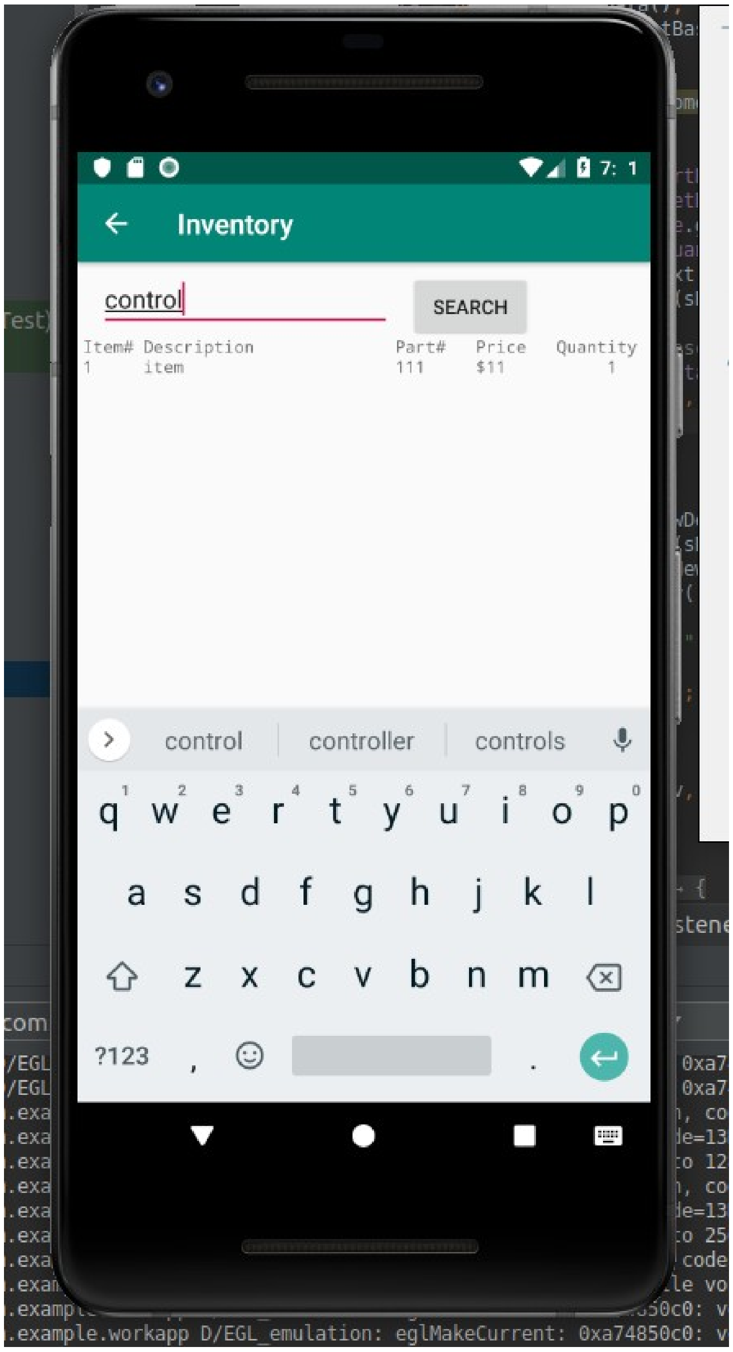
```
Item# Description           Part#   Price    Quantity
1     Elite control board   Q400    $500.00     2
2     AoM AC control board  002-001 $450        2
3     Lesson motor          Q018    $150        3
4     limit nut             Q020    $10         2
```

To update, you must include the row number. This is the identifier for the item. You must also include at least one field. The update button is not limited to changing one field. It can change all at once. Note in the following screenshot the price has changed for item #4.

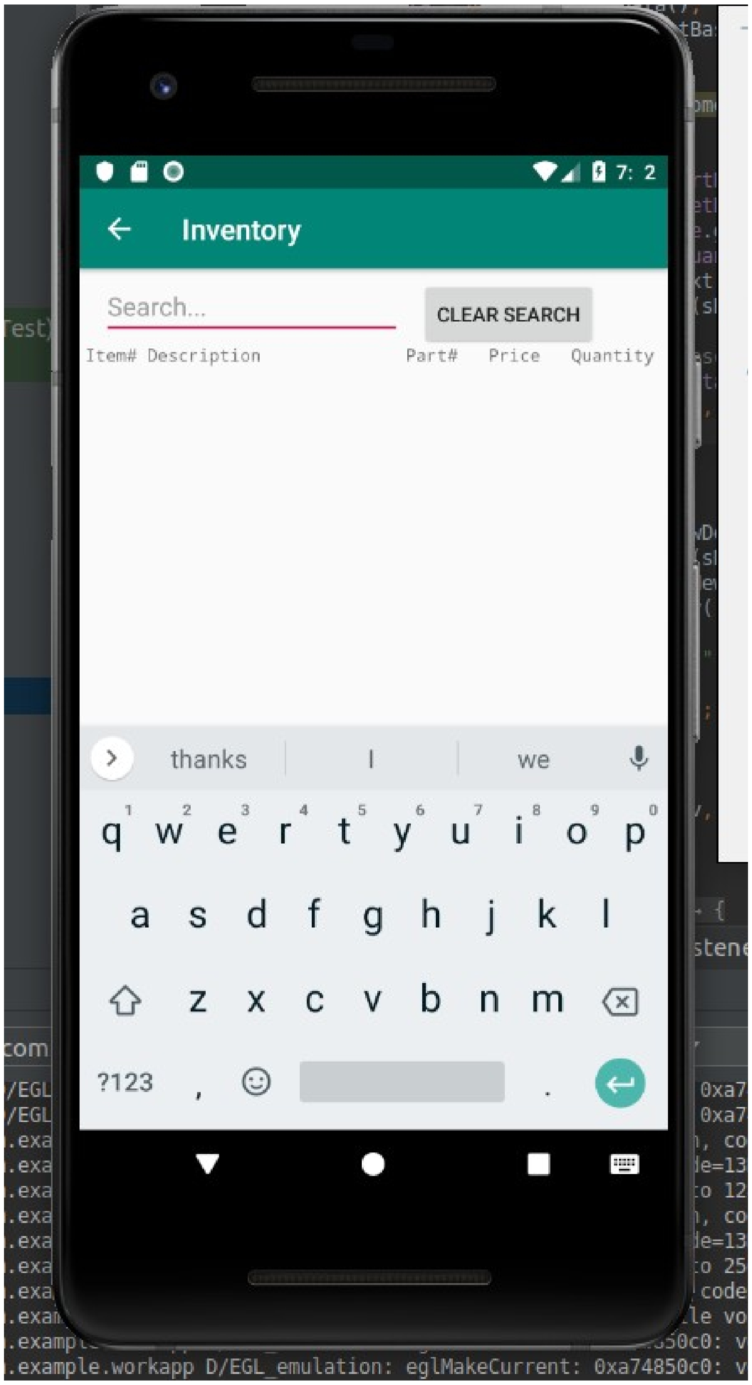


```
Item# Description          Part#    Price    Quantity
1     Elite control board  Q400     $500.00      2
2     AoM AC control board 002-001  $450         2
3     Lesson motor         Q018     $150         3
4     limit nut            Q020     $15          2
```

At the top of the page, there is a search function. You can type in any string and the application will show all items that have your inputted string as a substring. The purpose of this is to search for items when the list is well over a hundred items long. For this project, the maximum amount of items is 50. On the left is the capture before the search. On the right is after. Note that item has been removed because it does not contain control. Pushing 'clear search' prints the whole list again.

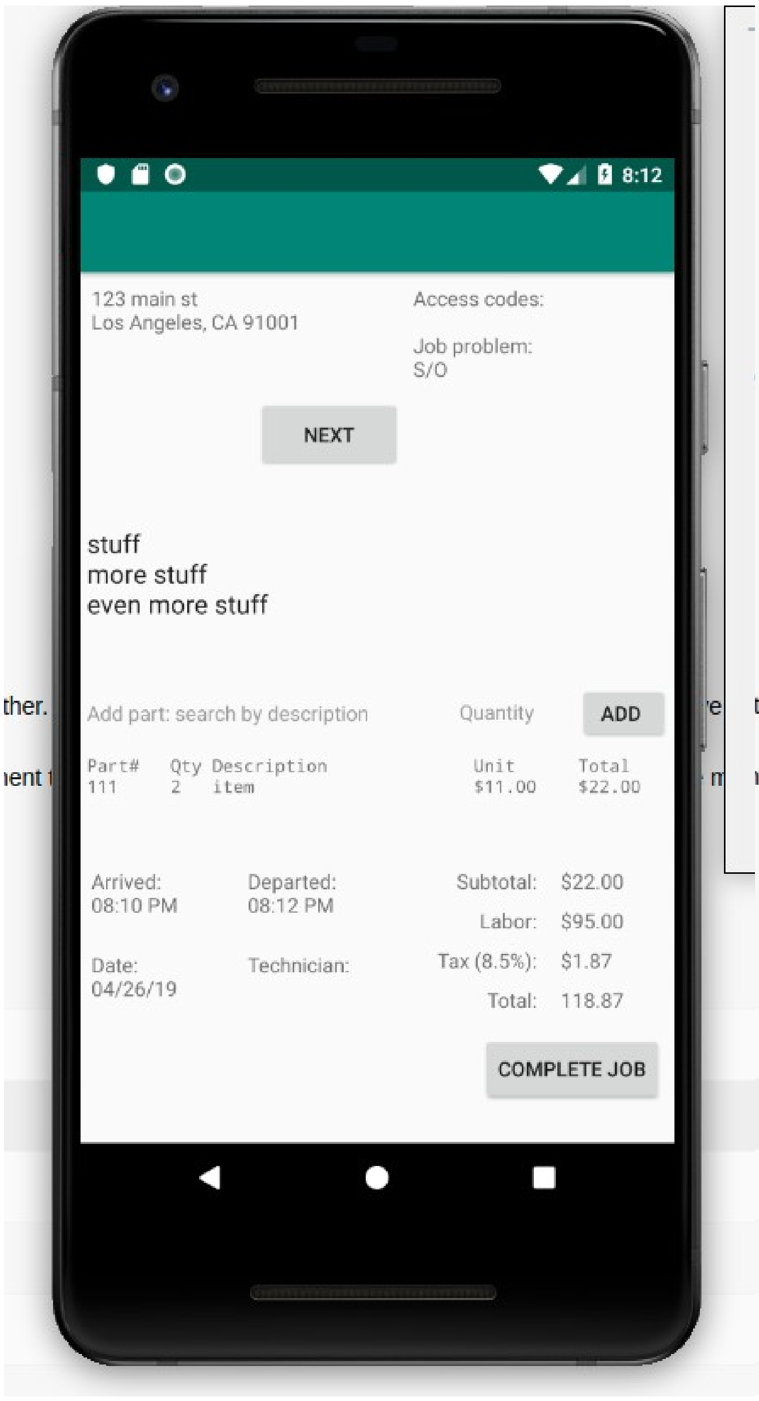


```
Item# Description          Part#   Price   Quantity
1      item                111     $11         1
```



```
Item# Description          Part#   Price   Quantity
```

Moving on to the work order section, when you open the activity, the array of work orders are initialized. Each work order will contain an address, any access codes, and the problem description upon creation. It will be up to the user to fill in the work performed, any parts used, start the time when the begin work and when they stop work. When a user enters this activity for the first time, the previous button will be invisible. Upon pushing ‘next’, it will appear and remain visible unless you are at the first work order. Similar function exists for ‘next’. It will remain visible all the way to the first empty WO. I left this intentionally as a way to signal the end of available calls. The two screenshots below show a completely filled work order and an empty one.



123 main st
Los Angeles, CA 91001

Access codes:

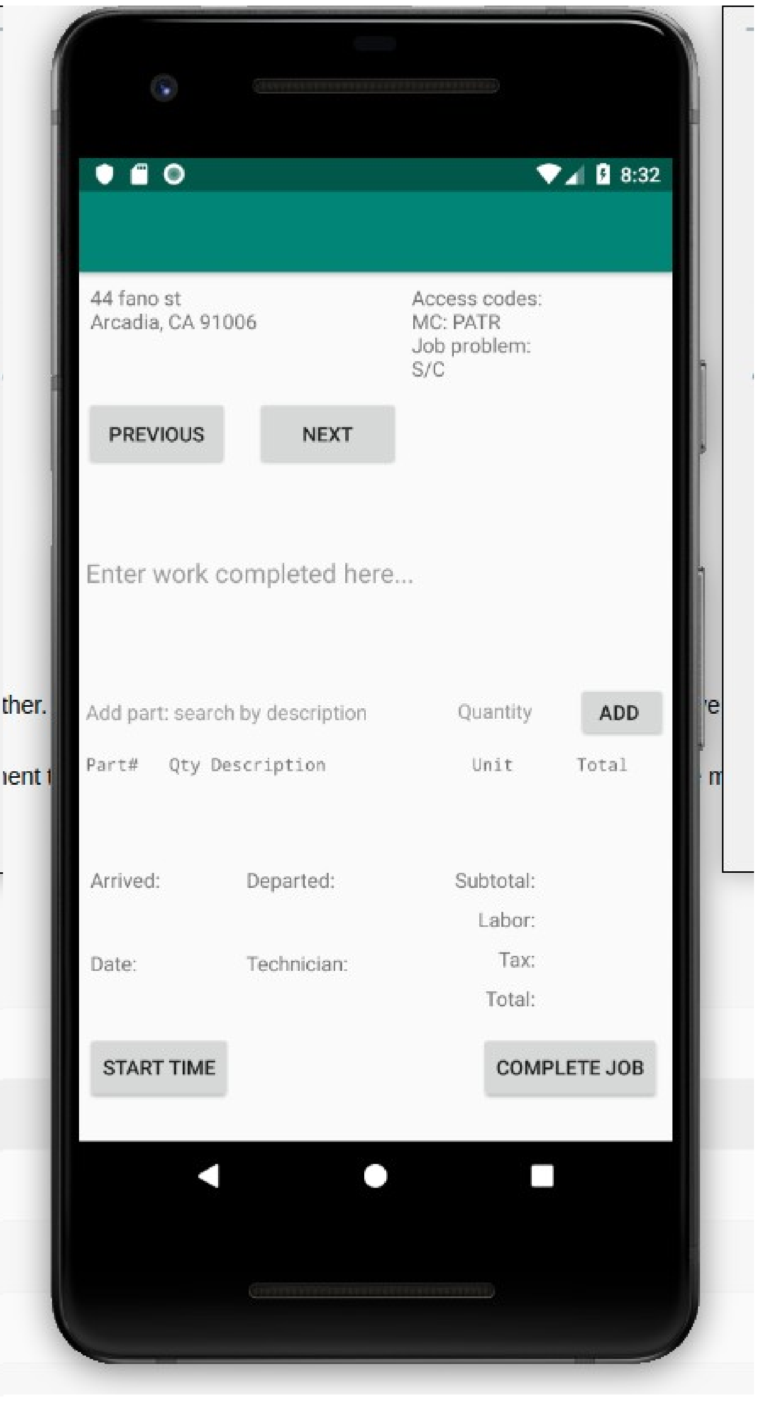
Job problem:
S/O

stuff
more stuff
even more stuff

```
Part#   Qty Description        Unit     Total
111     2   item               $11.00   $22.00
```

Arrived: 08:10 PM
Departed: 08:12 PM
Date: 04/26/19
Technician:

Subtotal: $22.00
Labor: $95.00
Tax (8.5%): $1.87
Total: 118.87



44 fano st
Arcadia, CA 91006

Access codes:
MC: PATR
Job problem:
S/C

```
Part#   Qty Description        Unit     Total
```

Arrived:
Departed:
Date:
Technician:

Subtotal:
Labor:
Tax:
Total:

When a user cycles through the work orders, it was intention to have any information filled in saved in a file. However, things get tricky when you’re trying to save an array of custom objects. So that feature will be developed later. In addition, when an user is entering parts, it is supposed to autosuggest parts, but I also did not get a working solution for the WorkOrders class to access the information saved by the Inventory class. For this project, any information saved will only persist when the application activity is active. It is not saved onPause and not retrieved onResume. In addition, any user will have to type in the full description to get the price information to show up in the work order page.

The two last buttons left unexplained are the start time button and the complete job button. The start time button is otherwise known as the job clock. When a user pushed that button, it grabs the current time and date and fills the respective fields. When the user is done with the work, they push the button again which has been renamed ‘stop time’, and it grabs just the current time. Then the labor field is computed and filled and the total updated. When a user cycles through work orders, each work order has a boolean variable to remember if time has been completed or not.

Complete job is rather simple for this project, even though it is not the final iteration of this method. When a user clicks on ‘complete job’, the application currently saves all data in the work order to a text file. In the future, instead of saving this information, it’ll be sent to a server to be stored in a SQL database.

Conclusion:

Play around with the application. Add some items of your own, and use them in the work order pages. You have three work orders to augment. When cycling through, all the information for each work order should be there.

I hope you enjoy this application. It has a lot of work still, but I have done a great amount in the 15 hours I spent writing this application.