

## CSC 668 - POST GUI - Assignment 2

**Group members:** Mahdi (Matt) Massoodi, Pablo Escriva, Bijan Mahdavi, Marcus Wong, David Lopez, Izaac Garcilazo, Gary Straub

**GitHub repository:** <https://github.com/sfsu-csc-668-spring-2019/post-assignment-new-group-6>

### Overview/Description

POST is a point of sale terminal in which customers can add store products using the GUI while being able to checkout and pay. While creating a new sale, managers are able to obtain a sales log for all the purchases made.

### Discussion/Difficulties

We decided to use microservices to connect the post-server API to our POST system. We use these microservices to generate instances from within our system classes to better abstract the logic between API requests/responses and how the actual POST system works. The services are only implemented to communicate with the API. Once the system and services were built, we were able to wire up the GUI and provide the respective data accordingly. Some difficulties included issues in how much work the services needed to do with GSON. However, we ended up putting the GSON logic into the system to keep the microservices clean. Another implementation concern was wiring up the GUI with the backend system. Mainly, in how we should implement the logic in when the GUI starts and ultimately how the manager fires up the store.

### Class Design/Implementation

[UML diagram link](#)

(Screenshot in last page of the document)

In the UML diagram linked above, you can see three different packages; system, services, gui. Each one defines a specific part of our POST system. The system package contains functionality for running the GUI and handling data being passed between the services and gui, and vice versa. The services package is strictly for handling requests and responses between the API POST server. This is convenient for the system package because the abstraction cleans the functionality all together. The GUI uses the data being fetched from the system called on service methods. While on the other hand the GUI creates new records being put from the system called on their respective service methods. Our system package includes Manager (main), Store, Sale, SaleItem, Payment, and Item. These classes work throughout the entire app but mainly the GUI in bridging the gap between the services and the views. Our services package includes PaymentService, ProductService, and SaleService. Along with respective GET, POST, PUT, Request, and Response classes to communicate with the post-server API. The GUI class consists of the PostView, ProductsPanel, PaymentPanel, InvoicePanel, and CustomerPanel. These classes are used to abstract the end-user(s) input and interaction with the GUI. This allows our system to pass this data to the services for a more linear flow.

## Output/Tests

## Test 1: Cash payment

Successfully creates a JSON object with the specified format and sends it to the server which responds with a 201 indicating the sale went through.

[illegible]

```
1. npm start (node)
→ API git:(master) ✖ npm start

> post-server@0.0.0 start /Users/pablo/git/API
> node ./bin/www

PUT /sales 201 45.406 ms - 9
█
```

## Test 2: Credit Card payment

Prompts user to insert a credit card number and sends payment confirmation request. Then sends JSON object to add sale to the database

[illegible]

```
1. npm start (node)
→ API git:(master) ✕ npm start

> post-server@0.0.0 start /Users/pablo/git/API
> node ./bin/www

POST /payments/credit 202 17.252 ms - -
PUT /sales 201 33.631 ms - 9
□
```

### Test 3: Check payment

[illegible]

```
1. npm start (node)
→ API git:(master) ✖ npm start

> post-server@0.0.0 start /Users/pablo/git/API
> node ./bin/www

POST /payments/check 202 23.087 ms - -
PUT /sales 201 41.819 ms - 9
```

## Test 4: Closing POST

When user closes the POST window all the sales log is printed through console.

```

● ● ● 2. pablo@Pablos-MacBook-Pro: ~/git/post-assignment...

Customer1      1550721077883
Paper Towel Touchless 1@63.2 - 63.2>
The Pop Shoppe Pinapple 2@20.13 - 40.26>
Loquat 2@70.61 - 141.22>
The Pop Shoppe Pinapple 5@20.13 - 100.649994>
-----
Total $345.33
Amount Tendered: 400.0
Amount Returned: 54.670013

Customer2      1550721103554
Soup - Campbells, Chix Gumbo 3@24.83 - 74.49>
Mousse - Banana Chocolate 2@73.91 - 147.82>
Coffee - Cafe Moreno 2@72.74 - 145.48>
-----
Total $367.78998
Paid by Credit Card 123456789

Customer3      1550721122482
Soup - Beef, Base Mix 5@17.59 - 87.95>
Coffee - Cafe Moreno 5@72.74 - 363.69998>
Loquat 2@70.61 - 141.22>
-----
Total $592.87
Paid by Check

➔ post-assignment-new-group-6 git:(master) ✕ |
```

### Other tests:

1. User inputs non-numeric characters in amount field → Sale is not created and UI contents are refreshed.
2. User inputs non-numeric characters in credit card number field → Sale is not created and UI contents are refreshed.
3. User leaves customer name field empty → Pay button has no effect.
4. User leaves amount field empty → Pay button has no effect.

