**Project Plan Cheat sheet:**

***\*This cheat sheet is working on planning we have done for application and thought process during programming.***

***Please input updates based on changes in application.***

Browser Application:

**Seller:**

**Signup & Login**:

* Login credentials of seller on browser and get store in db (use of cookies and uuid to authenticate seller, html templates)

**Itemsdetails table:**

* Includes item info which seller can CRUD through **json or html** templates and get stored in db.
* Username is foreign key in itemsdetails table to track items respective to seller.

**Buyer:**

**Signup & Login:**

* Need to use separate table in db where buyer redirected to search for available items.- **to be updated once seller side ready**
* Search items name or seller username(Linkedlist) – **working on it currently**
* Buyer CRUD by json or templates **TBC**

**-------------------------------------------------------------------------------**

**Notes: By Pallavi**

**Functions to talk to Seller JSON API for “buyer” only:**

//Data structure for each item for JSON

type ItemsDetails struct {

Item string `json:"Item"`

Quantity int `json:"Quantity"`

Cost float64 `json:"Cost"`

Username string `json:"Username"`

}

false in the argument means its for buyer, true means for seller. So for buyer must use false

\***by Nitu**: instead of boolean we can use flag with name **buyer** and **seller** to keep clear segregation but for now we can keep it

\*by Pallavi: This flag is temporary, later need to get this info from the session itself, so that code or any malicious user does not try to hack and get info that the user is not supposed to see. Also its always good to use boolean when only two values are there, so that no other options are possible

1) View All Items of all sellers, request made by a buyer : getItem("", "", false)

returns []ItemsDetails, bool

[]ItemsDetails - array containing all items for all seller [FOR BUYER]

true- when getItem is successful - need to use this to display items to template

false - when getItem is not successful - can use this to create a message to user in template

2) View A Particular Item, request made by a buyer : getItem(itemname,sellername, false)

returns []ItemsDetails, bool

[]ItemsDetails - array containing only one item for the specified seller [FOR BUYER]

true- when getItem is successful - need to use this to display items to template

false - when getItem is not successful - can use this to create a message to user in template

3) Add An item to a seller, request made by a buyer : This will not be allowed

returns bool

Always returns false for buyer

For SELLER:[

true - add is successful

false - add is not successful

can use to show status of add to template ]

4) Update An item of a seller, request made by a buyer : updateItem(itemname, sellername, false, ItemsDetails)

returns bool

true - update is successful

false - update is not successful

can use to show status of add to template

5) Delete An item of a seller, request made by a buyer : deleteItem(itemname, sellername, false)

returns bool

true - update is successful

false - update is not successful

can use to show status of add to template

\***Nitu** : points 3 4 and 5 shouldn't be allowed for buyer instead buyer only can see seller items and can add, update, delete in its own cart but can not touch items displayed by seller.

\*Pallavi : 3 Add is not possible as buyer will not be adding any item to sellers database.

But 4 is to update, that means if buyer buys say 4 out of 10 items, an update is needed to change the number of items to 6. cannot change seller name, item name and cost, that will be checked in code for buyer. but buyer still needs to put in the request so that api knows what is to be updated. But best to have check all entries in buyer code as well, otherwise the request will not be successful, that will lead to problems!

Point 5 also is needed, as when buyer buys all items of a particular seller, that item needs to be deleted. delete is better as it will clean up the database instead of keeping an item quantity to be 0

\***Nitu**:Little clear but I need to see how your code is running to get actual idea in case of above 1-5 points

Additional points to note:(\*Nitu:Agree on below 1-3 points)

1. Any change in quantity must be made in buyer.go code, that means need to request info for an item first, then looking at quantity of that item in shopping cart, quantity needs to be subtracted and then using update function need to update through seller API.

2. Quantity can be 0, cost can be 0.0

3. If quantity is 0, best to use delete instead of update.

4. Already implemented the above functions for talking to API, will upload tomorrow after final testing.

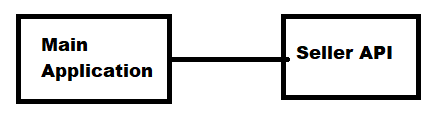
**Functions to talk to Seller JSON API for “seller” only:**

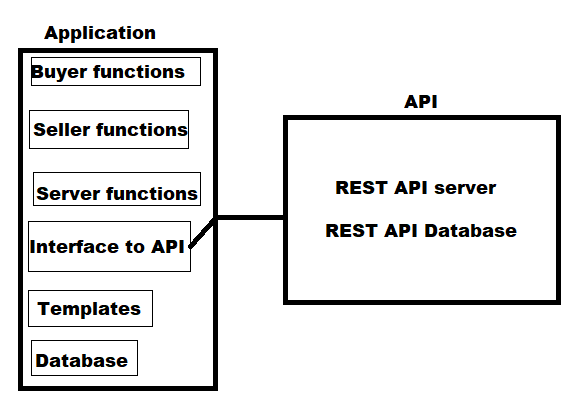
Similar functions with true in argument, already implemented, need testing

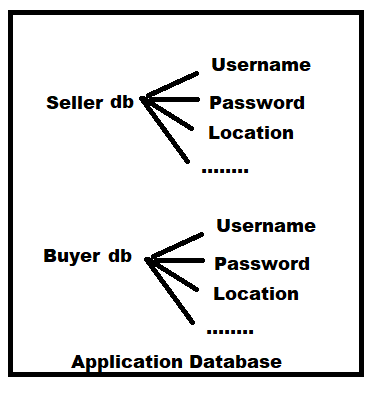
**Other notes:**

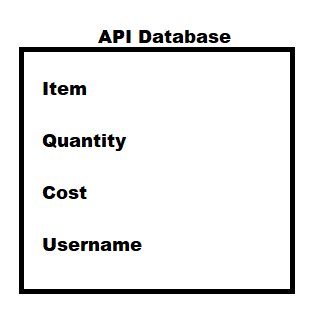
* **Items db successfully implemented in Seller API(\*need to see running to get clear picture)**
* **Db for application needs to contain only username and password, separate tables for buyer and seller to be implemented, as better to segregate the two(\*Nitu:Agree there should be separate login functionalities for buyer and seller )**

**overview of structure: (\*Nitu:Still little unclear about some features)**

****

****

****

****

**-------------------------------------------------------------------------------**

Rest API

TBC