PROJECT INCREMENT 4

PG4

SAI TEJA SARANAM
SRIKANTH NADENDLA
NAGENDER GOUD
RAM NIKHIL

Objectives:

The objectives of this iteration include implementing admin services, integration of - Login service, customer services, employee services, admin services and fixing minor bugs from the previous iteration. Admin services include, viewing all the items (present in the database), adding new items to the list of items (in database), updating items and monitoring open orders (placed by the customers). Other objectives include implementing REST services for performing above tasks (Implemented 2 new services, modified one old service and used one old service).

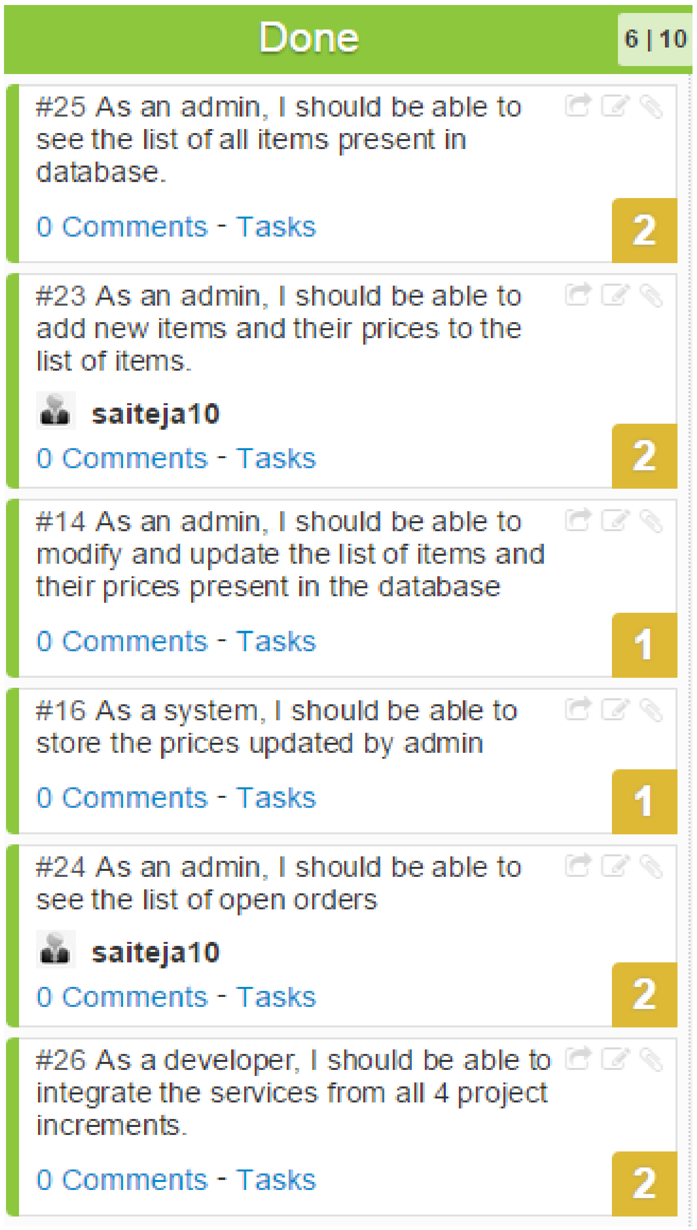
Existing Services/API:

For the frontend except for one (for filtering items by name) we haven't used any existing services. But for the development of mobile client we referred the following links and example codes to know how to consume REST services in client application, parsing JSON objects and few small concepts like array adapter, list view, toasts etc.
http://www.tutorialspoint.com/android/android_json_parser.htm
http://www.androidhive.info/2012/01/android-json-parsing-tutorial/
http://developer.android.com/guide/topics/ui/layout/listview.html
http://developer.android.com/reference/android/widget/ArrayAdapter.html
http://developer.android.com/guide/topics/ui/notifiers/toasts.html
For the backend, previously implemented RESTful services are extended with few more services in this iteration. For testing and rapid development we used **Jetty server plugin .** It will scan our project periodically for changes and redeploy the web application if any changes are found automatically.

Detail design of Services:

As a recap, in the previous iterations we implemented Login service, customer services, and employee services. In customer services, upon successful login to application a customer can see list items of different categories like fruits, vegetables, fast food restaurants, Restaurants, add items to cart, place an order. In Employee services, upon successful login an employee should see list of open orders placed by the customers, Accepts an order from the list of open orders. In this iteration we implemented admin services, integration of - Login service, customer services, employee services, admin services and fixing minor bugs from the previous iteration. Admin services include, viewing all the items (present in the database), adding new items to the list of items (in database), updating items and monitoring open orders (placed by the customers). Also we extended RESTful services for performing above tasks (Implemented 2 new services, modified one old service and used one old service).

User stories for Admin services and developer:



Done 6 | 10

#25 As an admin, I should be able to see the list of all items present in database.
0 Comments - Tasks 2

#23 As an admin, I should be able to add new items and their prices to the list of items.
saiteja10
0 Comments - Tasks 2

#14 As an admin, I should be able to modify and update the list of items and their prices present in the database
0 Comments - Tasks 1

#16 As a system, I should be able to store the prices updated by admin
0 Comments - Tasks 1

#24 As an admin, I should be able to see the list of open orders
saiteja10
0 Comments - Tasks 2

#26 As a developer, I should be able to integrate the services from all 4 project increments.
0 Comments - Tasks 2

Services:

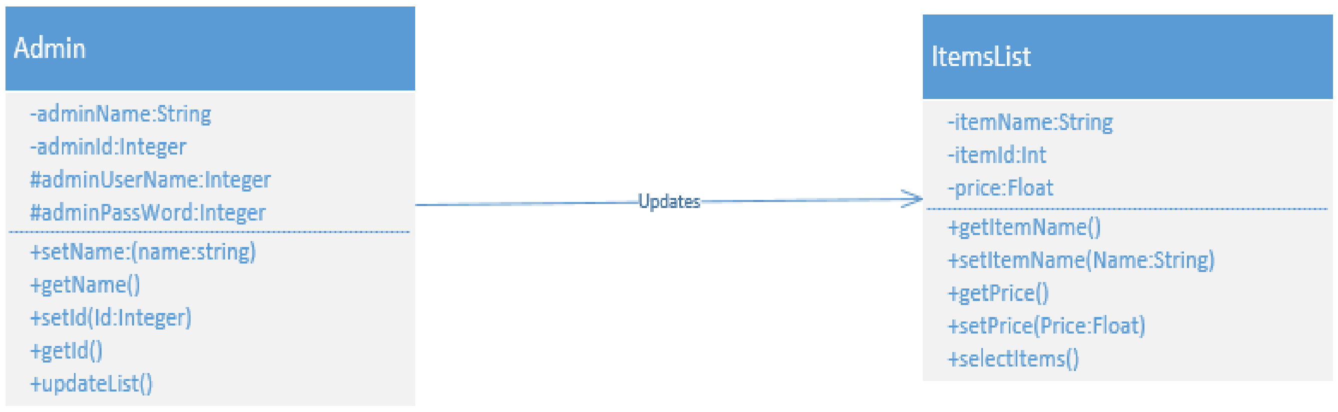
We have developed the registration, login service in first increment, for second increment we developed the customer service where customer will be able to select items and add them to cart, for Employee service, an employee upon successful login can see list of open orders placed by customers and accepting orders to process them. In this iteration apart from adding services to REST, we implemented Admin services.

Admin service:

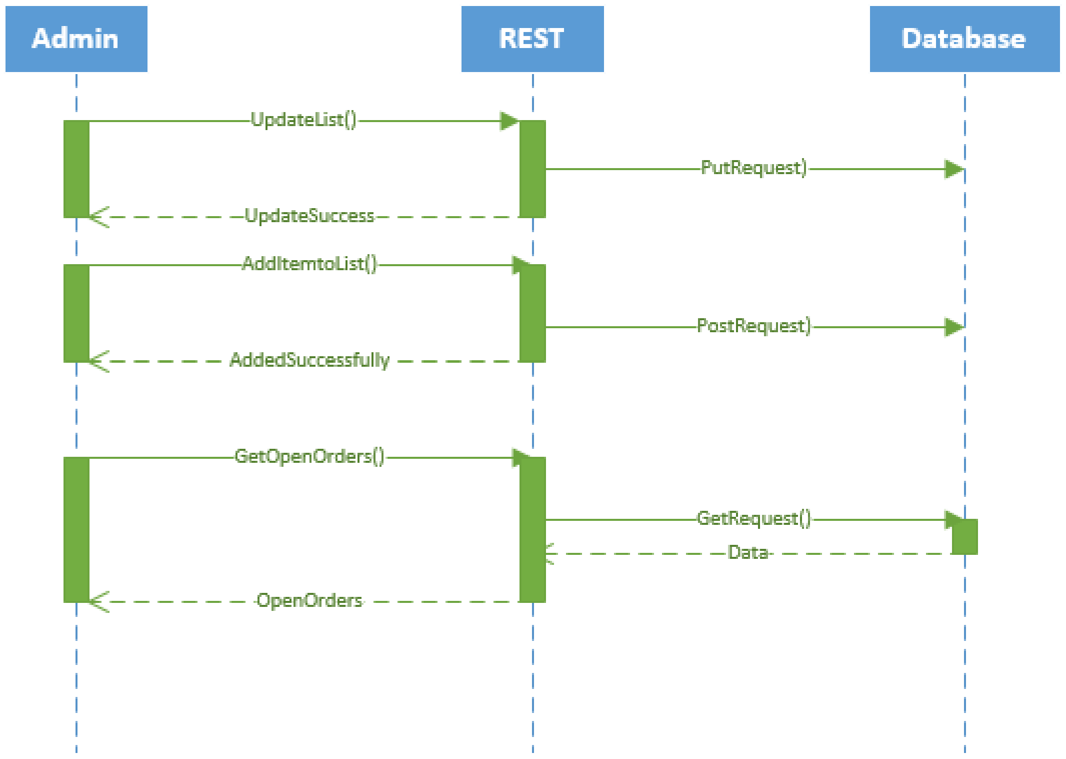
Service description:

Admin services include, viewing all the items (present in the database), adding new items to the list of items (in database), updating items and monitoring open orders (placed by the customers).

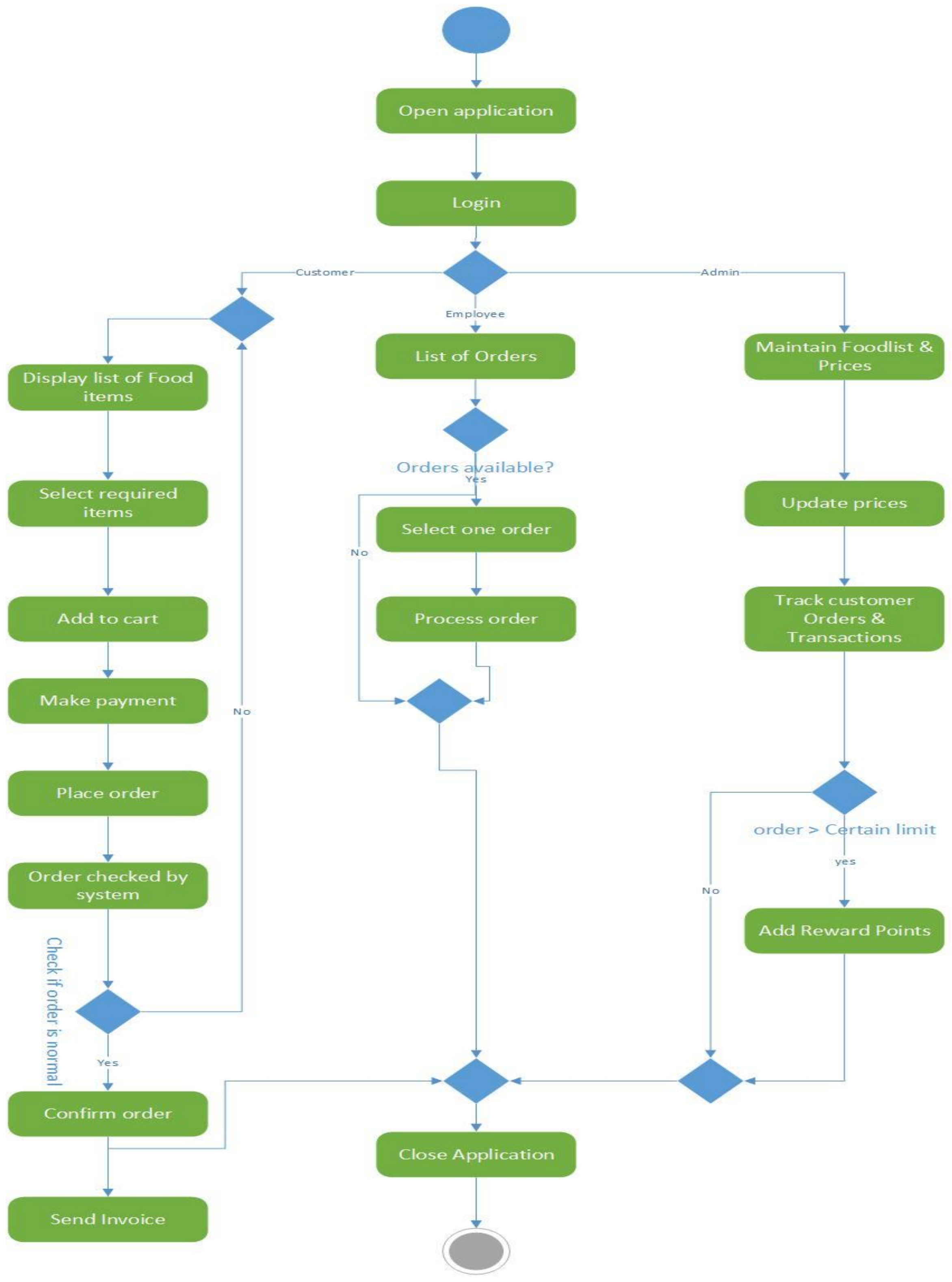
Class diagram:



Sequence diagram:



Activity Diagram:



<u>Implementation:</u>

<u>*Implementation of REST services:*</u>

For this iteration, we Implemented 2 new services, modified one old service and used one old service.

Two new services include modifying existing items and adding new items.

```
@Path("/item")
@POST
@ElementClass(request = ItemBean.class, response = BaseBean.class)
public BaseBean updateItem(@QueryParam("") ItemBean itemBean) throws Exception;

@Path("/item")
@PUT
@ElementClass(request = ItemBean.class, response = BaseBean.class)
public BaseBean addItem(@QueryParam("") ItemBean itemBean);
```

Used one old service to retrieve list of items.

```
@Path("/items")
@GET
@ElementClass(response = ItemBeanList.class)
public ItemBeanList getAllItems();
```

Modified one old service for admin to review the open orders.

```
@GET
@Path("/open")
@ElementClass(response = SaleOrderBeanList.class)
public SaleOrderBeanList getOpenedOrders();
```

We implemented the REST services using Apache CXF. It is open source and multi featured web services framework. It takes care of all the http requests. Every request from the mobile client goes to CXF servlet.

The REST service we developed provides all the data and information about prices and description of fruits, vegetables, fast foods, restaurant menu etc.

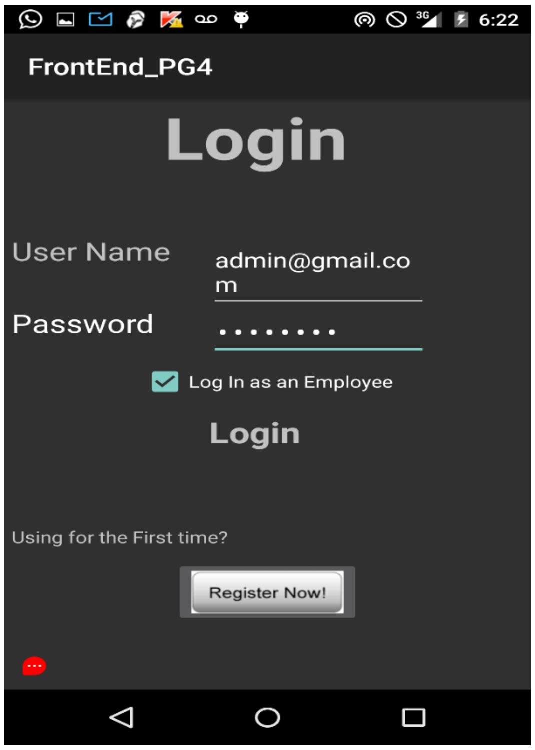
We used MySQL as database for our project. We developed and created REST services using Apache CXF and Spring dependency injection. For mapping Java classes to database tables, we used Hibernate.

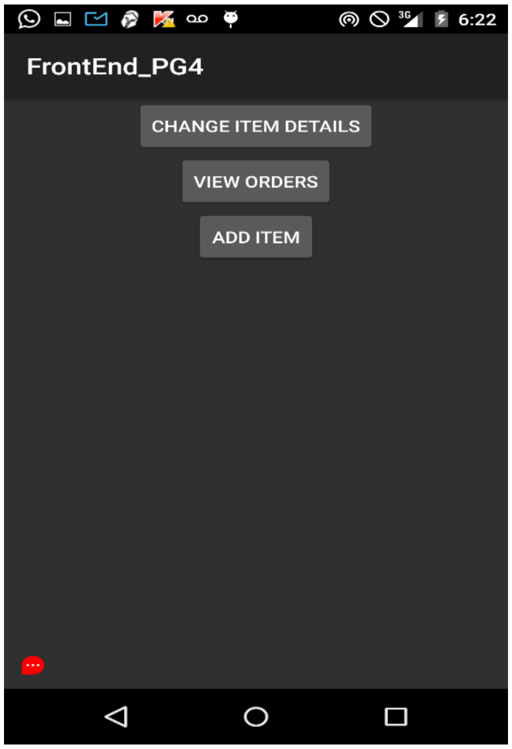
A detailed info about the implementation of REST services will be discussed later in the testing part.

Implementation of user interface:

For this iteration we have developed the activity page for Admin services, activity for admin to see the list of items, activity page to modify the item details, activity for adding new items to the list, activity page for reviewing list of open orders.

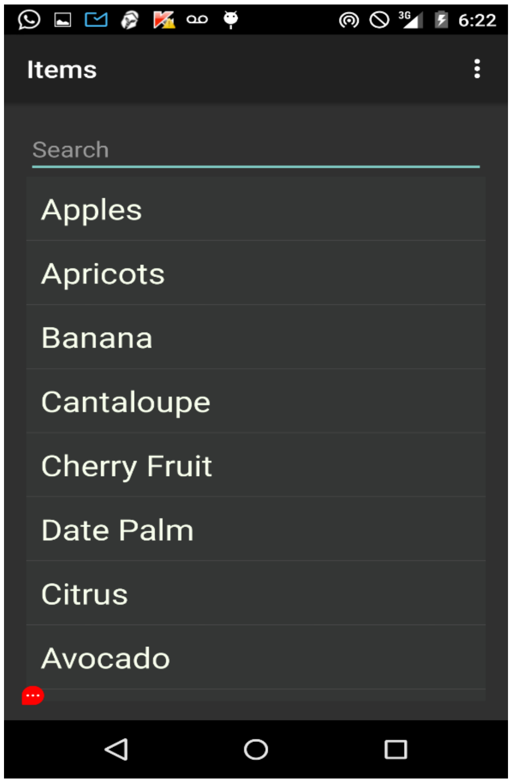
When an admin login to the system successfully, he can see list of all services available.





Upon selecting "Change item details" service, an activity page appears which shows list of all items. Upon selecting any item, an activity page will appear where we can edit the item details and update.

Shown below are the corresponding pages for what I described above.


6:22

Items

Search

Apples

Apricots

Banana

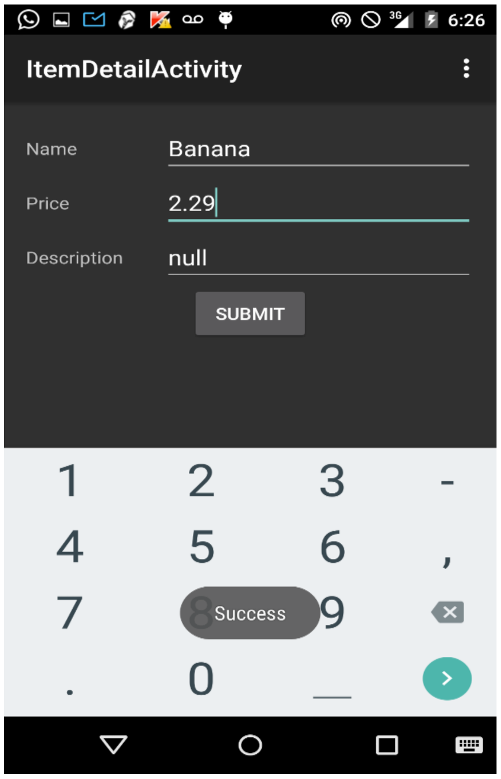
Cantaloupe

Cherry Fruit

Date Palm

Citrus

Avocado


6:26

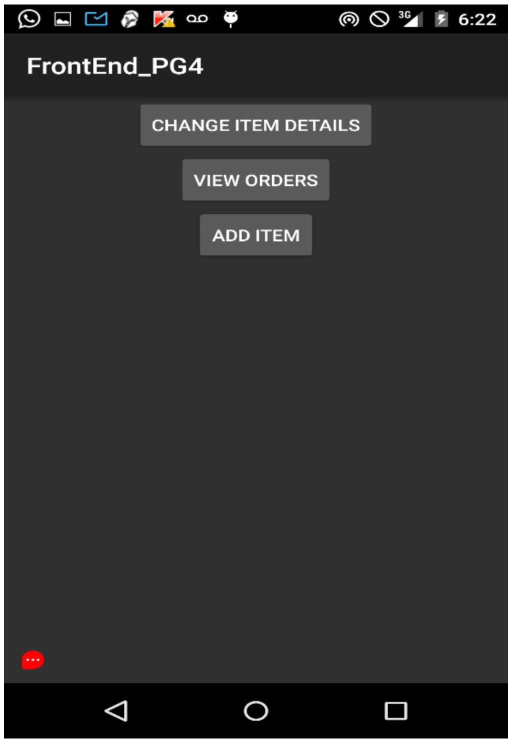
ItemDetailActivity

Name	Banana
Price	2.29
Description	null

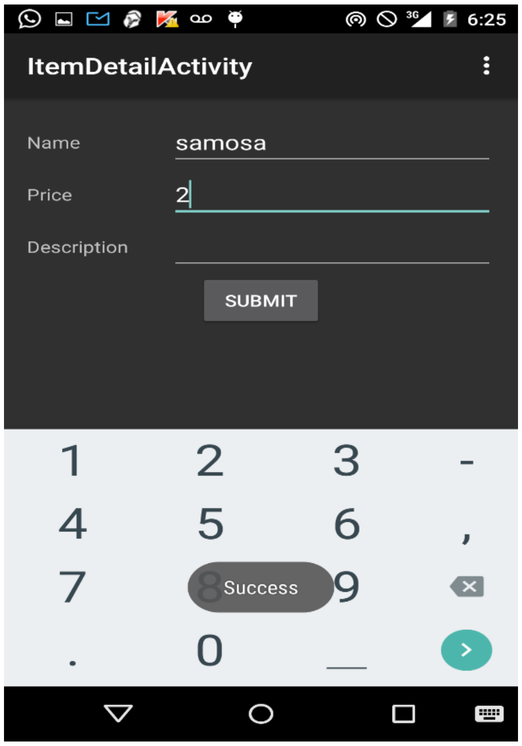
SUBMIT

Success

Another service from the admin services page is Add Item. The following activities shows adding an item:



Upon selecting "ADD ITEM":

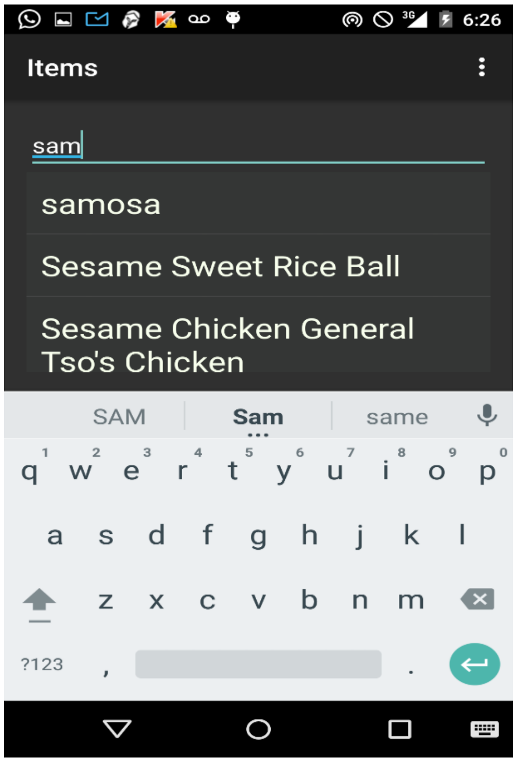


Name samosa

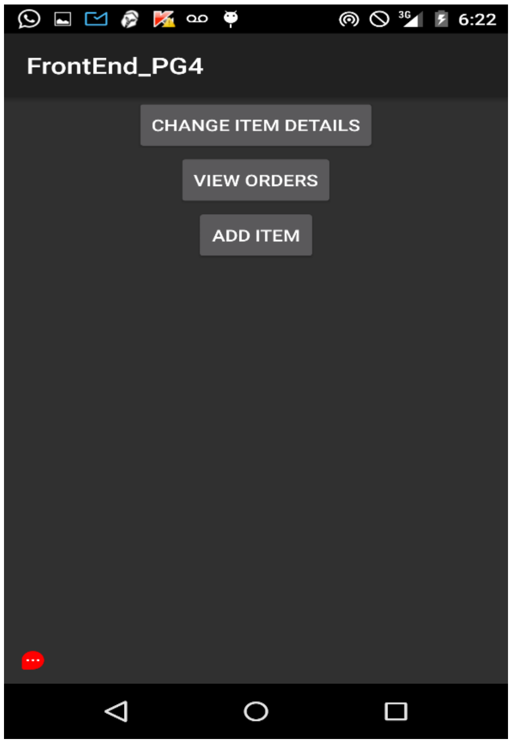
Price 2

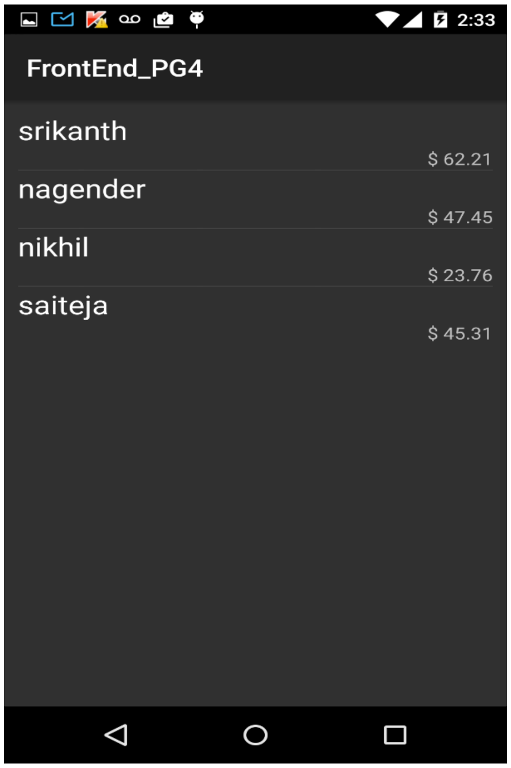
Description

Following activity is show the item I added above in the list of items and also to demonstrate the item filtering service:



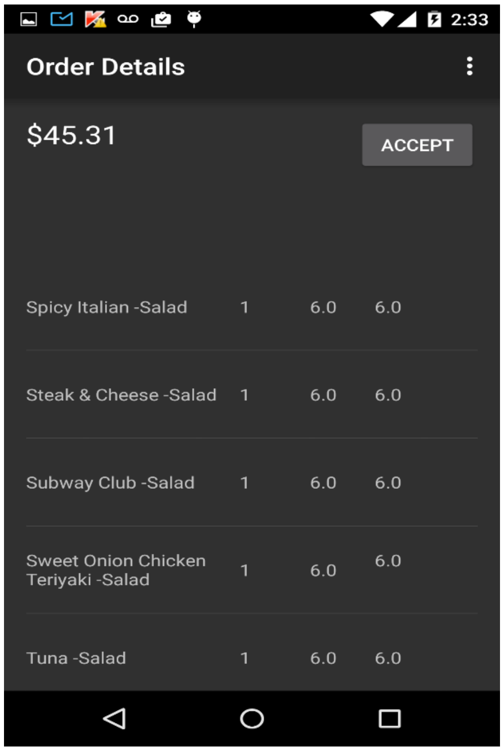
The following activities shows the list open orders which is one of admin services:




2:33

FrontEnd_PG4

srikanth	$ 62.21
nagender	$ 47.45
nikhil	$ 23.76
saiteja	$ 45.31


2:33

Order Details

$45.31

Spicy Italian -Salad	1	6.0	6.0
Steak & Cheese -Salad	1	6.0	6.0
Subway Club -Salad	1	6.0	6.0
Sweet Onion Chicken Teriyaki -Salad	1	6.0	6.0
Tuna -Salad	1	6.0	6.0

Testing:

We added few services to the API we developed in the previous iterations. Now, it is capable of retrieving items list, modify item details, add new items and reviewing orders placed by customers.

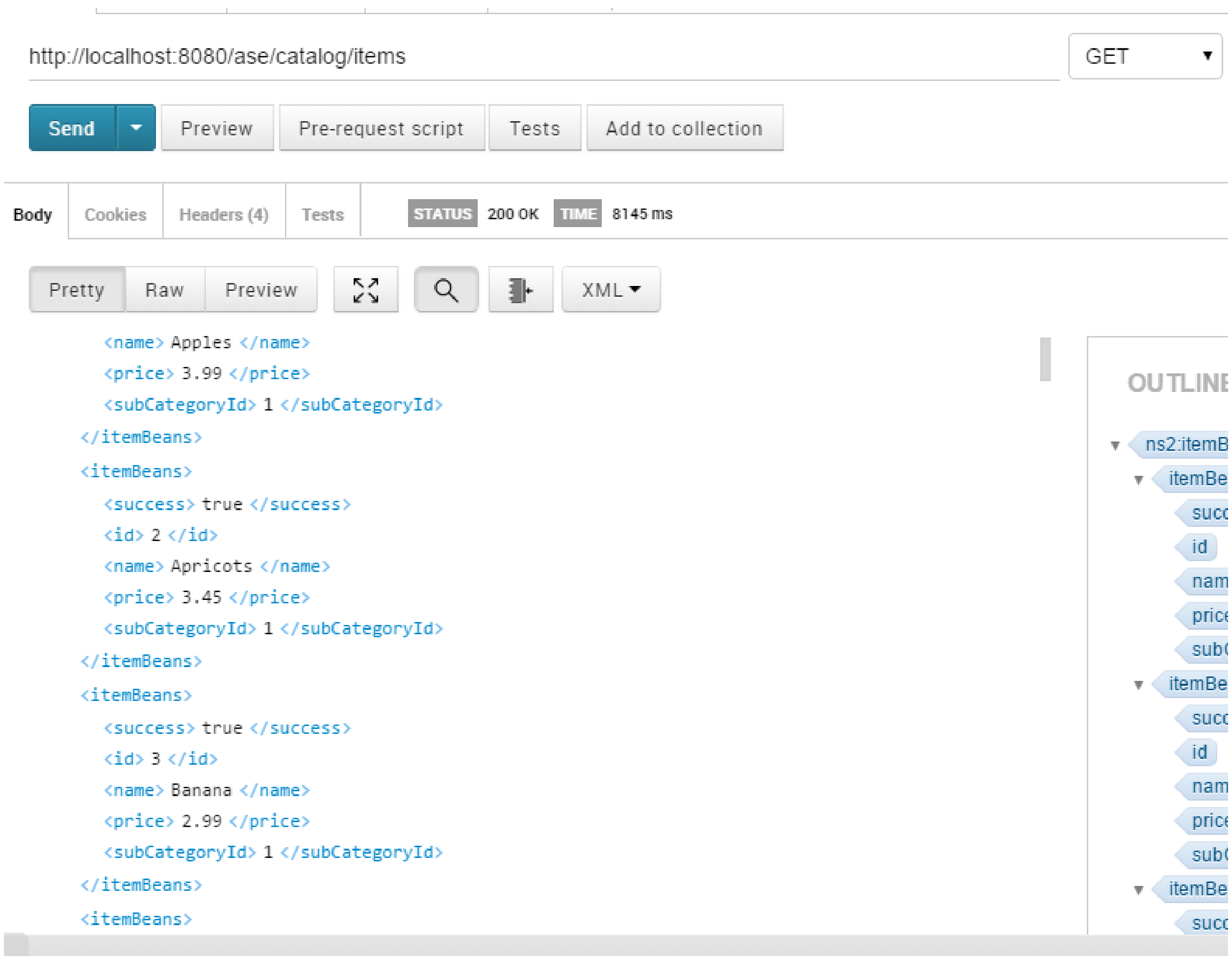
We are testing these services provided by API using POSTMAN an extension of google chrome.

Retrieving items list:

```
@Path("/items")
@GET
@ElementClass(response = ItemBeanList.class)
public ItemBeanList getAllItems();
```

url: http://localhost:8080/ase/catalog/items

operation: GET



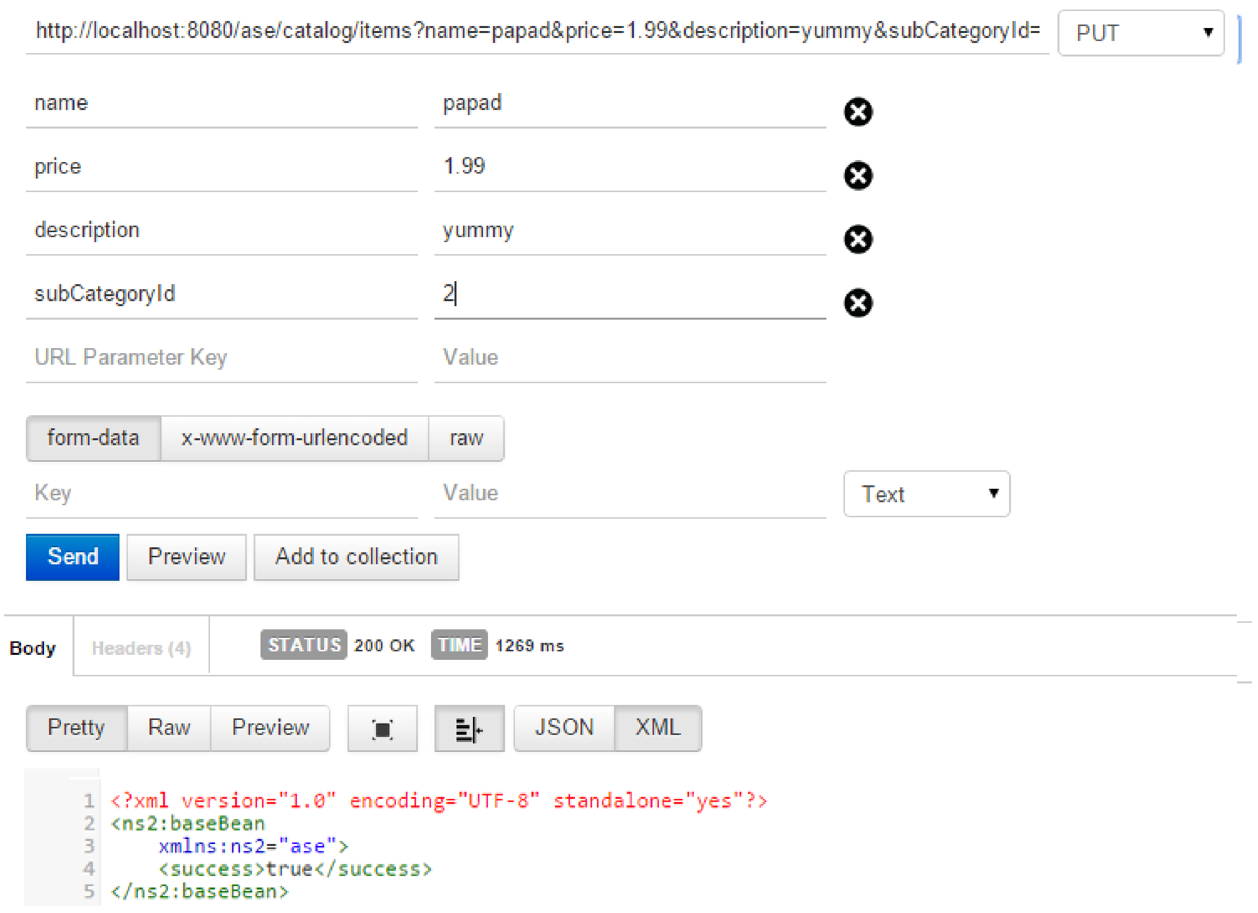
```
    <name> Apples </name>
    <price> 3.99 </price>
    <subCategoryId> 1 </subCategoryId>
  </itemBeans>
  <itemBeans>
    <success> true </success>
    <id> 2 </id>
    <name> Apricots </name>
    <price> 3.45 </price>
    <subCategoryId> 1 </subCategoryId>
  </itemBeans>
  <itemBeans>
    <success> true </success>
    <id> 3 </id>
    <name> Banana </name>
    <price> 2.99 </price>
    <subCategoryId> 1 </subCategoryId>
  </itemBeans>
  <itemBeans>
```

Modifying item details:

```
@Path("/item")
@PUT
@ElementClass(request = ItemBean.class, response = BaseBean.class)
public BaseBean addItem(@QueryParam("") ItemBean itemBean);
```

url: http://localhost:8080/ase/catalog/items

operation: PUT



http://localhost:8080/ase/catalog/items?name=papad&price=1.99&description=yummy&subCategoryId=

PUT

name	papad
price	1.99
description	yummy
subCategoryId	2

STATUS 200 OK TIME 1269 ms

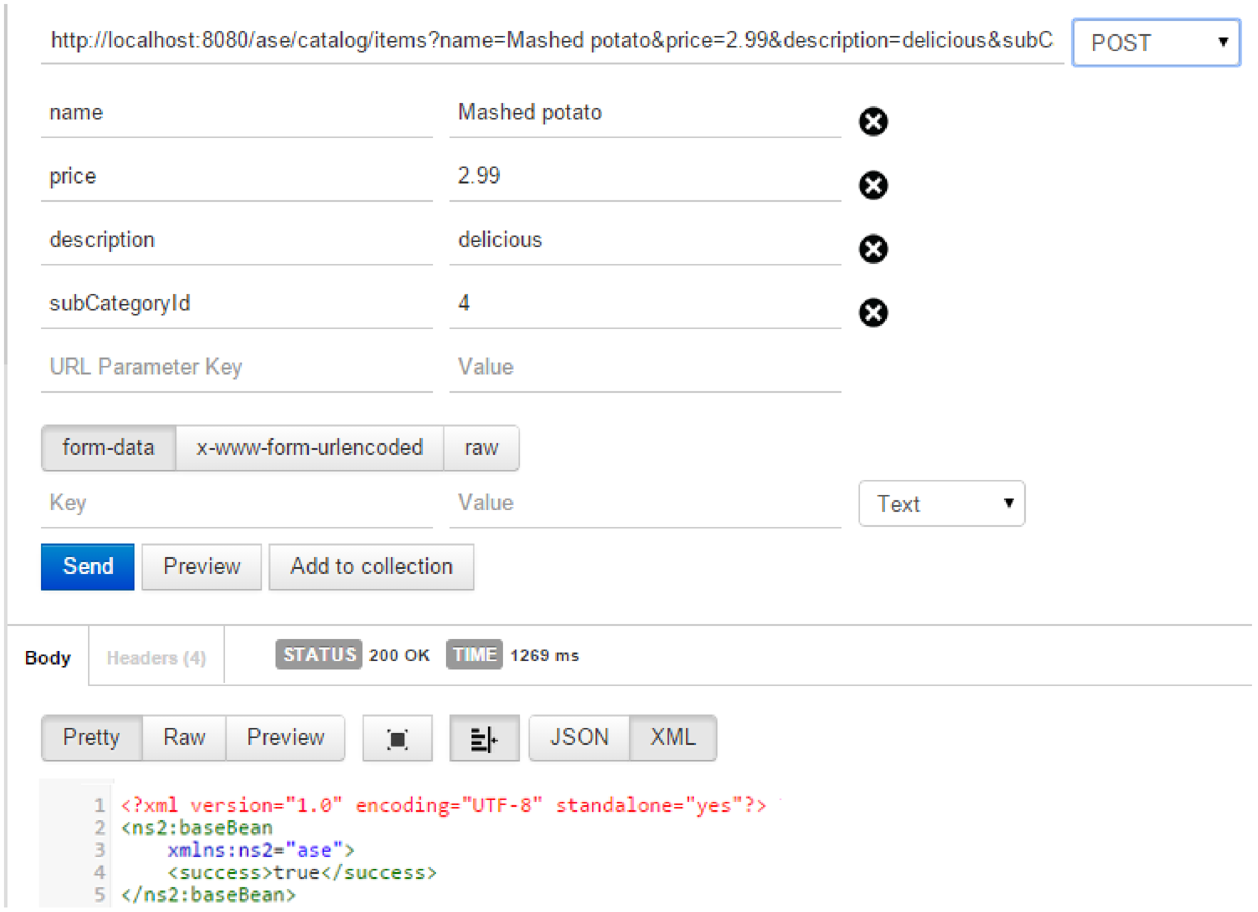
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:baseBean
    xmlns:ns2="ase">
    <success>true</success>
</ns2:baseBean>
```

<u>Adding new item:</u>

```
@Path("/item")
@POST
@ElementClass(request = ItemBean.class, response = BaseBean.class)
public BaseBean updateItem(@QueryParam("") ItemBean itemBean) throws Exception;
```

<u>url</u>: http://localhost:8080/ase/catalog/items

<u>operation:</u> POST



http://localhost:8080/ase/catalog/items?name=Mashed potato&price=2.99&description=delicious&subC

POST

name	Mashed potato
price	2.99
description	delicious
subCategoryId	4
URL Parameter Key	Value

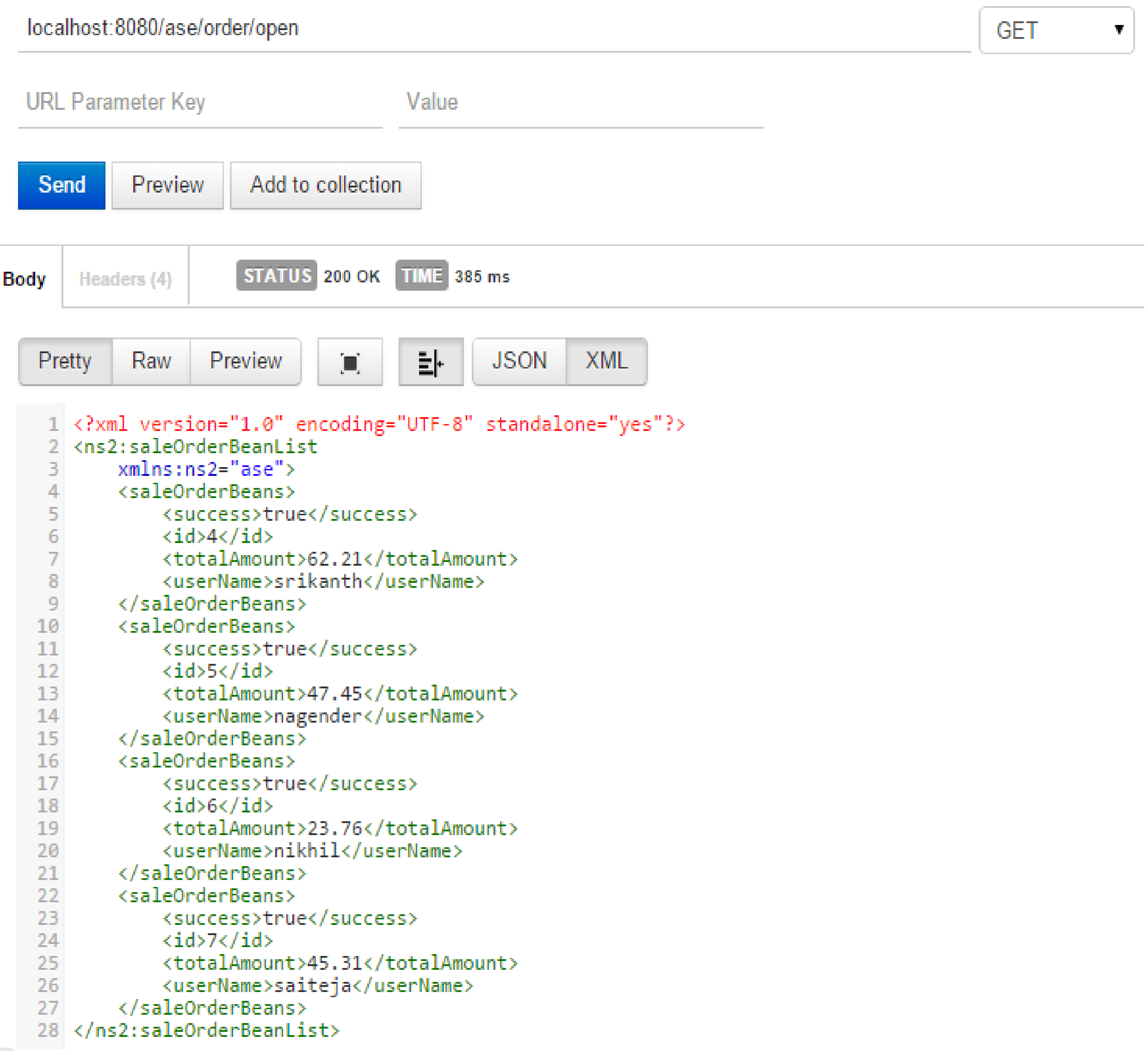
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:baseBean
    xmlns:ns2="ase">
    <success>true</success>
</ns2:baseBean>
```

Reviewing list of open orders:

```
@GET
@Path("/open")
@ElementClass(response = SaleOrderBeanList.class)
public SaleOrderBeanList getOpenedOrders();
```

url: http://localhost:8080/ase/order/open

operation: GET



localhost:8080/ase/order/open GET

URL Parameter Key | Value

Send | Preview | Add to collection

Body | Headers (4) | STATUS 200 OK | TIME 385 ms

Pretty | Raw | Preview | JSON | XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:saleOrderBeanList
    xmlns:ns2="ase">
    <saleOrderBeans>
        <success>true</success>
        <id>4</id>
        <totalAmount>62.21</totalAmount>
        <userName>srikanth</userName>
    </saleOrderBeans>
    <saleOrderBeans>
        <success>true</success>
        <id>5</id>
        <totalAmount>47.45</totalAmount>
        <userName>nagender</userName>
    </saleOrderBeans>
    <saleOrderBeans>
        <success>true</success>
        <id>6</id>
        <totalAmount>23.76</totalAmount>
        <userName>nikhil</userName>
    </saleOrderBeans>
    <saleOrderBeans>
        <success>true</success>
        <id>7</id>
        <totalAmount>45.31</totalAmount>
        <userName>saiteja</userName>
    </saleOrderBeans>
</ns2:saleOrderBeanList>
```

Deployment:

ScrumDo link:

https://www.scrumdo.com/projects/project/alpha8/iteration/121756/board

UMKC VM:

Since we developed backend in JAVA, we cannot use UMKC VM (only supports .NET)
We are using jetty server plugin in the backend project (maven).

- http://localhost:8080/ase/catalog/items (GET)
- http://localhost:8080/ase/catalog/items (POST)
- http://localhost:8080/ase/catalog/items (PUT)
- localhost:8080/ase/order/open (GET)

GitHub url:

https://github.com/saiteja10/ASE_INCREMENT_4

Project Management:

Scrumdo link:
https://www.scrumdo.com/projects/project/alpha8/iteration/121756/board

Implementation status report:

Work completed:
In this iteration, for the backend part we added few more services to the REST we developed in the previous iterations. Now, it is capable of retrieving items list, modifying item details, adding new items, obtaining orders placed by customers.
For the front end we have developed the activity page for Admin services, activity for admin to see the list of items, activity page to modify the item details, activity for adding new items to the list, activity page for reviewing list of open orders.
Also, we integrated all the services we implemented so far and fixed some minor bugs.

Task	**Person**
Functionality of mobile client	Sai Teja Saranam
Call rest services from android	Sai Teja Saranam
Parsing and representing data	Sai Teja Saranam
Layout of mobile client	Srikanth Nadendla
Implementation of new REST services (Apache CXF)	Sai Teja Saranam
Hibernate (mapping Java classes to database tables)	Nagender Goud, Srikanth Nadendla
Spring Dependency Injection	Nagender Goud, Ram Nikhil
Testing REST services	Nagender Goud, Ram Nikhil, Srikanth Nadendla
Integration of services	Ram Nikhil, Srikanth Nadendla
Fixing minor bugs	Nagender Goud
Contributions	
Member	**Percentage**
Sai Teja Saranam	25
Srikanth Nadendla	25
Nagender Goud	25
Ram Nikhil	25