

Third increment Report

Objectives:

The objective of this increment is to develop the employee services and extend the services of REST API implemented in the previous iterations which will be useful for developing employee services.

Existing Services/API:

For the front end 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, we implemented our own API in previous iterations and extended few 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 iteration a customer can see list of different food items (fruits, vegetables, fast foods and restaurant menu) and can add them to cart. In this iteration, we are storing all the orders placed by different customers. When an employee login to the application, he can see all the list of open orders. He can choose to pick any order and process them. Once an employee selects an order to process, it will automatically be removed from the list of open orders. When another employee logs in to the application, he can't see the processed orders.

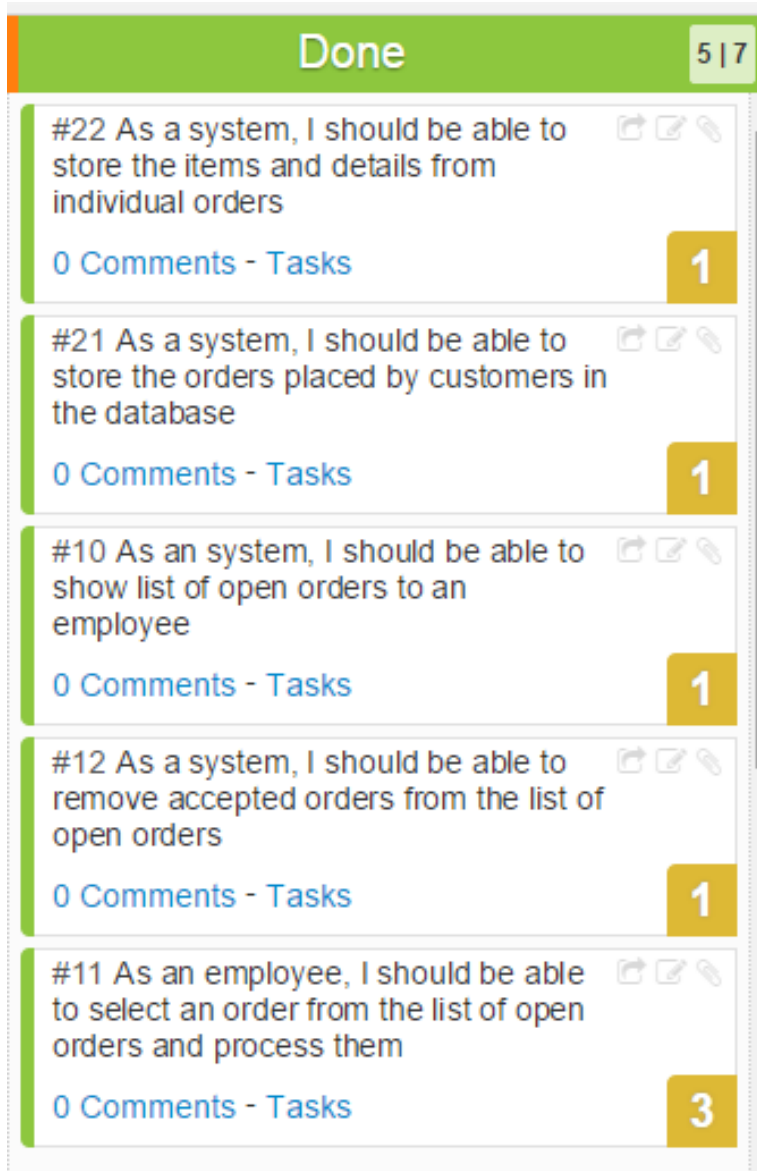
Increment 3 - March 19, 2015 - April 8, 2015

New Story

Filter Board

Todo	Doing	Reviewing	Done	517
			<div>#22 As a system, I should be able to store the items and details from individual orders</div> <div>0 Comments - Tasks</div> <div>1</div>	
			<div>#21 As a system, I should be able to store the orders placed by customers in the database</div> <div>0 Comments - Tasks</div> <div>1</div>	
			<div>#10 As a system, I should be able to show list of open orders to an employee</div> <div>0 Comments - Tasks</div> <div>1</div>	
			<div>#12 As a system, I should be able to remove accepted orders from the list of open orders</div> <div>0 Comments - Tasks</div> <div>1</div>	
			<div>#11 As an employee, I should be able to select an order from the list of open orders and process them</div> <div>0 Comments - Tasks</div> <div>3</div>	

User stories for Employee service:



Services:

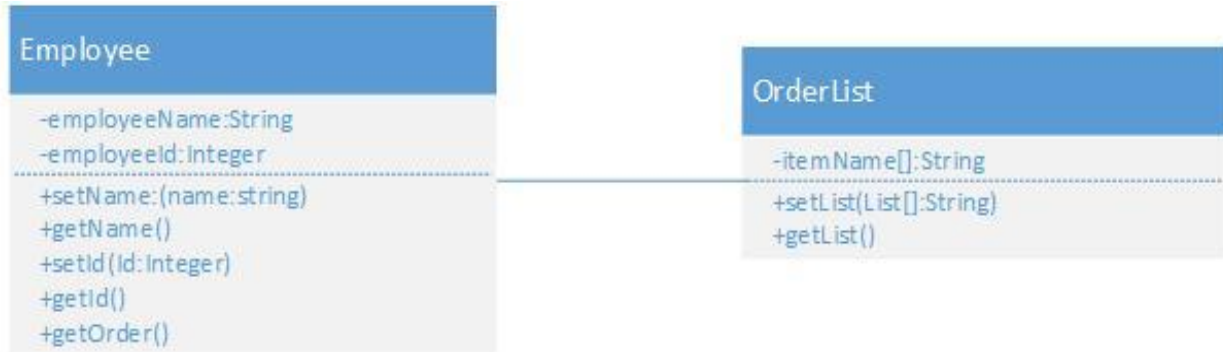
We have developed the registration, login service in first increment and for second increment, we developed the customer service where customer will be able to select items and add them to cart. In this iteration apart from adding services to REST API, we implemented Employee service.

Employee service:

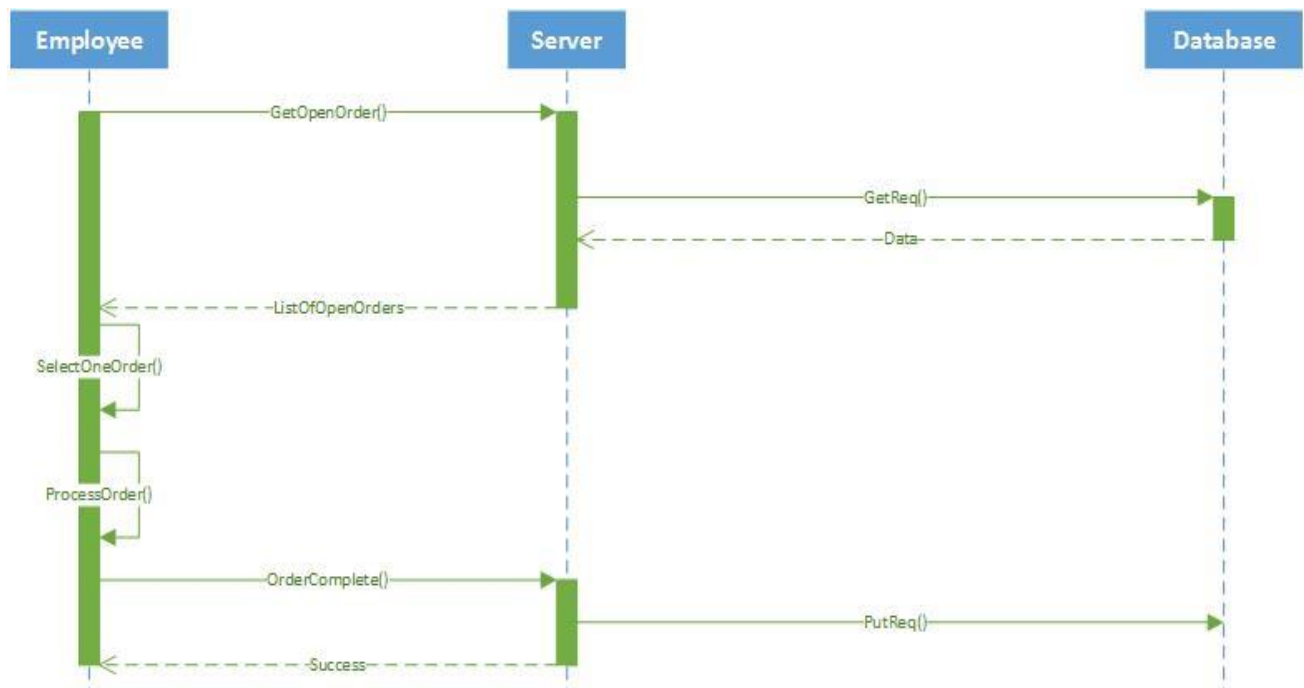
Service description:

When an employee logs into the application, he can see list of open orders and accepts an order of his interest and process them. Accepted orders will be removed from the list of open orders.

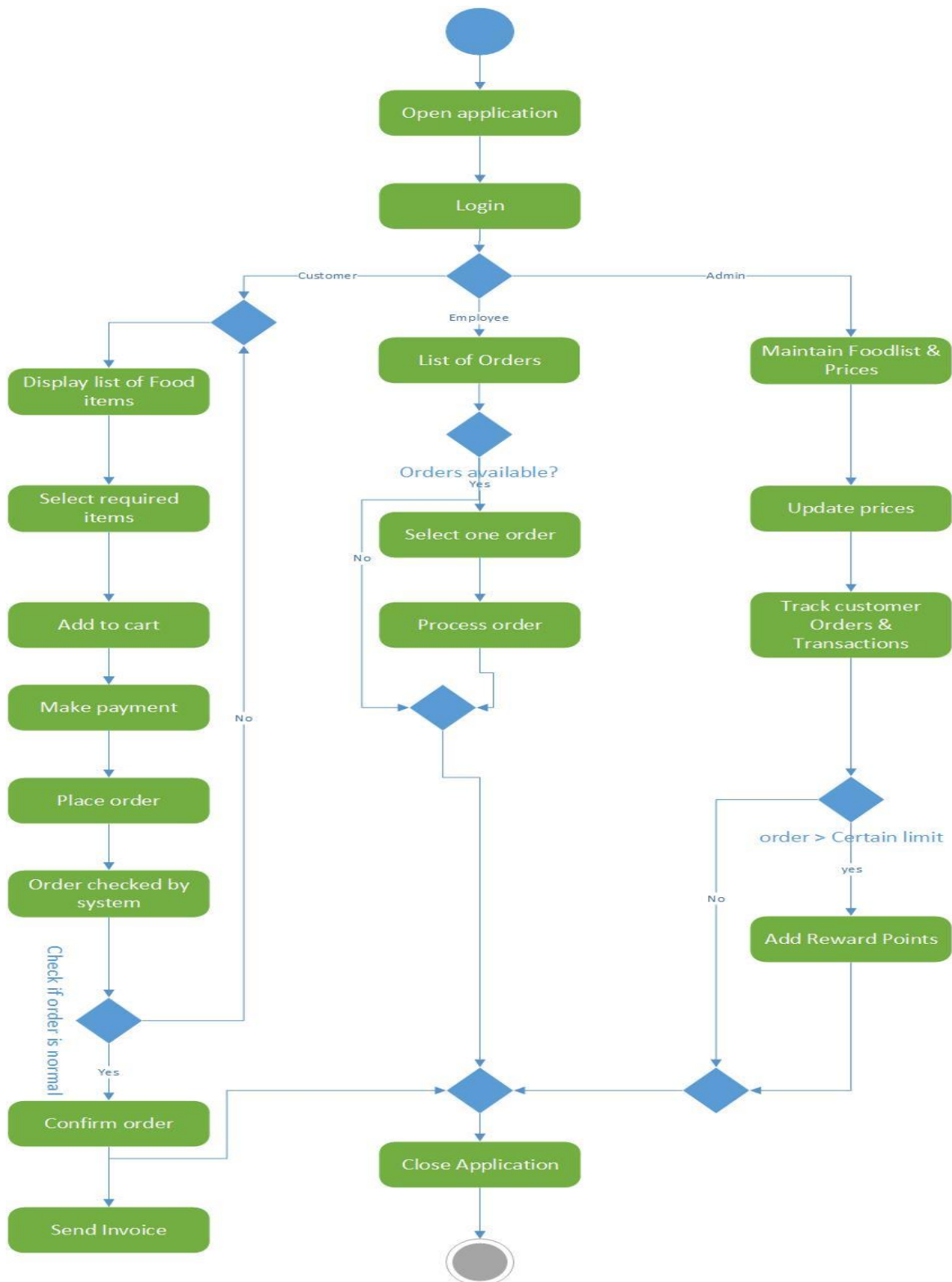
Class diagram:



Sequence diagram:



Activity Diagram:



Implementation:

Implementation of REST services:

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.

We added few more services to the API we developed in the previous iterations. Now, it is capable of obtaining and storing the orders placed by customers, provides orders and order details to the employees who logs in to the application and employee can request each order individually.

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 employee, activity page for displaying list of open orders and activity page for displaying each order (items in order and other details).

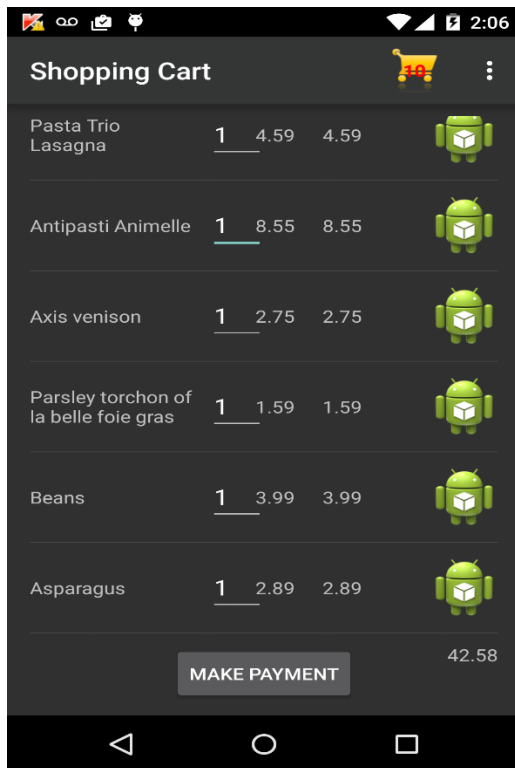
When a person logs in as an employee to the application, s/He will see list of open orders.

When an employee selects an order from the list, order details activity page will appear and an employee can choose to accept and process the order.

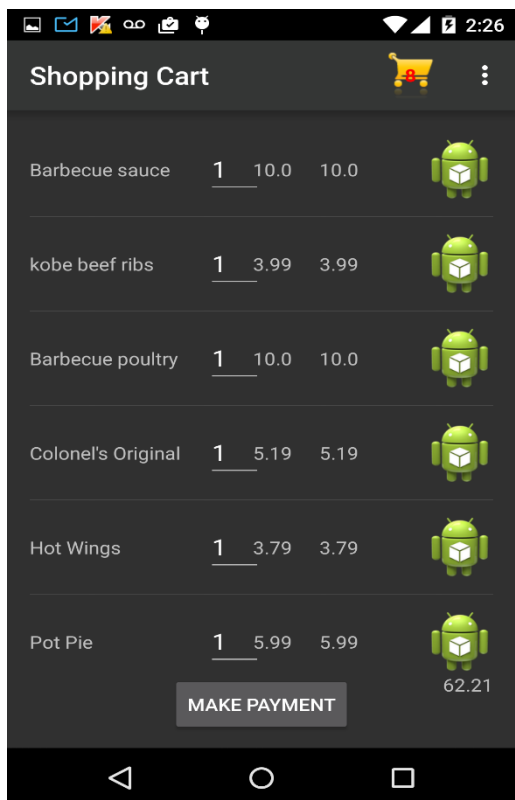
Once he selects an order for processing, no other employee logged into the application can see that order again.

Following are the screen shots for the activities discussed above

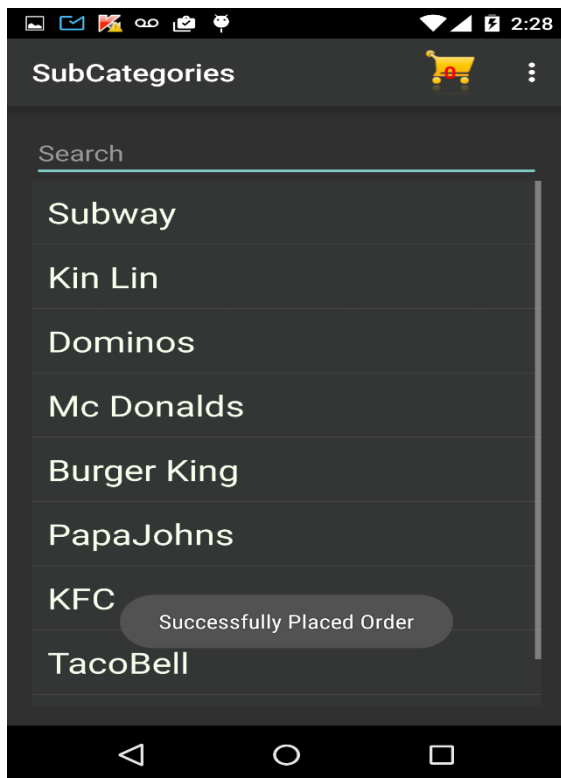
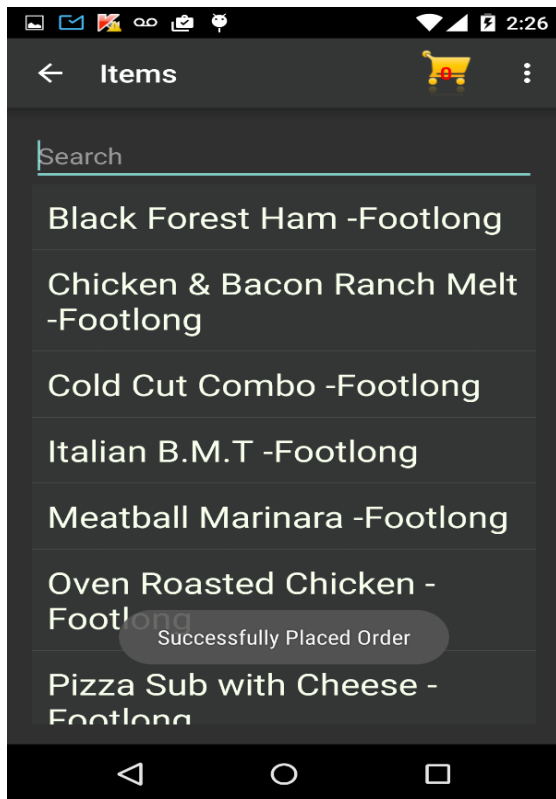
Items added to the cart by the customer 1:



Items added to the cart by the customer 2:



Toast showing that the order was placed successfully:



Employee logging into the application:

The screenshot shows the login interface of an application titled "FrontEnd_PG4". The screen has a dark background. At the top, the title "FrontEnd_PG4" is displayed. Below it, the word "Login" is written in a large, bold, white font. There are two input fields: "User Name" with the text "emp@gmail.com" and "Password" with masked characters ".....". Below the password field, there is a checkbox labeled "Log In as an Employee" which is checked. A "Login" button is positioned below the checkbox. At the bottom, there is a link "Using for the First time?" and a "Register Now!" button. The Android navigation bar is visible at the very bottom.

FrontEnd_PG4

Login

User Name emp@gmail.com

Password

☒ Log In as an Employee

Login

Using for the First time?

Register Now!

Activity page when employee logs in successfully:

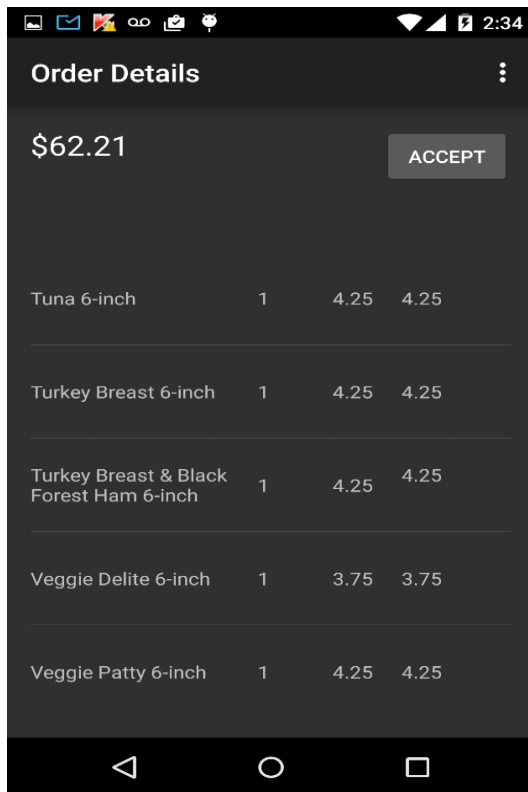
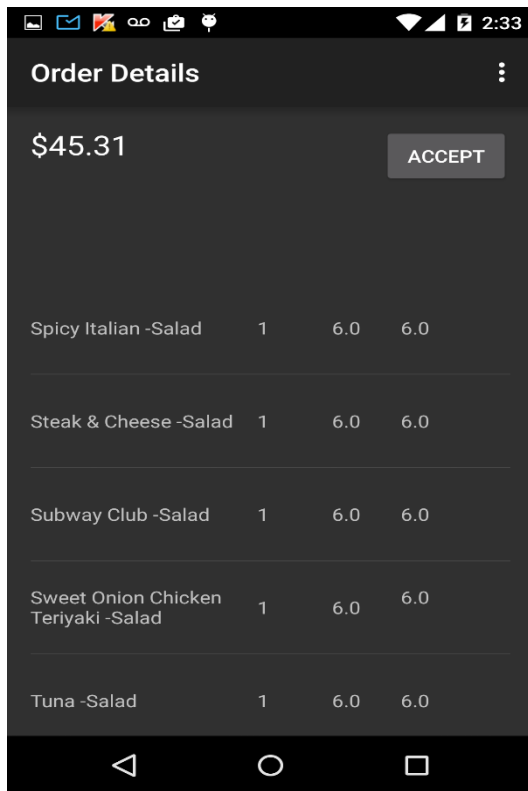
The screenshot shows the activity page of the application after a successful login. The title "FrontEnd_PG4" is at the top. Below it, there is a list of employee names and their corresponding activity values. The data is as follows:

Employee Name	Activity Value
srikanth	\$ 62.21
nagender	\$ 47.45
nikhil	\$ 23.76
saiteja	\$ 45.31

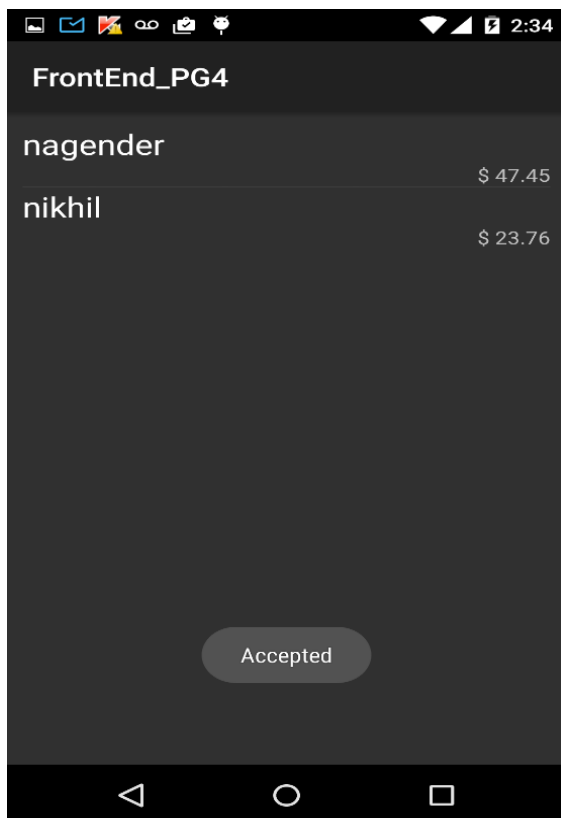
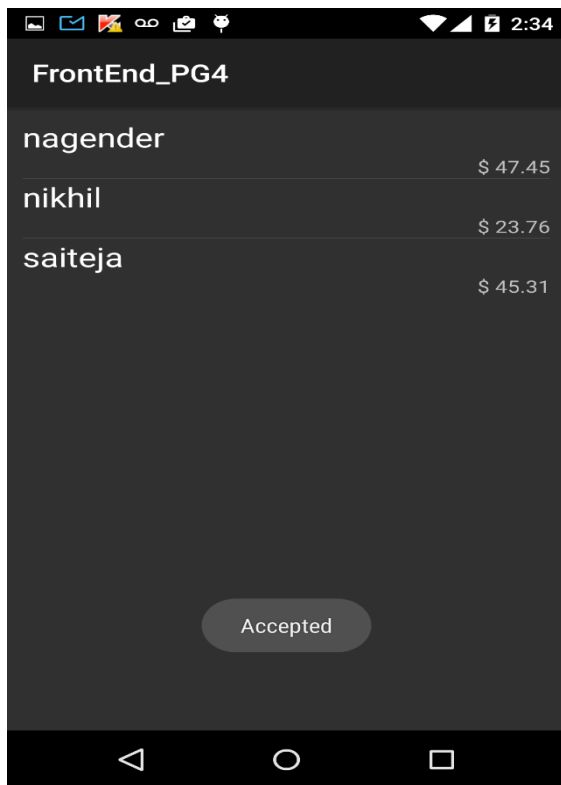
FrontEnd_PG4

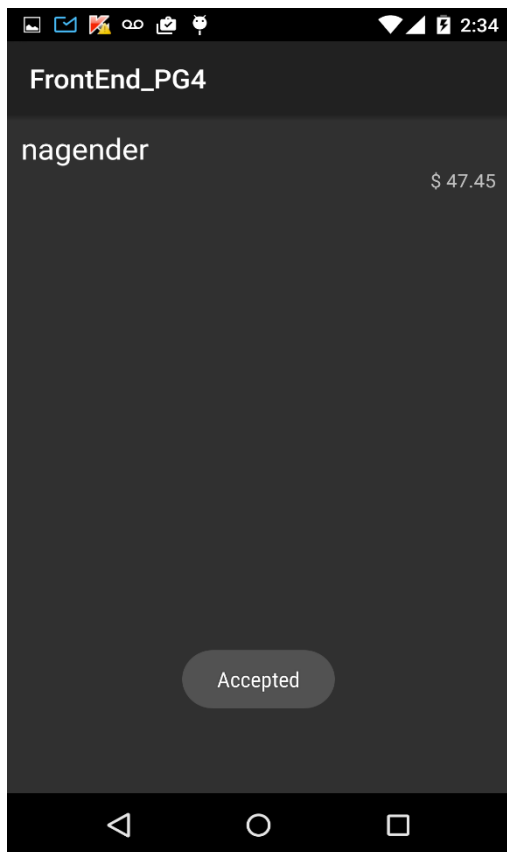
srikanth	\$ 62.21
nagender	\$ 47.45
nikhil	\$ 23.76
saiteja	\$ 45.31

Activity page when employee selects an order from the list:

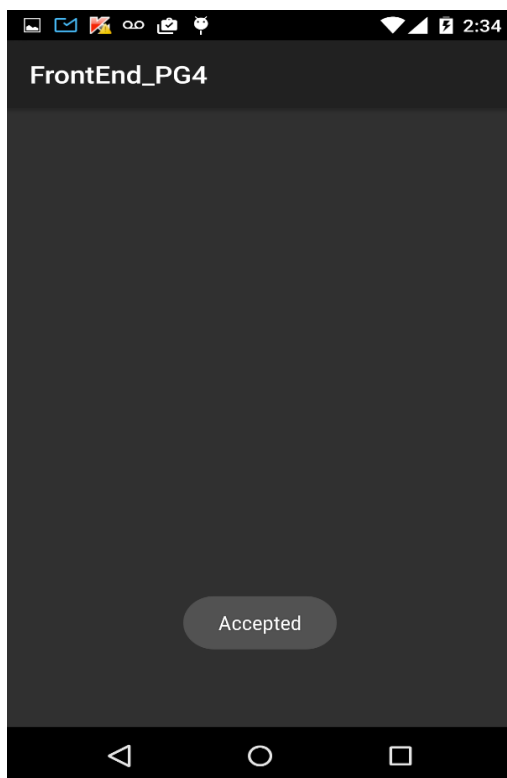


Activity page shown after employee accepts a particular order: (Order removed from the list of open orders):





Activity page showing no more open orders when all of them were accepted:



Testing:

We added few services to the API we developed in the previous iterations. Now, it is capable of obtaining and storing the orders placed by customers, provides orders and order details to the employees who logs in to the application and employee can request each order individually.

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

Placing order:

<http://localhost:8080/ase/order/place>

[http://localhost:8080/ase/order/place?userId=18&totalAmount=156&orderBeans\[0\].itemId=2&orderBeans\[0\].quantity=5&orderBeans\[0\].subTotal=20&orderBeans\[1\].itemId=3&orderBeans\[1\].quantity=2&orderBeans\[1\].subTotal=28](http://localhost:8080/ase/order/place?userId=18&totalAmount=156&orderBeans[0].itemId=2&orderBeans[0].quantity=5&orderBeans[0].subTotal=20&orderBeans[1].itemId=3&orderBeans[1].quantity=2&orderBeans[1].subTotal=28)

http://localhost:8080/ase/order/place?userId=18&totalAmount=156&orderBeans[0].itemId=2&orderBeans[0].quantity=5&orde

PUT ▼

userId	18	✕
totalAmount	156	✕
orderBeans[0].itemId	2	✕
orderBeans[0].quantity	5	✕
orderBeans[0].subTotal	20	✕
orderBeans[1].itemId	3	✕
orderBeans[1].quantity	2	✕
orderBeans[1].subTotal	28	✕
URL Parameter Key	Value	

form-data

x-www-form-urlencoded

raw

Key	Value	Text ▼
-----	-------	--------

Send

Preview

Add to collection

Body

Headers (4)

STATUS 200 OK

TIME 367 ms

Pretty

Raw

Preview

JSON

XML

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

List of open orders:

<localhost:8080/ase/order/open>

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

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ns2:saleOrderBeanList
3   xmlns:ns2="ase">
4   <saleOrderBeans>
5     <success>true</success>
6     <id>4</id>
7     <totalAmount>62.21</totalAmount>
8     <userName>srikanth</userName>
9   </saleOrderBeans>
10  <saleOrderBeans>
11    <success>true</success>
12    <id>5</id>
13    <totalAmount>47.45</totalAmount>
14    <userName>nagender</userName>
15  </saleOrderBeans>
16  <saleOrderBeans>
17    <success>true</success>
18    <id>6</id>
19    <totalAmount>23.76</totalAmount>
20    <userName>nikhil</userName>
21  </saleOrderBeans>
22  <saleOrderBeans>
23    <success>true</success>
24    <id>7</id>
25    <totalAmount>45.31</totalAmount>
26    <userName>saiteja</userName>
27  </saleOrderBeans>
28 </ns2:saleOrderBeanList>
```

Retrieving each order individually:

localhost:8080/ase/order/{saleOrderId}

example: localhost:8080/ase/order/6

GET

Send

Preview

Add to collection

Body

Headers (4)

STATUS 200 OK

TIME 298 ms

Pretty

Raw

Preview

JSON

XML

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ns2:itemBeanList
3   xmlns:ns2="ase">
4   <itemBeans>
5     <success>true</success>
6     <id>150</id>
7     <name>Double Chicken -Salad</name>
8     <price>7.5</price>
9     <quantity>1</quantity>
10    <subCategoryId>3</subCategoryId>
11    <total>7.39</total>
12  </itemBeans>
13  <itemBeans>
14    <success>true</success>
15    <id>151</id>
16    <name>Italian B.M.T. -Salad</name>
17    <price>6.0</price>
18    <quantity>1</quantity>
19    <subCategoryId>3</subCategoryId>
20    <total>7.49</total>
21  </itemBeans>
22  <itemBeans>
23    <success>true</success>
24    <id>152</id>
25    <name>Meatball Marinara -Salad</name>
26    <price>6.0</price>
27    <quantity>1</quantity>
28    <subCategoryId>3</subCategoryId>
29    <total>5.5</total>
30  </itemBeans>
31  <itemBeans>
32    <success>true</success>
33    <id>153</id>
34    <name>Oven Roasted -Salad Chicken</name>
35    <price>6.0</price>
36    <quantity>1</quantity>
37    <subCategoryId>3</subCategoryId>
38    <total>1.69</total>
39  </itemBeans>
40  <itemBeans>
41    <success>true</success>
42    <id>154</id>
```

localhost:8080/ase/order/7

GET

Send

Preview

Add to collection

Body

Headers (4)

STATUS 200 OK TIME 417 ms

Pretty

Raw

Preview



JSON

XML

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ns2:itemBeanList
3   xmlns:ns2="ase">
4   <itemBeans>
5     <success>true</success>
6     <id>155</id>
7     <name>Spicy Italian -Salad</name>
8     <price>6.0</price>
9     <quantity>1</quantity>
10    <subCategoryId>3</subCategoryId>
11    <total>10.0</total>
12  </itemBeans>
13  <itemBeans>
14    <success>true</success>
15    <id>156</id>
16    <name>Steak & Cheese -Salad</name>
17    <price>6.0</price>
18    <quantity>1</quantity>
19    <subCategoryId>3</subCategoryId>
20    <total>6.55</total>
21  </itemBeans>
22  <itemBeans>
23    <success>true</success>
24    <id>157</id>
25    <name>Subway Club -Salad</name>
26    <price>6.0</price>
27    <quantity>1</quantity>
28    <subCategoryId>3</subCategoryId>
29    <total>4.99</total>
30  </itemBeans>
31  <itemBeans>
32    <success>true</success>
33    <id>158</id>
34    <name>Sweet Onion Chicken Teriyaki -Salad</name>
35    <price>6.0</price>
36    <quantity>1</quantity>
37    <subCategoryId>3</subCategoryId>
38    <total>14.0</total>
39  </itemBeans>
```

localhost:8080/ase/order/2

GET

Send

Preview

Add to collection

Body

Headers (4)

STATUS 200 OK TIME 178 ms

Pretty

Raw

Preview



JSON

XML

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ns2:itemBeanList
3   xmlns:ns2="ase">
4   <itemBeans>
5     <success>true</success>
6     <id>104</id>
7     <name>Sweet Potato</name>
8     <price>1.59</price>
9     <quantity>1</quantity>
10    <subCategoryId>2</subCategoryId>
11    <total>3.99</total>
12  </itemBeans>
13  <itemBeans>
14    <success>true</success>
15    <id>105</id>
16    <name>Taro</name>
17    <price>3.99</price>
18    <quantity>1</quantity>
19    <subCategoryId>2</subCategoryId>
20    <total>2.89</total>
21  </itemBeans>
22  <itemBeans>
23    <success>true</success>
24    <id>106</id>
25    <name>Tomatillo</name>
26    <price>5.99</price>
27    <quantity>1</quantity>
28    <subCategoryId>2</subCategoryId>
29    <total>3.99</total>
30  </itemBeans>
31 </ns2:itemBeanList>
```

Accepting open orders:

<localhost:8080/ase/order/accept/{saleOrderId}/{employeeId}>

example <localhost:8080/ase/order/accept/7/2>

localhost:8080/ase/order/accept/7/2 POST ▼

form-data x-www-form-urlencoded raw

Key Value Text ▼

Send Preview Add to collection

Body Headers (4) STATUS 200 OK TIME 489 ms

Pretty Raw Preview   JSON XML

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

localhost:8080/ase/order/accept/6/3 POST ▼

form-data x-www-form-urlencoded raw

Key Value Text ▼

Send Preview Add to collection

Body Headers (4) STATUS 200 OK TIME 471 ms

Pretty Raw Preview   JSON XML

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


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/order/place>
- [http://localhost:8080/ase/order/place?userId=18&totalAmount=156&orderBeans\[0\].itemId=2&orderBeans\[0\].quantity=5&orderBeans\[0\].subTotal=20&orderBeans\[1\].itemId=3&orderBeans\[1\].quantity=2&orderBeans\[1\].subTotal=28](http://localhost:8080/ase/order/place?userId=18&totalAmount=156&orderBeans[0].itemId=2&orderBeans[0].quantity=5&orderBeans[0].subTotal=20&orderBeans[1].itemId=3&orderBeans[1].quantity=2&orderBeans[1].subTotal=28)
- localhost:8080/ase/order/open
- localhost:8080/ase/order/accept/{saleOrderId}/{employeeId}
- localhost:8080/ase/order/accept/7/2

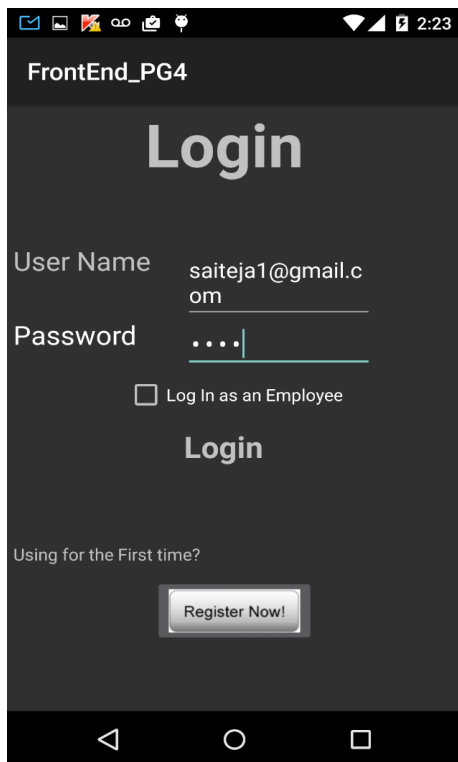
GitHub url:

https://github.com/saiteja10/ASE_INCREMENT_3

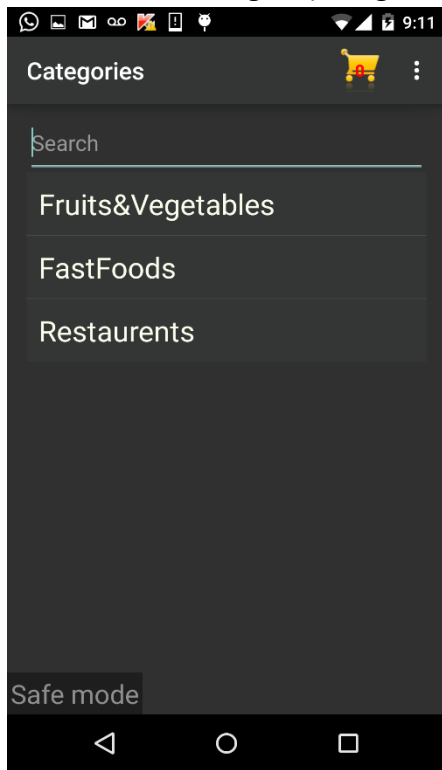
Report:

Mobile client Screenshots:

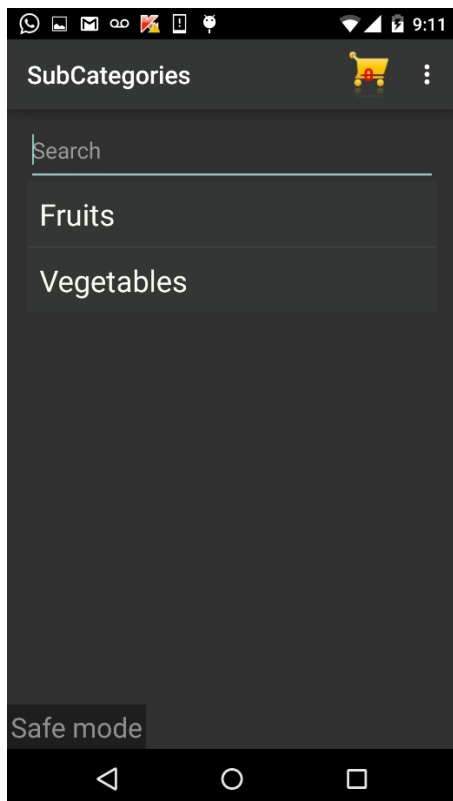
Login page: (customer)



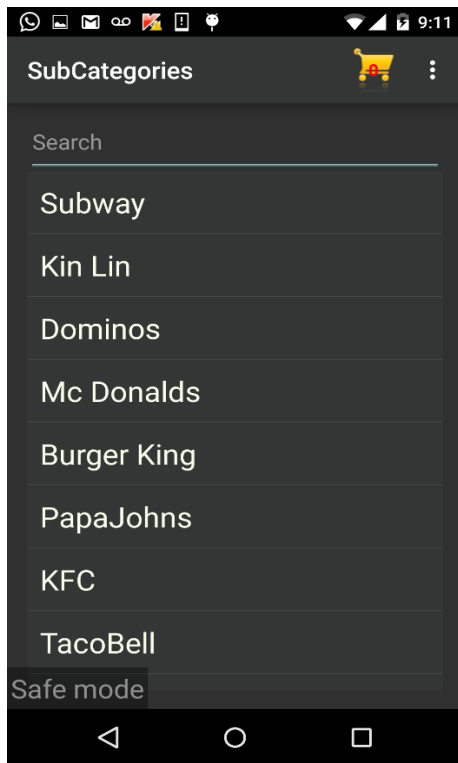
When customer logs in(Categories page):



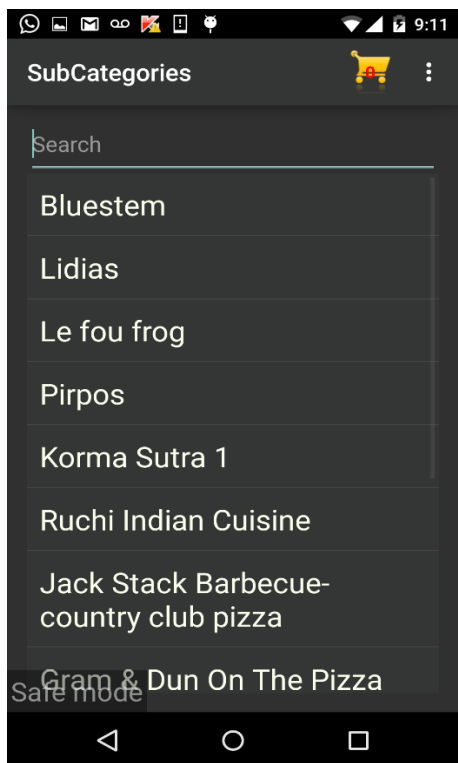
Subcategory page(on tapping fruits and vegetables from category page)



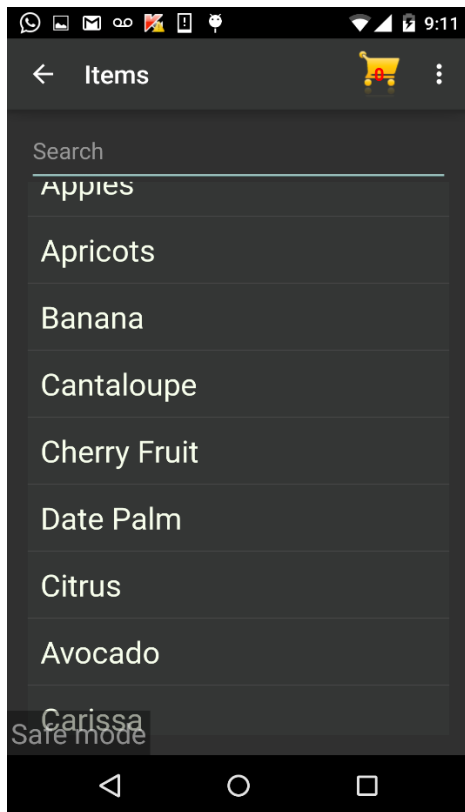
Subcategory page(on tapping Fastfoods from category page)



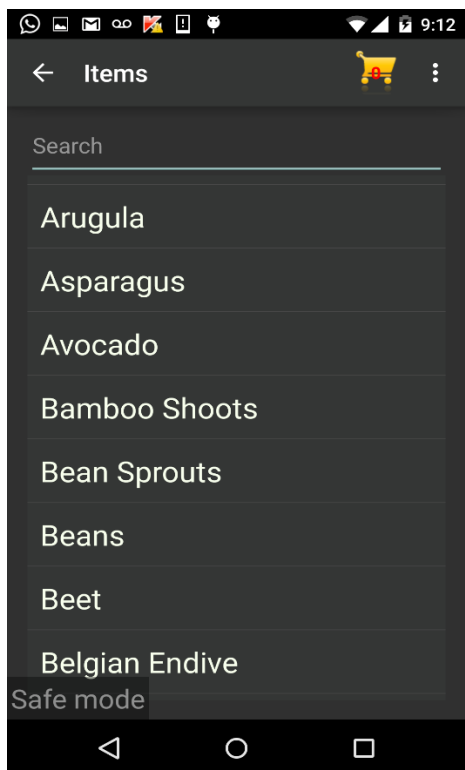
Subcategory page(on tapping Restaurents from category page)



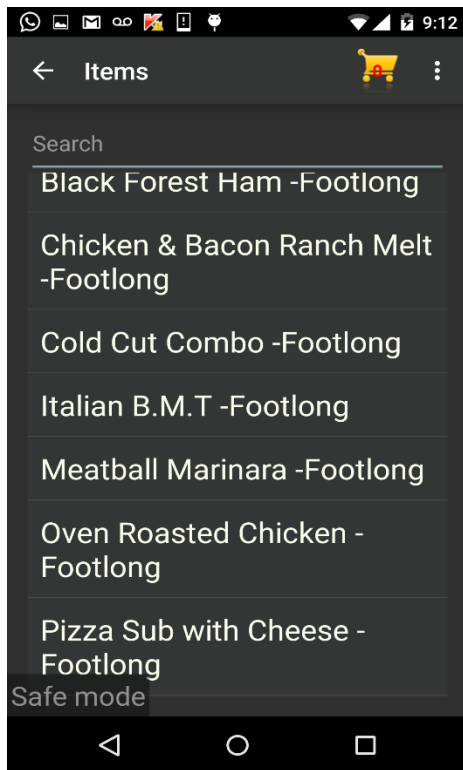
Items page(on tapping fruits from subcategory page)



Items page(Vegetables):



Items page(Subway menu):



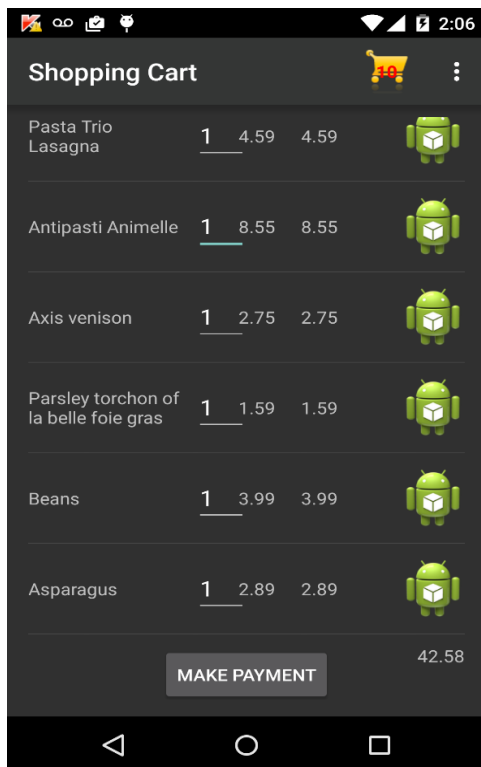
Items page(Burger King menu)



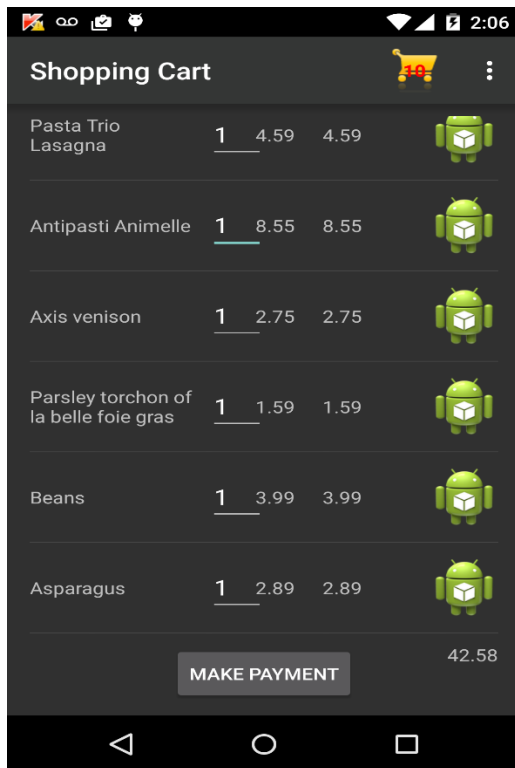
Items page(Ruchi restaurant menu)



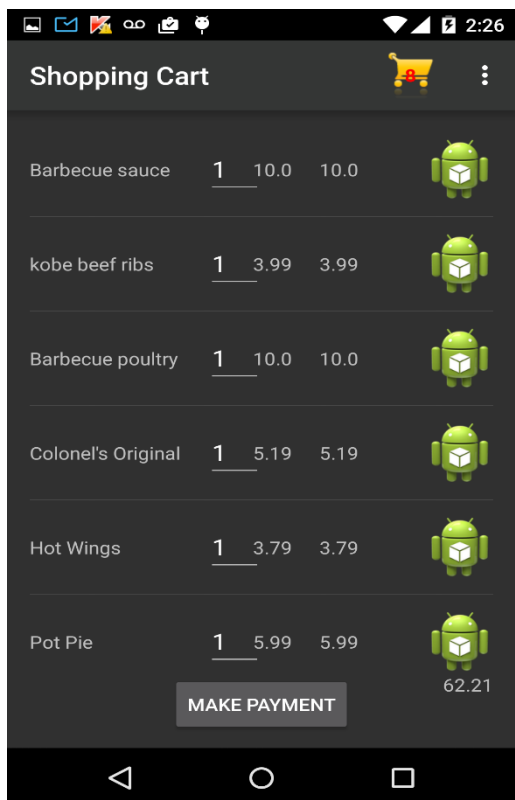
Shopping cart:



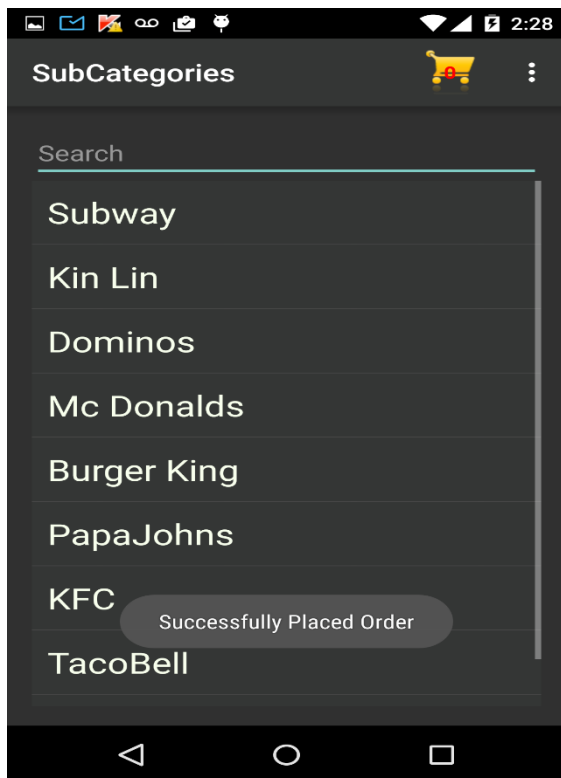
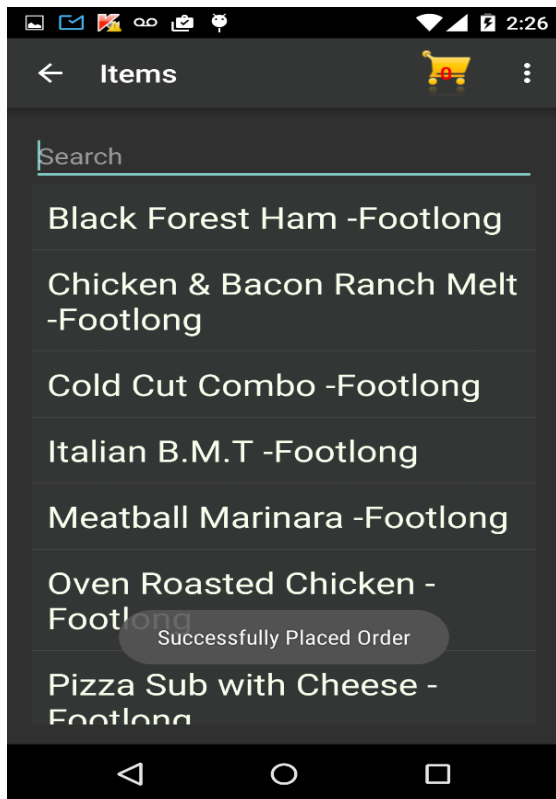
Items added to the cart by the customer 1:



Items added to the cart by the customer 2:



Toast showing that the order was placed successfully:



Employee logging into the application:

The screenshot shows the login interface of an application titled "FrontEnd_PG4". The background is dark grey. At the top, the title "FrontEnd_PG4" is displayed in white. Below it, the word "Login" is written in a large, bold, white font. There are two input fields: "User Name" with the text "emp@gmail.com" and "Password" with masked characters ".....". Below the password field, there is a checkbox that is checked, with the text "Log In as an Employee". A "Login" button is centered below the checkbox. At the bottom, there is a link "Using for the First time?" and a "Register Now!" button. The Android navigation bar is visible at the very bottom.

FrontEnd_PG4

Login

User Name

Password

☒ Log In as an Employee

Login

Using for the First time?

Register Now!

Activity page when employee logs in successfully:

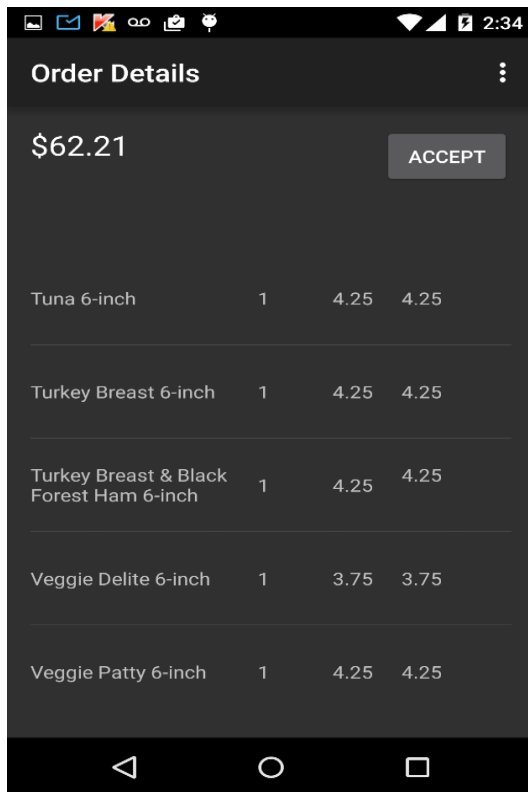
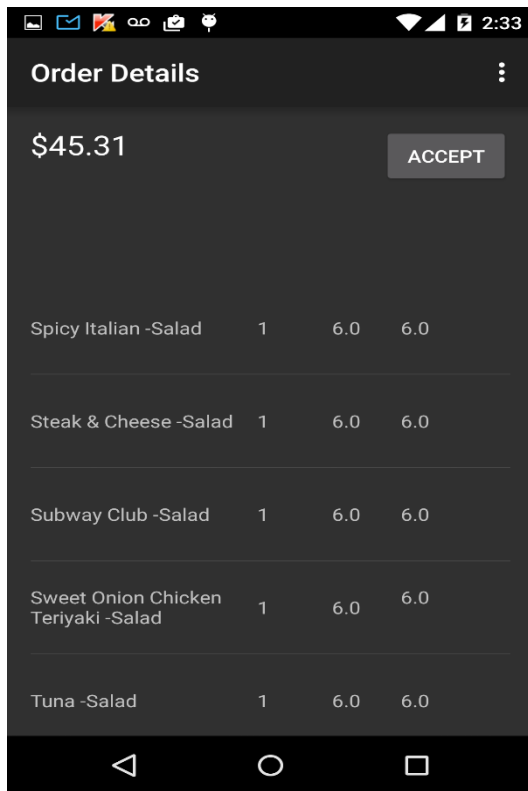
The screenshot shows the activity page of the application. The title "FrontEnd_PG4" is at the top. Below it, there is a list of employee names and their corresponding activity values. The data is as follows:

Employee Name	Activity Value
srikanth	\$ 62.21
nagender	\$ 47.45
nikhil	\$ 23.76
saiteja	\$ 45.31

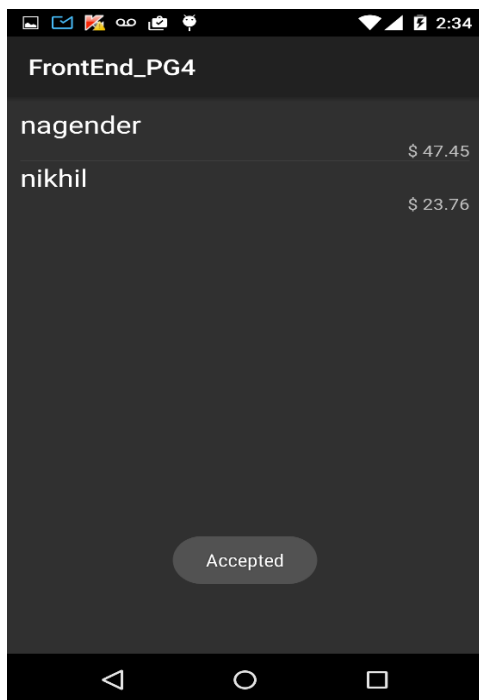
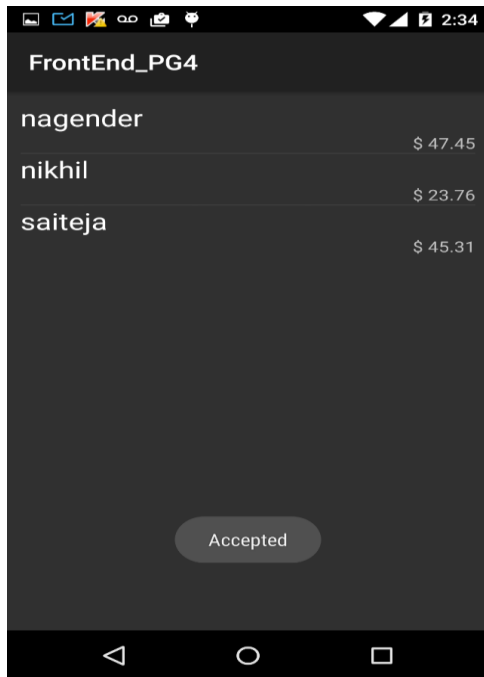
FrontEnd_PG4

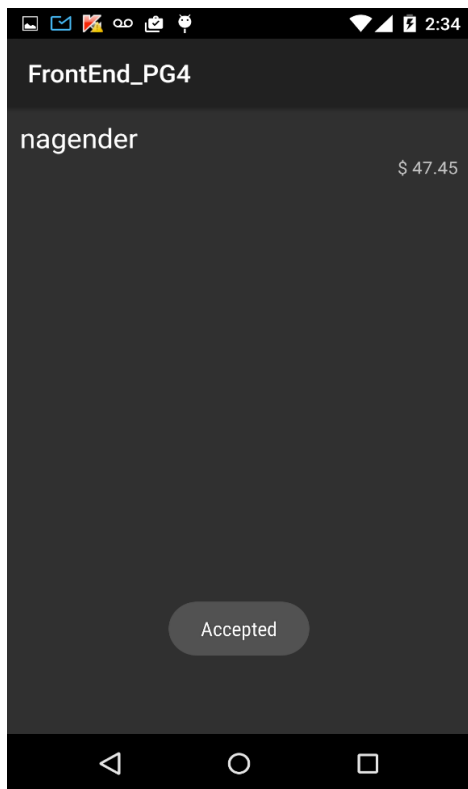
srikanth	\$ 62.21
nagender	\$ 47.45
nikhil	\$ 23.76
saiteja	\$ 45.31

Activity page when employee selects an order from the list:

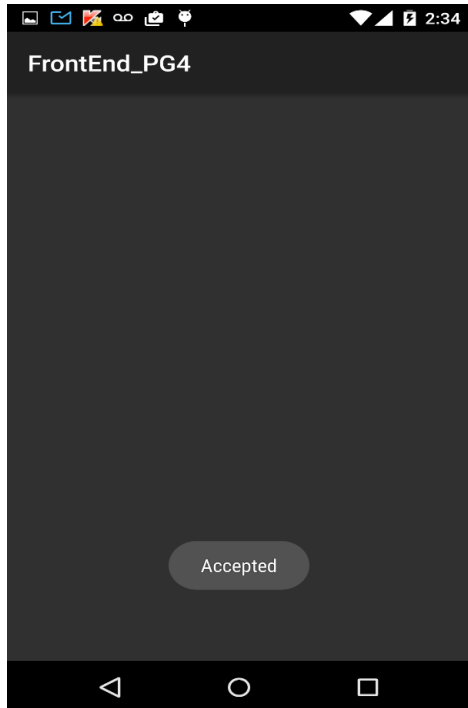


Activity page shown after employee accepts a particular order: (Order removed from the list of open orders):





Activity page showing no more open orders when all of them were accepted:



Project Management:

Scrumdo link:

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

Implementation status report:

Work completed:

In this iteration, we added few more services to the API we developed in the previous iterations. Now, it is capable of obtaining and storing the orders placed by customers, provides orders and order details to the employees who logs in to the application and employee can request each order individually.

Also, we have developed the activity page for employee, activity page for displaying list of open orders, activity page for displaying each order (items in order and other details) and functionality for accepting and processing orders.

Task	Person
Wrote queries for the newly added tables	Sai Teja Saranam
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
Creation of local database to store items in cart	Srikanth Nadendla, Ram Nikhil
Implementation of 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
Contributions	
Member	Percentage
Sai Teja Saranam	30
Srikanth Nadendla	30
Nagender Goud	20
Ram Nikhil	20