Enterprise Integration

Professor JIANG Lingxiao

Instructor ONG hongseng





Food Bear

"We are serious about food"

- Assignment 2 -

|  |  |
| --- | --- |
| **Team** | G5T3 |
| **Members** | Jeon Jemin, Lim Anyu, Lu Ning, Tan Kia Yong,  Yang Chengzhen, Yuan Yuxuan |

1. OVERVIEW OF BUSINESS SCENARIO

FoodBear is a service that is similar to FoodPanda. Our customers access our web interface which displays a list of available food “packages”. Next, customers may then make payment using paypal. On completion, we will:

1. Alert the customer about the order status and delivery time via SMS or Email.
2. Send the order and customer address to the respective restaurants for delivery.



1.1 Business Assumptions

1. FoodBear is the process of transitioning their client’s web interface onto the cloud to save costs by reducing reduncy while increasing scalability.

1.2 Involved IT Systems

|  |  |
| --- | --- |
| **Customer Order System (COS)** | This is a web-based application that will allow customers to log in, search for restaurants that can deliver to the region and then place orders based on the list of restaurants that are returned |
| **Restaurant Management System (RMS)** | This system allows the retrieval of a list of restaurants (and their corresponding menus) that can deliver to a specified location. It is connected to a MySQL restaurant database that stores restaurant details as well as menus offered by the restaurants |
| **Customer Relation Management System (CRM)** | This system manages customer information, including personal details such as email and handphone number as well as the list of restaurants that the customer has previously ordered from. It is connected to a MySQL customer database. |
| **Payments Platform**  **(PP)** | This system processes customer’s payments via paypal. It is part of the company’s initiative to transfer COS’s functionalities to the cloud for cost-savings and scalability. |

1.3 Business Scenario Process

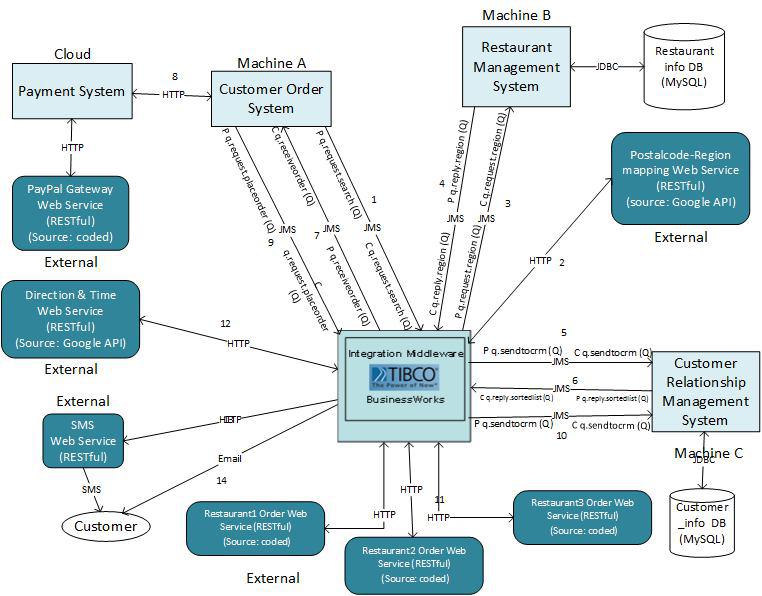
The business process starts when a customer logs onto the Customer Order System (COS) and enters a postal code.

1. The COS will send a synchronous request-reply JMS message to the Integration Middleware (IM) via a queue (q.request.search). The JMS message will contain the postal code and customer ID of the customer.
2. The IM will send a HTTP GET request to a Google web service to convert the region name of the postal code specified
3. The IM will then send a synchronous JMS message to the Restaurant Management System (RMS) via a queue (q.request.region). The JMS message will contain the region name.
4. The RMS will query the restaurant database to retrieve all restaurants that deliver to the specified region and that are currently open. RMS then returns the list of restaurants to the IM via a queue (q.reply.region)
5. The IM will send the customer ID and restaurant list to the Customer Relation Management System (CRM) through a synchronous request-reply JMS message via a queue (q.sendtocrm).
6. The CRM will retrieve the customer’s email, phone number and a list of past orders from the customer database and sort the list of restaurants received from the IM based on the list of past orders. The CRM will then send a reply to the IM with the sorted list of restaurants, customer email and customer hand phone number via a queue (q.reply.sortedlist).
7. The IM will then send the customer’s hand phone number, email and the sorted restaurant list as a reply to the COS via a queue (q.receiveorder). The COS will then store the customer’s email and hand phone number in the current session and display the sorted list of restaurants and menus for the customer to pick from.
8. When the customer has made a selection, he is required to make payment via PayPal
9. COS will send the customer’s ordered items and the total payment amount to our payments site. Our customer will also be re-directed to our payments site hosted.
10. On redirection to the payment’s site, our user will be asked to fill in his address and PayPal account details. These details are then sent from the payments site to PayPal. All transactions between our Payments page and PayPal utilizes SSL to protect the information transfer.
11. Payment status from PayPal will be received and validated by our payments’ site.
12. On validation, our payment’s site will inform COS by returning the customer’s payments status.
13. The COS then sends the order details, customer hand phone number and email and the delivery location postal code to the IM through a fire-and-forget JMS message via a queue (q.request.placeorder).
14. The IM invokes the relevant restaurant’s order RESTFUL web service though HTTP with the order details, and receives the estimated time for preparation by the restaurant.
15. The IM invokes a Google RESTFUL web service via HTTP with the geolocation of the restaurant and the postal code specified by the customer to calculate the estimated time for delivery.
16. The IM then sums the preparation time and delivery time and invokes an SMS RESTFUL web service with the customer’s hand phone number to send a SMS with an acknowledgement of the order and the estimated total time.
17. The IM utilizes an email plugin (within Tibco BusinessWorks suit) to send an email with an acknowledgement of the order and the estimated total time to the customer’s email.

2. Technological Aspects

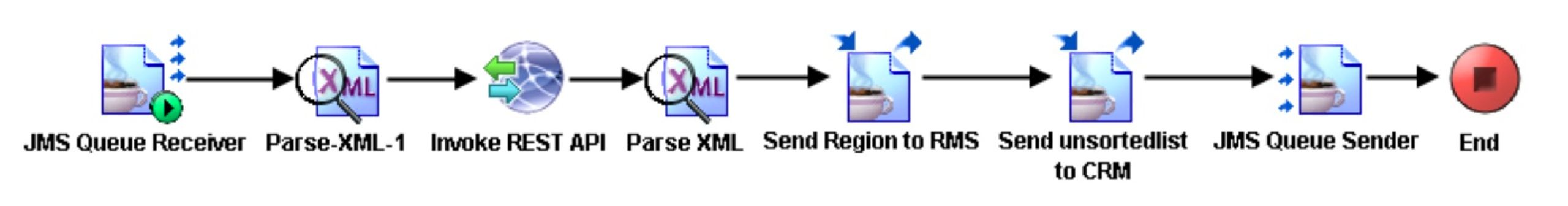
In this section, we cover the technologies that power FoodBear in detail. A list of the technologies that we used are: (1)Electronic Messaging System [Tibco EMS server], (2) XML & XSD Documents, (3) MySQL Databases, (4) Web Services, (5) Integration Middleware [Tibco Businessworks], (6) Cloud application platform [Heroku]

2.1 Technical Diagram

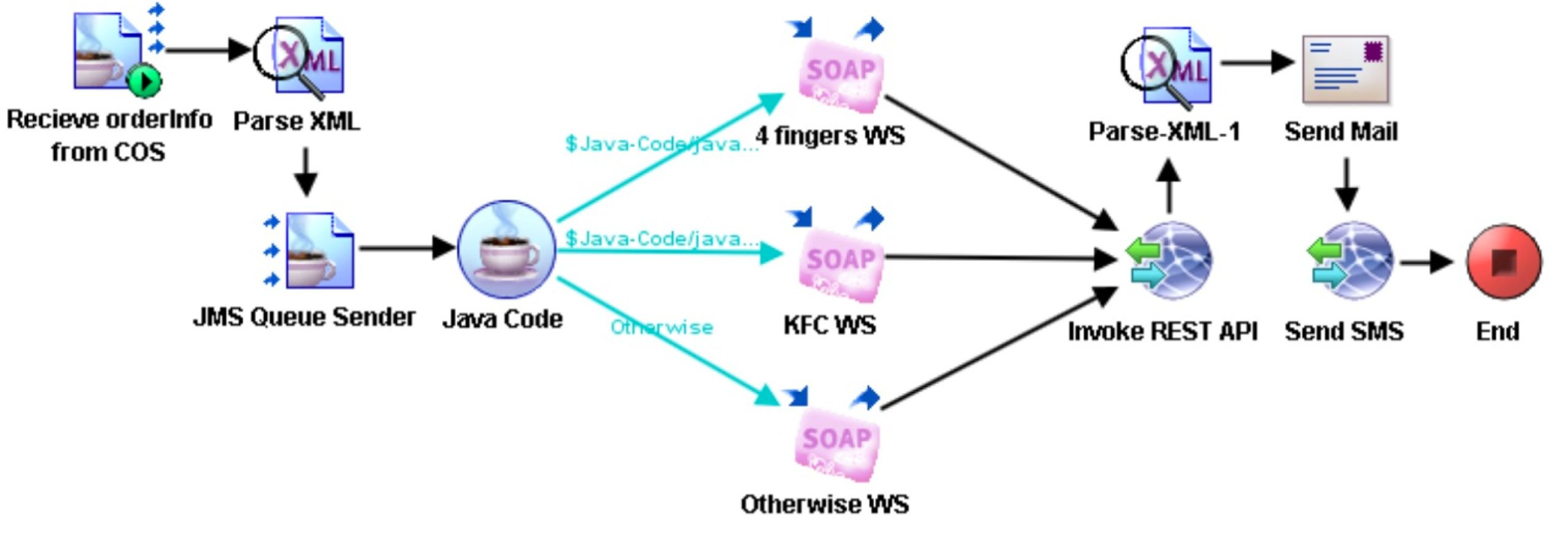


2.2 Process Definition Diagrams

**Process 1**



**Process 2**



2.3 Electronic Messaging System (Tibco EMS)

JMS is used to send data between our internal systems. We chose JMS as FoodBear needs to serve a large number of customers concurrently. By using queues to forward messages internally, we will be able to process high volumns of requests quickly and reliably.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Integration** | **From** | **To** | **Sync/Async** | **Fire and Forget/Request Reply** | **Pattern** | **Queue/Topic/Topic(durable subscribers)** | **Queue/Topic Name** |
| 1 | COS | IM | Sync | RR | P2P | Queue | q.request.search |
| IM | COS | Sync | RR | P2P | Queue | q. receiveorder |
| 2 | IM | RMS | Sync | RR | P2P | Queue | q.request.region |
| RMS | IM | Sync | RR | P2P | Queue | q.reply.region |
| 3 | IM | CRM | Sync | RR | P2P | Queue | q.sendtocrm |
| CRM | IM | Sync | RR | P2P | Queue | q.reply.sortedlist |
| 4 | COS | IM | - | FF | P2P | Queue | q. request.placeorder |
| 5 | IM | CRM | - | FF | P2P | Queue | q.sendtocrm |

**Integration 1 (q.request.search / q.receiveorder)  
COS** sends **SearchCriteria.xml** to the **IM** for querying the list of available food packages. The **IM** will return a sorted list of food packages that are (1) Still open (2) delivers to his area (3) sorted by his or her preferences.

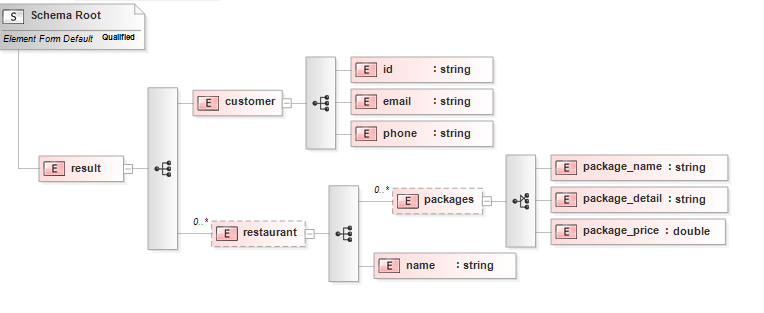
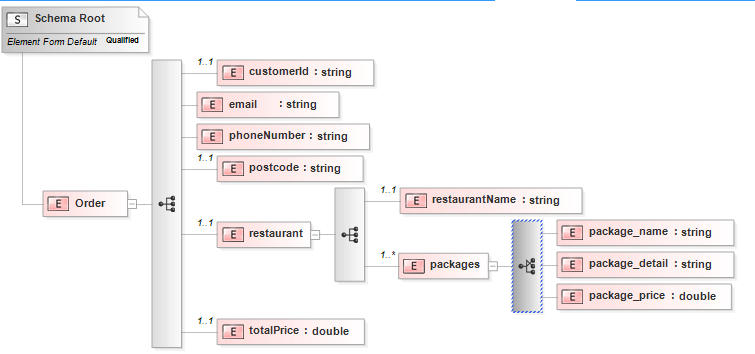
**Integration 2 (q.request.region / q.reply.region)**On parsing **the SearchCriterial.xml** from **COS**, the **IM** forwards the customer’s region as text to the **RMS**. **RMS** returns a list of available food packages, **retrieveregion.xml**, based on criteria (1) Still open, (2) delievers to area.

**Integration 3 (q.sendtocrm / q.reply.sortedlist)  
IM** extracts the available food packages from retrieveregion.xml and sends it as an **XML formatted text** to the **CRM.  
CRM** extracts the customer’s ID and searches for past orders (if any) from the database. Following which, the **CRM** sorts the list of food packages it received from the **IM** and returns it as a **XML formatted text** to the IM.

**Integration 4 (q.request.placeorder)**On confirmation of the customer’s payment, **COS** sends the customer’s order and address via an **XML formatted String** to **IM**.

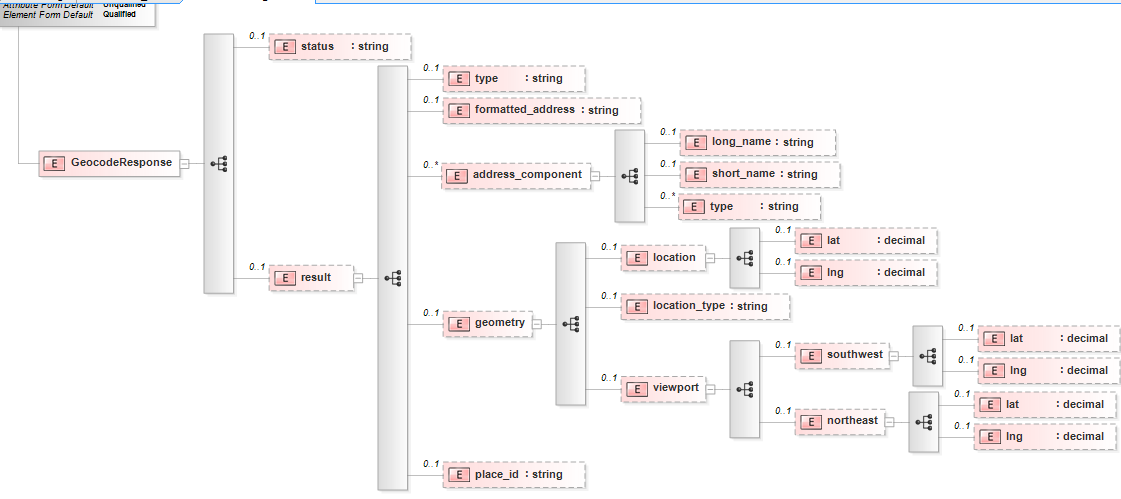
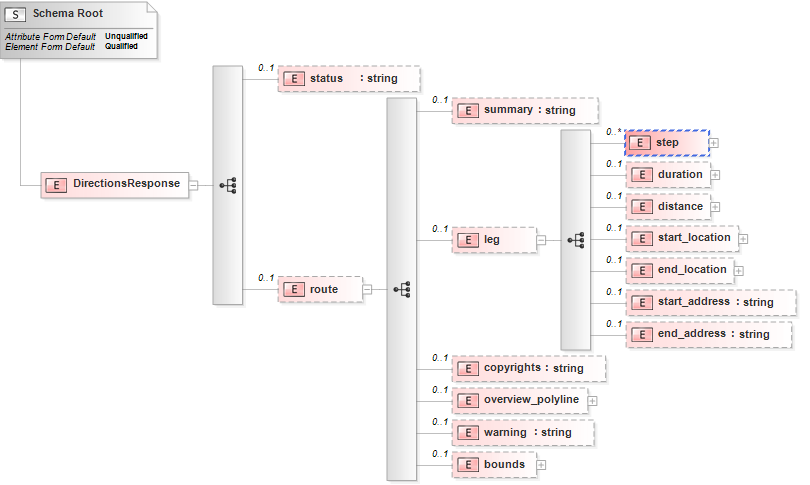
**Integration 5 (q.sendtocrm)**The **IM** will send details from the completed order to the **CRM** where the data will be used for **integration 3**.

2.4 XML & XSD Documents

****

Restaurant\_list.xsd

Order.xsd

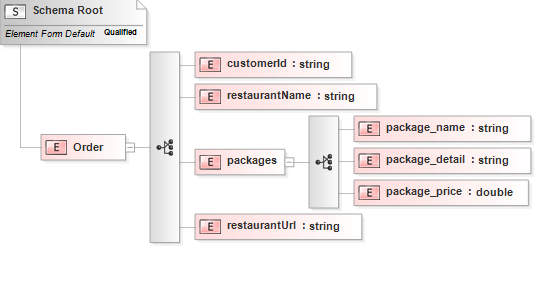
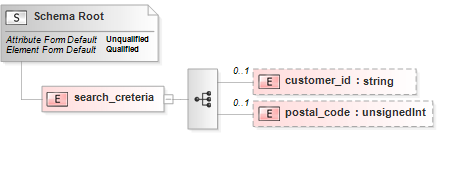
****

retrieveRoute.xsd

(Google APi)

retrieveRegion.xsd

(Google APi)

****

search\_criteria.xsd

sendOrder.xsd

2.5 MySQL Databases

MYSQL databases were used to store restaurants information as well as customer information (personal data, past orders). Databases are used because it is necessary for us to store and retrieve persistent data quickly. In addition, we are able to include criteria in and retrieve queries quickly using (Data indexing).

|  |  |
| --- | --- |
| **Name** | **Description** |
| Customer\_Info | This DB holds all the customers details and their order history. The order history is used to sort the restaurant list that is presented to the customer based on his previous preference. |
| Restaurant\_Info | This DB holds the **delivery scope**, **packages offered** and **restaurant details** of the restaurants. The **delivery scope** is of particular importance; from the customer’s postal code, we determine the area he lives in. From this information we determine the food packages that are available to him/her. |

2.6 Web Services

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input** | **Output** | **Description** |
| Google Directions APi | Mode,  Destination,  Origion | Status Code,  xml String containing route information | We use the directions API to retrieve the delivery time. |
| Google Geocode API | Postalcode | Status Code  xml String containing region information | We use the Geocode API to determine the customer’s area via his postal code. |
| Restaurant Orders | Restaurant, Region(args0),  Restaurant, Region(args1) | Cooking Time(return) | We use webservice to send the customer’s order to the respective restaurants |
| OnewaySMS | Message(containing cooking time and delivering time),  LanguageType,  Senderid,  Mobileno,  Apipassward,  Apiusername, | StatusCode,  msg | This webservice is used to send the delivery time to the customer’s phone |
| Paypal Payment Gateway | Item name, amount, business id, customer’s address, return\_url, notify url. | PayPal Instant Payment Notification | This webservice is used to process the customer’s payment. The return\_url is used to route the customer back after payment, the notify\_url is used by paypal to send notifications to our IPN listener. |

2.7 Integration Middleware (Tibco Businessworks)

We used Tibco BusinessWorks as our integration platform. Tibco businessworks is use for data transformation, content based routing, plugins and automation.

* **Data transformation**

|  |  |  |
| --- | --- | --- |
| **From** | **To** | **Description** |
| COS | Google Geocoding | Transform postal code to “Singapore%20postcode” for google map API |
| Parse XML from COS | CRM | **Concat** CustomerID, restaurantName from **order.xml** to one string and send the string to the **CRM** for database updating. |
| Parse XML from COS | Google destination,  Content-based routing transition (IM) | Split restaurant name from **order.xml** to 2 separate fields, company name and restaurant address where all spaces were replaced by “%20”. For example, “KFC@Dhouby Ghaut”(restaurant name) will be transferred to “KFC”(company name) and “Dhouby%20Ghaut”(address).  Transfer postcode to “Singapore%20postcode”. |
| Restaurant WS, Google destination | OnewaySMS | Transform delivery time into a human readable SMS message and send it to the customer |
| COS  (order.xml) | Email | Extract the order details and total amount paid into a invoice and send it via email. |

* **Content based routing** 
  + From the process 2 diagram above, based on the company name, we route the respective order details to their respective web services.
* **Plugins** **–**  we use the email plugin for tibco businessworks to send business process related information (acknowledgement of order) to the customer

2.8 Cloud Application Platform (Heroku)

Heroku was used to host the payment portion of our Customer Order System(COS) for customers. The management of FoodBear has plans to port the full functionality of our COS system onto a Cloud platform **progressively**. The 2 reasons we used cloud technologies are:

* Management wants to increase costs savings by reducing server maintenance costs.
* Management wants to reduce redundancy by reducing costs during months where customer volume is low.

3. BEYOND THE LABs

|  |  |
| --- | --- |
| **Component** | **Description** |
| Google Maps (WS) | We utilize Google maps direction service to get the estimated delivery time between the restaurant and the customer. Google maps geocode service to locate the input postal codes and translate into political regions. |
| OnewaySMS (WS) | OnewaySMS allows us to send the estimated delivery time to the customer through SMS. |
| Email (TIBCO plugin) | Tibco’s email plugin is easy to use and allows us to send an invoice to the customer through email. |
| Paypal | After the customer chooses their food packages, we will process their payment on the Payments Platform that is integrated with PayPal. Our implementation is orthodox and strictly follows PayPal guidelines; PayPal verifies payment in a unique way, PayPal requires us to secure all transactions with SSL.  In our demo, we utilize PayPal’s sandbox to simulate payments by FoodBear’s customers, we can log into both the “sandbox” business owner and customer accounts and verify that a cash transaction has taken place. Due to our strict adherence to PayPal’s guidelines, we can utilize our PayPal implementation in the real world by setting up a Paypal businessowner account and setting the POST submits to the actual Payments URL that PayPal utilizes. |
| Cloud Application Platform  (Heroku)  (Payments Platform) | FoodBear’s management wishes to migrate COS’s functionalities to a cloud platform in a bid to reduce redundancy and increase scalability. The **Payments Platform** is expected to replace the current **COS**. This will be done in an **incremental process** and for now, FoodBear developers have ported the *payment by PayPal* functionality on to the cloud (<http://ezname.heroku.me/Payments>). |
| Express Framework  (Node.js) | The Payments Platform is coded in Node.js. We also utilized the Express web application framework for development of our **Payments Platform**. There are benefits associated with these choices.   1. NodeJs:    1. Most web browsers already have a JavaScript compiler, this will increase the performance of our site and user’s experience.    2. Node.js utilizes JavaScript. JavaScript allows for call-back functions and by processing requests asynchronously, we will solve the “locking” calls that we experience in the Bank Lab where the server crashes from high load. 2. Express Framework:    1. Provides a framework that allows us to put config files where Heroku expects them    2. Helper libraries that help with our work (E.g. HTTP POST requests). |
| Github | All of our group members picked up Git commands as our team used Github to version all parts of our project (Tibco files, XML files, NodeJs code, Documentation). Our GitHub repository can be accessed at <https://github.com/EIGroupAmazing/eiassignment>. |

4. Systems walkthrough