1. OVERVIEW OF BUSINESS SCENARIO

FoodBear is a service that is similar to FoodPanda. Our customers access our service by visiting a website that displays a list of available food “packages”. Next, customers may then choose to make payment by cash or 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. |

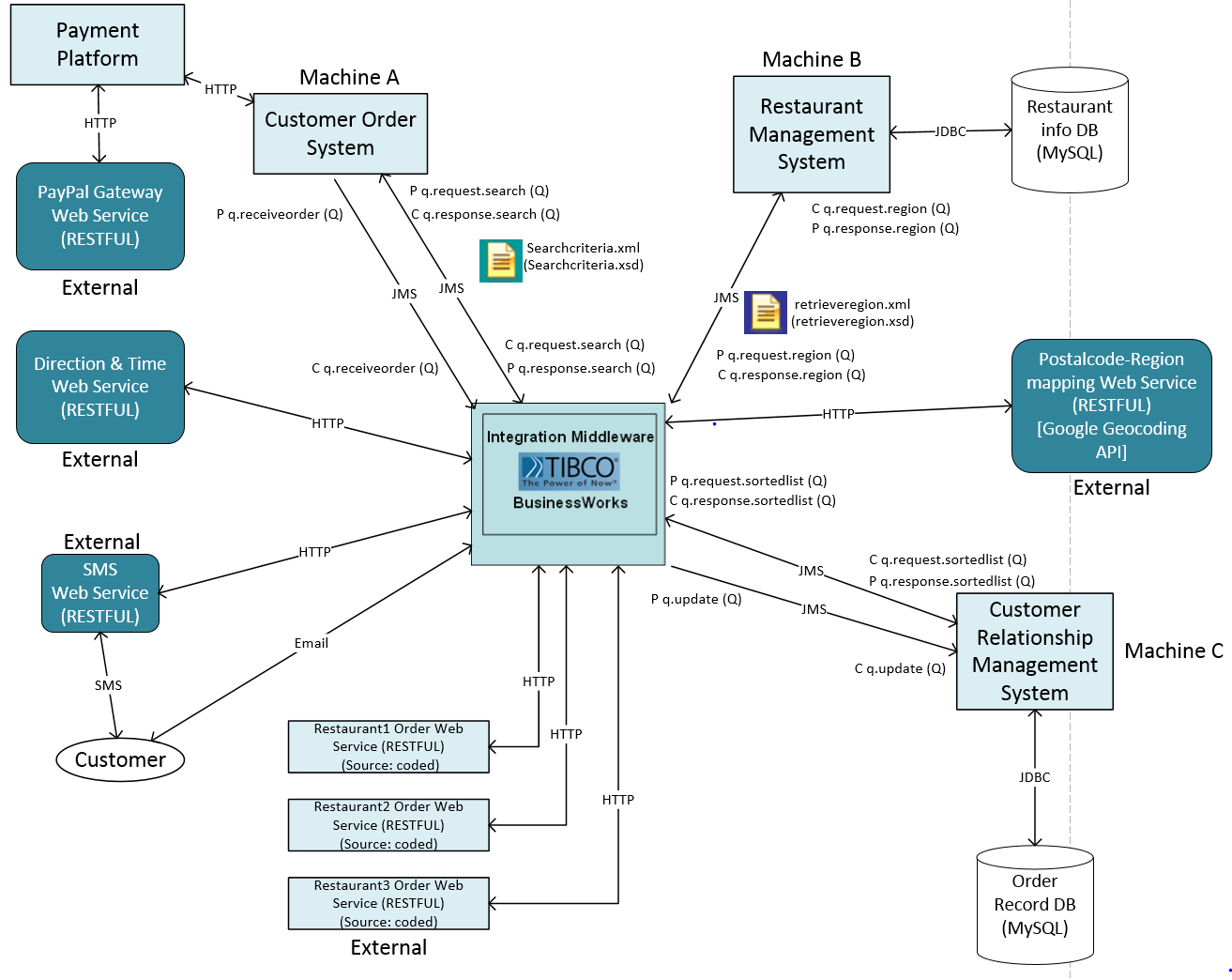
1.3 Business Scenario Process

1. The business process starts when a customer logs onto the Customer Order System (COS) and enters a postal code.
2. 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.
3. The IM will send a HTTP GET request to a Google web service to convert the region name of the postal code specified
4. 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.
5. The RMS will query the restaurant database to retrieve all restaurants that deliver to the specified region and return the list of restaurants to the IM via a queue (q.reply.region)
6. 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.request.sortedlist).
7. 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 handphone number via a queue (q.reply.sortedlist).
8. The IM will then send the customer’s handphone number, email and the sorted restaurant list as a reply to the COS via a queue (q.reply.search). The COS will then store the customer’s email and handphone number in the current session and display the sorted list of restaurants and menus for the customer to pick from.
9. When the customer has made a selection, he is asked to choose a payment method (cash on delivery or through PayPal)
10. If the customer chooses to pay though PayPal. COS will send the customer’s token, items he ordered and the total amount to our payments site. Our customer will also be re-directed to our payments site hosted.
11. 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.
12. Payment status from paypal will be received and validated by our payments’s site.
13. On validation, our payment’s site will inform COS by returning the customer’s token and payments status.
14. The COS then sends the order details, customer handphone number and email and the delivery location postal code to the IM through a fire-and-forget JMS message via a queue (q.receiveOrder).
15. 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.
16. 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.
17. The IM then sums the preparation time and delivery time and invokes an SMS RESTFUL web service with the customer’s handphone number to send a sms with an acknowledgement of the order and the estimated total time.
18. The IM utilizes an email plugin (Tibco BusinessWorks) 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 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 | Pattern | Queue/Topic | Data |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

2.3 XML & XSD Documents

The XML standard is used to format our data between **( ? ? ? )** as it is well structured and can be verified with an XML Schema Definition (XSD) to ensure that the XML being passed is valid.

2.4 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 criterias in and retrieve queries quicky using(Data indexing).

2.5 Web Services

We invoke many external web services for information. This information is then used to help us meet our business requirements.

* Google Maps for estimating delivery time
* OnewaySMS for sms services
* Paypal for payment
* Order placements via the APIs of the respective restaurants

2.6 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 **–** we use tibco to format the XML the order details passed between our systems and external web api. For example, we use bizworks to extract details from our order XML and use them as input for making our web services (SMS, Email)
* Content based routing **–** we use content based routing functionality to determine which restaurants to send the order too.
* Automation – We use Tibco’s process based design to automate api calls, data transfer and notifying customers of their order delivery times.
* Plugins **–**  we use the email plugin for tibco businessworks to send business process related information (acknowledgement of order) to the customer

2.7 Cloud Application Platform (Heroku)

Heroku was used to host the payment portion of our Customer Order System(COS) for customers. We hope to port the full functionality of our COS system to a Cloud platform in the near future. The 2 reasons we used cloud technologies are:

* We did not want to spend resources on setting up and managing our own servers
* We can redudancy, we can cut down on costs during months where customer volumn is lower.

3. BEYOND THE LABs

|  |  |
| --- | --- |
| **Component** | **Description** |
| Google Maps (WS) | We utilize Google maps to get the estimated delivery time between the restaurant and the customer |
| OnewaySMS (WS) | OnewaySMS allows us to send the estimated delivery time to the customer. |
| Email (TIBCO plugin) | Tibco’s email plugin is easy to use and allows us to send an invoice to the customer. |
| Paypal | After the customer chooses their food “package”, we give them the option of paying either via cash or paypal. If they select payment by paypal, 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 transcation 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>). |
| 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. Nodejs utilizes Javascript. Javascript allows for callback 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 |

4. Systems walkthrough