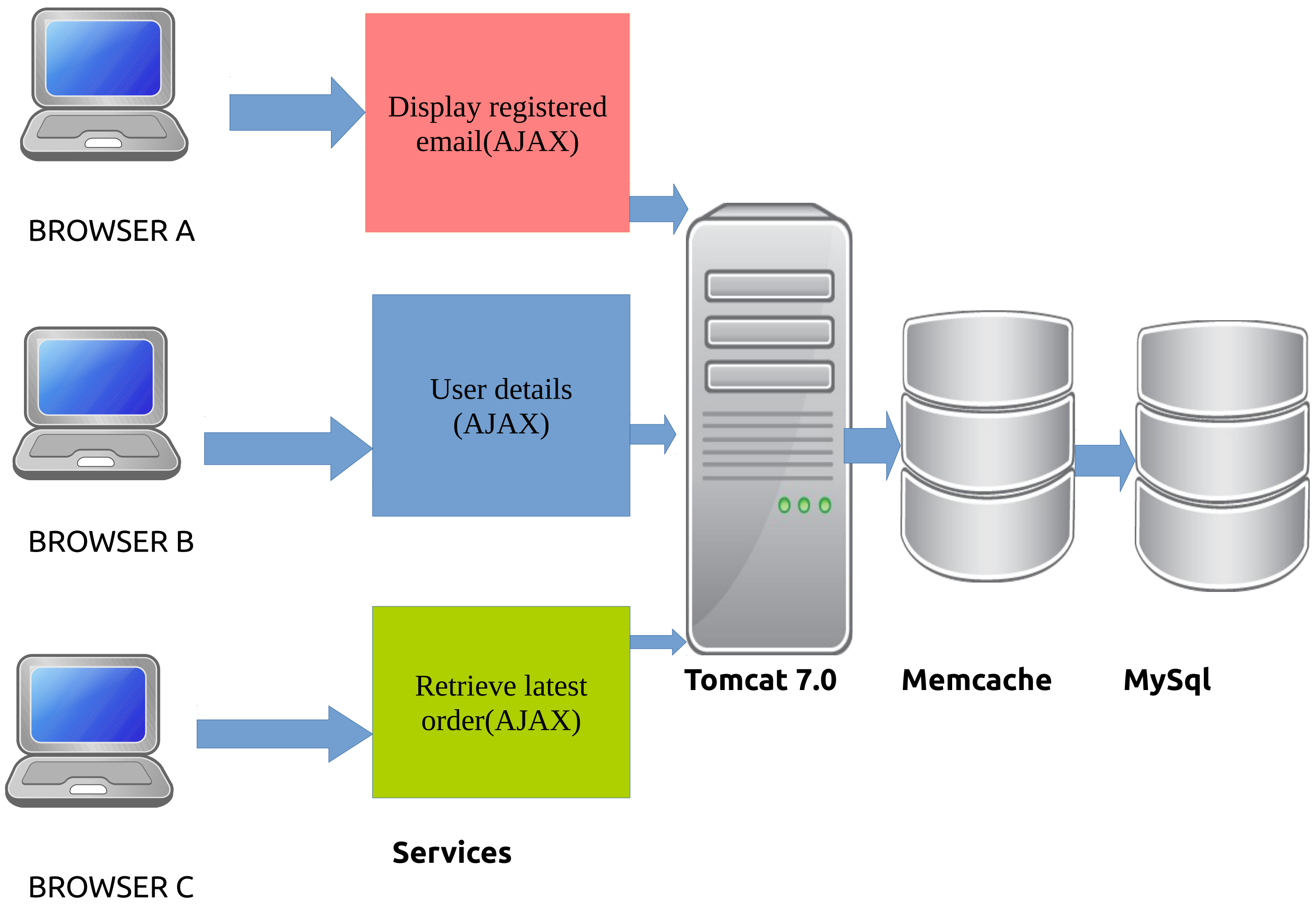
FOODZAPP

By:

GEORGE KOSHY

POOJA PURUSHOTHAMAN

ARCHITECTURAL DIAGRAM FOR FOODZAPP



Shown above is the architecture diagram of foodzapp. The functionalities are being served as services.

The application server used is Apache Tomcat 7, Memcache for distributed caching and MySQL as the database.

Modules implemented :-

1.We have used Spring Framework to write Restful Webservices and for MVC style. The reason behind choosing Spring was because it is widely used and great online support.

2.Maven was used for project building and resolving dependies.

3.Apache Tomcat and MYSQL was used as the database for the same reason mentioned above.

4.SSL/TLS has been implemented for encryption and better password protection.

5.Gzip compression of response messages received from the tomcat server has been enabled.

6.Other technologies that were considered initially were JAX-RS, Struts etc.

7.Memcache was used for caching. We did consider using REDIS for caching, as it is the current market leader and is highly scalable but later moved back to Memcache as REDIS was more into Linux based systems.

Three mandatory services provided:

1. **Display user details**:

Once the user logs into the application, the user can enter email id or their friends email id to retrieve related user information like Name , Phone Number and Address.

2.**Retrieve latest item bought and value**:
The user can see the amount they spent on their last order at Foodzapp.

3.**Display registered email**:

The new user registration page has a service where it checks if an email has already been registered with Foodzapp, and it returns email present or not present accordingly.

Other features are provided as functionalities.

Description of various functionalities:

1.Separate accounts are maintained for each user, and they are provided with username and password.

2.If a user doesnt have an account they can register at the new user registration page.

3.Once the user signs in to the web application choose the store page is shown.
The user chooses the store and is redirected accordingly.

4.The user gets to select from one of these food chains and select food items from a particular food chain and place an order.

5.On checking out the user is redirected to the payment page.
Here the user enters the his credit card information and finally checks out.

6.On successful completion of payment a receipt is sent to the users email id.

7.The user can then logout and close their session.

Problems encountered and their resolution

1.Great amount was spent on researching on an issue where AJAX request were not served properly with certian versions of spring releases. It was discovered late into the project and was resolved by using the appropriate stable spring versions.

2.Issues were faced with data binding using Jackson as mapper and core classes didnt match.

3.The idea of using a ORM framework came in late, by then all the required services were written, so we shunned the idea as we didnt want any major code change.

4.We tried implementing application server redundancy but major issues were seen in getting TLS SSL enabled in them, issues were seen in load balancing with TLS SSL.

5. After implementing ssl late in the project we realized that we had implemented all the content and Content Delivery Networks links using the http protocol, and even after wide searching we could find their https alternatives. So hoping appropriate considerations would be made.