THE UNIVERSITY OF TEXAS AT DALLAS

RAD’s Bid - An Online Service Bidding Platform

Project Report

Group Name

“RAD”

Authors

Raghav Mathur [rxm162130]
Amit Gupta [axg162930]
Deepan Verma [dxv160430]

A Web Programming Languages Project

Web Architecture

Rad's Bid is scalable, modular and a service oriented web application that let's any user post for a requirement of a service online and receive bids for its own posts and also bid for the services available, provided by other users. Our application focuses on technology items to sell/bid for but has scope for future enhancements and incorporation other items as well.

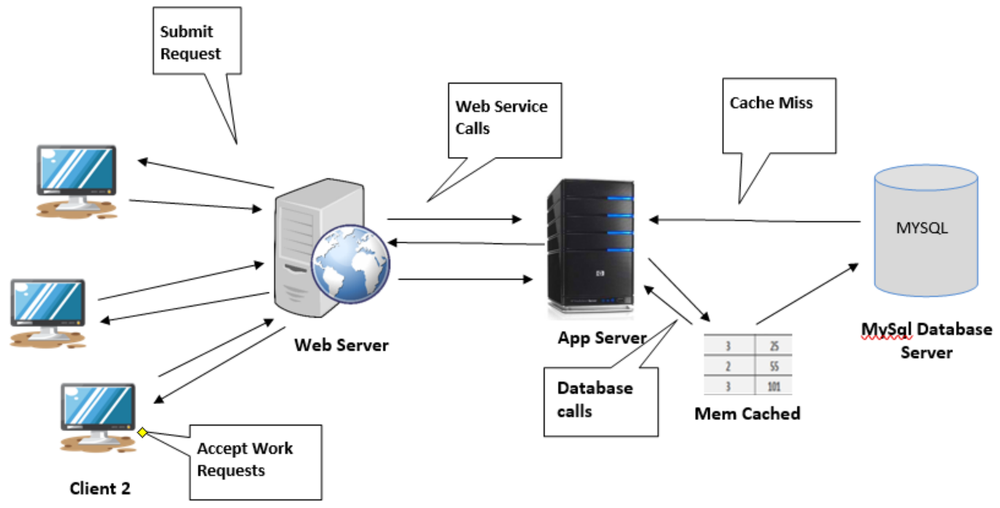

Fig. 1 Web Architecture

Here, the user visits RAD's Bid web application and logs in. When the user logs in, his credentials are passed along with the request method and these credentials are sent to the web server. The web server sends the data to the service running on the App Server which then looks for the user's username and password in the cache. If this information is unavailable in the cache, this corresponds to a cache miss and the web server looks for the information in the database. The user can then post a request for service which can be a physical thing such as a laptop, mobile, etc. After the user describes the nature of the request, it can put it up online with the original price it is willing to offer. As soon as the service is posted online. It can be seen by other users that can post their bids for offering that service (product). When a user chooses to search for a item, the service information like name, product information or type is passed along with the request method, the web server and app server finally looks for these items in the database. The product information is then formulated to a response object and it is sent back to the user by the web server as a response method.

Modules Used

Client Browser
The browser we are considering here is either Chrome Browser or Microsoft Edge. This browser is being run on a 64-bit machine.

Web Server
We used Apache Tomcat 8.0.39.
The web server hosts website and make calls to the Rest Services hosted on App Server to obtain data.

•The interaction between the web server and Rest Service; and browser and web server is secured using https protocol- TLS/SSL.
•The web server is developed using JSP/Servlets and deployed on Tomcat Server.
•The interaction requests and responses between the Webserver, Web service Server and Client occurs through a compressed channel.

Rest Service
•The Rest Service is the one which interacts with the database and provides data to the web server (servlets).
•The Rest Service first checks in the cache and then if there is cache miss, it interacts with database for fetching the data.
•The Rest Services are developed using Jersey Framework and deployed on Tomcat server.

Cache
This stores the frequently used data and makes data available faster to the rest service preventing expensive database calls.
•Caching is done using Memcached.
•Whenever there is cache miss, data will be populated from database.
•Whenever there is cache hit, data will be sent from cache. Which improves efficiency and performance of the application.

Database
We have chosen MySQL (Workbench 6.3)
The database will be called in case of cache miss.
•Whenever there is cache miss, it will first save data in cache and then return data to the rest Service.

Client-Server Communication Encryption:
Using TLS/SSL, we provided https protocols.

Technologies Used and its Features

Front End: HTML5, CSS, JavaScript, and jQuery
Other Technologies considered: As Responsive GUI was required we have used Bootstrap templates with HTML5 which is most comfortable and latest. Hence, other technologies need not be considered..
Reason:
The UI of the application was developed using HTML5, CSS, JS, and jQuery
We used sample bootstrap templates to build the application.

Database: MySQL 6.3 Workbench
Other Options considered: Oracle
Reason:
•MySQL is relatively light-weight, can be extremely fast when applications leverage architecture.
•Lots of features stay free as the database servers grow such as replication and partitioning.

•MySQL excels when high speed reads can be used for web, gaming, and small/medium data warehouses and OLTP systems.

Web Server: Tomcat Apache 8.0
Other Options: Apache Http Server.
Tomcat was preferred because of the following reasons:
•Supports multiple OS platforms
•Provides the Java Servlet and JSP support for dynamically served pages
•Works as a light-weight testing server
•Can be run in different modes to promote better performance

Rest Service Framework: JAX-RS Jersey Rest framework.
Other Options considered: Spring MVC
JAX-RS was considered because
•JAX-RS has the advantage of creating APIs that are simpler to create and digest messages for in different browsers and mobile devices.
•The Rest Services are deployed on Apache Tomcat server.
•JAX-RS targets the development of web services (as opposed to HTML web applications) while Spring MVC focuses in web application development.

Cache: MemCached Distributed Cache.
Other Options: EHCache
Reason:
•Memcached is a good option for implementing a distributed caching mechanism.
•It stores data in the format of key value pair. Currently, only single node is implemented.
•EHCache was used with Spring framework so cannot we used.

Client-Server Communication Encryption: Using TLS/SSL, we provided https protocols.
Reason:
Strong
Development Environment: JDK 1.8.0_111 (JSP/Servlet Framework)
Other Options: Python, PHP
Reason:
•Java is high performance compiled languages, with great support from a well known vendors, and entire ecosystem of companies that sell everything - from IDEs to libraries to automation tools.
•Supports multiple functionalities

Compression: GZip compression
Reason:
•Gives high and reliable compression.

Functionalities Implemented

1. *New User registration*:
A user, who is visiting RAD's Bid for the first time, can sign up using username and by creating his password. This allows a user to store his information like name, address, DOB, gender, telephone number etc. the username should be unique. The user can check for his information on his profile page.

2. *Existing User Login and Logout*:
A returning user can log in using his registered username and password. Once logged in, he can view profile, edit profile, post items he needs, bid for items he want to sell, search, add them to his cart by accepting a bid and buy items from shopping cart or just logout from the application. When a user is logged in, he has access to all his information and he can choose to edit them anytime.

3. *User Login Information-Date, Time, and Location of valid Login:*
For security purposes, after a user logs in, he is displayed with the time he logged in previously. This allows a user to keep track of his online activity. If an unauthorized user has logged in, the user can get to know about this unauthorized login as the last login time would be different from his actual last login time.
This is done by storing the login time and location information of a user to the database and every time a user logs in, before updating the login time, his valid login time and location is fetched and displayed.

4. *User Information Profile display and editing*:
A logged in user has access to his profile information that he once entered during his registration. At any point of time he can choose to update these details. This also lets the user to keep track of the products he added to his cart.

5. *Posting a Service*:
A User can post any service that he requires for all other users to see. He can describe the service or item that he requires by type, quality and quantity, price expected and address for the product to be delivered or picked up from.

6. *View Existing posts*
Use can see all the post he has already posted previously and keep track of all the bids on a particular product.

7. *Bidding a Service:*
A user can bid for the already available posts if he/she things they can fulfill the request. They mention their own price, quality of the product they have and its description for completing the request.

8. *View Bids on Items:*
The user can view all the bids that have come of all of his products a select the best bid.

9. *Navigation bar*:

The user has variety of options to do on RADS’s Bid. He can just view *home* or go for user *login*, or *search* for a few posts without logging in. Go to post page to post a service or product that you require or bid for already posted products. He can visit his profile page, enter shopping cart and also logout from any page.

10. *Search functionality:*
The user can search for all the posted items as well as the bidders that have bid for any of the services. The search result has filtering capacity and can be sorted on any column.

11. *Shopping cart and order purchase submission*: We have developed the shopping cart on our own without using any third-party libraries.

i. *Ability to add products/services*:
The user can add products to the cart. After he adds a product to the cart by selecting the bid he can either go ahead and checkout or go back to search to add few more products to his cart.

ii. *Ability to remove products from his cart*
If a user adds a product to his cart by mistake or feels that the decision of buying the product is not correct, he can always go delete the product from his cart.

iii. *Update Item Count*
Ability to add or reduce the number of items of a product from the cart. One the item count updates, the total prices of all the products also updates and the sum of all products can also be seen.

iv. *Email Notification upon Order Submission*
Ability of notifying both the bidder and the post user of the item, of the order confirmation when the order is placed.

12. *Error page if any unavailable page is accessed*:
If a user has accessibility to any page that is unavailable, a custom designed page with 404 error is displayed. This conveys the user that the requested page does not exist.

12. All items displayed in a Table:
All products can be viewed in a proper table with filtering and sortable properties on multiple columns.

13. *Email Confirmation:*
User gets notified when signing up or when placing the order from shopping cart.

Web Services

To implement RAD’s Bid, we have implemented two servers; one web server processes the requests requiring business logic running the services that fetch and insert data into the database and other non-database-requirement requests such as website code and transferring function calls to services running on the other server are handled by a second server. Both the servers we are using here are Tomcat Apache 8.0. They can also be run server Tomcat apache 7.0.

The following web services have been implemented:

1. *Sign Up Service*:
   The user is prompted to fill out a form to register onto the application. The user details such as name, address, phone number is asked for.
2. *Login Service*:
   The user enters the user name and password. This information is authenticated with the database and if a matching record is found, the user's first name is fetched.
3. *Update Profile Service*:
   Whenever a user wants to edit his profile, the user profile information is fetched from the database and the user can edit the information and the information is stored back to the database.
4. *Search functionality (Search Product Service and Search Bidder Service)*:
   Here the user's keywords are matched with the respective column (name or/and type) in the database and all the matching tuples are fetched. Using this functionality, the user can search for products and bids based on name of the product or type of the product/bidder. Also, to make it easy to look up for a product in the search result.
5. *Post Service:*
   Allows you to post a new product or service that you require. Inserts in the database the new product.
6. *Display Service:*
   Displays the products and services that are posted for bidding by fetching the information about the product from database.
7. *Bid Service*:
   Allows you to place a bid for a posted request, by specifying the name, type, and price of the service or the product you'll be offering.
8. *Display Bid Service*:
   Displays all the bids that have been posted along with bidder information and product id.
9. *Display Post Bids*:
   Displays all the information of the original request for the product/service along with the bidding information and their respective prices.
10. *Shopping cart*:
    a. Add to Cart Service:
    Once a user adds an item to his cart, this item is added to the cart table in the database for that user id. This saves the cart permanently in the database. Once the user pays for the entire order the user gets a confirmation about his payment and purchase submission.

b. *Delete Item from Cart Service*:
   Let’s you delete items from cart.
c. *Update Item Count*:
   Let’s you update the count of items you want in the cart of one specific item.
d. *View Cart Service*:
   Fetches data from cart and displays them.

11. *Email Service*:
    This service causes the user to be notified by an email on registration.

12. *Email Cart Service:*
    This service causes the user to be notified by an email on placing an order.

Problems Faced during Project:

1. *Email Response to clients*:
The email server configuration was not working correctly. We had to disable the security feature in our Gmail account which allowed us to send a mail from an external application.

2. *Handle Java Beans and Servlets together:*
The redirection from servlet to JSP was giving null pointer exception because of data in JSP being null. The solution was to set beans correctly in request object and call them in JSP pages using session object.

3. *Updating Shopping Cart items*:
The update count feature had problems of displaying the updates with the updated price. Due to multiple bids for the same item, the database query had to be changes and additional jquery script was added to perform the update count and delete function.

4. *Implementing Caching*
We had to begin from nothing and search for all the resources as we no knowledge of it. After days of research we could implement this functionality.

5. *Running security and compression on the same ports of server:*
It was giving error to run both techniques simultaneously on the same server, but with change in configurations of the servers and xml files, it was made possible.