Web Programming Languages Project

Team Name: CodeBreakers-APP

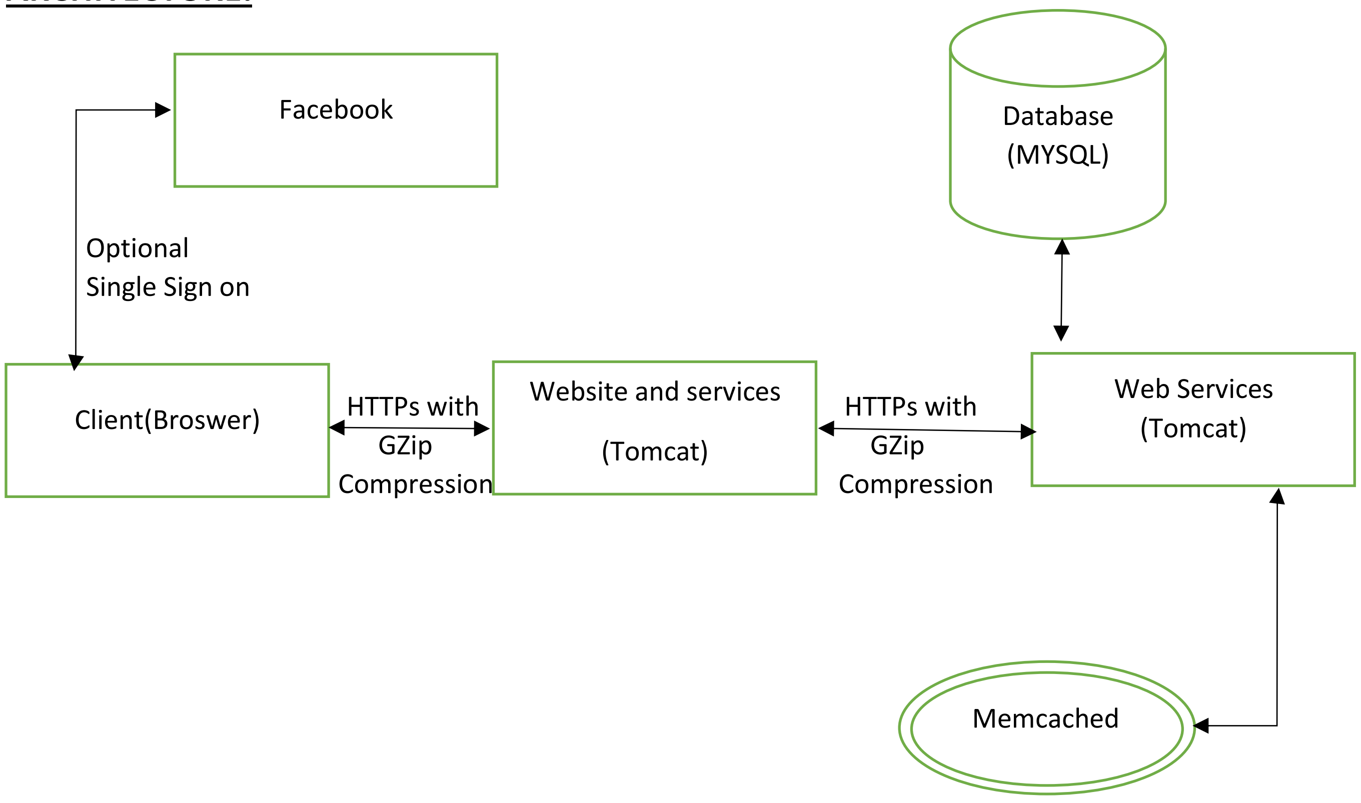
Team Members:

Abhiram Mandavalli	- axm167830
Pranathi Peri	- pxp162530
Parikshit Janagama	-pxj160730

INTRODUCTION:

We have designed an online Ecommerce system as a part of this project. This system gives the user ability to post for the items that user wants and user can bid for items (i.e. to sell) that somebody wants.

ARCHITECTURE:



- **Client:** The client makes a request from the browser to the website through HTTPS and the response is compressed using GZip, which is later uncompressed by the browser. The client can choose to create an account with our website or choose to sign in using Facebook and then use our website.

- **Website & Servlets:** The website and the servlets are hosted in a Tomcat Server running on port number 8443. The Tomcat server serves all the HTML, CSS and JSP files needed on the front-end. It also receives the requests from the clients, validates the session information and then forwards the request to the Web Services. All the requests and responses are served through HTTPS and the responses are compressed using GZip.

- **Web Services:** The Web Services run on a different Tomcat instance and listen on port 9443. The Web Services access the Memcached pool of servers to check if the requested data is available. If there is a cache hit, then the data is served from the cache. Else, the Web Services contact the Database to fetch the requested data. The requests and responses are served through HTTPS and compressed using GZip. The Web Services serve only the requests from the Servlets running on the other Tomcat instance. All other requests are turned down by default.

- **Memcached:** The Memcached pool of servers are used to cache the data, so that it avoids the overhead of contacting the Database every time to fetch the data. They store the data in memory and are typically much faster than the Database. We have used a single Memcached server and is deployed on the same machine.

- **Database**: The Database is hosted on the MySQL DB.

TECHNOLOGIES:

Front-end:	Back-end:
• HTML	• Servlets
• CSS	• Jersey framework
• jQuery	• JSON
• JavaScript	• Database
• JSON	• Memcached
	• HTTPS
	• GZip Compression
	• Single Sign-On (OAuth)

OVERALL DESCRIPTION:

Website is the first point of contact for the user. A user who wishes to register in the website, login or access all the functionalities accesses the website by using a browser. Initially, the user lands on home page of the website which has an image and navigation bar. The user can navigate through the site only after a valid login or registering as a new user and then logging in with the credentials.
We have two tomcat servers running. Any request by the user will be redirected to that particular servlet which handles that request. This servlet resided in the same server as the website. The servlet then calls upon the related rest service. This rest service resides on another tomcat server (app server). All the necessary database access is done within this rest service and the results are sent back to the servlet that called the service which handles the output and calls the necessary pages to be displayed.

For example, consider the case of new user registration. Here, the user enters his details in the newuser.jsp page. The details are sent to newuserservlet through a post request. This servlet calls the "newuserservice" meant for new user registration. This service contacts the database through hibernate framework and the result (whether the new user is registered or not) is sent back to the servlet. The servlet then calls either the newuser.jsp page (in case of failure) or login.jsp page (in case of success).

FUNCTIONALITIES OF THE SYSTEM:

- **New user registration:** New User can register using the signup page and access his/her account. User must enter First name, Last name email-id and password to create an account.
- **Existing user login and logout**: Existing users can access their account via login page. They can access different pages and functionalities like View all Posts, Make a new Post, Bid for an item, add to cart, accessing their user profile, contacting the customer care and logout.
- **User profile information display and editing:** User profile can be viewed and updated in Edit Profile page. User can update their First Name, Last Name, and password here.
- **User login information:** User login information consisting of User Name, Last time, date, and location of valid login.
- **Ability to post items that you want**: The User can post for items that he wants via selecting the image "Clicking Here to Post "from the image slider or by selecting the entry of Post button "Make a new post" in the navigation bar.
- **Ability to bid for items:** User can bid for any item by viewing the posts and selecting the item for which you want to bid for. This can be done via selecting the image "View All Posts "from the image slider or by selecting the entry of Post button "View All Posts" in the navigation bar.

- **Page listing all the bids for your post:** User can view the bids for his posts via selecting the entry of Post button “View My Posts” in the navigation bar.
- **Search for items that you would like to bid:** User can bid for any item by viewing the posts and searching for the item you want to bid for in the search box. This can be done via selecting the image “View All Posts “from the image slider or by selecting the entry of Post button “View All Posts” in the navigation bar.
- **Table display:** Results are displayed in a sortable table and Search results filtering capabilities are implemented. This functionality can be seen in “View my posts”, “View all posts”, “View Bids” pages.

- **Shopping cart and order purchase submission**: We have added a shopping cart functionality which can add items, delete items and update item count and on checkout will send mails to the bidder and purchaser.
- **Unavailable Page:** Accessing any unavailable page should retrieve a pretty and generic 404 page.

WEBSITE:

The majority part of the website is made using the Bootstrap framework.

We start with a index page. After a user logs in, he has access to different pages. These pages are linked on a navigation bar at the header. The pages are as follows:

1. **Home**

    - This is the home page which has a navigation bar at the top with 5 navigation links to different pages and a logout button.
    - We have the User Info displayed in the home page and a translucent image slider in the background.

2. **Sign Up**

    - This page will take the user to new User registration page.

3. **Posts**
    - The post button is a drop down which lists the different options to the User.
    - The User can navigate to one of the pages:
      Make a new Post: Lets the user make a new post.
      View My Posts: Lets the user to view the Posts made by a certain user.
      View All Posts: Lets the user view all the Posts.

4. **Login:**

   - The Login button is a drop down which lists the different options to the User.
   - The user can navigate to one of the pages:
     Login: This will take the user to login page.
     View Orders: This will display a error page designed by us.
     Edit Profile: Displays User Info and lets them to edit it.

5. **Shopping cart**
   - By clicking on cart the User is directed to cart page which displays the different items in the cart and lets user to delete and update the item count and check out.
6. **Contact us**

   - It is a form which asks the user for his Email and comments he has. On submitting the form, the user gets an acknowledgement on his email that the message has been received, if it is a valid email.

WEB SERVICES:

The Web Services are hosted on the Tomcat instance running on port number 9443. All the Web Services are REST based.

- **Frameworks used:**

  1. Jersey: For deploying REST based Web Services.
  2. Log4j: To log the messages.
  3. Hibernate ORM: ORM framework to handle the interaction between the Database and the Web Services.

- **Libraries used:**

  1. PJL Compression: To support GZip Compression.
  2. Memcached Java Client: To communicate with Memcached

- **List of Web Services:**

  1. NewUserService:
     - Takes the user details and creates a new user

2. LoginService:

   - The login service is user for authenticating the user. In this service the user's last login will be updated and the number of failed attempts will also be updated (only in case of failure).
   - Memcached is implemented here. As soon as a request for login is made to the login service (from the login servlet), the memcache is checked to see whether the user email is present. In case the user email is present, then it is a hit. Otherwise it is a miss.
   - In case of a miss, the database is accessed and the retrieved username and password are entered into the memcache.
   - In case of a miss, the database is accessed and the retrieved username and password are entered into the memcache.
   - Once the user is authenticated, the database is accessed to update the last login. Also, in case of failure, the database is accessed in order to update the number of failed login attempts.
   - It also supports Single Sign-On using Facebook. If the user is authenticated using Facebook, then a profile information is created for the first time, from details obtained from Facebook.

3. PostService:
   - Receives the post details from the servlet and stores in the post table in the database.
   - It also fetches the requested Post details from the database when the user requests to view all posts or view posts .

4. BidService:
   - Receives the bid details from the servlet and stores in the bid table in the database.
   - It also fetches the requested Bid details from the database when the user requests to view all bids made for his post .

5. CartService:
   - Processes the add to Cart requests from the Cart Servlet and stores it in the database.
   - It also fetches the Cart details from the Database when user request to display the cart items.
   - It also updates cart item count.
   - It also deletes the item from the cart when user does not need it.

        - It also removes from the cart once the checkout is done.

- **Working:**

    1. The services process requests only from the Servlets. Any request made to it directly will be turned down. A secret key is shared between the Servlets and the Web Services, which is added to the header of the request made by the Servlet. If the secret key exists and if it is valid, then the Web Service processes the request.
    2. For all the fetch operations, the Web Services look into the Memcached for the data. If there is a cache hit, then the data is returned by the cache. Otherwise, the Web Services query the Database using Hibernate ORM Framework and places the data in the cache before returning the data, so that it could be used for subsequent calls.
    3. The Web Services compress the data in GZip format and sends it to the Servlets.

SERVLETS:

The Servlets check if there is a valid session associated with a user request by checking if a user email is present as session attribute. The Servlets forward the valid requests to the appropriate Web Services and return the data given by the Web Services.

- **Frameworks used:**
    1. Jersey Client: For making a REST call to the Web Services.
    2. Log4j: To log the messages.

- **Libraries used:**
    1. PJL Compression: To support GZip Compression.
    2. Memcached Java Client: To communicate with Memcached.
- **List of Servlets:**
    1. Login
    2. newuser
    3. cart
    4. post
    5. getposts
    6. bid
    7. getbids
    8. Email

Other required functionalities:

1. **SSL encryption**:
    a. We enabled SSL in Apache Tomcat by using the following configuration:
        i. <Connector SSLEnabled="true" clientAuth="false" keystoreFile="C:\keystore" keystorePass="changeit" maxThreads="150" port="8443" protocol="org.apache.coyote.http11.Http11Protocol" scheme="https" secure="true" sslProtocol="TLS"/>
    b. We used keytool for generating the keystore file by using a keystorepass.
    c. This keystore file is stored in the folder where server.xml can access the file.
    d. After adding the configuration, we can access the website by using https protocol and on port 8443
2. **Compression:**
    a. We used compression for the requests between servers. We also used this for web services.
    b. We used compression for the requests between servers. We also used this for web services.
    c. Following is the configuration used:
        i. <Connector compressableMimeType="text/html, text/xml, text/plain, text/css, text/javascript, text/json, application/x-javascript, application/javascript, application/json" compression="on" compressionMinSize="2048" connectionTimeout="20000" noCompressionUserAgents="gozilla, traviata" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>

SINGLE SIGN-ON:

The user is given the flexibility to use his/her Facebook account to use our service using OAuth. A Facebook login button is added in the user login page. On clicking this button, a pop-up opens asking the user to login to Facebook. We request the user’s email address from Facebook during the login process. Once the user is successfully logged in, we obtain the email address of the user and map it the corresponding user in our system, if the user has already used our system. If no such mapping is found, then a new user profile is created in our Database with this mapping.

PROBLEMS ENCOUNTERED:

- Integrating several parts of the project was also a problem. Since we split up the project into parts, integrating all the parts was a problem because two of us had the one tomcat running on 8080 while the other had it running on 9080. Simply, forgetting to change the ports numbers in several places of the application caused a lot of trouble.
- We faced problems while implementing HTTPS. While changing our existing project from the 8080 ,9080 ports to 8443 ,9443 respectively we faced a SSL hand shake exception even though we had implemented the HTTPS properly.

CONCLUSION:

Through the implementation of this project, we have got a good understanding of the real world use of the web technologies and languages discussed in the class over the entire semester. Incorporating technologies like HTTPS, Memcached and Frameworks like Hibernate ORM helped us to realize the actual need and purpose of these in real world projects.