WEB PROGRAMMING LANGUAGES PROJECT

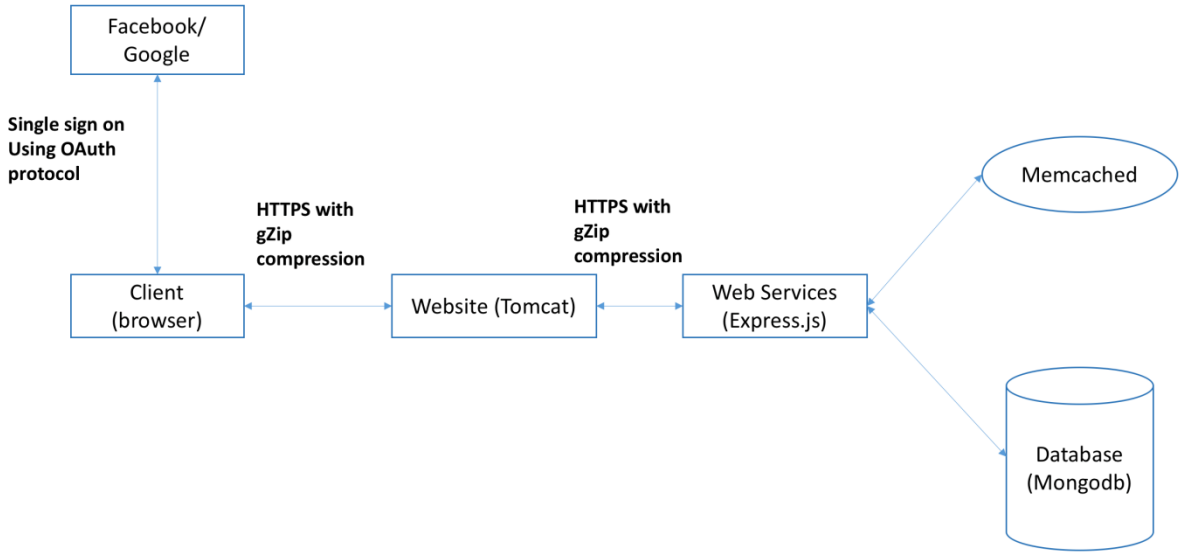
TEAM INTERLACE :

Shreya Kodolikar (sgk160130)
Aditi Ghamandi (axg165830)
Krishna Chaitanya Kaushik Hari (kxh161230)

INTRODUCTION:

We have designed an online bidding system as a part of this project. This system enables the user to post a bidding request and allows other users to bid for the request. The user selects the bid as per his requirements and adds it into the shopping cart. After the checkout, both the bidder and user receive a mail.

ARCHITECTURE:



- **Application Stack: MEAN Stack:** MEAN is a collection of JavaScript-based technologies — MongoDB, Express.js, AngularJS, and Node.js — used to develop web applications. From the client and server sides to databases, MEAN is a full-stack development toolkit.

- **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 create an account with our website or sign in using Facebook or Google for further access.

- **Website:** The website is hosted in a Tomcat Server running on port number 8443. The Tomcat server serves all the HTML, CSS and AngularJS 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 3000. 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 for the requested data. The requests and responses are served through HTTPS and compressed using gZip.

- **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. A single Memcached server is used and deployed on the same machine.

- **Database**: The Database is hosted on the MongoDB server.

- **ORM framework**: This has been implemented using Mongoose ODM(Object Document Modelling).

TECHNOLOGIES:

Front-end:	**Back-end**:
• HTML • CSS • jQuery • JavaScript • JSON • AngularJS	• NodeJS • ExpressJS • Mongoose • JSON • MongoDB Database • Memcached • HTTPS • gZip Compression • Single Sign-On (OAuth)

FUNCTIONALITIES OF THE SYSTEM:

- User Registration: User can create a new account using the signup page or sign in using Facebook or Google. User has to enter his name, email-id and password to create an account.
- Existing user login and logout: The existing user can access his account using login credentials such as username and password.
- User profile information display and editing: User profile can be viewed and updated in My Profile page. User can update his name and email id here.

- User login information: The session is created every time the user logs in. User's time, date and location of login is recorded. Information about the last login can be viewed by hovering on the 'last login'.
- Ability to post items that you want: After logging in, the user can post a request for an item. He needs to specify following information
    1. Title
    2. Description
    3. Quantity
    4. Price
- Ability to bid for items: Users can bid for the items that has been posted by other users. Search will display all the post requests from other users. The user can search for the posts according to the title, quantity, price and name of the user who has posted the item and then put bids for those posts.
- Page listing all the bids for your post: The user can view the bids on his post by clicking on my posts. He can add those bids to the cart and checkout.
- Search for items that you would like to bid for: The user can search for the post requests from other users and can bid for the ones he want.
- Table display: All the results of the search and posts are displayed in a tabular form. The search can be used to filter out the results. It also has the ability to sort the posts according all the attributes.
- Shopping cart and order purchase submission:
    1. ability to add items
    2. ability to remove items
    3. ability to update item counts
    4. checkout will allow the user to purchase that item.
- Confirmation mail is sent to the user and the bidder after the checkout.
- Any unavailable page will generate a pretty page that displays not found error.

SINGLE SIGN-ON:

The user is given the flexibility to use his/her Facebook or Google account to access our service using OAuth. The user login page contains additional Facebook and Google login buttons. On clicking any of these buttons, a pop-up to login to Facebook or Google appears. It will request for the username and password for Facebook or Google during the respective login process followed by the permission from the user in order to access his profile information. Once the user is successfully logged in, the user's name, profile picture and email address of the user is obtained and mapped to the corresponding user in the system, if the user has already used the system. In case of no match found, a new user profile is created in our Database for that user.

WEBSERVICES:

The Web Services are hosted on the Tomcat instance running on port number 8443. All the Web Services are REST based.

- **Frameworks used:**

    1. Express: For deploying REST based Web Services and compression of responses.
    2. Mongoose ODM: ODM framework to handle the interaction between the Database and the Web Services.

- **Libraries used:**

    1. nodemailer: For sending mails.
    2. memcached: To communicate with Memcached
    3. dateformat: For formatting the date and time values
    4. cors: For accepting request from other domains
    5. logger: For logging to console
    6. request: For sending http request

- **List of Web Route Methods:**

    1. POST /auth/login:
        - Takes username, password and location and validates the credentials and sends a JSON web token to the client.
    2. POST /auth/signup:
        - It takes the email, password and user's name and location and stores the credentials in database and sends a JSON web token to the client.
    3. POST /auth/google:
        - It validates permission with Google and and gets user information and stores it in database and sends a JSON web token to the client
    4. POST /auth/facebook:
        - It validates permission with Facebook and and gets user information and stores it in database and sends a JSON web token to the client
    5. POST /api/me:
        - It will update the user's information in the database.
    6. GET /api/prevloginInfo:
        - It returns user's previous login information.
    7. GET /api/post:
        - It returns all the posts that are not posted by user.
    8. GET /api/myposts:
        - It returns all the posts posted by the user.
    9. POST /api/post:

- It stores post information.

10. GET /api/post/{id}:
    - It returns the details of the posts with the id.
11. POST /api/{postid}/bid:
    - It will store bids for the post with id postid.
12. GET /api/bids/{postid}:
    - It returns all the bids for the post with id postid.
13. GET /api/isMyPost/{postid}:
    - It returns whether the post with id postid is the user's post.
14. GET /api/bid/{bidid}
    - It returns the bid with id bidid.
15. POST /api/cart:
    - It saves user's cart.
16. GET /api/cart:
    - It gets user's cart.
17. POST /api/checkout:
    - It saves the cart information to database as a record. Also sends mails to bidder and user.

PROBLEMS ENCOUNTERED:

- While working on the single sign-on module using Google
- While adding SSL/TLS to App Server
- Integrating the pretty generic 404 page
- Integrating Memcached server

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 Mongoose ORM enabled us to get hands on experience and learn the purpose of these in real world projects.