Maintenance Report

1. **Introduction**.

The purpose of this document is to present the fundamental principles that will guide the Software Maintenance and the User Support tasks within the project.

The Software Maintenance task is responsible to coordinate the continuous maintenance of the middleware components developed within the project and included in the distribution, preserving at the same time their stability in terms of interface and behavior, so that higher-level frameworks and applications can rely on them.

2. Security

The application uses various security mechanisms to protect user privacy:

2.1. Hashed and Salted Passwords

To prevent possibilities of leaking passwords from the database, we stored the salted and hashed password. Salting is the concatenation of a random string of characters to the passwords. The resulting string is hashed. Hashed passwords are hard to crack because they are scrambled versions of themselves.

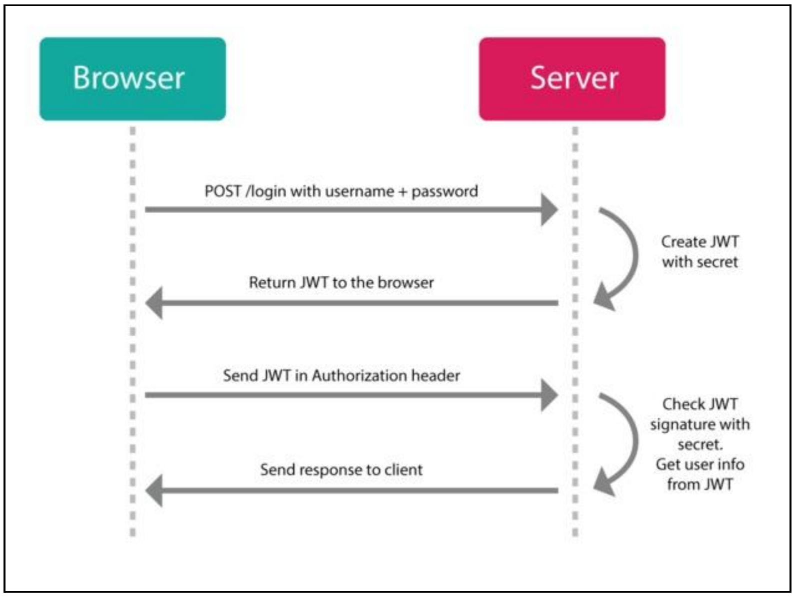
Large salts also protect against certain methods of attack on hashes, including rainbow tables or logs of hashed passwords previously broken. Both hashing and salting can be repeated more than once to increase the difficulty in breaking the security.

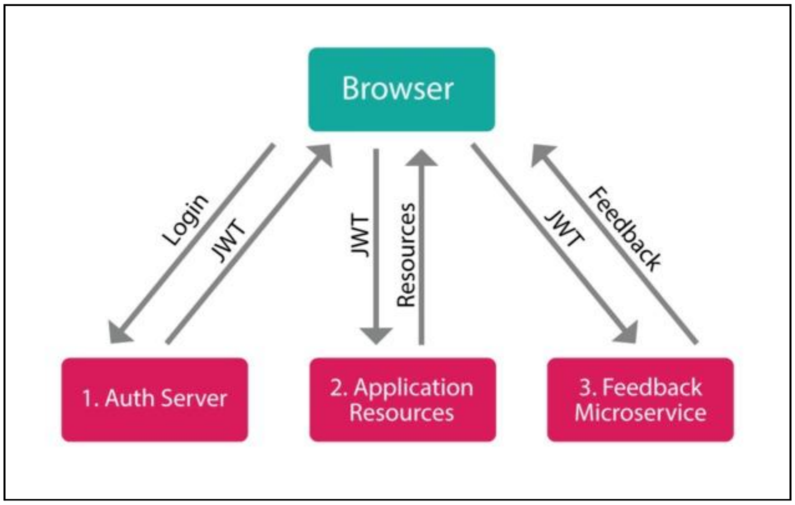
2.2 JWT Token Authentication

JWT Tokens to authorize users to make the browsing along the application seamless while maintaining high security and privacy for the user. JWT tokens are a compact and self-contained way for securely transmitting information between parties as a JSON object. The user credentials are embedded into a string encrypted with a private key at the server side. The token will be safe with the user. Each JWT token can have a time of expiry, after which the token becomes invalid.

This way, the client need not pass credentials and be authenticated for each request made to the server, but rather send just the token sent to the client on logging in.

JWT token also prevents session high-jacking where the user's credentials are logged, or cached in an intermediate server and exploited. Also no one can pretend to be the client after the original client has logged in since only the client has the token.


Browser
Server
POST /login with username + password
Create JWT with secret
Return JWT to the browser
Send JWT in Authorization header
Check JWT signature with secret. Get user info from JWT
Send response to client



2.3 HTTPS Connection

To maintain secure connection and communication between client-server for text messaging and authorization we use HTTP secured connection. It prevents the Man-in-the-Middle (MitM) attacks, in which hackers intercept the network traffic and steal confidential data like passwords, Aadhar numbers, birthdays, etc.

We generated a self-signed SSL certificate and passed it to the user. After the TLS handshake, the network traffic exchanged between the server and clients are encrypted by public-private key encryption.

2.4 Permission Management

Each group in the application has various roles that its members can define. Roles with a fixed set of permissions help managing thousands of users in a conversation, thus along with the authorizing middleware we also provide the Permission middleware.

3. Portability

After deploying the software to the cloud we hope to maintain a Node web server and a Mongo DataBase server. The client side software is to be installed as an apk from the client side. DNS servers are to be maintained and secured too.

After deploying we continue to maintain the servers by continuously monitoring traffic and testing various edge cases if they ever arise. Any new code is to be heavily tested on a test server (for testing) before actually pushing the code to production.

****