Documentation for JWT Authentication in SalesDashboard with PHP Validation

This documentation explains how the login, logout, refresh token, and JWT token generation functionality works for both the **App** and **Web** in the SalesDashboard system. The system utilizes JWT (JSON Web Token) to authenticate and authorize users, and PHP validation is used for verification.

1. Login - App

The `loginApp` function handles the login process for mobile applications (iOS/Android). It takes in `login_into`, `mob`, and `phpToken` as parameters.

Process:

Input Validation:
The mobile number (`mob`) is decoded using a custom decryption method (`myDecode`).
It ensures the mobile number is valid and not empty.
Employee Retrieval:
The employee's details are fetched from the database using the decoded mobile number.
It checks for valid employee credentials and verifies the associated `phpToken`.
Token Validation:
The `phpToken` is base64-decoded and validated against the database.
If the token is valid and hasn't expired, a new JWT is generated.
JWT Generation:
A JWT token is created for the employee using the `generateToken.forSDBApp()` function, and it's returned as part of the login result.

Response:

`emp_id`: Employee ID
`emp_name`: Employee Name
`JwtToken`: JWT Token

2. Login - Web

The `loginWeb` function is used for user authentication for the web platform. It takes `username` and `password` as inputs.

Process:

Input Validation:
Checks if the `username` and `password` are provided.
Employee Retrieval:
Queries the database for the employee details using the `username`.
Retrieves the stored hashed password for the user.
Password Validation:
The provided password is hashed using MD5 and compared with the stored hashed password.
If the passwords match, a new JWT token is generated using the `generateToken.forSDBWeb()` function.
Token Generation:
A new JWT token is created and returned.

Response:

`emp_id`: Employee ID
`emp_name`: Employee Name
`JwtToken`: JWT Token

3. Refresh Token - App

The `refreshTokenApp` function is responsible for refreshing the JWT token for the mobile app.

Process:

Authorization Header Extraction:
Extracts the JWT token from the `Authorization` header.
Token Validation:
The token is decoded, and the `emp_id` and `application_id` are extracted.
The token's validity is verified by checking if it has expired.
Token Expiry Check:

A database query is executed to check if the token is valid and has not expired.
JWT Generation:
If the token is valid, a new JWT is generated using the `generateToken.forSDBApp()` function.

Response:

`jwtToken`: New JWT Token

4. Refresh Token - Web

The `refreshTokenWeb` function is used to refresh the JWT token for the web platform.

Process:

Authorization Header Extraction:
Extracts the JWT token from the `Authorization` header.
Token Validation:
Decodes the token to extract the `emp_id` and `application_id`.
Verifies if the token is expired using `jwt.verify()`.
Token Expiry Check:
A database query checks if the token is valid and exists for the employee.
JWT Generation:
If the token is valid, a new JWT is generated using the `generateToken.forSDBWeb()` function.

Response:

`jwtToken`: New JWT Token

5. Logout - App

The `logoutApp` function logs out the user from the mobile application.

Process:

Authorization Header Extraction:
Extracts the JWT token from the `Authorization` header.

Token Validation:
Decodes the JWT token and checks if it's valid.
The token's validity is verified by checking if it exists in the `tbl_auth_mgmt` table and if it is still active.
Token Invalidation:
If the token is valid, it is marked as invalid by updating the `expiry_status` in the database.

Response:

`msg`: "Logout successful"

6. Logout - Web

The `logoutWeb` function logs out the user from the web platform.

Process:

Authorization Header Extraction:
Extracts the JWT token from the `Authorization` header.
Token Validation:
Decodes the JWT token and verifies its validity.
Checks if the token exists in the database and is valid.
Token Invalidation:
If the token is valid, it is invalidated by updating the `expiry_status` in the database.

Response:

`msg`: "Logout successful"

Error Handling

Unauthorized Access: If the token doesn't match the expected details, an "Unauthorized Access" error is thrown.
Session Expired: If the token has expired, a "Session Expired" error is returned.
Invalid Token: If the token is not valid, an "Invalid JWT Token" error is thrown.

Missing Authorization Header: If the authorization header is missing, an error is thrown.
Incorrect Credentials: If the user provides incorrect credentials, an error is returned stating "Please enter correct credentials."

JWT Token Format

The JWT token consists of:

`emp_id`: Employee ID
`application_id`: Application ID (for distinguishing between mobile and web)
`exp`: Expiry timestamp

Security Considerations

Token Expiry: Ensure that JWT tokens have an expiration time (`exp`), after which users must re-authenticate or refresh their tokens.
Token Invalidation: Tokens must be invalidated during logout or when the user session is explicitly terminated.

Conclusion

This JWT authentication mechanism ensures secure login, session handling, and token refreshing for both mobile apps and web platforms. By validating the JWT token with each request, we ensure that users can only access the application if their session is active and their credentials are valid.