```
    public function login(Request $request)
    {
        try {
            //Validate request input
            $request->validate([
                'email' => 'required|email',
                'password' => 'required',
            ]);

            // Find the user by email
            $user = User::where('email', $request->email)->first();

            if ($user) {
                if (Hash::check($request->password, $user->password)) {
                    //Generate token based on usertype
                    $token = null;
                    switch ($user->user_type) {
                        case 'Administrator':
                            $token = $user->createToken('admin-token',
['Administrator'])->plainTextToken;
                            break;
                        case 'Supervisor':
                            $token = $user->createToken('supervisor-token',
['Supervisor'])->plainTextToken;
                            break;
                        case 'TeamLeader':
                            $token = $user->createToken('teamleader-token',
['TeamLeader'])->plainTextToken;
                            break;
                        case 'Controller':
                            $token = $user->createToken('controller-token',
['Controller'])->plainTextToken;
                            break;
                        case 'DeanHead':
                            $token = $user->createToken('dean-token',
['Deahead'])->plainTextToken;
                            break;
                        default:
                            $response = ['message' => 'Unauthorized'];

                            $this->logAPICalls('login', $request->email,
$request->all(), $response); // Log API call
                            return response()->json($response, 403);
                    }

                    $sessionResponse = $this->insertSession($request-
>merge(['id' => $user->id]));

                    //Log successful login
                    $response = [
                        'isSuccess'=> true,
                        'message' => ucfirst($user->user_type) . ' logged in
successfully',
                        'token' => $token,
                        'user' => $user->only(['id', 'email']),
                        'user_type' => $user->user_type,
                        'session' => $sessionResponse->getData(),
                    ];
                    $this->logAPICalls('login', $user->id, $request-
>except(['password']), $response);
                    return response()->json($response, 200);

                } else {
```

```
                    $response = ['message' => 'Invalid credentials'];
                    $this->logAPICalls('login', $request->email, $request-
>except(['password']), $response);
                    return response()->json($response, 401);
                }
            } else {
                $response = ['message' => 'Invalid credentials'];
                $this->logAPICalls('login', $request->email, $request-
>except(['password']), $response);
                return response()->json($response, 401);
            }
        } catch (Throwable $e) {
            $response = [
                'message' => 'An error occurred',
                'error' => $e->getMessage() // Return the specific error message
            ];
            $this->logAPICalls('login', $request->email, $request->all(),
$response);
            return response()->json($response, 500);
        }
    }

//LOGOUT

    public function logout(Request $request)
{
    try {
        $user = Auth::user();

        if ($user) {
            Log::info('User logging out:', ['email' => $user->email]);

            // Find the latest session for this user with a null logout_date
            $session = Session::where('user_id', $user->id)
                ->whereNull('logout_date')
                ->latest()
                ->first();

            if ($session) {

                $session->update([
                    'logout_date' => Carbon::now()->toDateTimeString(),
                ]);
            }


            if ($user->currentAccessToken()) {
                $user->currentAccessToken()->delete();
            }

            $response = ['message' => 'User logged out successfully'];
            $this->logAPICalls('logout', $user->id, [], $response);

            return response()->json($response, 200);
        }

        // If no authenticated user is found, return unauthenticated response
        return response()->json(['message' => 'Unauthenticated'], 401);
    } catch (Throwable $e) {
        return response()->json([
            'message' => 'Failed to log out',
            'error' => $e->getMessage(),
        ], 500);
    }
```

```
}
```