# API Documentation

## Base URL

[http://127.0.0.1:5000](http://127.0.0.1:5000/)

## Authentication & User Management

### Register a New User

**Endpoint:** POST /register

**Description:** Registers a new user and sends a verification code to their email.

#### Request Body:

#### If email exist in database :

return a.jsonify({"error": "Email is already registered!"}), 400

#### else :

{

"name": "Hagar Mohamed", "email": "[hagar@example.com](mailto:hagar@example.com)", "phone": "123456789",

"password": "securepassword",

“company\_name” : “eden”

}

#### Response:

{

"message": "Verification code sent!"

}

### Confirm Registration

**Endpoint:** POST /confirm

**Description:** Confirms the user's email by verifying the code sent.

#### Request Body:

{

"email": "[hagar@example.com](mailto:hagar@example.com)", "code": "123456"

}

#### Response:

{

"message": "Registration successful!"

}

### Login

**Endpoint:** POST /login

**Description:** Authenticates a user and returns a JWT token upon successful login.

#### Request Body:

{

"email": "[hagar@example.com](mailto:hagar@example.com)", "password": "securepassword"

}

#### Response:

{

"message": "Login successful!", "token": "your\_jwt\_token\_here"

}

### Forgot Password

**Endpoint:** POST /forgot-password

**Description:** Sends a password reset code to the user's email.

#### Request Body:

{

"email": "[hagar@example.com](mailto:hagar@example.com)"

}

#### Response:

{

"message": "Reset code sent!"

}

### Reset Password

**Endpoint:** POST /reset-password

**Description:** Resets the user's password using the verification code.

#### Request Body:

{

"email": "[hagar@example.com](mailto:hagar@example.com)", "code": "123456",

"new\_password": "new securepassword"

}

#### Response:

{

"message": "Password updated successfully!"

}

1. **Google OAuth Authentication Endpoint:** GET /auth/google **Description:** Initiates Google OAuth login.

**Response:** Redirects the user to Google authentication.

### Google OAuth Callback

**Endpoint:** GET /auth/google/callback

**Description:** Handles the callback from Google OAuth and retrieves user information.

#### Response:

{

"message": "Login successful!", "name": "Ali Mohamed", "email": "[ali@example.com](mailto:ali@example.com)",

"profile\_picture": "profile\_image\_url"

}

### Check Token Validity

**Endpoint:** GET /check\_token

**Description:** Checks if the provided JWT token is valid.

#### Headers:

Authorization: Carry your\_jwt\_token\_here

#### Response:

{

"message": "Token is valid!" }

#### Error Response:

except a.jwt.ExpiredSignatureError:

        return a.jsonify({"error": "Token has expired!"}), 401

    except a.jwt.InvalidTokenError:

        return a.jsonify({"error": "Invalid token!"}), 401

    except Exception as e:

        return a.jsonify({"error": f"An error occurred: {e}"}), 500

### Get User Data

**Endpoint:** GET /user\_data

**Description:** Retrieves user data based on the JWT token.

#### Headers:

Authorization: Carry your\_jwt\_token\_here

#### Response:

#### If user is Owner :

user = {

                    "name": user\_data[0],

                    "email": user\_data[1],

                    "phone": user\_data[2],

                    "user\_name": user\_data[3],

                    "company\_name": user\_data[4],

                } if user\_data else None

                packdge = [

                    {

                        "p1": row[0], "p2": row[1], "p3": row[2],

                        "year": row[3], "month": row[4], "price": row[5]

                    } for row in package\_data

                ]

                return a.jsonify({"user": user, "packdge": packdge}), 200

#### else :

{

"name": "Hagar Mohamed", "email": "[hagar@example.com](mailto:hagar@example.com)", "phone": "123456789",

"user\_name": "Hagar Mohamed+id",

“company\_name” : company\_name

}