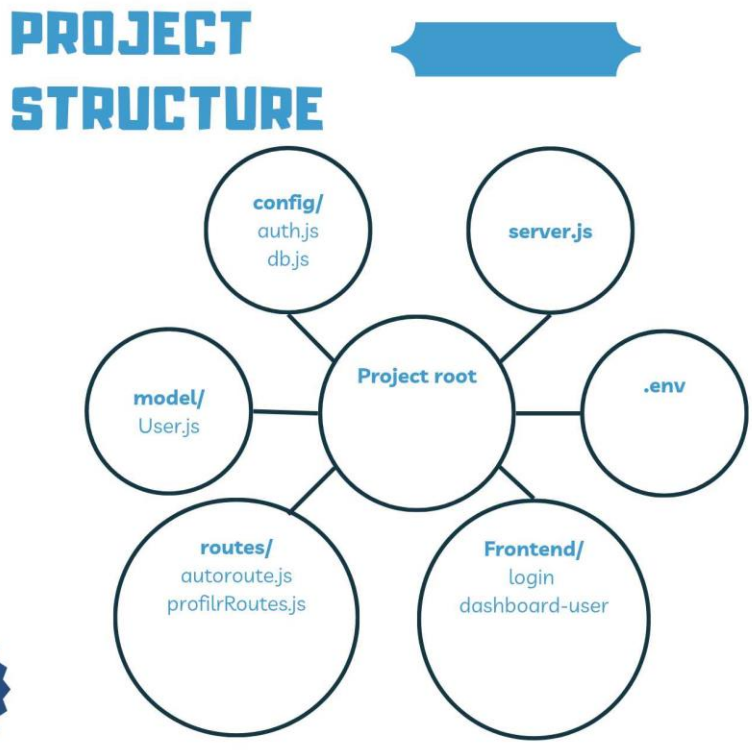
USER AUTHENTICATION & PROFILE API STACK

INTRODUCTION

This document provides a comprehensive guide to the **User Authentication & Profile API**, including setup, configuration, and endpoint details. This API is built using **Node.js**, **Express.js**, **MySQL**, and **JWT** for authentication.

PROJECT STRUCTURE



project_root/

1. **config/**
- auth.js # JWT authentication middleware
- db.js # MySQL database connection
2. **models/**
- User.js # User model and database queries
3. **routes/**
- authRoutes.js # Authentication-related routes (register/login)
- profileRoutes.js # Profile-related routes (get/update profile)
4. **server.js** # Main server file
5. **.env** # Environment variables
6. **Frontend**
- Login
- Dashboard-user

CONFIGURATION FILES

1. AUTHENTICATION MIDDLEWARE (AUTH.JS)

- This file contains middleware to verify JWT tokens.
- It checks if a token exists and verifies its validity.
- If valid, it attaches user details to `req.user`.
- If the token is missing or invalid, an error response is returned.

2. DATABASE CONNECTION (DB.JS)

- Establishes a connection with a **MySQL** database.
- Uses **dotenv** to load environment variables.
- Logs a message on successful connection.
- Terminates the process on failure.

MODEL

USER MODEL (USER.JS)

THE **USER MODEL** INTERACTS WITH THE MYSQL DATABASE AND PROVIDES THE FOLLOWING FUNCTIONALITIES:

1. Create a New User

```
createUser(email, fullName, userName, password, phone, callback);
```

- Hashes the password using bcrypt.
- Inserts the user details into the database.

2. Check User Credentials

```
checkUserCredentials(email, callback);
```

- Retrieves user details from the database using the email.
- Used during login to verify credentials.

3. Get User Data

```
getUserById(userId, callback);
```

- Fetches user details based on the user ID.
- Returns email, full name, username, phone number, account creation date, and last login.

4. Update User Profile

```
updateUser(userId, email, fullName, userName, phone, password, callback);
```

- Updates the user's email, name, username, phone, and password.
- Password is hashed before updating.

5. Update Last Login

```
updateLastLogin(userId, callback);
```

- Updates the `lastLogin` timestamp.

API ROUTES

1. AUTHENTICATION ROUTES (AUTHROUTES.JS)

Base URL: `/api/auth`

1. Register a New User

- **Endpoint:** `POST /register`
- **Request Body:**

```
{
  "email": "user@example.com",
  "fullName": "John Doe",
  "userName": "johndoe",
  "password": "password123",
  "phone": "1234567890"
}
```

- **Response:**

```
{
  "success": true,
  "message": "User created"
}
```

2. User Login

- **Endpoint:** `POST /login`
- **Request Body:**

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

- **Response:**

```
{
  "success": true,
  "token": "jwt_token"
```

}

- **Login Process:**

  - Retrieves user by email.
  - Compares hashed passwords.
  - Updates last login time.
  - Generates a JWT token valid for **30 days**.

2. PROFILE ROUTES (PROFILEROUTES.JS)

Base URL: `/api/profile`

1. Get User Profile

- **Endpoint:** `GET /profile`
- **Headers:**

Authorization: Bearer <jwt_token>

- **Response:**

{

"email": "user@example.com",

"fullName": "John Doe",

"userName": "johndoe",

"phone": "1234567890",

"createAt": "2024-01-01T00:00:00Z",

"lastLogin": "2024-03-01T00:00:00Z"

}

2. Update User Profile

- **Endpoint:** `PUT /profile`
- **Headers:**

Authorization: Bearer <jwt_token>

- **Request Body:**

```
{
  "email": "newemail@example.com",
  "fullName": "John Updated",
  "userName": "johnupdated",
  "phone": "0987654321",
  "password": "newpassword"
}
```

- **Response:**

```
{
  "message": "Profile updated"
}
```

SERVER SETUP & INSTALLATION & SETUP

MAIN SERVER FILE (SERVER.JS)

- Loads environment variables.
- Sets up **CORS** and JSON parsing middleware.
- Establishes database connection.
- Registers authentication and profile routes.
- Starts the server on the specified port.

1. **Clone the Repository**

```
git clone <repo-url>
cd project_root
```

2. **Install Dependencies**

```
npm install
```

3. **Configure Environment Variables**

Create a `.env` file with the following:

```
PORT=5000

JWT_SECRET=your_secret_key

DB_HOST=localhost

DB_USER=root

DB_PASSWORD=1425

DB_NAME=profile2
```

4. Start the Server

CONCLUSION

This documentation provides a complete reference for understanding and working with the **User Authentication & Profile API**. Ensure the necessary environment variables are setup and follow the API structure for seamless integration.

Authors:
Lian Abdallah
Ghadeer Alfaleh
Tasneem Arafeh
Noor Sonjoq