**Personal User Account Management API Documentation**

**Overview**

This API provides a set of endpoints to manage user accounts, including registration, authentication, email verification, password management, profile updates, and account deletion. The system is designed to handle the creation, modification, and secure management of user profiles, with authentication and authorization mechanisms built in.

The API is built using Django and Django Rest Framework and is fully RESTful.

**Base URL**

All API requests are made to the following base URL:

<https://yourdomain.com/api/v1/dashboard-controller/>

Note: For request refer to the postman collection to locate exact request body type either JSON or Form Data

**1. Register User**

**Endpoint:** /register/  
**Method:** POST  
**Description:** Registers a new user and sends a verification email.

**Request Body**

{

"username": "johndoe",

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

"password": "password123",

"profile\_picture": "optional-image-file"

}

* username (string): The unique username for the user.
* email (string): The email address of the user. Must be unique.
* password (string): The password to be used for authentication.
* profile\_picture (file, optional): The user's profile picture (PNG/JPG, max size: 5MB).

**Response**

* **201 Created**

{

"message": "Check your email for verification."

}

* **400 Bad Request:** Validation errors in the request data.

**2. Verify User Account**

**Endpoint:** /verify/<str:token>/  
**Method:** GET  
**Description:** Verifies the user account using the token sent via email.

**Response**

* **200 OK**

{

"message": "Account verified."

}

* **400 Bad Request:** Invalid or expired token.
* **404 Not Found:** User not found.

**3. Login**

**Endpoint:** /login/  
**Method:** POST  
**Description:** Authenticates the user and generates access and refresh tokens.

**Request Body**

{

"username\_or\_email": "johndoe",

"password": "password123"

}

* username\_or\_email (string): The username or email address of the user.
* password (string): The password for authentication.

**Response**

* **200 OK**

{

"access": "<access\_token>",

"refresh": "<refresh\_token>"

}

* **401 Unauthorized:** Invalid username/email or password.

**4. Logout**

**Endpoint:** /logout/  
**Method:** POST  
**Description:** Logs the user out by blacklisting the refresh token.

**Request Body**

{

"refresh": "<refresh\_token>"

}

**Response**

* **205 Reset Content**

{

"detail": "Logout successful."

}

* **400 Bad Request:** Missing or invalid refresh token.

**5. Resend Verification Email**

**Endpoint:** /resend-verification/  
**Method:** GET  
**Description:** Resends the verification email to the user.

**Response**

* **200 OK**

{

"detail": "Verification email resent. Please check your inbox."

}

* **400 Bad Request:** If the email is already verified.

**6. Delete User Account**

**Endpoint:** /delete-user/  
**Method:** POST  
**Description:** Deletes the user account after confirmation.

**Request Body**

{

"confirm": true

}

* confirm (boolean): Confirms the account deletion.

**Response**

* **204 No Content**

{

"detail": "User account deleted successfully."

}

* **400 Bad Request:** If confirmation is not provided.

**7. Forgot Password**

**Endpoint:** /forgot-password/  
**Method:** POST  
**Description:** Sends a password reset email with a reset token.

**Request Body**

{

"email": "johndoe@example.com"

}

* email (string): The email address associated with the user account.

**Response**

* **200 OK**

{

"detail": "Password reset email sent. Please check your inbox."

}

* **400 Bad Request:** If the email is not found.

**8. Set New Password**

**Endpoint:** /set-new-password/<uuid:token>/  
**Method:** POST  
**Description:** Allows the user to set a new password using the reset token.

**Request Body**

{

"password": "newpassword123"

}

* password (string): The new password for the user account.

**Response**

* **200 OK**

{

"detail": "Password has been reset successfully."

}

* **400 Bad Request:** If the token is invalid or expired.

**9. Update User Details**

**Endpoint:** /update-user/  
**Method:** POST  
**Description:** Allows the authenticated user to update their profile details.

**Request Body**

{

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

"profile\_picture": "optional-image-file",

"password": "newpassword123"

}

* email (string): The new email address for the user.
* profile\_picture (file, optional): The new profile picture for the user.
* password (string, optional): The new password for the user.

**Response**

* **200 OK**

{

"detail": "User details updated successfully."

}

* **400 Bad Request:** Validation errors in the request data.

**10. Get User Details**

**Endpoint:** /user-details/  
**Method:** GET  
**Description:** Retrieves the details of the authenticated user.

**Response**

* **200 OK**

{

"id": 1,

"username": "johndoe",

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

"profile\_picture": "http://example.com/profiles/profile.jpg",

"is\_verified": true,

"is\_paid": false

}

**Authentication**

For most endpoints (such as Login, Logout, User Detail, Update User, etc.), authentication is required. Authentication is performed using JSON Web Tokens (JWT):

* **Access Token**: Used to authenticate requests.
* **Refresh Token**: Used to generate a new access token when the current one expires.

**Authentication Flow**

1. **Login**: The user logs in with their credentials and receives access and refresh tokens.
2. **Authenticated Requests**: Use the access token in the Authorization header (e.g., Authorization: Bearer <access\_token>) for protected routes.
3. **Logout**: The user logs out, invalidating the refresh token.

**Error Responses**

* **400 Bad Request**: Invalid or missing data in the request body.
* **401 Unauthorized**: Invalid authentication credentials (e.g., incorrect password, inactive user).
* **403 Forbidden**: Insufficient permissions for the requested action.
* **404 Not Found**: Resource not found (e.g., invalid user ID).
* **500 Internal Server Error**: Server-side error.