Verify Email API — Documentation (for Frontend / Tejas)

Endpoint : `POST /api/auth/verify-email/`
Purpose : Accept the raw verification token (from the email link), validate it, mark the token used, and activate the user account (`email_verified=True`, `is_active=True`).

- Implementation notes (backend) : the endpoint hashes the raw token and looks up a DB row in `EmailVerificationToken` with `select_for_update()` inside a DB transaction to avoid race conditions (prevents two concurrent uses of the same token).

Request (required)

- **URL :** `POST /api/auth/verify-email/`
- **Headers**
    - `Content-Type: application/json`
    - `Accept: application/json`
- **Body (JSON)**

```
{
        "token": "raw_token_from_email_link"
}
```

- The `token` is the raw string provided in the verification email link : `https://<frontend-host>/verify-email?token=<RAW_TOKEN>`

Successful response

- **Status :** `200 OK`
- **Body :**

```
{
        "detail": "Email verified. You can now log in."
}
```

Frontend action : show success to the user (e.g., “Email verified — you can now log in.”) and redirect to the login page.

Error responses (common cases)

(1) Missing or empty token (bad request)

- **Status :** `400 Bad Request`
- **Example response :**

```
{
        "token": ["This field is required."]
}
```

Frontend : show message *“Verification token missing.”* and let user manually request resend.

---

2) Invalid token (no matching DB row)

- **Status :** `400 Bad Request`
- **Response :**

{ "detail": "Invalid verification link." }

Frontend : show *“Invalid verification link.”* Offer a **Resend verification** button and a link to support.

---

3) Already-used token

- **Status:** `400 Bad Request`
- **Response:**

  { "detail" : "This verification link has already been used." }

Frontend : treat as “already verified” — show message like *“This link was already used. Try logging in.”* Offer a link to the login page or a “Resend verification” option if the user still cannot log in.

---

4) Expired token

- **Status:** `400 Bad Request`
- **Response:**

  { "detail": "Verification link expired." }

Frontend : show *“Verification link expired.”* Provide a prominent **Resend verification** button (calls `POST /api/auth/resend-verification/`).

---

5) Generic / server error

- **Status:** `500 Internal Server Error`
- **Response :**

  { "detail": "Internal server error." }

Frontend : show a friendly error: *“Something went wrong. Please try again later.”* Allow user to try again or request a resend.

Frontend flow (recommended)

1. **User clicks email link** :
   `https://<frontend-host>/verify-email?token=<RAW_TOKEN>`.
    - Extract `token` from the URL query string.
2. **Call the API** :
    - POST `{ "token": "<RAW_TOKEN>" }` to `/api/auth/verify-email/`.
3. **Show a "Verifying…" state** while waiting for the response.
4. **On success (200)** :
    - Show success message: *"Email verified. You can now log in."*
    - Redirect to the login page or auto-navigate after a short delay.
5. **On token errors (400)**:
    - If `Invalid verification link.` → show *"Invalid link."* + **Resend** button.
    - If the `Verification link expired.` → show *"Link expired."* + **Resend** button.
    - If `This verification link has already been used.` → show *"Already used — please log in."*
6. **On 500**:
    - Show a generic message and provide Retry / Resend / Contact Support.