Resend Verification API — Documentation

Endpoint : `POST /api/auth/resend-verification/`
Purpose : Ask the backend to (re)generate a fresh verification token and send a verification email to the given email address. The endpoint is **rate-limited** (scope `resend_verification`) and responds **generically** to avoid account enumeration.

- **Implementation notes :** the server will mark any previously unused tokens for that user as `used`, create a fresh token, and **send the verification email synchronously** (so the request may fail with `500` if SMTP or template sending fails). If the email address does not match any account, the endpoint still returns a **200 OK** generic response.

Request (required)

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

  { "email" : "user@example.com" }

- The server normalizes the email (strip + lower) before lookup.

Successful / normal response

The endpoint intentionally returns a **generic success message** so callers cannot determine whether an account exists.

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

  { "detail": "If an account exists, a verification email has been sent." }

Frontend action : Always show the same message to the user: *“If an account exists, a verification email has been sent.”* — and optionally show guidance to check spam/junk and a “Contact support” link.

Other positive response (already verified)

If the email belongs to an already-verified account the endpoint returns an explicit message :

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

  { "detail": "Email already verified." }

Frontend action : You can show a friendly message like: *“This email is already verified — please log in.”* Optionally show a “Go to login” button.

Error responses and edge cases

1) Rate limit exceeded (throttle)

- **Status :** `429 Too Many Requests`
- **Headers :** may include `Retry-After: <seconds>`
- **Body (DRF default) :**

{ "detail": "Request was throttled. Expected available in <seconds> seconds." }

Frontend : disable the resend button and show a friendly cooldown message, optionally showing the retry seconds if present.

---

2) SMTP / email send failure (server-side)

Because the server sends the email synchronously, a failure during sending (SMTP/auth/network/template) can cause an exception and yield:

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

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

Frontend : show: *"We couldn't send the verification email right now. Please try again later."* Provide a retry button or ask user to contact support.

---

3) Invalid request body (400)

If the `email` field is missing or not a valid email format, the API will return field validation errors :

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

  { "email": ["This field is required."] }

  Or

  { "email": ["Enter a valid email address."] }

Frontend : validate email client-side and show the error returned.

Behaviour details (what backend does)

- Normalize `email = email.strip().lower()` before lookup.
- If a matching `User` exists :
    - If `user.email_verified` or `user.is_active` → respond `200` with `"Email already verified."` (no new token sent).
    - Else :
        - Mark any previous unused tokens for the user as `used` (prevent reuse).
        - Create a new `EmailVerificationToken` DB row (hashed token stored).
        - **Send verification email synchronously** using templates `emails/verify_email.html` and `emails/verify_email.txt`.
        - Respond `200` with the generic message.
- If no matching user → respond `200` generic message (no difference visible to caller).
- Endpoint is throttled via DRF ScopedRateThrottle with scope `resend_verification` (configured in settings).

Example flows & `curl` examples :

Normal (email exists, not verified)

```
curl -X POST https://api.yourdomain.com/api/auth/resend-verification/ \
  -H "Content-Type: application/json" \
  -d '{"email":"john.client@example.com"}'
```

Response:

{ "detail": "If an account exists, a verification email has been sent." }

Email already verified

```
curl -X POST https://api.yourdomain.com/api/auth/resend-verification/ \
  -H "Content-Type: application/json" \
  -d '{"email":"already.verified@example.com"}'
```

Response:

```
{ "detail": "Email already verified." }
```

Email not present in system (generic)

```
curl -X POST https://api.yourdomain.com/api/auth/resend-verification/ \
  -H "Content-Type: application/json" \
  -d '{"email":"notfound@example.com"}'
```

Response:

```
{ "detail": "If an account exists, a verification email has been sent." }
```

Rate limited

```
HTTP/1.1 429 Too Many Requests
{
  "detail": "Request was throttled. Expected available in 3600 seconds."
}
```

SMTP failure (possible)

```
HTTP/1.1 500 Internal Server Error
{ "detail": "Internal server error." }
```

Frontend integration & UX recommendations

1. **Always show the generic success message** after calling the endpoint (unless the response explicitly says `"Email already verified."`). Example: *“If an account exists, a verification email has been sent. Check your inbox (and spam).”*

2. **Disable the Resend button** while waiting for response to avoid accidental double-clicks.

3. If you get `429`, show a friendly message: *“Too many resend attempts. Please try again after some time.”* Use `Retry-After` header if present to show countdown.

4. If you get `500`, show: *“We couldn’t send the verification email right now. Please try again later.”* and provide a Retry button.

5. Offer a link to Login and Contact Support on the same UI.

6. Rate-limit on frontend: optionally disable Resend for ~30–60 seconds to reduce server load and improve UX.