Signup API — MindEase (SYNC EMAIL SEND)

Audience: Frontend developer (Tejas)
Base URL: `https://api.yourdomain.com`
Endpoint: `POST /api/auth/signup/`

Summary : Create a new user account (Client or Counsellor). New users are created with `is_active: false` and must verify their email before login. **Email verification is sent synchronously** during signup (the server sends the verification email before returning a response). If the email sending fails (SMTP or template error), the request may return an error (500). Frontend should handle both success and failure cases.

Request headers (required)

Content-Type : application/json
Accept : application/json

Fields (request body) — validation rules

- `first_name` — optional, max 150 chars.
- `last_name` — optional, max 150 chars.
- `email` — **required**, valid email, unique (case-insensitive).
- `phone` — optional, exactly 10 digits if provided, unique.
- `password` — required, validated against Django validators (complexity).
- `confirm_password` — required, must match `password`.
- `account_type` — `"client"` (default) or `"counsellor"`.

- `agreed_terms` — **required** and must be `true`.
- `age_group` (client) — one of `under_18`, `18_25`, `26_40`, `41_60`, `60_plus`.

Counsellor- only (required when `account_type == "counsellor"`)

- `license_number` — required.
- `specializations` — required list of integer IDs (must exist).
- `fees_per_session` — required (decimal string, e.g. `"700.00"`).
- `availability` — list of integer IDs (optional but validated if provided).

Behaviour (important, updated)

- Server creates `User` (with `is_active = False`) and a hashed DB `EmailVerificationToken` record.
- The server then **sends the verification email synchronously** using templates `emails/verify_email.html` and `emails/verify_email.txt`.
- If email sending is successful, the server returns `201 Created` with user info.
- **If email sending fails (SMTP / network / template error)** the send call raises and the request will likely return `500 Internal Server Error` (the current view does not catch send errors). In that failure case the DB user and token *may already be created*. Frontend should handle this by showing a clear error and allowing the user to retry or use a Resend Verification endpoint. (Backend improvement recommended: catch email errors and return `201` + warning or save failure — but not implemented now.)

Example : Client signup (request)

POST `/api/auth/signup/`
Headers as above

Request body:

```
{
  "first_name": "John",
  "last_name": "Doe",
  "email": "john.client@example.com",
  "phone": "9876543210",
  "password": "StrongPass123!",
  "confirm_password": "StrongPass123!",
  "account_type": "client",
  "age_group": "18_25",
  "agreed_terms": true
}
```

Successful response (201 Created)

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "detail": "Account created. Please verify your email to activate the account.",
  "user": {
    "id": 123,
    "email": "john.client@example.com",
    "first_name": "John",
    "last_name": "Doe",
    "role": "client",
    "is_active": false
  }
}
```

Frontend action : show “Account created — check your email” screen and link to login (login should still be blocked until verification).

Client error cases (400 Bad Request)

Duplicate email

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "email": ["A user with this email already exists."]
}
```

Duplicate phone

```
HTTP/1.1 400 Bad Request
{
  "phone": ["This phone number is already used."]
}
```

Password mismatch

```
HTTP/1.1 400 Bad Request
{
  "confirm_password": ["Passwords do not match."]
}
```

Terms not accepted

```
HTTP/1.1 400 Bad Request
{
  "agreed_terms": ["You must accept terms and conditions."]
}
```

Invalid phone format

```
HTTP/1.1 400 Bad Request
{
  "phone": ["Phone number must be exactly 10 digits."]
}
```

Field-level validation pattern
Backend returns `{"field": ["error1", "error2"]}` for field validation errors. Display the first error under each field.

Example: Counsellor signup (request)

POST `/api/auth/signup/`

Request body :

```
{
  "first_name": "Sara",
  "last_name": "K",
  "email": "sara.counsellor@example.com",
  "phone": "9123456789",
  "password": "CounsellorPass1!",
  "confirm_password": "CounsellorPass1!",
  "account_type": "counsellor",
  "agreed_terms": true,
  "license_number": "LIC-2025/001",
  "specializations": [1, 2],
  "fees_per_session": "700.00",
  "availability": [1, 3]
}
```

Success (201 Created) — same format as client above.

Counsellor error cases (400 Bad Request)

Missing required counsellor fields

```
HTTP/1.1 400 Bad Request
{
  "license_number": ["This field is required for counsellor accounts."],
  "specializations": ["This field is required for counsellor accounts."],
  "fees_per_session": ["This field is required for counsellor accounts."]
}
```

Invalid specialization or availability IDs

```
HTTP/1.1 400 Bad Request
{
  "specializations": ["One or more specializations are invalid."],
  "availability": ["One or more availability slots are invalid."]
}
```

Email sending failure (server-side) — how frontend will see it

If the SMTP send fails during signup (synchronous send used), the server may return:

Example — SMTP send error

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

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

Notes for frontend:

- If you receive `500` after submitting signup, the user account **may have been created** but the verification email was not delivered.
- Recommended UI flow : show an error message like **“We could not send the verification email. Please try again or request a resend.”** and provide a **Resend Verification** button that calls `POST /api/auth/resend-verification/`.
- Optionally, guide user to check the email they provided and try signup again with the same email (but duplicates will be rejected). Best user experience: call the resend endpoint rather than re-signup.

Other responses

429 Too Many Requests (throttled)

- If rate limit for signup is hit (DRF `ScopedRateThrottle` with scope `"signup"`), response:

```
HTTP/1.1 429 Too Many Requests
Content-Type: application/json
Retry-After: <seconds>

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

500 Internal Server Error (generic)

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

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

Recommended frontend UI (short)

- After `201`: show “Check your email” page with next steps and a **Resend verification** button.
- After `500`: show “We could not send verification email” with actions: **Resend** or **Contact support**.
- For `429`: show friendly cooldown message and disable retry.