Profile API — MindEase

Audience : Frontend developer (Tejas)
Base URL : `http://localhost:8000` (use production base in prod)

Endpoint : `GET | PUT | PATCH /api/auth/profile/`

Auth: Protected — requires a valid access token. Browser clients must use `credentials: "include"` so HttpOnly cookies are sent. If the access token is expired the frontend should call `POST /api/auth/token/refresh/` to refresh and retry (see refresh flow docs).

Purpose

Get or update the currently authenticated user's profile. The endpoint returns a combined view: core `User` fields plus the appropriate profile object depending on the user's role:

- **Client** → `client_profile` (age_group, agreed_terms, etc.)
- **Counsellor** → `counsellor_profile` (license_number, specializations, fees_per_session, availability, experience, etc.)

Use :

- `GET` — read profile
- `PUT` — replace profile (full update)
- `PATCH` — partial update (recommended for small edits)

All methods require authentication.

Common HTTP headers

Accept : application/json
Content-Type : application/json # for PUT/PATCH

Browser : always include `credentials: "include"`.

Returned shape (fields you will receive)

Top-level user fields (always present):

```
{
  "id": 42,
  "email": "user@example.com",
  "first_name": "Paras",
  "last_name": "Mani",
  "phone": "9876543210",
  "role": "client",           // or "counsellor"
  "profile_picture": "https://...",
  "bio": "short bio",         // if present (User-level)
  "gender": "male",           // "male" | "female" | "other" | "prefer_not_to_say"
  "date_of_birth": "1990-05-01", // ISO YYYY-MM-DD or null
  "is_active": true,
  "email_verified": true,
  "date_joined": "2025-10-01T12:00:00Z"
}
```

Role-specific containers (returned as nested objects):

Client example

```
"client_profile": {
  "age_group": "18_25",   // one of enum keys
  "agreed_terms": true
}
```

Counsellor example

```
"counsellor_profile": {
  "license_number": "LIC-1234",
  "specializations": ["Anxiety","Depression"],   // list of names (not ids)
  "fees_per_session": "1200.00",                 // string (decimal)
  "availability": ["weekdays","evenings"],       // list of names (not ids)
  "experience": "10 years private practice"      // free text
```

}

Validation rules (server-side)

User-level

- `email` — read-only here (changed on special flows). Unique.
- `first_name`, `last_name` — max length 150.
- `phone` — exactly 10 digits if provided (unique).
- `profile_picture` — must be a valid URL if provided.
- `bio` — free text (optional).
- `gender` — one of: `"male"`, `"female"`, `"other"`, `"prefer_not_to_say"`.
- `date_of_birth` — ISO date `YYYY-MM-DD` (optional). Server may validate logical age if desired.

ClientProfile

- `age_group` — one of: `"under_18"`, `"18_25"`, `"26_40"`, `"41_60"`, `"60_plus"`.
- `agreed_terms` — boolean; client must have previously accepted terms at signup.

CounsellorProfile

- `license_number` — 4–30 chars, letters/digits/`-` or `/`, must be unique.
- `specializations` — **list of names**. Each name must exist in `Specialization` table; server returns error if any name invalid.
- `fees_per_session` — decimal ≥ 0.
- `availability` — **list of names** from `AvailabilitySlot` table; server returns error if any name invalid.

- `experience` — free text (e.g. `"10 years private practice"`).

GET /api/auth/profile/ — Read current user's profile

Request

GET /api/auth/profile/
Credentials: include

Successful Response (200 OK)
Returns full user + profile as described above.


```
{
  "id": 42,
  "email": "paras0mani6@example.com",
  "first_name": "Paras",
  "last_name": "Mani",
  "phone": "9876543210",
  "role": "counsellor",
  "profile_picture": "https://.../avatar.png",
  "bio": "Clinical counsellor",
  "gender": "male",
  "date_of_birth": "1990-05-01",
  "is_active": true,
  "email_verified": true,
  "date_joined": "2025-09-10T10:00:00Z",
  "counsellor_profile": {
    "license_number": "LIC-1234",
    "specializations": ["Anxiety","Depression"],
    "fees_per_session": "1200.00",
    "availability": ["weekdays","evenings"],
    "experience": "10 years private practice"
  }
}
```


Errors

- `401 Unauthorized` — missing/invalid access token. Frontend should call refresh flow then retry.
- `500` — server error.

PATCH /api/auth/profile/ — Partial update (recommended)

Notes

- Only include fields you want to change.
- For role-specific fields the server will validate that user's role matches the profile object (e.g., only `counsellor` can update `counsellor_profile`).
- The server accepts `specializations` and `availability` as lists of names (strings). If a name is not found the request fails with 400.

Example: update phone + profile picture (client or counsellor)

PATCH /api/auth/profile/
Content-Type: application/json
Credentials: include

```
{
  "phone": "9123456780",
  "profile_picture": "https://example.com/newavatar.png"
}
```

Example: counsellor updating fees and specializations

PATCH /api/auth/profile/
Content-Type: application/json
Credentials: include

```
{
  "counsellor_profile": {
    "fees_per_session": "1500.00",
    "specializations": ["Anxiety","Relationship Counselling"]
  }
```

```
}
```

Success (200 OK) — returns updated profile JSON.

Validation / error responses

- `400 Bad Request` — validation errors. Example:

```
{
  "counsellor_profile": {
    "specializations": ["One or more specializations are invalid."]
  }
}
```

- `401 Unauthorized` — not authenticated.
- `403 Forbidden` — trying to update another role's profile (e.g., client trying to set `license_number`).

PUT /api/auth/profile/ — Full replace

Notes

- PUT expects a full snapshot payload for the user + profile (server-side will validate required role-specific fields). If fields are omitted they may be reset — use with care.
- Use PUT primarily if you truly want to replace the full profile; otherwise prefer PATCH.

Example (client full update)

```
{
  "first_name": "Amit",
  "last_name": "Kumar",
  "phone": "9123456780",
  "bio": "New bio",
  "gender": "male",
  "date_of_birth": "2000-01-01",
  "client_profile": {
```

```
    "age_group": "18_25",
    "agreed_terms": true
  }
}
```

Example (counsellor full update)

```
{
  "first_name": "Pooja",
  "last_name": "Shah",
  "phone": "9876501234",
  "bio": "Somatic therapist",
  "gender": "female",
  "date_of_birth": "1988-07-12",
  "counsellor_profile": {
    "license_number": "LIC-5555",
    "specializations": ["Depression","Stress Management"],
    "fees_per_session": "2000.00",
    "availability": ["weekends","evenings"],
    "experience": "8 years clinical practice"
  }
}
```

Success (200 OK) — returns full updated profile.

Errors

- `400` — validation errors (field messages).
- `401` — unauthorized.

Important behaviors & notes for frontend

1. **Only authenticated users may call this endpoint.** Use `fetchWithAuth` helper (or ensure on 401 you call refresh then retry).
2. **Phone and license unique constraints:** if the user attempts to update `phone` or `license_number` to an already-existing value, server returns `400` with a clear

message. Frontend should show that message inline.

3. **specializations / availability by name:** when updating counsellor fields, send arrays of names, e.g. `["Anxiety","Depression"]`. If you send names that don't exist, server returns `400`.

4. **Role enforcement:** server enforces role/profile consistency. A `client` cannot set `counsellor_profile.license_number` — attempting to do so returns `400` or `403`.

5. **Partial updates use PATCH.** This is the preferred, safer method for UI forms.

6. **Dates:** `date_of_birth` should be `YYYY-MM-DD`. Server may validate age ranges (if implemented).

7. **Profile picture:** send as a URL on `profile_picture`. If you want file upload support later, we can add a separate upload endpoint.

Example frontend usage (browser console)

GET (read profile)

```
fetchWithAuth("http://localhost:8000/api/auth/profile/", { method: "GET" })
  .then(async res => {
    if (res.status === 200) {
      console.log(await res.json());
    } else if (res.status === 401) {
      // not authenticated / refresh failed -> redirect to login
      window.location = "/login";
    } else {
      console.error("Profile error", res.status);
    }
  });
```

PATCH (update subset)

```
// change phone and bio
fetchWithAuth("http://localhost:8000/api/auth/profile/", {
  method: "PATCH",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({
```

```
    phone: "9123456780",
    bio: "Updated short bio"
  })
}).then(async res => {
  if (res.status === 200) {
    console.log("Updated:", await res.json());
  } else {
    console.error("Update failed", res.status, await res.json());
  }
});
```

PATCH (counsellor - change specializations)

```
fetchWithAuth("http://localhost:8000/api/auth/profile/", {
  method: "PATCH",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({
    counsellor_profile: {
      specializations: ["Anxiety","Stress Management"],
      fees_per_session: "1800.00"
    }
  })
}).then(async res => console.log(res.status, await res.json()));
```

Status codes summary

- `200 OK` — success (GET/PUT/PATCH).
- `400 Bad Request` — validation error (field-level info in JSON).
- `401 Unauthorized` — missing or invalid access token. Try refresh then retry.
- `403 Forbidden` — trying to update fields not allowed for this role.
- `500` — server error.