Here's what to tell your frontend team:

Frontend Integration Guide: New Signup & Authentication System

Overview

The signup system now supports 3 different flows:
1. **Individual Signup** - Solo practitioners
2. **Organization Signup** - Creating a new law firm/organization (becomes admin)
3. **Invitation Signup** - Joining an existing organization (becomes member)

---

1. INDIVIDUAL SIGNUP

When to Use:
- User wants a personal account
- No organization needed

API Request:
```http
POST /api/auth/signup
Content-Type: application/json

{
  "email": "john.doe@example.com",
  "phone": "+14155551234",
  "phoneCountry": "US",
  "password": "SecurePassword123!",
  "firstName": "John",
  "lastName": "Doe",
  "planType": "individual",
  "profile": {
    "barNumber": "123456",
    "specialty": "Criminal Law"
  }
}
```

Required Fields:
- ✅ `email` (string) - Valid email
- ✅ `phone` (string) - E.164 format (e.g., +14155551234)

- ✅ `phoneCountry` (string) - ISO country code (e.g., "US")
- ✅ `password` (string) - Min 8 characters
- ✅ `firstName` (string)
- ✅ `lastName` (string)
- ✅ `planType` (string) - Must be `"individual"`
- ⚪ `profile` (object) - Optional, any custom JSON data

Response (201 Created):
```json
{
  "userId": 1,
  "email": "john.doe@example.com",
  "phone": "+14155551234",
  "firstName": "John",
  "lastName": "Doe",
  "subscription": {
    "planType": "individual_free",
    "status": "active",
    "role": null,
    "organizationId": null
  }
}
```

What Frontend Should Do:
1. Show success message: "Account created successfully!"
2. Redirect to `/login` page
3. User logs in to get access token

---

2. ORGANIZATION SIGNUP (Become Admin)

When to Use:
- User wants to create a new law firm/organization
- They will become the organization admin

API Request:
```http
POST /api/auth/signup
Content-Type: application/json

{
  "email": "admin@lawfirm.com",
```

```
  "phone": "+14155551234",
  "phoneCountry": "US",
  "password": "SecurePassword123!",
  "firstName": "Jane",
  "lastName": "Smith",
  "planType": "organization",
  "organization": {
    "name": "Smith & Associates Law Firm",
    "details": {
      "address": "123 Main St, New York, NY",
      "taxId": "12-3456789",
      "phone": "+12025555000"
    }
  },
  "profile": {
    "title": "Senior Partner"
  }
}
```

Required Fields:
- ✅ `email`, `phone`, `phoneCountry`, `password`, `firstName`, `lastName` (same as individual)
- ✅ `planType` (string) - Must be `"organization"`
- ✅ `organization` (object) - **REQUIRED**
  - ✅ `organization.name` (string) - Organization name
  - ⚪ `organization.details` (object) - Optional metadata
- ⚪ `profile` (object) - Optional user metadata

Response (201 Created):
```json
{
  "userId": 2,
  "email": "admin@lawfirm.com",
  "phone": "+14155551234",
  "firstName": "Jane",
  "lastName": "Smith",
  "subscription": {
    "planType": "organization_basic",
    "status": "active",
    "role": "admin",
    "organizationId": 1
  },
  "organization": {
    "id": 1,
```

```
    "name": "Smith & Associates Law Firm",
    "details": {
      "address": "123 Main St, New York, NY",
      "taxId": "12-3456789",
      "phone": "+12025555000"
    }
  }
}
```

What Frontend Should Do:
1. Show success: "Organization created! You are now the admin."
2. Redirect to `/login`
3. After login, user will have admin privileges

---

3. INVITATION SIGNUP (Join Organization)

When to Use:
- User received an invitation email
- They clicked "Accept Invitation" button
- URL has `?token=xxx` parameter

Frontend Flow:

Step 1: Detect Invitation Token
```javascript
// On signup page load
const urlParams = new URLSearchParams(window.location.search);
const inviteToken = urlParams.get('token');

if (inviteToken) {
  // Show special UI: "You're joining [Organization Name]"
  // Pre-fill email if available from backend
}
```

Step 2: API Request
```http
POST /api/auth/signup
Content-Type: application/json

{
```

```
  "email": "member@example.com",
  "phone": "+14155552222",
  "phoneCountry": "US",
  "password": "MemberPass123!",
  "firstName": "Bob",
  "lastName": "Johnson",
  "inviteToken": "a1b2c3d4-e5f6-7890-abcd-ef1234567890"
}
```

Required Fields:
- ✅ `email`, `phone`, `phoneCountry`, `password`, `firstName`, `lastName`
- ✅ `inviteToken` (string) - From URL query parameter
- ❌ **DO NOT** include `planType` or `organization` - token handles everything

Response (201 Created):
```json
{
  "userId": 3,
  "email": "member@example.com",
  "phone": "+14155552222",
  "firstName": "Bob",
  "lastName": "Johnson",
  "subscription": {
    "planType": "organization_basic",
    "status": "active",
    "role": "member",
    "organizationId": 1
  },
  "organization": {
    "id": 1,
    "name": "Smith & Associates Law Firm"
  }
}
```

What Frontend Should Do:
1. Show success: "You've joined Smith & Associates Law Firm!"
2. Redirect to `/login`
3. After login, user will be a member (not admin)

---

4. LOGIN (All User Types)

API Request:
```http
POST /api/auth/login
Content-Type: application/json

{
  "email": "admin@lawfirm.com",
  "password": "SecurePassword123!"
}
```

Response (200 OK):
```json
{
  "sessionId": "550e8400-e29b-41d4-a716-446655440000",
  "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzZXNzaW9uSWQiOiI1NTBlODQwMC...",
  "refreshToken": "refresh_abc123def456",
  "expiresAt": "2025-11-23T21:00:00.000Z",
  "refreshExpiresAt": "2025-12-23T20:00:00.000Z",
  "context": {
    "sessionId": "550e8400-e29b-41d4-a716-446655440000",
    "userId": 2,
    "organizationId": 1,
    "roles": ["admin"],
    "planHints": {
      "planType": "organization_basic",
      "source": "organization"
    },
    "loginType": "organization"
  },
  "user": {
    "id": 2,
    "email": "admin@lawfirm.com",
    "phone": "+14155551234",
    "firstName": "Jane",
    "lastName": "Smith",
    "subscription": {
      "planType": "organization_basic",
      "role": "admin",
      "organizationId": 1
    },
    "organization": {
```

```
    "id": 1,
    "name": "Smith & Associates Law Firm"
   }
  }
}
```

What Frontend Should Store:

```javascript
// Store in localStorage/sessionStorage
const authData = {
  // Tokens - needed for API calls
  accessToken: response.accessToken,
  refreshToken: response.refreshToken,
  expiresAt: response.expiresAt,

  // User info
  userId: response.user.id,
  email: response.user.email,
  firstName: response.user.firstName,
  lastName: response.user.lastName,

  // Authorization - IMPORTANT for UI decisions
  role: response.user.subscription.role,  // 'admin', 'member', or null
  organizationId: response.user.subscription.organizationId,  // 1 or null
  organizationName: response.user.organization?.name,  // "Smith & Associates" or null
  planType: response.user.subscription.planType,  // 'individual_free' or 'organization_basic'
  loginType: response.context.loginType  // 'individual' or 'organization'
};

localStorage.setItem('auth', JSON.stringify(authData));
```

---

5. UI LOGIC BASED ON LOGIN TYPE

Individual User:
```javascript
if (authData.loginType === 'individual') {
  // Show: Personal workspace
  // Hide: Organization features, invite buttons
}
```

```

### **Organization Admin:**
```javascript
if (authData.loginType === 'organization' && authData.role === 'admin') {
  // Show: "Invite Members" button
  // Show: "View Members" page
  // Show: Organization settings
  // Can make requests to /organizations/:id/invitations
}
```

Organization Member:
```javascript
if (authData.loginType === 'organization' && authData.role === 'member') {
  // Show: Organization name in header
  // Show: "You are a member of [Org Name]"
  // Hide: Admin controls (invite, manage members)
  // Cannot access admin endpoints
}
```

---

6. MAKING AUTHENTICATED API CALLS

Always Include Authorization Header:

```javascript
// Example: Invite members (admin only)
fetch('http://localhost:3000/api/organizations/1/invitations', {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${authData.accessToken}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    emails: ['member1@example.com', 'member2@example.com']
  })
});
```

Setup Axios Interceptor (Recommended):
```

```javascript
import axios from 'axios';

axios.interceptors.request.use(config => {
  const auth = JSON.parse(localStorage.getItem('auth'));
  if (auth?.accessToken) {
    config.headers.Authorization = `Bearer ${auth.accessToken}`;
  }
  return config;
});

// Now all axios calls include the token automatically
axios.post('/api/organizations/1/invitations', {
  emails: ['member@example.com']
});
```

---

7. ERROR HANDLING

Common Errors:

Status	Error	Meaning	Frontend Action
400	"Missing required signup fields"	Missing email/phone/password/etc	Show validation errors
400	"Organization name required"	planType='organization' but no org name	Prompt for organization name
400	"Email already exists"	Duplicate email	Show "Email already registered. Try login."
401	"Authentication required"	No token or invalid token	Redirect to login
403	"Admin access required"	User is member, not admin	Show "Contact your organization admin"
403	"Email verification required"	Email not verified	Show "Check your email to verify"
404	"Invalid invitation token"	Token doesn't exist or expired	Show "Invitation expired. Contact admin."

Example Error Handling:

```javascript
try {
  const response = await fetch('/api/auth/signup', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
```

```
    body: JSON.stringify(signupData)
  });

  if (!response.ok) {
    const error = await response.json();
    // Show error.error to user
    alert(error.error);
  } else {
    const data = await response.json();
    // Success!
    navigate('/login');
  }
} catch (err) {
  alert('Network error. Please try again.');
}
```

---

8. SAMPLE FRONTEND SIGNUP COMPONENT

```jsx
import React, { useState, useEffect } from 'react';
import { useNavigate, useSearchParams } from 'react-router-dom';

function SignupPage() {
  const navigate = useNavigate();
  const [searchParams] = useSearchParams();
  const inviteToken = searchParams.get('token');

  const [formData, setFormData] = useState({
    email: '',
    phone: '',
    phoneCountry: 'US',
    password: '',
    firstName: '',
    lastName: '',
    planType: inviteToken ? null : 'individual',
    organizationName: ''
  });

  const handleSubmit = async (e) => {
    e.preventDefault();
```

```
  const payload = {
    email: formData.email,
    phone: formData.phone,
    phoneCountry: formData.phoneCountry,
    password: formData.password,
    firstName: formData.firstName,
    lastName: formData.lastName
  };

  // Add invitation token if present
  if (inviteToken) {
    payload.inviteToken = inviteToken;
  } else {
    payload.planType = formData.planType;

    // Add organization details if creating org
    if (formData.planType === 'organization') {
      payload.organization = {
        name: formData.organizationName
      };
    }
  }

  try {
    const response = await fetch('/api/auth/signup', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(payload)
    });

    if (response.ok) {
      navigate('/login');
    } else {
      const error = await response.json();
      alert(error.error);
    }
  } catch (err) {
    alert('Error: ' + err.message);
  }
};

return (
  <div>
    <h1>
```

```
  {inviteToken ? 'Join Organization' : 'Sign Up'}
</h1>

<form onSubmit={handleSubmit}>
  <input
    type="email"
    placeholder="Email"
    value={formData.email}
    onChange={(e) => setFormData({...formData, email: e.target.value})}
    required
  />

  <input
    type="tel"
    placeholder="Phone (+14155551234)"
    value={formData.phone}
    onChange={(e) => setFormData({...formData, phone: e.target.value})}
    required
  />

  <input
    type="password"
    placeholder="Password"
    value={formData.password}
    onChange={(e) => setFormData({...formData, password: e.target.value})}
    required
  />

  <input
    type="text"
    placeholder="First Name"
    value={formData.firstName}
    onChange={(e) => setFormData({...formData, firstName: e.target.value})}
    required
  />

  <input
    type="text"
    placeholder="Last Name"
    value={formData.lastName}
    onChange={(e) => setFormData({...formData, lastName: e.target.value})}
    required
  />
```

```
        {/* Only show plan selection if NOT invitation signup */}
        {!inviteToken && (
          <div>
            <label>
              <input
                type="radio"
                value="individual"
                checked={formData.planType === 'individual'}
                onChange={(e) => setFormData({...formData, planType: e.target.value})}
              />
              Individual Account
            </label>

            <label>
              <input
                type="radio"
                value="organization"
                checked={formData.planType === 'organization'}
                onChange={(e) => setFormData({...formData, planType: e.target.value})}
              />
              Create Organization
            </label>

            {formData.planType === 'organization' && (
              <input
                type="text"
                placeholder="Organization Name"
                value={formData.organizationName}
                onChange={(e) => setFormData({...formData, organizationName: e.target.value})}
                required
              />
            )}
          </div>
        )}

        <button type="submit">Sign Up</button>
      </form>
    </div>
  );
}
```

---

9. QUICK REFERENCE

Signup Endpoints:
- **POST** `/api/auth/signup` - All signup types

Signup Types:

Type	`planType`	`organization`	`inviteToken`	Result
Individual	`"individual"`	❌ Not needed	❌ Not needed	Solo account
Organization Admin	`"organization"`	✅ Required	❌ Not needed	New org, user is admin
Invited Member	❌ Omit	❌ Omit	✅ Required	Join existing org as member

Response Fields to Store:
- `accessToken` - For API authentication
- `user.subscription.role` - Show/hide admin features
- `user.subscription.organizationId` - Which org user belongs to
- `user.organization.name` - Display in UI
- `context.loginType` - "individual" or "organization"

Questions? Let me know if you need clarification on any part! 🚀