-- Users table (extends Supabase auth.users)

CREATE TABLE public.user\_profiles (

id UUID REFERENCES auth.users(id) ON DELETE CASCADE PRIMARY KEY,

role TEXT NOT NULL DEFAULT 'user' CHECK (role IN ('user', 'admin')),

full\_name TEXT,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Universities table

CREATE TABLE public.universities (

id UUID DEFAULT gen\_random\_uuid() PRIMARY KEY,

name TEXT NOT NULL UNIQUE,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Domains table

CREATE TABLE public.domains (

id UUID DEFAULT gen\_random\_uuid() PRIMARY KEY,

university\_id UUID NOT NULL REFERENCES universities(id) ON DELETE CASCADE,

name TEXT NOT NULL,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

UNIQUE(university\_id, name)

);

-- Subjects table

CREATE TABLE public.subjects (

id UUID DEFAULT gen\_random\_uuid() PRIMARY KEY,

domain\_id UUID NOT NULL REFERENCES domains(id) ON DELETE CASCADE,

name TEXT NOT NULL,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

UNIQUE(domain\_id, name)

);

-- Resources table

CREATE TABLE public.resources (

id UUID DEFAULT gen\_random\_uuid() PRIMARY KEY,

subject\_id UUID NOT NULL REFERENCES subjects(id) ON DELETE CASCADE,

title TEXT NOT NULL,

description TEXT,

submitted\_by UUID REFERENCES auth.users(id),

is\_approved BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- User submitted resources tracking

CREATE TABLE public.user\_submitted\_resources (

id UUID DEFAULT gen\_random\_uuid() PRIMARY KEY,

user\_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,

resource\_id UUID NOT NULL REFERENCES resources(id) ON DELETE CASCADE,

submitted\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

UNIQUE(user\_id, resource\_id)

);

-- User resource requests

CREATE TABLE public.user\_resource\_requests (

id UUID DEFAULT gen\_random\_uuid() PRIMARY KEY,

user\_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,

subject\_id UUID NOT NULL REFERENCES subjects(id) ON DELETE CASCADE,

title TEXT NOT NULL,

description TEXT,

status TEXT DEFAULT 'pending' CHECK (status IN ('pending', 'fulfilled', 'rejected')),

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- User saved resources

CREATE TABLE public.user\_saved\_resources (

id UUID DEFAULT gen\_random\_uuid() PRIMARY KEY,

user\_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,

resource\_id UUID NOT NULL REFERENCES resources(id) ON DELETE CASCADE,

saved\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

UNIQUE(user\_id, resource\_id)

);

-- Indexes for better performance

CREATE INDEX idx\_domains\_university ON domains(university\_id);

CREATE INDEX idx\_subjects\_domain ON subjects(domain\_id);

CREATE INDEX idx\_resources\_subject ON resources(subject\_id);

CREATE INDEX idx\_resources\_approved ON resources(is\_approved);

CREATE INDEX idx\_user\_saved\_resources\_user ON user\_saved\_resources(user\_id);

CREATE INDEX idx\_user\_resource\_requests\_user ON user\_resource\_requests(user\_id);

CREATE INDEX idx\_user\_resource\_requests\_status ON user\_resource\_requests(status);

-- Row Level Security (RLS) policies

ALTER TABLE public.user\_profiles ENABLE ROW LEVEL SECURITY;

ALTER TABLE public.universities ENABLE ROW LEVEL SECURITY;

ALTER TABLE public.domains ENABLE ROW LEVEL SECURITY;

ALTER TABLE public.subjects ENABLE ROW LEVEL SECURITY;

ALTER TABLE public.resources ENABLE ROW LEVEL SECURITY;

ALTER TABLE public.user\_submitted\_resources ENABLE ROW LEVEL SECURITY;

ALTER TABLE public.user\_resource\_requests ENABLE ROW LEVEL SECURITY;

ALTER TABLE public.user\_saved\_resources ENABLE ROW LEVEL SECURITY;

-- Basic RLS policies (adjust as needed)

-- Users can read their own profile

CREATE POLICY "Users can view own profile" ON user\_profiles

FOR SELECT USING (auth.uid() = id);

-- Users can update their own profile

CREATE POLICY "Users can update own profile" ON user\_profiles

FOR UPDATE USING (auth.uid() = id);

-- Everyone can read universities, domains, subjects

CREATE POLICY "Public read access" ON universities FOR SELECT TO authenticated USING (true);

CREATE POLICY "Public read access" ON domains FOR SELECT TO authenticated USING (true);

CREATE POLICY "Public read access" ON subjects FOR SELECT TO authenticated USING (true);

-- Users can read approved resources

CREATE POLICY "Users can read approved resources" ON resources

FOR SELECT TO authenticated USING (is\_approved = true);

-- Users can read their own submitted/saved resources and requests

CREATE POLICY "Users can manage own saves" ON user\_saved\_resources

FOR ALL TO authenticated USING (auth.uid() = user\_id);

CREATE POLICY "Users can manage own requests" ON user\_resource\_requests

FOR ALL TO authenticated USING (auth.uid() = user\_id);

CREATE POLICY "Users can view own submissions" ON user\_submitted\_resources

FOR SELECT TO authenticated USING (auth.uid() = user\_id);

-- Function to automatically create user profile on signup

CREATE OR REPLACE FUNCTION public.handle\_new\_user()

RETURNS trigger AS $$

BEGIN

INSERT INTO public.user\_profiles (id, full\_name)

VALUES (new.id, new.raw\_user\_meta\_data->>'full\_name');

RETURN new;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

-- Trigger to create profile on user signup

CREATE TRIGGER on\_auth\_user\_created

AFTER INSERT ON auth.users

FOR EACH ROW EXECUTE PROCEDURE public.handle\_new\_user();