Villa Manager Database Documentation

This document contains the complete Supabase database schema and Row Level Security (RLS) policies for the Villa Manager application.

Database Schema

Complete SQL Schema

```
-- WARNING: This schema is for context only and is not meant to be run.
-- Table order and constraints may not be valid for execution.

CREATE TABLE public.bookings (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  start_date date NOT NULL,
  end_date date NOT NULL,
  guest_name text NOT NULL,
  total_amount numeric NOT NULL DEFAULT 0,
  notes text,
  created_at timestamp with time zone DEFAULT now(),
  updated_at timestamp with time zone DEFAULT now(),
  user_id uuid,
  prepayment numeric NOT NULL DEFAULT '0'::numeric,
  source USER-DEFINED NOT NULL DEFAULT 'DIRECT'::booking_source,
  airbnb_id text,
  booking_com_id text,
  CONSTRAINT bookings_pkey PRIMARY KEY (id),
  CONSTRAINT bookings_user_id_fkey FOREIGN KEY (user_id) REFERENCES
auth.users(id)
);

CREATE TABLE public.expense_categories (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  name text NOT NULL UNIQUE,
  created_at timestamp with time zone DEFAULT now(),
  CONSTRAINT expense_categories_pkey PRIMARY KEY (id)
);

CREATE TABLE public.expenses (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  category_id uuid,
  amount numeric NOT NULL DEFAULT 0,
  date date NOT NULL,
  description text,
  created_at timestamp with time zone DEFAULT now(),
  updated_at timestamp with time zone DEFAULT now(),
  user_id uuid,
  months ARRAY CHECK (months IS NULL OR array_length(months, 1) > 0 AND
array_position(months, 0) IS NULL AND array_position(months, 13) IS NULL),
```

```
  CONSTRAINT expenses_pkey PRIMARY KEY (id),
  CONSTRAINT expenses_category_id_fkey FOREIGN KEY (category_id) REFERENCES
public.expense_categories(id),
  CONSTRAINT expenses_user_id_fkey FOREIGN KEY (user_id) REFERENCES
auth.users(id)
);

CREATE TABLE public.profiles (
  id uuid NOT NULL,
  updated_at timestamp with time zone,
  username text UNIQUE CHECK (char_length(username) >= 3),
  avatar_url text,
  website text,
  phone_number text,
  company_name text,
  email text,
  address text,
  vat_number text,
  full_name text,
  airbnb_ical_url text,
  booking_com_ical_url text,
  CONSTRAINT profiles_pkey PRIMARY KEY (id),
  CONSTRAINT profiles_id_fkey FOREIGN KEY (id) REFERENCES auth.users(id)
);
```

Table Descriptions

bookings

- **Purpose**: Stores villa booking information
- **Key Fields**:
  - `id`: UUID primary key
  - `start_date`, `end_date`: Booking date range
  - `guest_name`: Name of the guest
  - `total_amount`, `prepayment`: Financial amounts
  - `source`: Booking source enum (AIRBNB, BOOKING, DIRECT)
  - `airbnb_id`, `booking_com_id`: External platform IDs
  - `user_id`: Foreign key to auth.users for user isolation

expense_categories

- **Purpose**: Master list of expense categories
- **Key Fields**:
  - `id`: UUID primary key
  - `name`: Unique category name

expenses

- **Purpose**: Tracks villa expenses

- **Key Fields**:
  - `id`: UUID primary key
  - `category_id`: Foreign key to expense_categories
  - `amount`: Expense amount
  - `date`: Expense date
  - `months`: Array of months for recurring expenses (1-12)
  - `user_id`: Foreign key to auth.users for user isolation

profiles

- **Purpose**: User profile information extending auth.users
- **Key Fields**:
  - `id`: UUID primary key (matches auth.users.id)
  - `full_name`, `email`, `username`: User identity
  - `company_name`, `vat_number`: Business information
  - `airbnb_ical_url`, `booking_com_ical_url`: External calendar sync URLs

Row Level Security (RLS) Policies

Public Schema Policies

bookings Table Policies


```
{
  "schema_name": "public",
  "table_name": "bookings",
  "policy_name": "Users can create their own bookings",
  "command": "INSERT",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": null,
  "with_check_condition": "(auth.uid() = user_id)"
}
```

```
{
  "schema_name": "public",
  "table_name": "bookings",
  "policy_name": "Users can delete their own bookings",
  "command": "DELETE",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "(auth.uid() = user_id)",
  "with_check_condition": null
}
```

```
{
  "schema_name": "public",
  "table_name": "bookings",
  "policy_name": "Users can update their own bookings",
  "command": "UPDATE",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "(auth.uid() = user_id)",
  "with_check_condition": "(auth.uid() = user_id)"
}
```

```
{
  "schema_name": "public",
  "table_name": "bookings",
  "policy_name": "Users can view their own bookings",
  "command": "SELECT",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "(auth.uid() = user_id)",
  "with_check_condition": null
}
```


expense_categories Table Policies


```
{
  "schema_name": "public",
  "table_name": "expense_categories",
  "policy_name": "Users can view expense categories",
  "command": "SELECT",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "true",
  "with_check_condition": null
}
```


expenses Table Policies


```
{
  "schema_name": "public",
  "table_name": "expenses",
  "policy_name": "Users can create their own expenses",
  "command": "INSERT",
  "to_roles": "authenticated",
  "is_permissive": true,
```

```
  "using_condition": null,
  "with_check_condition": "(auth.uid() = user_id)"
}
```

```
{
  "schema_name": "public",
  "table_name": "expenses",
  "policy_name": "Users can delete their own expenses",
  "command": "DELETE",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "(auth.uid() = user_id)",
  "with_check_condition": null
}
```

```
{
  "schema_name": "public",
  "table_name": "expenses",
  "policy_name": "Users can update their own expenses",
  "command": "UPDATE",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "(auth.uid() = user_id)",
  "with_check_condition": "(auth.uid() = user_id)"
}
```

```
{
  "schema_name": "public",
  "table_name": "expenses",
  "policy_name": "Users can view their own expenses",
  "command": "SELECT",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "(auth.uid() = user_id)",
  "with_check_condition": null
}
```


profiles Table Policies


```
{
  "schema_name": "public",
  "table_name": "profiles",
  "policy_name": "Users can delete their own profile",
```

```
  "command": "DELETE",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "(auth.uid() = id)",
  "with_check_condition": null
}
```

```
{
  "schema_name": "public",
  "table_name": "profiles",
  "policy_name": "Users can insert their own profile",
  "command": "INSERT",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": null,
  "with_check_condition": "(auth.uid() = id)"
}
```

```
{
  "schema_name": "public",
  "table_name": "profiles",
  "policy_name": "Users can only view their own profile",
  "command": "SELECT",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "(auth.uid() = id)",
  "with_check_condition": null
}
```

```
{
  "schema_name": "public",
  "table_name": "profiles",
  "policy_name": "Users can update own profile.",
  "command": "UPDATE",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "(auth.uid() = id)",
  "with_check_condition": "(auth.uid() = id)"
}
```


Storage Schema Policies

objects Table Policies (avatars bucket)


```
```

```
{
  "schema_name": "storage",
  "table_name": "objects",
  "policy_name": "Allow authenticated delete on own avatars",
  "command": "DELETE",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "((bucket_id = 'avatars'::text) AND
((storage.foldername(name))[1] = (auth.uid())::text))",
  "with_check_condition": null
}
```

```
{
  "schema_name": "storage",
  "table_name": "objects",
  "policy_name": "Allow authenticated insert on own avatars",
  "command": "INSERT",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": null,
  "with_check_condition": "((bucket_id = 'avatars'::text) AND
((storage.foldername(name))[1] = (auth.uid())::text))"
}
```

```
{
  "schema_name": "storage",
  "table_name": "objects",
  "policy_name": "Allow authenticated update on own avatars",
  "command": "UPDATE",
  "to_roles": "authenticated",
  "is_permissive": true,
  "using_condition": "((bucket_id = 'avatars'::text) AND
((storage.foldername(name))[1] = (auth.uid())::text))",
  "with_check_condition": "((bucket_id = 'avatars'::text) AND
((storage.foldername(name))[1] = (auth.uid())::text))"
}
```

```
{
  "schema_name": "storage",
  "table_name": "objects",
  "policy_name": "Allow limited public access to avatars",
  "command": "SELECT",
  "to_roles": "authenticated, anon",
  "is_permissive": true,
  "using_condition": "((bucket_id = 'avatars'::text) AND
```

```
(storage.extension(name) = ANY (ARRAY['jpg'::text, 'jpeg'::text, 'png'::text,
'gif'::text])))",
  "with_check_condition": null
}
```

Security Model Summary

User Isolation

- All user data is isolated by `user_id` (foreign key to `auth.users.id`)
- Users can only access their own bookings, expenses, and profile
- RLS policies enforce data separation at the database level

Public Access

- Expense categories are readable by all authenticated users (shared master data)
- Avatar images are publicly accessible with file type restrictions

Storage Security

- Avatar uploads are restricted to user-specific folders
- Only image files (jpg, jpeg, png, gif) are allowed for public access

Usage Notes

- When creating new components that interact with the database, ensure they respect the RLS policies
- All database operations should include the `user_id` for proper isolation
- The `booking_source` enum has three values: `AIRBNB`, `BOOKING`, `DIRECT`
- Expense `months` array should contain values 1-12 (January-December)

Related Files

- `src/lib/database.types.ts`: TypeScript types generated from this schema
- `src/lib/definitions.ts`: App-specific type definitions
- `supabase/migrations/`: Database migration files