Villa Manager Codebase Index

Project Overview

This is a villa rental management application built with Next.js 14, designed for property owners to manage bookings, expenses, revenue, and reports. It supports multilingual functionality (English and Albanian) and integrates with Supabase for authentication, database, and storage. The app features a responsive dashboard with calendar views, booking management, financial tracking, and PDF report generation.

Key features:

- User authentication and profile management
- Booking CRUD operations with calendar visualization
- Expense tracking with categorization
- Revenue analysis with charts and monthly reports
- iCal synchronization for external platforms (Airbnb, Booking.com)
- PDF invoice and report generation
- CSRF protection and secure server actions

Technologies and Frameworks

- **Frontend Framework**: Next.js 14 (App Router, Server Components, Server Actions)
- **Backend/Database**: Supabase (PostgreSQL, Auth, Storage)
- **Styling**: Tailwind CSS 4, shadcn/ui components, clsx, class-variance-authority (cva)
- **State Management**: React Hook Form, Zod for validation
- **UI Components**: Radix UI primitives, Lucide React icons
- **Charts/Visualization**: Recharts, FullCalendar (React wrapper)
- **Internationalization (i18n)**: next-i18next, react-i18next
- **PDF Generation**: @react-pdf/renderer
- **Date Handling**: date-fns (with Albanian locale support)
- **Testing**: Jest, Playwright (E2E), Testing Library
- **Other Libraries**:
  - Authentication: @supabase/auth-ui-react
  - File Handling: ExcelJS (for exports), node-ical (for sync)
  - Utilities: sonner (toasts), next-themes (theme provider), embla-carousel-react
- **Build Tools**: TypeScript 5, ESLint 9, PostCSS
- **Deployment**: Vercel (with analytics and speed insights)

Project Structure

The project follows Next.js App Router conventions with organized directories for components, lib, and app routes.

Root Files

- `package.json`: Dependencies and scripts (dev, build, test, lint)
- `next.config.js`: Next.js configuration (images, webpack optimizations, i18n)
- `next-i18next.config.js`: i18n setup (locales: en, sq; default: en)
- `tailwind.config.js`: Tailwind CSS configuration
- `tsconfig.json`: TypeScript configuration (strict mode, paths: @/* -> src/*)
- `middleware.ts`: Locale detection, auth checks, CSRF handling
- `CODEBASE_INDEX.md`: This document

Public Assets

- `public/`: Static files (images, fonts, locales JSON for i18n)
    - `locales/en/common.json`, `locales/sq/common.json`: Translation files

Source Code (src/)

- **app/**: Next.js pages and layouts (internationalized with [lang] segment)
    - `[lang]/`: Root layout with metadata, theme provider, fonts (Inter)
    - `[lang]/page.tsx`: Redirects to login
    - `[lang]/(auth)/`: Authentication routes
        - `login/page.tsx`: Supabase Auth UI with custom views (sign_in, sign_up, update_password)
    - `[lang]/(app)/`: Protected app routes (requires auth via middleware)
        - `layout.tsx`: App shell with Header, Sidebar, Toaster
        - `dashboard/page.tsx`: Overview with custom calendar, today's check-ins/outs, upcoming bookings
        - `bookings/`: List view (card/table toggle), search, filters, CRUD modals
        - `expenses/`: Expense management with forms and filters
        - `revenue/`: Financial dashboard with charts, summaries, reports carousel
        - `settings/`: Profile, iCal sync, backup downloads
    - `api/`: API routes
        - `auth/`: Supabase auth callbacks and verification
        - `cron/ical-sync/`: Background iCal synchronization
        - `csrf/`: CSRF token generation

- **components/**: Reusable UI components (shadcn/ui based)
    - **ui/**: Primitive components (button, card, table, dialog, etc.)
    - **shared/**: Header, Sidebar (navigation)
    - **bookings/**: BookingCard, BookingForm
    - **dashboard/**: CustomBookingCalendar (FullCalendar integration)
    - **expenses/**: ExpenseCard, ExpenseForm, FilterSheet
    - **revenue/**: Charts (RevenueLineChart), Cards (MonthlySummaryCard, BookingListCard), Reports (carousel, PDF components)
    - **settings/**: iCalSettings, DownloadBackupButton
    - Others: LanguageSwitcher, SearchBar, theme-provider, auth forms

- **lib/**: Utilities, types, actions

  - **supabase/**: Client/server/middleware clients for Supabase integration
  - **actions/**: Server actions for CRUD (bookings.ts, expenses.ts, profile.ts, reports.ts, ical-sync.ts, backup.ts, invoice.ts)
  - **pdf/**: PDF generation (InvoicePDF, MonthlyReportPDF)
  - **csrf.ts/client.ts**: CSRF protection (token generation, verification)
  - **database.types.ts**: Supabase-generated types (bookings, expenses, profiles, enums)
  - **database-documentation.md**: Complete database schema and RLS policies documentation
  - **definitions.ts**: App-specific types (Booking, Expense, Profile)
  - **utils.ts**: Helpers (formatCurrency, cn for classNames)
  - **dictionary.ts/translations.ts**: i18n dictionary loading and category translation
  - **fullcalendar-sq-locale.ts**: Albanian locale for FullCalendar

Supabase

- `supabase/migrations/`: Database migrations (e.g., add_airbnb_id_to_bookings.sql, keep_alive table)

Testing

- `e2e/`: Playwright tests (auth.spec.ts, booking.spec.ts, csrf.spec.ts)
- Jest configuration in package.json for unit/integration tests

Database Schema (Supabase PostgreSQL)

Complete database schema and RLS policies are documented in `src/lib/database-documentation.md`.

Generated types in `src/lib/database.types.ts`. Key tables:

- **bookings**: Villa booking information with user isolation
- **expenses**: Expense tracking with categorization and recurring options
- **expense_categories**: Master list of expense categories (shared across users)
- **profiles**: User profile information extending auth.users

Security Model:

- Row Level Security (RLS) policies enforce user data isolation
- Users can only access their own bookings, expenses, and profiles
- Expense categories are readable by all authenticated users
- Avatar storage has user-specific folder restrictions

For complete SQL schema and detailed RLS policies, refer to the database documentation.

Authentication and Security

- **Supabase Auth**: Email/password, OAuth (providers disabled in UI), magic links, password recovery
- **Middleware**: Locale negotiation, auth checks for protected routes, CSRF token setting/verification
- **Server Actions**: CSRF-protected with Zod validation, user_id enforcement
- **Client-Side**: Supabase browser client, auth state listeners for redirects

Protected Routes: /dashboard, /bookings, /revenue, /expenses, /settings (via middleware)

Internationalization (i18n)

- Supported Locales: en (default), sq (Albanian)
- Configuration: next-i18next with cookie-based locale persistence
- Translations: JSON files in public/locales, loaded via getDictionary()
- Date Formatting: date-fns with sq locale
- UI: LanguageSwitcher component, RTL/LTR not implemented

Key Modules and Flows

1. **Authentication Flow**:

    - Login via Supabase Auth UI (customized views)
    - Middleware refreshes session, redirects unauth users
    - Profile updates via server actions

2. **Booking Management**:

    - Dashboard: Custom calendar (FullCalendar), check-in/out cards
    - Bookings Page: CRUD with forms, search/filters, card/table views
    - Actions: createOrUpdateBooking, deleteBooking (Zod validated)

3. **Expense Management**:

    - Expenses Page: List with forms, category filters
    - Actions: createOrUpdateExpense, deleteExpense

4. **Revenue and Reports**:

    - Revenue Page: Charts (Recharts), summaries, future/past bookings
    - Reports: Monthly carousel, PDF generation (invoices, reports)
    - Calculations: Server-side revenue allocation across months

5. **Integrations**:

    - iCal Sync: Cron job API route for Airbnb/Booking.com
    - Backups: SQL dumps via actions
    - Excel Exports: Via ExcelJS (not fully implemented in index)

6. **PDF Generation**:

    - lib/pdf/: InvoicePDF, MonthlyReportPDF with custom styles

Development and Deployment Notes

- **Scripts**: `npm run dev` (local), `npm run build` (production), `npm test` (Jest/Playwright)
- **Environment**: NEXT_PUBLIC_SUPABASE_* vars for Supabase
- **Caching**: Next.js cache handler, revalidatePath for dynamic updates
- **Testing**: Unit (Jest), E2E (Playwright), coverage via npm run test:coverage

- **Linting**: ESLint with security plugin
- **Deployment**: Vercel-ready (vercel.json, analytics)

Potential Improvements

- Add more tests for actions and components
- Implement full iCal two-way sync
- Enhance mobile responsiveness
- Add role-based access if multi-user
- Optimize PDF generation for large reports

This index provides a high-level overview. For detailed implementation, refer to specific files.