# Overview

# PulaFund is a comprehensive crowdfunding platform designed to help Namibian entrepreneurs raise funds for innovative projects across various categories including Technology, Health, Environment, and Social initiatives. The platform enables campaign creators to showcase their projects with detailed descriptions, funding goals in Namibian Dollars (NAD), and progress tracking, while allowing backers to discover and support campaigns they believe in. Built as a full-stack application, PulaFund features a modern React frontend with TypeScript, a Node.js/Express backend, and in-memory storage for development.

# Recent Changes (August 2025)

# **Namibian Localization**: Updated all creator profiles to authentic Namibian names including Amaamu Panduleni, Tulipohamba Shigwedha, and Dr. Julius Matundu

# **Currency Conversion**: Changed all monetary displays from USD to Namibian Dollar (N$) throughout the platform

# **Sample Data**: Adjusted funding amounts to realistic NAD values with campaigns ranging from N$592,020 to N$1,612,206

# **Complete Deployment Ready**: Application successfully tested and ready for production deployment

# User Preferences

# Preferred communication style: Simple, everyday language.

# System Architecture

# Frontend Architecture

# The client-side application is built with React 18 and TypeScript, utilizing Vite as the build tool for fast development and optimized production builds. The UI framework is based on shadcn/ui components with Radix UI primitives, providing accessible and customizable interface elements. Styling is handled through Tailwind CSS with a custom design system that includes predefined color schemes for different campaign categories and responsive design patterns.

# Key architectural decisions:

# **Component Library**: shadcn/ui was chosen for its accessibility-first approach and consistent design patterns

# **State Management**: TanStack Query (React Query) handles server state management, caching, and API interactions

# **Routing**: Wouter provides lightweight client-side routing without the complexity of React Router

# **Form Handling**: React Hook Form with Zod validation ensures type-safe form management

# **Animation**: Framer Motion adds smooth transitions and interactive animations

# The application follows a component-based architecture with clear separation between UI components, pages, and business logic. Path aliases are configured for clean imports, and the build process outputs to a dist/public directory for easy deployment.

# Backend Architecture

# The server is implemented using Express.js with TypeScript, providing a REST API for campaign management, user operations, and pledge processing. The architecture emphasizes middleware-based request processing with comprehensive logging and error handling.

# Core design patterns:

# **Storage Abstraction**: An IStorage interface allows switching between different data persistence implementations (currently includes both in-memory and database storage)

# **Route Organization**: API endpoints are centralized in a routes module with clear resource-based URL patterns

# **Request Logging**: Custom middleware tracks API performance and response data for debugging

# **Error Handling**: Centralized error handling middleware provides consistent error responses

# Data Storage Solutions

# The application uses PostgreSQL as the primary database with Drizzle ORM for type-safe database operations. The schema is defined in TypeScript with automatic type generation, ensuring consistency between database structure and application code.

# Database design decisions:

# **Primary Keys**: UUID-based primary keys provide scalability and avoid sequential ID enumeration

# **Relationships**: Foreign key constraints maintain data integrity between users, campaigns, pledges, and updates

# **Decimal Precision**: Financial amounts use decimal fields with appropriate precision for monetary calculations

# **Timestamps**: Created/updated timestamps enable audit trails and data versioning

# The schema supports essential entities: users with profiles, campaigns with detailed metadata, pledges with backer information, and campaign updates for ongoing communication.

# Development and Build Process

# The development environment integrates both frontend and backend with hot reloading and error overlays. The build process creates optimized bundles for both client and server components.

# Build configuration:

# **Frontend**: Vite bundles the React application with code splitting and asset optimization

# **Backend**: esbuild compiles the Express server with external package handling for Node.js deployment

# **Database**: Drizzle Kit handles schema migrations and database synchronization

# **Development**: TypeScript checking and path resolution work across the entire monorepo structure

# External Dependencies

# Database Services

# **Neon Database**: Serverless PostgreSQL provider (@neondatabase/serverless) for scalable database hosting

# **Connection Pooling**: Built-in connection management for efficient database resource utilization

# UI and Design System

# **Radix UI**: Comprehensive set of accessible, unstyled UI primitives for building consistent user interfaces

# **Tailwind CSS**: Utility-first CSS framework with custom configuration for design tokens and responsive design

# **Framer Motion**: Animation library for smooth transitions, gestures, and interactive elements

# **Lucide React**: Icon library providing consistent iconography throughout the application

# Development Tools

# **TypeScript**: Static type checking across the entire application stack

# **Drizzle ORM**: Type-safe database toolkit with migration management and query building

# **TanStack Query**: Data fetching and caching library for efficient API state management

# **React Hook Form**: Performant form library with validation integration

# **Zod**: Schema validation library for runtime type checking and form validation

# Build and Development

# **Vite**: Next-generation frontend build tool with fast HMR and optimized production builds

# **esbuild**: Fast JavaScript bundler for server-side code compilation

# **PostCSS**: CSS processing with Tailwind CSS and autoprefixer integration

# **ESBuild**: High-performance bundler for server-side TypeScript compilation

# ddincdiamond