

TeamJesus / PostClips Microsite

A production-ready Next.js donor-facing microsite built with modern design, animations, and accessibility.

Overview

Goal: Raise \$8.142M to tell 8.142 billion people about Jesus.

Impact: \$1 = 1,000 people reached with the Gospel message.

This microsite features:

- ✨ Animations with Framer Motion
- 🎨 Professional design with Lucide icons
- 📊 Live donation tracking with optimistic updates
- 📈 Results-based funding model (\$0.01 per 10 views)
- 💰 100% of donations go directly to clipper commissions
- ♿ WCAG 2.2 AA accessible
- 🚀 Production-ready architecture with mocked payment processing

Quick Start

Prerequisites

- Node.js 18+
- npm, yarn, or pnpm

Installation

```
# Install dependencies
npm install

# Run development server
npm run dev
```

Open <http://localhost:3000> to view the site.

Available Scripts

```
# Development
npm run dev          # Start dev server at localhost:3000

# Production
npm run build         # Build for production
npm run start         # Start production server

# Testing
npm test              # Run unit tests with Vitest
npm run test:ui       # Run tests with UI

# Linting
npm run lint          # Run ESLint
```

Design Features

What Makes This Tick

Modern Tech Stack:

- **Framer Motion** - Smooth animations throughout
- **Lucide React** - 30+ professional icons
- **Radix UI** - Accessible accordion, tabs, progress components
- **Google Fonts** - Poppins (display) + Inter (body)

- **TailwindCSS** - Custom gradients, shadows, animations

Key Design Patterns:

- 🌊 Glassmorphism with backdrop blur
- ✨ Shimmer effects on CTAs
- 🎪 Micro-interactions on hover
- 📱 Mobile-first responsive design
- ♿ Full keyboard navigation
- 🎭 Scroll-triggered animations

Color Palette

Primary (Indigo)

- 50-900 shades
- Main: #4338ca (700)

Accent (Emerald)

- 50-900 shades
- Main: #059669 (600)

Gradients

- Hero: Indigo → Purple → Emerald
- Buttons: Primary gradient with shimmer overlay
- Backgrounds: Soft blur circles

📁 Project Structure

```
team_jesus/  
├─ app/  
│   └─ api/metrics/route.ts      # Live metrics API (GET/POST)
```

```
|   └─ layout.tsx           # Root layout with SEO & fonts
|   └─ page.tsx             # Redirects to /teamjesus
|   └─ teamjesus/page.tsx   # Main microsite page
|   └─ globals.css          # Global styles + Tailwind
└─ components/
|   └─ DonateWidget.tsx     # Donation form (Radix Tabs, quick
amounts)
|   └─ Hero.tsx             # Animated hero with live counters
|   └─ ImpactCounter.tsx    # Animated number counter
|   └─ GoalProgress.tsx     # Animated progress bars with
shimmer
|   └─ HowItWorksStepper.tsx # 3-step process with icons
|   └─ WhyThisMatters.tsx   # Radix Accordion with verses
|   └─ RewardsInHeaven.tsx  # 4-card grid with icons
|   └─ Stewardship.tsx      # Trust messaging with stat badges
|   └─ TithingMonthly.tsx   # Monthly giving CTA
|   └─ TrustBand.tsx        # Partner logos (placeholders)
|   └─ FAQ.tsx              # Radix Accordion FAQ
|   └─ Footer.tsx           # Gradient footer with links
└─ providers/
|   └─ PaymentProvider.ts    # Payment provider interface
|   └─ mockPayment.ts       # Mock implementation (demo)
└─ hooks/
|   └─ useMetrics.ts         # SWR hook for live metrics
└─ lib/
|   └─ config.ts             # App configuration constants
|   └─ utils.ts              # Utility functions (cn)
└─ content/
|   └─ verses.ts             # Scripture references + summaries
└─ utils/
|   └─ analytics.ts          # Analytics tracking (console.log)
└─ __tests__/
|   └─ math.spec.ts          # People-per-dollar calculations
|   └─ donation.spec.tsx     # Donation widget tests
|   └─ accordion.spec.tsx    # Accordion accessibility tests
└─ tailwind.config.ts        # Custom theme, animations
```

└─ vitest.config.ts

Vitest configuration

└─ package.json

Component Highlights

Hero Component

- Animated floating background blobs
- Staggered text animations
- Gradient text heading with emoji sparkle
- Shimmer effect on primary CTA
- Glassmorphic stats card with backdrop blur
- Hover scale animations

DonateWidget Component

- Radix UI Tabs (One-time/Monthly)
- Quick amount buttons (\$10-\$250)
- Dollar sign icon in input
- Animated impact preview
- Loading spinner
- Success state with celebration design
- Shimmer button effect

Accordion Components (WhyThisMatters, FAQ)

- Radix UI Accordion
- Smooth height animations
- Rotating chevron indicators
- Keyboard accessible (Tab, Enter, Space)
- Focus rings that follow rounded corners

- Staggered entrance animations

Progress Bars (GoalProgress)

- Animated fill on mount
- Shimmer effect overlay
- Dual progress (dollars & people)
- Gradient fills

Configuration

Key Constants

Located in `lib/config.ts`:

```
GOAL_DOLLARS = 8142000           // $8.142M goal
GOAL_PEOPLE = 8142000000         // 8.142 billion
PEOPLE_PER_DOLLAR = 1000         // Impact ratio
PAYOUT_RATE = 0.01              // per 10 views
INITIAL_PEOPLE_REACHED = 41638201 // seed data
```

Environment Variables (Future)

```
# .env.local
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_live...
STRIPE_SECRET_KEY=sk_live...
NEXT_PUBLIC_GA_TRACKING_ID=G-...
```

Payment Integration

Current State (Mock)

Uses `mockPaymentProvider`:

- Simulates 1.2s processing delay

- Always succeeds (no actual charges)
- Returns mock transaction IDs

Integrating Real Payments

Stripe

1. Install SDK:

```
npm install @stripe/stripe-js stripe
```

2. Create providers/stripePayment.ts :

```
import { PaymentProvider } from '../PaymentProvider';
import Stripe from 'stripe';

export class StripePaymentProvider implements PaymentProvider {
  private stripe: Stripe;

  constructor(secretKey: string) {
    this.stripe = new Stripe(secretKey, { apiVersion: '2023-10-16' });
  }

  async processDonation(request: DonationRequest):
  Promise<DonationResponse> {
    // Implement Stripe payment intent
    // https://stripe.com/docs/payments/payment-intents
  }
}
```

3. Update components/DonateWidget.tsx :

```
// Replace:
import { mockPaymentProvider } from '@providers/mockPayment';
// With:
import { stripePaymentProvider } from '@providers/stripePayment';
```

TODO locations:

- providers/PaymentProvider.ts - Interface & notes
- components/DonateWidget.tsx - Provider swap
- utils/analytics.ts - Real analytics






Testing

```
# Run all tests
npm test

# Watch mode
npm test -- --watch









# With UI
npm run test:ui
```

Test Coverage:

-  People-per-dollar math (6 tests)
-  Donation widget interactions (8 tests)
-  Accordion accessibility (8 tests)
-  Keyboard navigation
-  ARIA attributes

Accessibility

WCAG 2.2 AA compliant:

-  Semantic HTML structure
-  All images have alt text
-  Form labels with aria-describedby
-  Keyboard navigation (Tab, Enter, Space)
-  Focus indicators with rounded corners
-  ARIA attributes (expanded, controls, live)
-  Reduced motion support
-  Color contrast meets AA standards

Analytics

Events tracked in `utils/analytics.ts` :

- `donation_amount_changed`
- `donation_frequency_toggled`
- `donation_submit`
- `donation_success`
- `donation_error`
- `accordion_opened`
- `donation_quick_amount`

Integration: Replace `console.log` with GA4, Mixpanel, etc.

API Routes

GET `/api/metrics`

```
{  
  "peopleReached": 41638201,  
  "amountRaised": 41638  
}
```

POST /api/metrics

```
// Request
{ "amount": 100 }

// Response
{
  "peopleReached": 41738201,
  "amountRaised": 41738
}
```

 **Note:** Uses in-memory storage. Replace with database for production.

Deployment

Vercel (Recommended)

```
npm install -g vercel
vercel
```

Other Platforms

Works on any Node.js hosting:

- Netlify
- Railway
- AWS Amplify
- DigitalOcean App Platform

Customization

Change Colors

Edit `tailwind.config.ts` :

```
colors: {
  primary: {
    DEFAULT: '#4338ca', // Your brand color
    // ...
  }
}
```

Adjust Animations

Edit `tailwind.config.ts`:

```
animation: {
  'fade-in': 'fadeIn 0.6s ease-out', // Adjust duration
}
```

Add More Icons

Browse [Lucide Icons](#):

```
import { Heart, Star, Check } from 'lucide-react';
```

Update Fonts

Edit `app/layout.tsx`:

```
import { Inter, Roboto } from 'next/font/google';
```

Customize Gradients

Edit `tailwind.config.ts`:

```
backgroundImage: {
  'my-gradient': 'linear-gradient(to right, #667eea, #764ba2)',
}
```



Content Updates

Verse References

Edit `content/verses.ts` :

```
{
  id: 'unique-id',
  title: 'Title',
  reference: 'Book Chapter:Verse',
  summary: '1-2 sentence summary',
}
```

FAQ

Edit `components/FAQ.tsx` - Add to `faqItems` array.

Partner Logos

Replace placeholders in `components/TrustBand.tsx` with actual logos.



Known Limitations

1. **In-memory metrics** - Resets on server restart → Use database
2. **Mock payments** - No actual charges → Implement real provider
3. **Placeholder logos** - Replace in TrustBand component
4. **No authentication** - Add if admin dashboard needed
5. **No email receipts** - Implement after payment integration



Dependencies

Production:

- `@radix-ui/react-accordion` - Accessible accordions

- `@radix-ui/react-progress` - Progress bars
- `@radix-ui/react-tabs` - Tab components
- `framer-motion` - Animations
- `lucide-react` - Icons
- `class-variance-authority` - Component variants
- `clsx` - Conditional classes
- `tailwind-merge` - Merge Tailwind classes
- `swr` - Data fetching

Development:

- `vitest` - Testing framework
 - `@testing-library/react` - Component testing
 - `@vitejs/plugin-react` - React support
-

Built with  for #TeamJesus