Software Requirements Specification

VERZZ Corporate Product Showcase Website

1 Introduction

1.1 Purpose

This document defines the Software Requirements Specification (SRS) for the **VERZZ Corporate Product Showcase Website**. The intent is to specify the system from a *technical and architectural standpoint*, clearly outlining scope, constraints, and implementation decisions.

This document is written for developers, reviewers, and technical stakeholders.

1.2 Scope

The VERZZ website is a **frontend-only, public-facing product presentation platform**. It communicates what VERZZ is, what problem it addresses, and how its product works *conceptually*.

The website:

- Does not implement VERZZ's internal product systems
- Does not include backend services, APIs, or databases
- Focuses on visual storytelling, performance, and interaction

This system is comparable to modern product marketing sites (e.g., Apple, Stripe).

1.3 Design Intent

The design intent is to deliver a:

- High-performance static web experience
- Smooth, animation-driven interaction model
- Modern component-based frontend architecture

2 System Overview

2.1 Architecture

- **Architecture Type**: Frontend-only web application
- **Framework**: Next.js

- **Rendering Model**:
  - Static Site Generation (SSG)
  - Client-Side Rendering (CSR) for interactions
- **Backend**: None
- **Database**: None

2.2 Operating Environment

- Chrome, Firefox, Safari, Edge
- Desktop and mobile browsers
- JavaScript-enabled execution environment

3 Functional Requirements

3.1 Content Presentation

- The system shall present structured information about VERZZ
- The system shall visually explain product features
- The system shall support:
  - Scroll-driven animations
  - Interactive sections
  - Smooth transitions

3.2 Navigation and Interaction

- Client-side navigation without full reloads
- Consistent interaction patterns across devices
- Immediate UI feedback

3.3 User Communication

- Email redirection (mailto / Gmail pop-up)
- Embedded third-party contact forms
- No data persistence within the website

4 Non-Functional Requirements

4.1 Performance

- Static assets shall be optimized through code splitting, compression, and caching strategies
- Client-side interactions shall remain responsive and non-blocking

4.2 Maintainability

- Component-driven architecture
- Version-controlled codebase
- Deterministic builds and deployments

4.3 Security

- No sensitive data handling
- No authentication or credentials
- Reliance on provider-side security for integrations

5 Technology Stack

5.1 Frontend

- Next.js
- HTML5, CSS3, JavaScript (ES6+)

5.2 Integrations

- Email connectors
- Embedded forms
- Analytics (optional)

5.3 Hosting

- Static deployment (e.g., Vercel)
- CDN-backed asset delivery

End of Document