# Figma-Website UI Comparison Tool

Technical Documentation

## Overview

This is a full-stack web application that automatically compares Figma designs with live websites to detect visual inconsistencies between design and implementation.

## Tech Stack

### *Backend (Python/FastAPI)*

- **FastAPI** - Modern async Python web framework for REST APIs
- **Playwright** - Browser automation for capturing website screenshots
- **ReportLab** - PDF report generation
- **Pillow (PIL)** - Image processing and comparison
- **SQLite** - Lightweight database for comparison history
- **Pydantic** - Data validation and settings management

### *Frontend (React/TypeScript)*

- **React 18** with TypeScript
- **Vite** - Fast build tool and dev server
- **TailwindCSS** - Utility-first CSS framework
- **Axios** - HTTP client for API calls
- **React-Compare-Slider** - Visual side-by-side comparison

## Architecture

The application follows a client-server architecture:

1. **React Frontend** (Port 5173) - User interface for input and results
2. **FastAPI Backend** (Port 8000) - REST API handling comparison logic
3. **Figma API** - External service for design data extraction
4. **Playwright** - Headless browser for website capture

## Key Features

### *1. Figma Design Extraction*

Connects to Figma REST API to extract design data. Supports Personal Access Tokens and OAuth 2.0 authentication. Uses /files/{key}/nodes endpoint for specific frames (faster for large files). Caches API responses for 30 minutes.

### *2. Website Capture*

Uses Playwright to launch headless Chromium browser. Captures full-page screenshots at specified viewport sizes. Extracts computed CSS styles from DOM elements.

### *3. Comparison Engine*

Structural comparison analyzes design tokens (colors, typography, spacing). Visual comparison does pixel-by-pixel image diff. Hybrid mode combines both approaches. Calculates match score (0-100%).

### *4. Difference Detection Types*

- Color (background, text, border)
- Typography (font family, size, weight)
- Spacing (margins, padding, gaps)
- Dimensions (width, height)
- Layout (position, alignment)
- Missing/Extra elements

### *5. Report Generation*

HTML Report - Interactive web-based report. PDF Report - Professional document with executive summary, match score, visual comparison screenshots, detailed differences with element names and coordinates, severity levels.

### *6. OAuth 2.0 Integration*

Implements Figma OAuth 2.0 flow for higher API rate limits (personal tokens limited to 2 requests/minute). Stores tokens securely and supports token refresh.

## API Endpoints

| Endpoint | Method | Description |
|---|---|---|
| /api/v1/compare | POST | Start a new comparison job |
| /api/v1/progress/{job_id} | GET | Get job progress (polling) |
| /api/v1/report/{job_id} | GET | Get comparison results |
| /api/v1/history | GET | List past comparisons |
| /api/v1/oauth/authorize | GET | Get Figma OAuth URL |
| /api/v1/oauth/callback | GET | OAuth callback handler |
| /api/v1/oauth/status | GET | Check OAuth status |

## Data Flow

1. User submits Figma URL + Website URL + Token
2. Backend creates job and returns job ID
3. Frontend polls /progress/{job_id} for updates
4. Backend extracts Figma design data via API
5. Backend captures website screenshot via Playwright
6. Comparison engine analyzes both and finds differences
7. Reports generated (HTML + PDF)
8. Results returned to frontend with match score

## Project Structure

- **backend/app/api/endpoints.py** - REST API routes
- **backend/app/services/figma_extractor.py** - Figma API integration
- **backend/app/services/figma_oauth.py** - OAuth 2.0 handling
- **backend/app/services/website_analyzer.py** - Playwright capture
- **backend/app/services/ui_comparator.py** - Comparison logic
- **backend/app/services/pdf_generator.py** - PDF reports
- **frontend/src/components/** - React components

## Key Technical Decisions

1. **Async job processing** - Long comparisons run in background, frontend polls
2. **Caching** - Figma API responses cached to avoid rate limits
3. **Node ID support** - Fetch specific frames instead of entire files
4. **Hybrid comparison** - Combines structural and visual analysis

5. **OAuth 2.0** - Higher rate limits vs 2 req/min for personal tokens

## Running the Project

**Backend:** cd backend and pip install -r requirements.txt and uvicorn app.main:app --reload --port 8000

**Frontend:** cd frontend and npm install and npm run dev