DesignSight - MERN Prototype Assignment

Assignment Brief for Applicants

Problem Statement

Design teams struggle with consistent, actionable feedback on their visual designs. Currently, feedback is scattered across Slack threads, design tool comments, and meeting notes. There's no systematic way to analyze designs for accessibility issues, visual hierarchy problems, or role-specific concerns.

Your mission: Build DesignSight, an AI-powered design feedback platform that provides structured, coordinate-anchored feedback on uploaded screen designs, with role-based filtering and collaborative discussion features.

The Challenge

You are tasked with building a MERN stack prototype that solves these key problems:

1. **Inconsistent Design Review:** Teams need systematic analysis of uploaded designs covering accessibility, visual hierarchy, content quality, and UX patterns.
2. **Scattered Feedback:** Design feedback should be anchored to specific screen coordinates and organized by team role (designer, reviewer, product manager, developer).
3. **Poor Handoff Process:** Teams need exportable feedback reports for development handoff and issue tracking.
4. **Limited Collaboration:** Multiple stakeholders should be able to discuss specific feedback items through threaded conversations.

Assignment Requirements

Core Constraints

- **Stack:** MERN (MongoDB, Express, React/Next, Node.js)
- **Timeline:** 72 hours to complete MVP
- **AI Integration:** Must use real AI vision models (no mocks)
- **Local Deployment:** Must run end-to-end with docker-compose
- **Demo Scope:** Keep usage minimal (1-2 test images max)

The "No Mocks" Rule

This is a critical requirement. You must integrate with real AI vision services such as:

- OpenAI GPT-4V, Google Vision API, Anthropic Claude Vision
- Hugging Face models (BLIP-2, LLaVA)
- Self-hosted open-source alternatives

Document API costs and provide `.env.example` with required variables.

MVP Feature Requirements

1. Image Upload & Project Management

Build a system where users can:

- Create projects and upload screen images (PNG/JPG)
- Store images with metadata and unique identifiers
- Handle file validation and basic error cases

2. AI-Powered Design Analysis

Integrate real AI vision models to analyze uploaded screens and generate structured feedback covering:

Analysis Categories:

- **Accessibility:** Color contrast, text readability, navigation issues
- **Visual Hierarchy:** Spacing, alignment, typography consistency
- **Content & Copy:** Tone, clarity, messaging effectiveness
- **UI/UX Patterns:** Button placement, user flow, best practices

Output Requirements:

- Each feedback item must include screen coordinates (x, y, width, height)
- Categorize by severity (high, medium, low)
- Include actionable recommendations

- Tag with relevant team roles

3. Role-Based Feedback Views

Implement four distinct user perspectives:

- **Designer:** Visual hierarchy, typography, spacing, brand consistency
- **Reviewer:** Overall quality, design system adherence, user experience
- **Product Manager:** Usability, content strategy, conversion optimization
- **Developer:** Accessibility requirements, implementation complexity

Users should be able to switch roles and see filtered feedback relevant to their current perspective.

4. Coordinate-Anchored Feedback System

Build an interactive overlay system where:

- Feedback items are visually indicated on the uploaded image
- Users can click highlighted areas to view detailed feedback
- Support both point-based and bounding-box highlighting
- Provide clear visual indicators for different feedback categories

5. Threaded Discussion System

Enable collaborative feedback through:

- Comments on individual feedback items
- Nested reply threads (replies to replies)
- Author identification and timestamps
- Basic CRUD operations for comment management

6. Export & Handoff Features

Generate development-ready exports:

- **PDF Reports:** Visual feedback overlays with detailed descriptions
- **JSON Data:** Machine-readable format for tool integrations
- **Role-Filtered Exports:** Customized reports per team member type

---

Technical Challenges to Solve

AI Integration Architecture

- How will you structure prompts for consistent, structured feedback?
- How will you handle AI service failures and rate limits?
- What's your strategy for extracting coordinate data from vision models?

Real-Time Collaboration

- How will multiple users collaborate on feedback discussions?
- What's your approach to handling concurrent comments and updates?

Performance & Cost Management

- How will you optimize image processing and AI analysis costs?
- What strategies will you use for handling large images efficiently?

User Experience Design

- How will you make coordinate-anchored feedback intuitive and discoverable?
- What's your approach to role-switching and filtered views?

Deliverables

1. Working Prototype

- **GitHub Repository:** Well-organized codebase with clear structure
- **Docker Deployment:** `docker-compose.yml` that runs the full stack locally
- **Environment Setup:** `.env.example` with all required variables documented

2. Documentation

- **README.md:** Clear setup instructions, API provider guidance, cost estimates
- **Design Notes:** 1-page explanation of your architectural decisions and tradeoffs
- **API Documentation:** If building custom APIs, document key endpoints

3. Demo Materials

- **Demo Video:** 2-4 minute walkthrough showing core functionality and edge cases
- **Test Data:** Sample images and scenarios that demonstrate your solution
- **TODO.md:** List any incomplete features or known limitations

4. Testing & Reliability

- **Test Suite:** `npm test` should validate core behaviors
- **Integration Tests:** Verify AI analysis pipeline and coordinate mapping
- **Error Handling:** Graceful failures when AI services are unavailable

---

Evaluation Criteria

Your solution will be assessed on:

Core Functionality (35%)

- Does the AI integration work reliably with real providers?
- Are feedback items properly anchored to image coordinates?
- Do role-based views filter content appropriately?
- Is the comment threading system functional?

Architecture & Design (20%)

- Is the codebase well-structured and maintainable?
- Are architectural decisions documented and justified?
- How well does the system handle edge cases and failures?

User Experience (15%)

- Is the coordinate-anchored feedback system intuitive?
- How clear is the role-switching and filtering experience?
- Are the export features useful for real handoff scenarios?

Reliability & Testing (10%)

- Are core behaviors covered by meaningful tests?
- Does the system handle AI service failures gracefully?
- Is error handling comprehensive and user-friendly?

Documentation & Setup (10%)

- Can a reviewer easily run your solution locally?
- Is the AI provider setup clearly documented?
- Are costs and limitations transparently explained?

Security & Privacy (10%)

- Are API keys properly secured?
- Is user data handled appropriately?
- Are there considerations for design confidentiality?

Getting Started

Step 1: Choose Your AI Provider

Research and select an AI vision service. Consider:

- **Cost implications** for your demo scope
- **Setup complexity** for reviewers
- **Output quality** for structured feedback generation
- **Rate limits** and usage restrictions

Step 2: Plan Your Data Flow

Map out how images flow through your system:

- Upload → Storage → AI Analysis → Coordinate Mapping → User Interface

Step 3: Design Your Architecture

Consider:

- How will you structure feedback data?
- What's your approach to role-based filtering?
- How will comments relate to feedback items?

Step 4: Build and Test Iteratively

Start with basic image upload, then add AI integration, then layer on collaboration features.

Success Tips

- **Start simple:** Get basic image upload and AI integration working first
- **Deploy early:** Set up hosting early so you can test the full pipeline in production
- **Document costs:** Be transparent about AI service usage and expenses
- **Handle failures:** AI services can be unreliable - plan for graceful degradation
- **Test with real data:** Use actual design screenshots for realistic testing

- **Focus on core value:** Prioritize AI-driven analysis over complex collaboration features
- **Pre-load demo data:** Include sample designs and analysis in your hosted demo

---

Questions to Consider

- How will you ensure AI-generated coordinates accurately map to visual elements?
- What's your strategy for making feedback actionable and specific?
- How will you handle different image sizes and resolutions?
- What happens when AI analysis fails or returns unexpected results?
- How will you make role-based filtering genuinely useful for different team members?

Remember: This is a prototype to demonstrate your technical capabilities and product thinking within the Monday midnight deadline. Focus on building something that works end-to-end rather than perfect polish on individual features.