Playground AI Design Tool Development Plan

1. Tools, Technologies, and Delivery Methods

Tools, Technologies, and Delivery Methods

There is no specific tech stack that is the right stack. The only correct tech stack is the one you are most comfortable with that solves your problem.

- Comfort makes you focus on the problem you're trying to solve. Development speed and efficiency feel like muscle memory.
- Although in some cases, technology can decide an app's performance, scalability, and maintainability
- Based on delivery methods you will need to make these decisions early.
    - Web application (primary, responsive design)
    - Progressive Web App for mobile devices
    - REST API for potential integrations with other platforms

Frontend

- Next.js 13+ (App Router) with TypeScript
- Tailwind CSS for styling
- shadcn/ui for component library

AI and Image Generation

- Replicate API for accessing various AI models (Stable Diffusion, Flux etc.)

Backend

- Next.js API Routes for serverless backend functions
- Supabase for database, authentication, and storage
- Upstash Redis for queueing, caching and rate limiting

Infrastructure and CI/CD

- Vercel for hosting and CI/CD
- Cloudflare for additional CDN and serverless functions
- GitHub for version control

2.Usage Volume Expectations and Costs

Usage Volume Expectations and Costs

This step is crucial for planning and forecasting your costs. It helps:

- Make sure your app can handle expected traffic (without downtime)
- Budget for cloud resources and third-party services
- Identify potential scalability issues before they become problems
- Make informed decisions about pricing strategies (especially if you see PMF)

User Volume

- Daily users: 500 (peak hours during evenings and weekends)
- Weekly users: 2,500
- Monthly users: 10,000

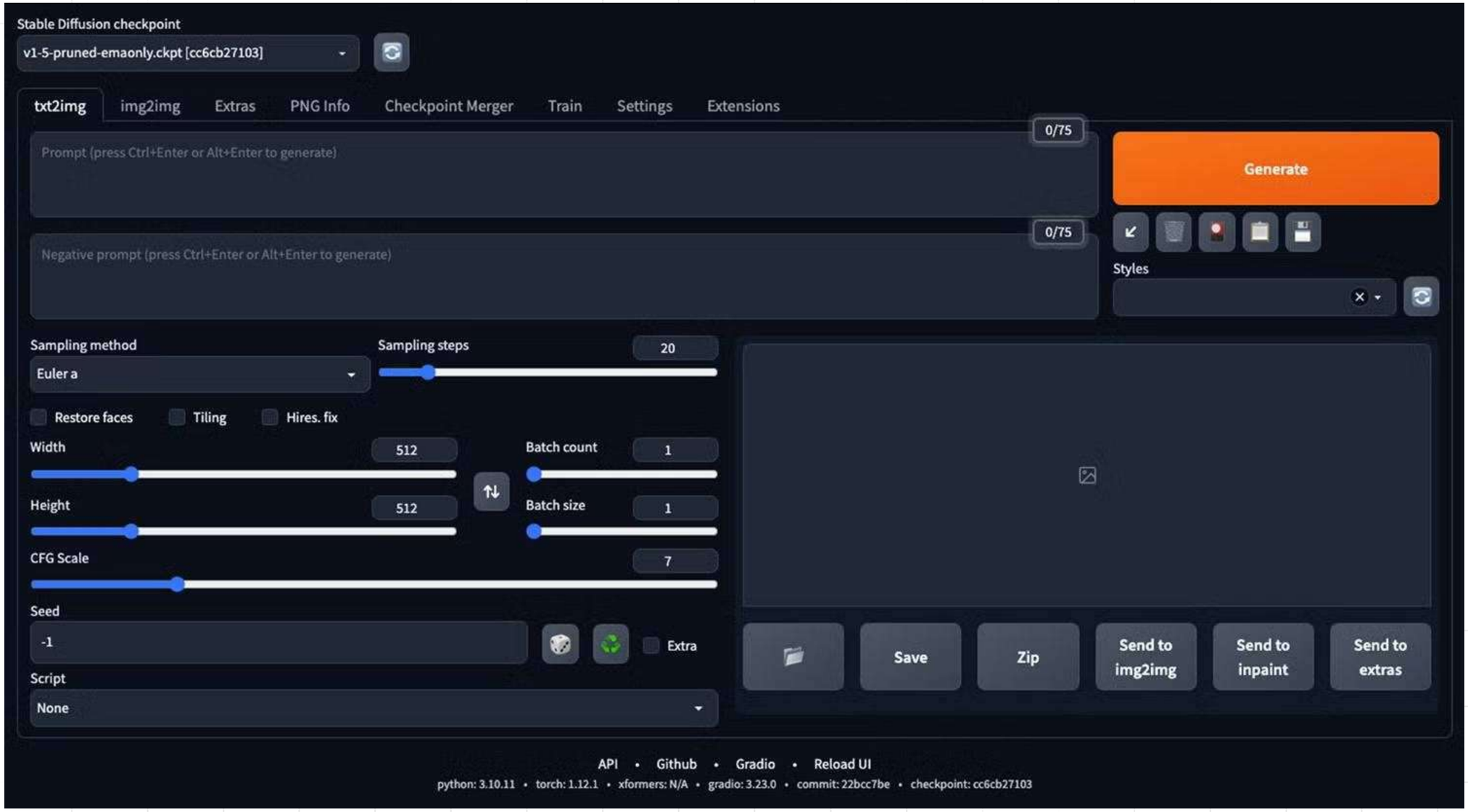
Estimated Costs

- Vercel Pro Plan: $20/month (includes hosting and CDN)
- Supabase Pro Plan: $25/month (database and storage)
- Replicate API: $0.001 per second of inference time (est. $500/month based on usage)
- Upstash Redis: $20/month for caching
- Cloudflare Workers: $5/month for additional serverless functions
- Total estimated monthly cost: $570 (excluding potential volume discounts)
- Note: None of these prices are accurate and are just estimations.

3.Pre-development Testing and Output Expectations

100x

Pre-development Testing and Output Expectations



Pre-development Testing

- Evaluate multiple AI models for each design category (logos, posters, social media posts, etc.)
- Benchmark generation times and quality across different styles
- Make tweaks for consistency and quality improvement

Output Expectations

- Quality: High-resolution, print-ready designs (300 DPI for graphics)
- File Formats: PNG, JPEG, SVG (for logos and vector graphics)
- Waiting Time:
  - Under 10 seconds for simple designs (e.g., logos, social media posts)
  - Up to 20 seconds for complex designs (e.g., detailed posters, multi-element compositions)
- Style Consistency: AI-generated designs should maintain consistent style within chosen templates

4.User Flow and Stages

100x

User Flow and Stages

1. User Authentication

- Sign up / Log in
- Choose subscription tier (if applicable)

2. Design Initiation

- Select design category (logo, poster, social media post, etc.)
- Choose style template

3. Design Customization

- Input customization prompt
- (Optional) Upload custom images or assets

4. AI Generation

- Submit design request
- View real-time generation progress

5. Review and Refine

- Preview generated design
- Make adjustments (colors, layout, text)
- Regenerate with modified prompt if needed

6. Finalization

- Apply final touches
- Preview in different formats/sizes

7. Export and Share

- Download in preferred format
- Direct share to social media platforms
- Save to user's design library

5. Execution Timeline and Team (Optional)

Execution Timeline and Team (Optional)

This step is for those who have already planned everything out and have enough conviction to spend money on it by putting together a team. This is where things like project management and resource allocation come in. It helps:

- Ensure the right mix of skills in the development team with the right development practices
- Identify potential bottlenecks in the development process

Timeline

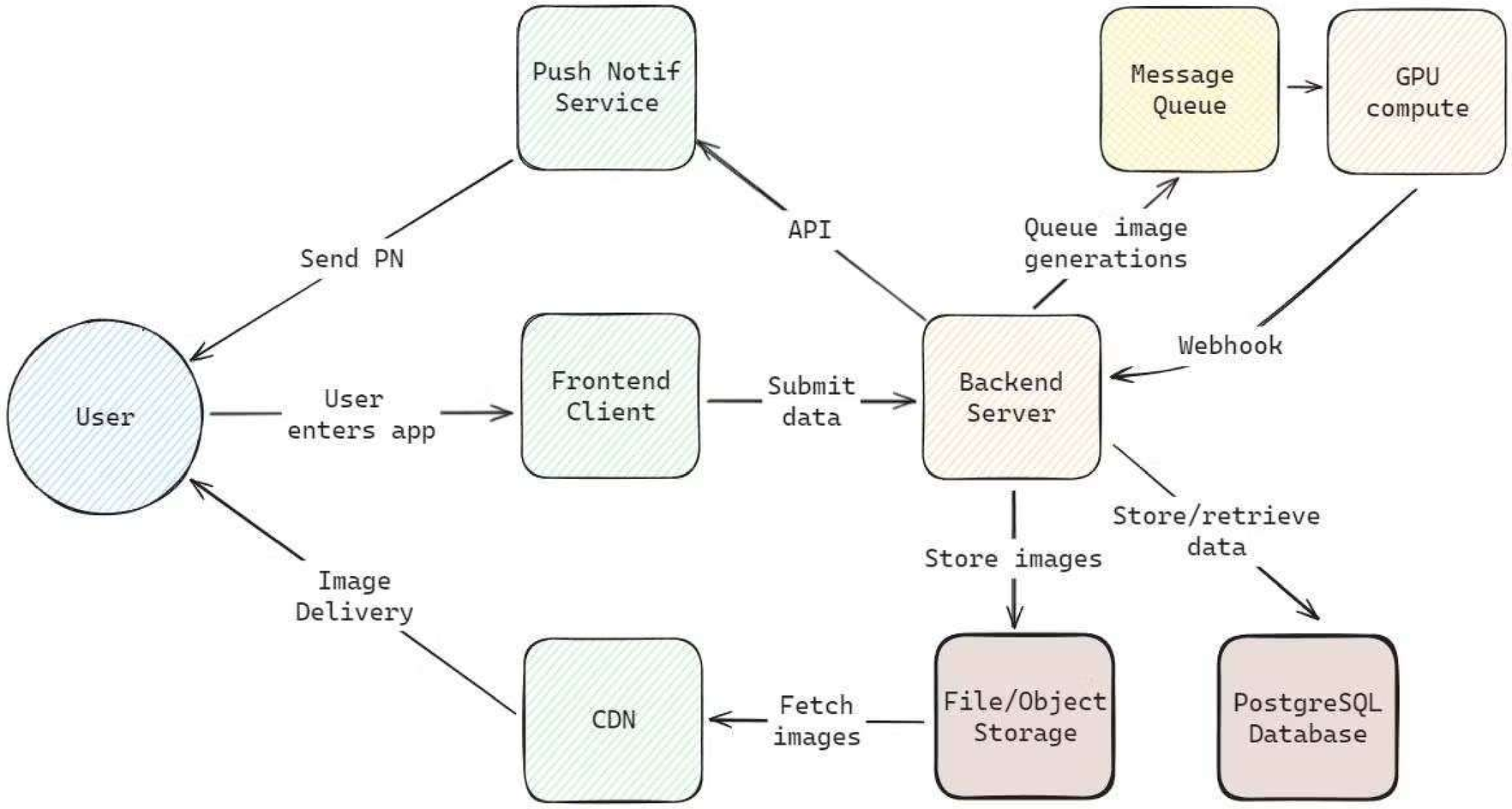
- MVP development: 6 weeks
- Testing and refinement: 2 weeks
- Full version launch: 10 weeks total

Team

- 2 Full-stack JavaScript/TypeScript developers
- 1 UI/UX designer (part-time, could be one of the developers with design skills)
- All team members should have basic understanding of AI concepts and prompt engineering

6. High Level Architecture & Data Modeling

High Level Architecture & Data Modeling



High Level Architecture & Data Modeling

Proper domain (and data) modeling is the foundation of efficient applications:

- It ensures data integrity and consistency
- Optimizes database performance and query efficiency
- Facilitates easier feature additions
- Sets up the foundation for product analytics

Entities/Tables

Users

id (uuid, primary key)

email (string, unique)

created_at (timestamp)

last_login (timestamp)

subscription_tier (string)

Designs

id (uuid, primary key)

user_id (uuid, foreign key to Users)

category (string)

prompt (text)

style_template (string)

created_at (timestamp)

updated_at (timestamp)

status (string: 'generating', 'completed', 'failed')

Assets

- id (uuid, primary key)
- design_id (uuid, foreign key to Designs)
- file_path (string, unique)
- file_type (string)
- created_at (timestamp)

Storage Buckets

1. user-uploads: For any user-uploaded images or assets
2. generated-designs: For storing AI-generated designs

Data Validations and Constraints

- Email format validation for user registration
- Prompt length limits (e.g., 5-500 characters)
- File size limits for user uploads (e.g., max 10MB)
- Enforce relationship integrity between Users, Designs, and Assets

7. Understanding Limitations & Scaling Strategies

Understanding Limitations & Scaling Strategies

Understanding limitations and planning for scale is important to prevent unexpected downtime or performance issues as user base grows.

Current Limitations

- Replicate API: Throughput depends on GPU availability and model size
- Supabase: 10,000 monthly active users on Pro plan
- Vercel: 6000 hours of serverless function execution per month
- Upstash: 256MB memory, 10,000 requests/day on Pro plan

Strategies


13
All violations - All o...
13
Synthetic Failures
Recent deployments
Re... Deplo... Description Time
1.5 tom Update README with endpoints 14:39, 25 Jun
1.4 matt Add instrumentation throughout app 14:39, 25 Jun
1.3 jon Add endpoint for users to manage account 14:39, 25 Jun
1.2 tomr Add ping endpoint 14:38, 25 Jun
Apdex score
0.4
0.2
0
00:00 03:00 06:00 09:00
WebTransactions
3s 0ms
2s 0ms
1s 0ms
0ms
Average response time
Standard deviation
12:00 18:00 00:00 06:00
Errors per minute
1.5
1
0.5
0
12:00 18:00 00:00 06:00
Datastore/all
100K
50K
0
Average value
Requests per minute
12:00 18:00 00:00 06:00
CPU User time
2m 30s
1m 40s
50s 0ms
0ms
Average time
Total time
11:15 11:20 11:25 11:30 11:35 11:40
Physical memory
300K MB
200K MB
100K MB
0MB
Total used mb
Used mb by host
12:00 18:00 00:00 06:00
CPU Time (Percent)
100%
50%
0%
11:15 11:30
Newrelic Dashboard
Powered by Geckoboard
8 days left in trial
11:45

Understanding Limitations & Scaling Strategies

The ease of making decisions when scaling applications is directly related how well we monitor our system. Integrating Error Reporting, Monitoring & Logging is crucial especially before you are ready for production.

Especially for complex systems, observability is a must have. Some examples are:

- Implement queuing system with Upstash for handling traffic spikes
- Implement caching strategies for frequently requested design templates and assets
- Set up Supabase read replicas for database scaling
- Use Vercel Edge Functions for low-latency, globally distributed computations
- Implement progressive loading and optimistic UI updates to improve perceived performance

8. Potential Third-Party Errors and Mitigations

Potential Third-Party Errors and Mitigations

Understanding limitations and planning for scale is important to prevent unexpected downtime or performance issues as user base grows.

Replicate API

- Error: Rate limiting or service unavailability
- Mitigation: Implement retry mechanism with exponential backoff, fallback to pre-generated templates

Supabase

- Error: Connection issues or query timeouts
- Mitigation: Implement connection pooling, use read replicas for heavy read operations

Vercel

- Error: Serverless function execution limit reached
- Mitigation: Optimize function execution time

Upstash Redis

- Error: Memory limit exceeded
- Mitigation: Implement LRU (Least Recently Used) cache eviction policy, prioritize caching for high-value operations

9. Security Considerations

Security Considerations

Security is paramount in protecting user data and keeping your system in safe. Some examples of things you can do :

- Protects against data breaches and unauthorized access
- Builds user confidence in the platform
- Implement HTTPS for all connections (default with Vercel)
- Use Supabase authentication with JWT tokens
- Implement rate limiting on API routes to prevent abuse
- Encrypt sensitive data at rest (using Supabase column encryption)
- Implement Content Security Policy (CSP) headers
- Use Cloudflare WAF (Web Application Firewall) for additional protection
- Secure all environment variables and API keys
- Implement proper CORS (Cross-Origin Resource Sharing) policies

10. Pre-Production Checks and Testing

Pre-Production Checks and Testing

Thorough testing before launch ensures a quality product:

- Identify and resolve issues before they reach users
- Ensure that all the components of your system are healthy and ready to handle scale
- Check for compatibility across different devices and browsers
- Confirms that all features work as intended

Pre-Production Checks and Testing

Checks:

- Comprehensive unit/integration testing for all components
- Load testing: Simulate 1000+ concurrent users
- Performance testing: Ensure <100ms server response time for non-AI operations
- Cross-browser testing: Chrome, Firefox, Safari, Edge (last 2 versions)
- Mobile responsiveness testing on iOS and Android devices
- API integration tests with all third-party services
- Error logging and monitoring setup (e.g., Sentry integration)
- Backup and disaster recovery procedure testing