Technical Documentation

Architecture Overview

Frontend Architecture

- Next.js application with React 19
- Pages and components structure following Next.js conventions
- State management using Zustand
- UI components from shadcn/ui library
- Drag and drop functionality using dnd-kit
- Chart visualizations using Chart.js

Backend Architecture

- Express.js server
- RESTful API endpoints
- OpenAI API integration

Code Organization

Frontend Structure

```
src/
├── app/              # Next.js app directory
├── components/       # Reusable React components
├── lib/             # Utility functions and helpers
├── hooks/           # Custom React hooks
├── types/           # TypeScript type definitions
└── styles/          # CSS and styling files
```

Backend Structure

```
src/
└── app/
    └── api/         # API routes and handlers
```

Development Guidelines

Code Style

- TypeScript for type safety
- ESLint for code linting
- Prettier for code formatting

Component Development

1. Create components in `src/components`
2. Follow atomic design principles
3. Implement proper TypeScript interfaces
4. Add JSDoc comments for complex functions

State Management

- Use Zustand for global state
- Keep component state local when possible
- Implement proper state hydration

API Integration

1. Create type-safe API clients
2. Handle errors appropriately
3. Implement proper request validation
4. Use environment variables for configuration

Deployment Process

Development Environment

1. Local setup
2. Development server setup
3. Environment variables

Third-party Integrations

OpenAI Integration

- API key management
- Rate limiting
- Error handling
- Prompt management

Development Workflow

Git Workflow

1. Branch naming conventions
2. Commit message format
3. Pull request process
4. Code review guidelines

CI/CD Pipeline

1. Build process
2. Test automation
3. Deployment automation
4. Quality gates

Troubleshooting Guide

Common Development Issues

1. Setup Problems

Common setup issues and their solutions:

a) Node Modules Installation

- Run `npm install` before starting development
- If you encounter dependency conflicts:

```
# Clear npm cache and node_modules
rm -rf node_modules
npm cache clean --force
npm install
```

- Ensure you're using the correct Node.js version (see .nvmrc)

b) Environment Variables

- Copy `.env.example` to `.env.local`
- Required environment variables:

```
OPENAI_API_KEY=your_api_key_here
```

- Common issues:
  - OpenAI API key not set or invalid
  - Missing required environment variables
  - Using production keys in development

2. Build Errors

a) TypeScript Errors

- Common type issues:
  - Missing type definitions for imported modules
  - Incorrect interface implementations
  - Fix by installing required @types packages:

```
npm install --save-dev @types/missing-package
```

- Run type checking: `npm run type-check`

b) Next.js Build Issues

- Clear `.next` directory if build is corrupted:

```
rm -rf .next
npm run build
```

- Common build failures:
  - Invalid API routes configuration
  - Missing dependencies
  - Environment variable issues
  - Invalid import paths

c) OpenAI Integration

- Verify API key permissions and quota
- Implement proper error handling for rate limits
- Test with smaller prompts first
- Monitor API usage and costs

Debugging Tips

- Check browser console for frontend errors
- Review server logs for backend issues
- Use Next.js debug mode: `NODE_OPTIONS='--inspect' npm run dev`

- Monitor API response times and error rates

References

Internal Documentation

- API Documentation
- README

External Resources

- Next.js Documentation
- Express.js Documentation
- OpenAI API Documentation