Example Project Overview

# Project Overview: Pokemon Trading Card Price Comparison Website

## Project Description

Develop a web application that provides price comparisons for Pokemon trading card products across various German online shops. The application will offer both free and premium features, implementing a freemium SaaS model with user accounts and subscriptions. The focus is on creating a Minimum Viable Product (MVP) that can be launched quickly while providing value to users and establishing a foundation for future enhancements.

## Key Features (MVP)

### Core Functionality

1. Automated product discovery and price tracking from specified German online shops (initially Pokeviert and Games Island)

2. Price comparison across tracked shops

3. User account system with basic dashboard for favorite products

4. Simple freemium model with essential premium features

5. Product search functionality

### Product Management

1. Automatic addition of new products found in tracked shops

2. Basic product categorization system

3. Admin section for manual review and management of products

4. Simple system for product matching and deduplication

### Data Collection and Display

1. Twice-daily price updates for all tracked products

2. Display of product details including image, description, and content

3. Direct links to product pages on source shops

4. Basic integration with Cardmarket API for price comparisons (premium feature)

### User Features

1. User accounts with simple customizable dashboards

2. Favorite product tracking

3. Basic price alerts for tracked products

4. Product search with filtering options

### Premium Features

1. Simple historical price charts for individual shops and average prices

2. Basic comparison with Cardmarket prices

3. Simple price evaluation markers (good/not good) based on market comparison

### Admin Features

1. Basic shop management system

2. Simple product review and categorization tools

## Technical Considerations

1. Web application development (desktop focus)

2. German language interface

3. Secure user authentication and data protection

4. Efficient web scraping and data management systems

5. Basic integration with Cardmarket API

6. Scalable architecture to support future growth

## Development Priorities

1. Focus on creating a functional MVP

2. Prioritize core features and basic premium functionality

3. Emphasize speed of development and deployment

4. Design for ease of future expansion and feature addition

## Target Audience

1. Pokemon trading card enthusiasts in Germany

2. Price-conscious collectors and resellers

3. Casual buyers looking for the best deals

## Success Metrics

1. User acquisition and retention rates

2. Conversion rate from free to premium users

3. User engagement (e.g., frequency of visits, time spent on site)

4. Accuracy and freshness of price data

This MVP project aims to quickly deliver a functional price comparison tool for Pokemon trading card products in the German market, with a clear path for future enhancements and expans

Example Project Plan

# Project Plan: Pokemon Trading Card Price Comparison Website

## Phase 1: Design and Architecture

1. Design database schema

2. Create system architecture diagram

3. Design API endpoints and data flow

4. Create UI/UX wireframes and mockups

5. Review and approve designs with stakeholders

## Phase 2: Development Setup

1. Set up development environments

2. Initialize version control repository

3. Set up CI/CD pipeline

4. Create initial project structure

## Phase 3: Core Backend Development

1. Implement database models and migrations

2. Develop core API endpoints

3. Implement user authentication and authorization

4. Develop web scraping functionality

5. Implement data processing and normalization

6. Integrate with Cardmarket API

7. Implement search functionality

## Phase 4: Frontend Development

1. Develop component library

2. Implement main pages (Home, Search, Product Detail, User Dashboard)

3. Integrate frontend with backend API

4. Implement state management

5. Develop admin interface

## Phase 5: Premium Features Development

1. Implement historical price charts

2. Develop price comparison with Cardmarket

3. Create price evaluation system

## Phase 6: Testing and Quality Assurance

1. Perform unit testing

2. Conduct integration testing

3. Perform user acceptance testin

Example Core Component Analysis

# Core Component Analysis: Pokemon Trading Card Price Comparison Website

## 1. Web Server

- Responsibility: Handle HTTP requests, serve static content, and route dynamic requests to the application server

- Technology: Nginx or Apache

- Key Considerations:

- Configure for optimal performance and caching

- Set up SSL/TLS for secure communications

## 2. Application Server

- Responsibility: Process dynamic content, implement business logic, and manage API endpoints

- Technology: Python with Flask or FastAPI

- Key Considerations:

- Implement RESTful API design

- Ensure efficient request handling and response generation

## 3. Database

- Responsibility: Store and manage all persistent data (products, prices, user information)

- Technology: PostgreSQL

- Key Considerations:

- Design efficient schema for price history and product relationships

- Implement proper indexing for fast query performance

## 4. Caching Layer

- Responsibility: Store frequently accessed data to reduce database load and improve response times

- Technology: Redis

- Key Considerations:

- Determine appropriate caching strategies for different data types

- Implement cache invalidation mechanisms

## 5. Search Engine

- Responsibility: Provide fast and relevant product search functionality

- Technology: Elasticsearch or PostgreSQL Full-Text Search

- Key Considerations:

- Optimize indexing for Pokemon card-specific attributes

- Implement efficient search algorithms and relevance scoring

## 6. User Authentication Service

- Responsibility: Manage user registration, login, and session handling

- Technology: Custom implementation with JWT

- Key Considerations:

- Ensure secure password hashing and storage

- Implement proper session management and token handling

## 7. Payment Service

- Responsibility: Handle premium subscription payments

- Technology: Integration with a payment gateway (e.g., Stripe)

- Key Considerations:

- Ensure PCI compliance for payment processing

- Implement proper error handling and transaction logging

## 8. Web Scraper

- Responsibility: Collect pricing data from target websites

- Technology: Python with BeautifulSoup or Scrapy

- Key Considerations:

- Implement robust error handling and retry mechanisms

- Ensure compliance with target websites' terms of service

## 9. Data Processor

- Responsibility: Clean, normalize, and structure scraped data

- Technology: Python with pandas or custom ETL scripts

- Key Considerations:

- Implement data validation and cleaning procedures

- Ensure efficient processing of large datasets

## 10. Cardmarket API Client

- Responsibility: Interface with Cardmarket API for additional pricing data

- Technology: Python with requests library

- Key Considerations:

- Implement proper rate limiting and error handling

- Ensure efficient use of API quota

## 11. Admin Interface

- Responsibility: Provide tools for system management and monitoring

- Technology: Django Admin or custom React-based admin panel

- Key Considerations:

- Implement role-based access control

- Provide intuitive interfaces for common administrative tasks

## 12. Monitoring & Logging System

- Responsibility: Track system performance, errors, and user activity

- Technology: ELK Stack (Elasticsearch, Logstash, Kibana) or Prometheus with Grafana

- Key Considerations:

- Set up comprehensive logging across all components

- Implement alerting for critical issues

## 13. Frontend Application

- Responsibility: Provide user interface for interacting with the price comparison service

- Technology: React or Vue.js

- Key Considerations:

- Implement responsive design for desktop-first approach

- Ensure efficient state management and API integration

Each of these components plays a crucial role in the overall functionality of the Pokemon Trading Card Price Comparison Website. The interactions between these components should be carefully designed to ensure optimal performance, scalability, and maintainability of the system.

Example Architecture Overview

graph TD

A[Client Browser] -->|HTTP/HTTPS| B[Load Balancer]

B --> C[Web Server]

C --> D[Application Server]

D --> E[Database]

D --> F[Caching Layer]

D --> G[Search Engine]

D --> H[User Authentication Service]

D --> I[Payment Service]

J[Admin Interface] --> D

K[Scheduled Jobs] --> L[Web Scraper]

L --> M[Data Processor]

M --> E

D --> N[Cardmarket API Client]

O[Monitoring & Logging] --> D

O --> C

O --> B

O --> L

Example Recommended Code Structure

3.31 KB • 94 extracted lines

Formatting may be inconsistent from source.

# Recommended Code Structure: Pokemon Trading Card Price Comparison Website

```

pokemon-price-comparison/

├── backend/

│ ├── app/

│ │ ├── api/

│ │ │ ├── \_\_init\_\_.py

│ │ │ ├── auth.py

│ │ │ ├── products.py

│ │ │ ├── prices.py

│ │ │ └── users.py

│ │ ├── models/

│ │ │ ├── \_\_init\_\_.py

│ │ │ ├── product.py

│ │ │ ├── price.py

│ │ │ └── user.py

│ │ ├── services/

│ │ │ ├── \_\_init\_\_.py

│ │ │ ├── scraper.py

│ │ │ ├── data\_processor.py

│ │ │ └── cardmarket\_client.py

│ │ ├── utils/

│ │ │ ├── \_\_init\_\_.py

│ │ │ ├── db.py

│ │ │ └── helpers.py

│ │ ├── \_\_init\_\_.py

│ │ └── main.py

│ ├── tests/

│ │ ├── \_\_init\_\_.py

│ │ ├── test\_api.py

│ │ ├── test\_models.py

│ │ └── test\_services.py

│ ├── alembic/

│ │ └── versions/

│ ├── .env

│ ├── requirements.txt

│ └── Dockerfile

├── frontend/

│ ├── public/

│ ├── src/

│ │ ├── components/

│ │ │ ├── Header.js

│ │ │ ├── Footer.js

│ │ │ ├── ProductList.js

│ │ │ └── PriceChart.js

│ │ ├── pages/

│ │ │ ├── Home.js

│ │ │ ├── Search.js

│ │ │ ├── ProductDetail.js

│ │ │ └── UserDashboard.js

│ │ ├── services/

│ │ │ ├── api.js

│ │ │ └── auth.js

│ │ ├── utils/

│ │ │ └── helpers.js

│ │ ├── App.js

│ │ └── index.js

│ ├── package.json

│ └── Dockerfile

├── nginx/

│ └── nginx.conf

├── docker-compose.yml

└── README.md

```

## Key Aspects of the Code Structure:

1. Separation of backend and frontend: Clear separation allows for independent development and deployment.

2. Modular backend structure:

- `api/`: Contains route handlers for different API endpoints.

- `models/`: Defines database models.

- `services/`: Houses business logic and external service integrations.

- `utils/`: Stores utility functions and helpers.

3. Frontend organization:

- `components/`: Reusable React components.

- `pages/`: Top-level components representing different routes.

- `services/`: API and authentication service integrations.

- `utils/`: Utility functions for the frontend.

4. Configuration and deployment:

- Use of `.env` for environment-specific configurations.

- Dockerfiles for both backend and frontend for containerized deployment.

- `docker-compose.yml` for defining and running multi-container applications.

5. Testing:

- Dedicated `tests/` directory in the backend for unit and integration tests.

6. Database migrations:

- Use of Alembic for managing database schema changes.

This structure promotes modularity, scalability, and maintainability of the codebase. It separates concerns effectively and provides a clear organization for different aspects of the application.

Example UI and UX Guide

3.30 KB • 89 extracted lines

Formatting may be inconsistent from source.

# UI and UX Guide: Pokemon Trading Card Price Comparison Website

## 1. Design Principles

- Simplicity: Keep the interface clean and uncluttered

- Efficiency: Prioritize quick access to price comparisons

- Consistency: Maintain uniform design elements throughout the application

- Responsiveness: Ensure usability across different desktop screen sizes

## 2. Color Palette

- Primary: #3D7DCA (Pokemon Blue)

- Secondary: #FFCB05 (Pokemon Yellow)

- Background: #F5F5F5 (Light Gray)

- Text: #333333 (Dark Gray)

- Accent: #FF5350 (Soft Red for alerts and important information)

## 3. Typography

- Main Font: Roboto

- Headings: Roboto Bold

- Body Text: Roboto Regular

- Font Sizes:

- H1: 28px

- H2: 24px

- H3: 20px

- Body: 16px

- Small Text: 14px

## 4. Layout

- Use a grid system for consistent spacing and alignment

- Implement a responsive design with breakpoints at 768px, 992px, and 1200px

- Maintain adequate white space to prevent cluttered appearance

## 5. Navigation

- Implement a clear, persistent top navigation bar

- Include search bar prominently in the header

- Use breadcrumbs for easy navigation on product detail pages

## 6. Components

- Buttons:

- Primary: Filled, rounded corners, Pokemon Blue

- Secondary: Outlined, rounded corners, Pokemon Yellow

- Cards: Use for displaying product information, with subtle shadows

- Forms: Clear labels, inline validation, and helpful error messages

- Modals: Use for login/signup and quick information display

## 7. Product Listings

- Display products in a grid layout

- Show key information: Image, name, current lowest price, and price trend indicator

- Use consistent image sizes and aspect ratios

## 8. Price Comparison

- Use clear, easy-to-read tables for price comparisons

- Highlight the lowest price

- Use icons or colors to indicate price trends

## 9. Search Functionality

- Implement auto-suggest for search queries

- Provide clear filters and sorting options

- Show number of results and current applied filters

## 10. User Dashboard

- Use a clean, card-based layout for displaying user's tracked products

- Implement easy-to-read charts for price history

- Provide clear calls-to-action for setting up alerts and accessing premium features

## 11. Accessibility

- Ensure sufficient color contrast for readability

- Provide alternative text for images

- Make sure the site is navigable via keyboard

- Use ARIA labels where appropriate

## 12. Loading States

- Implement skeleton screens for content loading

- Use subtle animations for loading indicators

## 13. Error Handling

- Display clear, friendly error messages

- Provide helpful suggestions for resolving errors or alternate actions

## 14. Responsive Design Considerations

- Use a fluid grid that adapts to different screen sizes

- Adjust font sizes and spacing for smaller screens

- Optimize touch targets for better usability on tablets

## 15. Feedback and Interactions

- Implement subtle hover effects on interactive elements

- Provide visual feedback for user actions (e.g., button press states)

- Use micro-animations to enhance the feel of interactivity

By following these UI and UX guidelines, we aim to create a cohesive, user-friendly interface for the Pokemon Trading Card Price Comparison Website. This guide should be referred to throughout the development process to ensure consistency and quality in the user experience.