Questions and Answers

1. How would you design a notification system that alerts users when one of their listings is outside of a certain user-defined threshold (say +/- 10%) of the average market price?

To design a notification system that alerts users when one of their listings is outside of a certain user-defined threshold, we would need to periodically retrieve pricing information from various marketplaces and calculate the average market price for each trading card product.

We can then compare the user's asking price against the average market price and alert them if their asking price is outside of the user-defined threshold (e.g. +/- 10%).

This can be achieved by setting up a background job or cron job that runs at regular intervals and checks the prices.

2. How do you maintain an accurate and up-to-date representation of the market?

To maintain an accurate and up-to-date representation of the market, we could use a combination of web scraping techniques and API integrations with various marketplaces.

3. What technologies would you use to store these different types of data?

We would use a NoSQL database, such as MongoDB, to store the data.

The data would be stored in a NoSQL database, such as MongoDB.

This would allow us to easily store and retrieve large amounts of unstructured data.

We can also set up a cache layer to improve query performance and reduce the load on the database.

4. What are their advantages and disadvantages of MongoDB?

Advantages:

- Flexible schema: Since the structure of the data can vary between documents in a collection, MongoDB allows users to make changes to their schema as needed without needing to modify the entire database.

- Scalability: MongoDB is designed to scale horizontally across multiple servers and clusters, making it easier to handle large volumes of data.

- High availability: MongoDB has built-in replication and sharding capabilities that ensure high availability and reliability.

- Performance: With its ability to store related data in a single document, MongoDB can access and process data more quickly than traditional relational databases.

Disadvantage:

- Lack of transaction support: Unlike traditional relational databases, NoSQL databases do not support transactions. This means that users will need to implement their own mechanisms to ensure data consistency.

- Limited query capabilities: While NoSQL databases are efficient at handling simple queries, they may not be as effective at handling complex queries.

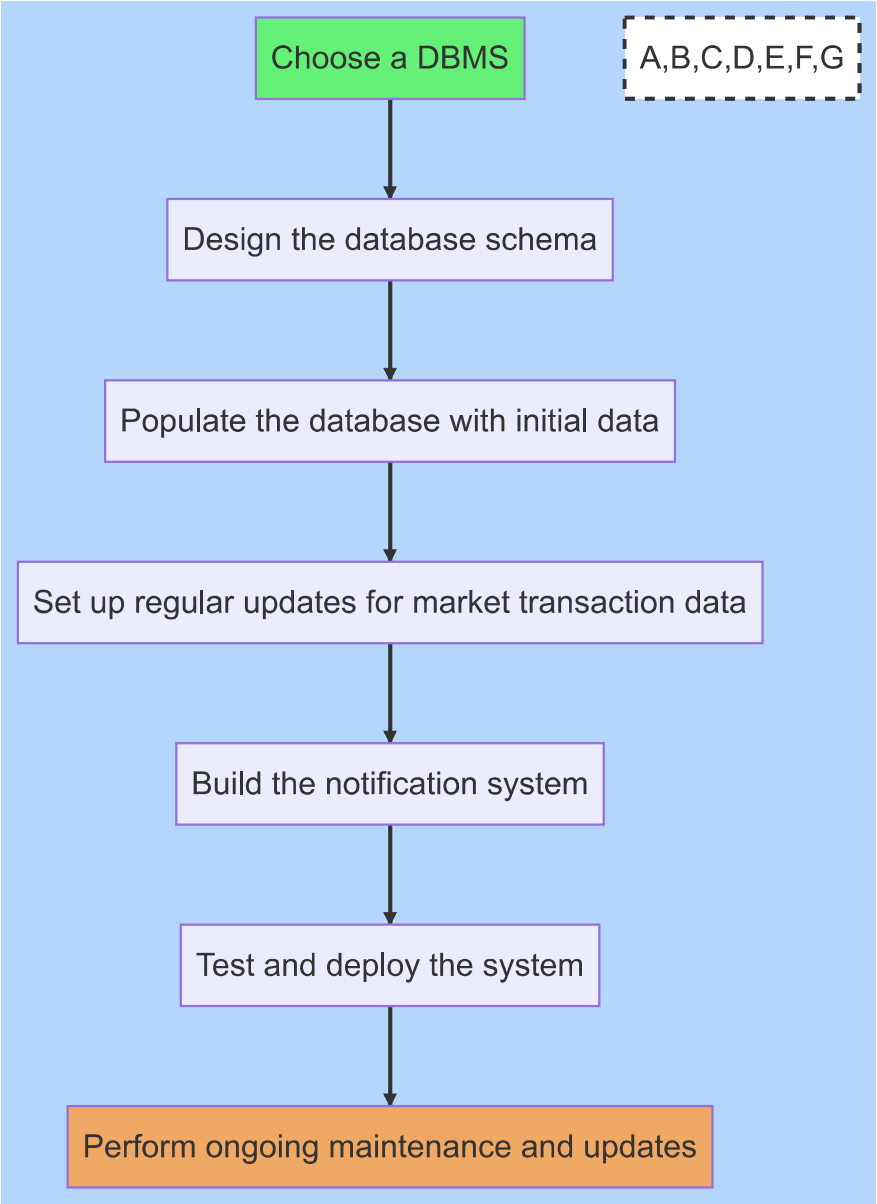
- Learning curve: Using NoSQL databases requires learning a new set of concepts and tools, which can be challenging for developers who are accustomed to working with traditional relational databases.

TRADING CARD SYSTEM

▲ Summary

> Building a trading card system involves choosing a DBMS, designing the database schema, populating the database with initial data, setting up regular updates for market transaction data, building the notification system, testing and deploying the system, and performing ongoing maintenance and updates. By following this process, we can create a scalable and reliable system that meets the needs of users and collectors.

Diagram



▲ Detailed Steps

A detailed explanation of the implementation process for the trading card system.

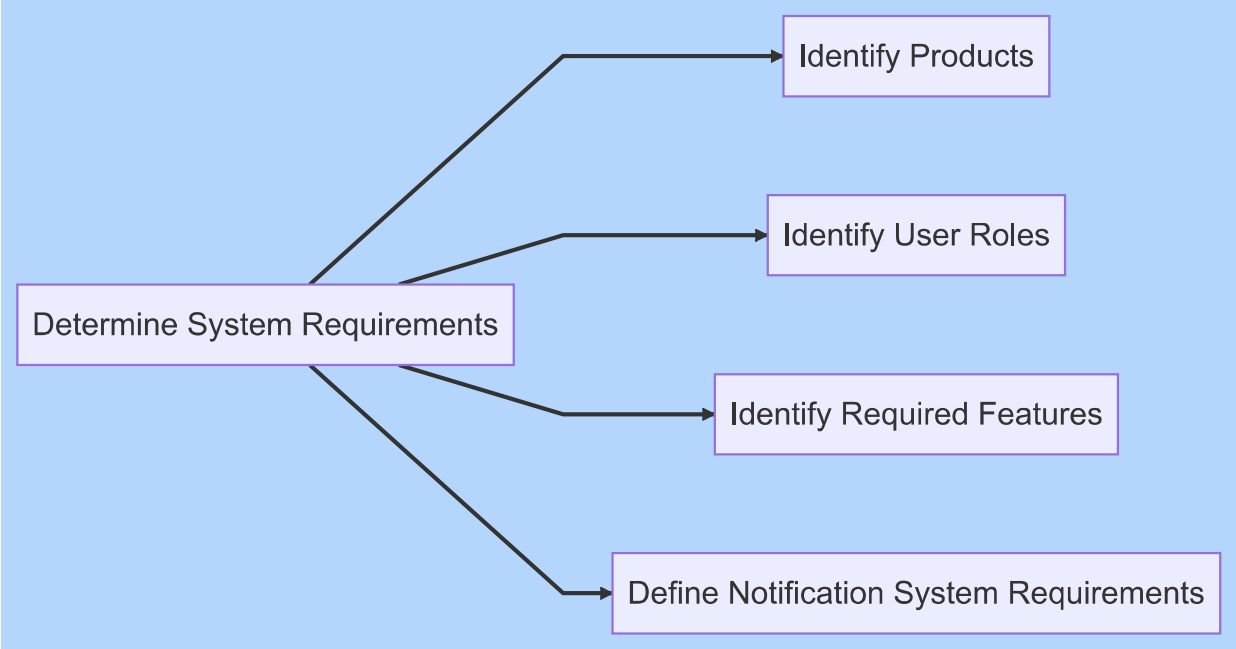
1. Determining System Requirements The first step in building any system is to determine its requirements and use cases. For the trading card system, we need to identify the following:

- The types of products that will be traded (baseball cards, basketball cards, etc.)

- The user roles (buyers, sellers, collectors)

- The features required for each role (listing cards for sale, browsing listings, buying and selling cards, tracking holdings)

- The notification system requirements (how notifications are sent, threshold percentage for notifications)



2. Choosing a Database Management System

Based on the requirements of the system, a NoSQL database would be a suitable choice as it can handle unstructured data and provide high scalability.

3. Designing the Database Schema

To represent the various entities in the trading card pricing system, we can design a schema in MongoDB with separate collections for users, holdings, products, and transactions.

Simple schema could look something like the following:

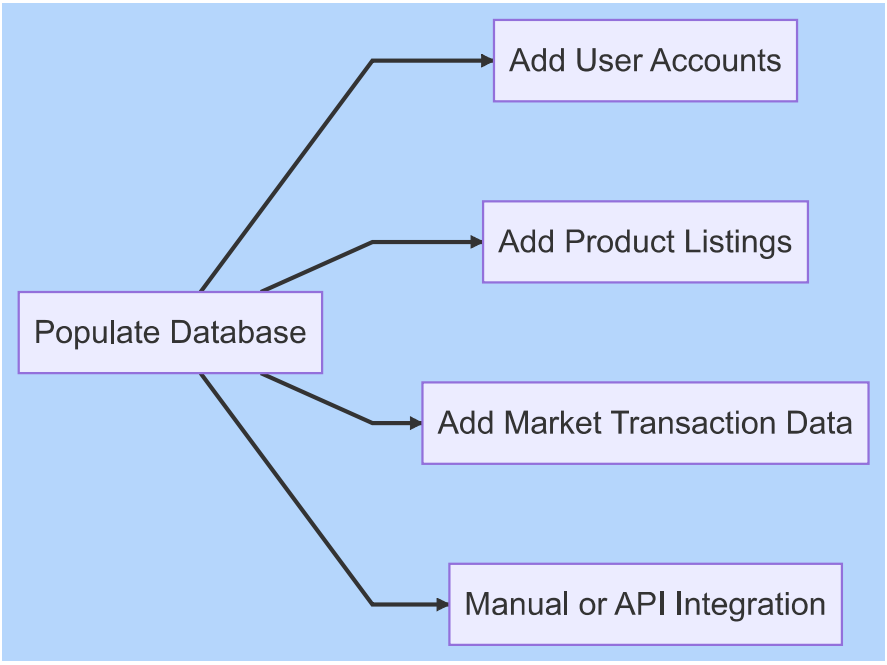
```
Users Collection:
{
  _id: ObjectId,
  username: string,
  email: string,
  password: string,
  holdings: [ObjectId],
  notifications: [Notification]
}

Holdings Collection:
{
  _id: ObjectId,
  user_id: ObjectId,
  product_id: ObjectId,
  condition: string,
  quantity: int,
  asking_price: float,
  listed_marketplaces: [string],
  last_updated: datetime
}

Products Collection:
{
  _id: ObjectId,
  name: string,
  description: string,
  image_url: string,
  avg_market_price: float,
  last_updated: datetime
}

Transactions Collection:
{
  _id: ObjectId,
  product_id: ObjectId,
  sale_date: datetime,
  sale_price: float,
  market_source: string
}
```

4. Populating the Database

Before we can start using the trading card system, we need to populate the database with initial data.

This includes adding user accounts, product listings, and market transaction data.

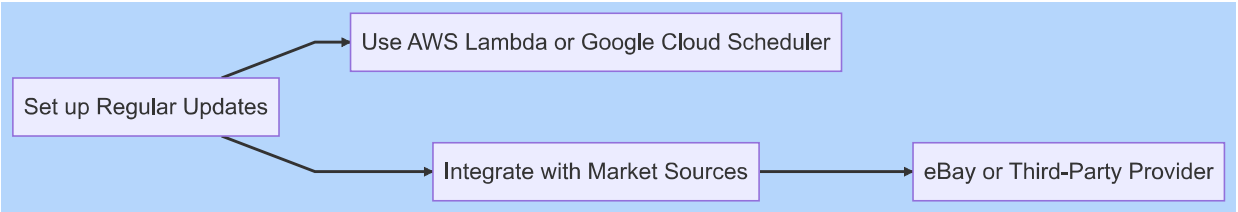
We can do this manually or through an API integration with market sources like eBay.



5. Set up Regular Updates for Market Transaction Data

To ensure that market transaction data is up-to-date, we need to set up regular updates using tools like AWS Lambda or Google Cloud Scheduler.

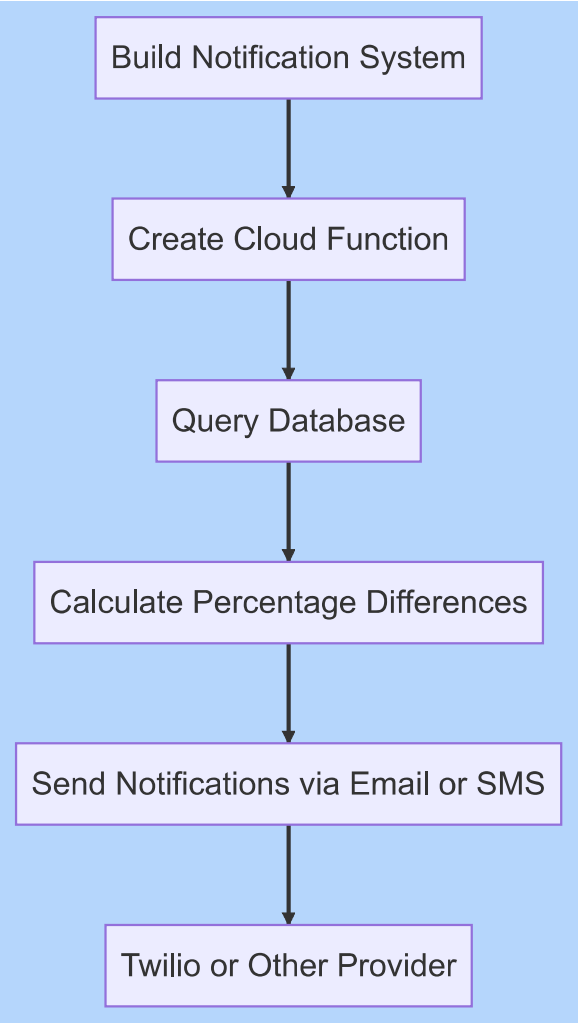
We can integrate with market sources like eBay or use a third-party provider that collects market data.



6. Building the Notification System

To build the notification system, we need to create a cloud function or serverless architecture that queries the database, calculates percentage differences between asking prices and average market prices, and sends notifications via email or SMS if the threshold is exceeded.

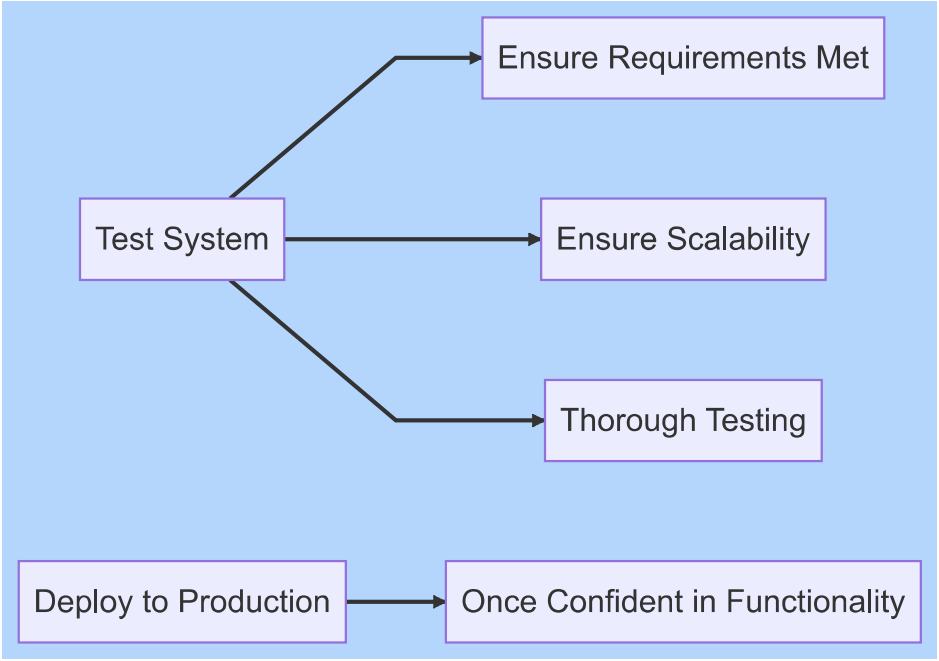
We can use AWS Lambda or Google Cloud Functions to create these functions and integrate with email or SMS providers like Twilio.



7. Testing and Deployment

After building the trading card system, we need to test it thoroughly to ensure that it meets the requirements and is scalable.

Once we are confident in the functionality of the system, we can deploy it to production.



8. Ongoing Maintenance and Updates

To ensure that the trading card system continues to function properly, we need to perform ongoing maintenance and updates.

This includes monitoring the system for errors or performance issues, updating the database schema as needed, and integrating new features based on user feedback.