Table of Contents

Does your service have any constraints or edge cases?
Data Schema
Architecture diagram
The story of how your service converts the inputs to the outputs
Tools and technologies I plan to use

App plan for BACKAZON

If you'd like to book some time for your group to meet with your tech mentor this week to review the current state of your project plan, please sign up for an open slot. A meeting is NOT REQUIRED, but completion of the project plan is required.

Which app did you pick as a target for building a minimal, data-oriented clone?

Amazon

What is the business question you aim to answer?

On a weekly basis, which recommendation method (content based filtering or collaborative filtering) generates more sales?

You must include a sample diagram/graph of what your final result will look like. Use Google Sheets and fabricated data to make a graph. The graph should be presented as a time-series, since the business question incorporates time. Be sure you fully understand what the numerator and denominator values in the ratio represent and have a plan for how to calculate them based on information that you will gather and/or generate throughout your app's services.

x - axis: time

y - axis : total of sales

plot points:

# of content sales/total sales,

# of collaborative sales/total sales,

# of regular listing sales/total sales,

# of other sales/total sales

What are your major system inputs?

Describe the user data or physical-world data that is flowing into your application. List all the things you need to know (both directly and indirectly) to answer the business question.

1. Listings / inventory of products (current, new)
2. User Activity (viewed items, searches, wishlists, purchases, etc.)
3. User Orders

User Analytics Service

1. Anytime a ‘user’ emits an action of some sort, it gets stored into a user analytics database paired with specific user. This will be used later by other services to calculate their outputs.

Inventory & Search Service

1. Returns trending items when user opens Backazon homepage.
2. Returns relevant listings of client search/category queries.
3. Handles client side requests for posting new listings.
4. Provides Filtering Service trending items upon request, source of truth of inventory.
5. Receives order notifications from Orders Service, updates inventory quantity.

6. Records user search events to User Analytics Service.

Filtering (Collaborative & Content-Based) Service

1. Query to user analytics for data.
2. Query to inventory api for data
3. Combine these in some way to return a filtered recommended list
4. return filtered list.
    - ID
    - ListingID
    - User Analytics
    - Client side requests

Orders

1.User purchases an item and makes a request to the orders service with unique session ID and item ID

> Assumption that transition is successful when requesting order service

> Eliminates the need to worry about ACID compliance and rollback failures

2. Orders service writes order information to the database
3. Sends the order information along to the queue in which USER analytics pull from
4. Calculates the metrics for revenue analysis weekly to report metrics on filtering effectivity

> content filtering vs collaborative filtering vs normal purchase

5. Creates visualization graphic for further analysis report

What are your major system outputs?

Describe the user data or physical-world data that is flowing out of your application. Note: this section is **not** the answer to your business question but rather the data produced in the process of answering the business question.

- Trending & Search Results
- Collaborative & Content-based Filtered Items
- New item confirmations
- Order confirmations
- Revenue data for analysis

Architecture diagram

The story of how your system converts the major inputs to the major outputs

Annotate the architecture diagram with entry and exit points and tell one or more step-by-step stories of how data moves through your system. What are the steps you need? What conversions occur? Create a sequence diagram if helpful.

Service Plan for Inventory & Search Service

Service Owner: Kayleigh Foley

What are your service's inputs and outputs (API Spec)?

Define all the API endpoints that will be used to consume and publish data. If you are using a message bus to send and receive data, what messages do you expect to receive or transmit? What is the shape of the data (i.e. list all the properties and types) for each API endpoint and/or message?

SEARCH	**Endpoint**	**Service**	**Data**	**Response**	**Description**
Inbound Requests	GET / queries	*Client*	keyword(s) - strings	Relevant items (summarized)	User searches via search bar
	GET / categories	*Client*	Category id	Relevant items (summarized)	User searches via categories
Outbound Requests	POST /	Analytics	User activity object	-	Record user activity (search queries) to analytics

INVENTORY	**Endpoint**	**Service**	**Data**	**Response**	**Description**
Inbound Requests	GET / trending	*Client*	-	Trending items (summarized)	User visits Backazon homepage and sees trending items
	GET / details	*Client*	Item id	Item details	User selects item to view details
	POST / newitem ***to queue	*Client*	New listing object	Success/confirmation	User submits new item to inventory
	GET / from newitem queue				
	POST / sale	Orders	Sale information object	Success	Recent sales received to update inventory quantities
	POST / trending	Filter	New listing object	Trending items (summarized)	Inventory updates Filter with recently added items
Outbound Requests	POST /	Analytics	User activity object	-	Record user activity (search queries) to analytics

Does your service have any constraints or edge cases?

Which operations need to happen in real-time? Which operations can be processed on schedule to arrive at eventual system consistency? Are there any other edge cases or constraints?

Real-time/Immediate:

Returning trending summarized item information in response to homepage request
Returning relevant summarized item information in response to search/category queries
Returning item detail information in response to item viewing request
Returning trending items to Filter Service upon request
Sending confirmation for new listing items

Eventual Consistency/Non-immediate:

Sending user activity (search queries / item views) to analytics
Updating inventory with new items or quantity changes

Inventory - mongoDB

Primary needs: fast reads, slow updates

- client visits homepage → send trending items (fast)
- client clicks on item → send full item details (query by itemID) (fast)
- client posts new item to queue → get item from queue, write to db (slow)
- Orders sends inventory update → update inventory of item (slow)
- Update trending items to cache for Filter (slow)
- Client submits new item → send confirmation (fast), update inventory (slow)

Schema: all product details

Search - elasticsearch

Primary needs: fast search results

- client enters search query → send summarized matches (fast)
- client clicks on category/dept → send summarized matches (fast)
- Inventory updates Search (cache?) with new/trending items (slow)
- Search updates Analytics with client search queries (slow)

Schema: id (matches inventory id), name, price, photoUrl, description, category, subcat, dept, reviews, rating, brand

Cache(s)

- Inventory: recently viewed items (full details) & trending items of the day
- Search: recent searches (summarized details)

Data Schema

How will your data be stored? What DBMS do you plan to use and why? If you are using a SQL database, what is the schema for this data (create an ER diagram)? It is useful to think about the organization of your data in a DMBS even if you are using a noSQL datastore. In that case describe the shape of the data for all collections you plan to use.

Inventory Database - *MongoDB*

The primary queries of the inventory database will be retrieving a single item's full product details, updating inventory quantity after sale, and retrieving the trending items in each particular category. An item ID will be used to retrieve a particular item as fast as possible, an item ID and sold quantity will be used for updating inventory, and a query for the top items in each category will be run once daily. Other uses will include adding new items to the database as they are submitted by users.

After determining these primary uses I chose to use a non-relational database as it will provide choice for efficient querying and rapid storage of millions of items. The use case demanding the fastest data return is the client requesting full product details, and by using a document based database I will be able to query for all details pertaining to one item without the need for joining relational tables.

Search Database - ElasticSearch

The elasticsearch database will contain a summarized version of the product item details contained in the inventory database and be used to return rapid search results to the user. The primary searches will be for related/matching items to a user's search box query or a user clicking to view specific category or department products.

Architecture Diagram

Draw an architecture diagram for your service.

The story of how your service converts the inputs to the outputs:

Annotate the architecture diagram with entry and exit points and tell a step-by-step story of how data moves through your service. What are the steps you need? What conversions occur? Use a sequence diagram if helpful.

See Architecture Diagram above

Tools and technologies I plan to use:

Express

MongoDB - *Inventory*
ElasticSearch - *Search*
Redis - *Caching*
SQS - *Inventory - new items*

Service Plan for Collaborative Filtering Service

Service Owner: Austin Shin

What are your service's inputs and outputs (API Spec)?

Define all the API endpoints that will be used to consume and publish data. If you are using a message bus to send and receive data, what messages do you expect to receive or transmit? What is the shape of the data (i.e. list all the properties and types) for each API endpoint and/or message?

Collab	Endpoint	Service	Data	Response	Description
Inbound Requests	GET / collaborativeRecommendedList	*Client*	none	collaborative recommended array list of products	User loads a page looking for a recommended list based off of other users.
	GET / contentRecommendedList	*Client*	none	content recommended array list of products	User loads a page looking for a recommended list based off of previous browsing history.
Outbound Requests	POST / TODO	Analytics	User activity object	-	Service requires user analytics data to compute recommendations
	POST / TODO	Inventory	Top trending products object	-	Service requires top trending products data to compute recommendations

Does your service have any constraints or edge cases?

Which operations need to happen in real-time? Which operations can be processed on schedule to arrive at eventual system consistency? Are there any other edge cases or constraints?

The collaborative and content filtering service has no calculating-operations that need to occur in real-time. The only real-time operations that will occur that are non-calculating is when the client sends a get request for a recommended list. The service will check the cache to see if it such data exists and return it.

The recalculation of recommended lists will happen at a set interval, most likely overnight. This recalculation will involve all ***n*** users, ***j*** hottest trending products (a subset of the products), and

k events. After the relationships are updated/created, neo4j's recommendation engine will output recommended lists for each user which will be stored inside of a cache (redis).

Data Schema

How will your data be stored? What DBMS do you plan to use and why? If you are using a SQL database, what is the schema for this data (create an ER diagram)? It is useful to think about the organization of your data in a DMBS even if you are using a noSQL datastore. In that case describe the shape of the data for all collections you plan to use.

Neo4J will hold a user id, product id and relevant information, and create relationships between the two with events.

Redis will be a cache having a key-user id and value-array of product ids i.e. {user id: [productId]}. There will be two separate caches. One for content-filtered recommended list and commended list.

NEO4

Architecture diagram

Draw an architecture diagram for your service.

The story of how your service converts the inputs to the outputs

Annotate the architecture diagram with entry and exit points and tell a step-by-step story of how data moves through your service. What are the steps you need? What conversions occur? Use a sequence diagram if helpful.

See above diagram

Tools and technologies I plan to use

Express - Basic server framework

(Maybe refactor to Koa when I get a chance)

Neo4J - a graph database for quick querying as well as establishing relationships between data. It also provides a recommendation engine.
(refactor to PostgreSQL in the event I implement my own algorithm)
Redis - for caching data
Faker - for generating large amounts of fake data.
--- unsure what the bottom three are used for, but as I do more research I'll figure it out ---
EC2 (t2.micro instances) - as needed
Amazon SQS - as needed
New Relic - dashboard
Docker

Service Plan for Orders Service

Service Owner: Chase Lee

What are your service's inputs and outputs (API Spec)?

Define all the API endpoints that will be used to consume and publish data. If you are using a message bus to send and receive data, what messages do you expect to receive or transmit? What is the shape of the data (i.e. list all the properties and types) for each API endpoint and/or message?

Define all the API endpoints that will be used to consume and publish data.

***** NEED TO ESTABLISH ROUTE NAME TO QUEUES

ORDERS	**Endpoint**	**Service**	**Data**	**Response**	**Description**

Inbound Requests	POST /placeOrder	*Client*	ORDER INFO
			{
			orderNum: 1,
			userId: 123,
			Item: [itemId: {qty: 2, rating: 4}....],
			date: Fri Jan 26 2018 16:37:59 GMT-0800 (PST),
			purchaseMethod: collaborative,
			totalPrice: 1234
			}
				Success / Confirmation that order was placed
				201
				Order Id
				(json obj)
					User purchases Item and sale is archived in orders database.
					
					For the scope of the project, assuming that the order was already successfully processed with customer payment method. Otherwise would need to structure service for ACID compliance and failure rollback.

Outbound Requests	POST / to new item queue*	Inventory	What item was purchased & quantity of items purchased (json object)	-	Updates the inventory service on how many items and what items were purchased for consistent quantity data throughout the system
	POST / to user analytics queue*	User Analytics	Order Information (json object)	-	Updates user analytics to update user activity profile for further analysis

Does your service have any constraints or edge cases?

Which operations need to happen in real-time? Which operations can be processed on schedule to arrive at eventual system consistency? Are there any other edge cases or constraints?

Real-time/Immediate

Feedback to the client to show that order services received the order

Eventual System Consistency

Processing the order and notifying the user
Sending user order data to user analytics
Updating inventory on qty of item purchased
Weekly revenue analysis reports at low traffic time

Edge Cases

I plan to have a middle message queue between my service and the client, so that my services can have optimal horizontal scaling to maximize throughput. Because of the queue model and how orders need to be processed in a FIFO manner, I have to engineer a solution to ensure the orders coming off the message queue are being processed in sequential order. I also have to devise a system, in event of service worker crashing while processing order, that the order gets put back on the queue ASAP and gets prioritized to the front of the queue.

Data Schema

Draw an architecture diagram for your service. use.

Queries

Query items by UserID (PK) sort by Date (CK)
Query sum(totalPrice) by purchaseMethod (PK) sort by Date (CK) (Materialized View)

Created the materialize view to have ability to query sum revenue based on purchase method partition key.

How will your data be stored? What DBMS do you plan to use and why?

For the systems design capstone my service is operating under the assumption that whenever an order is placed it was successfully processed and approved through the transaction service. For that reason, I will not necessarily have to worry about ACID compliance for transactions and construct mechanisms to handle failure rollbacks.

The data I am keeping track of will eventually be a catalogue of all the orders, and I am only interested in the method of purchase (filtering, collab, normal) and the total price of the sale. I am interested in those two primary categories and I do not have to worry about ACID compliance, I was drawn to choose a noSQL type database for extremely fast writes. Because my analysis of the orders will be performed weekly at low traffic hours, fast read speeds are a trade off for high write throughput.

The reason I chose Cassandra database over MongoDB was for high availability, write scalability, and durability. In terms of high availability, Cassandra was the right choice because of the multiple master node model. If a node were to crash or go down, it does not affect the ability of the cluster to take writes so there will be a 100% uptime and orders will always be able to be taken in and recorded. When the orders service is down or the database is not able to read that is lost revenue for a business. Cassandra's write scalability was also superior to MongoDb's due to the masterless model. MongoDB can only write to one master and the other secondary servers are used for reads. When I need to scale I will be able to add more servers in the cluster for better performance. The last factor that lead me to choose Cassandra over MongoDB was the durability factor. The orders service is important to have a durable database because if the server crashes and the service loses the data for that order, it reflects poorly on the business. Writes in Cassandra are durable and writes to a replica node are recorded both in memory and in a commit log on a disk before they are regarded as a success. If a crash or server failure were to a occur before the memtables are flushed to disk, the commit log is replayed to restart to recover any lost writes. In addition to local durability, the masterless node model makes it so replication of data on other nodes strengthens durability.

Describe the shape of the data for all collections you plan to use.

TABLE ORDERS
NEW VERSION

```
{
   orderNum: 98,
   purchaseMethod: 'content',
   date: 2017-03-22T03:16:43.541Z,
   timestamp: 1517360768317,
   userId: 58006,
```

```
    Itemid: 123,
    Qty: 1,
    Rating: 5,
    totalPrice: 80
}

OLD VERSION
{
    orderNum: 98,
    purchaseMethod: 'content',
    date: 2017-03-22T03:16:43.541Z,
    timestamp: 1517360768317,
    userId: 58006,
    items:
     { '2180': { qty: 2, rating: 4 },
       '4217': { qty: 1, rating: 5 },
       '5517': { qty: 4, rating: 3 },
       '5610': { qty: 2, rating: 5 } },
    totalPrice: 80
}
```

Architecture diagram

The story of how your service converts the inputs to the outputs

Annotate the architecture diagram with entry and exit points and tell a step-by-step story of how data moves through your service. What are the steps you need? What conversions occur? Use a sequence diagram if helpful.

I will be polling a queue that responds directly with the client. My orders service will take in an orders JSON object and write that object to the database. On a routinely basis (maybe every 12 hours?) I will send what items were purchased to the inventory service to update the stock as well as what items users purchased to user analytics. Because the data I am sending out does not need to be processed immediately I will be reporting to the inventory and user analytics message bus queues. Based on my schema I will only be pulling and sending the items key/values to inventory in a JSON object and items+userID to user analytics queue in a JSON object.

Tools and technologies I plan to use

Cassandra

Sqs queue

Express

Docker

Something for data viz and analytics?

Service Plan for User Analytics

Service Owner: Ben Basuni

What are your service's inputs and outputs (API Spec)?

USER ANALYTICS	Endpoint	Service	Data	Response	Description
Inbound Requests	GET /queue/analytics	*Poll Event Queue*	JSON	No Response	Event clicks either to purchase, favorite, search, or click on item
Outbound Requests	POST /queue/filtering/ :start/:end	Filtering	JSON	No Response	Write messages to filtering queue to be used for recommendation rendering

What messages do you expect to receive or transmit?

My inbound request will heavily depend on how the Client, the Orders and Listings microservices send their data. These 3 services will be posting every event onto the queue. My job is to poll data from the queue and process it into a report.

My outbound request will provide the filtering service the list of all the events that has happened with user_id, product_id, and event as properties, weekly. This will be served onto the filtering service through a queue.

What is the shape of the data (i.e. list all the properties and types) for each API endpoint and/or message?

Data from ORDERS

Data from LISTINGS

Data from CLIENT

Does your service have any constraints or edge cases?

Which operations need to happen in real-time?

When an event is triggered either by the client, the orders microservice, or the listings microservice, it should write onto the analytics queue in real time.

When my microservice polls data from the analytics queue, the write to the database needs to happen in real time.

Other than that, everything else is time sensitive. The sending to the filtering queue and the polling from the filtering service onto the queue does NOT have to happen in real time.

Which operations can be processed on schedule to arrive at eventual system consistency?

Both inputs and outputs of the analytics microservice communicates with a job queue. The trade-off of having a job queue is that it can help filter and accept POST requests from different services into a general platform.

Because the user analytics microservice is not time-sensitive, it allows multitudes of clicks, favorites, searches, purchased events to be saved by its' timestamp. It also avoids idling computer resources and can be used in a set amount of time without having to have minute by minute human supervision.

Are there any other edge cases or constraints?

The only edge case is if the server shuts down, there was a possibility of loss of data. But because all of the writes happen in real time, everything should be preserved onto the job queue or the database. Other than that, there are no edge cases, I am accepting clicks, favorites, and purchased with the Listing ID, Timestamp and User ID of each event.

There are no constraints aside from the general requirements of the schema (User ID, Listing ID, type of Action, & Timestamp)

Data Schema

How will your data be stored?

The data will be stored in a Cassandra Database.

What DBMS do you plan to use and why?

I am using Cassandra as my database because I am going to write more data than I am going to read. Cassandra's read time is fairly quick and I believe that by structuring my tables based on my queries, I can get super optimal write and read speeds.

With Cassandra's partition and cluster keys, it has optimized my reads and writes for me out of the bag.

Architecture diagram

The story of how your service converts the inputs to the outputs

Annotate the architecture diagram with entry and exit points and tell a step-by-step story of how data moves through your service.

Step 1. Recording all User Inputs/Events

The first step is taking inputs from the client, listing/search, and orders microservice into a queue.

All 3 services will be performing a POST request onto the analytics queue. All events are recorded onto the analytics queue.

Step 2. Take objects from queue and write into Writing Database (Cassandra)
The second step begins with the data readily provided by the 3 services. After the data is polled, it goes through a parser to make sure that when the data is written, it is consistent and standardized. Then analytics queue will be continuously ‘polled’ by the User Analytics microservice on a set schedule to write into the database.

One thing to note is that writing into the database is the first introduction to the User Analytics Microservice. The database will be using Cassandra

Step 3. Query the Database and send messages onto the Filtering Queue

Then once the data is written, every week the database will update and send however many messages onto the filtering queue. This is in preparation so that the filtering microservice will have data ready to poll.

Step 4. The last step is the Filtering microservice polling to the report queue

This requires the microservice to ping to a queue to get the data to update user recommendations on the client side.

Tools and technologies I plan to use

Cassandra DB

AWS SQS queue

AWS EC2

Node, Express

Docker

New Relic