

accountType [ENUM]
- Admin (level 0)
- BuyerAndSeller (level 1)
- Seller (level 2)
- Buyer (level 3)

You cannot change your account type into a lower level, you cannot change your account from Buyer to Seller or vice-versa, you will only have the option to be both BuyerAndSeller (upgrading to level 1)

If there is at least one active order that is in a state of being processed, you cannot change your delivery address

RED = shown on public
GREEN = internal
mongodb key.

To reduce complexity (and finish this portfolio quickly), did not implement the following features:

- 1. Multiple Product Categories (& editing of available categories)
- 2. Product Returns
- 3. Chat/Messaging
- 4. Two-Factor-Authentication
- 5. Federated Security (sign up/sign in via Google or Facebook account)
- 6. Multiple Product Images (cloudinary comes into mind for media hosting on cloud)
- 7. Discount / Flash Sale Events
- 8. SMS Notifications on order operations.
- 9. Address Lookup via existing APIs to validate it's existence.
- 10. Verification via SMS (email only, made possible by using captcha to mitigate spammers)
- 11. Audit Trail, yep it's required to track changes for investigative purposes, but decided to skip it.
- 12. Inventory SMS notifications of low stock level

accountActivities
<div><div><div>PK</div><div>_accountActivityId</div></div></div>
accountId
ipAddress
loginDate
metadata
createdAt

sellers
<div><div><div>PK</div><div>_sellerId</div></div></div>
accountId
sellerId
businessName
avatarImage
storeBannerImage
privateMobileNumber
lastUpdatedAt

inventories
<div><div><div>PK</div><div>_inventoryId</div></div></div>
inventoryId
sellerId
[inventoryItems] [{ productId, stockCount }]
lastUpdatedAt

track products even when deleted, so that the associated reviews will still exist

reviews linked to orderId, why? Because one product from a seller can't always be 100% perfect, there might be defects, and it is common to provide feedbacks per product bought that is related to a specific order, it doesn't make sense to leave a review if you haven't even bought a product.

accounts
<div><div><div>PK</div><div>_accountId</div></div></div>
accountId
[accountType]
username
password
salt
email
verifiedAt
verificationHash
forgotAccountHash
createdAt
lastUpdatedAt

products
<div><div><div>PK</div><div>_productId</div></div></div>
productId
[productCategory]
sellerId
name
sku
image
description
sellingPrice
avgReviewScore
deleteReason
deletedAt
createdAt
lastUpdatedAt

buyers
<div><div><div>PK</div><div>_buyerId</div></div></div>
accountId
buyerId
firstName
lastName
privateMobileNumber
deliveryAddress
deliveryCity
deliveryPostalCode
deliveryCountry
deliveryContactNumber
lastUpdatedAt

orderItemGroups
<div><div><div>PK</div><div>_orderItemGroupId</div></div></div>
orderItemGroupId
orderId
[orderItems] [{ productId, orderCount }]
lastUpdatedAt

reviews
<div><div><div>PK</div><div>_reviewId</div></div></div>
reviewId
orderId
productId
buyerId
rating
comment
createdAt
lastUpdatedAt

orders
<div><div><div>PK</div><div>_orderId</div></div></div>
orderId
buyerId
taxPercent
totalOrderPrice
[paymentMethod]
paymentResult
deliveredAt
paidAt
createdAt
lastUpdatedAt

paymentMethod [ENUM]
- Paypal (0)
- Stripe (1)

Expanded list of unimplemented features:

- 13. Handling of fees with regards to listing product on online store (of course we don't list items for free, so we have to take a fee, it's an ecommerce business and not charity)
- 14. External courier API integration, so we just pass on the heavy lifting with regards to deliveries to an existing courier, buyers need up-to-date information on their deliveries (or else they get so paranoid).
- 15. Management of taxes
- 16. Deletion and Cancellation of Orders.
- 17. Generated store pages should be in Nextjs (so that we can use SSR for SEO optimization sake, but we are short in time).
- 18. Deleting and Deactivating of accounts.

ProductService
<div><div>QUERY</div><div>+getAllActiveProducts(): Promise<Product[]> +getProductById(productId: string): Promise<Product> +getAllProductsBySellerId(sellerId: string): Promise<Product[]> +getProductsByName(name: string): Promise<Product[]></div></div>
<div><div>MUTATION</div><div>+createProduct(createProductInput: CreateProduct): Promise<Product> +updateProduct(updateProductInput: UpdateProduct): Promise<Product> +deleteProduct(productId: string): Promise<Product></div></div>
AuthService
<div><div>QUERY</div><div>+getAccountByUsername(username: string): Promise<Account> +login(username: string, password: string): Promise<String></div></div>
AccountService
<div><div>QUERY</div><div>+getAccountById(accountId: string): Promise<Account></div></div> <div><div>MUTATION</div><div>+createAccount(createAccountInput: CreateAccount): Promise<Account> +updateAccount(updateAccountInput: UpdateAccount): Promise<Account></div></div>