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

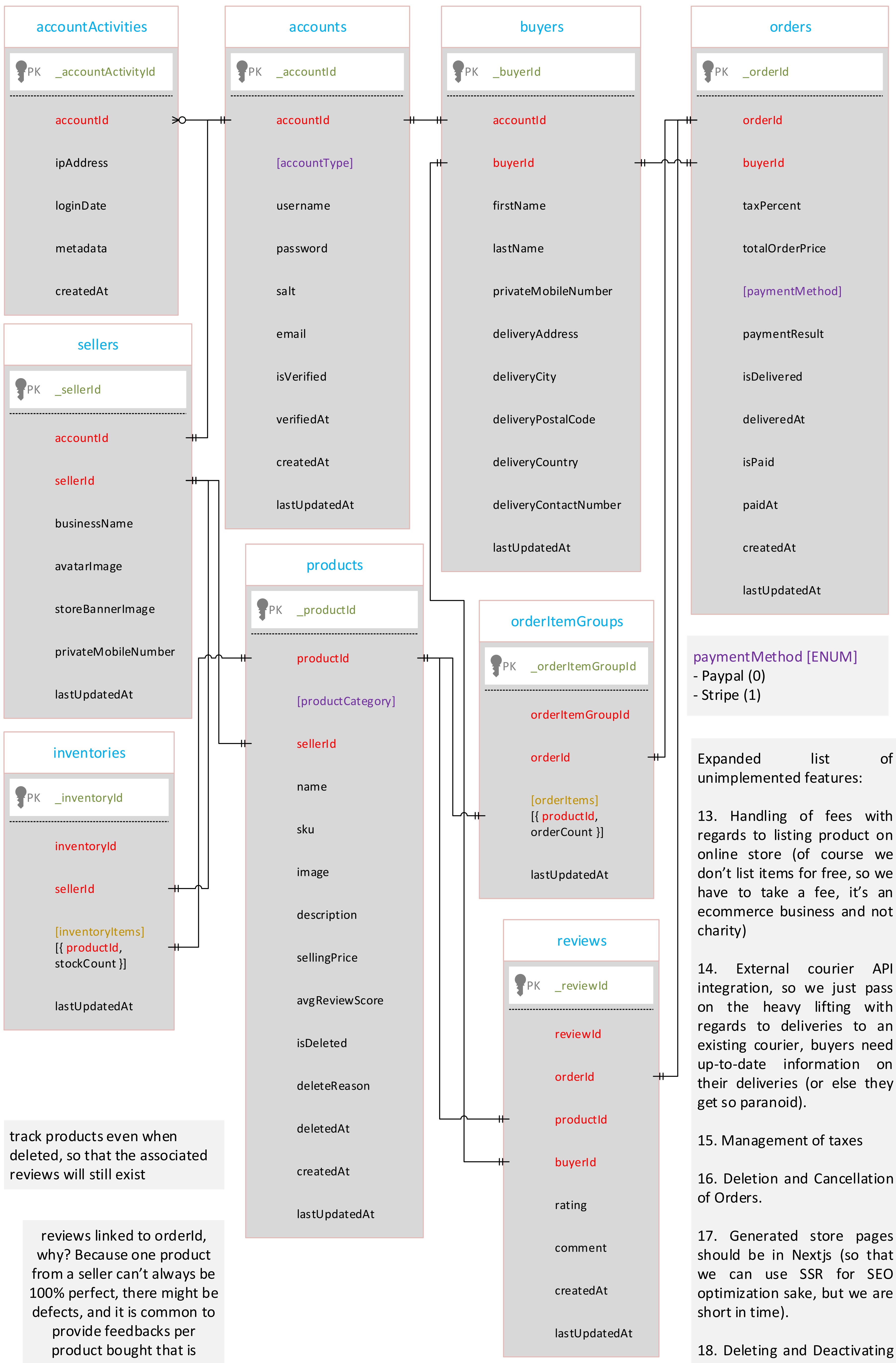
8. SMS Notifications on order operations.

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



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.

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 feedback 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.

ProductService

QUERY
+getAllActiveProducts(): Promise<Product[]>
+getProductById(productId: string): Promise<Product>
+getAllProductsBySellerId(sellerId: string): Promise<Product[]>
+getProductsByName(name: string): Promise<Product[]>

MUTATION
+createProduct(createProductInput: CreateProduct): Promise<Product>
+updateProduct(updateProductInput: UpdateProduct): Promise<Product>
+deleteProduct(productId: string): Promise<Product>

AccountService

QUERY
+getAccountById(accountId: string): Promise<Account>

MUTATION
+createAccount(createAccountInput: CreateAccount): Promise<Account>
+updateProduct(updateProductInput: UpdateAccount): Promise<Account>
+deleteProduct(productId: string): Promise<Product>