**Backend side:**

1) We send HTTP POST request to the endpoint BASE\_URL/products/:id/qr

2) Controller executes service method "generateProductQR(id)"

3) Service method validates if product exists or if the user has rights to use that product.

4) Service method calls IQRGenerator method "generateProductQR(id)" passing as parameter product id. We use public app url to embed product id inside of qr code data. For example PUBLIC\_APP\_URL/products/:productId.

5) Service method calls IStorageService method "uploadFile(filename, file)"to upload generated QR and gets url back

6) After we get url, database is called again to update product url

\*\*Note: IQRGenerator and IStorageService are interfaces meant to abstract external libraries and services.

**Frontend side:**

1) Since page can display only 10 QRs per page, we need to use pagination or search solution by name/tag/etc/ for better UX.

2) We use previously saved url from the product table in database to display image.

3) By clicking on QR code, a popup(modal etc.) pops and shows enlarged QR code.

4) The user can scan QR code with mobile phone and it should redirect to the website.

\*\*Note: I am assuming that we are scanning with phone, because that makes sense only.

**Potential problems:**

1) Potential storage service unavailability

2) Potential qr code library problems(maybe the library has some internal issues)

3) Slow loading time, because of generating and uploading file

4) Url stored in qr should not be too long because qr is limited with data

5) Potential unauthorized access(this can be done with some 3rd party tool and refers only to backend. )

6) Qr code can be small and maybe not scanable if it's not properly displayed.

**Possible solutions:**

1) 2) For storage service we need to find a provider that has the best uptime and for library we should look for most used libraries. Also it would be great to give user a notification what is happening.

3) For slow loading, it can't be avoided if we generate qr when user wants it. But we can bypass that by generating qr in the background while creating a product.

But that depends on use case that you want to provide.

4) In general if we use classic auto increment number, there shouldn't be any problems.

5) Protect storage with private buckets and signed urls if needed. Validate that user has access rights by putting guard or class validation inside of service.

6) Maximize content visibility on smaller screens. Allow every qr to be enlarged in a modal or some popup screen.