1)

A)

The most effective access control model would be attribute-based (ABAC). It is more scalable than DAC, making it more suitable for a large company like Norton. RBAC may not be suitable for Norton due to the high likelihood of role creep in a large system like Norton, which would present more opportunities for the attackers to move between systems and escalate their privileges.

Due to the fact that the attackers were able to access a wide range of data, but not the main medical record system, I suspect Norton were already using the MAC model. The attackers were probably able to gain access to the second-most privileged label, allowing them to access everything but the main medical record system. The issue with this is the lack of granularity. MAC doesn't support more granular access control based on factors like location, working hours, etc. Using ABAC with these factors would've been likely to further limit the attackers.

B)

A large company like Norton Healthcare would benefit from the scalability of this model. The fact that the hackers were able to access a wide range of data, but not the main medical record system implies that the company's exists controls are no granular enough. The most suitable model to address this would be ABAC, which would allow for fine grained access depending for a variety of factors.

For example, access to sensitive data could be limited to users whose location is in the offices of Norton, or access could be limited to working hours. This would be useful against malicious actors as attackers are likely to attack outside working hours in order to avoid being caught and are likely to be physically located far away from Norton. And so, considering what I said in A) and B), ABAC is the most effective model.

C)

One improvement that could be made to enhance security, would be multi-factor authentication (MFA). In order to gained access to the system and to have moved laterally through the system, the attackers would have had to assume the identity of various people. If only single factor authentication is used, usually a knowledge-based factor like a password, the attackers would be able to obtain hashes and crack them or perform a pass-the-hash attack to authenticate as other users.

With multi-factor authentication, the attacker would at least need have to have access to another factor. Usually this would be a possession factor (known as token-based authentication). This would limit the attacker's ability to authenticate as other users, and if they did try then the user may get notified if their possession factor is an authentication app on their phone.

The fact that the attacker was able to access a wide range of sensitive data, implies a lack of Defence-in-Depth practices. Defence-in-Depth means putting in place additional security checks so that if an attacker manages to get through a set of checks, there will be more afterwards, reducing the attackers ability to penetrate all of the checks. Implementing more security checks is likely to have reduced the data's exposure to the attackers and reduced the range of sensitive data accessed.

An extension of this idea is zero-trust security. In this model no one is trusted by default, no matter whether they are accessing from inside or outside the system. This means verification is required by everyone trying to gain access. This also enforces the idea of least privilege and combines well with the idea of MFA, in order to confine the movement and impact of unauthorised users.

2)

A)

The top 5 risks during the design phase are:

- Lack of developing threat modelling during the design phase
- Lack of attention following secure design principles
- Lack of security design awareness, guidance, and training
- Improper secure design documentation
- Lack of building and maintaining abuse case models and attack patterns

The top 5 risks during the coding phase are:

- Tampering (unauthorised modification of data)
- SQL Injection
- Cross Site Scripting and Cross-Site Request Forgery
- Denial of Services (process of making a system or application unavailable)
- Repudiation (ability of users to deny that they performed specific actions)

The top 5 risks during the testing phase are:

- Lack of penetration analysis security testing
- Lack of static and dynamic analysis security testing
- Lack of final security review
- Lack of fuzz testing
- Lack of testing against brute force attacks

The top 5 risks during the deployment phase are:

- Lack of default software configuration
- Incorrectly implemented logout
- Improperly enabled services and ports
- Ignoring security breaks
- Lack of output validation

B)

I agree with the top risk factor in the design phase being developing a threat models, as a defined threat model is required in order to understand what security measures are most needed in a design and what should be the main security priorities during the design phase. I believe that improper security design review and verification should a risk higher than ‘improper secure design documentation’, due to the importance of having a formal security assessment in the early stages of design which will reduce the number of security issues further into development. I would also suggest over-reliance on third-party applications as a risk factor, as these can lead to vulnerabilities if their security is not verified.

I agree with SQL Injection being a high risk factor due to the prevalence of SQL databases and ease of performing SQL injection attacks. The process of SQL injection can even be automated by attackers using tools like sqlmap, allowing for quick exploitation and information exfiltration of databases. I would add NoSQL injection to the list of risk factors due to the increasing use of NoSQL databases such as MongoDB, Cassandra and Redis. As NoSQL is more difficult and less common than SQL injection, it often isn't considered when coding. Hackers know this and so are likely to testing NoSQL injections. Another factor that should be considered a higher risk is Information Disclosure. This is because information leaked from Information Disclosure gives attackers a better understanding of the system they are attacking and so opens the door to discovering and exploiting other vulnerabilities.

I agree with a lack of penetration testing being a high risk during the testing phase. Penetration tests are vital for discovering vulnerabilities before attackers are able to. Security professionals that emulate the behaviour of attackers are vital for assessing the security of a system as this allows for a realistic assessment of security without actually being hacked by malicious actors. Another factor that should be added is API security testing. APIs are relied upon by many applications although they often don't receive the same level of scrutiny as other services, which often leads to them becoming a major vector for attacks.

I agree with improperly enabled services and ports being a high risk factor. A common vector for attack is services made accessible by accident. This is due to misconfiguration, exposing ports that don't need to be exposed. Attackers are constantly scanning the ports of every device on the internet, and so this is major security risk if the service wasn't designed to be exposed. An important risk factor to also consider is supply chain security and verification of products offered to users. Over the past few years, there has been more and more attacks targeting the supply chain of software products, such as the SolarWinds supply chain hack and xz-utils backdoor. This shows the importance of carrying output verification of the product throughout the deployment phase and giving users the tools to also verify the legitimacy of the software being downloaded.