Birla Institute of Technology & Science, Pilani
Work Integrated Learning Programmes Division
Second Semester 2023-2024

Mid-Semester Test
(EC-2 Regular)

Course No.	: SE ZG651
Course Title	: Software Architecture
Nature of Exam	: Closed Book
Weightage	: 30%
Duration	: 2 Hours
Date of Exam	: 16/03/2024 (FN)

No. of Pages = 3
No. of Questions = 3

Note to Students:
1. Please follow all the *Instructions to Candidates* given on the cover page of the answer book.
2. All parts of a question should be answered consecutively. Each answer should start from a fresh page.
3. Assumptions made if any, should be stated clearly at the beginning of your answer.

Q.1 [10 Marks]

You are building an inventory management system for a manufacturing company. The inventory of raw material and finished products need to be managed so that optimum raw material stocks can be maintained. The finished products will be transferred to various warehouses in container. The raw material will be used in FIFO basis based on expiry date. Any unused raw material to be taken back in the inventory and placed in such manner so that next time it will be issued based on above rule. Based on the sales booking by different location the finished products will be sent to the customer from the nearest warehouse. The management should made necessary steps so that finished products manufacturing does not get disrupted and is based on the sales order.

a. Explain the benefits of layered architecture with respect to above case study. [1]
b. How would you go about implementing layered architecture in this case? Explain with a diagram. [3]
c. Is Architecture evaluation in conflict with Agile process? Explain how an Agile Process would work in the above case. You answer should be in the context of the above case study and should cover at least 3 agile principles. [3]
d. Explain with rough sketches how you would use Kruchten's 4+1 architectural view model with respect to the above case study? [3]

a. Explain the benefits of layered architecture with respect to the above case study. [1]

Layered Architecture offers several benefits in the context of this inventory management system:

1. **Separation of Concerns:** Each layer can focus on specific responsibilities. For example:
    - **Presentation Layer**: Handles user interfaces (warehouse staff, production managers, sales team).
    - **Business Logic Layer**: Implements inventory and product handling rules (FIFO, expiry management, sales booking).
    - **Data Layer**: Manages data persistence, like storing inventory, product, and warehouse details.
2. **Maintainability and Scalability:** Changes can be made in one layer (e.g., adding a new sales booking rule in the business logic layer) without affecting other layers.
3. **Reusability:** Business logic for managing raw materials (FIFO, expiry) can be reused in other parts of the system or in future modules without repeating code.
4. **Testability:** Each layer can be tested independently. For example, the business logic can be tested separately from the user interface.

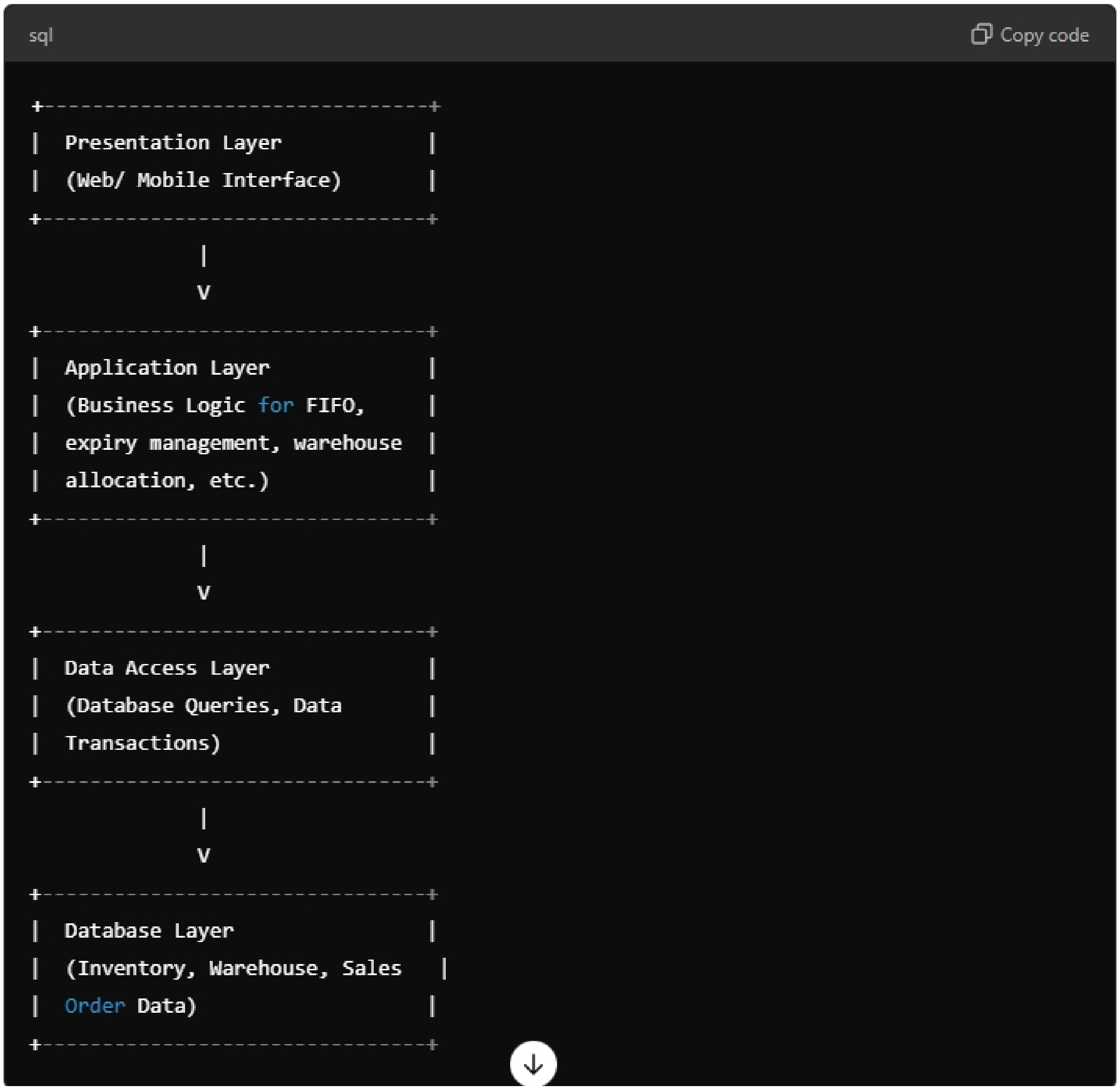
---

b. How would you go about implementing layered architecture in this case? Explain with a diagram. [3]

To implement the layered architecture for this inventory management system, we can break it into the following layers:

1. **Presentation Layer (UI Layer):**
    - Interfaces for warehouse staff, production managers, and sales teams to manage and view inventory, production, and sales orders.
2. **Application Layer (Business Logic Layer):**
    - Logic for raw material management (FIFO, expiry dates, stock replenishment).
    - Logic for finished goods allocation to warehouses based on sales bookings.
3. **Data Access Layer:**
    - Handles data transactions with databases that store inventory, product details, sales orders, and warehouse information.
4. **Database Layer:**
    - Stores raw material stock, finished products, warehouse locations, and sales order data.

Example Diagram:



```
+-------------------------------+
|  Presentation Layer           |
|  (Web/ Mobile Interface)      |
+-------------------------------+
              |
              v
+-------------------------------+
|  Application Layer            |
|  (Business Logic for FIFO,    |
|  expiry management, warehouse |
|  allocation, etc.)            |
+-------------------------------+
              |
              v
+-------------------------------+
|  Data Access Layer            |
|  (Database Queries, Data      |
|  Transactions)                |
+-------------------------------+
              |
              v
+-------------------------------+
|  Database Layer               |
|  (Inventory, Warehouse, Sales  |
|  Order Data)                  |
+-------------------------------+
```

---

c. Is architecture evaluation in conflict with Agile process? Explain how an Agile Process would work in the above case. Your answer should be in the context of the above case study and should cover at least 3 agile principles. [3]

Architecture evaluation and **Agile processes** are not in conflict. In fact, Agile supports continuous evaluation and iteration of architecture to ensure it meets the changing business needs. Here's how Agile principles would work in this case study:

1. **Customer Collaboration over Contract Negotiation:**
    - The system can be built iteratively based on continuous feedback from different stakeholders (warehouse managers, sales team). For example, early versions might focus on raw material management, and later iterations could add warehouse allocation and sales features.
2. **Working Software over Comprehensive Documentation:**

   - Rather than focusing on upfront, detailed documentation of the entire system, Agile encourages delivering a working version of the inventory system early, such as basic raw material management with FIFO. This working system can then be expanded based on feedback.
3. **Responding to Change over Following a Plan:**
   - In the context of this case study, the requirements for inventory handling, sales booking, or FIFO might evolve. Agile allows for flexibility, so changes (like new sales strategies or raw material handling procedures) can be incorporated during development without major disruptions.

Thus, Agile processes allow continuous refinement and testing of the architecture, ensuring that the system adapts to the business's changing needs.

---

d. Explain with rough sketches how you would use Kruchten's 4+1 architectural view model with respect to the above case study. [3]

Kruchten's 4+1 View Model is a framework for describing the architecture of a system using five views:

1. **Logical View:**
   - This captures the key abstractions (like RawMaterial, FinishedProduct, Warehouse, SalesOrder).

```diff
- RawMaterial
- FinishedProduct
- Warehouse
- SalesOrder
```

2. **Development View:**
   - This view focuses on the organization of the codebase. For example, it might show how different modules (like InventoryManagement, WarehouseManagement, SalesModule) are structured in the code.

diff Copy code

```diff
+ InventoryManagement
+ WarehouseManagement
+ SalesModule
```

3. **Process View:**
   - Describes the system's runtime behavior. For example, how raw materials are moved in FIFO order or how sales orders are processed.

mathematica Copy code

```mathematica
FIFO Process -> Stock Check -> Expiry Check -> Allocate Material
```

4. **Physical View:**
   - Describes the physical deployment of the system, such as which servers or databases will host the inventory and warehouse information.

diff Copy code

```diff
+ Database Server
+ Web Server
+ Warehouse Systems
```

5. **Scenarios (+1 View):**
   - This view ties all the other views together using real-world scenarios. For example:

- "How is raw material stock updated based on expiry?"
- "How is finished product allocated to a warehouse based on sales orders?"

mathematica Copy code

```mathematica
Scenario: Sales Order Processing
-> Check Nearest Warehouse -> Allocate Finished Product
```

These sketches help visualize how Kruchten's 4+1 views can be applied to this case study.

Q.2 [10 Marks]

The GoLearn system is a digital learning environment used to support learning in schools for students from age 4 to 18. It is intended to replace an existing system (Utech) that was specially built for this purpose, and which includes its own applications for e-mail, etc. Utech was a closed system where it was impossible for users to introduce their own applications. It had a poor reliability and was not popular as the facilities in freely available systems were far superior to those offered in this closed system.

One of the most important requirements for the GoLearn system was that it should be an open system that could easily accommodate new features and existing services. Developers aimed to achieve this by designing the system so that everything was a service and that, with appropriate permissions, users could replace pre-specified services with their own service version. This approach also allowed the developers to deal with the complexity of integrating with existing network management systems (local areas had different policies on which websites could be visited by school students, depending on age and content) and school administration systems. Hence, the developers created a service interface to this system which in turn can accommodate different underlying systems.

There are three types of service in the system:

- Utility services that provide basic application-independent functionality and which may be used by other services in the system. Utility services are usually developed or adapted specifically for this system.
- Application services that provide specific applications such as email, conferencing, photo sharing etc. and access to specific educational content such as scientific films or historical resources. Application services are external services that are either specifically purchased for the system or are available freely over the Internet.
- Configuration services that are used to adapt to the environment with a specific set of application services and define how services are shared between students, teachers and their parents.

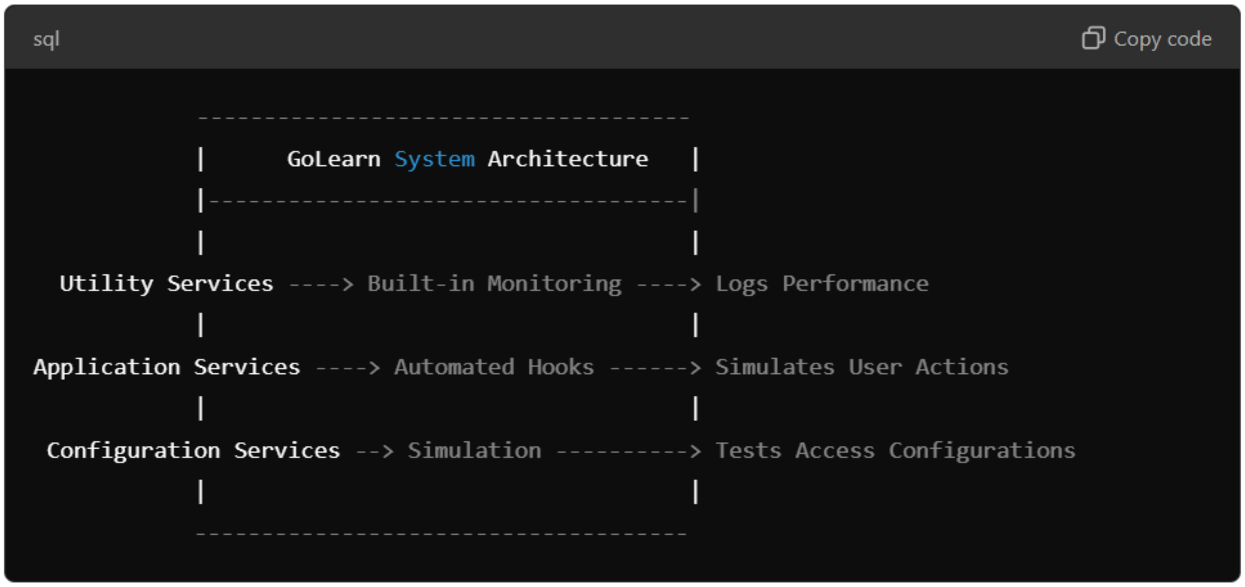
a. How can the testability tactics be applied for the utility services, application services and configuration services? Explain your answer by drawing a diagram giving the scenario only one tactic for each of these services in the context to the case study [5]
b. "Utech is replaced by GoLearn"– Justify the role of usability-tactics in this context. [2]
c. Please give examples of 3 Design Decision for Performance that you would take for the services described in the above case. The answer must be in the context of the case study given. [3]

a. Testability Tactics for Utility, Application, and Configuration Services:

Testability tactics improve the system's ability to be tested, which ensures that defects can be identified and resolved easily. In the context of the GoLearn system, these tactics apply to different services.

- **Utility Services**: These are core services providing application-independent functionality.
- **Testability Tactic: Built-In Monitoring**
- *Scenario*: For utility services, using built-in monitoring enables automatic detection of failures or performance bottlenecks.
- *Explanation*: By integrating monitoring tools within utility services, we can track their behavior during testing, making it easier to analyze test results and diagnose failures without intrusive modifications. For instance, monitoring the load balancer's functionality helps ensure the utility services are distributing load properly.
- **Application Services**: These provide the functionality that users directly interact with (e.g., email, photo sharing).
    - **Testability Tactic: Automated Testing Hooks**
        - *Scenario*: Application services like email or content delivery can benefit from automated testing hooks.
        - *Explanation*: Integrating testing hooks into these services allows the system to automatically test application functionality (e.g., sending and receiving emails) by simulating user interactions and detecting issues. This tactic reduces manual testing overhead.
- **Configuration Services**: These services define how users (students, teachers, parents) interact with the system.
    - **Testability Tactic: Simulation**
        - *Scenario*: Configuration services should simulate different environment configurations during testing to ensure robustness.
        - *Explanation*: Simulation of different configurations (e.g., different access policies for students vs. teachers) during testing allows developers to verify that the system adapts to varying environments as expected. This simulation helps in testing configuration services for flexibility.

Diagram Representation: The following diagram shows how these testability tactics apply to different services:


sql Copy code

```sql
             -----------------------------------
             |      GoLearn System Architecture   |
             |-----------------------------------|
             |                                   |
  Utility Services ----> Built-in Monitoring ----> Logs Performance
             |                                   |
Application Services ----> Automated Hooks ------> Simulates User Actions
             |                                   |
 Configuration Services --> Simulation ----------> Tests Access Configurations
             |                                   |
             -----------------------------------
```

b. Role of Usability Tactics in Replacing Utech with GoLearn

The usability tactics play a crucial role in the transition from Utech to GoLearn. Utech was unpopular due to its lack of features and a closed system design, which made it difficult for users to adopt or introduce new services.

- **Feedback Tactic**: Providing clear, real-time feedback helps users understand the current system status and confirms their actions (e.g., when teachers upload content or students send emails).
- **Error Prevention Tactic**: Ensuring that the system prevents common user mistakes (such as configuration errors or accidental deletion of student data) enhances usability and encourages the adoption of GoLearn over Utech.
- **Consistency Tactic**: Maintaining consistency across different interfaces (like email, video conferencing, or resource sharing) helps users seamlessly navigate the system without relearning functions, a key improvement from Utech's fragmented interface.

Thus, the **role of usability tactics** ensures GoLearn is more intuitive, easy to navigate, and adaptable to users' needs compared to the closed, rigid Utech system.

c. 3 Design Decisions for Performance:

1. **Service Caching for Application Services**:
   - **Decision**: Introduce caching for frequently used application services like educational content delivery.
   - **Context**: Application services (e.g., scientific films) may require large data downloads. Caching these resources minimizes the load on external servers and reduces response times, leading to improved performance during peak usage by students and teachers.
2. **Load Balancing for Utility Services**:
   - **Decision**: Use load balancing to distribute the requests to utility services (e.g., authentication or session management) across multiple servers.
   - **Context**: Utility services handle critical operations like login authentication, which multiple users (students, teachers, parents) may access simultaneously. Load balancing ensures these services scale properly and maintain performance under heavy load.
3. **Prioritization of Critical Services for Configuration Services**:
   - **Decision**: Implement prioritization mechanisms for critical configuration services.
   - **Context**: Certain configuration services (e.g., network access controls) are more critical than others and should receive higher priority in processing to maintain system integrity. For example, when students from different age groups access restricted content, the system must prioritize checking and enforcing appropriate access rules swiftly to ensure a secure environment.

Q.3 [10 Marks]

Case Hospital Patient Management System

Objective: A hospital seeks to modernize its patient management system to improve security and user experience.

Background:

The hospital is in the business of healthcare delivery. The patients in the hospital should be safe and any information about them must be secure. It should facilitate the working of all employees and consider the needs of visitors and patients.

Challenges:

- Identify ASRs related to security and usability.
- Understand user needs and workflows for effective system design.

Answer the following questions in the above context. You may use your own knowledge of the working of hospitals while answering this question:

a. What are the hospital's goals for the new patient management system? [1]
b. What are the roles and responsibilities of stakeholders involved in the system (doctors, nurses, patients, administrators)? [2]
c. Describe using diagrams 4 Brainstorm scenarios for potential security breaches and usability issues. [4]
d. What methodology have you studied in the course to assist in prioritizing scenarios for a small project where the time available is short. [1]
e. Prioritize scenarios based on their criticality, considering the potential impact on patient safety and user experience. [1]
f. Is there any international standard for deciding what scenarios are the be given priority? [1]

a. What are the hospital's goals for the new patient management system? [1]

The hospital's primary goals for the new patient management system include:

1. **Enhancing Security:** Ensuring the safety and confidentiality of patient data, complying with healthcare regulations such as HIPAA (Health Insurance Portability and Accountability Act) or GDPR (General Data Protection Regulation).
2. **Improving Usability:** Facilitating an easy-to-use interface for hospital staff (doctors, nurses, administrators) to manage patient information efficiently, while considering the needs of patients and visitors.

---

b. What are the roles and responsibilities of stakeholders involved in the system (doctors, nurses, patients, administrators)? [2]

1. **Doctors:**

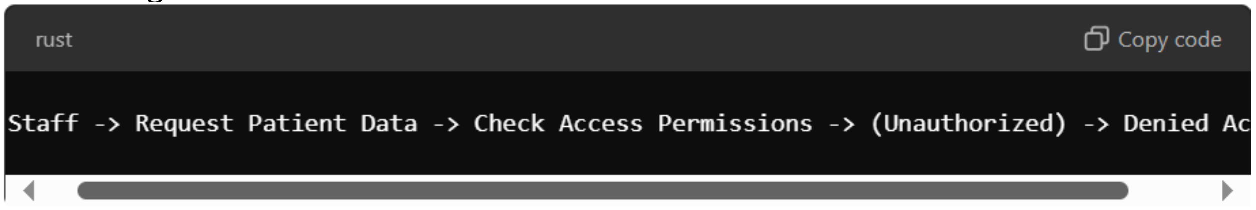
   - **Role:** Primary caregivers responsible for diagnosing and treating patients.
   - **Responsibilities:**
     - Access patient medical history, prescriptions, lab results, and update treatment plans.
     - Ensure that sensitive medical data remains secure while in use.
2. **Nurses:**
   - **Role:** Support doctors in patient care and treatment.
   - **Responsibilities:**
     - Manage day-to-day patient care by recording vital signs, administering medication, and updating patient records.
     - Ensure patient data is updated accurately and securely.
3. **Patients:**
   - **Role:** Recipients of medical care.
   - **Responsibilities:**
     - Interact with the system to access personal health information, appointment schedules, and communication with healthcare providers.
4. **Administrators:**
   - **Role:** Manage the overall functioning of the hospital and its systems.
   - **Responsibilities:**
     - Maintain the patient management system, ensuring it operates efficiently.
     - Control access permissions, ensuring compliance with data protection regulations and overseeing system security.

---

c. Describe using diagrams 4 brainstorm scenarios for potential security breaches and usability issues. [4]

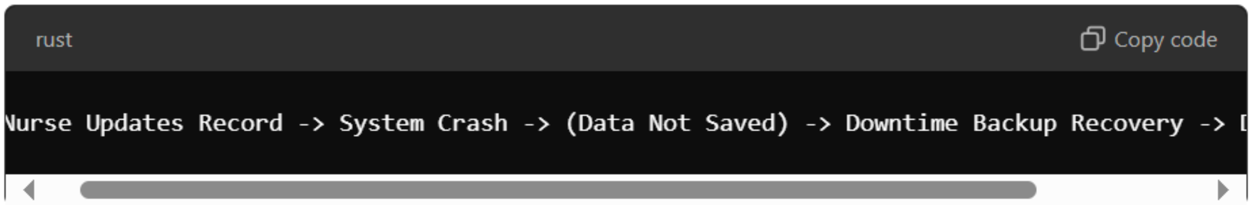
Here are 4 brainstormed scenarios for security breaches and usability issues in a hospital patient management system. I will outline the situations, and you can visualize them with simple flowcharts.

1. **Unauthorized Access to Patient Records:**
   - **Scenario:** A staff member accesses a patient's medical history without proper authorization.
   - **Security Breach:** Sensitive patient data is exposed to individuals who do not have a valid reason to view it.
   - **Solution:** Implement role-based access control (RBAC) and detailed logging of all access events.
   - **Diagram:**



```
Staff -> Request Patient Data -> Check Access Permissions -> (Unauthorized) -> Denied Ac
```

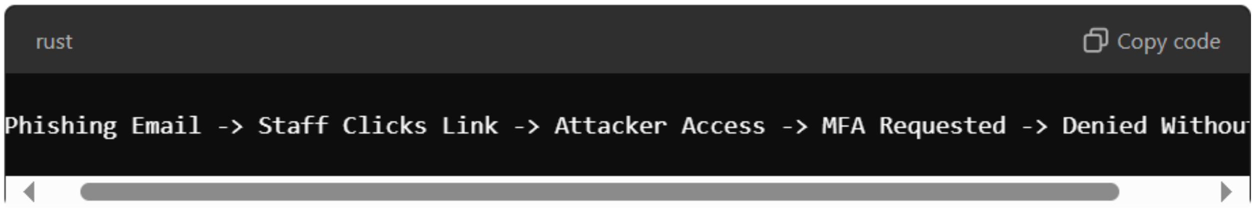
2. **Data Loss During System Downtime:**
   - **Scenario:** System experiences unexpected downtime, and patient data updates are not saved.
   - **Usability Issue:** Critical medical data could be lost or not available during emergencies.
   - **Solution:** Implement automatic data backups and ensure real-time data replication.
   - **Diagram:**



```rust
Nurse Updates Record -> System Crash -> (Data Not Saved) -> Downtime Backup Recovery ->
```

```
-> Data Restored
```

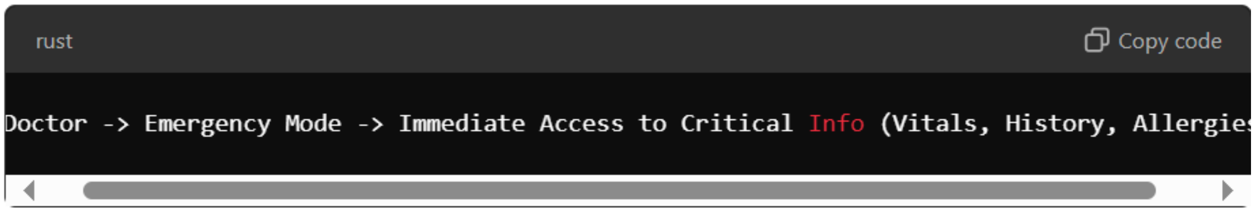
3. **Phishing Attack on Hospital Staff:**
   - **Scenario:** A staff member clicks a phishing email link, unknowingly granting access to malicious actors.
   - **Security Breach:** The attacker gains unauthorized access to the patient management system.
   - **Solution:** Implement email security training, multi-factor authentication (MFA), and phishing detection mechanisms.
   - **Diagram:**



```rust
Phishing Email -> Staff Clicks Link -> Attacker Access -> MFA Requested -> Denied Withou
```

```
-> Denied Without MFA
```

4. **Difficulty Navigating System During Emergencies:**
   - **Scenario:** A doctor in an emergency struggles to navigate the system to access critical patient information.
   - **Usability Issue:** Poor user interface design hinders quick access to life-saving information.
   - **Solution:** Implement a user-friendly interface with an emergency mode for rapid access to essential data.
   - **Diagram:**



```rust
Doctor -> Emergency Mode -> Immediate Access to Critical Info (Vitals, History, Allergie
```

---

d. What methodology have you studied in the course to assist in prioritizing scenarios for a small project where the time available is short? [1]

The **MoSCoW (Must have, Should have, Could have, Won't have)** method is commonly used for prioritizing tasks in time-constrained projects. It helps focus on the most critical requirements (Must haves) while deprioritizing lower-priority tasks that can be addressed later (Should/Could haves).

---

e. Prioritize scenarios based on their criticality, considering the potential impact on patient safety and user experience. [1]

1. **Unauthorized Access to Patient Records (High Priority):**
   Impact on patient privacy and data security is critical; thus, ensuring proper access control is a "Must have."
2. **Difficulty Navigating System During Emergencies (High Priority):**
   Direct impact on patient safety during emergency situations makes this scenario a "Must have."
3. **Data Loss During System Downtime (Moderate Priority):**
   While important, regular backups and failover mechanisms can reduce the immediate risk. This would be a "Should have."
4. **Phishing Attack on Hospital Staff (Moderate Priority):**
   Security training and MFA mitigate this risk, but it's a "Should have" rather than an immediate "Must have."

---

f. Is there any international standard for deciding what scenarios are to be given priority? [1]

Yes, the **ISO/IEC 29148:2011** standard provides guidelines for requirements engineering, including how to prioritize scenarios and requirements based on their criticality, risk, and impact on the system. Additionally, **ISO/IEC 27001** outlines security management practices to help prioritize security-related scenarios based on risk assessment.