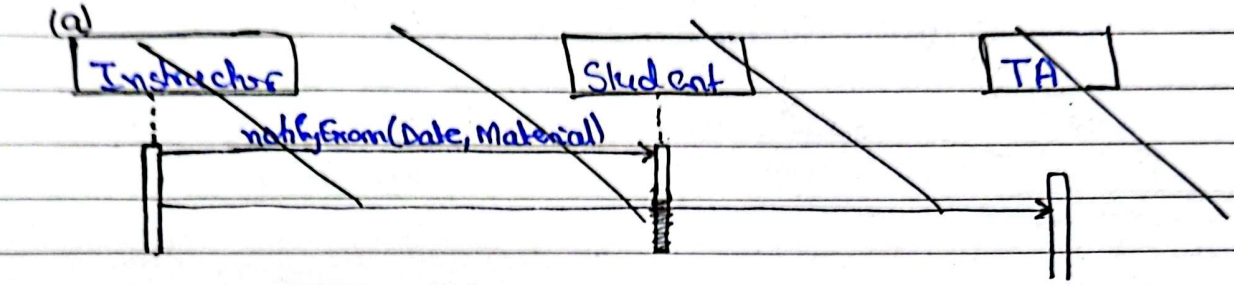
# Assignment 2

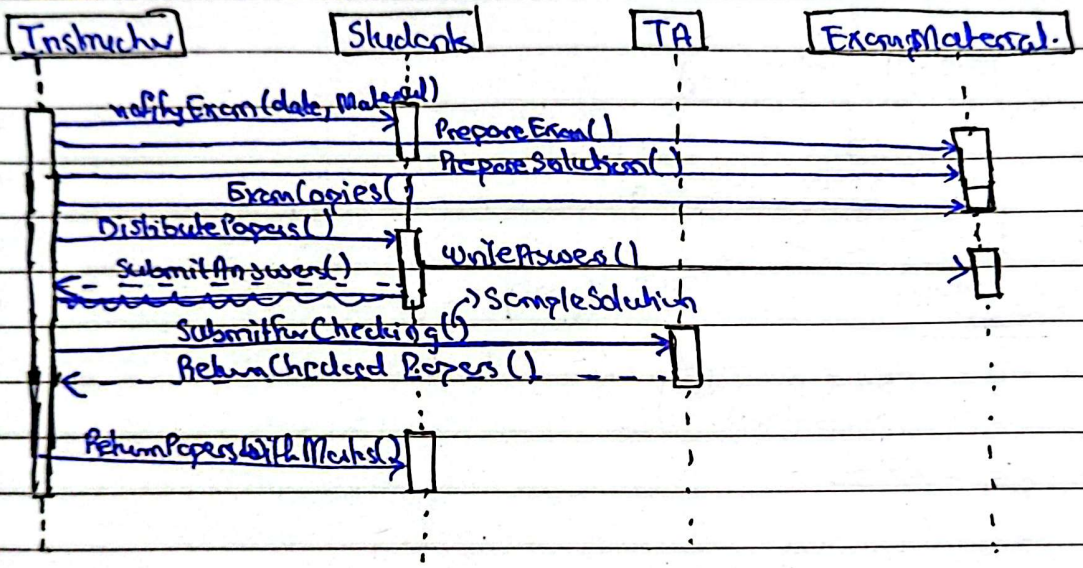
Date _____

## Question 1

(a)

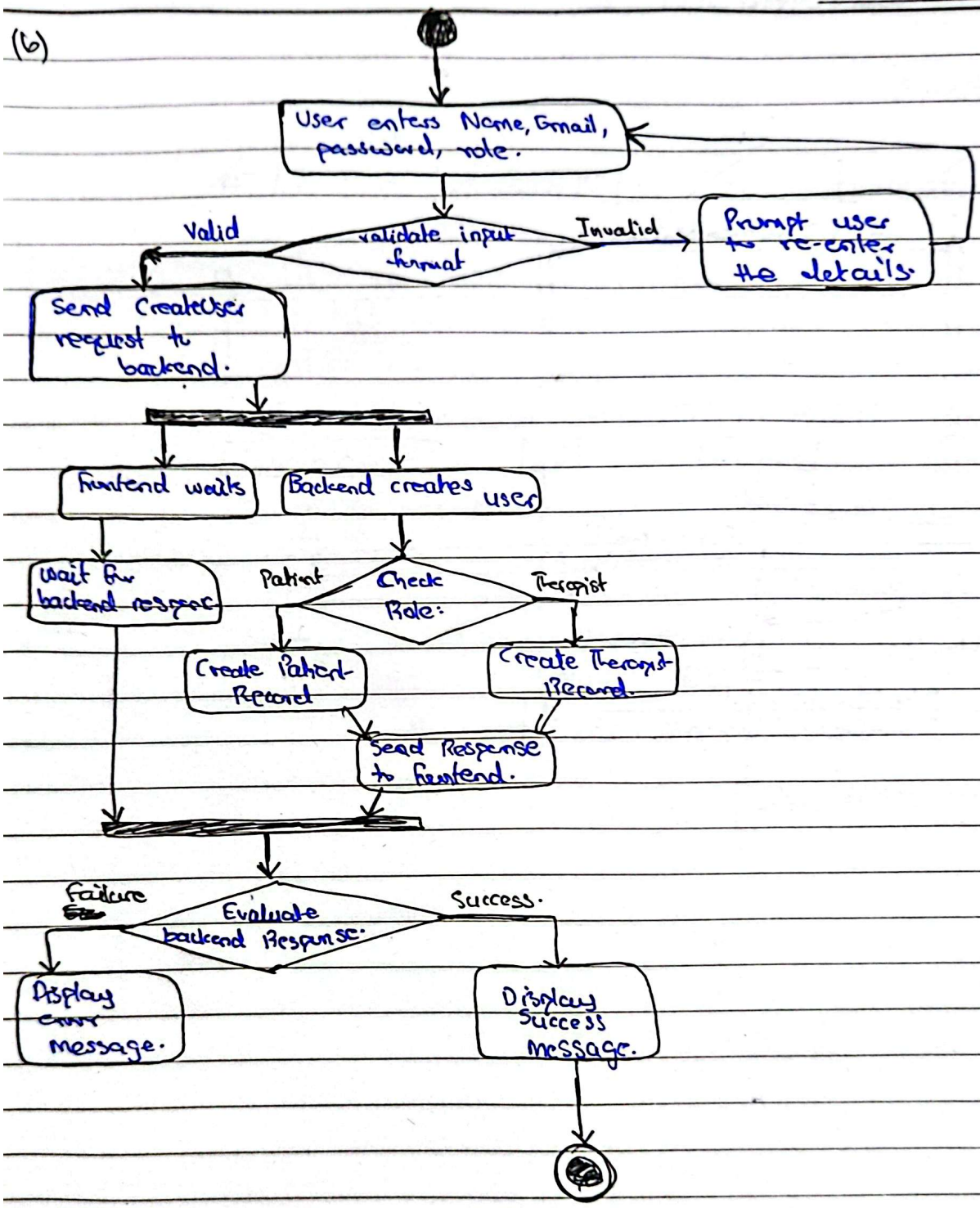



~~Instructor~~ ~~Student~~ TA



Date

(6)


User enters Name, Gmail, password, role.
validate input format
Valid
Invalid
Prompt user to re-enter the details.
Send CreateUser request to backend.
Frontend waits
Backend creates user
wait for backend response
Check Role:
Patient
Therapist
Create Patient Record
Create Therapist Record.
Send Response to frontend.
Evaluate backend Response:
Failure
Success.
Display error message.
Display Success message.

Date

Question 2:

1) The encryption module and User Interface are tightly coupled, which means any change in encryption logic necessitates modifications to the UI code. This results in much more effort needed for maintenance, and may also increase the risk of bugs. Moreover, tight coupling may expose sensitive encryption logic to the UI layer. potentially leading to security vulnerabilities. Furthermore, adding new encryption algorithm or updating existing ones becomes difficult due to dependencies on the UI.

2) This issue is associated with low cohesion in notification module. The notification module combines multiple unrelated responsibilities into a single component, which results in the code becoming complex and harder to debug or extend. A change in one functionality might inadvertently affect others. Moreover, mixing unrelated logic increases the likelihood of bugs. In addition, the module's purpose is unclear, making it difficult for new developers to understand or modify.

3) This issue is related to low cohesion in Logging system. The system combines user activity logs and encryption logs, making it difficult to separate security-related logs from general activity logs., this may result in delayed detection of security issues. Moreover, Troubleshooting becomes harder because logs are not modular.

Date

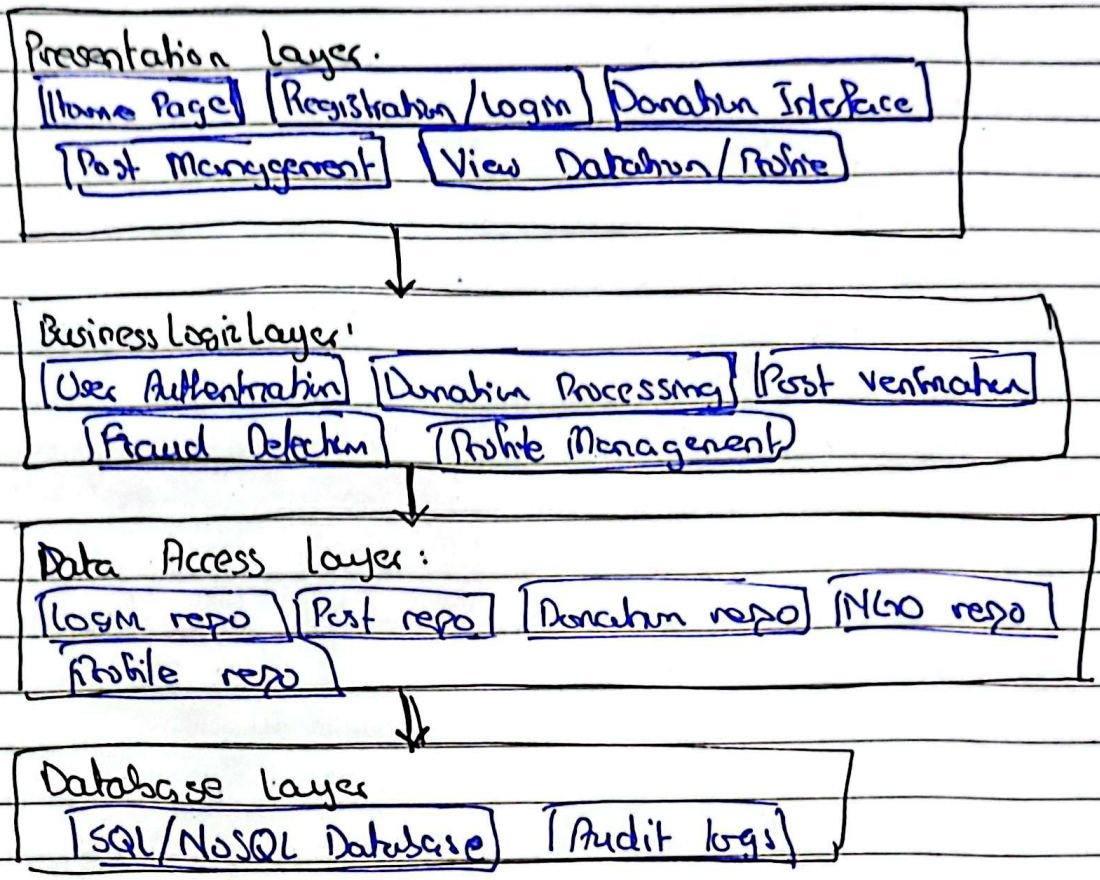
4) This issue is associated with the design of the system. The monolithic design leads to high coupling. This suggests that a failure in one component can crash the entire system. Moreover, as components cannot be updated or deployed independently, scaling the system also becomes difficult. It also causes the system to become vulnerable to security threats as a vulnerability in one part of the system can compromise the entire application.

Question 3:

1) Presentation Layer: {Home, Registration/Login, Donation Interface, Post Management, View Profile}
2) ~~Business logic~~
3) Business Logic Layer: {User Authentication, Donation Processing, Post verification, Fraud Detection, Profile Management}
3) Data Access Layer: {Login repo, Post repo, Donation repo, NGO repo, User Profile repo}
4) Database Layer: {SQL/NoSQL Database, Audit logs}

23K-0089

Date



Date

Question 4:

⇒ Selected Pattern: Event-Driven microservices architecture.

→ Explanation:

1) The system relies on live data from sensors, GPS, and CCTV cameras, and event-driven architecture allows asychronous processing of events.
2) Each component can be a separate microservice, enabling independent updates and scaling.
3) If one microservice fails others remain operational.
4) The system can easily integrate with external systems via event brokers.
5) Machine learning models can subscribe to relevant events for live analysis.