Name: Abid

Roll Number: SP21-BSE-085

Assignment 01

Layered Architecture: Success and Failure

Introduction

This report explores the Layered Architecture, examining a successful and a failed software implementation in relation to this design pattern. We will also delve into a different architecture, highlighting its pros and cons.

Successful Example: Amazon.com

Amazon.com exemplifies the Layered Architecture's success. It separates its application into distinct layers:

- Presentation Layer: Manages the user interface (website and mobile app).
- Business Logic Layer: Handles core functionalities like order processing and product search.
- Data Access Layer: Interacts with databases for product information, customer details, and order management.

Reasons for Success

- **Maintainability:** Each layer is independent, making changes and updates easier without affecting other parts.
- **Scalability:** Layers can be scaled independently to handle increased traffic or data volume.
- **Reusability:** Business logic components can be reused across different applications within Amazon.

Failed Example: Therac-25 Radiation Therapy Machine

The Therac-25 radiation therapy machine is a cautionary tale of Layered Architecture gone wrong. Software bugs in the Therac-25's control system led to several patient overdoses and deaths in the 1980s.

Reasons for Failure

- **Inadequate Layer Separation:** Mixing safety checks with treatment calculations in the same layer led to critical errors.
- **Lack of Testing:** Insufficient testing across different input scenarios allowed bugs to remain undetected.
- **Poor Documentation:** Unclear documentation hampered understanding and troubleshooting.

Alternative Architecture:

Microservices Architecture

Microservices Architecture is a popular alternative to Layered Architecture. It decomposes an application into smaller, independent services that communicate via APIs.

Pros:

- **Increased Agility:** Independent services allow for faster development and deployment cycles.
- **Improved Fault Tolerance:** Issues in one service don't bring down the entire application.
- **Scalability:** Individual services can be scaled based on specific needs.

Cons:

- **Complexity:** Managing numerous distributed services can be challenging.
- **Debugging:** Troubleshooting issues across multiple services requires more effort.
- **Testing:** Testing interactions between services becomes more intricate.

Cost Considerations

The Therac-25 incident resulted in lawsuits and settlements costing the company millions of dollars. Implementing robust testing procedures and stricter software engineering practices could have prevented such tragedies, highlighting the importance of upfront investment for long-term cost savings.