Mini Speech 1: Informative Speaking - Topic Proposal

# Figure Selection:

I will select a chart comparing microservices architecture and monolithic architecture in terms of scalability, fault isolation, and development flexibility.

# Specific Purpose Statement:

To inform my audience, I will explain the differences between microservices architecture and monolithic architecture in software development and discuss their suitable use cases.

# Thesis Statement:

In this speech, I will analyze how microservices architecture enhances system scalability and maintainability through modularization, while monolithic architecture remains more efficient for simpler applications.

# Figure Explanation:

Microservices architecture and monolithic architecture are two significant system design methodologies in software development, each with its advantages and challenges. Monolithic architecture involves bundling the entire application into a single executable package where all functionality modules share the same code base. This design is efficient for small applications but becomes increasingly difficult to maintain and scale as the application grows.  
  
Microservices architecture, on the other hand, breaks an application into a set of small, autonomous services, each focusing on a single functionality. These services communicate through lightweight protocols like HTTP or message queues. This modular approach enables teams to develop and maintain code more independently, thus enhancing scalability and fault isolation.

# Architecture Comparison:

1. Scalability:  
 - Microservices: Allows independent scaling of specific services based on demand, rather than scaling the entire application.  
 - Monolithic: Typically requires scaling the entire system, which becomes inefficient as the application size increases.  
2. Fault Isolation:  
 - Microservices: Services are isolated, meaning a failure in one service does not affect the entire system.  
 - Monolithic: Since all functionality shares the same code base, a failure in one part can affect the entire application.  
3. Development Flexibility:  
 - Microservices: Teams can work in parallel on different services, and each service can use a different technology stack, offering greater flexibility.  
 - Monolithic: Requires tighter collaboration among developers since all changes are made in the same code base, increasing development complexity.

# Research Sources:

1. Fowler, Martin. \*Microservices: A Definition of This New Architectural Term\*. https://martinfowler.com/articles/microservices.html  
2. Richardson, Chris. \*Microservices Patterns: With Examples in Java\*. Manning Publications, 2018.  
3. Newman, Sam. \*Building Microservices: Designing Fine-Grained Systems\*. O'Reilly Media, 2021.  
4. Namiot, Dmitry, and Manfred Sneps-Sneppe. \*On Micro-services Architecture\*. International Journal of Open Information Technologies, 2014. https://ijoit.org/articles/10.13140/RG.2.1.4075.9287  
5. Jamshidi, Pooyan, et al. \*Microservices: The Journey So Far and Challenges Ahead\*. IEEE Software, 2018. https://doi.org/10.1109/MS.2018.2141039