**System Design Document**

**1. Introduction**

**Purpose:** This document describes the system design options evaluated for the Capstone project, providing a thorough analysis of each option and the rationale for the final decision.

**Scope:** The project aims to develop a web-based application for task management, requiring a reliable, scalable, and maintainable system architecture.

**2. System Design Options**

**Option 1: Monolithic Architecture**

* **Description:** All components (frontend, backend, and database) are packaged and deployed as a single unit.
* **Block Diagram:**
  + Components:
    - Frontend
    - Backend
    - Database

**Option 2: Microservices Architecture**

* **Description:** The application is divided into smaller, independent services, each responsible for a specific functionality.
* **Block Diagram:**
  + Components:
    - Authentication Service
    - Task Management Service
    - Notification Service
    - Database

**Option 3: Serverless Architecture**

* **Description:** The application runs on serverless functions, with each function handling a specific task. This design leverages cloud services for execution.
* **Block Diagram:**
  + Components:
    - Authentication Function
    - Task Management Function
    - Notification Function
    - Database

**3. Evaluation of Design Options**

**Criteria for Evaluation:**

* **Scalability:** Ability to handle increased load.
* **Maintainability:** Ease of updating and maintaining the system.
* **Performance:** Speed and efficiency of the system.
* **Cost:** Overall cost of development and operation.
* **Complexity:** Difficulty level of implementation.

**Evaluation of Option 1: Monolithic Architecture**

* **Scalability:** Limited, as the entire application needs to be scaled together.
* **Maintainability:** Challenging, as changes in one part affect the whole system.
* **Performance:** Good for small to medium-sized applications.
* **Cost:** Lower development cost but higher maintenance cost.
* **Complexity:** Less complex to implement initially.

**Evaluation of Option 2: Microservices Architecture**

* **Scalability:** High, as individual services can be scaled independently.
* **Maintainability:** High, easier to update specific services without affecting the entire system.
* **Performance:** High, as services can be optimized individually.
* **Cost:** Higher initial development cost but lower maintenance cost.
* **Complexity:** More complex to implement due to service management and inter-service communication.

**Evaluation of Option 3: Serverless Architecture**

* **Scalability:** Very high, as functions scale automatically.
* **Maintainability:** High, with independent, easily replaceable functions.
* **Performance:** High, with optimized execution times.
* **Cost:** Pay-per-use model can be cost-effective but may become expensive with high usage.
* **Complexity:** Moderate complexity in initial setup and function orchestration.

**4. Final Decision**

**Justification for the Chosen Option:**

After evaluating all options, **Microservices Architecture** is chosen for the following reasons:

* **Scalability:** Allows independent scaling of services, making it suitable for potential future growth.
* **Maintainability:** Easier to update and manage specific services, reducing downtime and improving agility.
* **Performance:** High performance due to the ability to optimize individual services.
* **Cost:** Although initial development costs are higher, long-term maintenance costs are lower due to improved maintainability and scalability.

**5. Conclusion**

The Microservices Architecture was selected based on its advantages in scalability, maintainability, and performance. This architecture aligns with the project's goals of developing a reliable, scalable, and maintainable web-based task management application.