# SDD (Project detailed documentation)

## 1- Introduction

Taskaty is a task management application designed to help users organize their daily tasks efficiently.Taskaty empowers users to create, track,delete, and manage their tasks seamlessly to stay organized and productive.

### 1.1 Purpose :

This document aims to comprehensively outline the design aspects of Taskaty, focusing on TaskMicroservice and UserMicroservice applications. It provides insights into the system's architecture, components, and interactions.

### 1.2 Scope :

The document covers the design and implementation details of TaskMicroservice and UserMicroservice for the Taskaty app. It includes architecture, validation mechanisms, Aspect-Oriented Programming (ASOP) implementations, microservices design principles, API Documentation, and deployment on cloud infrastructure.

### 1.3 Overview :

TaskMicroservice and UserMicroservice are built using Spring Boot, emphasizing scalability, maintainability, and reliability.

# 2- Architectural Overview

## 2.1 High-Level Architecture :

Taskaty follows a three-tier architecture: :

1- Presentation Layer: Frontend components (HTML, CSS, JavaScript).

2- Application Layer: Backend services (Java, Spring Boot, Spring Data JPA,Spring Cloud).

3- Data Layer: Database (MySQL) for user and task data.

## 2.2 Design Principles

The architecture follows the MVC design pattern for separating concerns and ensuring maintainability and scalability.

# 3- System Description(Requirements)

## 3.1 Task Microservice :

Manages tasks for users with functionalities including task creation, retrieval, update, and deletion:

- Creating a task for a user.
- Retrieving tasks for a user.
- Updating task status.
- Updating task title.
- Deleting a task.

## 3.2 User Microservice :

Manages user authentication and profile management, including sign-up, login, and profile updates:

- User sign-up.
- User login.
- Updating user email.
- Updating user password.

# 4- Architecture

## 4.1 Validation :

Both microservices utilize validation mechanisms to ensure data integrity and enforce business rules. Validation is performed using annotations such as @NotBlank, @Size,@Pattern and @Email from the Jakarta Bean Validation API.

- The Validation used :
    a. Email addresses must not be null, blank, and must be unique within the system.
    b. Password must not be null or blank and must meet complexity requirements:
        i. Minimum length of 6 characters and maximum length of 20 characters.
        ii. Must contain at least one uppercase letter, one lowercase letter, and one digit.
    c. Task title length (between 1 and 100 characters).

## 4.2 ASOP (Aspect-Oriented Programming) :

Aspect-Oriented Programming is implemented in both microservices to address cross-cutting concerns such as logging and measuring method execution time.

Two aspects are defined:

- LoggingAspect: Logs method execution before and after invocation.
- MethodExecutionTimeAspect: Measures and logs method execution time.

## 4.3 Microservices:

The microservices architecture follows a distributed design pattern where each service is independent, loosely coupled, and focused on a specific domain. They communicate via RESTful APIs, allowing for scalability and flexibility. Service discovery and registration are achieved using Spring Cloud Netflix Eureka.

## 4.4 Cloud Deployment :

Both microservices are designed to be deployed on a cloud infrastructure. They leverage Spring Cloud dependencies for seamless integration with cloud services. Configuration properties are externalized to allow for easy configuration in cloud environments.

# 5- API documentation

TaskService :

1- POST (CreateTask) :
http://host.docker.internal:8081/api/users/tasks/${userId}
2- GET (GetTasks) :
http://host.docker.internal:8081/api/users/tasks/${userId}
3- PUT(UpdateTitle) :
http://host.docker.internal:8081/api/users/tasks/${userId}/${taskId}
4- PUT(UpdateStatus) :
http://host.docker.internal:8081/api/users/tasks/${userId}/toggle/${taskId}
5- DELETE(DeleteTask) :
http://host.docker.internal:8081/api/users/tasks/${userId}/${taskId}

UserService :
1- POST (SignUp) :
http://host.docker.internal:8080/api/users/signup
2- POST (Login) :
http://host.docker.internal:8080/api/users/login
3- PUT(UpdateEmail) :
http://host.docker.internal:8080/api/users/${userId}/email?newEmail=${newEmail}
4- PUT(UpdatePassword) :
http://host.docker.internal:8080/api/users/${userId}/password?newPassword=${newPassword}

# 6- Conclusion

This Software Design Document provides a detailed overview of Taskaty app withTaskMicroservice and UserMicroservice applications, including their purpose, scope, architecture,APIs, and key functionalities. It serves as a guide for developers and stakeholders involved in the development and deployment of these microservices.