

# Task Manager Bot System Design Document

---

## Executive Summary

Project Type: Web Application

**A task manager bot that manages all tasks**

### Core Features

- CRUD operations
- Task Management
- User Authentication

### Constraints

- Performance
- Cost
- Security
- Compliance
- Time-to-market
- Scalability

User Scale: 100-500 MAU, 500 concurrent users

## System Overview

### Functional Goals

- Manage tasks efficiently
- Provide user authentication
- Integrate with external systems

### Non-Functional Requirements

- High performance

- Scalability
- Security
- Compliance

### Primary User Personas

- End-users
- Administrators

## Architecture Design

The Task Manager bot will be built using a microservices architecture

### System Components

| Component    | Responsibility  | Technologies        | Interfaces |
|--------------|---|---------------------|------------|
| API Gateway  | Handle incoming requests and route them to appropriate services | NGINX, AWS Lambda   | REST API   |
| Task Service | Manage tasks and provide task data to clients                   | Node.js, Express.js | REST API   |
| User Service | Manage user data and authentication                             | Node.js, Express.js | REST API   |

## Database Design

Database Type: Relational database

High availability and durability

### Table: tasks

| Column      | Type     | Nullable | Description        |
|-------------|----------|----------|--------------------|
| id          | int      | False    | Task ID            |
| title       | varchar  | False    | Task title         |
| description | text     | True     | Task description   |
| created_at  | datetime | False    | Task creation date |

### Table: users

| Column   | Type    | Nullable | Description |
|----------|---------|----------|-------------|
| id       | int     | False    | User ID     |
| username | varchar | False    | Username    |
| password | varchar | False    | Password    |

|                   |          |       |                    |
|-------------------|----------|-------|--------------------|
| <b>created_at</b> | datetime | False | User creation date |
|-------------------|----------|-------|--------------------|

### Cost Estimation

| Cost Item          | Monthly Cost | Rationale                             |
|--------------------|--------------|---------------------------------------|
| Compute resources  | \$1000.0     | Estimated cost for compute resources  |
| Storage resources  | \$500.0      | Estimated cost for storage resources  |
| Database resources | \$2000.0     | Estimated cost for database resources |

### Testing & QA Strategy

- Unit testing – All components
- Integration testing – All integrations
- End-to-end testing – All user flows
- Load and stress testing – All components
- Security testing – All components

### Appendices

#### Glossary

- Task Manager bot: A bot that manages tasks
- User authentication: Process of verifying user identity
- External systems: Systems outside of the Task Manager bot

#### References

- AWS documentation
- Kubernetes documentation
- This document is confidential and not for public distribution

### System Architecture & Diagrams

#### System Architecture Diagram (Mermaid Code):

flowchart LR  
User -->|Requests| API\_Gateway

```
API_Gateway --> | Routes | Task_Service
Task_Service --> | Reads/Writes | Task_DB
Task_DB --> Task_Service
Task_Service --> API_Gateway
API_Gateway --> User
```

### **User Flow Diagram (Mermaid Code):**

```
graph TD
    Start --> Login
    Login --> Dashboard
    Dashboard --> ViewTasks
    ViewTasks --> End
```

### **Database ER Diagram (Mermaid Code):**

```
erDiagram
    TASKS {
        int id
        string title
        text description
        datetime created_at
    }
    USERS {
        int id
        string username
        string password
        datetime created_at
    }
    TASKS ||--o{ USERS : belongs_to
```