هيئة تطوير منطقة المدينة المنورة
Al Madinah Region Development Authority

Information Technology **Application unit**

System requirement specification (SRS)

Task Management System

Approved date	0-0-2023
Created by	**Munira Alhumaid – Abeer Asiri – Reuof Alharrah – Rana Kheshaim**
Version	1.0

Table of content

List of Tables

List of Figures

1. Introduction

1.1. Purpose

The application intends to give administrators the ability to assign tasks for employees and allocate those duties to employees.

1.2. Overview

Application that aids in task organization, tracking, and prioritization for MDA management and staff.

1.3. Document organization

The document will be divided into 6 sections, the first section is about the Introduction, here we will discuss what is **Tasker** and the purpose of it. Then we move to section two to be more specific of how the system will work and what it needs to be developed. For section three we will determine the requirements of **Tasker** app for every user. In section four we will administer the architecture of **Tasker** and how it will move from point to another. Furthermore, in section five the UML diagrams will analyze the flow of the actions in **Tasker** app. Finally in section six we will show the API's and DB.

1.4. References

Notion is a web application that gives users the ability to create teams and projects to divide up tasks inside one workspace.

1.5. Nomenclature

Term	Description
Tasker	The application name.
MDA	Medinah Development Authority.

Table 1-1 Nomenclature

2. OVERALL DESCRIPTION

2.1. Product Perspective

The application allows administrators to create new projects, assign them to suitable personnel, track their progress, write notes and comments, and receive notifications when tasks are approaching their due dates.

2.2. System interfaces:

2.2.1. User interfaces

The user interface (UI) serves as the focal point for interaction and communication between humans and a device.

Figure 2-1 Login

Figure 2-2 Login

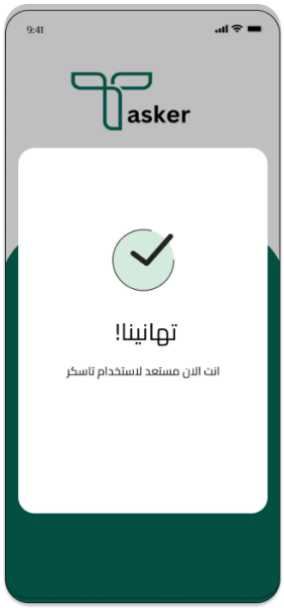

Figure 2-3 Login

Figure 2-4 Change Password

Figure 2-5 OTP code

Figure 2-6 New Password

Figure 2-7 Home Page Manager

Figure 2-8 Manager Statistics

Figure 2-9 Callender

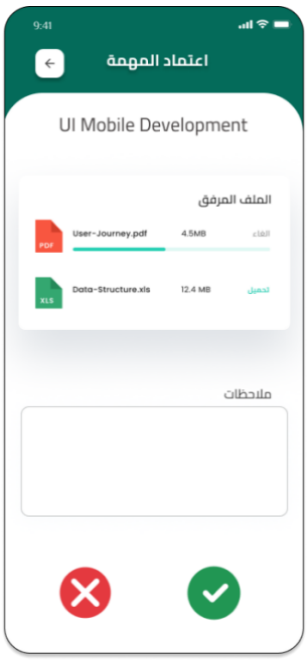

Figure 2-10 Task Approval

Figure 2-11Tasks

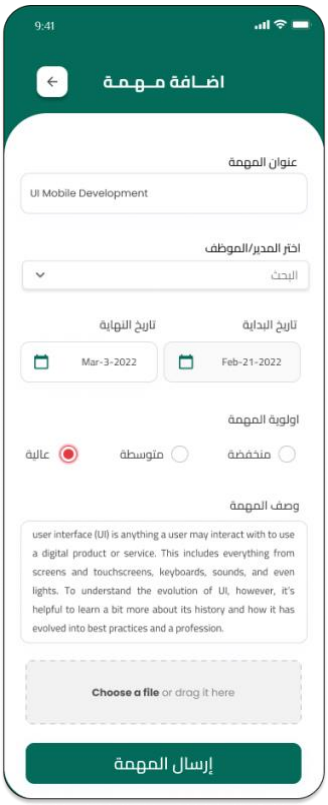

Figure 2-12 Add Task

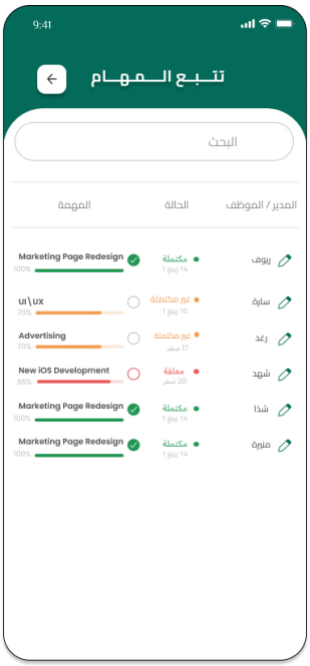

Figure 2-13 Track Task

Figure 2-14 Homepage Employee

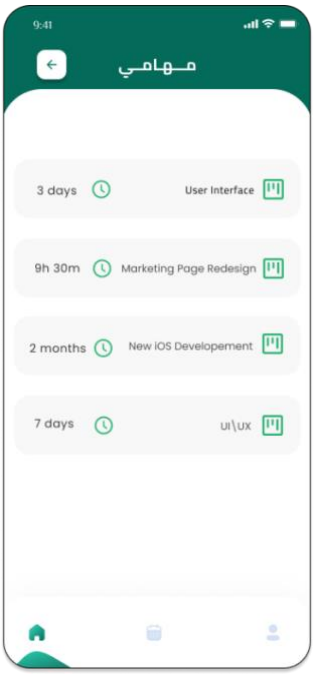

Figure 2-15 Employee Tasks

Figure 2-16 Task details

Figure 2-17 Add Subtasks

Figure 2-18 Submit Task

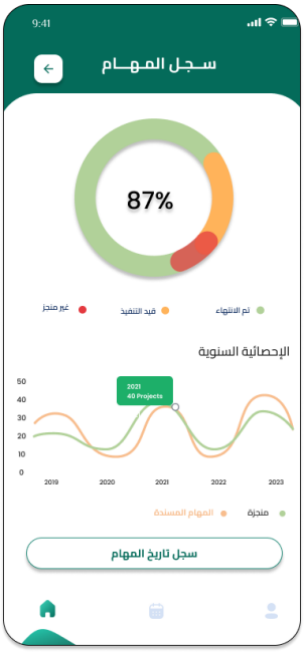

Figure 2-19 Employee Task Statistic Figure 2-20 Manager, Employee

Figure 2-21 Setting Profile

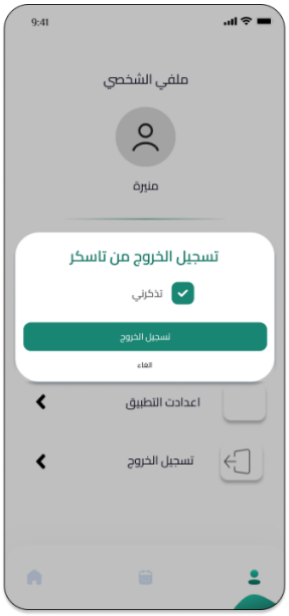

Figure 2-22 Manager, Employee logout

2.2.2. Hardware interfaces
The tasker app does not need any hardware devices.

2.2.3. Software interfaces

Name	Description	Version	Notes
Operating System	IOS, Android		
Database	SQL		
Visual studio code	code editor redefined and optimized for building and debugging modern web and cloud applications.		
Flutter	Programming languages for mobile and web apps.		

Table 2-1 Software Interface

2.3. Memory
Tasker app requires at least 100 MB of RAM of device storage.

2.4. Constraints

- Tasker can’t be used without internet connection.
- It can only be used by MDA users.

2.5. Standards

- We used OWASP top 10 as a secure code standard.

2.6. Risks

Impact:

- High Risks from 7 to 10
- Midum Risks from 4 to 6
- Low Risks from 0 to 3

Possibility:

- High Risks from 7 to 10
- Midum Risks from 4 to 6
- Low Risks from 0 to 3

Risk	Impact	Possibility	Mitigation
System crash	7-10	4-6	Back-up system manually
Lose data	4-6	0-3	Back-up DB
Lose internet connection	4-6	7-10	Cloud-based back-up services

Table 2-2 Risks

3. Requirements

3.1 Functional Requirements for the Manager:

- The user must be able to create new tasks and specify their details such as description, priority, and deadline.
- The user must be able to assign appropriate tasks to specific employees.
- The user must be able to track the progress of assigned tasks to employees and know their current status.
- The user must be able to access task statistics.
- The user must be able to use a calendar to organize daily tasks.
- The user must be able to access their personal profile, change their password, and configure application settings.
- The user must be able to modify tasks for each employee.
- The user must be able to supervise subordinate departments.
- The user must be able to evaluate and approve or reject tasks.

3.2 Functional Requirements for the Employee:

- The user must be able to view their daily tasks and completed tasks.
- The user must be able to see the tasks assigned to them, including details such as description, priority, and deadline.
- The user must be able to update task status.
- The user must be able to add subtasks.
- The user must be able to submit tasks, attach associated files, and record notes and comments.
- The user must be able to access their personal profile, change their password, and configure application settings.
- The user must be able to use a calendar to organize daily tasks.
- The user must be able to view the task history.

3.3 Non-Functional Requirements:

- Usability: An intuitive and user-friendly user interface.
- Security and Confidentiality: Employee permissions must be defined.
- Response Time: Fast performance and immediate response for task loading and updates.
- Flexibility: Ability to expand and adapt to future company needs.

- Compatibility: The system should be compatible with various operating systems and devices.

4. System Architecture

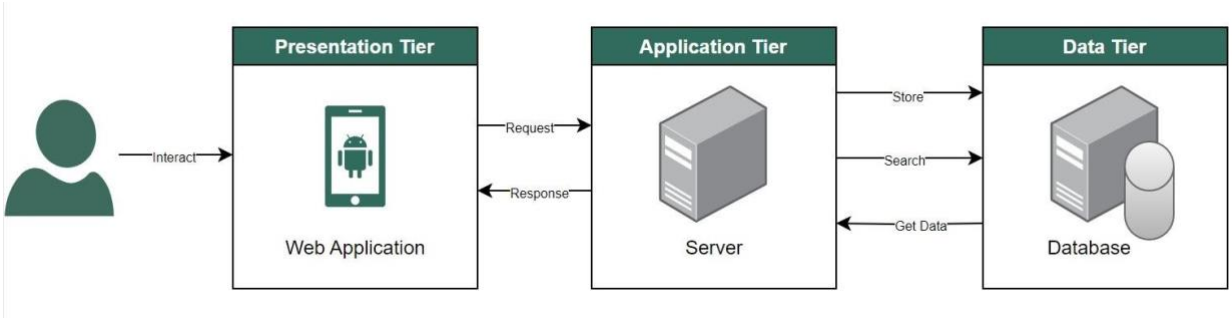

Figure 4-1 System architucture

5. UML (UML ANALYSIS MODELS)

5.1. Use Cases

A use case is a brief description of how a product or system is used to achieve a specific goal or address a particular problem.

5.1.1. Actors -

Manager
- Employee

5.1.2. Use case diagram.

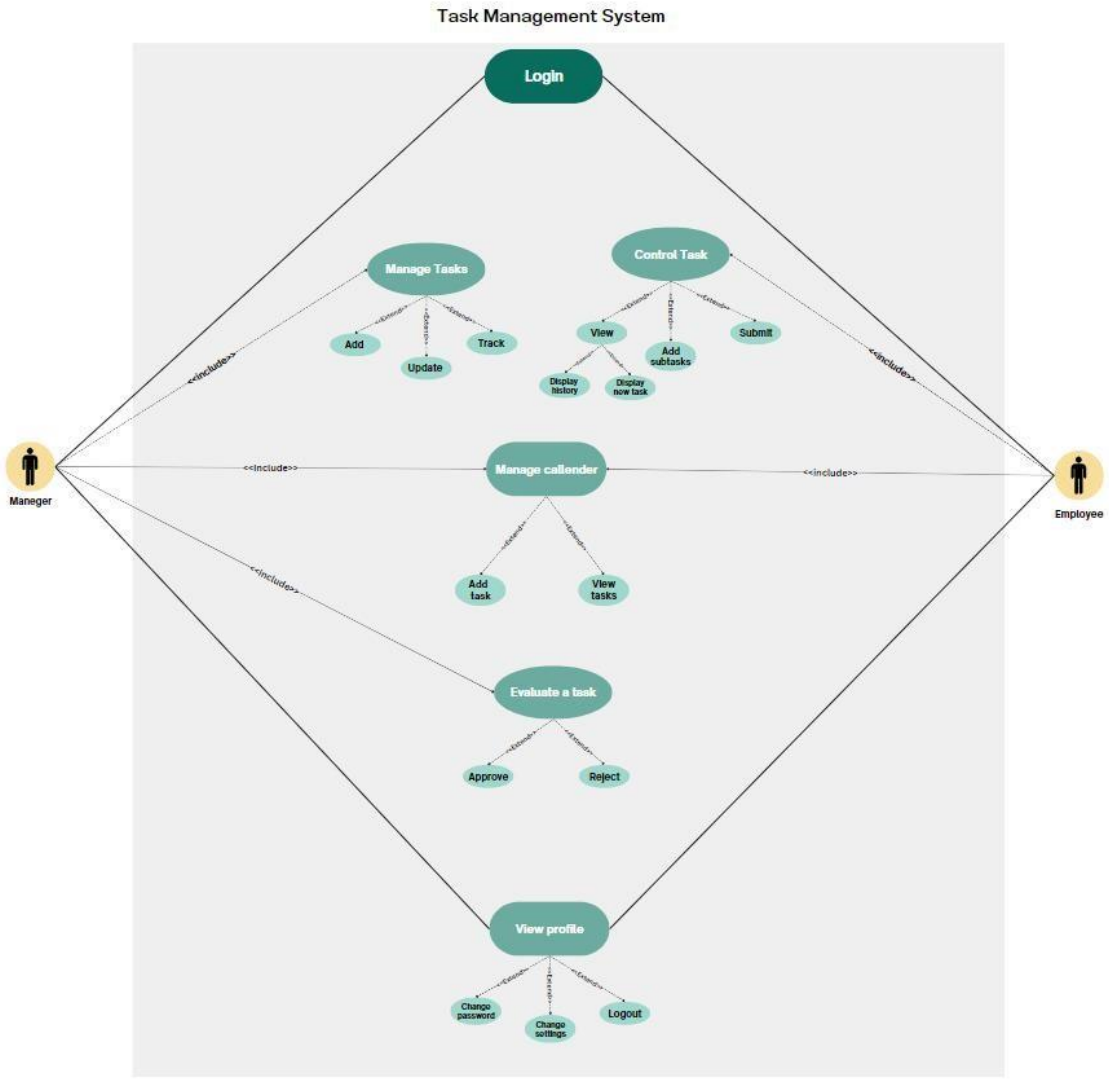

Figure 5-1 Tasker Usecase Diagram

5.1.3. User stories

Use case	Login \logout
Actor	Manager, Employee
Pre-condition	Already exist in the System

Flow of event	The Manger/Employee Login to the application by filling the Email and Password, then the system will authenticate the entered information.

Table 5-1Login, Logout Usecase

Use case	Control tasks
Actor	Employee.
Pre-condition	The Employee login
Flow of event	- The employee can View the tasks. - The employee can Submit the tasks. - The employee can Add subtasks.

Table 5-2Control Tasks

Use case	**View profile**
Actor	Manager, Employee
Pre-condition	Manager/ Employee Login
Flow of event	The Manager and Employee can change password, change settings, logout.

Table 5-3View Profile Usecase

Use case	Manage tasks

Actor	Manager.
Pre-condition	The Manager login
Flow of event	-The manager can Add new task . -The manager can Update the tasks . -The manager can Track the tasks

Table 5-4 Manage tasks

5.2. Class Diagram

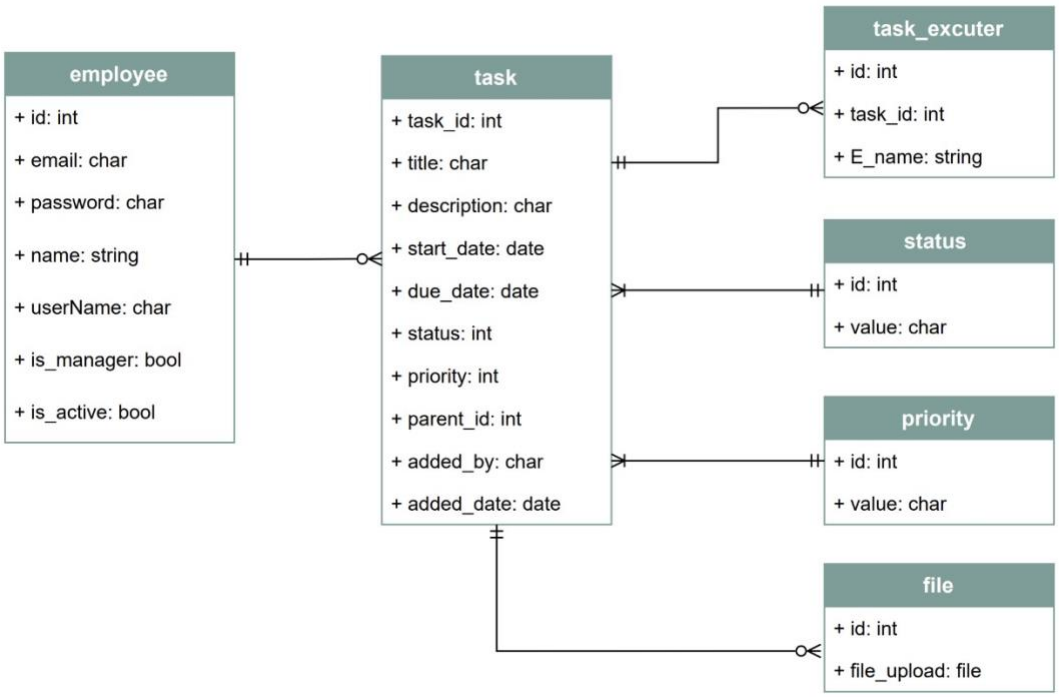

Figure 5-2 : Tsker Class diagram

5.3. Flowchart Diagram

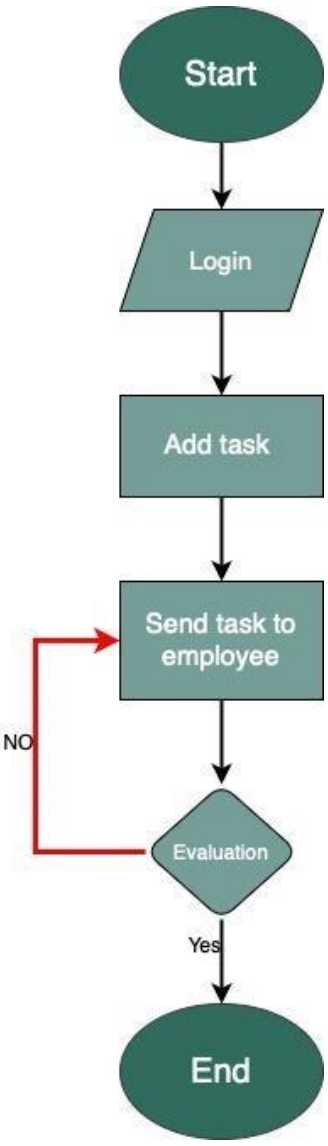

Figure 5-3 Manager flowchart

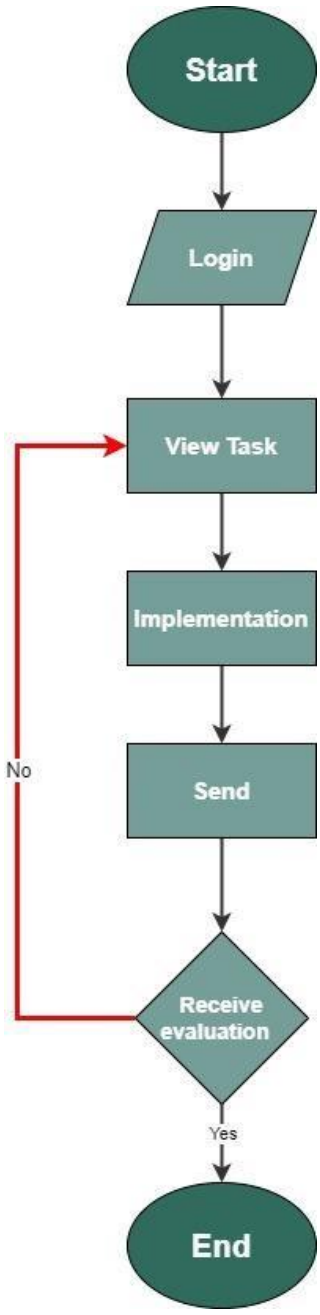

Figure 5-4 Employee flowchart

6. APPENDIX

6.1. Appendix A –APIs

Api

Add Task	
Url	**add_task**
Type	**Post**
Parameter	**Manager_id: int** **Employee_id: int** **title: char** **Start_date: date** **Due_date: date** **Priority: int** **Description: char** **File_upload: file**

Result	{
	message: Task created
	}

Update Task	
Url	**update_task/task_id**
Type	**Put**
Parameter	**Manager_id: int** **Employee_id: int** **title: char** **Start_date: date** **Due_date: date** **Priority: int** **Description: char** **File_upload: file**: [x.x]?

Result	{
	Api status: successful
	Note: null
	
	}

Track Task	
Url	**track_task**
Type	**Get**
Parameter	**Manager_id: int Employee_id: int title: char Status: int Tracking_percent: int**
Result	**{ message: Tracking_percent: ((done_subtask / all_subtasks)*100) }**

Display Task history	
Url	**Display_task_history**
Type	**Get**
Parameter	**Id: int Task_id: int E_name: string**

Result	{
	Api status: successful
	Note: null
	Data:{ id: Employee
	Task_id: Employee
	E_name: Employee}
	
	}

Display New Task	
Url	**Display_new_task**
Type	**Get**
Parameter	**Id: int Task_id: int E_name: string**
Result	**----**

Submit task	
Url	**Submit_task**
Type	**post**
Parameter	**title: char** **Task_id: int** **Description: char** **E_name: string**
Result	**{** **Message: Task submitted** **}**

Add Subtask	
Url	**add_subtask**
Type	**Post**
Parameter	**Task_id: int title:** **char**
Result	**{** **message: subtask added** **}**

Approve	
Url	**Approve_task**
Type	**Post**
Parameter	**Task_id: int title:** **char file_upload:** **file**
Result	**{** **message: Task approved** **}**

Reject	
Url	**Reject_task**
Type	**Post**
Parameter	**Task_id: int title: char file_upload: file**
Result	**{ message: Task rejected. }**

Change settings	
Url	**Change_settings**
Type	**Put**
Parameter	**Personal_information : char Change_password : char Application_settings: char**
Result	**{ Message: Change settings }**

Change password	
Url	**Change_password**
Type	**Put**
Parameter	**Id:int Number_phone:int code_number:int**
Result	**{ Message: Change password }**

Api structure

Api structure	
Api header	Token :”xxxxxxxxxxxxx”

Api structure


```
{
 "actionResult": {
 "status": 0,
 "message": "OK",
 "additionalinformation": null
 },
 "actionData":[
 {
 },
 {
 }]
}
```


Api url + secrets

6.2. Appendix B - database tables

xxxxxx Table

English name	Type	Arabic name	Notes
ID	Guid	الرمز	

6.3. Appendix E -

Project basic information

Project Name	
Scope details	
Classification	
Developed by	

Business Owner	
Beneficiary	
Application end of life	