Tasky

Software Design Document

Sappir Bohbot

Hagai Hen

Bar Goldenberg

Elad Sezenayev

Date: 10/01/23.

Table of Content:

1. Introduction

1.1. Purpose

This Software Design Document describes the architecture and system design of Tasky. We will discuss the software components, interfaces, and data necessary for the implementation phase, as It is the primary refer for code development.

1.2. Scope

The scope of this application includes the ability for managers to create and assign tasks to employees, specify criteria for prioritizing tasks, and automatically prioritize tasks based on the specified criteria.
Employees will be able to view and update their assigned tasks, as well as add sub-tasks for their own organization. Additionally, employees will be required to provide daily updates on their progress.

1.3. Overview

Tasky will provide a user-friendly interface for both managers and employees. Managers will be able to create and assign tasks to their employees and will be able to specify criteria for prioritizing tasks.

The application will automatically prioritize tasks based on the specified criteria and will allow employees to view and update their assigned tasks. Employees will also be able to add sub-tasks for their own organization and will be required to provide daily updates on their progress.

2. System Overview

There are many task management tools currently available in the market, such as Trello, Asana, and Jira. However, many of these tools focus on project management rather than individual task management and lack the specific features and functionality including priority that our application aims to provide.

Tasky is a tool for employees and managers to effectively prioritize and track their tasks, leading to increased productivity and collaboration within teams. The application will allow managers to assign tasks to employees and offer a priority for each task. Employees will be able to view their assigned tasks and update their progress daily.

3. System Architecture

3.1. Architectural Design

The application will be divided into several modules, each with a specific responsibility:

1. User Interface (UI) Module: This module will handle all user interactions and display the necessary information on the screen.

2. Task Management Module: This module will handle the creation, assignment, and tracking of tasks.

3. Data Management Module: This module will handle data storage and retrieval.

4. Communication Module: This module will handle communication between the frontend and backend.

3.2. Decomposition Description

1. The User Interface (UI) Module is responsible for handling all user interactions and displaying the necessary information on the screen. This module communicates with the Task Management Module to retrieve and display information about tasks and progress. Will develop using React.

2. The Task Management Module handles the creation, assignment, and tracking of tasks. This module communicates with the Data Management Module to store and retrieve task data, and with the Authentication and Authorization Module to ensure that only authorized users can perform certain actions.

3. The Data Management Module handles data storage and retrieval using Firebase. This module communicates with the REST API to handle data manipulation and retrieval, and with the Task Management Module to store and retrieve task data.

4. The Communication Module is responsible for handling communication between the frontend and backend, using REST API.

3.3. Design Rationale

The architecture described in the Decomposition Description was chosen for several reasons.

First, by using a modular structure, we can separate the different responsibilities of the system into distinct units, making the application more maintainable and easier to scale in the future.

Second, the use of React.js for the frontend and Node.js for the backend allows for a smooth and responsive user experience, while also providing a stable and efficient server-side environment. In addition, Firebase was chosen for the database as it is a popular database and because NoSQL database is well-suited for handling large amounts of unstructured data.

Another important consideration is the use of REST API as a communication protocol between the frontend and backend, to ensure that the system can easily integrate with other systems, it's a widely adopted standard and it's simple to use.

We considered using other architectures such as Microservices architecture, but it would require more resources and time to implement, and it's not necessary for our application's current requirements.

Overall, the chosen architecture strikes a balance between functionality, scalability, and maintainability, and will allow us to deliver a high-quality task management application that meets the needs of both managers and employees.

4. Data Design

4.1. Data Description

The information domain of the system is transformed into data structures by using Firebase as the database system. The major data or system entities, such as tasks, user profiles, and progress, are stored in the Firebase database. The data is organized in collections.

4.2. Data Dictionary

- User:
  - Fields:
    - email (string): the email address of the user
    - password (string): the password of the user
    - role (string): the role of the user (manager or employee)
    - name (string): the name of the user
    - avatar (string): the user's avatar
  - Methods:
    - login (email:string, password:string): verify the user's email and password
    - signup (user:object): create a new user
    - update (user:object): update the user's information
    - delete (user:object): delete the user's account
    - verifyEmail (email:string): verify if the email already exists
- Task:
  - Fields:
    - title (string): the title of the task
    - description (string): the description of the task
    - dependencies (List of string): list of tasks that must complete before the current task
    - business value (int): the business value of the task (1-5)
    - urgency (int): the business value of the task (1-5)
    - risk reduction (int): the probability that the task can help me reduce risk in future (1-5)
    - development effort (int): the development effort of the task (1-5)
    - assignedTo (string): the email of the employee the task is assigned to
    - assignedBy (string): the email of the manager who assigned the task
    - status (string): the status of the task (pending, in progress, completed)
    - createdAt (date): the creation date of the task
    - updatedAt (date): the date the task was last updated

- Methods:
  - create (task:object): create a new task
  - update (task:object): update an existing task
  - delete (task:object): delete a task
  - view (task:object): view the task's information
  - assign (task:object, employee:object): assign the task to an employee
  - changeStatus (task:object, status:string): change the status of the task
  - viewByAssignee (employee:object): view the tasks assigned to an employee
  - viewByAssigner (manager:object): view the tasks assigned by a manager
  - viewByStatus (status:string): view the tasks by status

5. ***Component Design***

1. Task Creation and Assignment: This component will allow managers to create tasks and assign them to specific employees. The manager will be able to specify criteria for prioritizing tasks, such as urgency, business value, development effort, risk reduction and dependencies.

2. Task Prioritization: The application will automatically prioritize tasks based on the criteria specified by the manager. The prioritized tasks will be displayed in a list for employees to view.

3. Task Management: Employees will be able to view and update their assigned tasks, including marking them as complete or adding sub-tasks for their own organization. They will be able to see the progress of their assigned tasks and the progress of sub-tasks that have been assigned to them.

4. Daily Updates: Employees will be required to provide daily updates on their progress for the tasks that have been assigned to them. This will allow managers to track progress and make adjustments as needed.

5. User Management: This component will allow the manager to add and remove employees from the system and assign them different roles and permissions.

6. Reporting: This component will provide reporting and analytics on the tasks and progress of the employees, it will allow managers to track progress and identify areas that need improvement.

7. User Interface: The application will have a user-friendly interface that is easy to navigate for both managers and employees.

6. ***Requirements Metrix***

ID	***Description***	***Type***	***Priority***	***Info***
1	Managers should be able to create and assign tasks to employees	Functional	High	
2	Managers should be able to specify criteria for prioritizing tasks	Functional	High	
3	Managers should be able to see the sub-task of the employee and the advance in his mission	Functional	Low	
4	The application should automatically prioritize tasks based on the criteria specified by the manager	Functional	High	
5	Employees should be able to view and update their assigned tasks	Functional	High	
6	Employees can add sub-tasks for their own organization	Functional	Low	
7	Employees should be required to provide daily updates on their progress	Functional	Low	
8	The application should provide a clear and user-friendly interface	Non Functional	Low	

9	The application should be secure, with user authentication and data encryption to protect sensitive information	Non Functional	High	
10	The application should be reliable, with minimal downtime and robust error handling	Non Functional	High	
11	The application should be scalable to accommodate a large number of users and tasks	Non Functional	High	