Jaypee Institute of Information Technology, Noida

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING AND INFORMATION TECHNOLOGY

Project Title: MongoDB Task Manager

Enrollment	Name
9921103011	**Yashi Ratnakar**
9921103101	**Disha Purwar**
9921103145	**Rahi Agarwal**

Course Name: Introduction to Large Scale Database Systems
Course Code: 21B12CS314

Program: B. Tech. C.S.E.
3rd Year 6th Sem
2023 – 2024

Acknowledgement

We would like to express our sincere gratitude to all those who contributed to the completion of this project. Firstly, we extend our appreciation to **Dr. Devpriya Soni**, whose guidance and expertise were invaluable in shaping our understanding of task management principles and methodologies.

We are also grateful to the members of our team whose dedication and collaboration were instrumental in the successful execution of this project.Furthermore, we acknowledge the support and encouragement of our colleagues and mentors throughout the duration of this project. Finally, we extend our heartfelt thanks to our families and loved ones for their unwavering support and understanding during this endeavor.

INDEX

Introduction

In today's fast-paced work environments, effective task management is pivotal for ensuring productivity and meeting deadlines. However, reliance on traditional methods like spreadsheets or email often proves inefficient and prone to errors. As organizations expand and tasks become more intricate, there emerges a pressing need for a centralized and efficient task management system. The challenge lies in the lack of a dedicated tool capable of seamlessly tracking tasks, assigning responsibilities, and prioritizing workloads. Without such a system in place, teams face difficulties in staying organized, which can result in missed deadlines, duplicated efforts, and overall decreased productivity.

To address these challenges, organizations require a streamlined and automated solution that simplifies task management processes. By implementing a user-friendly task management platform, teams can enhance collaboration, transparency, and accountability. Such a system enables efficient task tracking, facilitates effective assignment delegation, and provides insights into task statuses and priorities. Ultimately, investing in a robust task management solution empowers organizations to optimize resource utilization, minimize bottlenecks, and foster a culture of productivity and success.

Problem statement

The problem statement highlights the inefficiencies and challenges inherent in traditional task management methods within today's dynamic work environments. One major issue is fragmented communication, where tasks are discussed and assigned through various channels, leading to misunderstandings and difficulty in tracking progress. Additionally, the absence of a structured task prioritization system exacerbates the problem, as teams struggle to identify and focus on high-priority tasks, resulting in delays and resource allocation issues. Moreover, the decentralized nature of traditional methods contributes to limited accountability, making it challenging to assign and track responsibilities within the team. This lack of accountability further compounds the problem of inefficient resource allocation, as managers struggle to allocate resources effectively without clear visibility into team members' workloads, leading to overwork and under-utilization of talents. Furthermore, traditional methods offer limited capabilities for analyzing task performance and team productivity, hindering organizations' ability to make data-driven decisions and optimize workflows. Thus, there is a pressing need for a centralized and efficient task management solution to address these challenges and enhance organizational productivity and collaboration.

Objective:

1. **Centralized Task Management**: Implement a centralized system for tracking tasks, assigning responsibilities, and prioritizing work to streamline communication and enhance coordination within the team.

2. **Efficient Prioritization**: Develop a structured task prioritization system that enables teams to identify and focus on high-priority tasks, thereby reducing delays and optimizing resource allocation.

3. **Enhanced Accountability**: Create mechanisms for assigning and tracking responsibilities to promote accountability and ownership within the team, fostering a culture of transparency and collaboration.

4. **Optimized Resource Allocation**: Provide managers with visibility into team members' workloads to facilitate effective resource allocation, preventing overwork and underutilization of talents.

5. **Data-driven Insights**: Incorporate analytics capabilities to analyze task performance and team productivity, enabling organizations to make informed decisions and optimize workflows.

6. **User-friendly Interface**: Design an intuitive and user-friendly interface that simplifies task management processes and encourages adoption by team members at all levels of the organization.

Requirement Analysis:

To meet the objectives outlined above, the following key features are required:

Task Retrieval: The application should be able to fetch and display all tasks stored in the MongoDB database.

Task Filtering: Users should be able to filter tasks based on various criteria, such as due date, priority, status, assigned user, and tags.

Task Addition: Users should have the ability to add new tasks to the database, providing details such as title, description, due date, priority, assigned user, and tags.

Task Statistics: The application should provide insights into task statistics, such as the count of tasks based on their status (e.g., pending, completed) and the oldest task in the database.

Functionality

The application uses MongoDB, a NoSQL database, for storing and retrieving task data. MongoDB is well-suited for this application due to its flexibility and scalability12.

The application provides the following functionalities:

- **View All Tasks**: Fetches and displays all tasks from the database.
- **View Tasks Due Today**: Fetches and displays tasks that are due today.
- **View Pending Task**s: Fetches and displays tasks that are currently pending.
- **View High Priority Tasks**: Fetches and displays tasks that are marked as high priority.
- **View Task Count by Status**: Fetches and displays the count of tasks grouped by their status.
- **View Oldest Task**: Fetches and displays the oldest task in the database.
- **View Tasks by User**: Fetches and displays tasks assigned to a specific user.
- **View Tasks by Tags**: Fetches and displays tasks that contain specific tags.
- **Add Task**: Allows users to add a new task to the database.

Proposed Approach:

To fulfill the requirements identified in the analysis, the following approach will be adopted:

- **Streamlit Interface**: Utilize Streamlit, a Python library for building web applications, to create an intuitive and interactive user interface for the task management application.

- **MongoDB Database**: Establish a connection to a MongoDB database to store and retrieve task data. Define a suitable data structure for storing tasks, including fields such as title, description, due date, priority, assigned user, tags, status, and creation date.

- **CRUD Operations**: Implement functions to perform CRUD operations (Create, Read, Update, Delete) on tasks within the MongoDB database. These functions will enable tasks to be added, retrieved, updated, and deleted as needed.

- **User Interface Component**s: Develop user interface components using Streamlit for displaying tasks, filtering tasks, adding tasks, and presenting task statistics. These components will include buttons, input fields, selection boxes, and data tables for an intuitive user experience.

- **Application Sections**: Structure the application into distinct sections, including viewing all tasks, filtering tasks, adding tasks, and displaying task statistics. Each section will serve a specific purpose and provide the necessary functionality to users.

- **Testing and Validation**: Thoroughly test the application to ensure functionality, usability, and data integrity. Conduct validation testing to verify that the application meets the specified requirements and performs as expected in various scenarios.

Implementation

```
import streamlit as st
from pymongo import MongoClient
from datetime import datetime, timedelta

# Connect to MongoDB
client = MongoClient("mongodb://localhost:27017/")
db = client["task_manager"]
tasks_collection = db["tasks"]

# Function to fetch all tasks
def get_all_tasks():
    tasks = tasks_collection.find()
    return list(tasks)

# Function to fetch tasks due today
def get_tasks_due_today():
    today = datetime.today()
    tasks_due_today = tasks_collection.find({"due_date": {"$lte": today}})
    return list(tasks_due_today)

# Function to fetch pending tasks
def get_pending_tasks():
    pending_tasks = tasks_collection.find({"status": "pending"})
    return list(pending_tasks)

# Function to fetch high priority tasks
def get_high_priority_tasks():
    high_priority_tasks = tasks_collection.find({"priority": "high"})
    return list(high_priority_tasks)

def get_task_count_by_status():
    task_count_by_status = tasks_collection.aggregate([
        {"$group": {"_id": "$status", "count": {"$sum": 1}}}
    ])
    return list(task_count_by_status)

# Function to get oldest task
def get_oldest_task():
    oldest_task = tasks_collection.find_one({}, sort=[("created_at", 1)])
    return oldest_task

# Function to get tasks assigned to a specific user
def get_tasks_by_user(user_id):
    user_tasks = tasks_collection.find({"assigned_to": user_id})
    return list(user_tasks)

def get_tasks_by_tags(tags):
    tasks_with_tags = tasks_collection.find({"tags": {"$in": tags}})
```

```
    return list(tasks_with_tags)

def get_total_task_count():
    total_task_count = tasks_collection.count_documents({})
    return total_task_count

def add_task(title, description, due_date, priority, assigned_to, tags):
    new_task = {
        "title": title,
        "description": description,
        "due_date": datetime.combine(due_date, datetime.min.time()),  # Convert date to datetime
        "priority": priority,
        "assigned_to": assigned_to,
        "tags": tags,
        "status": "pending",  # Default status
        "created_at": datetime.now()
    }
    tasks_collection.insert_one(new_task)

# Main Streamlit application
def main():
    st.title("Task Manager")

    # Front page with options to perform tasks
    st.write("# Task Manager Dashboard")
    task_options = st.sidebar.radio("Select Task", ["All Tasks", "Tasks Due Today", "Pending Tasks",
                                    "High Priority Tasks", "Task Count by Status",
                                    "Oldest Task", "Tasks by User", "Tasks by Tags",
                                    "Add Task"])

    if task_options == "All Tasks":
        st.write("## All Tasks")
        if st.button("Fetch All Tasks"):
            tasks = get_all_tasks()
            st.write(tasks)

    elif task_options == "Tasks Due Today":
        st.write("## Tasks Due Today")
        if st.button("Fetch Tasks Due Today"):
            tasks_due_today = get_tasks_due_today()
            st.write(tasks_due_today)

    elif task_options == "Pending Tasks":
        st.write("## Pending Tasks")
        if st.button("Fetch Pending Tasks"):
            pending_tasks = get_pending_tasks()
            st.write(pending_tasks)

    elif task_options == "High Priority Tasks":
        st.write("## High Priority Tasks")
```

```
        if st.button("Fetch High Priority Tasks"):
            high_priority_tasks = get_high_priority_tasks()
            st.write(high_priority_tasks)

    elif task_options == "Task Count by Status":
        st.write("## Task Count by Status")
        if st.button("Fetch Task Count by Status"):
            task_count_by_status = get_task_count_by_status()
            st.write(task_count_by_status)

    elif task_options == "Oldest Task":
        st.write("## Oldest Task")
        if st.button("Fetch Oldest Task"):
            oldest_task = get_oldest_task()
            st.write(oldest_task)

    elif task_options == "Tasks by User":
        user_id = st.text_input("Enter User ID:")
        if user_id and st.button("Fetch Tasks by User"):
            user_tasks = get_tasks_by_user(user_id)
            st.write("## Tasks by User")
            st.write(user_tasks)

    elif task_options == "Tasks by Tags":
        tags = st.text_input("Enter Tags (comma-separated):")
        if tags and st.button("Fetch Tasks by Tags"):
            tags = [tag.strip() for tag in tags.split(",")]
            tasks_with_tags = get_tasks_by_tags(tags)
            st.write("## Tasks by Tags")
            st.write(tasks_with_tags)

    elif task_options == "Add Task":
        st.write("## Add Task")
        with st.form(key="add_task_form"):
            title = st.text_input("Title")
            description = st.text_area("Description")
            due_date = st.date_input("Due Date")
            priority = st.selectbox("Priority", ["Low", "Medium", "High"])
            assigned_to = st.text_input("Assigned To")
            tags = st.text_input("Tags (comma-separated)")
            submit_button = st.form_submit_button("Add Task")

            if submit_button:
                add_task(title, description, due_date, priority, assigned_to, tags)
                st.success("Task added successfully!")

    st.sidebar.write("Total Tasks:", get_total_task_count())

if _name_ == "_main_":
    main()
```

Results

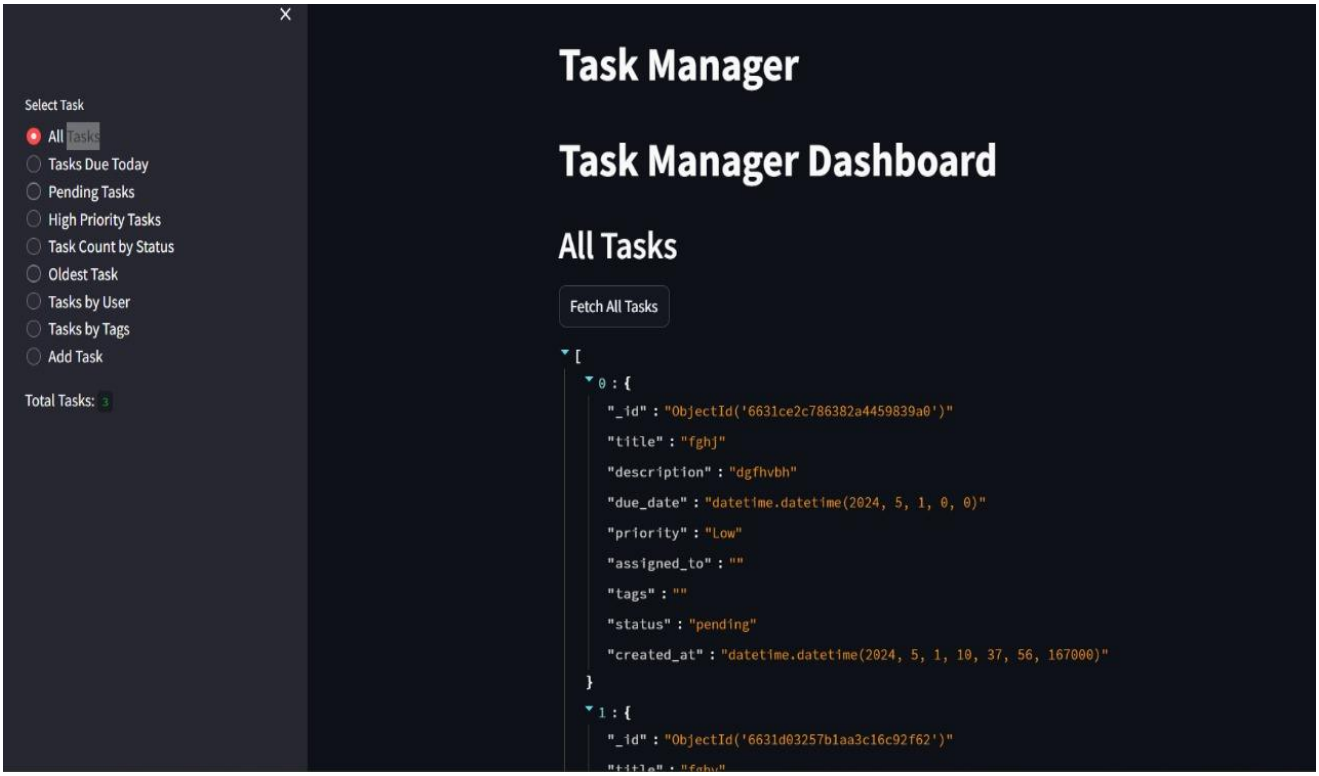

Fig 1: All Tasks

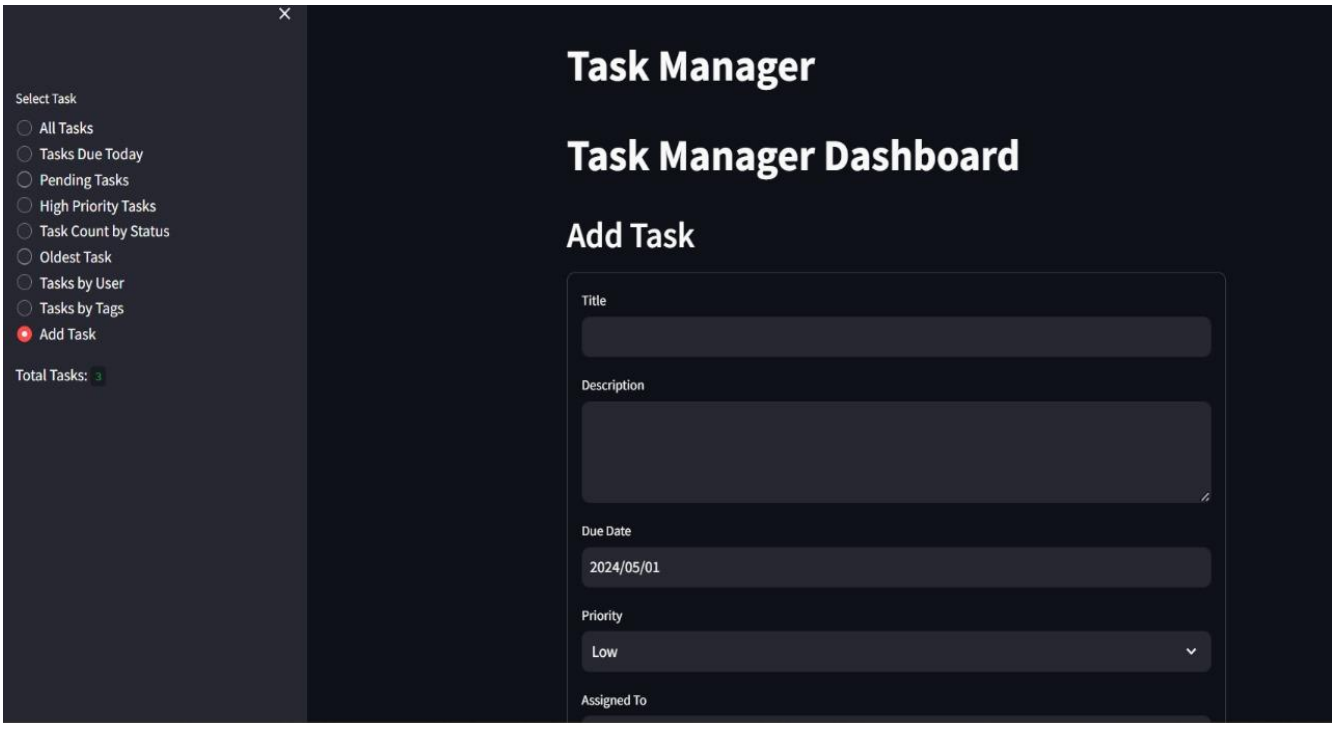

Fig 2: Add Task

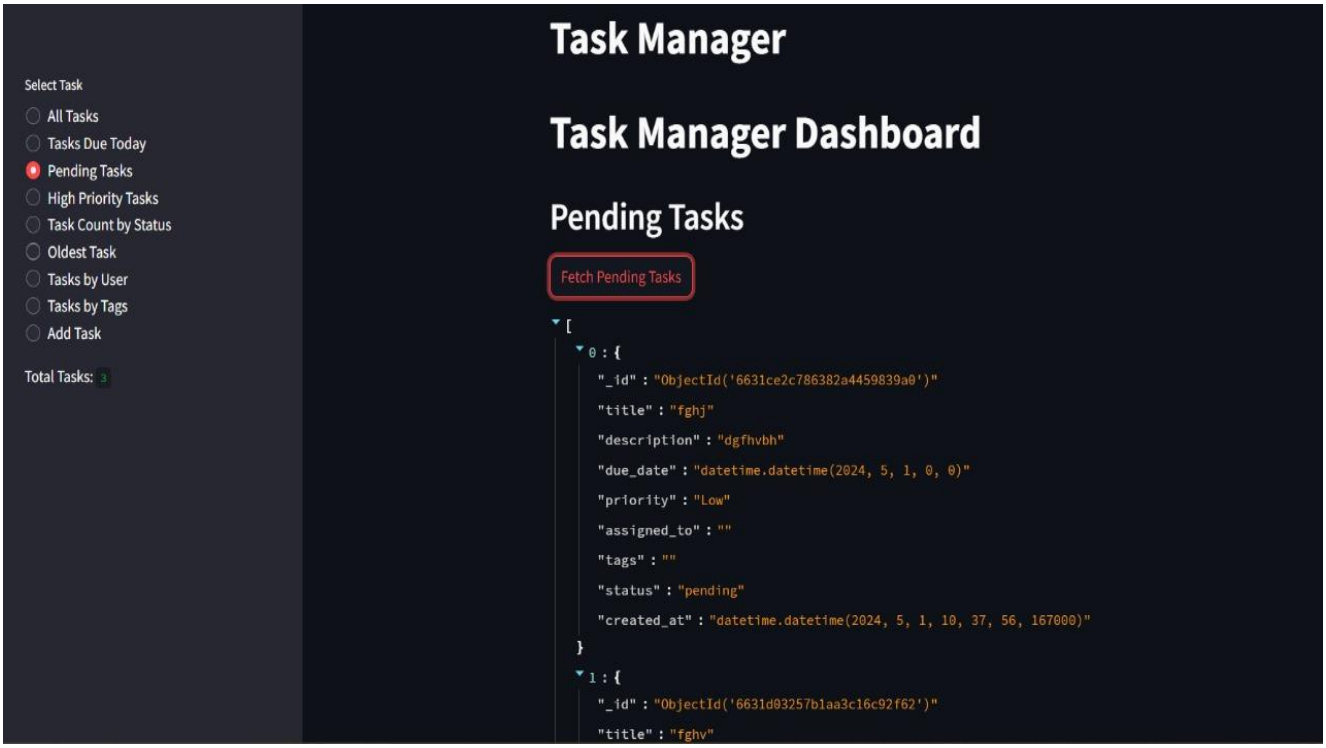

Fig 3: Pending Task

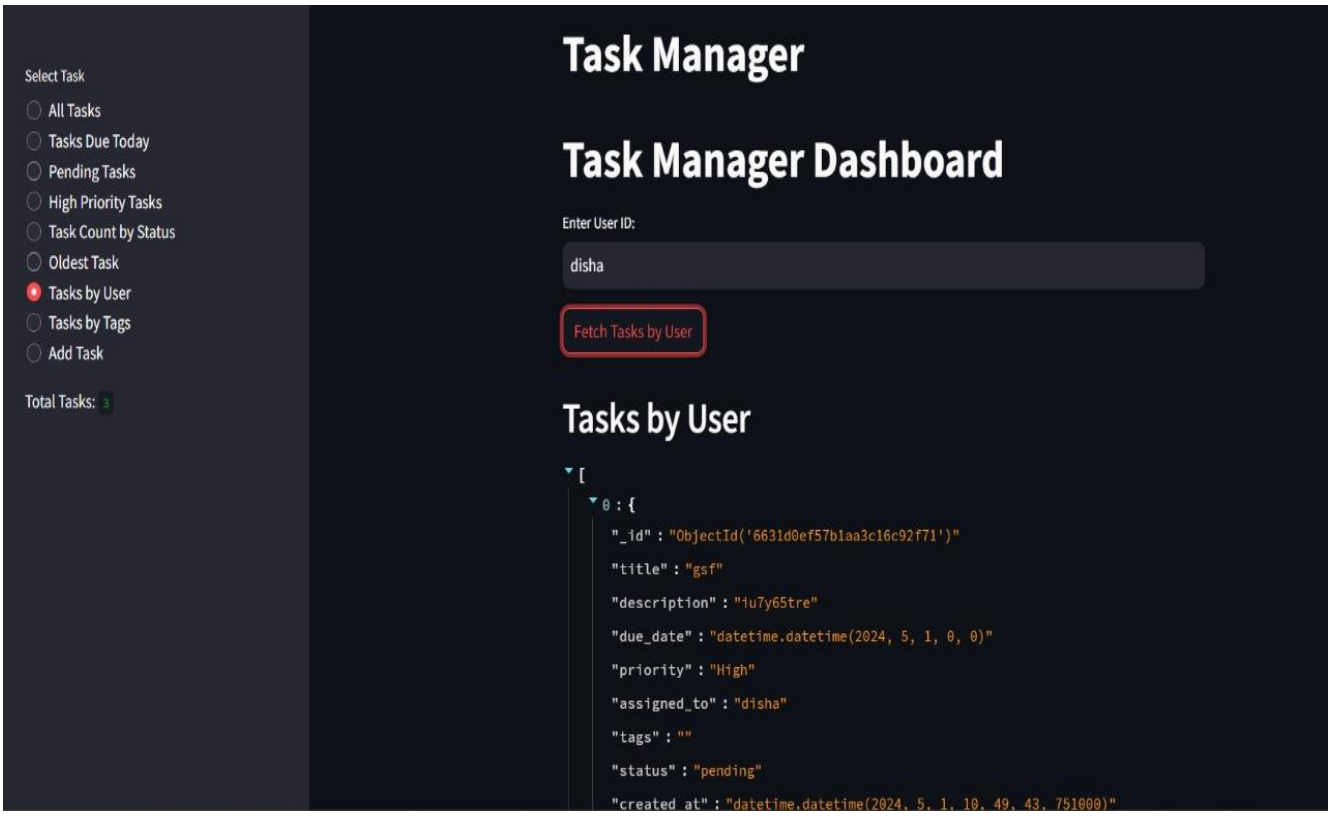

Fig 4: Tasks By User

Findings:

- **Inefficient Communication**: Traditional methods lack a centralized system for task management, leading to fragmented communication and difficulty in tracking task progress.

- **Lack of Prioritization**: Without a structured task prioritization system, teams struggle to identify and focus on high-priority tasks, resulting in delays and resource allocation issues.

- **Limited Accountability**: The absence of a dedicated task management tool makes it challenging to assign and track responsibilities, leading to a lack of accountability within the team.

- **Ineffective Resource Allocation**: Managers face difficulties in allocating resources effectively due to limited visibility into team members' workloads, resulting in overworked individuals and underutilized talents.

- **Lack of Insights and Analysis**: Traditional methods offer limited capabilities for analyzing task performance and team productivity, hindering organizations' ability to make data-driven decisions and optimize workflows.

Conclusion:

In conclusion, the findings presented here strongly advocate for the urgent adoption of a centralized, efficient, and automated task management solution by organizations. The imperative to address the challenges posed by traditional methods becomes glaringly apparent, necessitating a paradigm shift towards modernized task management approaches. Such a comprehensive solution holds the promise of not only streamlining communication but also revolutionizing task prioritization methodologies. Moreover, it stands poised to bolster accountability within teams, thereby fostering a culture of ownership and responsibility. Additionally, the benefits extend to optimizing resource allocation, a critical aspect often plagued by the lack of visibility into team workloads. Through the implementation of advanced task management tools, organizations stand to gain invaluable insights into task performance and team productivity, enabling them to make informed decisions and drive efficiency. Thus, investing in a robust task management system emerges not merely as an option but as an imperative for organizations striving to navigate the complexities of the modern business landscape successfully. By embracing the transformative power of modern task management tools, organizations can empower their teams to operate at peak efficiency, collaborate seamlessly, and ultimately realize their objectives with unparalleled ease and effectiveness.

References:

[1] Teamwork. (2022). The 7 Biggest Challenges of Traditional Task Management. [Online]. Available: https://www.teamwork.com/blog/traditional-task-management-challenges/.

[2] Zapier. (n.d.). The Ultimate Guide to Task Management. [Online]. Available: https://zapier.com/learn/task-management/.

[3] Asana. (n.d.). What Is Task Management? Definition and Best Practices. [Online]. Available: https://asana.com/guide/task-management/what-is-task-management.

[4] Trello. (n.d.). Task Management: A Complete Guide for Managers and Teams. [Online]. Available: https://trello.com/guide/task-management.

[5] Wrike. (n.d.). Task Management: Definition, Process, Tools, and Best Practices. [Online]. Available: https://www.wrike.com/blog/task-management-definition-process-tools-best-practices/.