

Project Report

On

To-Do List Application

Submitted by :

Iragavarapu Datta Eswara Abhinay Venkata Sai	213837102013
Kovvuri Dharma Reddy	213837102018
Pampana Vara Prasad	213837146071

Acknowledgement

I would like to express my sincere gratitude to everyone who has contributed to the development and completion of this MERN stack to-do list app. Without their support, this project would not have been possible.

First and foremost, I would like to thank IIDT Blackbucks, whose guidance, expertise, and support have been invaluable throughout the entire development process.

I am deeply thankful to my supervisor Srinivas sir, for their continuous encouragement, insightful feedback, and unwavering support, which have significantly enriched the quality of this project.

I would also like to express my gratitude to the open-source community for providing the tools, frameworks, and resources that have been instrumental in the development of this project.

TABLE OF CONTENTS

ABSTRACT	4
1. INTRODUCTION	5
1.1 PROJECT AIMS AND OBJECTIVES	5
1.2 BACKGROUND OF PROJECT	5
2. SYSTEM ANALYSIS	6
2.1 SOFTWARE REQUIREMENT SPECIFICATION	7
2.3 SOFTWARE TOOL USED	8
3. SYSTEM DESIGN	9
3.1 TABLE DESIGN	10
3.2 DATA FLOW DIAGRAM'S	12
4. SYSTEM IMPLEMENTATION	14
4.1 MODULE DESCRIPTION	15
4.2 SCREENSHOTS	16
5. SYSTEM TESTING	19
5.1 UNIT TESTING	19
5.2 INTEGRATION TESTING	20
6. CONCLUSION & FUTURE SCOPE	21
7. REFERENCES	22

Abstract :

The TaskNexa is a web-based application developed to help users manage their tasks by providing a user-friendly interface for adding, editing, and deleting tasks. The app also supports task filtering based on status (All, Pending, Completed) which helps users to sort the tasks effectively.

At its core, our TaskNexa Application simplifies task organization by allowing users to create, edit, and categorize tasks according to their importance and urgency. Moreover, customizable reminders and notifications help users stay on track, reducing the likelihood of missed deadlines and overlooked responsibilities.

TaskNexa employs robust security measures to protect your data. Using advanced encryption and secure server protocols, we ensure that your information remains safe and confidential at all times. You can trust TaskNexa to store your data securely, allowing you to focus on managing your tasks without worry.

Key Features:

- **User-Friendly Interface:** TaskNexa boasts a simple and intuitive interface, allowing users to manage tasks with ease. From creating new tasks to setting deadlines and priorities, every feature is designed for seamless navigation and minimal learning curve.
- **Task Customization:** Users can customize task lists, categorize tasks into different groups, and set reminders to stay organized. TaskNexa offers flexibility in managing tasks, empowering users to structure their workflow according to their preferences.
- **Real-Time Updates:** TaskNexa provides real-time updates and notifications, keeping users informed about upcoming deadlines, task completions, and any changes made to their task lists. This ensures users stay on top of their responsibilities without missing important deadlines.
- **Secure Data Storage:** The application employs robust backend architecture to ensure data security and integrity. TaskNexa stores task information securely, utilizing encryption and authentication mechanisms to safeguard sensitive data, providing users with peace of mind.
- **Search Functionality:** Quickly find specific tasks using keywords for easy retrieval.
- **Seamless Task Export:** TaskNexa offers a seamless task export feature, allowing users to effortlessly export their task lists in various formats such as CSV or PDF. This functionality enables users to share their tasks with others or integrate them into other tools or platforms.

CHAPTER 1

INTRODUCTION

This chapter gives an overview about the aim , objectives ,background and operation environment of the system.

1.1 PROJECT AIMS AND OBJECTIVES

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

- Real-Time Updates:
- Efficient Task Management
- Task Customization
- Secure Data Storage

1.2 Background of Project

The to-do list application aims to streamline task management processes by providing users with a digital platform to organize and track their tasks. Inspired by the need for efficient task management in modern workflows, the application seeks to enhance productivity and organization for individuals and teams alike.

By leveraging technology, the application aims to offer features such as search functionality, task downloading, and integration with external resources, providing users with a comprehensive task management solution. The inclusion of an user accounts further enhances the application's usability by allowing for efficient content management and administration of various tasks.

In summary, the to-do list application aims to empower users to manage their tasks effectively, thereby improving productivity and organization in their daily lives.

CHAPTER 2

SYSTEM ANALYSIS

The system analysis phase involves a comprehensive examination of the requirements, functionalities, and constraints of the to-do list application. This phase aims to identify and define the system's scope, objectives, and user needs to guide the development process effectively.

2.1 Requirements Gathering:

During the requirements gathering phase, various techniques such as interviews, surveys, and brainstorming sessions were utilized to gather input from stakeholders and potential users. The primary focus was on understanding the key features and functionalities desired in the to-do list application, including task management, search capabilities, user authentication, and administrative functions.

2.2 Functional Requirements:

Based on the gathered requirements, the following functional requirements were identified for the to-do list application:

- User registration and login: Allow users to create accounts and authenticate securely.
- Task creation and management: Enable users to create, edit, prioritize, and delete tasks.
- Search functionality: Implement a search feature to facilitate quick retrieval of tasks based on keywords or categories.
- Task downloading: Provide users with the option to download their task lists for offline access.
- Admin panel: Create an administrative interface for managing tasks, users, and application settings.

2.3 Non-Functional Requirements:

In addition to functional requirements, non-functional requirements were considered to ensure the performance, security, and usability of the application. These include:

- Performance: Ensure fast response times and scalability to accommodate increasing user loads.
- Security: Implement robust authentication and authorization mechanisms to protect user data and prevent unauthorized access.
- Usability: Design an intuitive and user-friendly interface to enhance user experience and adoption.

2.4 Constraints:

Constraints such as time, budget, and technology limitations were taken into account during the system analysis phase. These constraints influenced decisions regarding the scope, features, and implementation approach of the to-do list application.

2.5 System Scope:

The system scope encompasses the functionalities and features to be included in the initial release of the to-do list application. It defines the boundaries of the project and provides a clear understanding of what will be delivered to users.

2.4 SOFTWARE TOOLS USED

The whole Project is divided in two parts the front end and the backend.

Frontend Technologies:

- **HTML (HyperText Markup Language):** HTML provides the foundational structure for the to-do list application's user interface. It defines the layout and content elements such as task lists, buttons, and input fields.
- **CSS (Cascading Style Sheets):** CSS styles the HTML elements of the to-do list application, controlling aspects such as colors, fonts, spacing, and layout. It ensures a visually appealing and consistent user experience across different devices and screen sizes.
- **JavaScript:** JavaScript adds interactivity and dynamic behavior to the to-do list application's frontend. It handles tasks such as form validation, task manipulation, and asynchronous requests to the server, enhancing the user experience and functionality.
- **React.js:** React.js serves as the core frontend library for the to-do list application, enabling the creation of reusable UI components and efficient state management. It allows for the dynamic rendering of task lists, real-time updates, and seamless interaction with the user interface.

Backend Technologies:

- **Node.js:** Node.js powers the backend of the to-do list application, allowing for server-side JavaScript execution. It handles tasks such as user authentication, task storage, and API endpoints, providing a robust foundation for backend development.
- **Express.js:** Express.js, built on top of Node.js, serves as the backend framework for the to-do list application. It provides a minimalist and flexible structure for building RESTful APIs and handling HTTP requests, enabling seamless communication between the frontend and backend components.
- **MongoDB:** MongoDB serves as the backend database for the to-do list application, storing task data in a flexible and scalable NoSQL format. It allows for efficient querying, updating, and retrieval of task information, ensuring smooth and reliable operation of the application.

CHAPTER 3

SYSTEM DESIGN

VARIOUS TABLES TO MAINTAIN INFORMATION

➤ todos Table from Database

The screenshot shows the MongoDB Compass interface. On the left, the database structure is visible, including the 'todoapp' database and the 'todos' collection. The main panel displays the 'todos' collection with 8 documents. The first document is expanded, showing the following fields:

```
{
  "_id": ObjectId("661bb6c2ebc8c569a37525f2"),
  "username": "User",
  "todoData": "Buy groceries",
  "time": "17:30",
  "completed": false,
  "expired": false,
  "targetDate": "2024-04-14T10:57:47.623+00:00",
  "date": "2024-04-14T10:57:47.624+00:00",
  "__v": 0
}
```

The second document is also expanded, showing the following fields:

```
{
  "_id": ObjectId("661bb6c2ebc8c569a37525f3"),
  "username": "User",
  "todoData": "Finish report for work",
  "time": "21:30",
  "completed": false,
  "expired": false,
  "targetDate": "2024-04-14T10:58:10.789+00:00",
  "date": "2024-04-14T10:58:10.790+00:00",
  "__v": 0
}
```

The third document is also expanded, showing the following fields:

```
{
  "_id": ObjectId("661bb702ebc8c569a37525f9"),
  "username": "User",
  "todoData": "Pay bills",
  "time": "14:20",
  "completed": true,
  "expired": false,
  "targetDate": "2024-04-14T10:59:14.816+00:00",
  "date": "2024-04-14T10:59:14.816+00:00",
  "__v": 0
}
```

➤ users Table from Database

▼ config

▼ local

startup_log ...

▼ todoapp

todos

users ...

Document modified. CANCEL UPDATE

```
1 _id: ObjectId('661b5120e710708b97f959f3')
2 username: "smith"
3 email: "smith@email.com"
4 password: "$2b$10$gEkx/iCP3ceKqpsa7x/hxOmFiUUNkXyq07u24nLfRAZRyggT0yqb0g"
5 notes: 0
6 todos: 5
7 todosCompleted: 3
8 date: 2024-04-14T03:44:32.771+00:00
9 __v: 0
```

ObjectId
String
String
String
Int32
Int32
Int32
Date
Int32

Document modified. CANCEL UPDATE

```
1 _id: ObjectId('661b771451e1247e1f430fea')
2 username: "david"
3 email: "david@davidmail.com"
4 password: "$2b$10$u0f017ZHVfGmMSIYSWrUQ.tPpo0hLT4NVau2j4LbTmXKswfSnGmG2g"
5 notes: 0
6 todos: 0
7 todosCompleted: 0
8 date: 2024-04-14T06:26:28.304+00:00
9 __v: 0
```

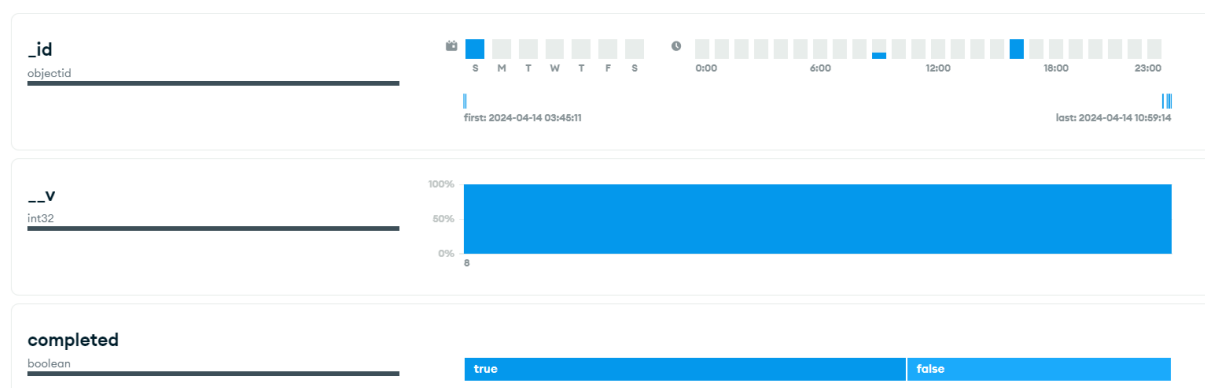
ObjectId
String
String
String
Int32
Int32
Int32
Date
Int32

Document modified. CANCEL UPDATE

```
_id: ObjectId('661bb5faebc8c569a37525dd')
username: "User"
```

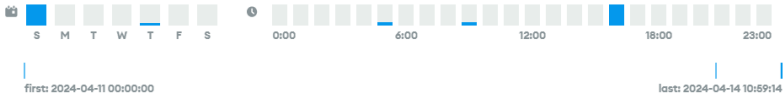
```
1 _id: ObjectId('661b771451e1247e1f430fea')
2 username: "david"
3 email: "david@davidmail.com"
4 password: "$2b$10$u0f017ZHVfGmMSIYSWrUQ.tPpo0hLT4NVau2j4LbTmXKswfSnGmG2g"
5 notes: 0
6 todos: 0
7 todosCompleted: 0
8 date: 2024-04-14T06:26:28.304+00:00
9 __v: 0
```

Sample Schema



date

date



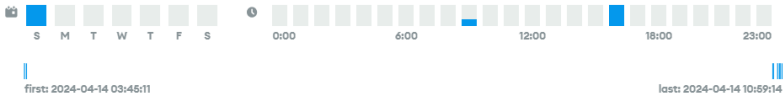
expired

boolean



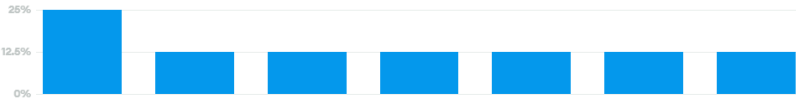
targetDate

date



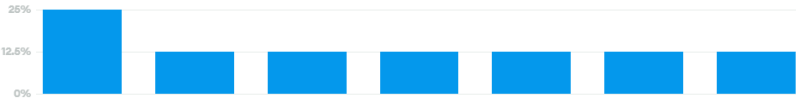
time

string



todoData

string



notes

int32



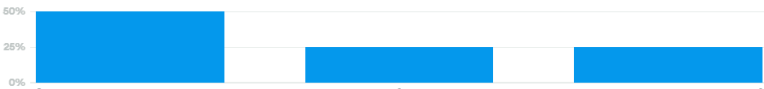
password

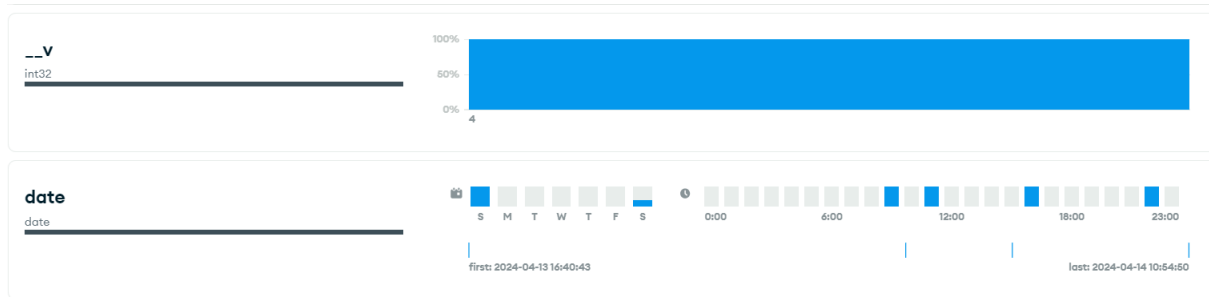
string



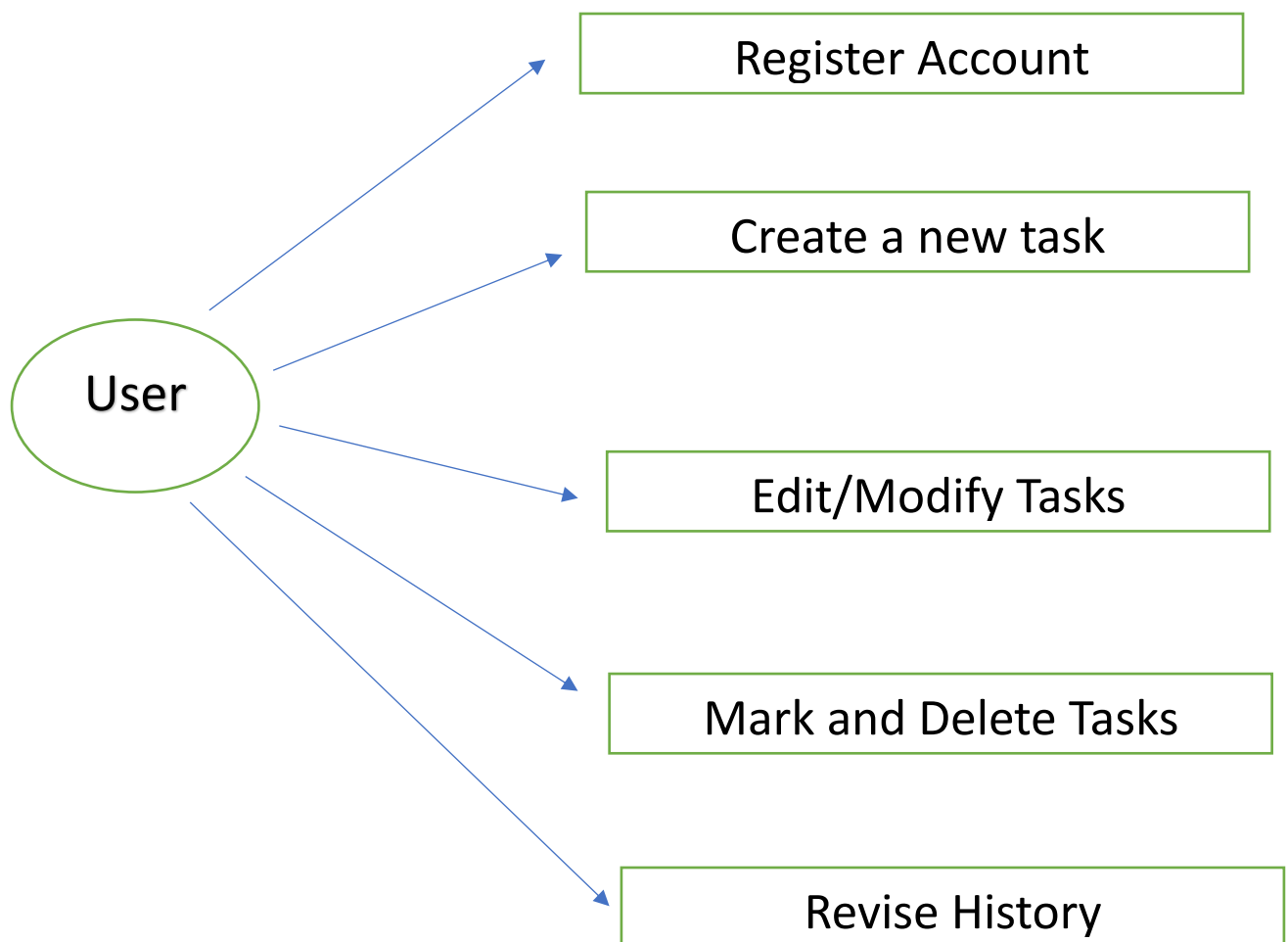
todos

int32





USE CASE DIAGRAM FOR USER



After accessing the homepage of the to-do list application, users can select the "User Login" option to proceed. Upon selecting this option, they are prompted to enter their username and password. If the provided credentials are valid, the user will be granted access to the to-do list application's dashboard, where they can view and manage their tasks.

This process ensures that only authenticated users can access the application's features, maintaining the security and integrity of the user's task data. Additionally, the login functionality provides a personalized experience for each user, allowing them to access their own set of tasks and preferences within the application.

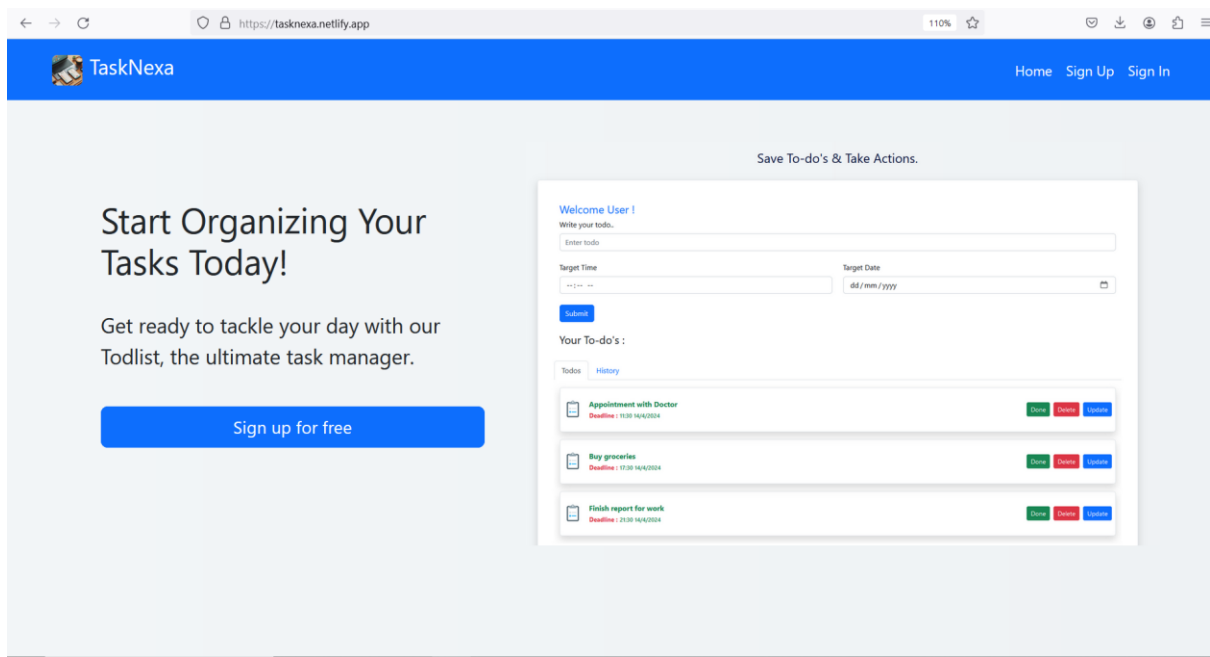
Upon successful authentication, users are directed to their personalized dashboard within the to-do list application. Here, users can view their existing tasks, create new tasks, edit task details, mark tasks as completed, and delete tasks as needed. The dashboard provides a centralized hub for users to manage their to-do lists efficiently, helping them stay organized and productive in their daily tasks and activities.

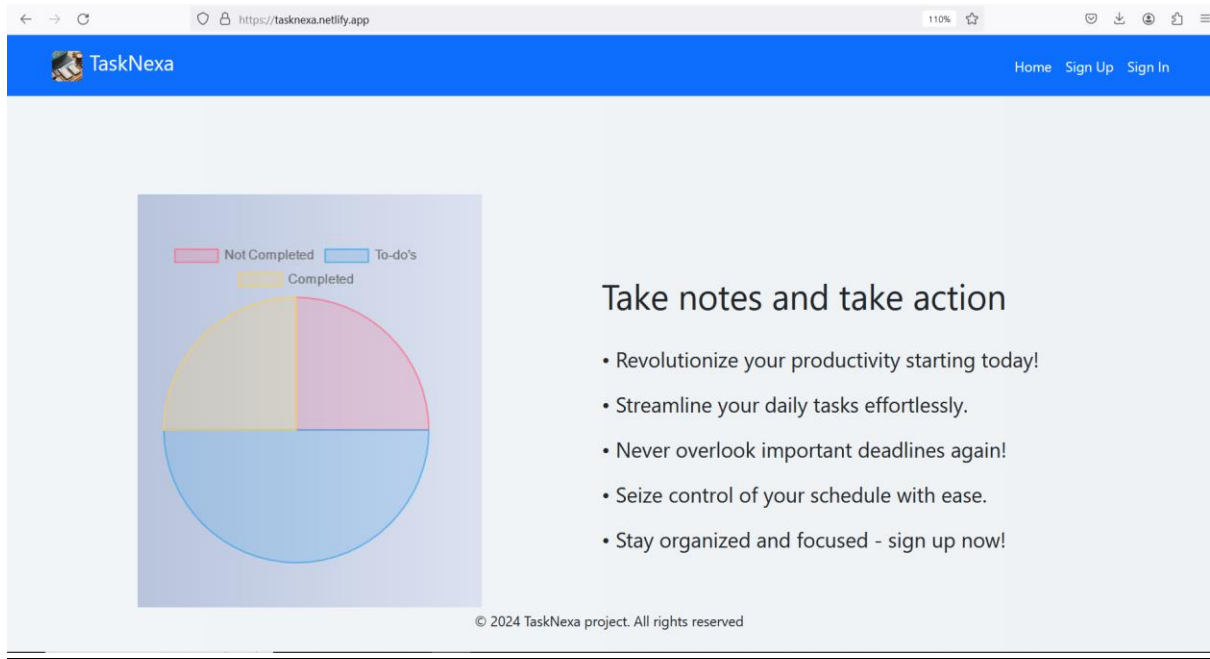
Furthermore, the user login feature enhances the overall user experience of the to-do list application by allowing users to maintain their task lists securely. By requiring authentication before granting access to the application's features, the login functionality ensures that users' task data remains confidential and protected from unauthorized access. This not only instills trust and confidence in users but also reinforces the reliability and professionalism of the application.

CHAPTER 4

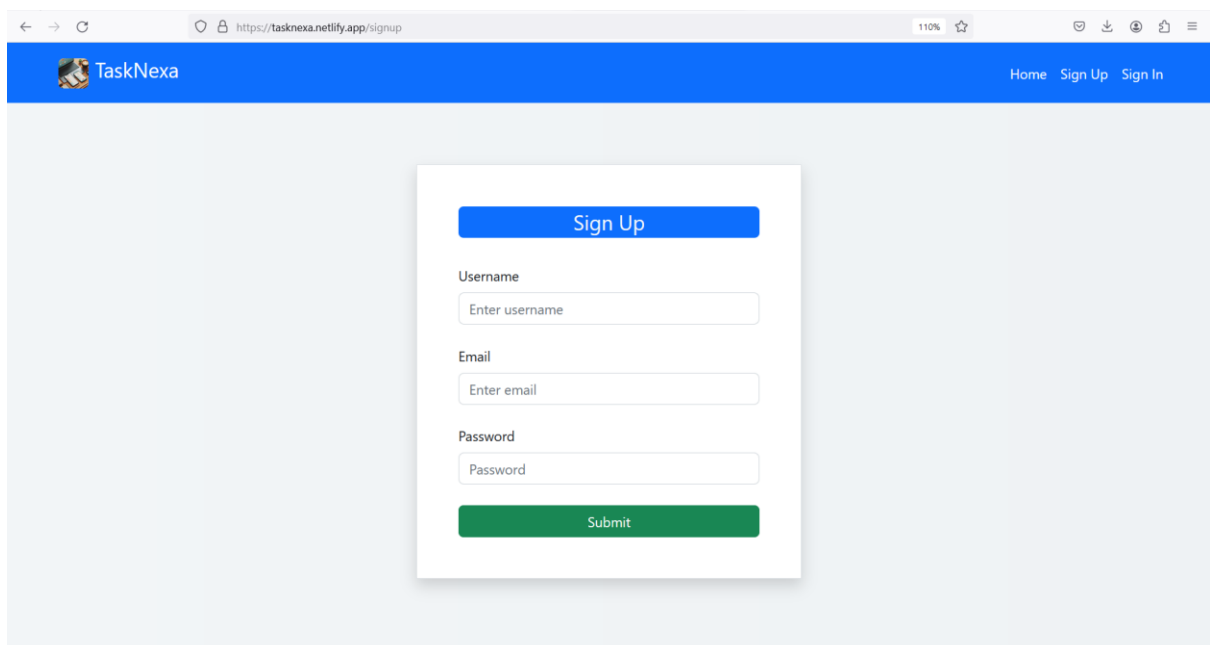
SYSTEM IMPLEMENTATION

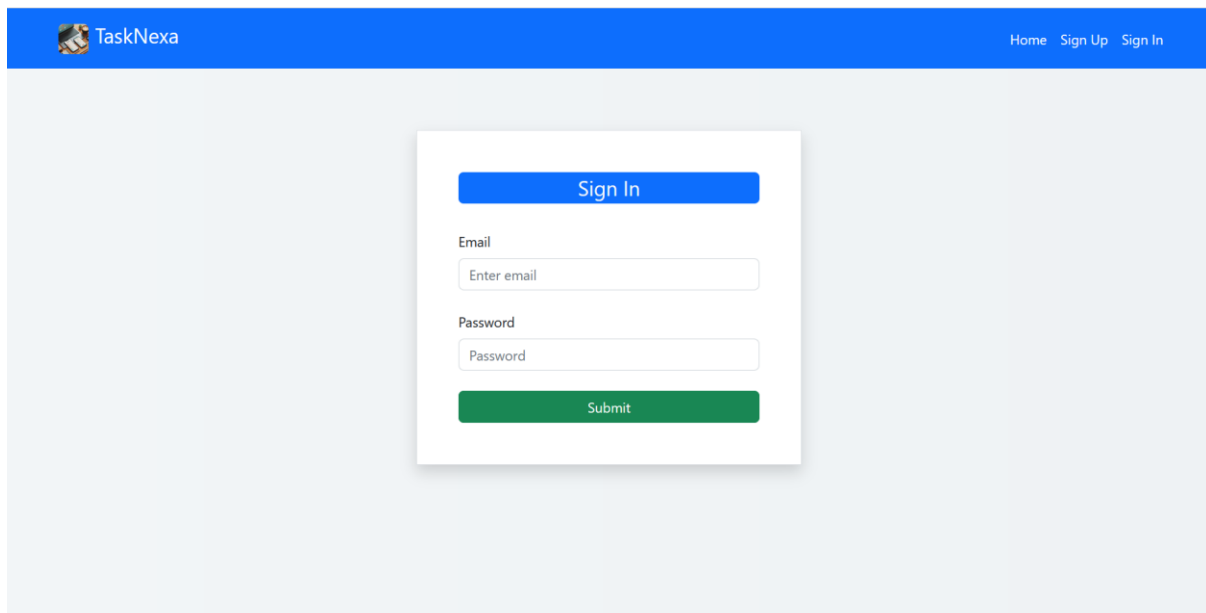
4.1 Screenshot for Homepage :





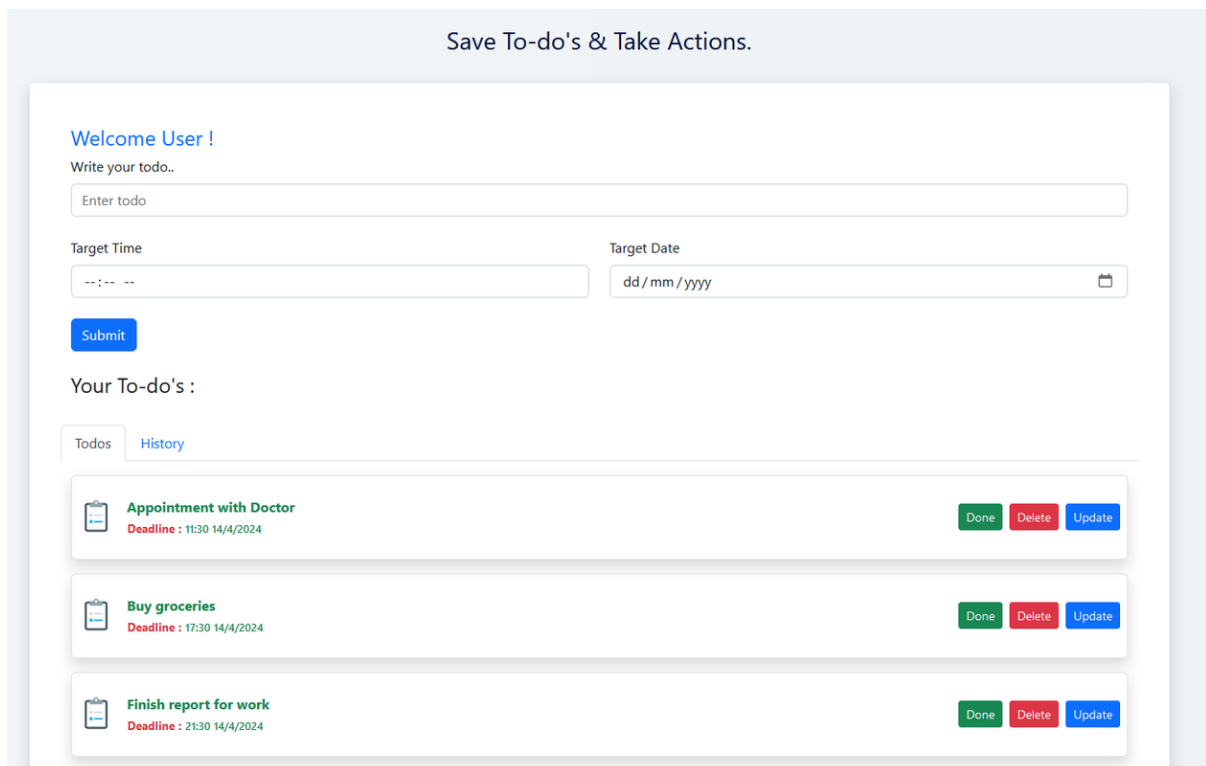
4.2 Screenshot of Sign-up and Sign-In Pages :





The screenshot shows the TaskNexa Sign In interface. At the top, there is a blue header with the TaskNexa logo on the left and navigation links 'Home', 'Sign Up', and 'Sign In' on the right. The main content area is light gray and features a white sign-in form in the center. The form has a blue 'Sign In' button at the top, followed by an 'Email' input field with the placeholder 'Enter email', a 'Password' input field with the placeholder 'Password', and a green 'Submit' button at the bottom.

4.3 Screenshot of Task Management Interface :



The screenshot displays the Task Management Interface with the heading 'Save To-do's & Take Actions.' The interface is divided into two main sections. The top section, 'Welcome User!', includes a text input for 'Write your todo..' and a 'Submit' button. Below this are fields for 'Target Time' (with a time picker) and 'Target Date' (with a date picker). The bottom section, 'Your To-do's:', has two tabs: 'Todos' and 'History'. The 'Todos' tab is active, showing a list of three tasks. Each task card includes a clipboard icon, the task name, a deadline, and three action buttons: 'Done' (green), 'Delete' (red), and 'Update' (blue).

Task Name	Deadline	Action Buttons
Appointment with Doctor	11:30 14/4/2024	Done, Delete, Update
Buy groceries	17:30 14/4/2024	Done, Delete, Update
Finish report for work	21:30 14/4/2024	Done, Delete, Update

4.4 Screenshot of Completed History of User Tasks:

TaskNexa

Todos Profile

Save To-do's & Take Actions.

Welcome User !

Write your todo..

Enter todo

Target Time

Target Date

dd / mm / yyyy

Submit

Your To-do's :

Todos History

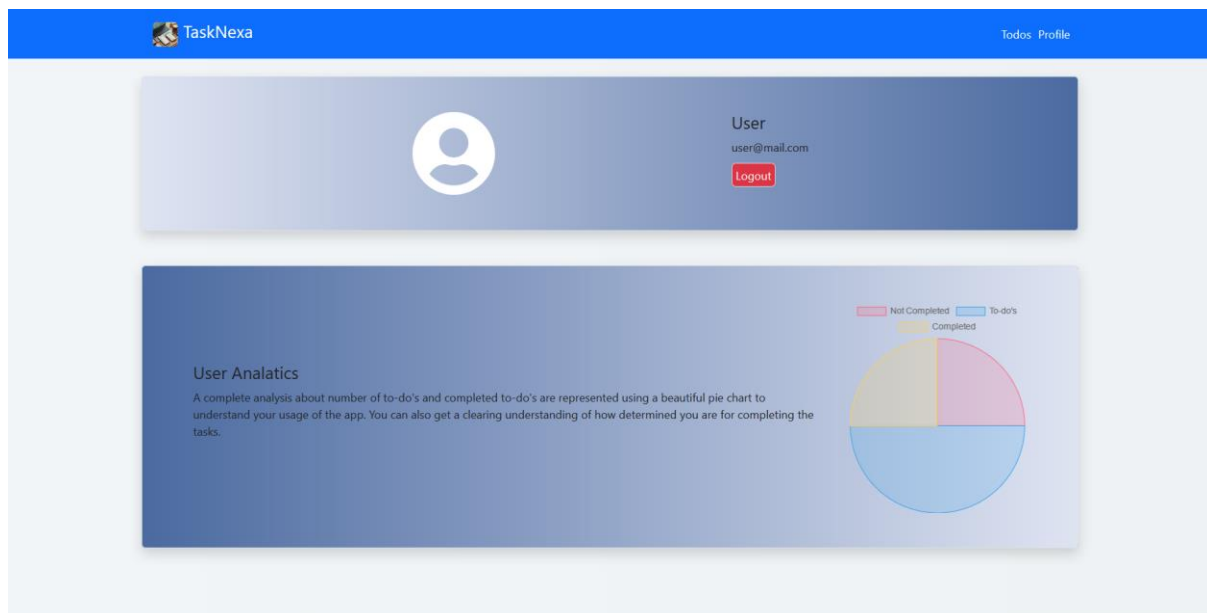
✓ Appointment with Doctor
11:30

Completed

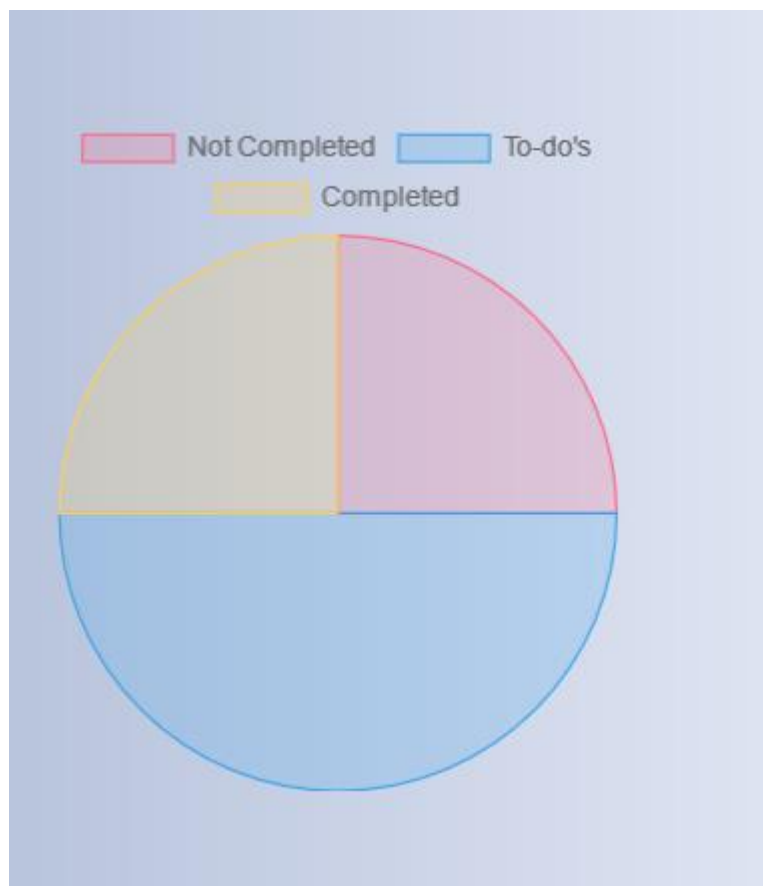
✓ Pay-bills
14:20

Completed

4.5 Screenshot of User Profile and Analytics :



4.6 Detailed Screenshot of User Analytics :



CHAPTER 5

SYSTEM TESTING

The aim of the system testing process was to determine all defects in our project .The program was subjected to a set of test inputs and various observations were made and based on these observations it will be decided whether the program behaves as expected or not. Our Project went through two levels of testing.

- 1.Unit testing
- 2.Integration testing

5.1 UNIT TESTING

Unit testing is undertaken when a module has been created and successfully reviewed. In order to test a single module we need to provide a complete environment ie besides the module we would require

- The procedures belonging to other modules that the module under test call
- Non local data structures that module accesses
- A procedure to call the functions of the module under test with appropriate Parameters

Unit testing was done on each and every module that is described under module description of chapter 4

1. Test for Adding Tasks

- The unit testing phase validated the addition of tasks to the Todo List application. It successfully handled scenarios such as adding tasks to an empty list and including tasks with special characters. However, the test uncovered a vulnerability as the application failed to prevent the addition of tasks with empty descriptions.

2. Marking Tasks as Complete

- Testing confirmed the functionality of marking tasks as complete within the Todo List app. It effectively managed tasks both individually and in bulk. Nonetheless, the absence of handling non-existing tasks surfaced as an area requiring attention, as the application did not gracefully handle such cases.

3. Deleting Tasks

- Unit tests affirmed the deletion of tasks from the Todo List application. The app exhibited proficiency in removing tasks, whether singularly or in batches. Yet, it exposed a flaw in error handling, as the application faltered when attempting to delete tasks that did not exist in the list. Addressing this issue will enhance the robustness of the deletion feature.

5.2 INTEGRATION TESTING

In this type of testing we test various integration of the project module by providing the input. The primary objective is to test the module interfaces in order to ensure that no errors are occurring when one module invokes the other module.

Integration testing for the Todo List application assessed the interaction between its components to ensure seamless functionality. The testing phase encompassed scenarios where different modules of the app collaborated, including adding, marking, and deleting tasks. Through integration testing, the application's ability to handle these operations in conjunction was validated. However, potential vulnerabilities in data consistency and flow between modules were identified, highlighting the need for further refinement to guarantee smooth interaction and overall performance.

CHAPTER 6

CONCLUSION & FUTURE SCOPE

The Todo List application represents a significant step forward in task management, offering users a digital platform to organize and prioritize their activities efficiently. While the current version fulfills basic functionalities admirably, there exists a compelling vision for its future development.

This application presents a digital solution for task management, benefiting users by streamlining organization and productivity. It centralizes the task management process online, allowing users to add, edit, and delete tasks effortlessly. The platform offers a user-friendly interface where individuals can create accounts, manage their tasks, and track progress efficiently. Furthermore, it incorporates features such as task categorization, deadline setting, and priority levels to enhance task organization and prioritization.

Moreover, a commitment to ongoing innovation and responsiveness to user feedback will ensure that the Todo List app remains a vital tool for individuals seeking to optimize their productivity and organization in an increasingly fast-paced world.

As a future prospect, the application can evolve to include additional functionalities such as task collaboration, real-time updates, and integration with other productivity tools, thereby offering a comprehensive solution tailored to meet diverse user needs effectively.

CHAPTER 7

REFERENCES

https://www.w3schools.com/html/html_intro.asp

<https://getbootstrap.com/docs/5.3/getting-started/introduction/>

https://www.w3schools.com/css/css_intro.asp/

https://www.w3schools.com/js/js_datatypes.asp

https://www.w3schools.com/nodejs/nodejs_intro.asp

https://www.w3schools.com/nodejs/nodejs_mongodb.asp

<https://www.mongodb.com/docs/>

<https://expressjs.com/en/5x/api.html>

<https://www.geeksforgeeks.org/express-js/>

<https://react.dev/learn>