Project Report

On

“Task Management System using Notes”

**Submitted by:**  **Guided by:**

Mrs. Sonal Fatangare

**1. Vishvaraj Deshmukh** (Roll no: 18 )

**2. Snehal Chougule** (Roll no: 15 )

**3. Anishka Ahuja** (Roll no: 3 )



DEPARTMENT OF COMPUTER ENGINEERING

R.M.D Sinhgad School of Engineering ,Warje, Pune-58

Savitribai Phule Pune University

2024-25



DEPARTMENT OF COMPUTER ENGINEERING

R.M.D Sinhgad School of Engineering ,Warje, Pune-58

Savitribai Phule Pune University

**CERTIFICATE**

This is to certify that **“TASK MANAGEMENT SYSTEM USING NOTES”** embodies the original work done by **Vishvaraj Deshmukh, Snehal Chougule, Anishka Ahuja** during this project submission as a partial fulfillment of the requirement for the Mini Project in subject Database Management System of Third year Computer Engineering students of Savitribai Phule Pune University during the academic year 2024-2025

**Date:**

**Place:** Pune

Project Guide Head of Department Principal

(Mrs. Sonal Fatangare) (Dr. Vina M. Lomte) (Dr.V.V. Dixit)

**ACKNOWLEDGEMENT**

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are grateful to our project guide Mrs. Sonal Fatangare for the guidance, inspiration and constructive suggestions that helpful us in the preparation of this project. We also thank our other staff members who have helped in successful completion of the project.

Vishvaraj Deshmukh

Snehal Chougule

Anishka Ahuja

Contents

|  |  |  |
| --- | --- | --- |
| Sr. No | Title | Page No. |
| 1 | Introduction   * 1. Objective   2. Problem Definition | 5  6  7 |
| 2 | Scope and Limitations | 8-9 |
| 3 | Requirement Analysis | 12 |
| 4 | System Requirements | 12-13 |
| 5 | ER Diagram | 13-14 |
| 6 | ER to table conversion | 14 |
| 7 | Normalization | 15-16 |
| 8 | Schema Diagram | 16-17 |
| 9 | Coding | 18-20 |
| 10 | Screenshots | 20-22 |
| 11 | Future Scope and Enhancement | 22-24 |
| 12 | Conclusion | 25 |
| 13 | References | 26 |

1. **INTRODUCTION**

Project Name: Task Management System using Notes

Platform: MongoDB Atlas

Programming Language: NodeJS ,ExpressJS , EJS

Database: MongoDB

Project Type: Mini project

* 1. **Objectives:**

The primary objective of this project, **Task Management System using Notes**, is to develop a web-based platform that enables users to efficiently manage tasks by creating, updating, and organizing notes. The project aims to achieve the following:

1. **Streamline Task Management**: To provide users with an easy-to-use interface to create, edit, update and delete notes, helping them organize tasks efficiently.
2. **Ensure Data Integrity**: Used MongoDB as the backend database to store and retrieve notes securely and reliably, ensuring that data is well-organized and quickly accessible.
3. **Enable Flexibility and Scalability**: Designed a system that can be extended to include more features in the future, such as task deadlines, categories, or user-specific notes, making it adaptable for future enhancements.
4. **Provide Real-Time Interaction**: Ensure the system allows users to make real-time updates and see the results immediately, offering an intuitive and responsive user experience.
5. **Secure Access Control**: Implement a basic user authentication system to control access to notes, ensuring that only authorized users can view or manipulate the data.

**1.2 Problem Definition:**

To address challenges like Disorganized Task Tracking, Lack of Accessibility, Data Insecurity this project aims to create a Task Management System that allows users to create, edit, organize, and delete tasks in a secure and accessible manner, with the flexibility to scale as needed.

In today’s fast-paced environment, individuals and teams need an efficient way to organize and manage tasks. Traditional paper-based methods or scattered digital tools make it difficult to maintain and prioritize tasks effectively. Users often struggle with:

1. **Disorganized Task Tracking**: Without a centralized system, managing multiple tasks becomes cumbersome, leading to lost or forgotten tasks, missed deadlines, and reduced productivity.
2. **Lack of Accessibility**: Many task management tools are not easily accessible across different platforms, leading to inconsistency in tracking and managing tasks when switching between devices.
3. **Inefficient Collaboration**: Collaborating on tasks becomes difficult without a system that allows users to share and modify notes in real-time, especially when working on joint projects.
4. **Data Insecurity**: Storing notes and tasks without proper security measures exposes sensitive information to unauthorized access, making it crucial to have secure access control.

**2. Scope & Limitations**

**Scope of Project:**

The Task Management System using Notes is designed to offer users a convenient and efficient platform for organizing and managing their tasks. The scope of the project includes the following:

1. **User-Friendly Interface**: The system will provide a simple and intuitive web interface for users to create, view, update, and delete notes associated with tasks.
2. **Task Management**: Users can create notes with task details such as titles and descriptions, allowing for easy tracking and organization of their work.
3. **Real-Time Operations**: The system will handle real-time operations like updating and deleting tasks instantly, providing immediate feedback to users.
4. **Data Storage in MongoDB**: Notes will be securely stored in the MongoDB database, allowing for scalable, efficient, and fast data retrieval and management.
5. **Future Enhancements**: The project is designed to be flexible, with the potential for future extensions like adding task prioritization, deadlines, and multi-user support.

**Limitations:**

1. The current version of the system supports only basic task management features (CRUD operations for notes) and is designed for individual users. Collaborative features may be added in future updates.
2. While there are no basic security features and it does not cover advanced encryption or multi-factor authentication for user data.

**3. Requirement Analysis**

**3.1 Requirement gathering and analysis:**

The requirement gathering process involved identifying and documenting the needs of users for the Task Management System using Notes. This was achieved through various methods, including interviews, surveys, and feedback from potential users.

**3.2 Functional Requirements:**

These requirements define what the system should do and the specific functionalities it must provide:

1. Create Notes:
   * Users should be able to add new notes by providing a title and description.
   * The system must store these notes in the MongoDB database.
2. Read/View Notes:
   * Users must be able to view a list of all their created notes.
   * Each note should display its title and description.
3. Update Notes:
   * Users should have the ability to edit existing notes.
   * The system should allow changes to the title and description of the selected note.
4. Delete Notes:
   * Users should be able to delete any note they have created.
   * A confirmation prompt should be displayed before deletion to prevent accidental loss of data.
5. Search Functionality (Future Implementation):
   * Implement a search feature to allow users to find notes quickly based on keywords in titles or descriptions.

**3.3 Non-Functional Requirements:**

These requirements define the quality attributes and constraints of the system:

1. Usability:
   * The interface must be user-friendly, with a clear layout and easy navigation for users of all technical levels.
2. Performance:
   * The system should be responsive, providing quick feedback to users (e.g., immediate display of notes after actions).
3. Security:
   * User data must be protected through secure authentication mechanisms, and sensitive information should not be exposed.
4. Scalability:
   * The system should be designed to accommodate an increasing number of notes and users without significant performance degradation.
5. Maintainability:
   * The codebase should be structured and documented to facilitate future updates and enhancements.
6. Reliability:
   * The system should be robust, with minimal downtime, ensuring users can access their notes at any time.

**4. System Requirements**

**H/w and S/w requirement: -**

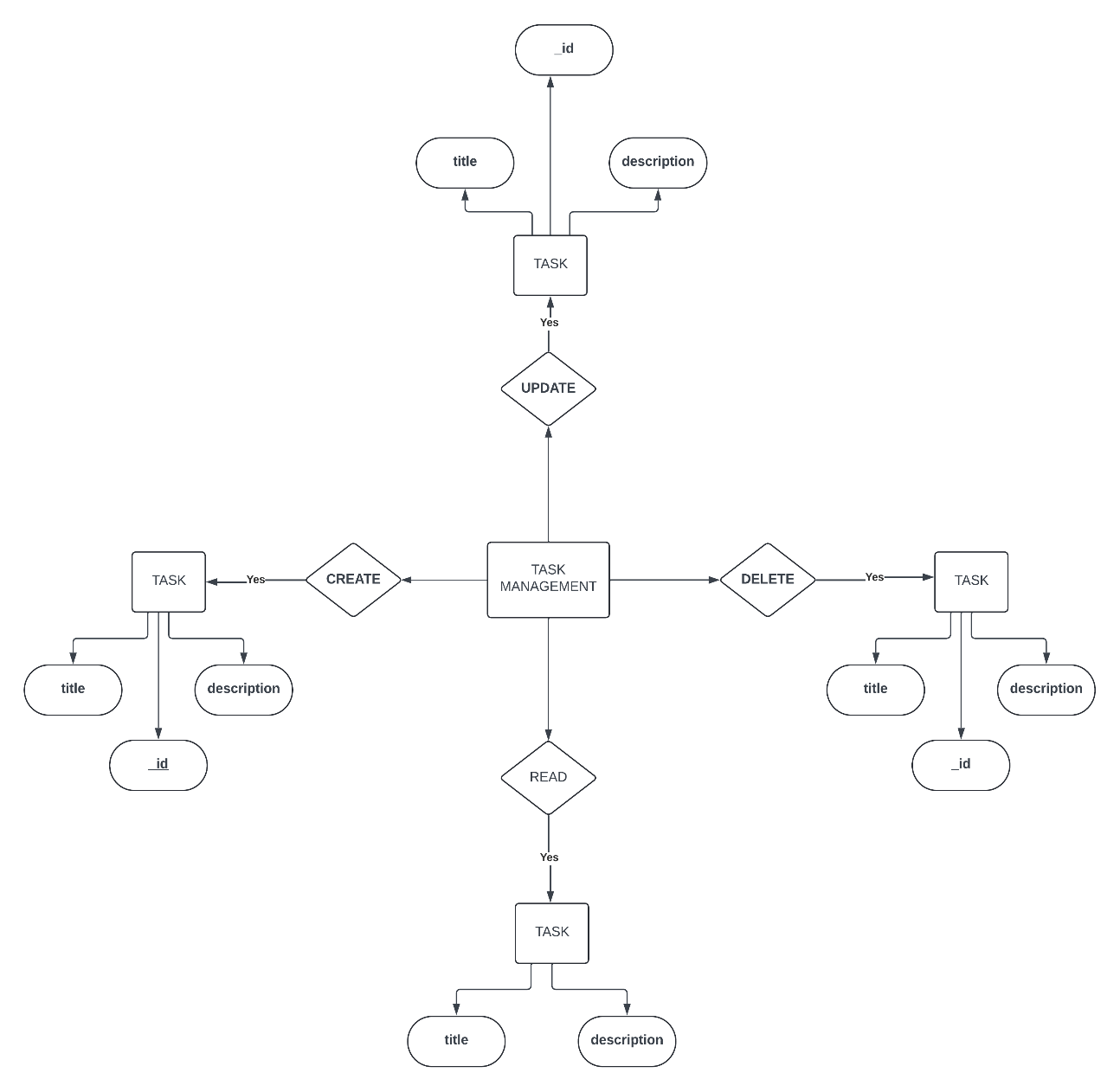
**Hardware Requirements**

|  |  |
| --- | --- |
| RAM | 8 GB or more for optimal performance, especially when running multiple applications alongside the database. |
| Hard disk | 50 GB or more to accommodate the operating system, and any data files or backups generated by the application. |
| Processor | Intel Core i5 or higher for better performance, particularly when handling multiple user requests or processing larger datasets. |

**Software Requirements**

|  |  |
| --- | --- |
| Database | MongoDB |
| Framework | Express |
| Operating System | Windows |

1. **ER Diagram**

****

**5. Database Schema**

In MongoDB, the database schema is defined by the collections and the documents within those collections. Below is the schema representation for the Notes collection used in the project.

**1. Notes Collection**

* **Collection Name**: Notes

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| \_id | ObjectId | A unique identifier automatically generated by MongoDB for each document. |
| title | String | The title of the note. It provides a brief overview of the note's content. |
| |  | | --- | | description | | String | Detailed content of the note, allowing users to elaborate on the task or reminder. |

**Example Document in the Notes Collection**

Here’s an example of how a document in the Notes collection might look:

{

"\_id": ObjectId( "671161efed31c7edcd2590a6")

"taskName": "DBMS MINI PROJECT",

"taskContent": "to add a user login which will provide authentication to our website",

"\_\_v": 0

}

**2. Future Considerations for Additional Collections (Optional)**

If you decide to extend the functionality in the future, you might consider adding the following collections:

**User Collection (for multi-user support)**

* **Collection Name**: Users

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| \_id | ObjectId | Unique identifier for each user (auto-generated) |
| username | String | Unique username for authentication |
| password | String | Password for user authentication (hashed) |
| email | String | User's email address |

**Example Document in the Users Collection**

json

{

"\_id": ObjectId("60c72b2f9b1e8a0f8c8f1234"),

"username": "john\_doe",

"password": "hashed\_password",

"email": "john.doe@example.com"

}

**6. Coding**  
// const exp = require('constants');

const express = require('express');

const fs = require('fs');

const path = require('path')

const mongoose = require('mongoose');

const dotenv = require('dotenv');

const bodyParser = require('body-parser')

const app = express();

const port = 3001;

app.use(express.urlencoded({extended:true}));

app.use(express.json());

app.use(express.static(path.join(\_\_dirname,'public' )));

app.set('view engine','ejs');

//database connection

dotenv.config();

const connectDB = async () => {

try {

await mongoose.connect(process.env.mongo\_url)

console.log(`DB Connected`);

} catch (error) {

console.log(`Error while connecting db ${error}`);

}

}

connectDB();

const tutSchema = new mongoose.Schema({

taskName:{

type:String,

required:true

},

taskContent:{

type:String,

required:true

}

});

const collection =mongoose.models.tasks || new mongoose.model('tasks',tutSchema);

app.get('/',async (req,res)=>{

const tasks = await collection.find({});

res.render('index.ejs',{tasks});

});

app.post('/create',async (req,res)=>{

const {title ,content} = req.body;

const newTask = new collection({ taskName: title, taskContent:content });

await newTask.save();

res.redirect('/');

});

app.get('/edit/:filename',async (req,res)=>{

const task = await collection.findById(req.params.filename);

res.render('edit.ejs', { task });

})

app.post('/edit/:filename',async (req,res)=>{

try{

const {title ,content } = req.body;

await collection.findByIdAndUpdate(req.params.filename,{taskName:title,taskContent:content

});

res.redirect('/');

}catch(err){

console.log(err);

res.redirect('/')

}

})

app.post('/delete/:filename',async (req,res)=>{

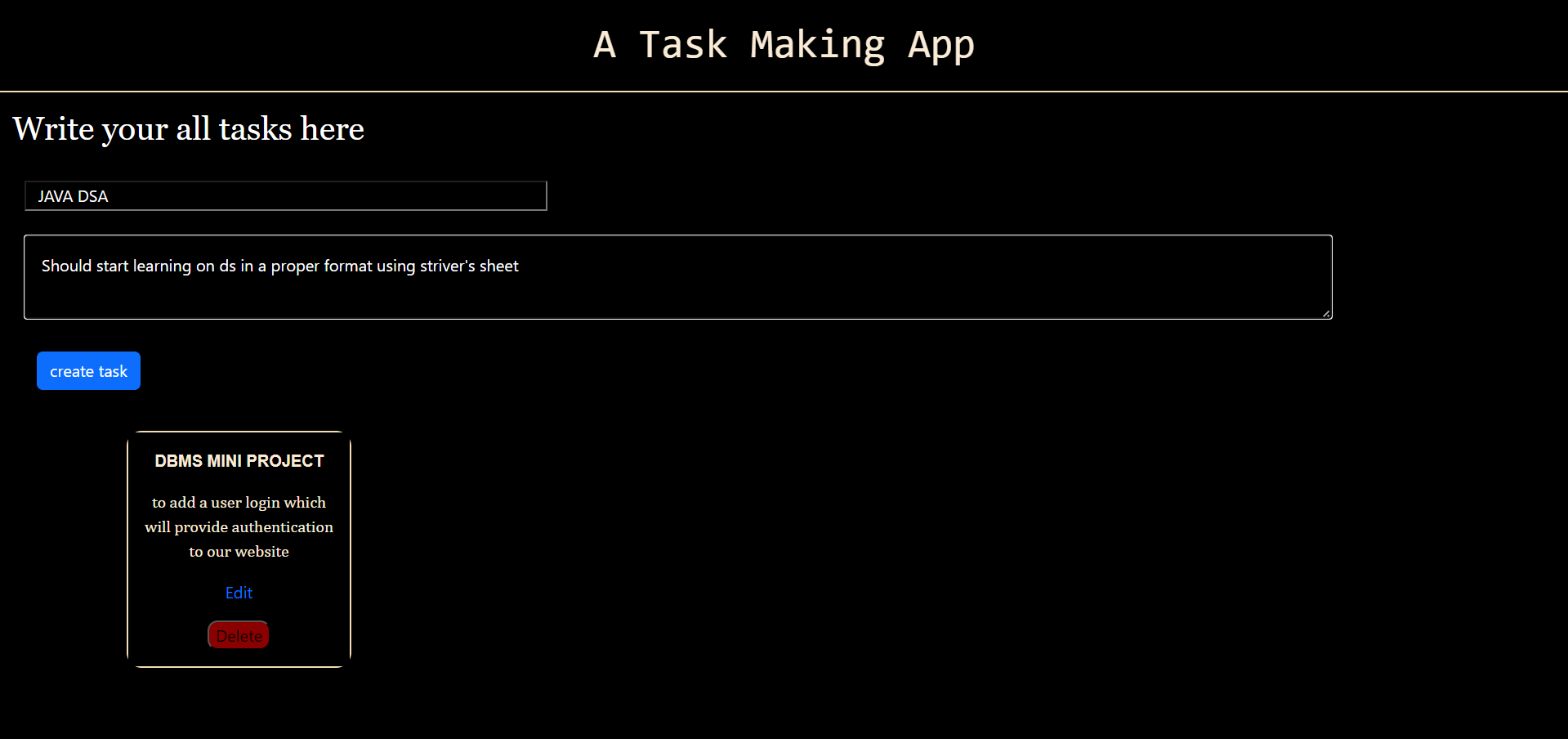
await collection.findByIdAndDelete(req.params.filename);

res.redirect('/');

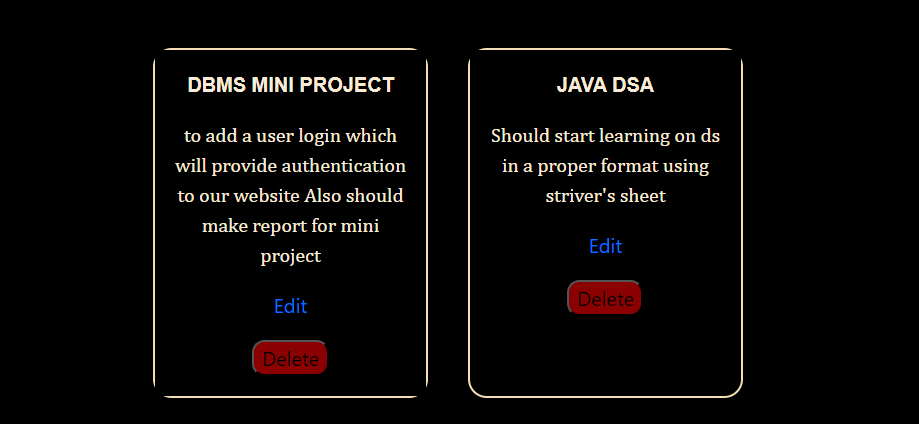
})

app.listen(`${port}`,);

1. **Screenshots**

****

****



**8. Future Scope and Enhancement**

The Task Management System using Notes has a solid foundation, but there are several potential enhancements and features that can be implemented in the future to improve its functionality, user experience, and overall performance:

1. User Authentication and Roles:
   * Implement a robust user authentication system that includes password recovery options.
   * Introduce user roles (e.g., admin, standard user) to control access to certain functionalities, allowing for better data management and security.
2. Collaboration Features:
   * Allow multiple users to collaborate on notes, enabling real-time sharing and editing of tasks.
   * Implement notifications for users when notes are updated or comments are added by collaborators.
3. Due Dates and Reminders:
   * Integrate a due date feature that allows users to set deadlines for tasks.
   * Implement reminder notifications via email or in-app alerts to remind users of upcoming deadlines.
4. Mobile Application:
   * Develop a mobile application version of the task management system to increase accessibility and usability for users on the go.
   * Ensure a responsive design for the web version to improve usability on mobile devices.
5. Analytics and Reporting:
   * Introduce analytics features that provide users with insights into their task completion rates, overdue tasks, and productivity metrics.
   * Generate reports that can be exported as PDF or Excel files for better tracking of tasks over time.
6. Enhanced Security Measures:
   * Implement advanced security features, such as two-factor authentication (2FA) and data encryption, to protect user data and enhance system security.
7. Integration with Third-Party Services:
   * Allow integration with third-party services like Google Calendar, Slack, or email platforms for a more comprehensive task management experience.
   * Enable data import and export functionalities to facilitate migration between different task management tools.
8. Customizable User Interface:
   * Provide options for users to customize the interface, such as themes or layout preferences, to enhance user experience.
9. Artificial Intelligence Features:
   * Explore AI-driven features, such as intelligent task suggestions based on user behavior and preferences, to enhance productivity and task management.

**9. Conclusion**

The Task Management System using Notes is a robust web-based application designed to simplify the process of managing tasks and notes for users. By leveraging the flexibility and scalability of MongoDB, the system provides an intuitive interface that allows users to create, read, update, and delete notes effortlessly.

Throughout the development of this project, key objectives have been achieved:

* User-Centric Design: The application is built with a focus on user experience, ensuring that the interface is easy to navigate and interact with.
* Efficient Task Management: Users can efficiently manage their tasks, enhancing productivity and organization.
* Future-Proof Architecture: The system’s design accommodates potential future enhancements, such as collaboration features and advanced analytics, ensuring that it can grow and adapt to user needs.

The project demonstrates the effectiveness of using modern web technologies for task management and highlights the advantages of NoSQL databases like MongoDB in providing flexibility in data handling. By allowing users to have a personalized and adaptable experience, this project sets a foundation for future developments in task management solutions.

**10.References**

* + MongoDB University: <https://university.mongodb.com/>
  + MongoDB Connectivity: <https://www.mongodb.com/docs/ops-manager/current/tutorial/connect-to-mongodb/#:~:text=To%20connect%20to%20a%20MongoDB,port%20for%20the%20mongos%20process>.
  + <https://nodejs.org/en>
  + <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
  + <https://www.json.org/json-en.html>