COLLEGE CODE: 9504
COLLEGE NAME: DR.G.U.POPE COLLEGE OF ENGINEERING
DEPARTMENT: CSE
STUDENT NM-ID : 974149B9DA591BFBA464DEE7049B2CD4
ROLL NUMBER: 950423104037
DATE: 06-10-2025

COMPLETED THE PROJECT NAME-:
PHASE 5

TECHNOLOGY PROJECT NAME
To-Do List Application

SUMITTED BY:
NAME: s.shenbagalakshmi
MOBILE NO: 7812844204

Record or present a demo showing:

Adding, editing, deleting, marking tasks as complete

.

User-friendly interface and smooth flow.

Any extra features (due dates, priority, categories, etc.).

Deployment link working on browser (Netlify/Vercel/Cloud).

Project Report

Structure example:

1. Introduction – Purpose of the To-Do List App.
2. Problem Statement – Why users need such an app.
3. Proposed Solution – Key features (CRUD, UI/UX, responsiveness).
4. Tech Stack – Angular/React/Node.js/MongoDB (or whichever used).
5. Implementation Details – Architecture, components, APIs.
6. Testing – Enhancements tested, bug fixes.
7. Deployment – How you deployed (Netlify/Vercel/Cloud).
8. Conclusion & Future Enhancements.

1. Introduction - Purpose of the To-Do List App

The main purpose of a To-Do List App is to help users **organize and manage their daily tasks efficiently** It allows users to create, view, update, and delete tasks in one place, ensuring that no important activity is forgotten.

The app acts as a **digital planner**, improving productivity and time management for both students and professionals.

2. Problem Statement - Why users need such an app

In today's busy lifestyle, people often **struggle to keep track of their tasks and deadlines**.Traditional methods like sticky notes or paper lists can easily be lost or forgotten. Users need a convenient, accessible, and reliable way to manage tasks — something that's **available anytime and from any device**.

A To-Do List App solves this by providing **real-time task tracking, reminders, and an easy-to-use interface**.

3. Proposed Solution - Key Features

The proposed To-Do List App includes essential features that make task management simple and efficient:

CRUD operations:

Create new tasks

Read/View existing tasks

Update/Edit tasks

Delete completed or unwanted tasks

User-friendly UI/UX:

Clean, minimal, and intuitive design for all users.

Responsive design:

Works smoothly on mobile, tablet, and desktop screens.

Optional features:

Task prioritization, deadlines, notifications, and category filters.

4. Tech Stack

Depending on your project choice, the app can be built with:

Frontend: React.js or Angular (for dynamic and interactive UI)

Backend: Node.js with Express.js (for API and server logic)

Database: MongoDB (for storing user data and tasks)

Deployment: Netlify or Vercel (for frontend) and Render or Railway (for backend, if separate)

5. Implementation Details

Architecture:

Frontend: Handles user interface and communicates with backend APIs.

Backend: RESTful APIs for CRUD operations (e.g., `/addTask`, `/getTasks`, `/updateTask`, `/deleteTask`).

Database: MongoDB collections store task details (title, description, status, date).

Components (React Example):

`App.js` – Main entry component

`TaskList.js` – Displays all tasks

`TaskForm.js` – For adding or editing tasks

`TaskItem.js` – Individual task card with edit/delete buttons

6. Testing

Conducted **unit testing** for components and API endpoints.

Verified **CRUD functionality** works without errors.

Checked **UI responsiveness** across devices.

Fixed minor bugs like incorrect task deletion or input validation issues.

Ensured smooth user experience and fast loading times.

7. Deployment

Frontend: Deployed using **Netlify** or **Vercel**, making the app accessible online with a shareable link.

Backend: Hosted on platforms like **Render**, **Railway**, or **AWS**

Database: Connected via a **MongoDB Atlas** cluster for cloud-based storage.

Verified that the frontend communicates correctly with the live backend APIs.

8.Conclusion & Future Enhancements

The To-Do List App successfully provides an efficient way to **manage daily tasks digitally**

.It demonstrates good use of modern web technologies and clean UI design.

Future Enhancements:

Add **user authentication** (login/signup).

Enable **notifications or reminders** for deadlines.

Support **dark/light mode** for better accessibility.

Implement **drag-and-drop task ordering** or **task categories**.

Create a **mobile version (React Native/Flutter)**.

🖼 Screenshots / API Documentation

Take screenshots of main features:

Task list, add task form, update task, completed tasks, etc.

If you used APIs (backend), document endpoints:

Example:

POST /tasks – Add a task
GET /tasks – Fetch all tasks
PUT /tasks/:id – Update task

DELETE /tasks/:id – Delete task

⚡ Challenges & Solutions

Example:

Challenge: Tasks not updating dynamically.

Solution:
Used React state management / Angular reactive forms.

Challenge: Deployment errors on Netlify.

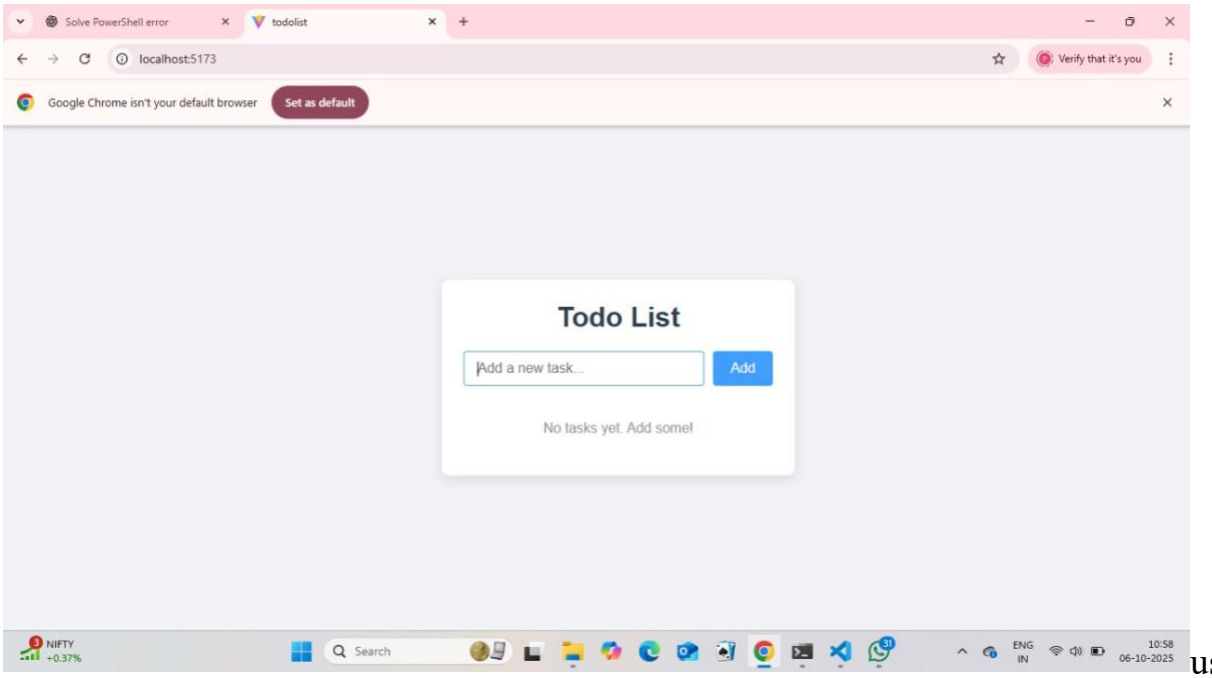
Solution:
Fixed build script and environment variables.

🗀 GitHub README & Setup Guide

README should include:

Project description.

Features list
.
Tech stack



used.

Setup steps:

git clone <repo-link>
cd project-folder
npm install
npm start

Deployment link (Netlify/Vercel).

Screenshots if possible.

🚀 **Final Submission**

GitHub repo link (with code + README).

https://github.com/Subashinimaharaja/To-Do-List.git

Deployed link (working demo).