“LINE UP”

A Project Report Submitted to

Rajiv Gandhi University of Knowledge Technologies, Srikakulam

For the fulfilment of mini project

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

G.Laskhmi Neeraja (S180746)

P.H.K.Maha Lakshmi(S180404)

G.Srinu(S180989)

Under the Esteemed Guidance of

Ms. Roopa Musidi

(Assistant Professor,Dept.of CSE)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Rajiv Gandhi University of Knowledge Technologies

Srikakulam 2023

CERTIFICATE

This is to certify that the mini project work titled **“Line Up”** was successfully submitted by **G.Lakshmi Neeraja (s180746), P.H.K.MahaLakshmi(s180404), G.Srinu(s180989)** under the guidance **of Ms.Roopa Musidi (Asst.Prof,Dept of CSE)** to Rajiv Gandhi University of Knowledge Technologies, Srikakulam, India is record of bona-fide project work carried out by them under my/our supervision and guidance and is worthy of consideration for the fulfilment of mini-project of Bachelor of Technology in Computer Science and Engineering of the institute.

Ms. Musidi Roopa M. Tech

Project Supervisor

Assistant professor

Department of CSE

RGUKT IIIT SRIKAKULAM

Mr. N. Sesha Kumar M. Tech

Head of the Department

Assistant professor

Department of CSE

RGUKT IIIT SRIKAKULAM

DECLARATION

We declared that this mini project work titled **“Line UP”** is carried out by us during the year 2022-2023 in fulfilment of Mini Project in **Computer Science and Engineering**

We further declare that this dissertation has not been submitted elsewhere for any Degree.The matter embodied in this dissertation report has not been submitted elsewhere for any other degree.Furthermore, the technical details furnished in various chapters of this thesis are purely relevant to the above project and there is no deviation from the theoretical point view for design, development and implementation.

G.Lakshmi Neeraja(S180746)

P.H.K.MahaLakshmi (S180404)

G.Srinu(S180989)

ACKNOWLEDGEMENT

I would like to express my profound gratitude and deep regards to our guide **Asst.Prof.Ms. M.Roopa Mam**, for her exemplary guidance, monitoring and constant encouragement to me throughout the Completion of this project. I am extremely grateful for the confidence bestowed in me and entrusting my project entitled "**Line Up**".

I express my gratitude **to Mr. N. Sesha Kumar** Sir ,Head of the CSE and other faculty members for being a source of inspiration and constant encouragement which helped me in completing the project successfully.

My sincere thanks to friends who have helped me with their suggestions and guidance and have been helpful in various stages of project completion.

Project Associate

G.Lakshmi Neeraja (S180746)

P.H.K.Maha Lakshmi(S180404)

G.Srinu(S180989)

ABSTRACT

A Line Up is a list of tasks that need to be completed, typically organized in a order . It is one of the simplest solutions for task management and provides a minimal and elegant way for managing tasks a person wishes to accomplish. Our aim is to design a simple and elegant website for people to keep a track of the status of their tasks .The immense satisfaction that one gets when completing the task and marking it on the list are incomparable .Moreover, creating a list of tasks ensure you don’t miss out on anything. It’s a scientific fact that when you write the task that you need to complete, you are even more motivated to complete it. With this in mind, we come to build a platform which will help people create their own task list. With the help of modern tools and technologies, we strive to build a minimal and efficient Line Up which minimizes distractions and helps people achieve task management with ease and without hassle.

Table of Contents

CHAPTER 1

INTRODUCTION

Introduction

In today’s fast-paced and busy world, individuals face numerous challenges in managing their time, staying organized, and achieving a healthy work-life balance.The demands of work, personal commitments, and constant connectivity can often lead to stress, burnout, and a sense of being contantly behind scheddule.The pace of modern life often requires individuals to find ways to optimize their productivity and make the mostof their limited time.

As a result ,there is a clear need for solution that can help individuals effectively manage their time, tasks, and overall well-being.

The objective of this project is to create a simple prioritized list of the tasks a person must complete. People make a list of everything they need to do, ranked according to priority from the most critical task at the top to the least critical task at the bottom.

A few of the features of a good Line Up application include:

- Plan and execute simple actions.
- Prioritize, manage, and reason about tasks.
- Record notes, action items and ideas.

By implementing this system, we can provide an efficient solution, a well-designed Line Up appliaction can significantly enhance task management, boost productivity, improve colloboration and reduce stress for individuals and teams.

1.1 Statement of the Problem

The use of traditional to-do lists and existing to-do list applications present several limitations and challenges that hinder effective task management, productivity and user experience.These limitations create the need for a new and improved to-do list application.Many existing applications do not provide a centralized and unified platform for users to manage all their tasks in one place.Exisiting appliactions lack robust prioritization feature, making it difficult for users to differentiate level of the task.Therefore, the problem at hand is to develop an a simple prioritized list of the tasks a person must complete. People make a list of everything they need to do, ranked according to priority from the most critical task at the top to the least critical task at the bottom.

Objective

The objective of the to-do list project is to develop a user-friendly and efficient application that enables individuals and teams to effectively manage their tasks, prioritize activities, enhance productivity, and achieve their goals. The project aims to address the limitations and challenges of traditional to-do lists and existing task management solutions by providing a comprehensive and intuitive platform for organizing and tracking tasks.

To implement robust features that enable users to prioritize tasks based on urgency, importance, and deadlines. Facilitate efficient task sorting to help users allocate their time and resources effectively.

1.2 Goals

- To provide users with a centralized platform to organize and manage their tasks effectively.
- The application should allow users to capture, categorize, and prioritize tasks in a structured and easily accessible manner.
- To allow users to add comprehensive details to their tasks, such as descriptions, due dates, attachments, subtasks, and labels.
- Customization options should be provided to tailor the task management system to individual preferences and workflows.
- To prioritize data security and privacy to protect users' sensitive information and task data.
- Appropriate measures should be implemented to ensure secure storage and transmission of data.

1.3 Scope

The Project aims to create an user-friendly and efficient application that enables individuals and teams to effectively manage their tasks, prioritize activities, enhance productivity, and achieve their goals.

The different areas where can use this project is:

- Students can benefit from using Line Up to manage their study schedules, assignments, exam preparation, and academic deadlines.
- It is essential in event planning to ensure that all necessary tasks and preparations are managed efficiently. They help event organizers list tasks such as venue selection, vendor coordination, guest invitations, logistics, and marketing efforts, ensuring a smooth event execution.
- To-do lists are useful for managing household tasks and chores. They help individuals or families keep track of daily chores, shopping lists, home maintenance

tasks, and other household responsibilities. To-do lists ensure that necessary tasks are completed, and nothing is overlooked.

CHAPTER 2

LITERATURE SURVEY

Basically , a literature survey provides a descriptionor summary of the works related to the research problem that is being investigated.

2.1 Collect Information

We have taken the information from the other sources like research online websites.Proper guidance and help using statistical data will surely help to prioritize the tasks . A background study is done to review similar existing systems used to helps them to keep at track of the task it is a simple side which requires no signing or login or any personal details but still records your task mark the completed task and store them even if you visit the site after a few days.

Mobile Applications for Task Management: Patel and Connelly (2016) examine the role of mobile applications in task management. They highlight the benefits and challenges of using mobile to-do list apps, emphasizing the importance of user preferences and the impact of technology on task management practices.

2.2 Study

Line Up key features:

- Task Creation
- Task Organization
- Task prioritization
- Task Details
- Search and Filtering

2.3 Benefits

This application provide a centralized platform for organizing and managing tasks effectively. They allow users to capture, categorize, and prioritize tasks, ensuring that important tasks are not overlooked and enabling users to stay organized.

Line Up application enable users to prioritize tasks based on their importance and urgency. By setting priorities, individuals can focus on completing critical tasks first, ensuring that their time and energy are allocated effectively.

This application allow users to track the progress of their tasks. It shows the completion status, providing a sense of accomplishment and motivation as tasks are marked as complete.

CHAPTER 3

ANALYSIS

3.1 Existing System

Existing systems for to-do list management vary widely in terms of functionality, platforms, and features. Todoist is a widely used task management application available on web browsers, mobile devices and desktop platforms. It offers features such as task creation, due dates, priorities, labels, and project organization. It provides collaboration options for team-based task management.

3.2 Drawbacks of existing system:

- **Limited Customization:** Some to-do list applications have limited customization options, which may restrict users from tailoring the application to their specific preferences and workflows.
- **Complex user interface:** Certain to-do list applications can be complex, with a steep learning curve, which may make it challenging for new users to adopt and utilize all the features effectively.
- **Dependency on internet connectivity:** Any to-do list applications rely on an internet connection to synchronize tasks, updates, and data across devices. This means that users may not be able to access or update their to-do lists when they are in areas with limited or no internet connectivity.

3.3 Proposed System

- Line Up application is to provide users with a user-friendly interface where they can create , organize, and monitor their tasks in a centralized location. The application will enable users to easily capture and record their tasks, set deadlines and remainders, assign priorities, and track their progress. By having all their tasks in one place, users can avoid forgetting important responsibilities and ensure that everything is completed on time.
- The Line Up project will offer several key features to enhance task management and productivity. Users will be able to create new tasks by providing essential details such as task titles, descriptions, due dates and priority levels. They can categorize tasks into different projects or categories for better organization.

3.4 Advantages

- The application will provide options to edit, delete, mark tasks as complete or incomplete, and search for specific tasks based on various criteria.
- It can do Task Prioritization and Sorting. It assigns priority levels of task(i.e : high, medium, low) indicate their importance or urgency.
- Line Up can sort tasks based on priority ,due date, or custom criteria.
- It can include a status field for each task, representing its current state or progress.
- Allow users to update the status of tasks as they progress. Users can manually change the status of a task based on its current state.
- Free version of application.
- Independent of internet connectivity
- Simple and friendly interface

System Requirements

Hardware Requirements:

Laptop or PC with:

- ❖ I3 processor system or higher
- ❖ 4 GB RAM or higher
- ❖ 100 GB ROM or higher

Software Requirements:

- ❖ Operating System : Windows 7 or higher
- ❖ Web Technologies : HTML, CSS, Javascript, Django, React.js
- ❖ IDE : Visual Studio
- ❖ Database : MySQL

CHAPTER 4

TECHNOLOGY OVERVIEW

Introduction:

Line Up application utilize various technologies to provide their functionality and deliver a seamless user experience. Here is a technology overview of to-do list applications:

HTML:

HTML (Hypertext Markup Language) is a standard markup language used for creating the structure and content of web pages. It provides a set of predefined elements and tags that define the layout, text, images, links, and other components of a web page.

CSS:

CSS (Cascading Style Sheets) is a styling language used to describe the presentation and visual appearance of HTML (or XML) documents. It defines how HTML elements should be displayed on a web page, including aspects such as layout, colors, fonts, and animations.

Javascript:

Javascript is a high level,interpretedprogramming language that is widely used in web development.In this application, it can be used to add interactivity to the web pages. JavaScript is supported by all modern web browsers and is an essential component of front-end and full-stack web development.

Django:

Django is a popular high-level Python web framework that is well-suited for building robust and scalable web applications, including Line Up application. Django provides an Object-Relational Mapping (ORM) layer, allowing developers to work with databases using Python classes and objects. This simplifies database operations and facilitates tasks such as storing and retrieving tasks, managing task relationships, and handling user authentication and authorization.

MySQL:

MySQL is a widely used open-source relational database management system that can be integrated with a Line Up application built using various programming languages and frameworks, including Django.MySQL can also be used to retrieve the data from the database and display it on the webpages.

CHAPTER 5

REQUIREMENT SPECIFICATIONS

Requirement discovery is the process by which analysis acquire organizational knowledge and apply it as they select the right technology for their particular application. The Software Requirements Specifications (SRS) in a complete description of the behavior of the system being developed. It contains a set of use cases that describe all user interactions with the software .Use cases are also known as functional requirements .In addition to use cases , SRS also include non-functional requirements . Non-functional requirements are requirements that impose constraints on design or implementation (such as quality standards).

5.1 Functional Requirements

Functional requirements are specific functionalities or actions that a software system or application must perform to meet the needs of its users. They describe what the system should do in terms of inputs, outputs, and behaviors. Functional requirements typically define the expected behavior of the system in response to various inputs or user actions.

Functional Requirements for our project

Task Creation: Users can be able to create new tasks by providing task name, task assigned date , due date , and description of the task.

Task Management: Task management for a Line Up involves various activities and processes to effectively handle tasks within the application like edit task , delete task , modify task and maintaining the tasks.

Search with Criteria: Users should be able to sort tasks based on criteria like by task name or level of the task like high, low, medium.

Filter with Priority: Users should be able to filter the tasks based on by selecting the options all tasks or this week tasks or last week tasks.

Status Tracking: The system should allow users to track the status of their tasks whether they are complete or incomplete the task .

5.2 Non Functional Requirements

Non-functional requirements are the criteria that define the quality attributes, characteristics, constraints, and limitations of the software system. Unlike functional requirements that focus on what the system should do, non-functional requirements focus on how the system should perform and behave. These requirements are often related to performance, security, usability, reliability, scalability, and other aspects that affect the overall quality and user experience

5.3 System Requirements

System requirements refer to the specifications and conditions that a software system must meet to operate effectively and efficiently. These requirements encompass the hardware, software, and network infrastructure needed to support the system. System requirements provide a guideline for system designers, developers, and users to ensure compatibility, performance, and functionality of the software. They typically include the following aspects:

5.3.1 Hardware Requirements

These specifications outline the minimum and recommended hardware components and capabilities needed to run the software effectively. This may include details such as the processor type and speed, RAM (memory) capacity, storage space, display resolution, and other specific hardware requirements.

Hardware requirements for project-
Laptop or PC with:
1. I3 processor system or higher
2. 4 GB RAM or higher
3. 100 GB ROM or higher

5.3.2 Software Requirements

Software requirements are a detailed description of the functionalities, features, and constraints that the software system should possess. These requirements serve as a foundation for software development, guiding the design, implementation, and testing processes.

Software requirements for our project-
1. Operating System : Windows 7 or higher
2. Web Technologies : HTML, CSS, Javascript, Django, React.js
3. IDE : Visual Studio
4. Database : MySQL

CHAPTER 6

SYSTEM DESIGN

DESIGN OF THE SYSTEM

Unified Modelling Language (UML) was created in 1995 by using merging diagramming conventions used by three application development methodologies: OMT by James Rumbaugh , Objector by Invar Jacobson and the Brooch procedure by using Grady Brooch. Previous to this time, these three amigos, together with a few dozen other practitioners had promoted competing methodologies for systematic program development, each and every with its possess system of diagramming conventions.

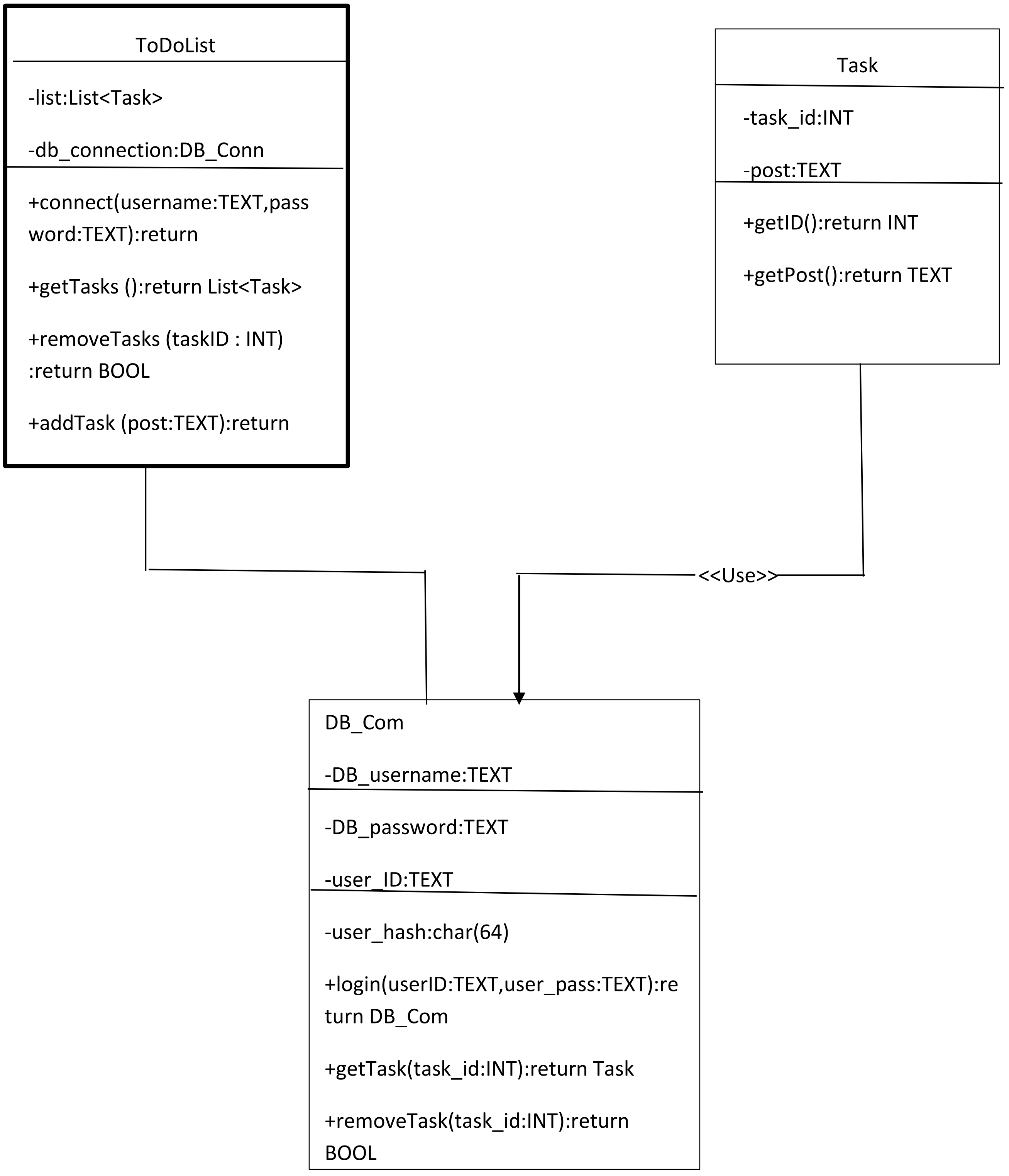
The methodologies adopted a sort of cookbook sort of pushing a application task via a succession of life cycle stages, culminating with a delivered and documented software. One purpose of UML was once to slash the proliferation of diagramming techniques by way of standardizing on a original modelling language, as a result facilitating verbal exchange between builders. It performed that goal in 1997 when the (international) Object administration team (OMG) adopted it as a commonplace. Some critics don't forget that UML is a bloated diagramming language written by means of a committee.

That said, I do not forget it to be the nice manner to be had today for documenting object-oriented program progress. It has been and is fitting more and more utilized in industry and academia . Rational Rose is a pc Aided program Engineering (CASE) software developed by way of the Rational organization underneath the course of Brooch, Jacobson and Rumbaugh to support application progress using UML. Rational Rose is always complex due to its mission of wholly supporting UML. Furthermore, Rational Rose has countless language extensions to Ada , C++, VB, Java, J2EE, and many others. Rational Rose supports ahead and reverse engineering to and from these langue ages.

However, Rational Rose does now not aid some usual design tactics as knowledge drift diagramsand CRC cards, due to the fact that these will not be a part of UML. Considering that Rational Rose has so many capabilities it is a daunting task to master it. Happily, loads can be executed making use of only a small subset of these capabilities. These notes are designed to introduce beginner builders into making productive use of the sort of subset.

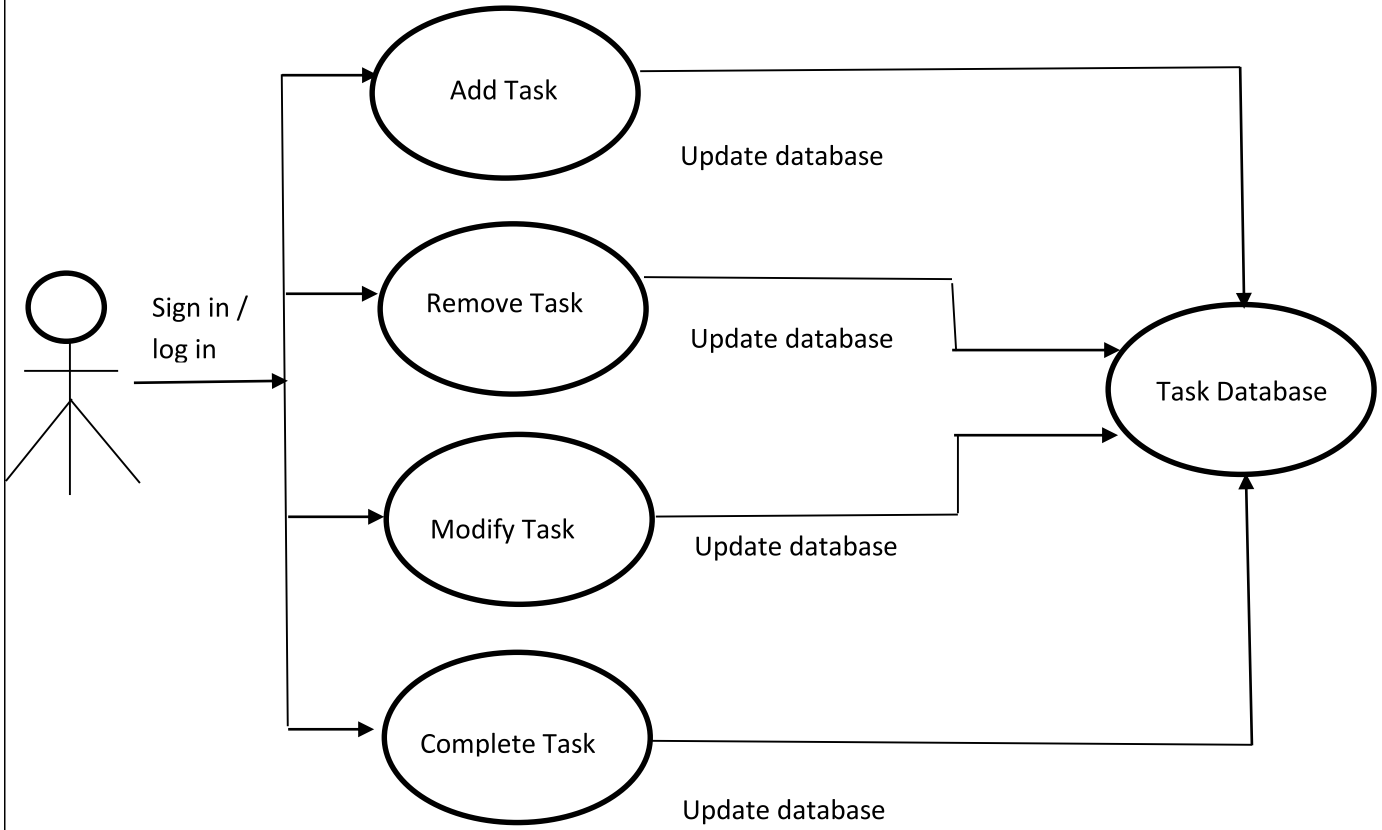
6.1 Class Diagram

Class diagram in the Unified Modelling Language (UML), is a kind of static structure diagram has describes the constitution of a process through showing the system's classes, their attributes, and the relationships between the class. The motive of a class diagram is to depict the classes within a model. In an object-oriented software, classes have attributes (member variables).operations (member capabilities) and relation.


ToDoList
-list:List<Task>
-db_connection:DB_Conn
+connect(username:TEXT,password:TEXT):return
+getTasks ():return List<Task>
+removeTasks (taskID : INT) :return BOOL
+addTask (post:TEXT):return
Task
-task_id:INT
-post:TEXT
+getID():return INT
+getPost():return TEXT
<<Use>>
DB_Com
-DB_username:TEXT
-DB_password:TEXT
-user_ID:TEXT
-user_hash:char(64)
+login(userID:TEXT,user_pass:TEXT):return DB_Com
+getTask(task_id:INT):return Task
+removeTask(task_id:INT):return BOOL

6.2 Use Case Diagram

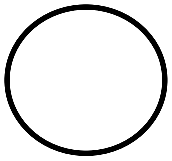
It is a visually representation what happens when actor interacts with system .A use case diagram captures the functional aspects of a system .The system is shown as a rectangle with name of the system inside ,the actor is shown as stick figures ,the use case is shown as solid bordered ovals labelled with name of the use case and relationships are lines or arrows between actor and use cases .Symbols used in use case are follows-



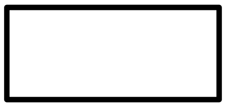
6.3 Data Flow Diagrams (DFD's)

A data flow diagram or bubb]e chart (DFD) is a graphical representation of the "flow" of data. through an information System, modelling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kinds of information will be put to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about Whether processes will operate in sequence or in parallel (which 1s shown on a flowchart).
The primitive symbols used for constructing DFD's are:

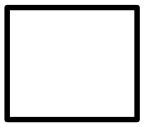
Symbols used in DFD



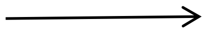
A Circle represents a "Process"



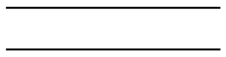
A Rectangle represents "External Entity"



A Square define a source or destination data

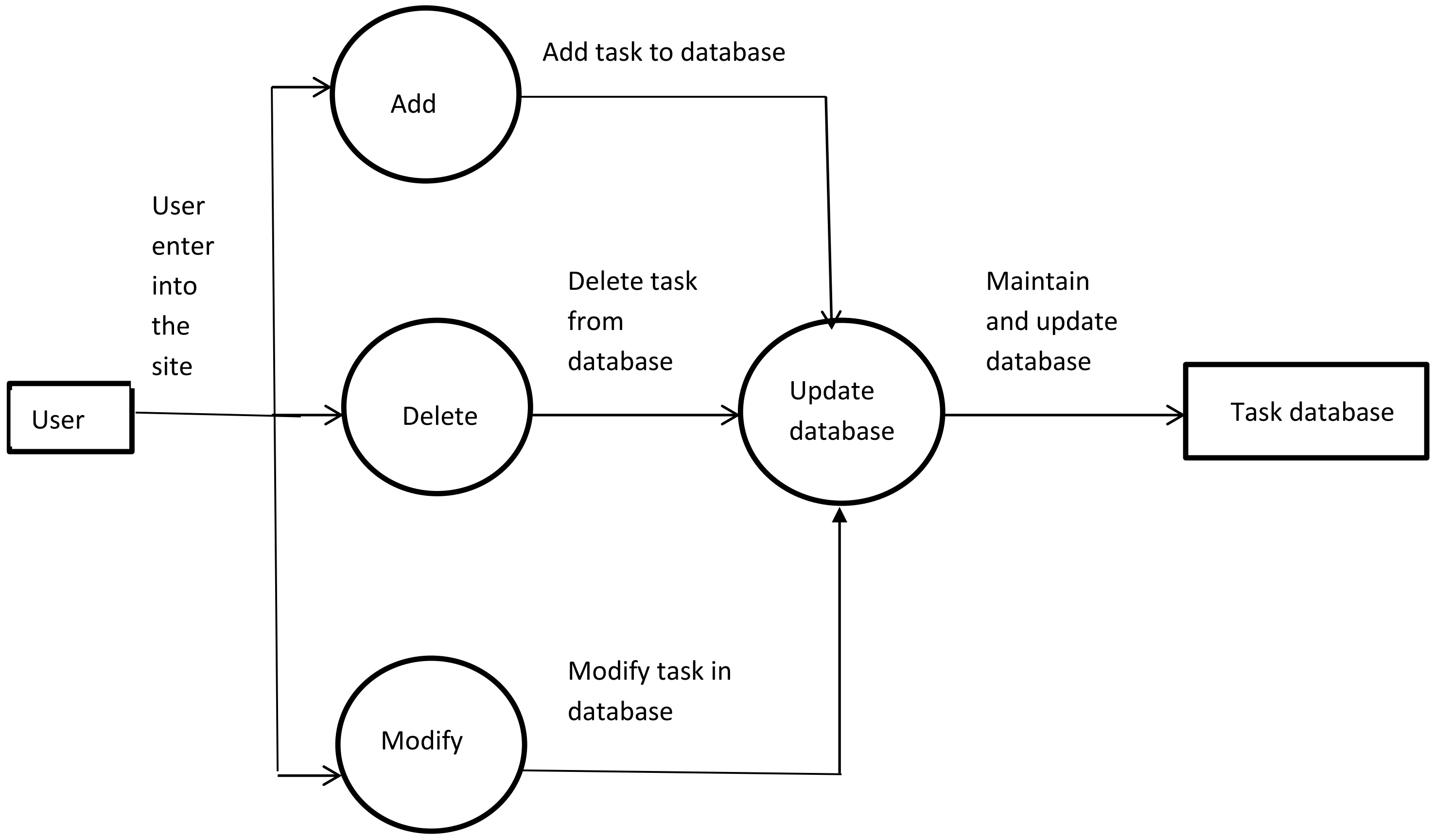


— An arrow identifies Dataflow



Storage indication of the data

Data flow Diagram

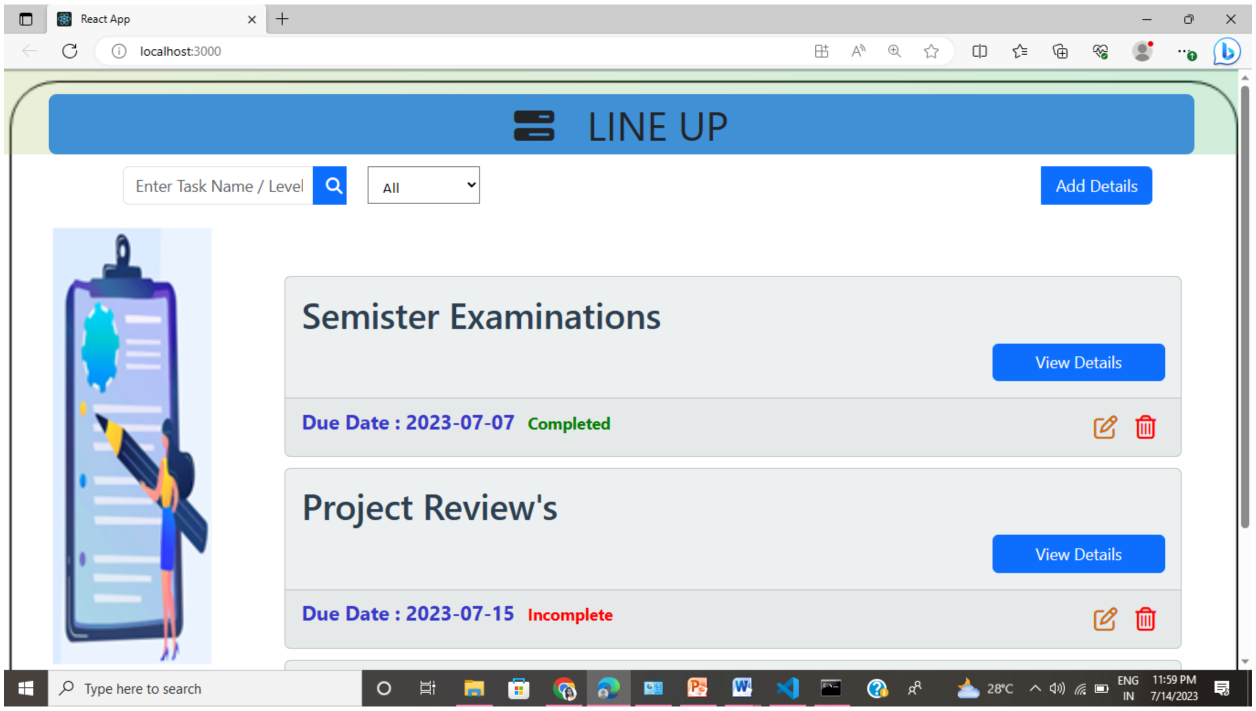


DFD for Line Up

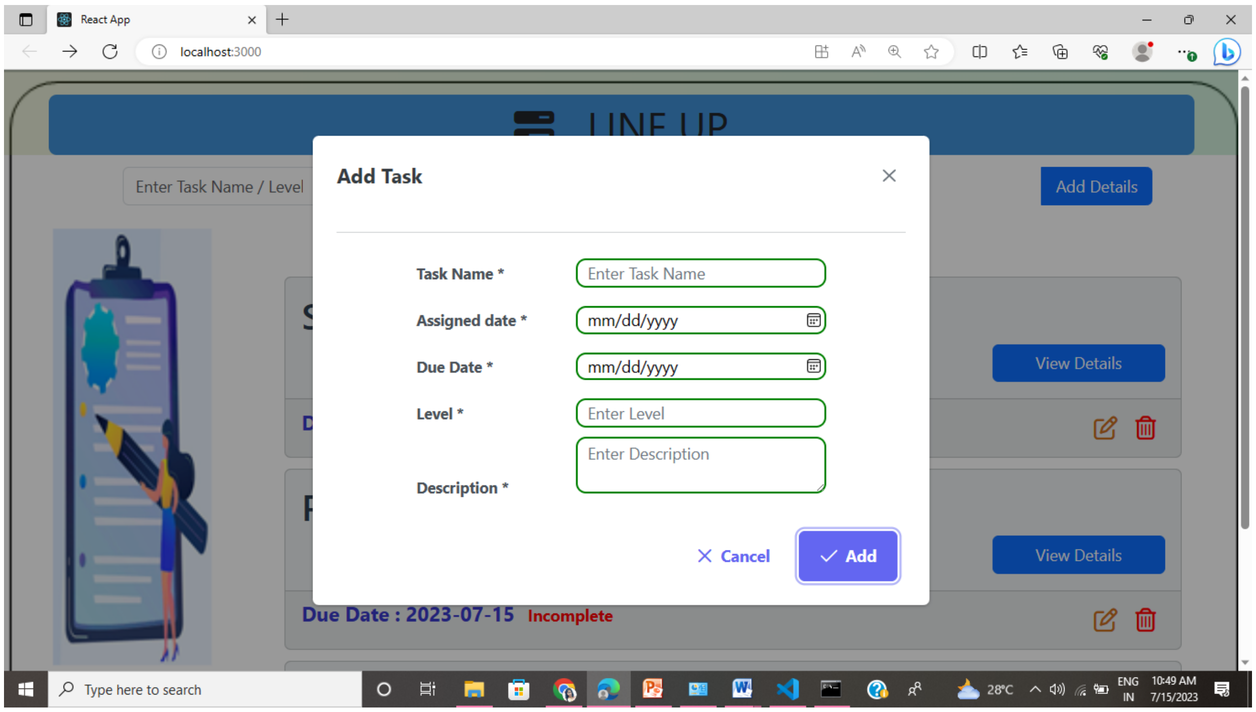
CHAPTER 7

IMPLEMENTATION

7.1 Interface of the project



7.2 Creating a New Task



Add Task

Task Name * Enter Task Name

Assigned date * mm/dd/yyyy

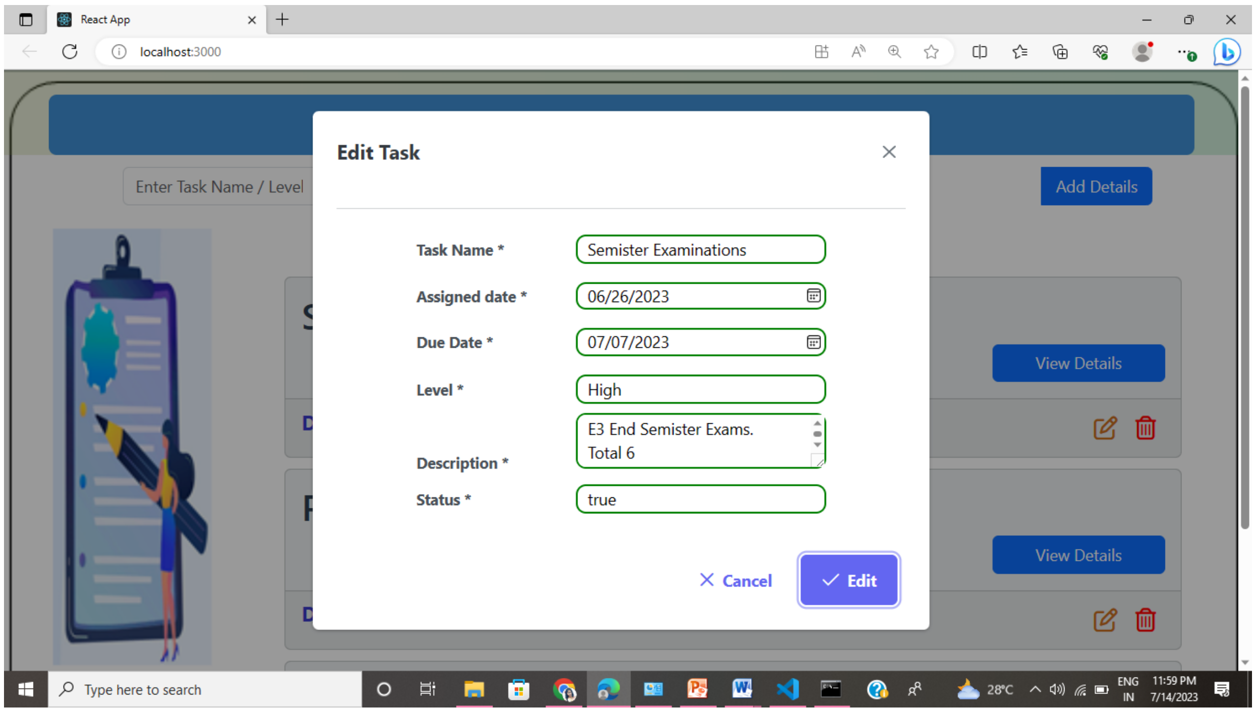
Due Date * mm/dd/yyyy

Level * Enter Level

Description * Enter Description

Due Date : 2023-07-15 Incomplete

7.3 Task Management



Edit Task

Task Name * Semister Examinations

Assigned date * 06/26/2023

Due Date * 07/07/2023

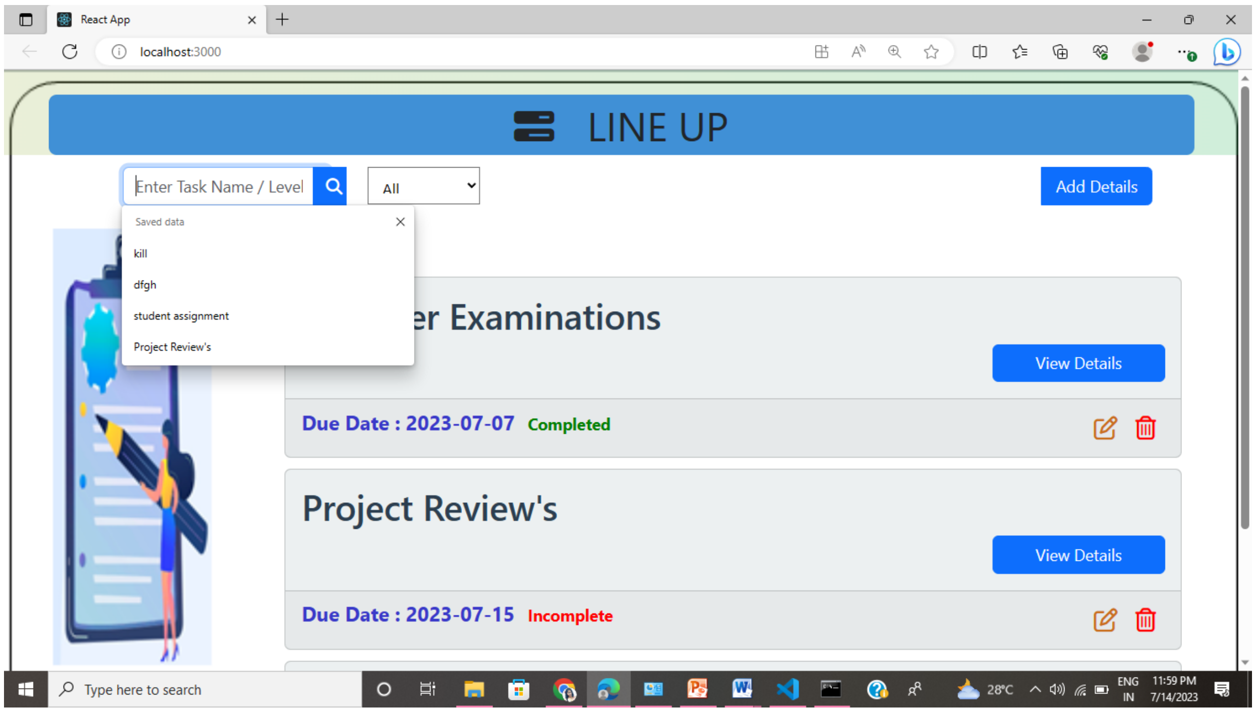
Level * High

Description * E3 End Semister Exams.
Total 6

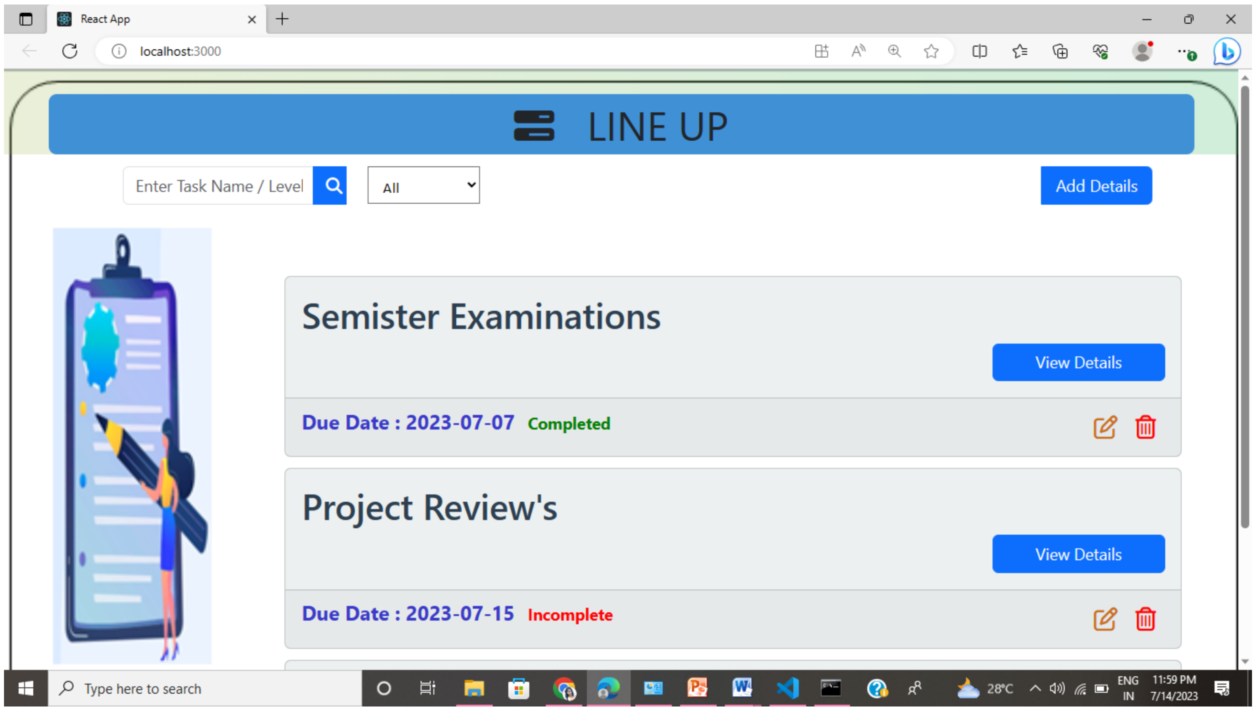
Status * true

Cancel Edit

7.4 Search on Different Criteria



7.5 Sorting With Criteria



Semister Examinations

Due Date : 2023-07-07 Completed

Project Review's

Due Date : 2023-07-15 Incomplete

CHAPTER 8

SOURCE CODE

Home Page

//importing modules

```
import React, { useState, useEffect } from 'react'

import './Main.css'

import { Button } from 'primereact/button';

import { Dialog } from 'primereact/dialog';

import { Paginator } from 'primereact/paginator';

import axios from 'axios';

import Popup from 'reactjs-popup';

import { useNavigate } from 'react-router-dom';
```

//functions declaration

```
var baseurl = 'http://localhost:8000/Task/task-data/'

export const Main = () => {

const [visible, setVisible] = useState(false)

const [data, setData] = useState([])

const [name, setName] = useState('')

const [isError, setIsError] = useState('')

const [assignedDate, setAssignedDate] = useState()

const [dueDate, setDueDate] = useState()

const [level, setLevel] = useState()

const [description, setDescription] = useState()

const [input, setInput] = useState('')
const [week, setWeek] = useState('')

const [form, setForm] = useState('')

const [openEdit,SetOpenEdit] = useState(false)

const [openDialog,SetOpenDialog] = useState(false)

const [idata,setIdata] = useState([])

const navigate = useNavigate()

async function getTaskData() {

try {

await axios.get(baseurl)

.then((response) => {

setData(response.data)

console.log(response.data)

})

} catch (error) {

console.log(error.message)

setIsError(error.message)

}

}

useEffect(() => {

getTaskData();


}, [])


const Addhandler = () => {

try {
```

```
axios.post(`${baseurl}`, {
task_name: name,
task_assigned_date: assignedDate,
task_due_date: dueDate,
priority_level: level,
description: description

})
.then((response) => {
console.log(response.data)
// setData(response.data)
getTaskData();

})
SetOpenDialog(false)
} catch (error) {
console.log(error.message)
setIsError(error.message)
}
}

const deleteHandler = (id) => {
try {
var result = window.confirm("Are you sure ..? \n Do You Want to delte this task..?");
if (result) {
```

```
axios.delete(`${baseurl}${id}`)

.then(() => {

getTaskData();


})

alert('task deleted')

}

} catch (error) {

console.log(error)

setIsError(error.message)

}

}


const searchHandler = (value) => {

try {

axios.get(`${baseurl}filter/${value}`)

.then((response) => {

console.log(response.data)


// fetchHandler(week)

setData(response.data)

// getTaskData()

})

} catch (error) {

console.log(error)
```

```
setIsError(error.message)

}

}


const editHandler = (id) => {

console.log(idata)

try {

axios.put(`${baseurl}${id}`,{

task_name: idata.task_name,

task_assigned_date: idata.task_assigned_date,

task_due_date: idata.task_due_date,

priority_level: idata.priority_level,

description: idata.description,

status:idata.status,

})

.then((response) => {

console.log(response.data)

// setData(response.data)

getTaskData();


})

SetOpenEdit(false);

} catch (error) {


}
```

```
}


const getForm = (id) => {
console.log(id, '**************')
console.log("clicked...!")
setVisible(true)
SetOpenEdit(true)
try {
axios.get(`${baseurl}${id}`)
.then((response) => {
console.log(response.data)
setIdata(response.data)


})
} catch (error) {


}
}


const taskData = (task) => {
const tdata = {
task_name: task.task_name,
task_assigned_date:task.task_assigned_date,
task_due_date:task.task_due_date,
priority_level:task.priority_level,
description:task.description
}
navigate("/task",{state : tdata});
}

const fetchHandler = (fetch, filter) => {
try {
if (filter === "") {
console.log(fetch)
axios.get(`${baseurl}fetch/${fetch}`)
.then((response) => {
console.log(response.data)
// searchHandler(input)
setData(response.data)
// getTaskData()
})
}
else if (fetch === "") {
axios.get(`${baseurl}filter/${filter}`)
.then((response) => {
console.log(response.data)

// fetchHandler(week)
setData(response.data)
// getTaskData()
})
}
else {
console.log(fetch, filter)
axios.get(`${baseurl}full-search/${fetch}/${filter}`)
.then((response) => {
console.log(response.data)
// searchHandler(input)
setData(response.data)
// getTaskData()
})
}
} catch (error) {
console.log(error)
setIsError(error.message)
}
}
const addfooterContent = (

<div>
<Button label="Cancel" icon="pi pi-times" onClick={() => SetOpenDialog(false)}
className="p-button-text" />
<Button label="Add" icon="pi pi-check" onClick={Addhandler} autoFocus />
</div>
);
```

```
const editfooterContent = (

<div>

<Button label="Cancel" icon="pi pi-times" onClick={() => SetOpenEdit(false)} className="p-button-text" />

<Button label="Edit" icon="pi pi-check" onClick={() => editHandler(idata.id)} autoFocus />

</div>

);

//Rendering to the web page

return (

<>

<Dialog header="Add Task"  visible={openDialog}  style={{ width: '50%' }} onHide={() => SetOpenDialog(false)} footer={addfooterContent}>

<hr />

<form className='form-horizontal' style={{ width: '80%', marginLeft: '10%' }}>

<label htmlFor='add'> Task Name *</label>

<input type='text' placeholder='Enter Task Name' onChange={(e) => setName(e.target.value)} />

<label htmlFor='add'> Assigned date *</label>

<input type='date' placeholder='Enter Assigned date' onChange={(e) => setAssignedDate(e.target.value)} />

<label htmlFor='add'> Due Date *</label>

<input type='date' placeholder='Enter Due Date' onChange={(e) => setDueDate(e.target.value)} />

<label htmlFor='add'> Level *</label>

<input type='text' placeholder='Enter Level' onChange={(e) => setLevel(e.target.value)} />
```

```
<label htmlFor='add'> Description *</label>

<textarea type='text' placeholder='Enter Description' onChange={(e) => setDescription(e.target.value)} />

</form>

</Dialog>

<Dialog header="Edit Task"  visible={openEdit} data={idata} style={{ width: '50%' }} onHide={() => SetOpenEdit(false)} footer={editfooterContent}>

<hr />

<form className='form-horizontal' style={{ width: '80%', marginLeft: '10%' }}>

<label htmlFor='edit'> Task Name *</label>

<input type='text' placeholder='Enter Task Name' defaultValue={idata.task_name} onChange={(e) => setIdata((prevValues)=> ({...prevValues,task_name:e.target.value}))} />

<label htmlFor='edit'> Assigned date *</label>

<input type='date' placeholder='Enter Assigned date' defaultValue={idata.task_assigned_date} onChange={(e) => setIdata((prevValues)=> ({...prevValues,task_assigned_date:e.target.value}))}/>

<label htmlFor='edit'> Due Date *</label>

<input type='date' placeholder='Enter Due Date' defaultValue={idata.task_due_date} onChange={(e) => setIdata((prevValues)=> ({...prevValues,task_due_date:e.target.value}))} />

<label htmlFor='edit'> Level *</label>

<input type='text' placeholder='Enter Level' defaultValue={idata.priority_level} onChange={(e) => setIdata((prevValues)=> ({...prevValues,priority_level:e.target.value}))} />

<label htmlFor='edit'> Description *</label>

<textarea type='text' placeholder='Enter Description' defaultValue={idata.description} onChange={(e) => setIdata((prevValues)=> ({...prevValues,description:e.target.value}))} />

<label htmlFor='edit'> Status *</label>

<input type='text' placeholder='Status of task' defaultValue={idata.status} onChange={(e) => setIdata((prevValues)=> ({...prevValues,status:e.target.value}))} />
```

```
</form>

</Dialog>


<div className='container'>

<div className='header'>

<i className="fa fa-tasks" aria-hidden="true"></i> &nbsp;

LINE UP

</div>

<div className='Search'>

<div className="input-group">

<div className="form-outline">

<input type="search" id="form1" className="form-control" placeholder='Enter Task Name /
Level' onChange={(e) => setInput(e.target.value)} />

<label className="form-label" htmlFor="form1"></label>

</div>

<button type="button" className="btn btn-primary" onClick={() => searchHandler(input)}>

<i className="fas fa-search"></i>

</button> &nbsp;&nbsp;

<select className="fetch-data" onChange={(e) => setWeek(e.target.value)} onClick={() =>
fetchHandler(week, input)}>

<option value="all">All</option>

<option value="thisweek">This Week</option>

<option value="lastweek">Last Week</option>


</select>
```

```
<button type="button" className="btn btn-primary add-task" onClick={() => SetOpenDialog(true)}> Add Details </button>

</div>


</div>

<div className='content'>

{

data !== [] && data?.map(task => {

return (


<div key={task.id} className='card' style={{ width: '80%', marginLeft: '20%', backgroundColor: '#ECF0F1 ' }} >


<div className="card-body" onClick={() => taskData(task)}>

<h4 style={{ color: '#2C3E50 ', fontSize: '35px' }}>{task.task_name}</h4>

<button type="button" className='btn btn-primary' style={{color:'white',width:"20%",float:'right'}} onClick={() => taskData(task)}>View Details</button>


{/* <p>{task.description}</p> */}

</div>

<div className='card-footer'>

<span style={{ color:'rgb(57, 57, 201)', fontWeight: 'bolder', fontSize: '20px' }}>Due Date : {task.task_due_date}</span>&nbsp;&nbsp;&nbsp;

{

(task.status)?<span style={{ color:'green', fontWeight: 'bold' }}>Completed</span>:<span style={{ color:'red', fontWeight: 'bold' }}>Incomplete</span>
}
<i className="fa fa-trash-o" id="pointer1" aria-hidden="true" style={{ color: 'red' }} onClick={() => deleteHandler(task.id)}></i>
<i className="fa fa-pencil-square-o" id="pointer2" aria-hidden="true" onClick={() => getForm(task.id)} style={{ color: '#CA6F1E' }}></i>


</div>


</div>
);
})}
</div>
</div>
</>
)
}
//Backend connection
from django.shortcuts import render
from rest_framework.response import Response
from rest_framework.views import APIView
from rest_framework import status
from django.db.models import Q
from .models import taskModel
from .serializers import taskModelSerializer
from datetime import datetime,timedelta
import calendar
```

```
class tasks(APIView):

def get(self,request):
obj = taskModel.objects.all()
serializer = taskModelSerializer(obj, many = True)
return Response(serializer.data,status=status.HTTP_200_OK)

def post(self,request):
serializer = taskModelSerializer(data = request.data)
if serializer.is_valid():
serializer.save()
return Response(serializer.data,status=status.HTTP_201_CREATED)
else:
return Response(serializer.errors,status=status.HTTP_400_BAD_REQUEST)

class taskDataManipulate(APIView):

def get(self,request,id):
try:
obj = taskModel.objects.get(id = id)
serializer = taskModelSerializer(obj)
return Response(serializer.data,status=status.HTTP_200_OK)
except taskModel.DoesNotExist:
msg = {'msg':'not found'}
```

```
return Response(msg,status=status.HTTP_404_NOT_FOUND)


def put(self,request,id):
try:
obj = taskModel.objects.get(id = id)
print(obj)
updatedTaskData = {}
updatedTaskData['task_name'] = request.data['task_name']
updatedTaskData['task_assigned_date'] = request.data['task_assigned_date']
updatedTaskData['task_due_date'] = request.data['task_due_date']
updatedTaskData['priority_level'] = request.data['priority_level']
updatedTaskData['description'] = request.data['description']
updatedTaskData['status'] = request.data['status']
print(updatedTaskData)


serializer = taskModelSerializer(obj,data = updatedTaskData)
if serializer.is_valid():
serializer.save()
return Response(serializer.data,status=status.HTTP_205_RESET_CONTENT)
else:
return Response(serializer.errors,status=status.HTTP_400_BAD_REQUEST)


except taskModel.DoesNotExist:
msg = {'msg':'not found'}
return Response(msg,status=status.HTTP_404_NOT_FOUND)
```

```
def delete(self,request,id):
try:
obj = taskModel.objects.get(id = id)
msg = {'msg':'deleted'}
obj.delete()
return Response(msg,status=status.HTTP_204_NO_CONTENT)

except taskModel.DoesNotExist:
msg = {'msg':'Not Found'}
return Response(msg,status=status.HTTP_404_NOT_FOUND)



class taskDataFilter(APIView):
def get(self,request,text):
try:
obj = taskModel.objects.filter(Q(task_name__icontains = text) | Q(priority_level__icontains = text))
serializer = taskModelSerializer(obj,many=True)
return Response(serializer.data,status=status.HTTP_200_OK)
except taskModel.DoesNotExist:
msg = {'msg':'Not found'}
return Response(msg,status=status.HTTP_404_NOT_FOUND)

class FetchTaskData(APIView):
```

```
def get(self,request,text):

try:

startdate = datetime.today()

if(text == 'thisweek'):

# startdates = startdate + timedelta()

startdate = startdate + timedelta(days = -(startdate.weekday() ))

enddate = startdate + timedelta(days = 6)

print(f"start date : {startdate} - enddate : {enddate}")

obj = taskModel.objects.filter(Q(task_assigned_date__range = (startdate,enddate)) | Q(task_due_date__range = (startdate,enddate)) | Q(task_assigned_date__lte = enddate,task_due_date__gte = enddate))


elif(text == 'lastweek'):

enddate = startdate + timedelta(days = -(startdate.weekday()+1))

startdate = enddate+ timedelta(days = -(enddate.weekday() ))

print(f"start date : {startdate} - enddate : {enddate}")

obj = taskModel.objects.filter(Q(task_assigned_date__range = (startdate,enddate)) | Q(task_due_date__range = (startdate,enddate)) | Q(task_assigned_date__lte = enddate,task_due_date__gte = enddate))


elif(text == 'all'):

obj=taskModel.objects.all()


else:

raise NameError(text)
```

```
serializer = taskModelSerializer(obj,many=True)

return Response(serializer.data,status=status.HTTP_200_OK)


except taskModel.DoesNotExist:

msg = {'msg':'Not Found'}

return Response(msg,status=status.HTTP_404_NOT_FOUND)


class FullSearch(APIView):

def get(self,request,filter,fetch):


# fetch = request.data['week']

if fetch == "" and filter == "":

obj = taskModel.objects.all()

serializer = taskModelSerializer(obj,many=True)

return Response(serializer.data,status=status.HTTP_200_OK)


elif filter == "" :

try:

obj = taskModel.objects.filter(Q(task_name__icontains = filter) | Q(priority_level__icontains = filter))

serializer = taskModelSerializer(obj,many=True)

return Response(serializer.data,status=status.HTTP_200_OK)

except taskModel.DoesNotExist:

msg ={'msg':'not found'}

return Response(msg,status=status.HTTP_404_NOT_FOUND)
```

```
else:

startdate = datetime.today()

if fetch == "thisweek":

startdate = startdate + timedelta(days = -(startdate.weekday() ))

enddate = startdate + timedelta(days = 6)

print(f"start date : {startdate} - enddate : {enddate}")

elif fetch == "lastweek":

enddate = startdate + timedelta(days = -(startdate.weekday()+1))

startdate = enddate+ timedelta(days = -(enddate.weekday() ))

print(f"start date : {startdate} - enddate : {enddate}")


else:

raise NameError(fetch)


fetched_records = taskModel.objects.filter(Q(task_assigned_date__range = (startdate,enddate)) |
Q(task_due_date__range = (startdate,enddate)) | Q(task_assigned_date__lte =
enddate,task_due_date__gte = enddate))


try:

obj = fetched_records.filter(Q(task_name__icontains = filter) | Q(priority_level__icontains =
filter))

serializer = taskModelSerializer(obj,many=True)

return Response(serializer.data,status=status.HTTP_200_OK)

except fetched_records.DoesNotExist:

msg ={'msg':'not found'}
```

```
return Response(msg,status=status.HTTP_404_NOT_FOUND)
```

CHAPTER 9

CONCLUSION

Line Up Project is used to provide a practical and efficiency solution for managing task and organisingdaily activities by implementing a lineup application uses can benefit from various features such as task creation due date management collaboration and more the project focus on simplifying task Management increasing productivity and ensuring that important as are completed on time.

Throughout the development process it is crucial to consider user requirements and feedback to create a user friendly interface that need that needs additionally integrating the application with other tools and platform such as calendars of third party applications can enhance its functionality and usability

The success of a lineup project lies in its ability to provide it streamlined and intuitive user experience allowing uses to easily create organised and track their task regular updates and improvements based on user feedback can further enhance the application and keep it aligned with evolving user requirements.

Ultimately the goal of a lineup project is to empower uses to efficiently manage their tasks, increase productivity and reduce stress by providing a reliable and accessible tool for task organisation and management

CHAPTER 10

REFERENCES

1. https://www.researchgate.net/publication/353176797_To_Do_List_Web_App
2. Mobile Applications for Task Management: Patel and Connelly (2016)
3. "The To-Do List Effect: Conserving Task Completion Capacity Through Task Preloading" by Jihae Shin and Katherine L. Milkman (2017)
4. IEEE papers on Astudy on TODO Task Management at Home.