UCS1611 INTERNET PROGRAMMING LAB
EX8 - MINI PROJECT REPORT

ONLINE TO-DO APP

RANJANI A – 195001089
POOJA M – 195001076
SANJANA A – 195001097
NITHISH BHARATHWAJ K R – 195001072

PROBLEM STATEMENT

Our project is carried out to develop a website for online to-do app. This system has various options like adding a new task, editing the old tasks, deleting the completed and uncompleted tasks. This system helps the user to update his/her task whenever they want.

The user directly enters the main page, where the tasks are displayed and maintained. If the user wants to add a task, they can add any number of tasks and if they want to change a task's details, they can update it and they can also delete a task, if they have completed it or don't need it anymore. The user is not allowed to add an empty task to the list. The task which has been completed is stricken off, by clicking on the task card.

Whenever the user adds a task or updates a task or deletes a task or strikes out a task, the corresponding changes are updated in the database simultaneously.

The date and the description about the tasks are also taken from the user as input when adding a task and it is stored in the database. While editing the task the data can be edited.

SOFTWARE REQUIREMENTS SPECIFICATIONS

2.1 Functionality

2.1.1 Getting Input

The system enables the user to add the task by add task bar and edit the task by update the task in a popup window and also delete a task by delete icon, which updates the back end database.

2.2 Usability

2.2.1 User Accessibility

The system allows the users to access the system from the Internet using HTML or its derivative technologies.

2.2.2 User Friendliness

The system is such that it stands up to the user’s expectations. The system is easy to learn and understand. A native user can also use the system effectively, without any difficulties.

2.3 Performance

2.3.1 Response Time

The response of all the operations is good. This has been made possible by careful programming.

2.3.2 Error Handling

The system is able to respond to user errors and undesired situations, without halting or crashing.

2.3.3 Safety and Robustness

The system is able to avoid or tackle disastrous actions. In other words, it should be fool proof. The system safeguards against undesired events, without human intervention.

2.3.4 Portability

The software is not architecture specific. It should be easily transferable to other platforms if needed.

2.4 Reliability

2.4.1 Separate Components' Reliability

The reliability of the whole project depends on the reliability of separate components. The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated to reflect the current changes.

2.4.2 Computer Stability

The system will be functioning inside a computer. Thus, the overall stability of the system depends on the stability of the computer and its underlying operating system.

2.4.3 Availability

The system is available 100% for the user and is used 24 hrs a day and 265 days a year. The system shall be operational 24 hours a day and 7 days a week.

2.4.4 Mean Time Between Failures (MTBF)

The system will be developed in such a way that it may fail once in a year.

2.4.5 Mean Time to Repair (MTTR)

Even if the system fails, the system will be recovered back up within an hour or less.

2.4.6 Accuracy

The accuracy of the system is limited by the accuracy of the speed at which the user use the system.

2.4.7 Access Reliability

The system shall provide 100% access reliability.

2.5 Supportability

2.5.1 Maintenance

The maintenance of the system shall be done as per the maintenance contract.

2.5.2 Internet Protocols

The system shall comply with the TCP/IP protocol standards and shall be designed accordingly.

2.5.3 Standards

The coding standards and naming conventions will be as per the American standards.

2.6 Security

The system uses SSL (Secure Socket Layer) in all transactions that include any confidential customer information. The system will automatically log out all customers after a period of inactivity. The system should not leave any cookies on the customer's computer containing the user's passwords. The system's back-end servers shall only be accessible to authenticated management.

2.7 Interfaces

2.7.1 User Interface

The system will make use of the existing Web Browsers such as Google Chrome or Mozilla Firefox. The user interface of the system shall be designed as shown in the user interface prototypes.

2.7.2 Hardware Interfaces

The existing Local Area Network (LAN) will be used for collecting data from the users.

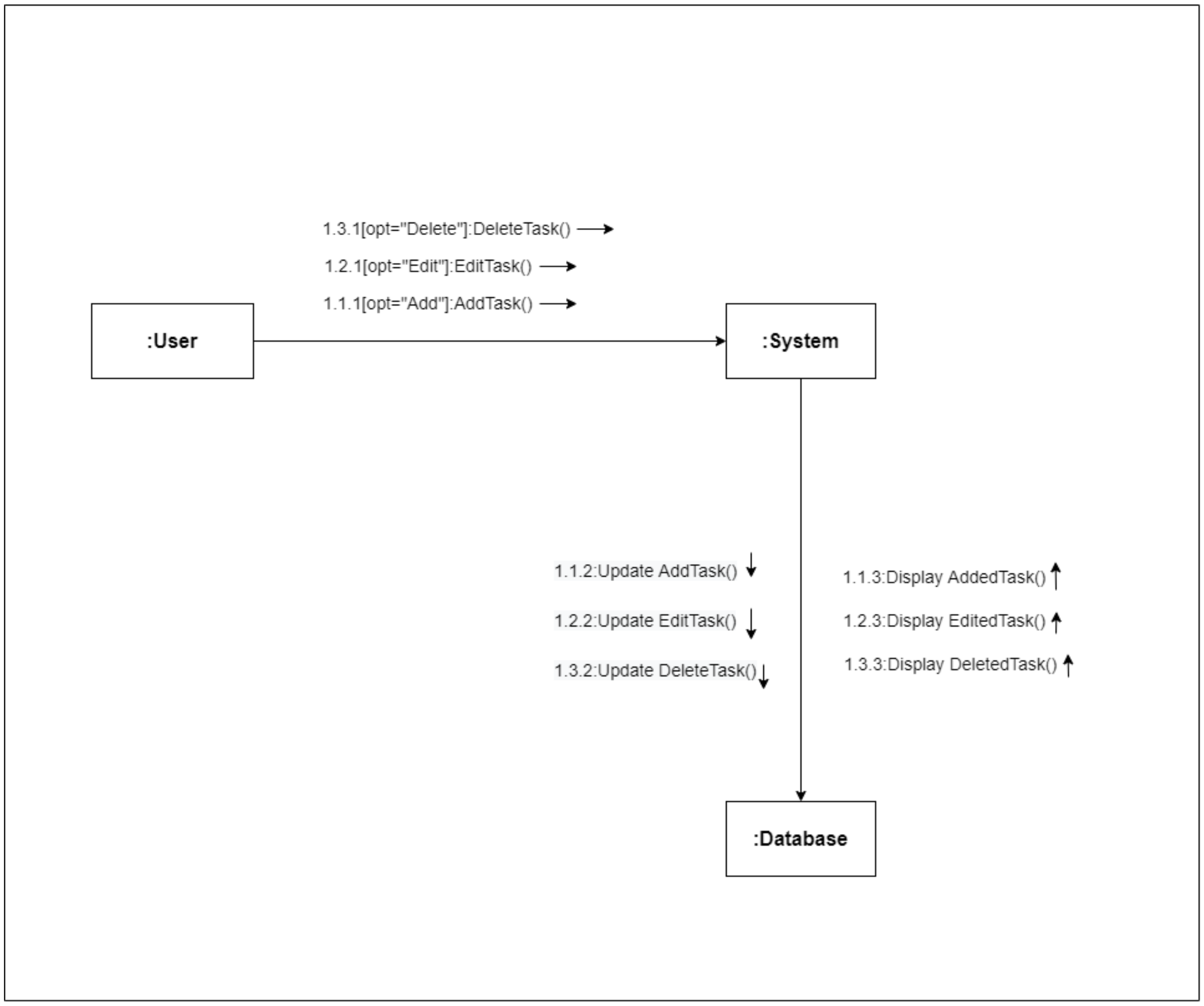
2.7.3 Software Interfaces

A firewall will be used with the server to prevent unauthorized access to the system.

2.7.4 Communications Interfaces

The Online Notes will be connected to the World Wide Web.

SYSTEM DESIGN


1.3.1[opt="Delete"]:DeleteTask()
1.2.1[opt="Edit"]:EditTask()
1.1.1[opt="Add"]:AddTask()
:User
:System
1.1.2:Update AddTask()
1.2.2:Update EditTask()
1.3.2:Update DeleteTask()
1.1.3:Display AddedTask()
1.2.3:Display EditedTask()
1.3.3:Display DeletedTask()
:Database

FUNCTIONALITIES

4.1 AddTask

The User can type the new task name in the ‘add Task’ bar and append the new task to the previous list by using the ‘add task’ button. Empty task name or task name with only space cannot be added as a task. This ‘Add Task’ button adds your current task to the existing task card list in the main page and also in the database.

4.2 EditTask

Edit Task is used to edit the existing tasks in the list. Edit option is available in each task card. By clicking the edit option, the popup window will be displayed to type the task details to replace the existing task details. After editing, the edited task will be updated in our main page and also in the database simultaneously.

4.3 TaskComplete

Task Complete is used to mark the completed tasks. User can click the task name which triggers the task complete function and then the task name is stricken off to show that the task is completed. If the User want to redo the task then the user can undo the strike by clicking the stricken task card and he/she can edit if they want.

4.4 DeleteTask

Delete task is used to delete the unwanted or completed tasks. The user can delete the task by clicking the delete icon which is available in each task card.

LOGIC

5.1 Adding Task

A react component for adding a task is placed at the top of the webpage. An input field takes the task name. The ‘add task’ button is clicked. The component then calls the post function in server side, which posts the newly added task name in MongoDB database. A response is sent back to the client side. When the response is received, a new task card with the given name is appended in the task card list in the webpage. But, description & due date of the task can be updated later.

5.2 Editing Task

Each task card has an ‘edit task’ button on the right. When clicked, a popup window is overlaid on the screen, displaying three input fields for task name, description & due date respectively. The desired data is filled & ‘update’ button is clicked. This triggers the update function in the server side, via a signal from the client side. The update function adds the given data to the database and sends success response to the client. Now, the updated contents are also displayed in the task card.

5.3 Deleting Task

Each task card has a ‘delete task’ button on the right. When clicked, a delete function in the server side is called through a signal from client side. The function removes the task from the database and sends OK response to the client. The client then updates its task list accordingly.

.

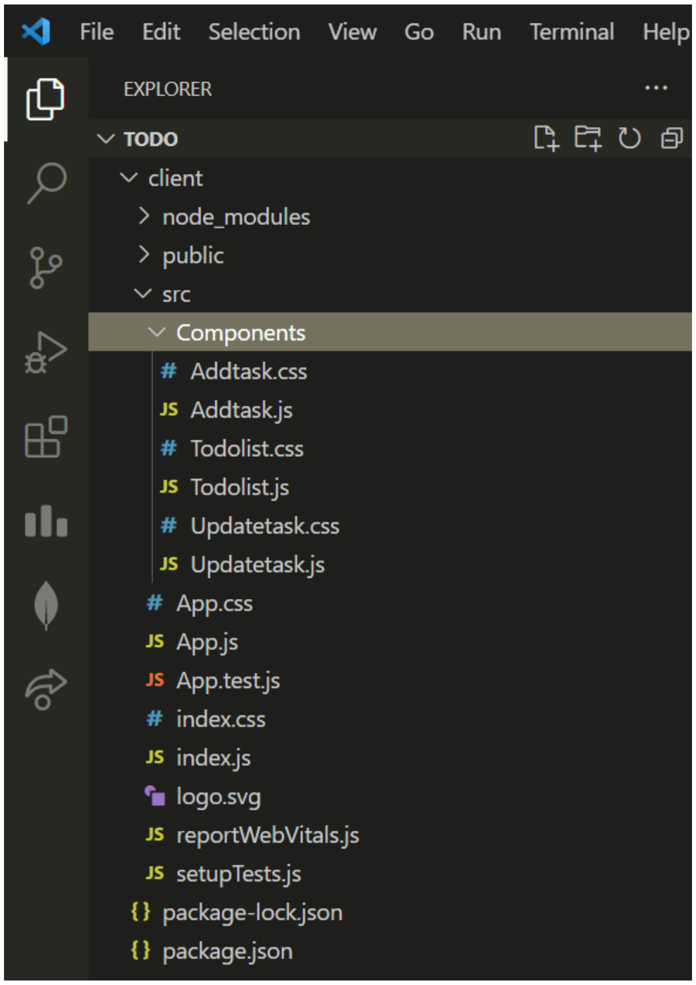
5.4 Completing a Task

When a task card is clicked, the contents are to be stricken off. To achieve this, an ‘onclick’ event is added to the task card, which adds a ‘text-decoration: line-through’ style to the contents. Now, it will appear as stricken off, meaning it is completed.

IMPLEMENTATION

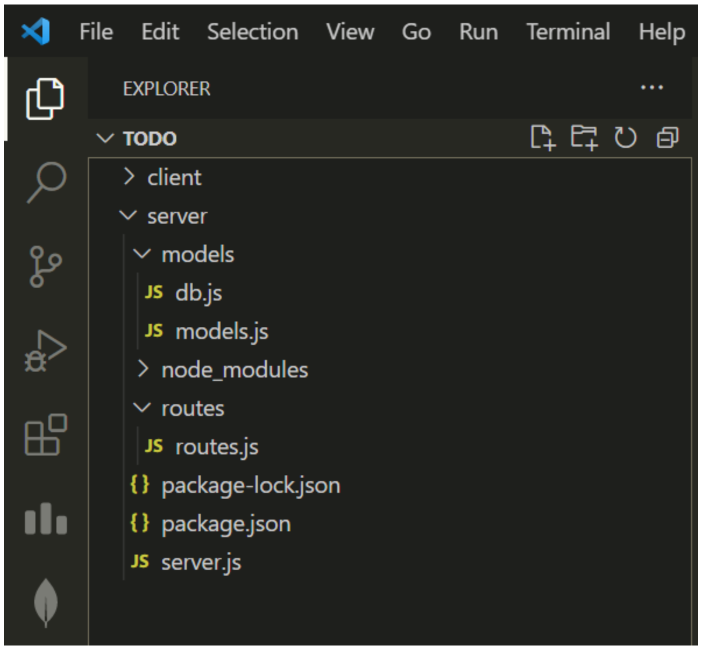
DIRECTORY STRUCTURE

CLIENT SIDE:



```
TODO
  client
    node_modules
    public
    src
      Components
        Addtask.css
        Addtask.js
        Todolist.css
        Todolist.js
        Updatetask.css
        Updatetask.js
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    package-lock.json
    package.json
```

SERVER SIDE:



```
TODO
  client
  server
    models
      db.js
      models.js
    node_modules
    routes
      routes.js
    package-lock.json
    package.json
    server.js
```

SOURCE CODE

db.js:

```
const mongoose = require('mongoose')

module.exports =
mongoose.connect('mongodb://localhost:27017/Todo',{
    // useUnifiedTopology : true,
    // useNewUrlParser : true,
    // useFindAndModify : false
}, err=> {
    if(err) console.log(`Error in DB Connection ${err}`)
    console.log(`MongoDB Connection Suceeded...`)
})
```

models.js:

```
const mongoose = require('mongoose')

constTaskSchema = new mongoose.Schema({
    todo : String,
    desc: String,
    date: String,
    isComplete : Boolean

})

const Task = mongoose.model('task',TaskSchema)

module.exports = Task
```

routes.js:

```
const express = require('express')
constTask = require('../models/models')
const router = express.Router()

router.get('/' , (req,res) => {
    Task.find((err,docs) => {
        if(err) console.log(err)
        res.json(docs)
    })
})

router.post('/',(req,res) => {
    const task = new Task(req.body)
    task.save((err,doc) => {
        if(err) console.log(err)
        res.json(doc)
    })
})

router.put('/:id',(req,res) => {
    Task.findOneAndUpdate({
        _id :req.params.id
    },req.body,{
        new : true
    },(err,doc) => {
        if(err) console.log(err)
        console.log(doc)
        res.json(doc)
    })
})

router.delete('/:id',(req,res) => {
    Task.findByIdAndDelete(req.params.id,(err,doc) => {
        if(err) console.log(err)
        res.json(doc)
    })
})

module.exports = router
```

server.js:

```
const express = require('express')
constcors = require('cors')
const router = require('./routes/routes')
const app = express()
require('./models/db')
app.use(express.json())
app.use(cors())
app.use('/api/tasks' , router)
app.listen('8000' , err=> {
    if(err) console.log(err)
    console.log('Server is started at PORT number : 8000')
})
```

Addtask.js:

```
import './Addtask.css'
import React ,{useState} from 'react'
import axios from 'axios'
functionAddtask(props) {
    const [task,Settask] = useState("")
    const [desc] = useState("")
    const [date] = useState("")
    constaddtask = () => {
        if(task.trim() === ''){
            return
        } else {
            axios.post('http://localhost:8000/api/tasks' , {
                todo : task,
                desc :desc,
                date : date,
                isComplete : false
            }).then(res=> {
                Settask("")
                props.addTask(res.data)
            }).catch(err=> console.log(err))
        }
    }
    return (

        <div className = 'addtask'>
            <input style={{borderRadius:'0px'}} type='text' 
placeholder = 'Add Task . . .' value = {task} onChange = 
{event=>Settask(event.target.value)}/>
            <button onClick = {() =>addtask()}>Add Task</button>
        </div>
    )
}

export default Addtask
```

Addtask.css:

```
.addtask{
    display: flex;
    justify-content: center;
    align-items: center;
    margin-top: 0px;
    border-top: 2px solid rgba(0,0,0,0.1);
}

.addtask input[type=text]{
    width: 45%;
    height: 30px;
    outline: none;
    border: none;
    border-bottom: 2px solid black;
    font-size: 25px;
    padding: 20px;
    padding-bottom: 10px;
    color: rgb(62, 183, 192);
    font-weight: bold;
    margin-left:40px;
    margin-top: 50px;
}

.addtask input[type=text]::placeholder{
    color: rgb(223, 224, 224);
}

.addtask button{
    width: 10%;
    padding: 15.6px 19px;
    margin-left: 50px;
    margin-right: 0px;
    margin-top: 52px;
    border: none;
    outline: none;
    font-size: 18px;
    cursor: pointer;
    color: black;
    font-weight: bolder;
    border-radius: 6px;
    background: rgb(79, 218, 242);
    background-size: 200% 200%;
}
```

Todolist.js:

```
import './Todolist.css'
import React from 'react'
import CheckIcon from '@mui/icons-material/Check';
import EditIcon from '@mui/icons-material/Edit';
import CloseIcon from '@mui/icons-material/Delete';

import axios from 'axios'
functionTodolist(props) {
    consttodolist = props.todolist.map((task,index) => {
        consttaskComplete = task=> {

axios.put(`http://localhost:8000/api/tasks/${task._id}` , {
                _id :task._id,
                todo: task.todo,
                desc: task.desc,
                date: task.date,
                isComplete : !task.isComplete

}).then(res=>props.taskComplete(res.data)).catch(err=>
console.log(err))
        }
        constremoveTask = id=> {

axios.delete(`http://localhost:8000/api/tasks/${id}`).then(res=>
props.removeTask(res.data)).catch(err=> console.log(err))
        }
        return <li key = {index}>
            <div style = {{display :
'flex',flexDirection:'row'}}>
                <CheckIconclassName = {task.isComplete ?
'isComplete' : 'checkicon'}/>
                <p className = {task.isComplete ? 'taskcomplete'
: ''} onClick = {() => {
```

```
                   taskComplete(task)
               }}>
                    <div style={{fontSize:'30px',fontFamily:
'Book Antiqua',fontWeight: '1000px', color:
'#040669'}}>{task.todo.toUpperCase()}</div>
                    <br/>
                    <div style={{fontSize:'22px',fontWeight:
'normal',fontFamily: 'Century Gothic'}}>{task.desc}</div>
                    <br/>
                    <div
style={{fontWeight:'normal',fontSize:'14px', color:
'#040669'}}>{task.date}</div>
                </p>
            </div>
            <div>
                <EditIconclassName = { task.isComplete ? 'edit
disappear' : 'edit' } onClick = {() => {
                    props.tasktoUpdate(task)
                    props.showPopup()
                }}/>
                <CloseIconclassName = 'close' onClick = {() => {
                    removeTask(task._id)
                }}/>
            </div>
        </li>
    })
    return (
        <div className = 'tasklist'>
            <ul>
                {todolist}
            </ul>
        </div>
    )
}

export default Todolist
```

Todolist.css:

```
.tasklist{
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    /* margin-top: 30px; */
    padding: 30px;
    /* margin-left: 80px;
    width: 90%;
    background-color: aqua; */
}

.tasklist ul{
    list-style: none;
    width: 70%;
}

.tasklist ul li{
    display: flex;
    justify-content: space-between;
    font-size: 20px;
    padding: 14px;
    font-weight: bolder;
    transition: .4s;
    user-select: none;
    border-radius: 20px;
    margin: 20px 0px;
}

.tasklist ul li .isComplete{
    visibility: visible;
    color: rgb(9, 108, 66);
    margin-right: 10px;
    transform: scale(1.2);
}

.tasklist ul li .checkicon{
    visibility: hidden;
    margin-right: 10px;
}

.tasklist ul li p.taskcomplete{
    text-decoration: line-through;
}
.disappear{
    visibility: hidden;
}
.tasklist ul li .edit, .tasklistul li .close{
    margin-left: 10px;
}

.tasklist ul li .edit:hover{
    background-color: rgb(7,94,233);
    color: #fff;
}
```

```
.tasklist ul li .close:hover{
    background-color: rgb(253,92,92);
    color: #fff;
}

.tasklist ul li:nth-child(odd){
    background-color: #c5f1fb;
}
.tasklist ul li:hover{
    background-color: rgb(79, 218, 242);
    cursor: pointer;
}
```

Updatetask.js:

```
import './Updatetask.css'
import React ,{useState} from 'react'
import axios from 'axios'
functionUpdatetask(props) {
    const [task,setTask] = useState(props.task.todo)
    const [descr,setDesc] = useState(props.task.desc)
    const [datee,setDate] = useState(props.task.date)
    constupdateTask = () => {
        if(task.trim() === ''){
            props.removePopup()
        } else {
            if((props.task.todo===task
&&props.task.desc===descr&&props.task.date!==datee) ||
(props.task.todo===task
&&props.task.desc!==descr&&props.task.date!==datee) ||
(props.task.todo!==task
&&props.task.desc===descr&&props.task.date!==datee) ||
(props.task.todo!==task
&&props.task.desc!==descr&&props.task.date!==datee) ||
(props.task.todo===task
&&props.task.desc!==descr&&props.task.date===datee) ||
(props.task.todo!==task
&&props.task.desc===descr&&props.task.date===datee) ||
(props.task.todo!==task
&&props.task.desc!==descr&&props.task.date===datee))

axios.put(`http://localhost:8000/api/tasks/${props.task._id}`,{
                _id :props.task._id,
                todo : task,
                desc :descr,
                date :datee,
                isComplete :props.task.isComplete
            }).then(res=> {
                props.removePopup()
                props.updatetask(res.data)
            }).catch(err=> console.log(err))
            else
                props.removePopup()
        }
    }
    return (
        <div className = 'popup'>
            <div className = 'popup-content'>
                <input type = 'text'  placeholder = 'Update Task
. . .' value = {task} onChange =
{event=>setTask(event.target.value)}/>
                <input type = 'text'  placeholder =
'Description...' value = {descr} onChange =
{event=>setDesc(event.target.value)}/>
                <input type = 'date' min={new
Date().toISOString().split('T')[0]}  placeholder = 'Date' value
= {datee} onChange = {event=>setDate(event.target.value)}/>
                <button onClick = {()
=>updateTask()}>Update</button>
            </div>
        </div>
    )
}

export default Updatetask
```

Updatetask.css:

```
.popup{
    background-color: rgba(0,0,0,0.6);
    width: 100%;
    height: 100%;
    position: fixed;
    top: 0%;
    display: flex;
    justify-content: center;
    align-items: center;
}

.popup .popup-content{
    width: 500px;
    background-color: #fff;
    padding: 20px;
    border: 4px double black;
    border-radius: 5px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    box-shadow: 0 0 15px 4px rgb(0,0,0);
}

.popup .popup-content input[type=date],[type=text]{
    width: 90%;
    padding: 20px;
    margin: 10px 0px;
    height: 10px;
    outline: none;
    border: 1.5px solid rgb(79, 218, 242);
    border-radius: 6px;
    font-family:cursive;
}

.popup .popup-content button{
    margin-top: 20px;
    padding: 10px 5px;
    cursor: pointer;
    background-color: rgb(79, 218, 242);
    color: black;
    border: none;
    outline: none;
    border-radius: 10px;
    width:100px;
    font-weight: bold;
    font-family: cursive;
}
```

App.js:

```
import React , {useState , useEffect} from 'react'
import axios from 'axios'
import './App.css'
import Addtask from './Components/Addtask'
import Todolist from './Components/Todolist'
import Updatetask from './Components/Updatetask'
import LibraryBooksOutlinedIcon from '@mui/icons-
material/LibraryBooksOutlined';

functionApp() {
  const [todolist,setTodolist] = useState([])
  const [tasktoUpdate ,setTasktoUpdate] = useState({})
  const [showPopup,setShowPopup] = useState(false)
  useEffect(() => {
    axios.get('http://localhost:8000/api/tasks').then(res=> {
      setTodolist(res.data)
    }).catch(err=> console.log(err))
  },[])
  constaddTask = newTask=> {
    setTodolist([newTask,...todolist])
  }
  consttaskComplete = task=> {
    constnewList = [...todolist]
    newList.forEach(item=> {
      if(item._id === task._id){
        item.isComplete = task.isComplete
      }
    })
    setTodolist(newList)
  }
  constremoveTask = task=> {
    constnewList = todolist.filter(item=> !(item._id ===
task._id))
    setTodolist(newList)
  }
  constupdatetask = task=> {
    constnewList = [...todolist]
    newList.forEach(item=> {
      if(item._id === task._id){
        item.todo = task.todo;
        item.desc = task.desc;
        item.date = task.date;
      }
    })
    setTodolist(newList)
  }
  return (
    <div>
      <div style={{display: 'flex', flexDirection: 'row'}}>
      <LibraryBooksOutlinedIconclassName = 'book' />
      <span style={{fontSize:'45px',fontFamily:'sans-
serif',marginRight:'100px', padding: '25px 0 25px
55px'}}>NOTES</span>
      </div>

      <AddtaskaddTask = {addTask}/>
      <Todolisttodolist = {todolist} taskComplete =
{taskComplete} removeTask = {removeTask} tasktoUpdate =
{task=>setTasktoUpdate(task)} showPopup = {()
=>setShowPopup(!showPopup)}/>
      {showPopup&&<Updatetask task = {tasktoUpdate} updatetask =
{updatetask} removePopup = {() =>setShowPopup(!showPopup)}/>}
    </div>
  )
}

export default App
```

App.css:

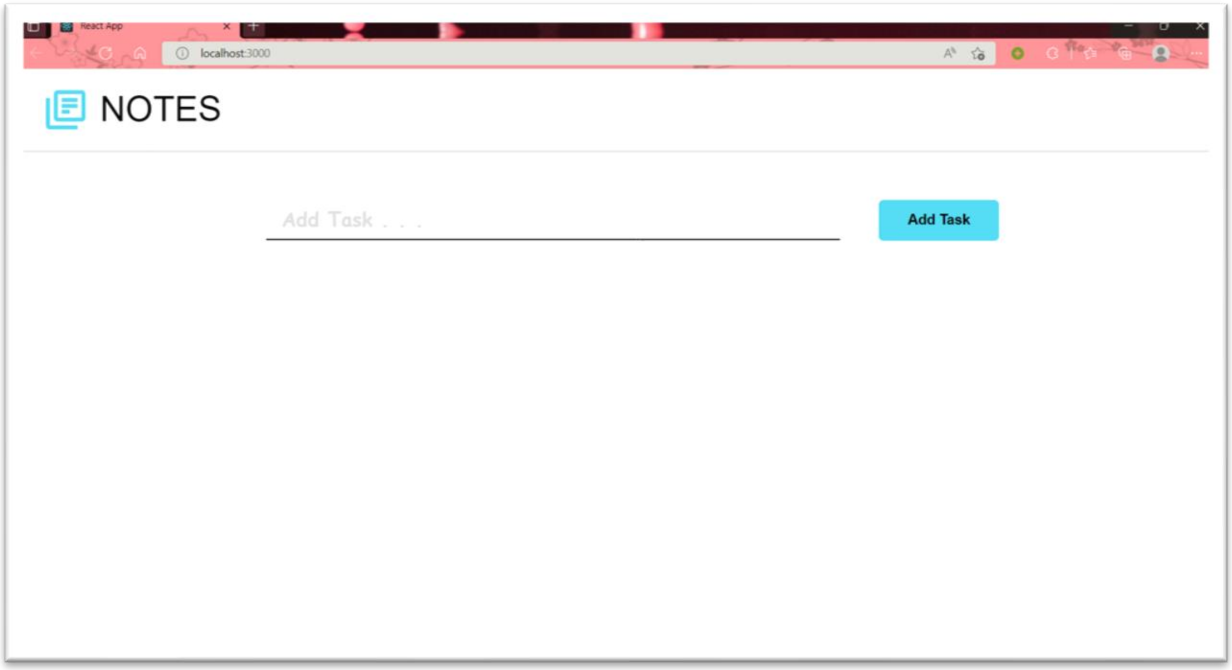
```
.book{
  color: rgb(79, 218, 242);
  padding: 40px 0 40px 30px;
  /* margin-left: -150px; */
  /* margin-right: 20px; */
  transform: scale(2.5);
}
```

Index.js:

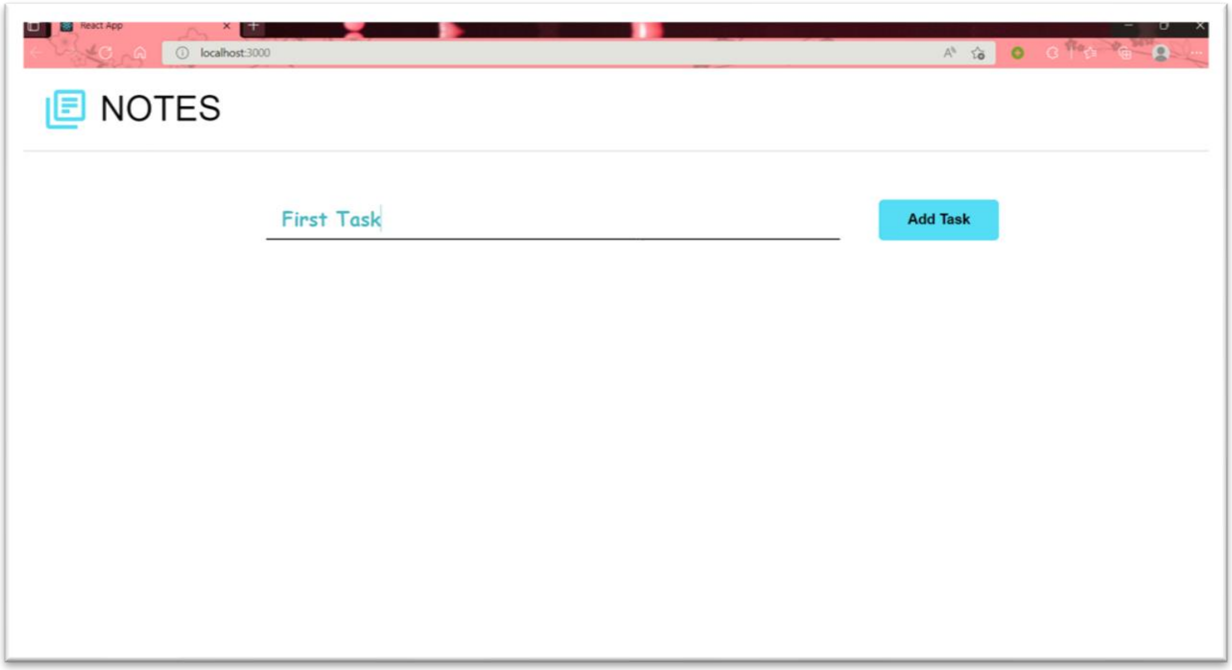
```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root =
ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

OUTPUT

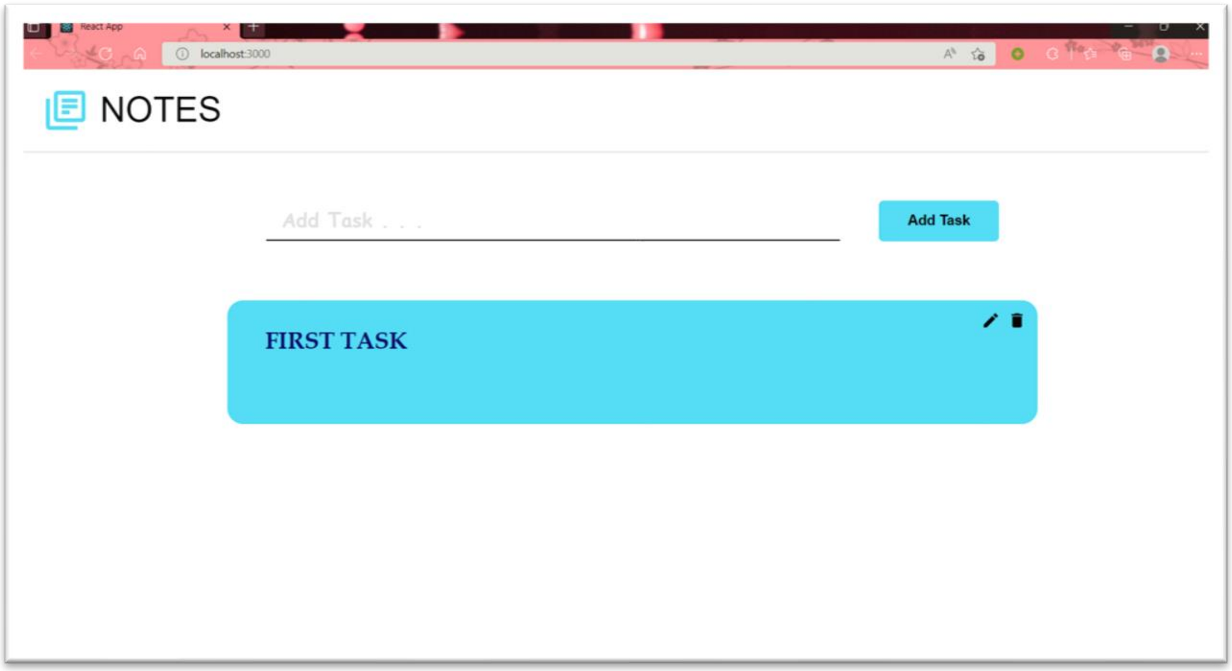
The main page with no tasks yet.



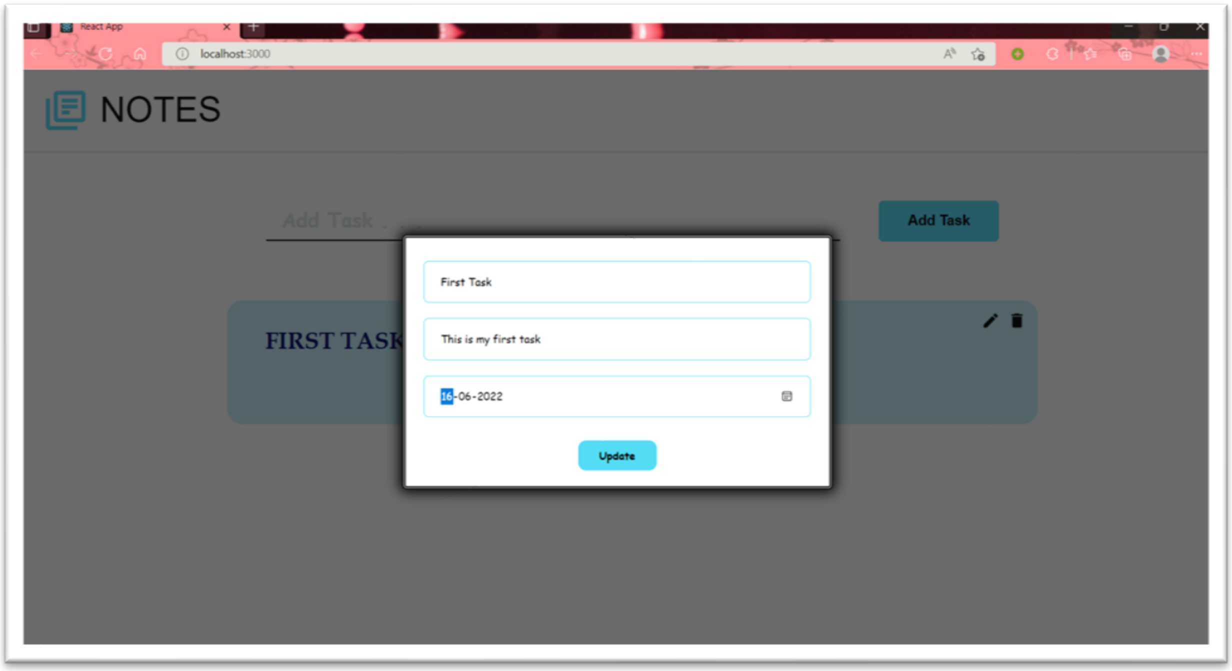
Entering first task name.



After adding a task with no extra information.



Giving description & due date to the newly added task.

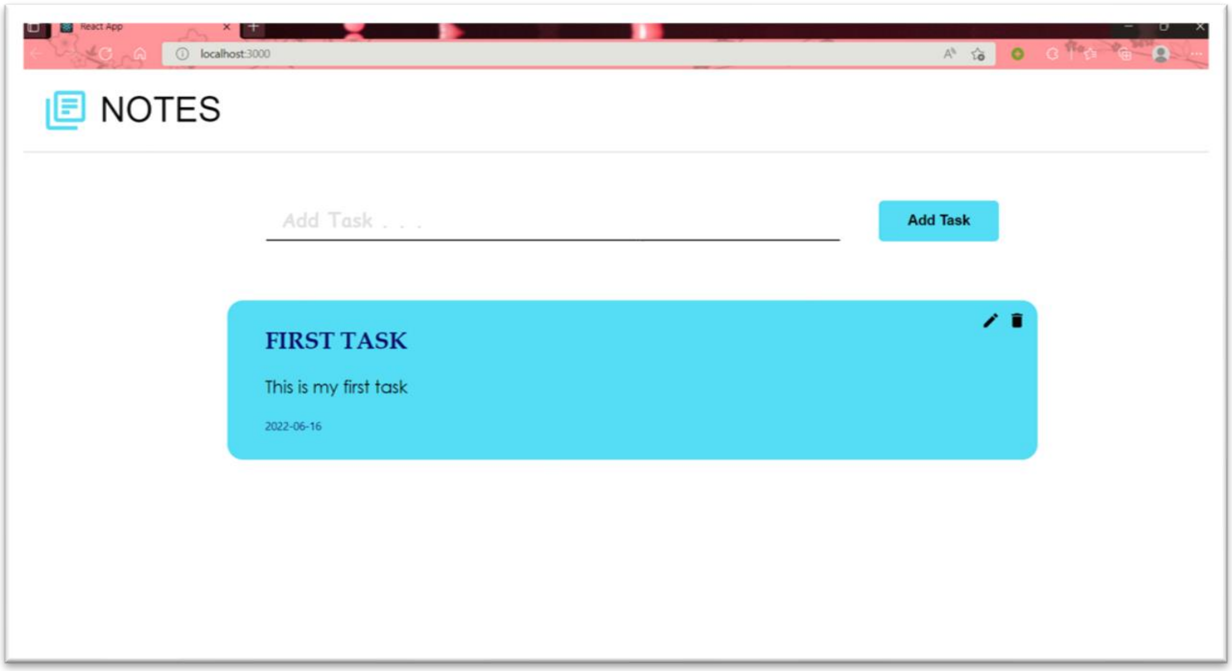


First Task

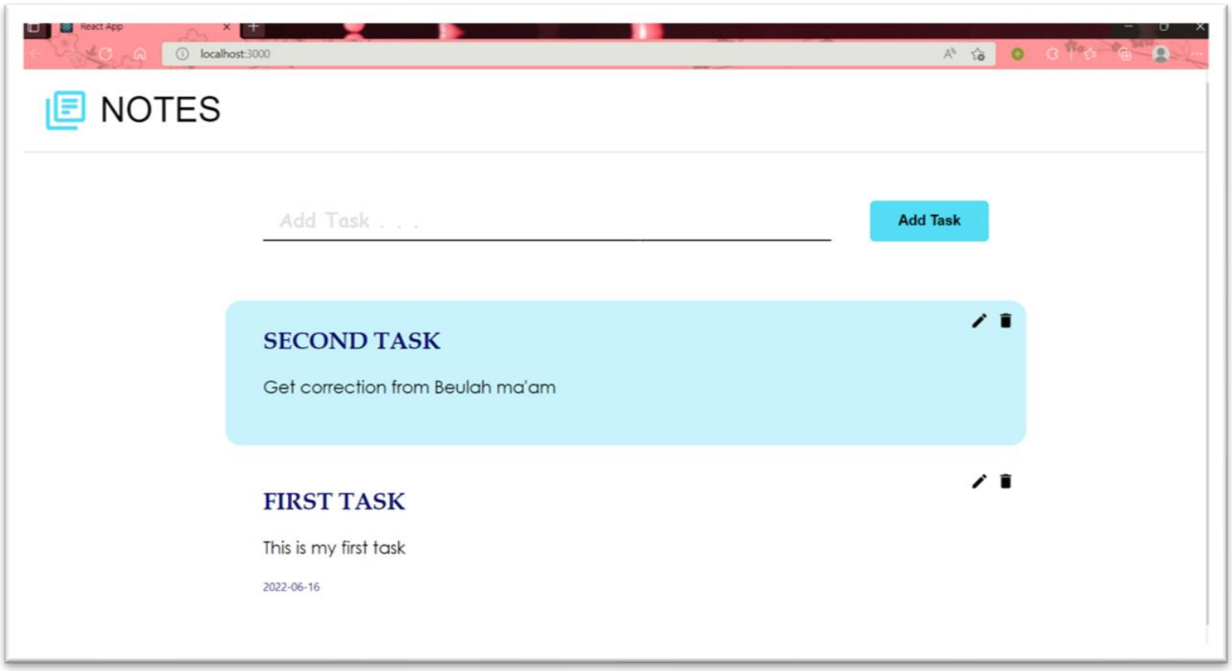
This is my first task

16-06-2022

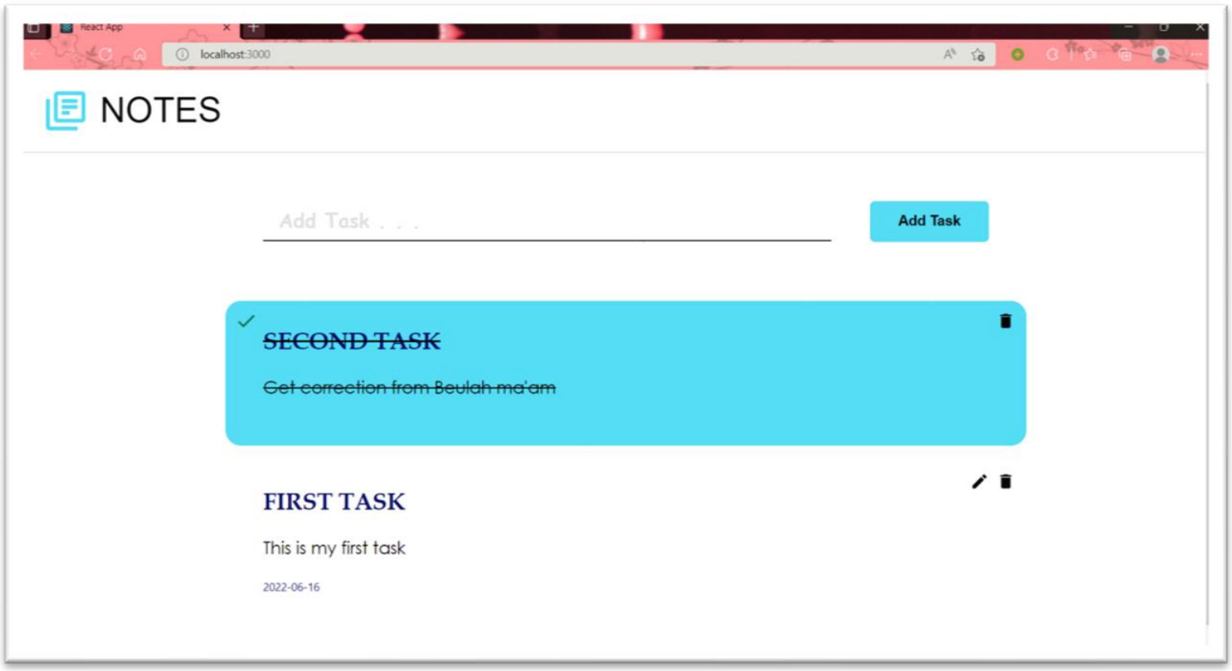
Added information to the newly added task.


React App
localhost:3000
NOTES
Add Task . . .
Add Task
FIRST TASK
This is my first task
2022-06-16

Added another task without due date.


React App
localhost:3000
NOTES
Add Task . . .
Add Task
SECOND TASK
Get correction from Beulah ma'am
FIRST TASK
This is my first task
2022-06-16

To indicate that a task is completed, its contents are stricken off.



~~SECOND TASK~~

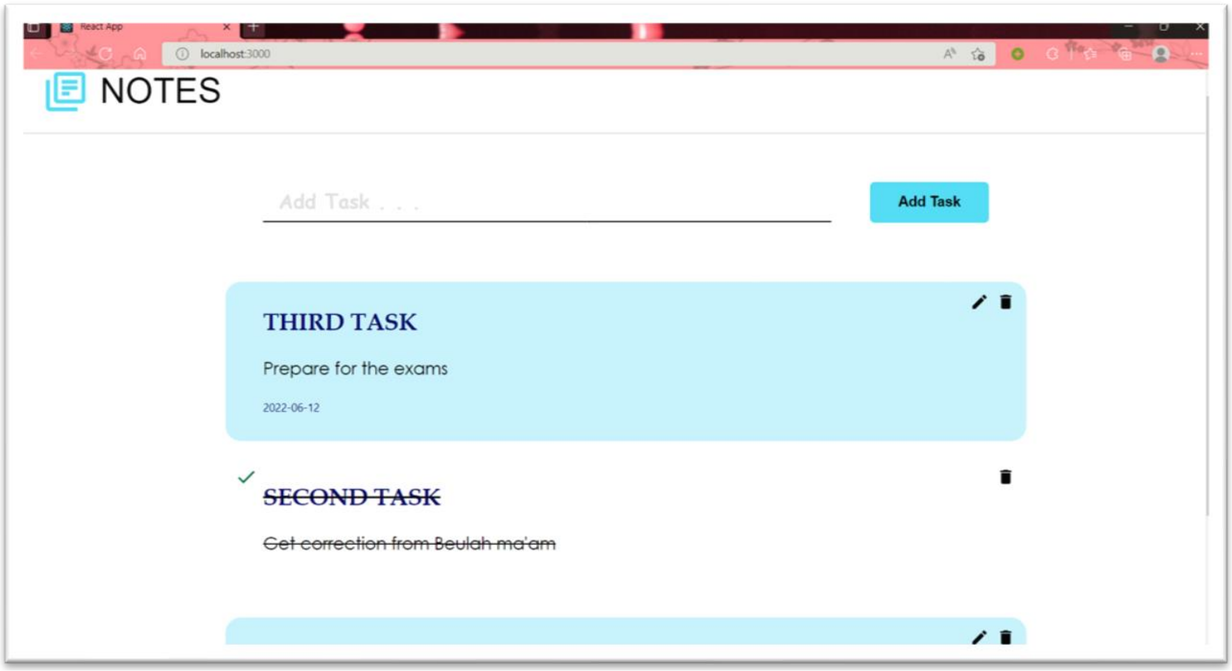
~~Get correction from Beulah ma'am~~

FIRST TASK

This is my first task

2022-06-16

Added another task.



THIRD TASK

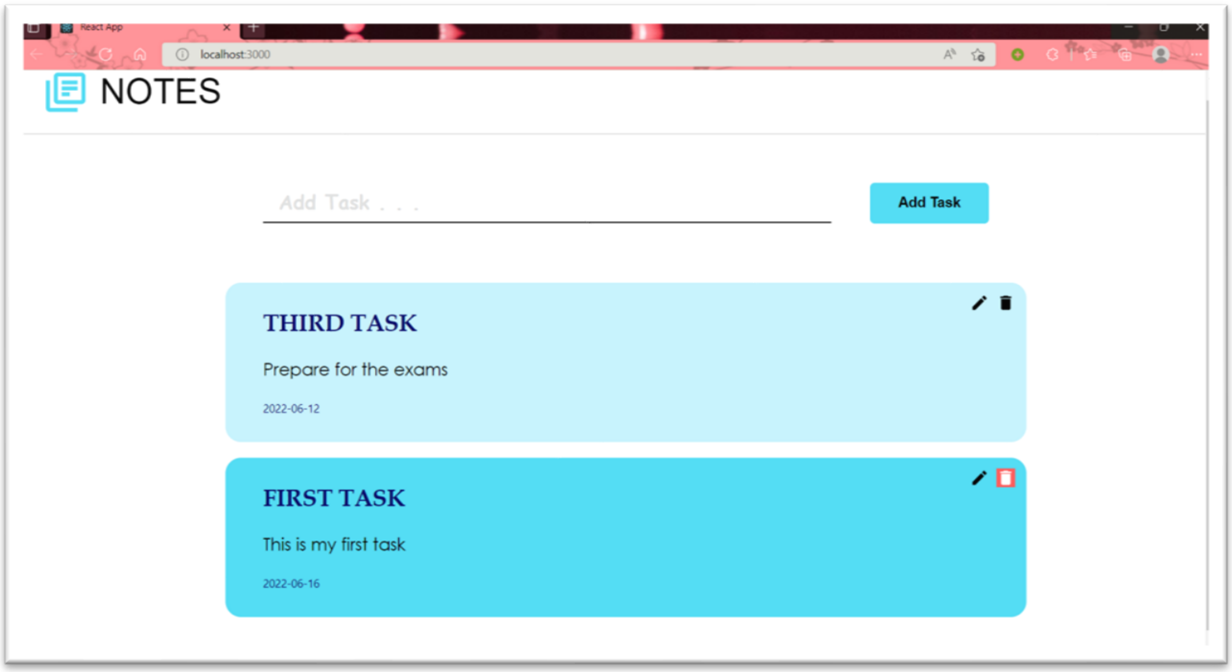
Prepare for the exams

2022-06-12

~~SECOND TASK~~

~~Get correction from Beulah ma'am~~

Deleted the completed task.



THIRD TASK

Prepare for the exams

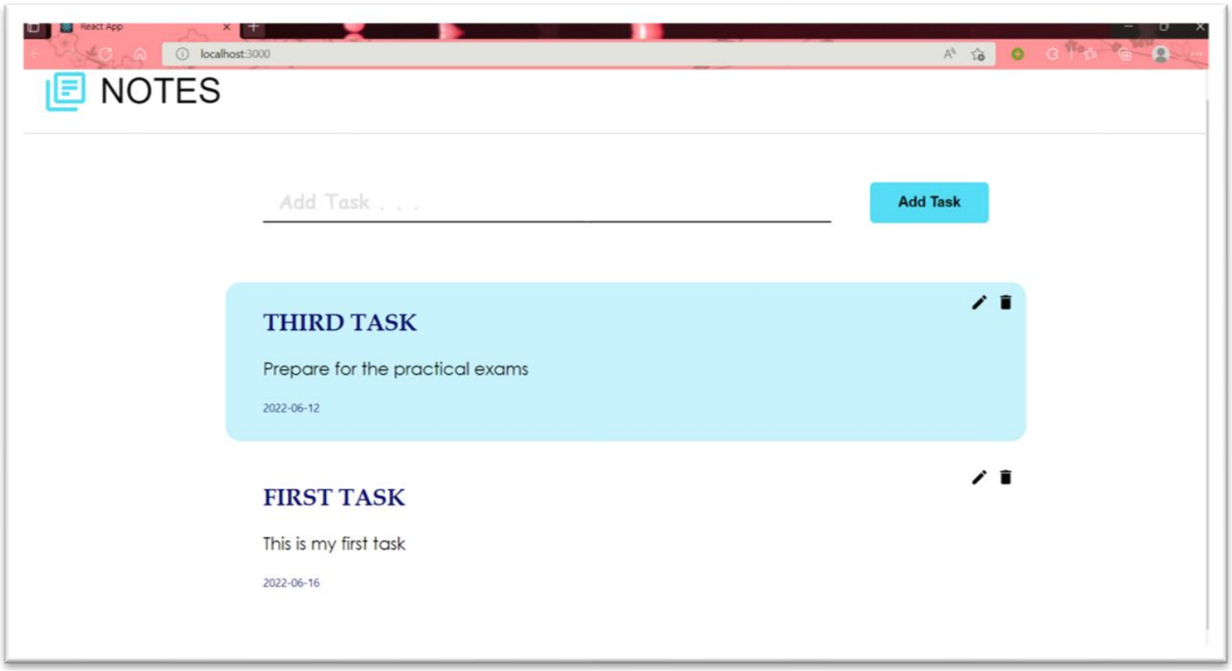
2022-06-12

FIRST TASK

This is my first task

2022-06-16

Edited an existing task.



THIRD TASK

Prepare for the practical exams

2022-06-12

FIRST TASK

This is my first task

2022-06-16

USAGE OF NEW TOOLS

Express

Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application.

Mongoose

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB. MongoDB is a schema-less NoSQL document database.

Cors

Cross-Origin Resource Sharing (CORS) is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources.

Nodemon

Nodemon is a command-line interface (CLI) utility that wraps your Node app, watches the file system, and automatically restarts the process.

Axios

Axios is a promise-based HTTP Client for node.js and the browser. It is isomorphic. It can run in the browser and nodejs with the same codebase. On the server-side it uses the native node.js http module, while on the client (browser) it uses XMLHttpRequests.

FEATURES NOT USED IN PROJECT

React Routers

React routers are not used in our project, because our project is a single page application and so we have not used react routers. In future, if we update our webpage as multipage application, it can be used.