

**UCS 2611 Internet Programming Lab**  
**Exercise 12. Full Stack Todo Web Application using ReactJS, Node and MongoDB**


Develop a **Todo Web Application** that enables the user to maintain and keep track of their Todo list [CO2, K3]

Draw a diagram depicting the design flow of your project with necessary components, endpoints and collections

Application has the following:

- Front end to provide necessary requests using ReactJS
- Maintain the details in MongoDB
- Define necessary functionalities in the required components and endpoints in node server to display the Todo application like below:

### TodoInput



 New Todo

Add new task



### TodoList

AllDoneTodo



~~Learn ReactJS basics~~

☒  



Practice ReactJS

☐  



Learn Redux

☐  

Code portfolio in React

☐  

Learn React Native

☐  

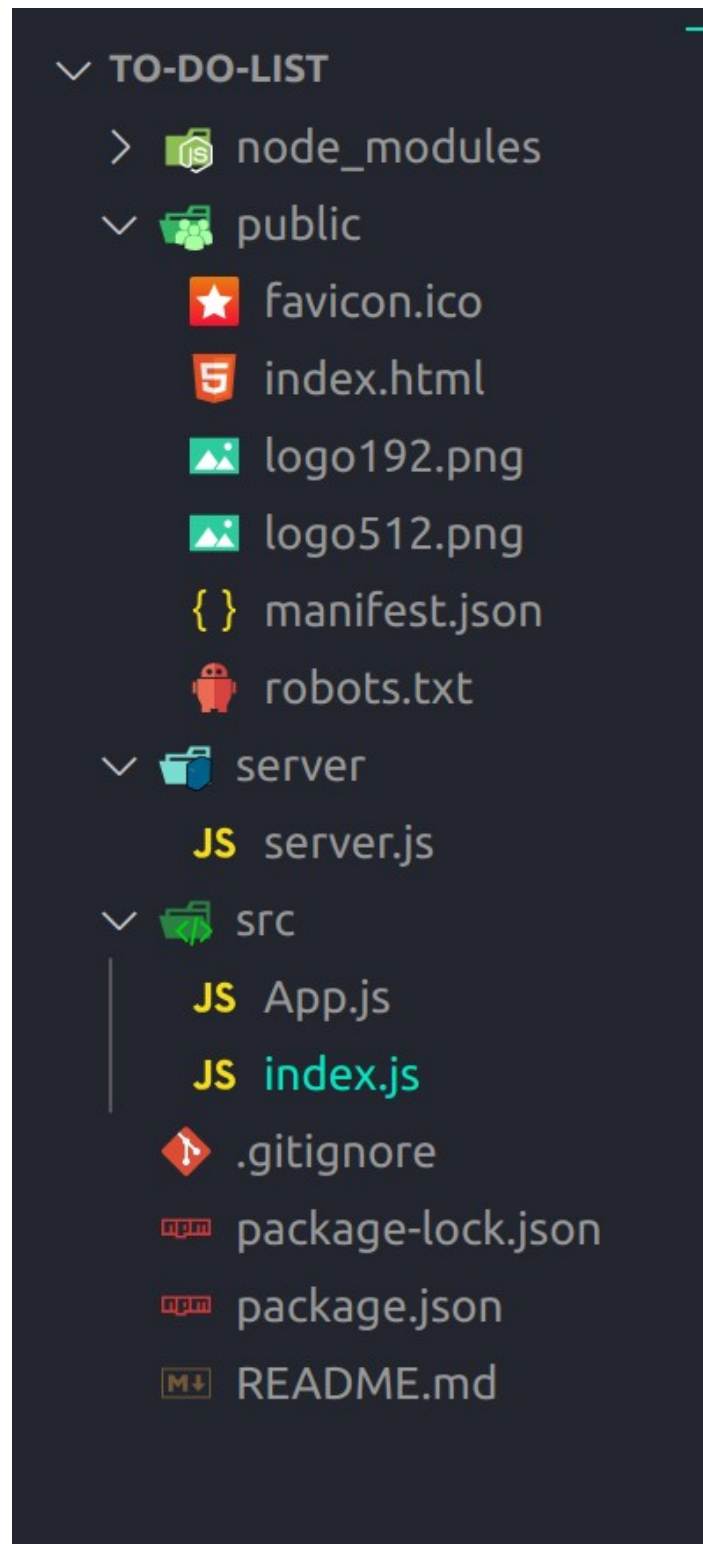
Delete done tasks

Delete all tasks

Best Practices to be followed:

1. Design before coding
2. Incremental coding
3. Usage of proper naming convention
4. Usage of Comments to the code
5. Indentation of code

**Hierarchy:**



## **Program code:**

### **server.js**

```
const express = require('express');
const bp = require("body-parser");
const cors = require('cors');
const { MongoClient } = require('mongodb');

const server_name = express();
server_name.use(cors());
server_name.use(bp.json());

// Database Config
const client = new MongoClient("mongodb://localhost:27017");
client.connect();
const db = client.db("learning");
const usersCollection = db.collection("todo");
const completedCollection = db.collection("completed");
console.log("Database Connected Successfully....");

// AddTask END POINT
async function AddTask(newTask){
  const result = await usersCollection.insertOne({ task: newTask });
  console.log("Task inserted successfully:", result.insertedId);
}
server_name.post('/addTask', (req, res)=>{
  const data = req.body;

  AddTask(data.task);

  res.status(200).send("success");
});

// ReadTask END-POINT
async function ReadTask(){
  console.log('read function is called');
  try {
    const cursor = usersCollection.find({});
    const tasks = await cursor.toArray();
    const result = [];
    for(var i=0; i<tasks.length; i++)
      result.push(tasks[i].task)

    return result;
  } catch (error) {
    console.error("Error reading tasks:", error);
    return [];
  }
}
server_name.get('/readTask', async (req, res) => {
```

```

    try {
        const tasks = await ReadTask();
        console.log(tasks);
        res.status(200).json({task:tasks});
    } catch (error) {
        res.status(500).send("Internal server error");
    }
})

```

// DeleteTask END-POINT

```

async function DeleteTask(pos){
    try {
        const tasks = await ReadTask();
        if (pos >= 0 && pos < tasks.length) {
            const taskToDelete = tasks[pos];
            await usersCollection.deleteOne({ task: taskToDelete });
        } else {
            console.log("Invalid position:", pos);
        }
    } catch(error){
        console.error("Error in deleting record in DB:", error);
    }
}

```

```

server_name.post('/deleteTask', (req, res)=>{
    try{
        const data = req.body;

        DeleteTask(data.index);

        res.status(200).send("success");
    }catch(error){
        res.status(402).send("Delete Error");
    }
})

```

// MoveTask END-POINT

```

async function MoveTask(pos) {
    try {
        const tasks = await ReadTask();
        if (pos >= 0 && pos < tasks.length) {
            const taskToMove = tasks[pos];
            const result = await completedCollection.insertOne({ task: taskToMove });
            console.log("Task moved to completed collection:", result.insertedId);
            await DeleteTask(pos);
        } else {
            console.log("Invalid position:", pos);
        }
    } catch (error) {
        console.error("Error moving task:", error);
    }
}

```

```

server_name.post('/moveTask', (req, res) =>{
  try {
    const data = req.body;
    const index = data.index;

    MoveTask(index);

    res.status(200).send("success");
  } catch (error) {
    console.error("Move Error:", error);
    res.status(500).send("Move Error");
  }
})

// ReadCompletedTask END-POINT
async function ReadCompletedTask() {
  try {
    const cursor = completedCollection.find({});
    const tasks = await cursor.toArray();
    const result = tasks.map(task => task.task);
    return result;
  } catch (error) {
    console.error("Error reading completed tasks:", error);
    return [];
  }
}

server_name.get('/readCompletedTask', async (req, res) => {
  try {
    const completedTasks = await ReadCompletedTask();
    res.status(200).json({ completedTasks });
  } catch (error) {
    console.error("Error fetching completed tasks:", error);
    res.status(500).send("Internal server error");
  }
});

server_name.listen(4000, ()=>{
  console.log("Server Listening on http://localhost:4000");
})

```

### **App.js**

```

import { useState, useRef, useEffect } from "react";
const SERVER_URL = "http://localhost:4000"
function App() {

  const userInput = useRef("");
  const [tasks, setTasks] = useState([]);
  const [completedTasks, setCompletedTasks] = useState([]);

```

```

const addTask = () => {
console.log("AddTask() Function is called");
const newTask = userInput.current.value;
console.log("New Task Added: " + newTask);

// ADD API "http://localhost:4000/addTask"
fetch(SERVER_URL+'/addTask', {
  method: 'POST',
  headers: {
    'Content-Type': "application/json"
  },
  body: JSON.stringify({ task: newTask })
})
.then(res => res.text())
.then(data => readTask())
.catch(error => console.log("ADD ERROR: " + error));

// readTask();
}

// READ API "http://localhost:4000/readTask"
const readTask = () =>{
  console.log("ReadTask() function is called");
fetch(SERVER_URL+"/readTask", {
  method: 'GET'
})
.then(res => res.json())
.then(data => {
  console.log(data.task);
  setTasks(data.task);
})
.catch(e => console.log("READ ERROR: " + e));
}

// Delete API "http://localhost:4000/deleteTask"
const deleteTask = (taskIndex) =>{
  console.log("DeleteTask() function is called");
  console.log("data: " + taskIndex);
  fetch(SERVER_URL+'/deleteTask', {
    method: "POST",
    headers:{
      "Content-Type":"application/json"
    },
    body: JSON.stringify({index:taskIndex})
  })
  .then(res => res.text())
  .then(data => readTask())
  .catch(error => console.log("DELETE ERROR: " + error));
}

// Move API "http://localhost:4000/moveTask"

```

```

const moveTask = (taskIndex) => {
  console.log("DeleteTask() function is called");
  console.log("data: " + taskIndex);
  fetch(SERVER_URL + '/moveTask', {
    method: "POST",
    headers: {
      "Content-Type": "application/json"
    },
    body: JSON.stringify({index:taskIndex})
  })
  .then(res => res.text())
  .then(data => readTask())
  .catch(error => console.log("MOVE ERROR: " + error));
}

const readCompletedTask = () => {
  fetch(SERVER_URL + "/readCompletedTask")
    .then(res => res.json())
    .then(data => {
      console.log(data.completedTasks);
      setCompletedTasks(data.completedTasks);
    })
    .catch(e => console.log("READ COMPLETED ERROR: " + e));
};

useEffect(() => {
  readTask();
  readCompletedTask();
}, []);

const style = {
  input: {
    padding: '12px',
    fontSize: '20px',
    width: "300px"
  },
  btn: {
    padding: '10px',
    fontSize: '20px',
    fontWeight: "bold",
  },
  displayTask: {
    marginTop: "25px",
    marginLeft: "25px",
    fontSize: '22px',
    textAlign: 'left',
  },
  taskBtn: {
    border: '0',
    borderRadius: '5px',
    marginLeft: '10px',

```

```

        padding: '7px',
      },
      taskStyle: {
        padding: '10px',
        backgroundColor: 'lightblue',
        margin: '2px',
        borderRadius: '12px',
      },
    },
  },
  return (
    <div style={{textAlign:"center", padding:"5px"}} >
      <input type="text" ref={userInput} placeholder='Enter Your New Task'
style={style.input} /> <button type="submit" onClick={addTask} style={style.btn}> Add
Task</button>

      <h2> --- Your ToDo Task --- </h2>
      <section id="todo-task" style={style.displayTask}>
        {tasks.map((task, index) => (
          <div key={index} style={style.taskStyle}>
            {task}
            <span style={{float: "right"}}>
              <button type="submit"
onClick={()=>deleteTask(index)} style={style.taskBtn}></button>
              <button type="submit" onClick={() =>
moveTask(index)} style={style.taskBtn}> </button>
            </span>
          </div>
        ))}
      </section>
      <h2> --- Completed Task --- </h2>
      <section id="completed-task">
        {completedTasks.map((task, index) => (
          <div key={index} style={{backgroundColor:"lightgray",
padding:'10px', borderRadius:'15px', margin:'10px', marginLeft:'25px', width:"100px"}}>
            {task}
          </div>
        ))}
      </section>
    </div>
  );
}

export default App;

```

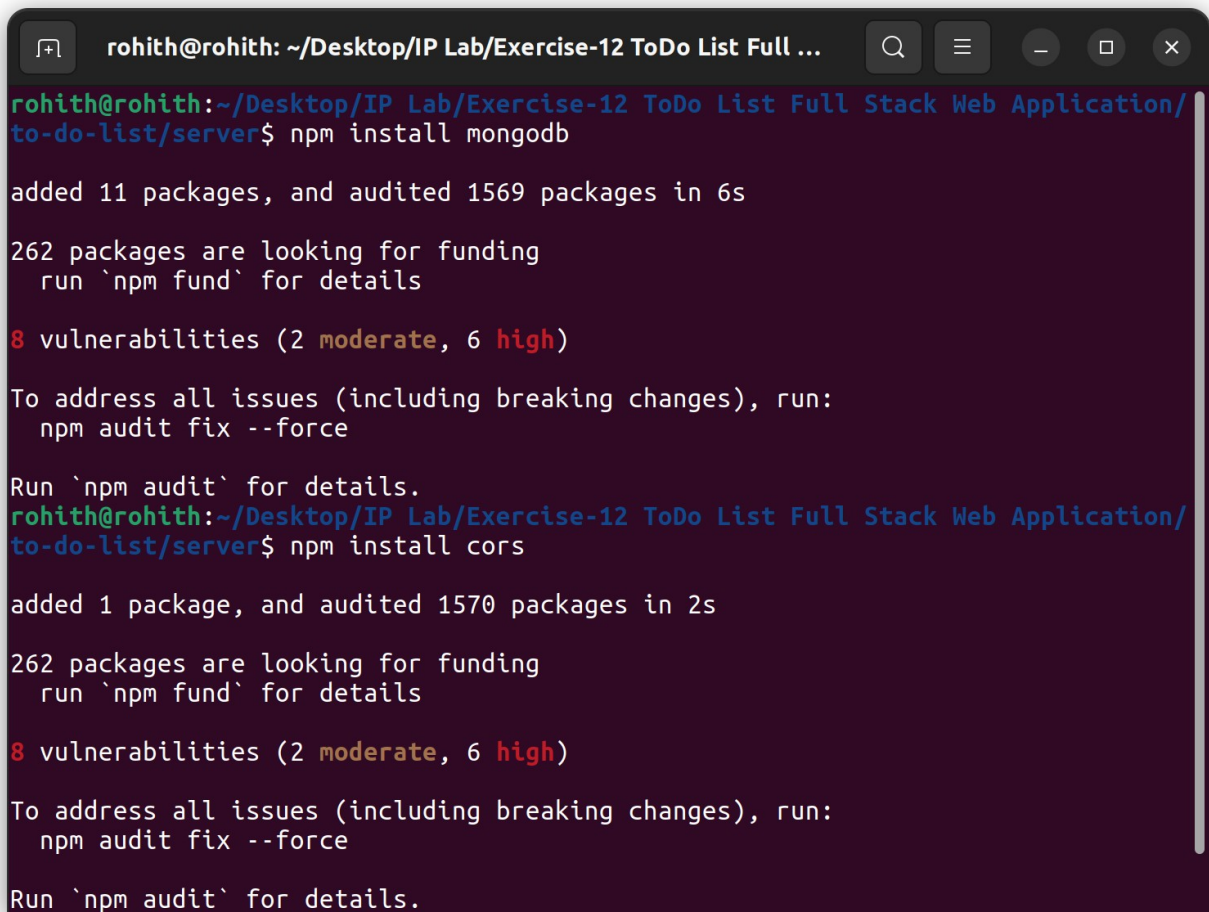


## index.js

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

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

## Output:

A terminal window with a dark background and light-colored text. The window title is "rohith@rohith: ~/Desktop/IP Lab/Exercise-12 ToDo List Full ...". The terminal shows two separate npm install commands and their outputs. The first command is "npm install mongodb", which results in 11 packages being added and 1569 packages audited in 6 seconds. It also shows 262 packages looking for funding and 8 vulnerabilities (2 moderate, 6 high). The second command is "npm install cors", which results in 1 package being added and 1570 packages audited in 2 seconds. It also shows 262 packages looking for funding and 8 vulnerabilities (2 moderate, 6 high). Both outputs include instructions to run "npm audit fix --force" to address all issues.

```
rohith@rohith: ~/Desktop/IP Lab/Exercise-12 ToDo List Full Stack Web Application/
to-do-list/server$ npm install mongodb

added 11 packages, and audited 1569 packages in 6s

262 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
rohith@rohith:~/Desktop/IP Lab/Exercise-12 ToDo List Full Stack Web Application/
to-do-list/server$ npm install cors

added 1 package, and audited 1570 packages in 2s

262 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

```
rohith@rohith: ~/Desktop/IP Lab/Exercise-12 ToDo List Full ...
262 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
rohith@rohith:~/Desktop/IP Lab/Exercise-12 ToDo List Full Stack Web Application/
to-do-list/server$ npm install cors

added 1 package, and audited 1570 packages in 2s

262 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
rohith@rohith:~/Desktop/IP Lab/Exercise-12 ToDo List Full Stack Web Application/
to-do-list/server$ node server.js
Database Connected Successfully....
Server Listening on http://localhost:4000
```

```
rohith@rohith: ~/Desktop/IP Lab/Exercise-12 ToDo List Full ...
rohith@rohith:~/Desktop/IP Lab/Exercise-12 ToDo List Full Stack Web Application/
to-do-list$ npm start
```



Add Task

--- Your ToDo Task ---

--- Completed Task ---

After typing the task, reload the page.



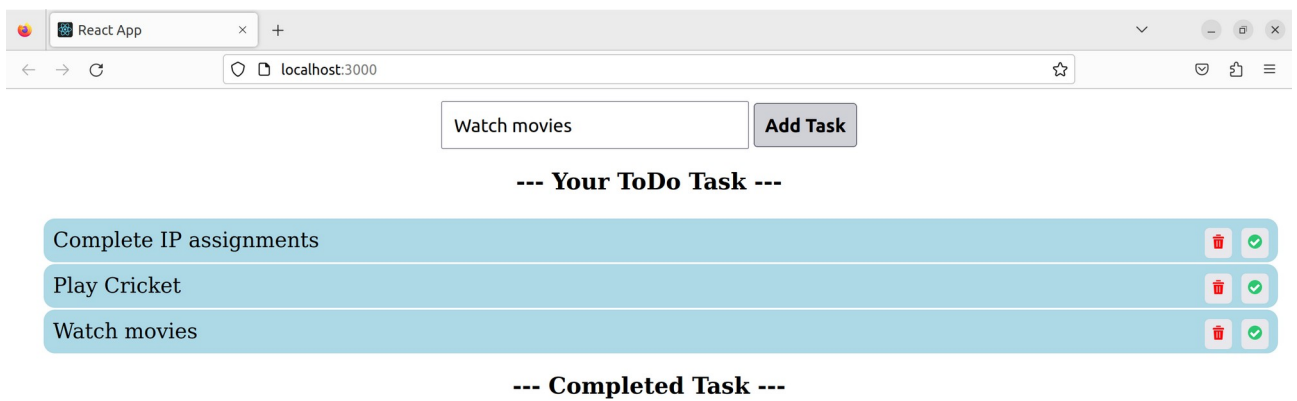
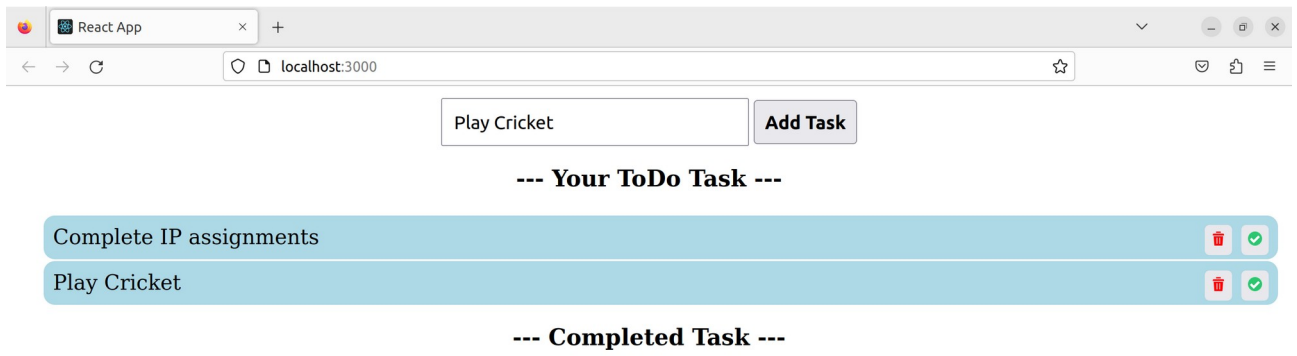
Add Task

--- Your ToDo Task ---

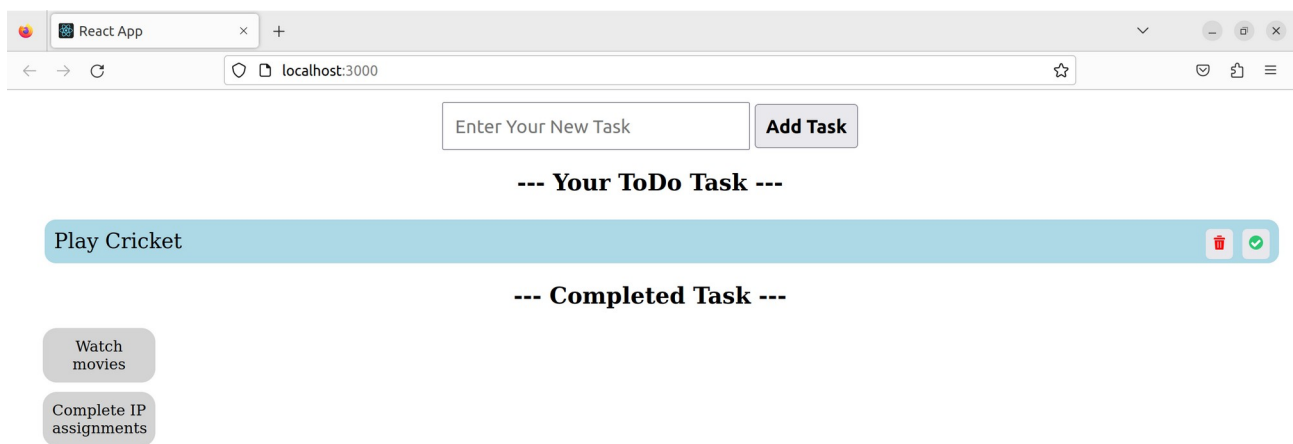
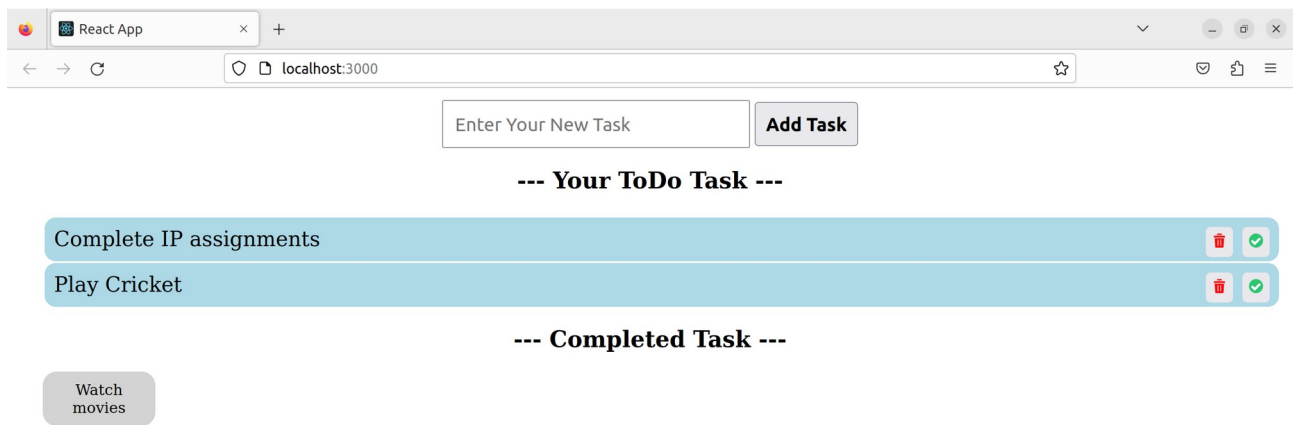
Complete IP assignments

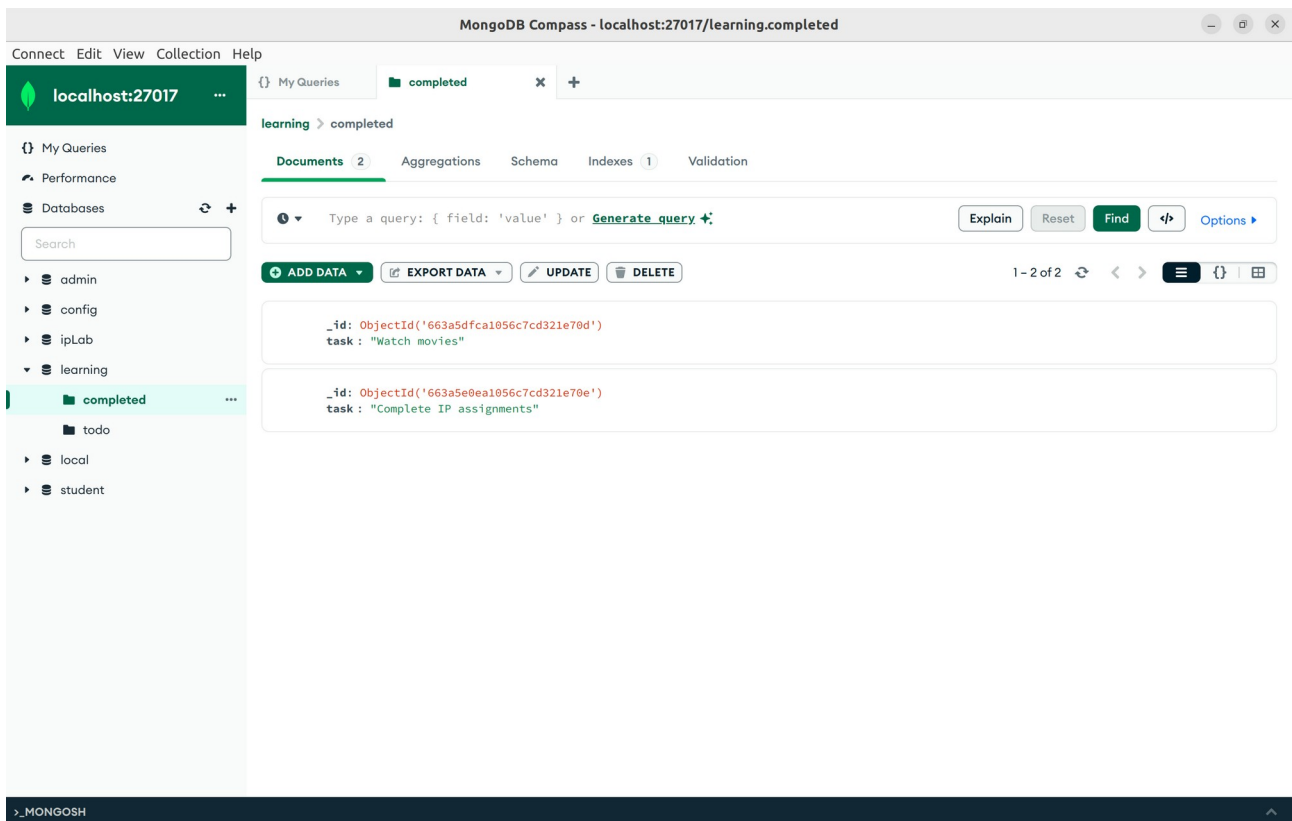
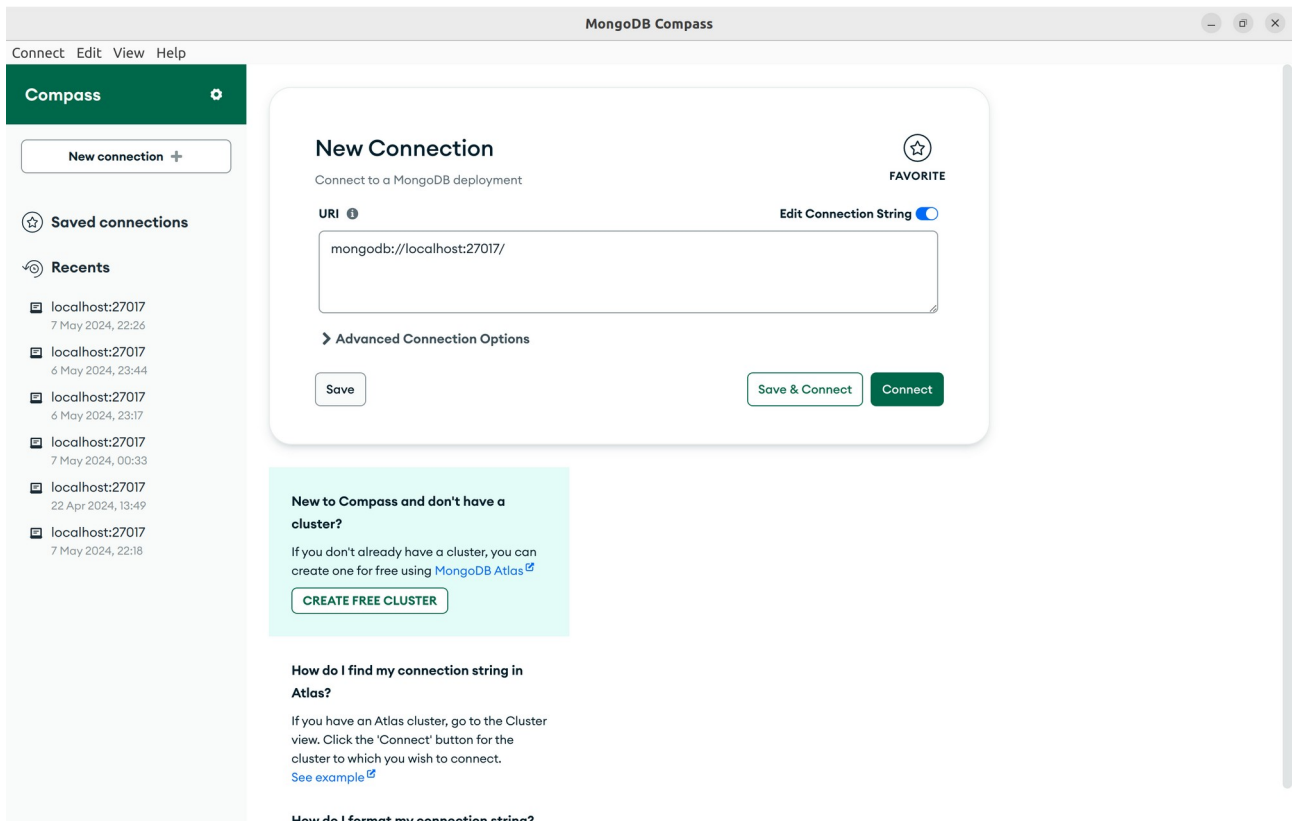


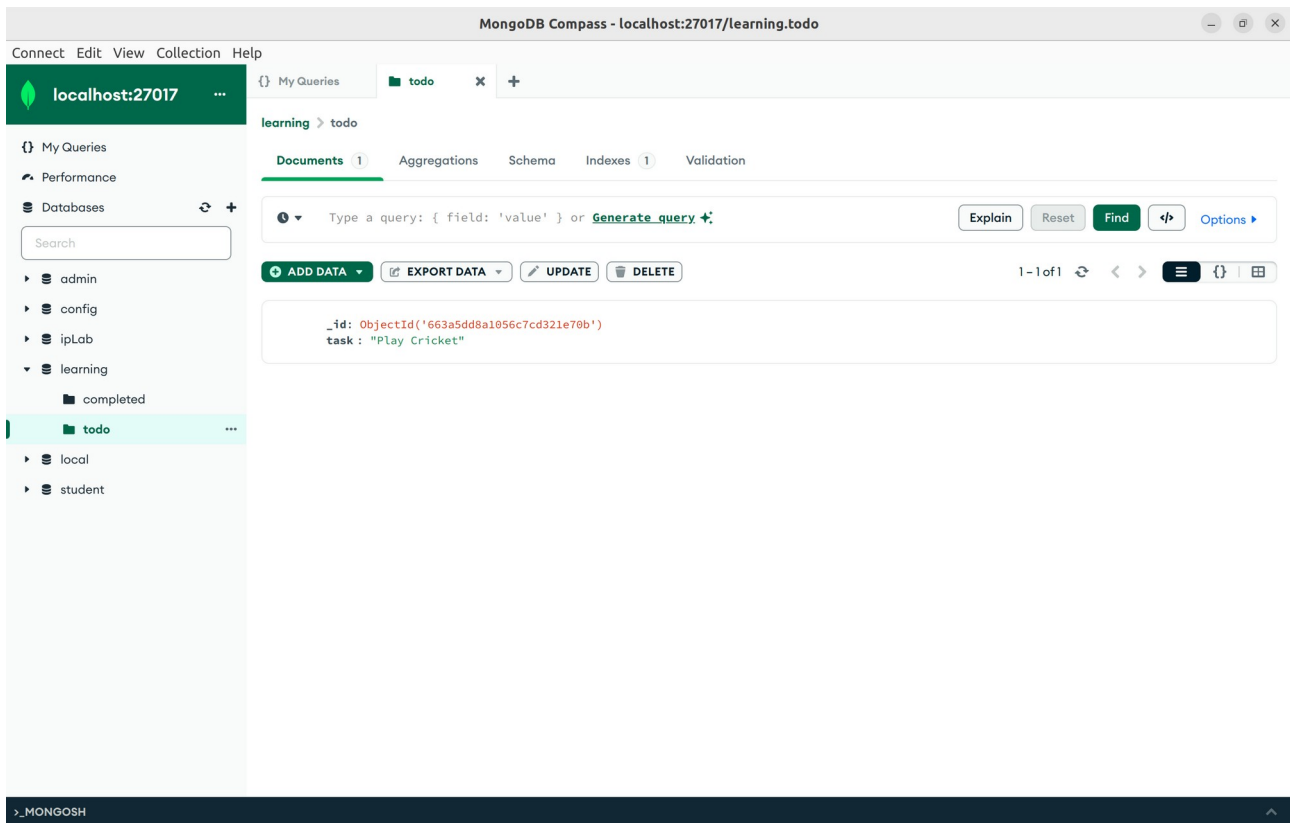
--- Completed Task ---



Once the task is completed by pressing the tick icon, reload the page.







```
rohith@rohith: ~/Desktop/IP Lab/Exercise-12 ToDo List Full ...  
Compiled successfully!  
  
You can now view to-do-list in the browser.  
  
Local:      http://localhost:3000  
On Your Network: http://10.17.34.88:3000  
  
Note that the development build is not optimized.  
To create a production build, use npm run build.  
  
webpack compiled successfully  
^C  
rohith@rohith:~/Desktop/IP Lab/Exercise-12 ToDo List Full Stack Web Application/  
to-do-list$
```

```
rohith@rohith: ~/Desktop/IP Lab/Exercise-12 ToDo List Full ...
[ 'Complete IP assignments' ]
Task inserted successfully: new ObjectId('663a5dd8a1056c7cd321e70b')
read function is called
[ 'Complete IP assignments', 'Play Cricket' ]
Task inserted successfully: new ObjectId('663a5deda1056c7cd321e70c')
read function is called
[ 'Complete IP assignments', 'Play Cricket', 'Watch movies' ]
read function is called
read function is called
[ 'Complete IP assignments', 'Play Cricket', 'Watch movies' ]
Task moved to completed collection: new ObjectId('663a5dfca1056c7cd321e70d')
read function is called
read function is called
[ 'Complete IP assignments', 'Play Cricket' ]
read function is called
[ 'Complete IP assignments', 'Play Cricket' ]
read function is called
Task moved to completed collection: new ObjectId('663a5e0ea1056c7cd321e70e')
read function is called
read function is called
[ 'Play Cricket' ]
read function is called
[ 'Play Cricket' ]
read function is called
[ 'Play Cricket' ]
^C
rohith@rohith:~/Desktop/IP Lab/Exercise-12 ToDo List Full Stack Web Application/
to-do-list/server$
```

Now running the todolist application again,



```
rohith@rohith: ~/Desktop/IP Lab/Exercise-12 ToDo List Full ...
[ 'Complete IP assignments', 'Play Cricket' ]
Task inserted successfully: new ObjectId('663a5deda1056c7cd321e70c')
read function is called
[ 'Complete IP assignments', 'Play Cricket', 'Watch movies' ]
read function is called
read function is called
[ 'Complete IP assignments', 'Play Cricket', 'Watch movies' ]
Task moved to completed collection: new ObjectId('663a5dfca1056c7cd321e70d')
read function is called
read function is called
[ 'Complete IP assignments', 'Play Cricket' ]
read function is called
[ 'Complete IP assignments', 'Play Cricket' ]
read function is called
Task moved to completed collection: new ObjectId('663a5e0ea1056c7cd321e70e')
read function is called
read function is called
[ 'Play Cricket' ]
read function is called
[ 'Play Cricket' ]
read function is called
[ 'Play Cricket' ]
^C
rohith@rohith:~/Desktop/IP Lab/Exercise-12 ToDo List Full Stack Web Application/
to-do-list/server$ node server.js
Database Connected Successfully....
Server Listening on http://localhost:4000
```

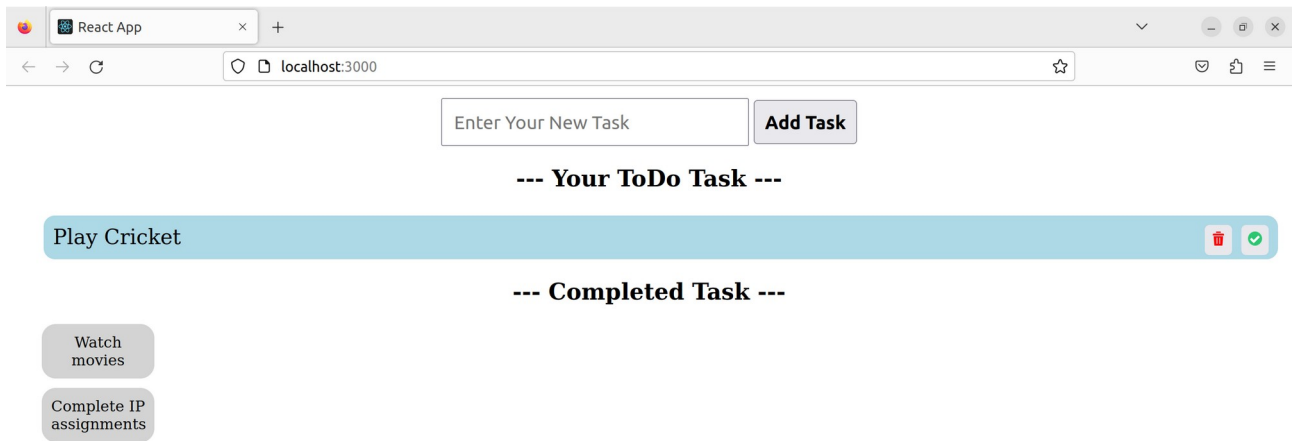
```
rohith@rohith: ~/Desktop/IP Lab/Exercise-12 ToDo List Full ...
Compiled successfully!

You can now view to-do-list in the browser.

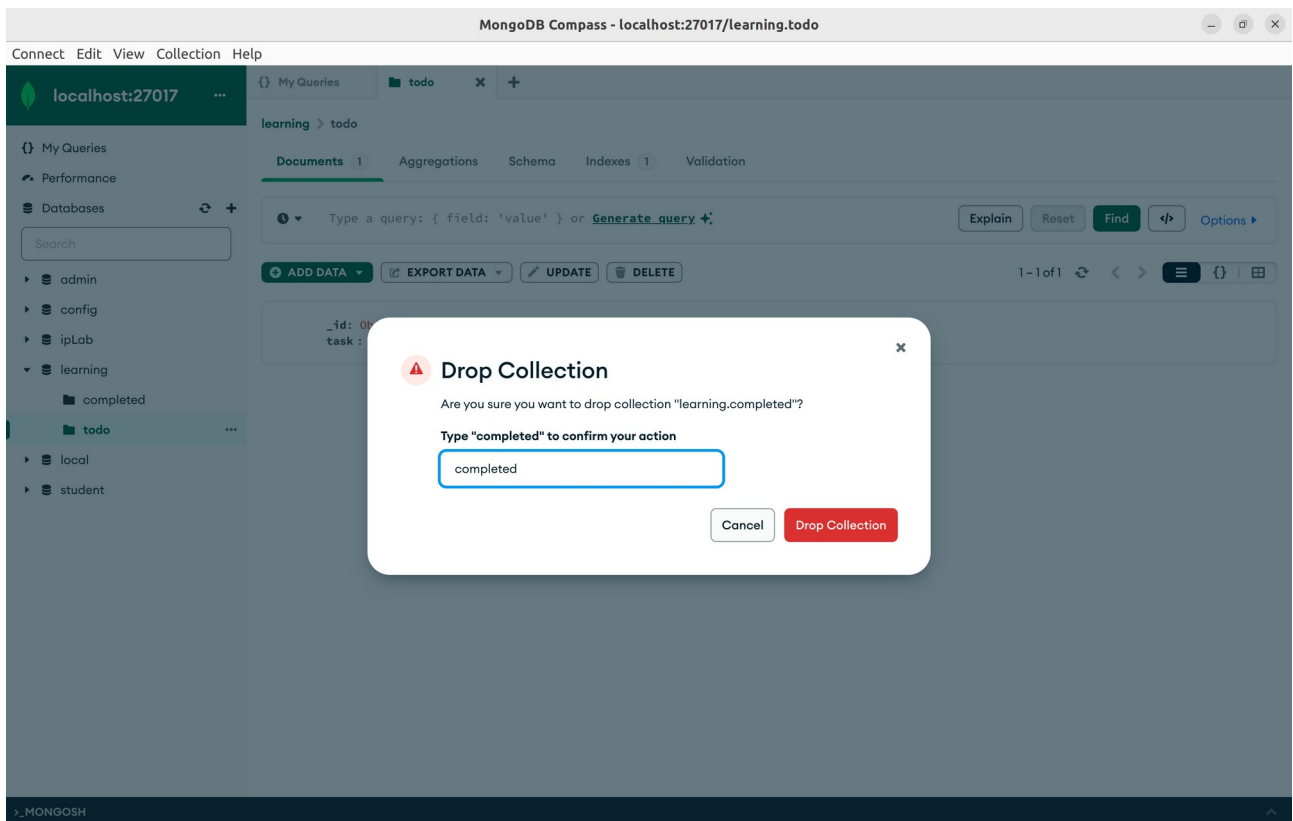
Local:           http://localhost:3000
On Your Network: http://10.17.34.88:3000

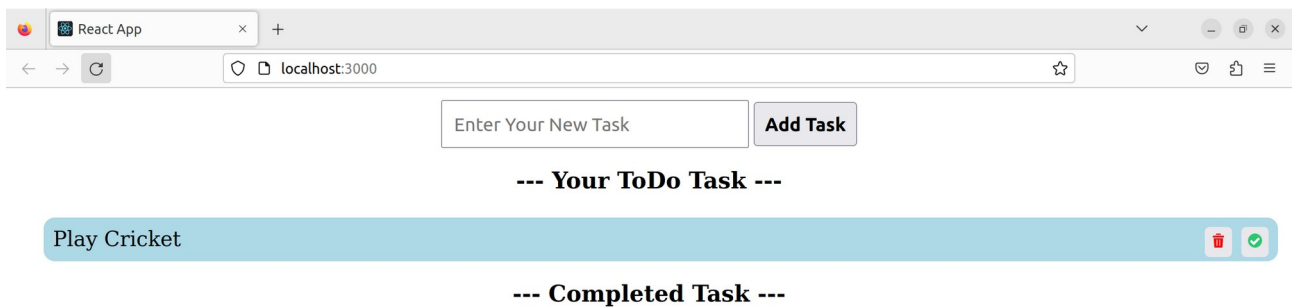
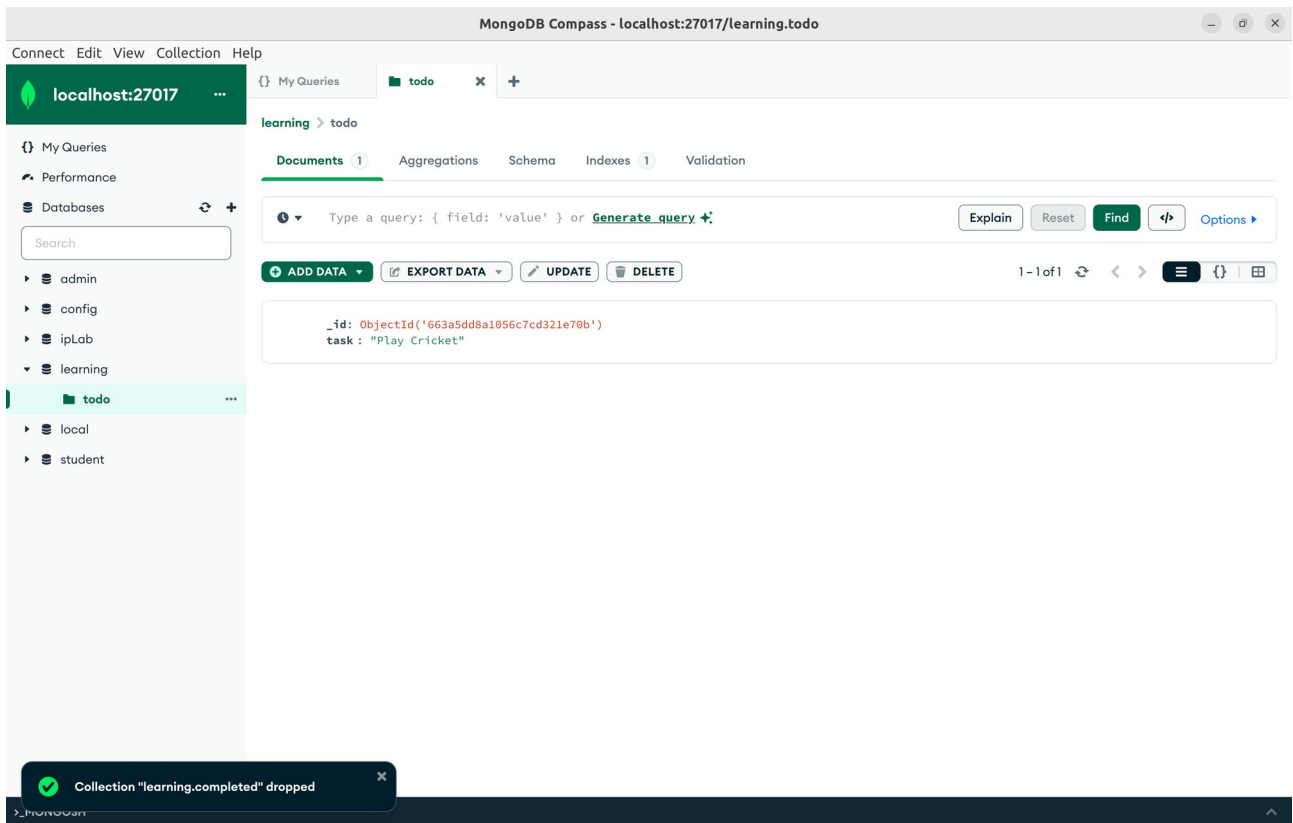
Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
^C
rohith@rohith:~/Desktop/IP Lab/Exercise-12 ToDo List Full Stack Web Application/
to-do-list$ npm start
```

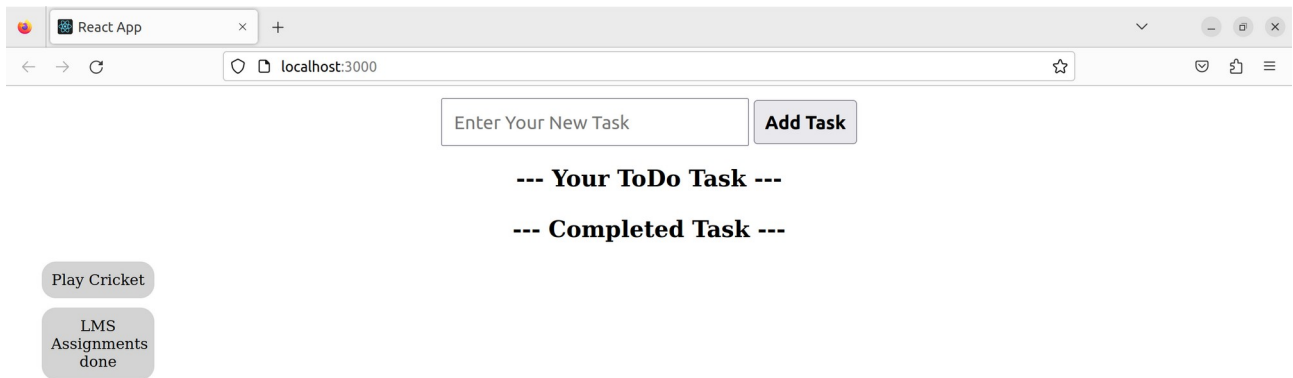


Previous data from the MongoDB Compass is retained. In order to modify it, you can delete the data from the MongoDB Compass and reload the webpage.





Now “LMS Assignments done” task is added using Add task button, then marked as completed.



Now refresh the MongoDB Compass Database to check the modified data.

