Server Side Implementation

Create a route file for managing tasks

Route “get” method to return tasks in res

router.get('/', function(req, res, next) {

  res.send(tasks);

});

Post method needs a task name to create a new task. Task name to be sent in the request body. [req.body.name]

router.post('/', function(req, res, next) {

  const name = req.body.name;

  if(name){

    tasks.push({id: tasks.length + 1, name: name, pending: true})

    res.send("Task Added");

  }

  else{

    res.send("Task Name Not Sent");

  }

});

“tasks” is an array object. “Push” method to insert new task to existing “tasks” array

Post method to toggle task status from pending to completed and vice versa. Route to /toggle/:id within tasks route. Id is retrieved using params inside request object

router.post('/toggle/:id', function(req, res, next) {

  const id = parseInt(req.params.id);

  const task = tasks.find(task => task.id === id);

  if(task){

    task.pending = !task.pending;

    res.send(`Updated status of task with id ${task.id}`);

  }

  else

    res.send(`Couldn't find task with ${task.id}`);

});

parseInt method to retrieve id from request obj. “find” method to retrieve particular task matching the task id.

Client Side

Service Class

A Service class to communicate with the server. [Uses whatwg-fetch]

getTasks to get tasks from server. Using “fetch” api of whatwg-fetch. Fetch api returns a promise

static getTasks(){

        function extractData(response){

            if(response.status === 200){

                return response.json();

            }

            throw new Error(`Unable to get data. status ${response.status}`);

        }

        return fetch('/tasks')

        .then(response => extractData(response));

        // .then(data => console.log(data));

    }

addTask is to add new task to existing tasks. Uses POST method using task name as input. taskname is passed in the body of the POST request.

static addTask(taskName){

    return fetch(`/tasks`, {

             method : 'POST',

             headers : { 'Content-Type' : 'application/json' },

             body : JSON.stringify({name : taskName})

                           }

);

     }

togglePending to toggle task status using POST method, passing task id as param.

static togglePending(taskId){

        return fetch(`/tasks/toggle/${taskId}`, {method : 'POST'});

    }

React App Component

Holds Add Task and Display Task Component as child.

Tasks are the state variable of App Component.

Fetches tasks from server on mounting event – componentDidMount function

Uses Service class to communicate with Server.

getTaskFromServer(){

        Service.getTasks()

        .then(tasks => this.setState({tasks}))

        .catch(err => this.setState({error: err.message}));

    }

componentDidMount(){

        this.getTaskFromServer();

    }

Handles child component events (Add Task – create task event & Display Task – toggle task event) using props.

AddTask components create task event is propagated back to App parent class to handle

App component code [addTask props]

<div><AddTask addTask = { taskName => this.addTask(taskName) }/></div>

AddTask component code [addTask props]

<button onClick = {() => this.props.addTask(this.\_taskName.current.value)}>Create</button>