```
// package.json

/ First of All we install json-server and concurrently package

{
  "name": "feedback-app",
  "version": "0.1.0",
  "proxy" : "http://localhost:5000", (/ We set the url as proxy so we don't use same
http:localhost:5000 over and over again)
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.2",
    "@testing-library/react": "^12.1.4",
    "@testing-library/user-event": "^13.5.0",
    "framer-motion": "^4.1.17",
    "json-server": "^0.17.0",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-icon": "^1.0.0",
    "react-icons": "^4.3.1",
    "react-router-dom": "^6.2.2",
    "react-scripts": "5.0.0",
    "uuid": "^8.3.2",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject",
    "server" : "json-server --watch db.json --port 5000", / Here we set server key word for run
json-server command
    "dev" : "concurrently \"npm run server\" \"npm start\""
      (/ Here we set to dev keyword to run server and run npm start.(It comes from concurrenty
package
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

FeedbackContext.js

```
import { createContext, useState, useEffect } from "react";

const FeedbackContext = createContext();

export const FeedbackProvider = ({ children }) => {
  const [isLoading, setIsLoading] = useState(true); // This state for our spinner component.Our spinner
will be shown until we change it.
  const [feedback, setFeedback] = useState([]);
  const [feedbackEdit, setFeedbackEdit] = useState({
    item: {},
    edit: false,
  });

  useEffect(() => {
    fetchFeedback();
  }, [])

  // Fetch feedback
  const fetchFeedback = async () => { // We fetch our feedback with json-server,  at below we didn't
write localhost cause we define it as proxy our package.json
    const response = await fetch(`/feedback?_sort=id&_order=desc`)
    const data = await response.json();
    // We set data variables to our response.json() and we use for setFeedback
    setFeedback(data);
    setIsLoading(false); // When our data been fetched, then our spinner will not appear
  }

  // Delete Feedback
  const deleteFeedback = async (id) => {
    if (window.confirm("Are you sure you want to delete?")) {
      await fetch(`/feedback/${id}`, // This ${id} for select specific id
                  { method: "DELETE" }
                 )

      setFeedback(feedback.filter((item) => item.id !== id));
    }
  };

  // Set item to be update

  const editFeedback = (item) => {
    setFeedbackEdit({
      item,
      edit: true,
    });
  };

  // Add Feedback
  const addFeedback = async (newFeedback) => {
    const response = await fetch("/feedback", { // Here we fetch for feedback
      method: "POST", // We set method as POST
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify(newFeedback)
    })

    const data = await response.json(); // We set data and send it to setFeedback instead newFeedback
    setFeedback([data, ...feedback]);
  };

  // Update Feedback Item
  const updateFeedback = async (id, updItem) => {
    const response = await fetch(`/feedback/${id}`, {   // This ${id} for select specific id
      method: "PUT",    // We set method as PUT
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(updItem)     // We use updItem because we sent it as param
    })
    const data = await response.json(); // Here set our response to data and above we use it for our
map function(Because we set to data, our response from fetch, ) - We did same thing to addFeedback
function
    setFeedback(
      feedback.map((item) => (item.id === id ? { ...item, ...data } : item))
    );

     setFeedbackEdit( // If we don't add this, when we updated any item, after that we can't add new
Feedback
```

```
      item: {},
      edit: false,
    })
  };

  return (
    <FeedbackContext.Provider
      value={{
        feedback,
        deleteFeedback,
        isLoading,
        addFeedback,
        editFeedback,
        feedbackEdit,
        updateFeedback,
      }}
    >
      {children}
    </FeedbackContext.Provider>
  );
};

export default FeedbackContext;
```

Spinner.jsx

```
import spinner from "../assets/spinner.gif"; / We import a gif as spinner and we use it below

function Spinner() {
  return (
    <img
      src={spinner}
      alt="Loading..."
      style={{ width: "100px", margin: "auto", display: "block" }}
    />
  );
}

export default Spinner;
```