****

**Name: Mariyam Lateef**

**ID: 20006105009**

**Department: Bs\_IT**

**Batch: 06**

**Subject: Web Technolgy**

**Assignment no: 03**

**React App**

1. **Command**

Here are steps to resolve this issue:

1. **Ensure npm directory exists**:
   * Manually check if the directory C:\Users\Mariyam\AppData\Roaming\npm exists. If not, create it.
2. **Reinstall Node.js**:
   * Uninstall Node.js completely from your system.
   * Download and install the latest version of Node.js from [the official Node.js website](https://nodejs.org/).
3. **Set npm prefix**:
   * Configure npm to use the correct global directory by running the following commands in your terminal:
4. **Reinstall npx**:

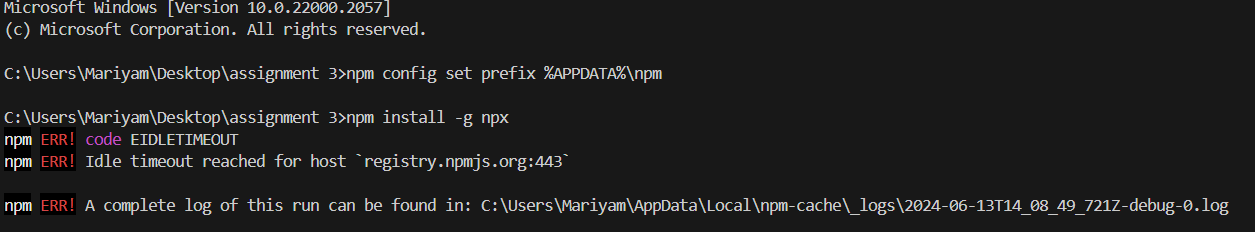
* Install npx globally again:

npm install -g npx

1. **Clean npm Cache**:

* Clear the npm cache to ensure there are no corrupted files:

npm cache clean --force

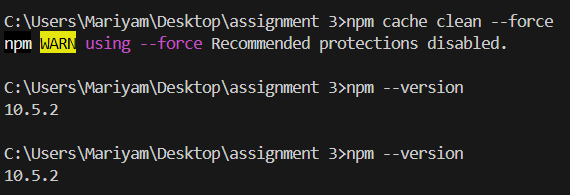
****

**2. Verify npm and npx Installation**:

* Check if npm and npx are installed correctly by running:

npm --version

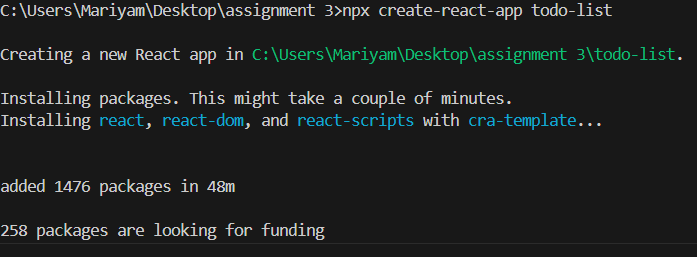
npx --version

****

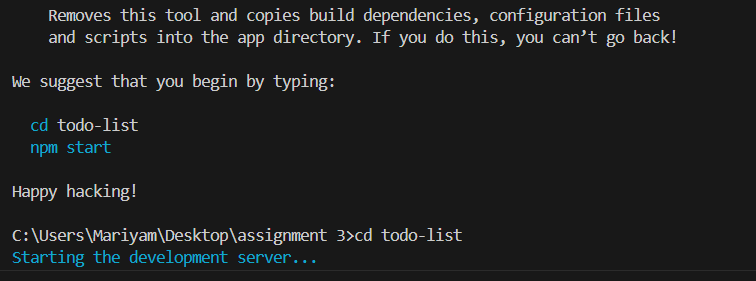
**3. Create React App**:

* Now, try creating the React app again:

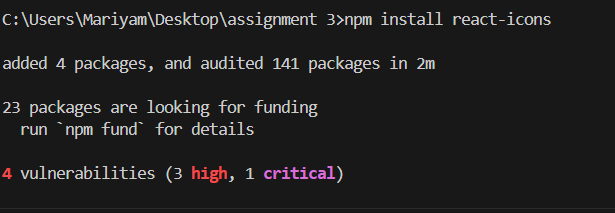
npx create-react-app todo-list

****

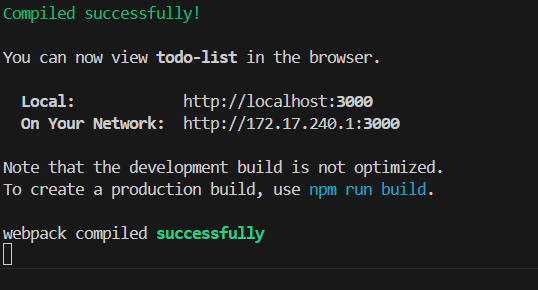
**4. cd todo-list:**

****

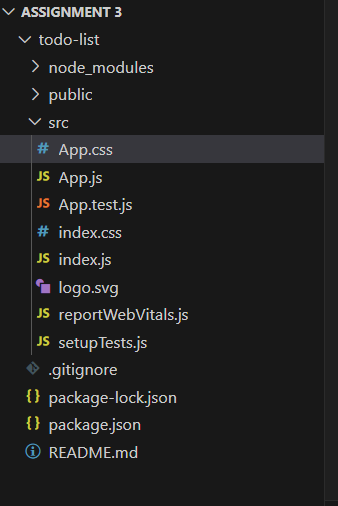
* 1. **npm install react-icon:**

****

* 1. **installed successfully**

****

* 1. **folder**

****

* 1. **App.js**

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

import './App.css';

import { AiOutlineDelete, AiOutlineEdit } from 'react-icons/ai';

import { BsCheckLg } from 'react-icons/bs';

function App() {

  const [isCompleteScreen, setIsCompleteScreen] = useState(false);

  const [allTodos, setTodos] = useState([]);

  const [newTitle, setNewTitle] = useState('');

  const [newDescription, setNewDescription] = useState('');

  const [newPriority, setNewPriority] = useState('medium');

  const [completedTodos, setCompletedTodos] = useState([]);

  const [currentEdit, setCurrentEdit] = useState('');

  const [currentEditedItem, setCurrentEditedItem] = useState('');

  useEffect(() => {

    const savedTodo = JSON.parse(localStorage.getItem('todolist'));

    const savedCompletedTodo = JSON.parse(localStorage.getItem('completedTodos'));

    if (savedTodo) {

      setTodos(savedTodo);

      console.log('Loaded todos from local storage:', savedTodo);

    } else {

      console.log('No todos found in local storage.');

    }

    if (savedCompletedTodo) {

      setCompletedTodos(savedCompletedTodo);

      console.log('Loaded completed todos from local storage:', savedCompletedTodo);

    } else {

      console.log('No completed todos found in local storage.');

    }

  }, []);

  useEffect(() => {

    localStorage.setItem('todolist', JSON.stringify(allTodos));

    localStorage.setItem('completedTodos', JSON.stringify(completedTodos));

    console.log('Saved todos to local storage:', allTodos);

    console.log('Saved completed todos to local storage:', completedTodos);

  }, [allTodos, completedTodos]);

  const handleAddTodo = () => {

    let newTodoItem = {

      title: newTitle,

      description: newDescription,

      priority: newPriority,

      completed: false,

    };

    let updatedTodoArr = [...allTodos];

    updatedTodoArr.push(newTodoItem);

    setTodos(updatedTodoArr);

    setNewTitle('');

    setNewDescription('');

    setNewPriority('medium');

  };

  const handleDeleteTodo = index => {

    let reducedTodo = [...allTodos];

    reducedTodo.splice(index, 1);

    setTodos(reducedTodo);

  };

  const handleComplete = index => {

    let now = new Date();

    let completedOn = now.toLocaleString();

    let filteredItem = {

      ...allTodos[index],

      completedOn: completedOn,

      completed: true,

    };

    let updatedCompletedArr = [...completedTodos];

    updatedCompletedArr.push(filteredItem);

    setCompletedTodos(updatedCompletedArr);

    handleDeleteTodo(index);

  };

  const handleDeleteCompletedTodo = index => {

    let reducedTodo = [...completedTodos];

    reducedTodo.splice(index, 1);

    setCompletedTodos(reducedTodo);

  };

  const handleEdit = (ind, item) => {

    setCurrentEdit(ind);

    setCurrentEditedItem(item);

  };

  const handleUpdateTitle = value => {

    setCurrentEditedItem(prev => {

      return { ...prev, title: value };

    });

  };

  const handleUpdateDescription = value => {

    setCurrentEditedItem(prev => {

      return { ...prev, description: value };

    });

  };

  const handleUpdatePriority = value => {

    setCurrentEditedItem(prev => {

      return { ...prev, priority: value };

    });

  };

  const handleUpdateToDo = () => {

    let newToDo = [...allTodos];

    newToDo[currentEdit] = currentEditedItem;

    setTodos(newToDo);

    setCurrentEdit('');

  };

  const handleClearCompleted = () => {

    setCompletedTodos([]);

  };

  return (

    <div className="App">

      <h1>My Todos</h1>

      <div className="todo-wrapper">

        <div className="todo-input">

          <div className="todo-input-item">

            <label>Title</label>

            <input

              type="text"

              value={newTitle}

              onChange={e => setNewTitle(e.target.value)}

              placeholder="What's the task title?"

            />

          </div>

          <div className="todo-input-item">

            <label>Description</label>

            <input

              type="text"

              value={newDescription}

              onChange={e => setNewDescription(e.target.value)}

              placeholder="What's the task description?"

            />

          </div>

          <div className="todo-input-item">

            <label>Priority</label>

            <select value={newPriority} onChange={e => setNewPriority(e.target.value)}>

              <option value="high">High</option>

              <option value="medium">Medium</option>

              <option value="low">Low</option>

            </select>

          </div>

          <div className="todo-input-item">

            <button

              type="button"

              onClick={handleAddTodo}

              className="primaryBtn"

            >

              Add

            </button>

          </div>

        </div>

        <div className="btn-area">

          <button

            className={`secondaryBtn ${isCompleteScreen === false ? 'active' : ''}`}

            onClick={() => setIsCompleteScreen(false)}

          >

            Todo

          </button>

          <button

            className={`secondaryBtn ${isCompleteScreen === true ? 'active' : ''}`}

            onClick={() => setIsCompleteScreen(true)}

          >

            Completed

          </button>

        </div>

        <div className="todo-list">

          {isCompleteScreen === false &&

            allTodos.map((item, index) => {

              if (currentEdit === index) {

                return (

                  <div className="edit\_\_wrapper" key={index}>

                    <input

                      placeholder="Updated Title"

                      onChange={e => handleUpdateTitle(e.target.value)}

                      value={currentEditedItem.title}

                    />

                    <textarea

                      placeholder="Updated Description"

                      rows={4}

                      onChange={e => handleUpdateDescription(e.target.value)}

                      value={currentEditedItem.description}

                    />

                    <select value={currentEditedItem.priority} onChange={e => handleUpdatePriority(e.target.value)}>

                      <option value="high">High</option>

                      <option value="medium">Medium</option>

                      <option value="low">Low</option>

                    </select>

                    <button

                      type="button"

                      onClick={handleUpdateToDo}

                      className="primaryBtn"

                    >

                      Update

                    </button>

                  </div>

                );

              } else {

                return (

                  <div className={`todo-list-item ${item.priority}`} key={index}>

                    <div>

                      <h3>{item.title}</h3>

                      <p>{item.description}</p>

                      <p className="priority-label">{item.priority}</p>

                    </div>

                    <div>

                      <AiOutlineDelete

                        className="icon"

                        onClick={() => handleDeleteTodo(index)}

                        title="Delete?"

                      />

                      <BsCheckLg

                        className="check-icon"

                        onClick={() => handleComplete(index)}

                        title="Complete?"

                      />

                      <AiOutlineEdit

                        className="check-icon"

                        onClick={() => handleEdit(index, item)}

                        title="Edit?"

                      />

                    </div>

                  </div>

                );

              }

            })}

          {isCompleteScreen === true &&

            completedTodos.map((item, index) => {

              return (

                <div className="todo-list-item" key={index}>

                  <div>

                    <h3>{item.title}</h3>

                    <p>{item.description}</p>

                    <p><small>Completed on: {item.completedOn}</small></p>

                  </div>

                  <div>

                    <AiOutlineDelete

                      className="icon"

                      onClick={() => handleDeleteCompletedTodo(index)}

                      title="Delete?"

                    />

                  </div>

                </div>

              );

            })}

          {isCompleteScreen && completedTodos.length > 0 && (

            <button className="clearBtn" onClick={handleClearCompleted}>

              Clear Completed

            </button>

          )}

        </div>

      </div>

    </div>

  );

}

export default App;

* 1. **App.css**

body {

  background-color: #f0f0f0;

  color: #333;

  overflow: hidden;

}

h1 {

  text-align: center;

  color: #333;

}

.todo-wrapper {

  background: #ffffff;

  padding: 2%;

  width: fit-content;

  margin-left: auto;

  margin-right: auto;

  margin-top: 3%;

  max-height: 80vh;

  overflow-y: auto;

  box-shadow: 0px 5px 7px rgba(0, 0, 0, 0.1);

  border-radius: 10px;

}

.todo-input {

  display: flex;

  align-items: center;

  justify-content: center;

  border-bottom: 1px solid #ddd;

  padding-bottom: 25px;

  margin-bottom: 25px;

}

.todo-input-item {

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  margin-right: 15px;

}

.todo-input-item input,

textarea {

  background: #f9f9f9;

  padding: 10px;

  border: 1px solid #ddd;

  border-radius: 7px;

  color: #333;

}

.todo-input-item select {

  padding: 10px;

  border: 1px solid #ddd;

  border-radius: 7px;

  background: #f9f9f9;

}

.primaryBtn {

  background: #007bff;

  padding: 10px;

  border-radius: 7px;

  border: none;

  color: white;

  font-weight: bold;

  cursor: pointer;

}

.secondaryBtn {

  background: #6c757d;

  padding: 10px;

  border-radius: 7px;

  border: none;

  color: white;

  font-weight: bold;

  cursor: pointer;

  margin-right: 15px;

}

.secondaryBtn.active {

  background: #007bff;

}

.todo-list {

  margin-top: 25px;

}

.todo-list-item {

  background: #f9f9f9;

  display: flex;

  align-items: center;

  justify-content: space-between;

  padding: 2%;

  margin-bottom: 10px;

  border-radius: 7px;

  border-left: 4px solid transparent;

  box-shadow: 0px 1px 3px rgba(0, 0, 0, 0.1);

}

.icon,

.check-icon {

  margin-left: 20px;

  cursor: pointer;

  color: #6c757d;

}

.icon:hover,

.check-icon:hover {

  color: #333;

}

.edit\_\_wrapper {

  display: flex;

  flex-direction: column;

  margin-bottom: 10px;

}

.edit\_\_wrapper input,

textarea,

select {

  margin-bottom: 10px;

  padding: 10px;

  border-radius: 7px;

  border: 1px solid #ddd;

  color: #333;

}

.edit\_\_wrapper .primaryBtn {

  margin-top: 10px;

  background: #007bff;

}

.priority-label {

  font-size: 0.75em;

  background: #007bff;

  padding: 2px 5px;

  border-radius: 5px;

  color: white;

  display: inline-block;

}

.high {

  border-left: 4px solid #dc3545;

}

.medium {

  border-left: 4px solid #ffc107;

}

.low {

  border-left: 4px solid #28a745;

}

.clearBtn {

  background: #dc3545;

  padding: 10px;

  border-radius: 7px;

  border: none;

  color: white;

  font-weight: bold;

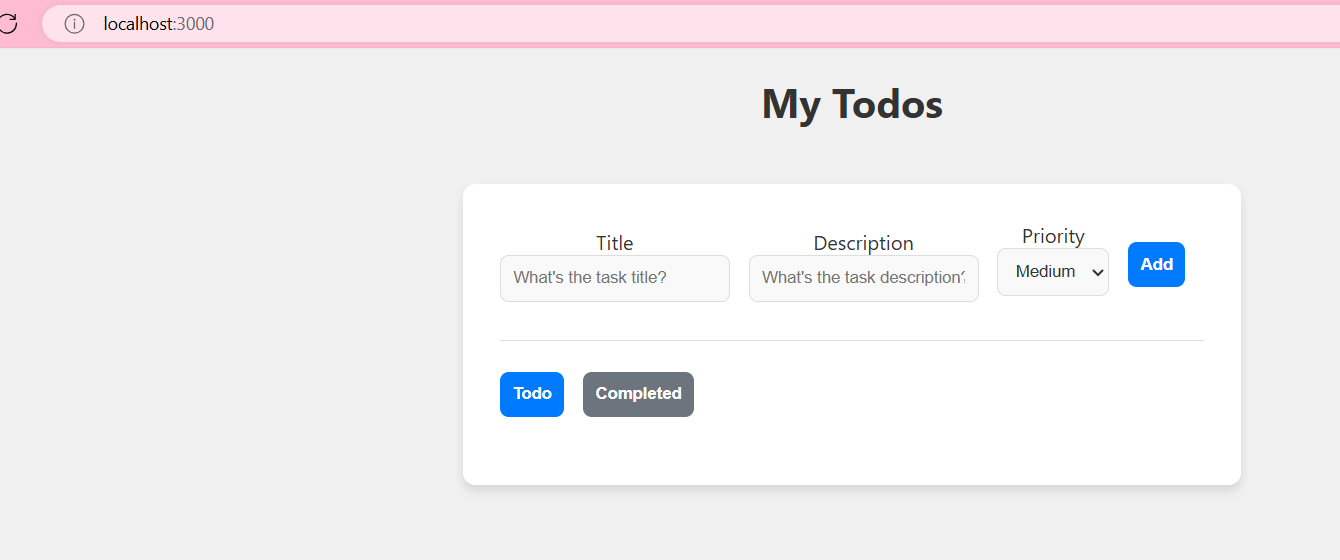
  cursor: pointer;

  display: block;

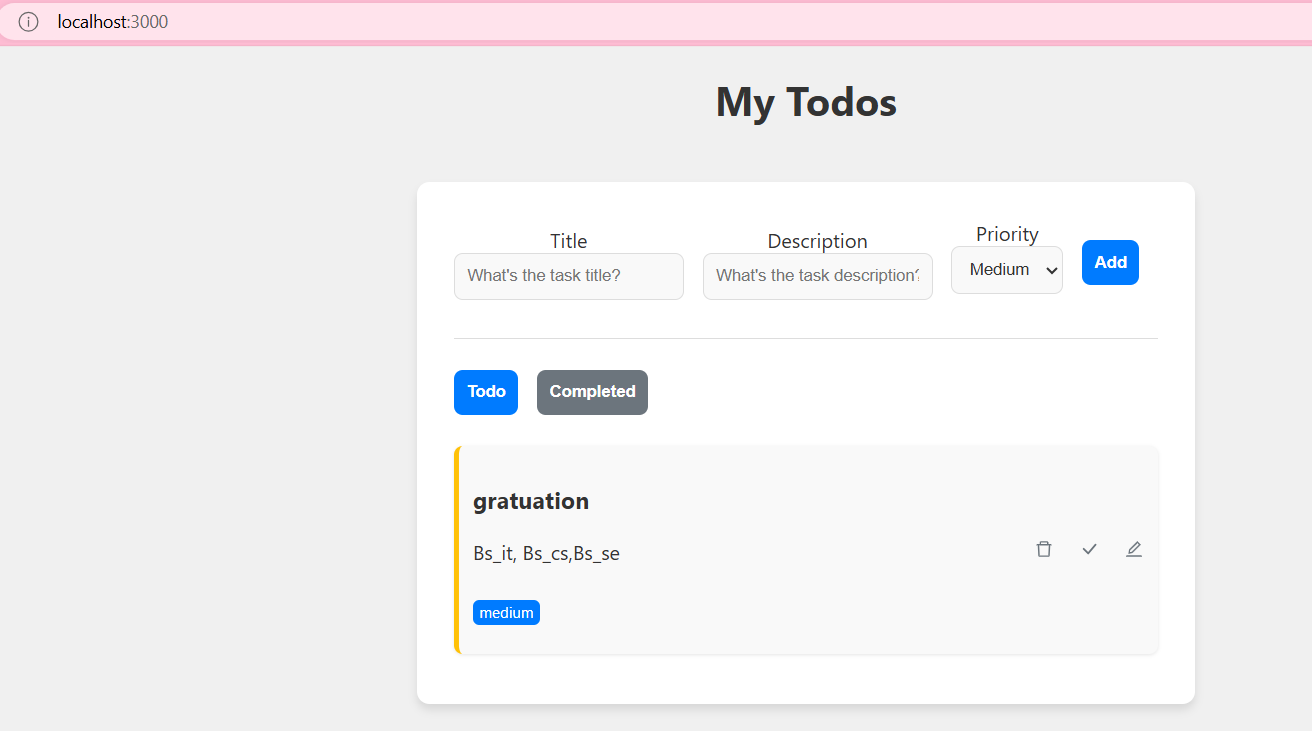
  margin: 20px auto;

}

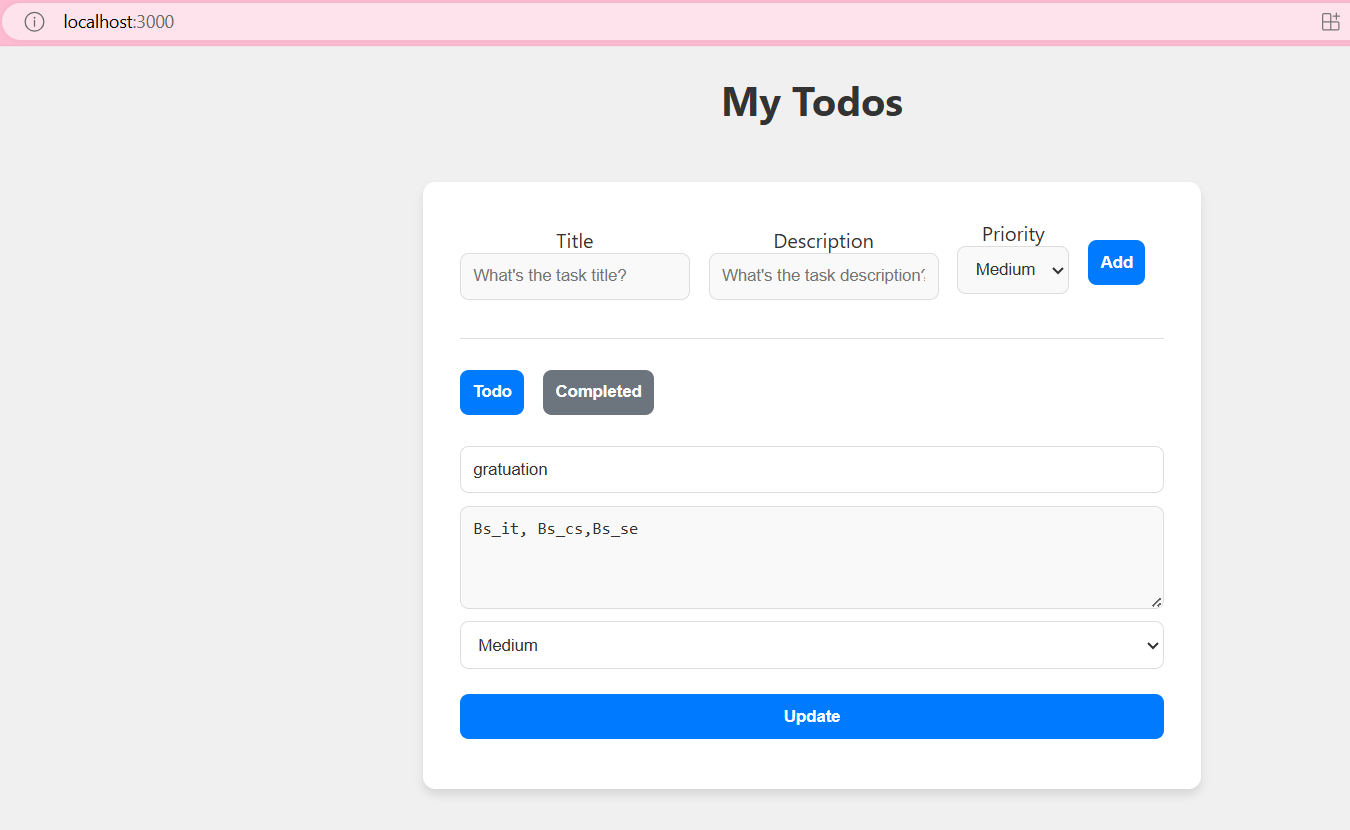
* 1. **Output:**

****

* 1. **Add todos**

****

* 1. **Update todos**

****

**link**

[React App](http://localhost:3000/)