Code App.css

```css
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
  font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande',
    'Lucida Sans', Arial, sans-serif;
}

body {
  background: linear-gradient(
    90deg,
    rgba(48, 16, 255, 1) 0%,
    rgba(100, 115, 255, 1) 100%
  );
}
.todo-app {
  display: flex;
  flex-direction: column;
  justify-content: start;
  width: 520px;
  min-height: 600px;
  background: #161a2b;
  text-align: center;
  margin: 128px auto;
  border-radius: 10px;
  padding-bottom: 32px;
}

h1 {
  margin: 32px 0;
  color: #fff;
  font-size: 24px;
}

.complete {
  text-decoration: line-through;
  opacity: 0.4;
}

.todo-form {
  margin-bottom: 32px;
}

.todo-input {
  padding: 14px 32px 14px 16px;
  border-radius: 4px 0 0 4px;
  border: 2px solid #5d0cff;
  outline: none;
  width: 320px;
  background: transparent;
  color: #fff;
}

.todo-input::placeholder {
  color: #e2e2e2;
}

.todo-button {
  padding: 16px;
  border: none;
```

```
  border-radius: 0 4px 4px 0;
  cursor: pointer;
  outline: none;
  background: linear-gradient(
    90deg,
    rgba(93, 12, 255, 1) 0%,
    rgba(155, 0, 250, 1) 100%
  );
  color: #fff;
  text-transform: capitalize;
}

.todo-input.edit {
  border: 2px solid #149fff;
}

.todo-button.edit {
  background: linear-gradient(
    90deg,
    rgba(20, 159, 255, 1) 0%,
    rgba(17, 122, 255, 1) 100%
  );
  padding: 16px 22px;
}

.todo-container {
  display: flex;
  flex-direction: row;
  position: relative;
}

.todo-row {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin: 4px auto;
  color: #fff;
  background: linear-gradient(
    90deg,
    rgba(255, 118, 20, 1) 0%,
    rgba(255, 84, 17, 1) 100%
  );

  padding: 16px;
  border-radius: 5px;
  width: 90%;
}

.todo-row:nth-child(4n + 1) {
  background: linear-gradient(
    90deg,
    rgba(93, 12, 255, 1) 0%,
    rgba(155, 0, 250, 1) 100%
  );
}

.todo-row:nth-child(4n + 2) {
  background: linear-gradient(
    90deg,
    rgba(255, 12, 241, 1) 0%,
    rgba(250, 0, 135, 1) 100%
  );
```

```
}

.todo-row:nth-child(4n + 3) {
  background: linear-gradient(
    90deg,
    rgba(20, 159, 255, 1) 0%,
    rgba(17, 122, 255, 1) 100%
  );
}

.icons {
  display: flex;
  align-items: center;
  font-size: 24px;
  cursor: pointer;
}

.delete-icon {
  margin-right: 5px;
  color: #fff;
}

.edit-icon {
  color: #fff;
                                                                    }
```

Code App.js

```
import React from 'react';
import './App.css';
import TodoList from './components/TodoList';

function App() {
  return (
    <div className='todo-app'>
      <TodoList />
    </div>
  );
}

                                                    export default App;
```

code index.js

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

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

Code Todo.js

```
import React, { useState } from 'react';
import TodoForm from './TodoForm';
import { RiCloseCircleLine } from 'react-icons/ri';
import { TiEdit } from 'react-icons/ti';

const Todo = ({ todos, completeTodo, removeTodo, updateTodo }) => {
  const [edit, setEdit] = useState({
    id: null,
    value: ''
  });

  const submitUpdate = value => {
    updateTodo(edit.id, value);
    setEdit({
      id: null,
      value: ''
    });
  };

  if (edit.id) {
    return <TodoForm edit={edit} onSubmit={submitUpdate} />;
  }

  return todos.map((todo, index) => (
    <div
      className={todo.isComplete ? 'todo-row complete' : 'todo-row'}
      key={index}
    >
      <div key={todo.id} onClick={() => completeTodo(todo.id)}>
        {todo.text}
      </div>
```

```
      <div className='icons'>
        <RiCloseCircleLine
          onClick={() => removeTodo(todo.id)}
          className='delete-icon'
        />
        <TiEdit
          onClick={() => setEdit({ id: todo.id, value: todo.text })}
          className='edit-icon'
        />
      </div>
    </div>
  ));
};

export default Todo;
```

code TodoForm.js

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

function TodoForm(props) {
  const [input, setInput] = useState(props.edit ? props.edit.value : '');

  const inputRef = useRef(null);

  useEffect(() => {
    inputRef.current.focus();
  });

  const handleChange = e => {
    setInput(e.target.value);
  };

  const handleSubmit = e => {
    e.preventDefault();

    props.onSubmit({
      id: Math.floor(Math.random() * 10000),
      text: input
    });
    setInput('');
  };

  return (
    <form onSubmit={handleSubmit} className='todo-form'>
      {props.edit ? (
        <>
          <input
            placeholder='Update your item'
            value={input}
            onChange={handleChange}
            name='text'
            ref={inputRef}
            className='todo-input edit'
          />
          <button onClick={handleSubmit} className='todo-button edit'>
            Update
          </button>
        </>
      ) : (
        <>
```

```
          <input
            placeholder='Add a todo'
            value={input}
            onChange={handleChange}
            name='text'
            className='todo-input'
            ref={inputRef}
          />
          <button onClick={handleSubmit} className='todo-button'>
            Add todo
          </button>
        </>
      )}
    </form>
  );
}

export default TodoForm;
```

code TodoList.js

```jsx
import React, { useState } from 'react';
import TodoForm from './TodoForm';
import Todo from './Todo';

function TodoList() {
  const [todos, setTodos] = useState([]);

  const addTodo = todo => {
    if (!todo.text || /^\s*$/.test(todo.text)) {
      return;
    }

    const newTodos = [todo, ...todos];

    setTodos(newTodos);
    console.log(...todos);
  };

  const updateTodo = (todoId, newValue) => {
    if (!newValue.text || /^\s*$/.test(newValue.text)) {
      return;
    }

    setTodos(prev => prev.map(item => (item.id === todoId ? newValue : item)));
  };

  const removeTodo = id => {
    const removedArr = [...todos].filter(todo => todo.id !== id);

    setTodos(removedArr);
  };

  const completeTodo = id => {
    let updatedTodos = todos.map(todo => {
      if (todo.id === id) {
        todo.isComplete = !todo.isComplete;
      }
      return todo;
    });
    setTodos(updatedTodos);
  };

  return (
    <>
      <h1>What's the Plan for Today?</h1>
      <TodoForm onSubmit={addTodo} />
      <Todo
        todos={todos}
        completeTodo={completeTodo}
        removeTodo={removeTodo}
        updateTodo={updateTodo}
      />
    </>
  );
}

export default TodoList;
```