Human Readable Algorithm

1. **Project Setup**

1.1. Initialize a new React project.
1.2. Install necessary dependencies: react-router-dom, styled-components or tailwindcss, etc.
1.3. Create a project structure with directories for components, pages, and utilities.

2. **Routing and Navigation**

2.1. Set up react-router-dom with routes for Registration, Login, Home, and Profile pages.
2.2. Implement guarded routes to restrict access to authenticated users.

3. **User Authentication**

3.1. **Registration Page:**

- Create a form to capture user details (username, password).
- Implement form validation.
- Save user details to localStorage.

3.2. **Login Page:**
- Create a form for username and password input.
- Validate user credentials against localStorage.
- Store authentication state in context or state management.

4. **Home Page (Task Management)**

4.1. **Task List Display:**

- Fetch tasks from localStorage.
- Display tasks in a list with color-coded priorities.

4.2. **Task CRUD Operations:**
  - ***Add Task:***
  - Create a form to input task details (name, description, priority, due date, status).
  - Save the task to localStorage.
  - ***Update Task:***
  - Allow users to edit task details.
  - Update the task in localStorage.
- ***Delete Task:***
  - Provide a delete button for each task.
  - Remove the task from localStorage.
- ***Mark Task as Completed:***
  - Add a checkbox/button to mark tasks as completed.

- Move completed tasks to a separate list in localStorage.

5. **Task Filtering and Search**

5.1. ***Filter Tasks***:
- Implement filtering options for due date, priority level, and status.

5.2. ***Search Tasks:***
- Add a search bar to filter tasks by name or description.
- Enable URL-based searching using query parameters.

6. ***Profile Page***

6.1. ***View User Profile:***
- Fetch and display user details from localStorage.

6.2. ***Update User Profile:***
- Provide a form to update user details.
- Save updated details to localStorage.

7. **User Interface and Experience**

7.1. ***Design Considerations:***
- Create a user-friendly and intuitive interface.
- Ensure consistent typography and color scheme.
- Implement hover effects and interactive elements.

7.2. ***Responsive Design:***
- Make the app responsive across various screen sizes (320px, 480px, 768px, 1024px, 1200px).

8. ***Notifications and Feedback***

8.1. Show notifications for background processes (e.g., task added, task deleted).
8.2. Provide feedback for user actions (e.g., successful login, form validation errors).

9. ***Data Persistence***

9.1. Use localStorage to persist task data, user details, and authentication state.
9.2. Implement functions to read from and write to localStorage.

10. Bonus Features

10.1. Add due date warning for overdue tasks.
10.2. Create a page to view completed tasks.
10.3. Include the date of task completion.
10.4. Enable URL-based search functionality.

Implementation Steps

1. **Setup and Initialization:**
   - Initialize React app.
   - Install react-router-dom, styled-components or tailwindcss.
   - Set up basic project structure.

2. **Routing:**
   - Configure routes in App.js.

- Create components for Registration, Login, Home, and Profile pages.

3. **Authentication:**
   - Implement registration and login forms.
   - Create authentication context or state.

4. **Task Management:**
   - Develop task list component.
   - Create forms for adding and updating tasks.
   - Implement CRUD operations.

5. **Filtering and Search:**
   - Add filtering options.
   - Implement search functionality.

6. **Profile Management:**
   - Create profile view and update forms.

7. **UI/UX Enhancements:**
   - Design user-friendly interface.
   - Ensure responsive design.

8. **Notifications and Feedback:**
   - Implement notification system.

9. **Data Persistence:**
   - Set up localStorage functions for data management.

Pseudocode

1. **Project Setup**

```
INITIALIZE new React project
INSTALL dependencies: react-router-dom, styled-components/tailwindcss
CREATE project structure with directories for components, pages, and utilities
```

2. **Routing and Navigation**

```
IMPORT BrowserRouter, Routes, Route from 'react-router-dom'
DEFINE routes in App.js:
  - Route for RegistrationPage
  - Route for LoginPage
  - Guarded Route for HomePage
  - Route for ProfilePage
```

3. **User Authentication**

```
// Registration Page
CREATE RegistrationForm component
  RENDER input fields for username, password
  ON submit:
    VALIDATE form data
    IF valid:
      SAVE user details to localStorage

// Login Page
CREATE LoginForm component
  RENDER input fields for username, password
  ON submit:
    VALIDATE credentials against localStorage
    IF valid:
      SET authentication state (e.g., in context or state management)
      REDIRECT to HomePage
```

4. **Home Page (Task Management)**

```
// Task List Display
CREATE TaskList component
  FETCH tasks from localStorage
  DISPLAY tasks with color-coded priorities

// Add Task
CREATE AddTaskForm component
  RENDER input fields for task name, description, priority, due date, status
  ON submit:
    VALIDATE form data
    IF valid:
      SAVE new task to localStorage

// Update Task
CREATE UpdateTaskForm component
  RENDER input fields pre-filled with existing task details
  ON submit:
    VALIDATE form data
    IF valid:
      UPDATE task in localStorage

// Delete Task
```

```
CREATE DeleteTaskButton component
  ON click:
    REMOVE task from localStorage

// Mark Task as Completed
CREATE MarkCompletedButton component
  ON click:
    MOVE task to completed list in localStorage
```

5. **Task Filtering and Search**

```
// Filter Tasks
CREATE FilterTasks component
  RENDER filtering options for due date, priority level, status
  APPLY filters to tasks list

// Search Tasks
CREATE SearchBar component
  RENDER search input field
  ON input change:
    FILTER tasks based on search query

// URL-based Search
UPDATE routing logic to handle search query in URL
  PARSE search query
  APPLY search filter to tasks list
```

6. Profile Page

plaintext

```
// View User Profile
CREATE ProfilePage component
  FETCH user details from localStorage
  DISPLAY user details

// Update User Profile
CREATE UpdateProfileForm component
  RENDER input fields pre-filled with existing user details
  ON submit:
    VALIDATE form data
    IF valid:
      UPDATE user details in localStorage
```

7. **User Interface and Experience**

```
// Design Considerations
CREATE reusable components for buttons, input fields, forms
```

```
APPLY consistent styles using styled-components/tailwindcss
IMPLEMENT hover effects for interactive elements

// Responsive Design
APPLY responsive design techniques using CSS media queries or tailwindcss
ENSURE components adjust layout for different screen sizes (320px, 480px, 768px,
1024px, 1200px)
```

8. Notifications and Feedback

```
 Show Notifications
CREATE Notification component
  ON successful operation:
    DISPLAY success message
  ON error:
    DISPLAY error message

Provide Feedback
UPDATE form components to show validation errors
ADD loading indicators for asynchronous operations
```

9. **Data Persistence**

```
// LocalStorage Management
CREATE utility functions for reading from and writing to localStorage
  FUNCTION saveToLocalStorage(key, data)
    CONVERT data to JSON
    SAVE JSON to localStorage with key

  FUNCTION loadFromLocalStorage(key)
    FETCH JSON from localStorage with key
    CONVERT JSON to object
    RETURN object
```