Ministry of SMEs and Startups

KOSME Korea SMEs and Startups Agency

LIKELION

JOBKOREA

LESSON 10
To-Do App Requirement Analysis

WEEK 02

Overview of To-Do App

❖ App Description

- ➢ A web app to create, manage, and track tasks with priorities and statuses.

❖ Target Users

- ➢ Individuals or teams needing task organization.

❖ Core Features

- ➢ Task CRUD, filtering, sorting, user assignment.

Task Data Structure

- **Task Model**
  - Defines the properties of a task in the app.
- **Key Fields**
  - id, title, description, start_date, due_date, status, priority, assignee_id.

Functional Requirements - Task Creation

- **Create Task**
  - Users can add a new task with required fields (title, start_date, etc.).
- **Validation**
  - Ensure title is not empty, start_date is valid.
- **Example**
  - Form with inputs for title, priority, and due_date.

Functional Requirements - Task Update

- **Update Task**
  - Edit task details like status, priority, or due_date.
- **Audit Fields**
  - Update updated_by and updated_time on changes.
- **Use Case**
  - Mark task as done and set completed_date.

Functional Requirements - Task Deletion

- **Delete Task**
  - Remove a task from the list, confirm before deletion.
- **Soft Delete Option**
  - Optionally mark as deleted instead of permanent removal.
- **Example**
  - Button with confirmation dialog.

Functional Requirements - Task Listing

- **List Tasks**
  - Display all tasks in a table or card view.
- **Columns / Fields**
  - Show title, status, priority, due_date.
- **Example**
  - Table with sortable columns.

Functional Requirements - Filtering

- **Filter Tasks**
  - Allow filtering by status, priority, or assignee_id.
- **Dynamic Updates**
  - Update list in real-time as filters are applied.
- **Example**
  - Dropdown to select status: done.

Functional Requirements - Sorting

- **Sort Tasks**
  - Sort by due_date, priority, or created_time.
- **Direction**
  - Support ascending and descending order.
- **Example**
  - Clickable column headers for sorting.

Functional Requirements - Task Assignment

- **Assign Task**
  - Link task to a user via assignee_id.
- **Display Assignee**
  - Show assignee name or ID in task list.
- **Use Case**
  - Team collaboration on tasks.

Non-Functional Requirements - Performance

- **Fast Rendering**
  - App should load task list in under 2 seconds.
- **Scalability**
  - Handle up to 1,000 tasks without lag.
- **Optimization**
  - Use memoization to reduce re-renders.

Non-Functional Requirements - Usability

- **Intuitive UI**
  - Clear layout, easy-to-use forms and buttons.
- **Accessibility**
  - Support screen readers, keyboard navigation.
- **Reference**
  - Web Accessibility

Non-Functional Requirements - Security

- **Data Validation**
  - Sanitize inputs to prevent injection attacks.
- **Authentication**
  - Restrict task editing to authorized users (created_by).
- **Example**
  - Check user ID before allowing updates.

User Interface - Task List

- **Design**
  - Table or card layout with columns for key fields.
- **Features**
  - Filter dropdowns, sortable headers, edit/delete buttons.
- **Example**
  - Mockup: Table with title, status, priority.

User Interface - Task Form

- **Form Fields**
  - Inputs for title, description, due_date, priority, etc.
- **Validation Feedback**
  - Show errors for invalid inputs.
- **Example**
  - Form with required field indicators.

Component Breakdown - App

❖ App Component

- ➢ Root component managing task state and routing.

❖ Responsibilities

- ➢ Fetch tasks, pass data to child components.

Component Breakdown - TaskList

❖ TaskList Component

- ➢ Displays tasks in a table or card format.

❖ Features

- ➢ Filtering, sorting, edit/delete actions.

Component Breakdown - TaskForm

❖ TaskForm Component

- ➢ Handles task creation and editing.

❖ Features

- ➢ Input validation, submit handling.

Using React Hook Form

❖ **Why React Hook Form?**

Simplifies form handling with minimal re-renders.

State Management

- **Options**
  - Local state (useState), Context API.
- **Recommendation**
  - Use useState for small apps, Context for larger ones.

Data Persistence

- **Local Storage**
  - Store tasks in browser for persistence.
- **Backend API**
  - Save tasks to a server for multi-user access.

API Integration

- **REST API**
  - CRUD endpoints for tasks (GET /tasks, POST /tasks, etc.).
- **REST API Auth**
  - Login

Authentication

- **User Roles**
  - Restrict task editing to created_by or assignee_id.
- **Implementation**
  - Use JWT or session-based auth.

Performance Optimization

- **Memoization**
  - Use useMemo for filtered/sorted tasks.
- **Lazy Loading**
  - Load tasks incrementally for large lists.

Error Handling

- **Validation Errors**
  - Display errors for invalid form inputs.
- **API Errors**
  - Handle network failures gracefully.

Advanced Features

- **Task Reminders**
  - Notify users when due_date is near.
- **Bulk Actions**
  - Delete or update multiple tasks at once.
- **Example**
  - Checkbox to select tasks for bulk delete.

Project Milestones

- **Phase 1**
  - Build UI and local state management.
- **Phase 2**
  - Add API integration and authentication.
- **Phase 3**
  - Implement advanced features and testing.

Conclusion

❖ Key Takeaways

- ➢ Requirement analysis ensures a clear roadmap for the To-Do App.

❖ Next Steps

- ➢ Start coding the app, begin with CreateTaskForm and TaskList.

❖ Resources

- ➢ React-Router, React Hook Form, React Testing Library