Project Presentation

https://github.com/DreyWesson/taskflow

Emmanuel

User Story

Basically, I have a medium blog where I share technical stuffs I find fascinating, so the web app was implemented to give my audience a deep understanding of what happens under the hood in most of the popular packages - Nodemon, Postman, Express.JS and ReactJS framework.

Goal: To make use of vanilla fullstack application that mimics popular packages and framework.

https://medium.com/@oduwoledare
https://www.npmjs.com/~dreywesson

Development Cycle

1. Research & Planning
   - Deep dive into source code of target frameworks/packages
   - Identified core functionality to replicate
   - Planned architecture to support objectives

2. Prototyping
   - Created basic implementation of Express.JS core features
   - Built simple server with middleware pattern
   - Established routing foundation

3. Iterative Development
   - Implemented packages one at a time
   - Started with backend (Express, Nodemon)
   - Followed with frontend (React custom implementation)
   - Added UI components and task management features

4. Testing & Refinement
   - Implemented E2E testing with Cypress
   - Set up Lighthouse scoring to measure performance
   - Refined accessibility features based on testing

5. Documentation & Education
   - Added comments explaining internal workings
   - Developed diagrams to visualize framework internals

Features Overview

Utilizes custom implementations

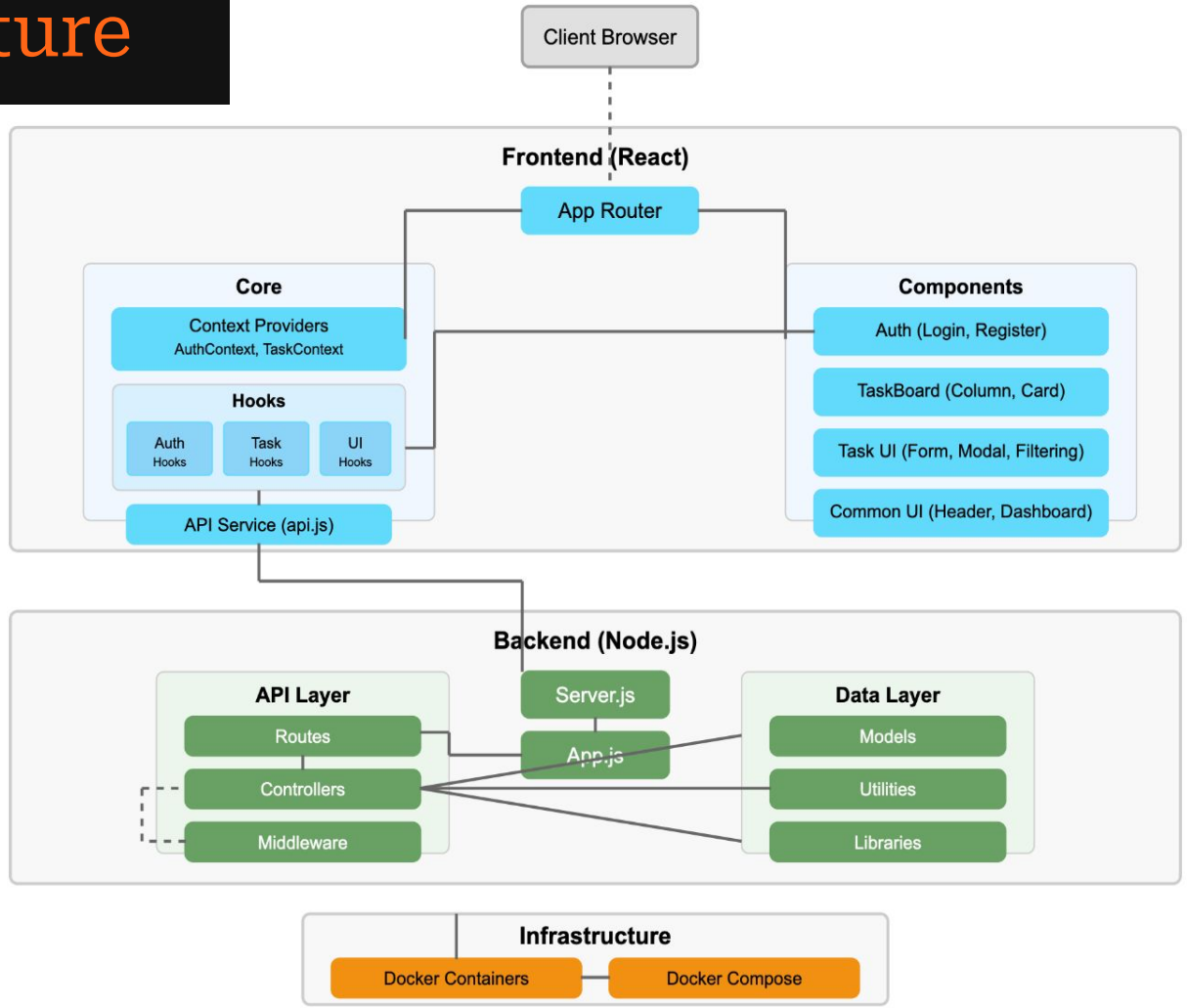
1. Custom Express.JS
2. Custom Postman
3. Custom Nodemon
4. Custom React.JS (not fully ready)

Application Overview

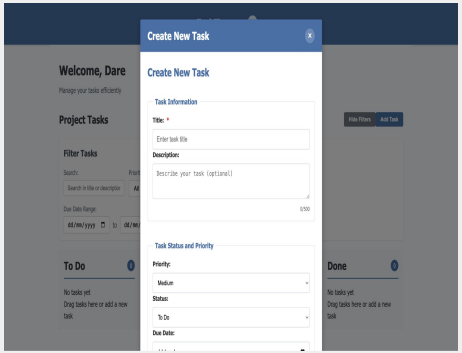
TaskFlow is a Kanban-style task management application built with React and Express

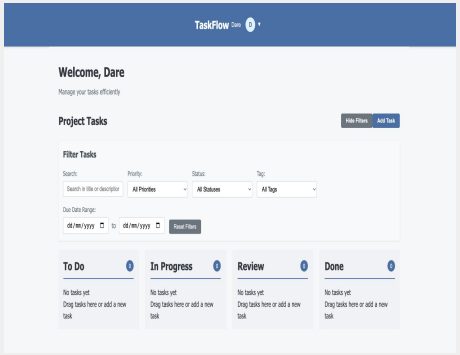
1. Task CRUD
2. Drag and Drop
3. Advanced Filtering
4. Search Functionality
5. Responsive Design
6. Accessibility Features
7. Robust E2E Testing
8. Containerization
9. Documentation

Architecture

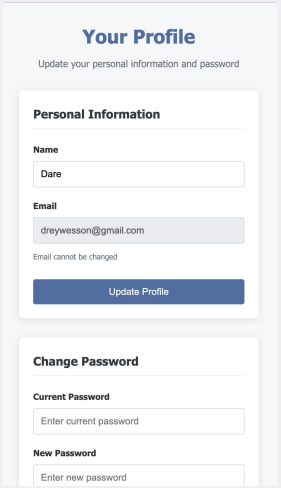


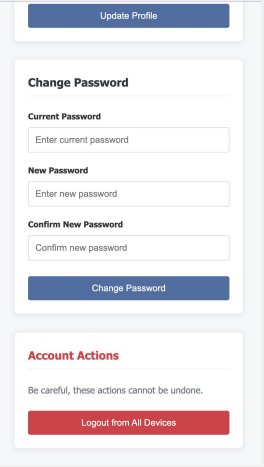
Wireframes

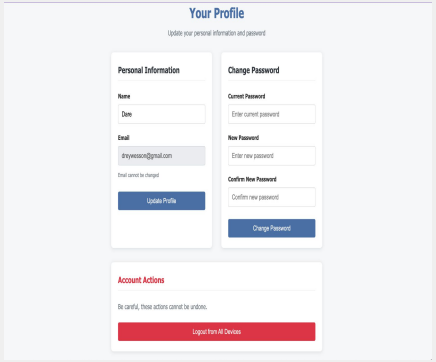


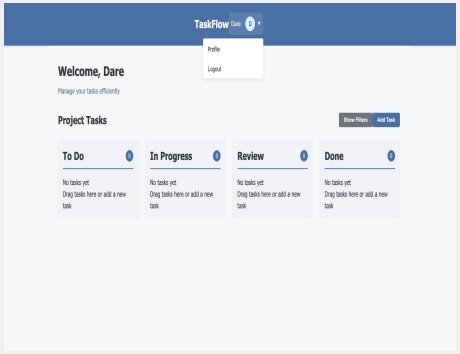


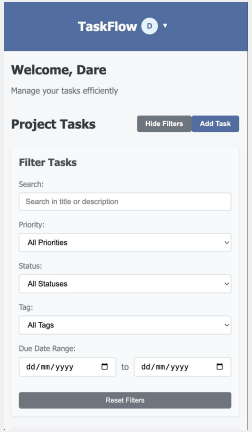
MOBILE

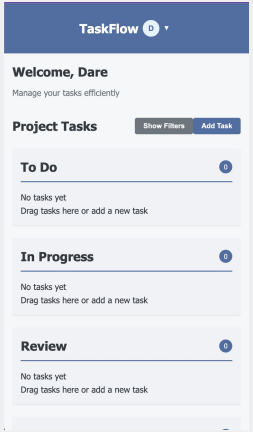


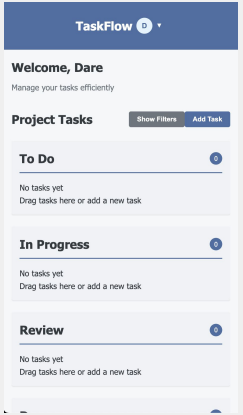












Functional features

Task Management System:

- Organizes tasks into columns with different statuses (Kanban-style layout)
- Allows tasks to be moved between different status columns

Accessibility and Keyboard Navigation:

- Task selection with Enter key
- Movement between columns using left/right arrow keys
- Navigation between tasks in a column using up/down arrow keys
- Confirmation of task status changes with Enter key
- Cancellation of selection with Escape key

Task Status Transitions:

- Tasks can be moved from one status to another (e.g., from "To Do" to "In Progress")
- System dispatches custom events to track task movements

User Feedback System:

- Provides feedback when tasks are selected, moved, or when selections are canceled
- Visual indicators for selected tasks

Non-Functional features

Accessibility Compliance:

- Screen reader compatibility with ARIA attributes
- Keyboard-only navigation support
- Focus management for keyboard users
- Visual indicators for keyboard focus states
- Screen reader announcements for important actions

User Experience Enhancements:

- Visual feedback for selected items (box-shadow effect)
- Focus indicators only appear for keyboard users, not mouse users
- Consistent navigation patterns

Performance Optimization:

- Lazy loading of components to improve initial load times
- Only loading necessary resources when they're needed
- Deferred loading of non-critical UI elements
- Efficient DOM manipulation and event handling

Maintainability:

- Modular code structure with separate functions for different concerns
- Clear function naming that describes purpose
- Cleanup routines to prevent memory leaks
- Separation of concerns (accessibility, keyboard navigation, styling)

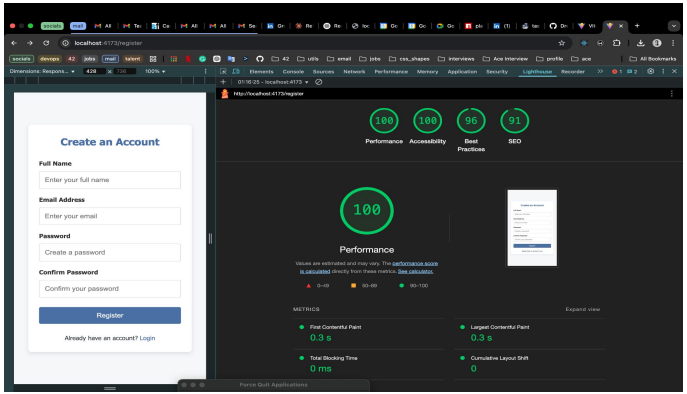
Quality Assurance:

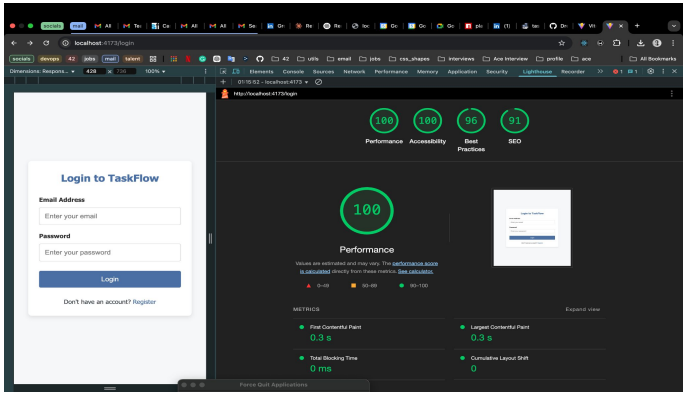
- End-to-end testing to verify complete user workflows
- Test coverage for critical user journeys
- Automated testing to catch regressions
- Validation of accessibility features in real-world scenarios

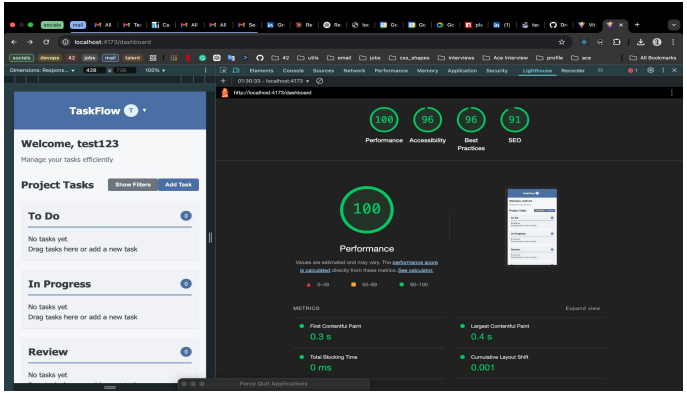
Cross-cutting Concerns:

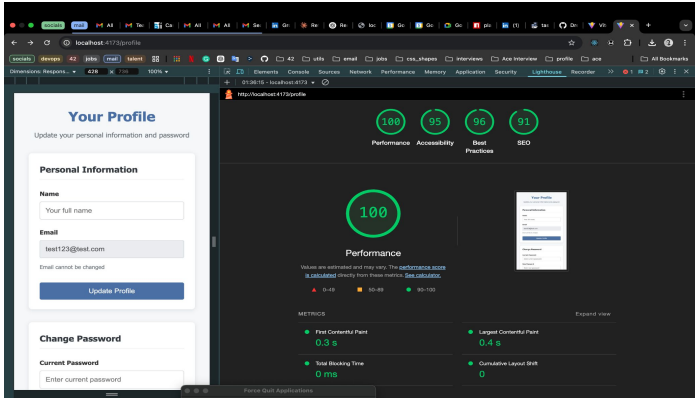
- Screen reader announcements system
- Focus management system
- Keyboard navigation framework
- Visual styling for accessibility

Lighthouse


Create an Account
Full Name
Enter your full name
Email Address
Enter your email
Password
Create a password
Confirm Password
Confirm your password
Register
Already have an account? Login
100
96
91
Performance
Accessibility
Best Practices
SEO
100
Performance
First Contentful Paint
0.3 s
Largest Contentful Paint
0.3 s
Total Blocking Time
0 ms
Cumulative Layout Shift
0


Login to TaskFlow
Email Address
Enter your email
Password
Enter your password
Login
Don't have an account? Register
100
100
96
91
Performance
Accessibility
Best Practices
SEO
100
Performance
First Contentful Paint
0.3 s
Largest Contentful Paint
0.3 s
Total Blocking Time
0 ms
Cumulative Layout Shift
0


TaskFlow
Welcome, test123
Manage your tasks efficiently
Project Tasks
Show Filters
Add Task
To Do
No tasks yet
Drag tasks here or add a new task
In Progress
No tasks yet
Drag tasks here or add a new task
Review
No tasks yet
100
96
96
91
Performance
Accessibility
Best Practices
SEO
100
Performance
First Contentful Paint
0.3 s
Largest Contentful Paint
0.4 s
Total Blocking Time
0 ms
Cumulative Layout Shift
0.001


Your Profile
Update your personal information and password
Personal Information
Name
Your full name
Email
test123@test.com
Email cannot be changed
Update Profile
Change Password
Current Password
Enter current password
100
95
96
91
Performance
Accessibility
Best Practices
SEO
100
Performance
First Contentful Paint
0.3 s
Largest Contentful Paint
0.4 s
Total Blocking Time
0 ms
Cumulative Layout Shift
0

Chrome is being controlled by automated test software.

Search specs

cypress / e2e

- Authentication.spec.js
- Profile-Management....
- Task-Creation.spec.js
- Task-Deletion.spec.js
- Task-DragDrop.spec.js
- Task-Filtering.spec.js
- Task-Lifecycle.js
- Task-Movement.spec...
- Task-Update.spec.js

Specs

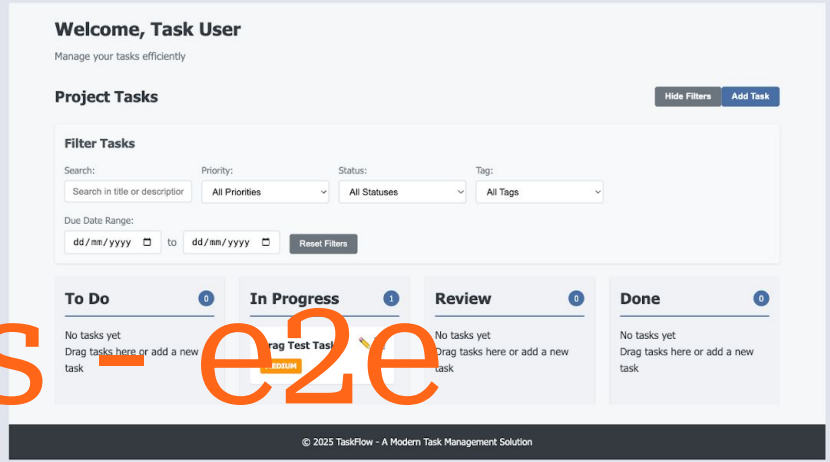
✓ 12 × -- ↻ --

Task-Lifecycle.js 01:02

Task Lifecycle

- ✓ should create a new task
- ✓ should edit the task details
- ✓ should filter the task by priority
- ✓ should move the task to In Progress using the edit form
- ✓ should filter the task by status
- ✓ should filter the task by search term
- ✓ should move the task to Review using the edit form
- ✓ should move the task to Done using the edit form
- ✓ should set due date and filter by
- ✓ should delete the task
- ✓ should create and move a task with drag and drop
- ✓ should create, filter, and clean up multiple tasks

http://localhost:5173/dashboard 1280x720 50% Chrome 133



Cannot Show Snapshot While Tests Are Running

Cypress - e2e

Force Quit Applications

The Role

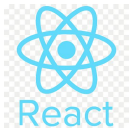
Responsibilities

- Develop and support web Client and web Services for Aftersales (Assist AR) solution
- Design and develop new functionality and components for our software using coding languages but not limited to JavaScript, React, Typescript, CSS3
- Optimize and enhance our software with new, efficient algorithms and modern software technologies
- Work collaboratively in a Scrum development team within a Continuous Integration environment
- Monitor the latest technical developments on your field and make proposals for improvements

Requirements

- University or college degree in Computer Science, Business Informatics, Software Engineering, or any related subject
- More than 3 years of professional work experience in web development
- Proven background in working with JavaScript, TypeScript, HTML and CSS
- Solid knowledge of JavaScript frameworks and tools like React.js, Redux, Webpack, etc.
- Knowledge of C#, asp.net or C++ is a plus
- Experience with testing frameworks such as cypress is a plus
- Understanding of Clean Code and Design Patterns for object-orientated programming
- Fluency in English is mandatory, German is a plus

Stack Used




The project demonstrate many of the key technologies listed in the job description. Beyond the stack used on this project. I have experience using C/C++, Typescript. I am proficient with Object-Oriented Programming. Scrum

Improvements

1. **Security**: Replace localStorage token storage with httpOnly cookies to prevent XSS vulnerabilities. This trade-off was made to simplify the custom ExpressJS implementation.
2. **API Mocking**: Complete API mocking coverage for remaining endpoints to improve testing reliability and development workflow.
3. **State Management**: Current Context+useReducer implementation works well, but Redux could be considered for complex state requirements, middleware support, and dev tools.
4. **Code Splitting**: Implement React.lazy() with Suspense for component-level code splitting to reduce initial bundle size and improve load times.
5. **TypeScript Integration**: Add TypeScript for static type checking to catch errors earlier in development and improve code documentation and maintainability.
6. **Performance**: Implement request batching for task updates, especially during drag-and-drop operations, to reduce API call frequency and improve responsiveness.
7. **Offline Capabilities**: Implement service workers for offline functionality, enabling a progressive web app (PWA) approach with data caching and synchronization when connection is restored.

I possess a strong capacity to both learn new technologies and unlearn outdated approaches as needed. My adaptability remains one of my core professional strengths.

Emmanuel

Thanks for your time