TaskFlow247 – Project Documentation

1. How to Run the Application Locally

1. Clone the repository:

```bash
git clone https://github.com/pavansingh888/todo-nextjs
```

2. Navigate into the project:

```bash
cd todo-nextjs
```

3. Install dependencies:

```bash
npm install
```

4. Run in development mode:

```bash
npm run dev
```

5. Open the app at:

```
http://localhost:3000
```

2. Approach

TaskFlow247 was built with a modern full-stack approach using Next.js App Router, React Query for data synchronization, Zustand for global state (authentication, UI preferences), and DummyJSON as a mock backend. The goal was to create a production-grade architecture even with a fake API source.

3. Assumptions Made Using DummyJSON

- DummyJSON does not persist created tasks — so the app stores tasks in local state after successful API calls.
- User preferences (theme, idle timeout, stay-signed-in choice) are stored in localStorage via Zustand.
- Logout is implemented on frontend because DummyJSON API has no logout endpoint.

4. Challenges & Solutions

- **Challenge:** DummyJSON overwrites updated fields (e.g., editing todo content resets completion status).
  Solution: Partial PATCH request logic + merging server/optimistic data manually.
- **Challenge:** Auto-logout triggering incorrectly inside modal due to hover/scroll events.
  Solution: Added event filtering and allowed modal to intercept events properly.
- **Challenge:** Keeping landing, login, dashboard, and profile under a shared header layout with dynamic nav controls.
  Solution: Created a reusable global Header component with page-aware link rendering.
- **Challenge:** Protected routes showing UI briefly before redirect.
  Solution: Reworked ProtectedClient to delay rendering until session validation is complete.

5. Technical Documentation

- **Next.js App Router** – modern routing and layouts.
- **React Query** – caching, deduplication, retry logic, optimistic updates.
- **Zustand** – lightweight global state without boilerplate.
- **Sonner** – clean toast notifications.
- **TailwindCSS** – fast, utility-driven UI development.
- **Framer Motion** – smooth animations.
- **DummyJSON** – mock API for auth + todos.

Project includes:

- Authentication flow with HttpOnly cookies
- Automatic inactivity logout with user-configurable timeout
- Dark/Light theme with persistence
- Full CRUD for tasks with optimistic UI

6. Features

- Secure login using DummyJSON mock auth
- Dashboard with statistics, filters, animations
- Auto-logout after inactivity (user-configurable)
- Stay signed-in option
- Edit, delete, update tasks (with optimistic UI)

- Local-only tasks supported when API cannot persist
- Fully responsive UI
- Theming system (dark/light)

7. Tech Stack & Why

- **Next.js 14+** → Full-stack React framework with server & client components
- **React Query** → Industry-standard data fetching layer
- **Zustand** → Minimal global state solution (better than Redux for this scale)
- **TailwindCSS** → Fast styling and dark-mode support
- **Sonner** → Lightweight and modern toast UI
- **Framer Motion** → Natural animations

8. Links

- **GitHub Repository:** https://github.com/pavansingh888/todo-nextjs
- **Live Deployment:** https://taskflow247.vercel.app/