Thank you, Shawn. With this updated structure, I've regenerated your architectural breakdown and onboarding summary to reflect the rollback accurately and ensure the handoff to another AI or dev is seamless.

Here’s your updated **ToDoApp project profile**, tailored for reusability and shared development across mobile and web:

**🧠 App Overview**

* **Project**: ToDoApp (modular prototype for ManageApp)
* **Platform Targeting**: React Native mobile-first with future React Web integration
* **Design Goals**:
  + Reusable business logic across platforms
  + Modular code structure by feature
  + Offline-ready with local persistence (future: sync and multi-user)
  + Developer-friendly onboarding with clear documentation

**🏗️ Updated Architecture**

**🗂️ Folder Layout (Essentials)**

/

├── App.tsx

├── package.json

├── tsconfig.json

├── index.js

├── database/

│ └── TaskDB.ts # Local storage (future: pluggable DB backend)

├── shared/

│ ├── components/ # Shared UI components

│ ├── constants/taskConstants.ts

│ ├── types/Task.ts

│ └── utils/dateUtils.ts

├── modules/

│ └── tasks/

│ ├── AddTask.tsx

│ ├── TaskListScreen.tsx

│ ├── routes.ts # Screen-routing (expandable to web routing)

│ ├── components/TaskCard.tsx

│ ├── hooks/useTasks.ts

│ ├── services/TaskService.ts

│ └── types/Task.ts # Likely should consolidate with shared/types

├── 00\_docs/

│ ├── ToDoApp.md

│ ├── ToDoApp\_Migration.md

│ ├── Documentation Structure.ipynb

│ └── init\_docs/, templates/

└── \_\_tests\_\_/App.test.tsx

**🧩 Functional Summary**

| **Feature** | **Status** | **Location** |
| --- | --- | --- |
| Add new task | ✅ Done | AddTask.tsx, useTasks.ts |
| Search by title | ✅ Done | TaskListScreen.tsx |
| Filter by date | ✅ Done | filterDate + DateTimePicker |
| Toggle completion | ✅ Done | TaskCard.tsx, useTasks.ts |
| Archive completed | ✅ Done | Header archive button |
| Delete with modal | ✅ Done | Custom modal confirmation |

**🛠️ Design Patterns & Abstractions**

* **Modular Feature Folder**: modules/tasks is a self-contained feature module with screen logic, services, and UI.
* **Shared Types & Utils**: colocated in /shared for consistent access across all feature modules and platforms.
* **Database Layer**: TaskDB.ts handles low-level persistence; can be swapped for web backend or offline storage.
* **Service Abstraction**: TaskService.ts is a logic gate between UI and data layer—future-proof for web API, sync logic, or auth control.
* **Documentation**: Markdown and Jupyter notebooks in 00\_docs provide live architectural thinking and guidance.

**🔌 Extensibility Path**

| **Future Feature** | **Recommended Action** |
| --- | --- |
| 🌐 React Web Integration | Use [react-native-web](https://necolas.github.io/react-native-web/) + shared components for maximum reuse |
| 🔐 Admin Privileges for Deletion | Add global context or user role service to conditionally expose destructive actions |
| 🔁 Undo/Recycle Bin | Keep deleted tasks in memory or archive to a separate storage namespace |
| 🧩 Modal Service Extraction | Move modal to shared/components/ConfirmModal.tsx and wrap with a promise-based API |
| ☁️ API Sync / Cloud Storage | Replace TaskDB.ts with a cloud adapter (Supabase, Firebase, etc.) behind TaskService.ts |
| 👥 Multi-User/Workspace Support | Extend Task type with userId/workspaceId, scope queries accordingly |

**📋 Onboarding for Other AI/Devs**

Any AI or engineer picking this up should:

1. Read 00\_docs/ToDoApp.md and project-structure.txt for architectural intent.
2. Use TaskService.ts as the entrypoint for modifying persistence and logic.
3. Maintain shared types and date utils in /shared/ to ensure platform parity.
4. Respect feature folder boundaries—each modules/{feature} is scoped and autonomous.
5. Contribute updates to Jupyter docs and templates for repeatable documentation practices.

Would you like this formatted as a markdown doc to drop into 00\_docs/ToDoApp\_Structure.md? I can prep that file immediately, fully polished for version control and future contributors.