React Native Hometask — Payslips (iOS & Android)

Build a small **React Native** application (TypeScript) for managing and viewing payslips. Use **mock data** only. Focus on native file handling, clean architecture, accessibility, and a polished mobile UX.

Functional Requirements

Payslip Model

Each payslip contains:

- **id:** unique identifier
- **fromDate:** start date of the covered period (ISO format suggested)
- **toDate:** end date of the covered period (ISO format suggested)
- **file:** a bundled asset representing the payslip (PDF or image). You may reuse the same file across items.

Screens

1. **Payslip List**

- Display a scrollable list of payslips.
- Each list item shows the period: “fromDate – toDate”.
- Tapping an item navigates to the details screen.

2. **Payslip Details**

- Display **id**, **fromDate**, **toDate**, and an indicator of the file type (PDF/image).
- Provide a **Download Payslip** action:
    - Saves the associated file to device storage and confirms success/failure to the user (e.g., show an alert).
- (Optional) Provide an **Open/Preview** action to view the file using a native viewer.

Data & State

- Initialize the app with several payslips in **in-memory state (React Context, Redux etc**)
- No API calls or server is required.

Non-Functional Requirements

Platform Scope

- **React Native (CLI) or Expo**, **TypeScript**, platforms **iOS** and **Android**.

Native File Handling (Download)

Implement the “download” by copying the bundled file to a user-accessible or app-specific directory and informing the user of the saved location.

Acceptable approaches:

- Use a well-supported React Native library for filesystem operations and handle Android runtime permissions where applicable.
- (Optional) Opening/previewing the saved file is a plus.

Suggested Project Structure (Informational)

- Separate concerns (screens, navigation, components, state/store, utilities).
- Keep components small and composable.
- Include basic date formatting utilities and a simple theme.

Evaluation Criteria

Reviewers will assess:

- **Architecture & Code Quality:** clear separation of concerns, idiomatic React Native and TypeScript usage, maintainable structure.
- **Type Safety:** appropriate types and interfaces across the app.
- **Native Handling:** correct file save logic, permission handling, actionable user feedback on success/failure.
- **Stability:** no obvious crashes or red screens; handles error cases gracefully.
- **Documentation:** a concise README with accurate setup and run instructions.

Submission Checklist

Repository

- Public Git repository link.
- Clean commit history (helpful but not mandatory for the task timebox).
- Include a README file with instructions on how to run the project locally.