React Native Major Assignment: Personal Task Manager with Categories

Objective:

Build a **Personal Task Manager** mobile application that allows users to create, categorize, and manage tasks effectively. The app should provide a seamless user experience with **navigation, state management, API integration, and custom UI components**.

Requirements:

Frontend (React Native)

1. Navigation

- Implement **Stack + Tab navigation**.
- Stack Navigation: Authentication screens (Login, Register).
- Tab Navigation: **Home, Categories, Completed Tasks, Profile**.

2. State Management

- Use **RTK Query**.
- Manage task states (create, update, delete tasks).
- Fetch tasks from API and sync with local storage.

3. API Handling

- Use **RTK Query** for API requests.
- CRUD operations for tasks:
    - Create Task (title, description, due date, priority, category).
    - Update Task.
    - Delete Task.
    - Fetch Task list.

4. Custom Hooks

- Create custom hooks for **API calls, form handling, and local storage**.

5. Form Handling

- Use **Formik + Yup** for form validation.
- Implement task creation/editing form.

6. Local Storage

- Store tasks locally using **AsyncStorage**.
- Sync local tasks with API when online.

7. Custom Components

- Design reusable components:
    - Task Card (Displays title, due date, priority).
    - Category Card (Shows number of tasks per category).
    - Input Fields & Buttons.

8. UI/UX

- Use **React Native Paper or Shadcn/UI** for a polished UI.

Backend (Based on Webknot's Tech Stack)

Use the backend tech stack as per Webknot’s sessions taught so far for API development.

1. Backend Framework

- Use **Nest.js** OR **Spring-boot** as per Webknot’s standard.

2. Database

- Use **MongoDB** (via Mongoose if using Node.js) OR **PostgreSQL**

3. API Endpoints

- **POST /tasks** → Create a task.
- **GET /tasks** → Get all tasks.
- **GET /tasks/:id** → Get task details.
- **PUT /tasks/:id** → Update a task.
- **DELETE /tasks/:id** → Delete a task.

4. Middleware

- **Express Middleware / Flask Middleware** for authentication.
- Implement CORS, error handling, and logging.

5. Deployment

- Deploy backend using **Render / Vercel / Railway**.
- Store frontend app data via **Firebase / Supabase (optional)**.

Deliverables

1. **Frontend Repository**: React Native app with all features.
2. **Backend Repository**: REST API with proper documentation.
3. **Demo Video**: A short walkthrough of app functionality.