ROPULVA | TECHNICAL ASSESSMENT | MOHAMED ABDELHAMED

Introduction 🎯

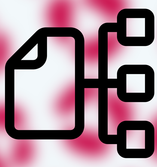
- Briefly describe the purpose of the application (Task Management App as per the assessment) 📝
- Mention the core technologies used (Flutter 💙, Firebase 🔥, Bloc State Management 🧠)
- High-level overview of the application's functionality (add ➕, edit ✏️, delete 🗑️ tasks, offline sync 🔄, platform builds 📱💻)

Project Structure Overview 🗂️

- Explain the project structure. 🗺️
- app Directory 📱: Describe what the app directory contains.
  - controllers Directory 🕹️: Contains all logic related to state management using the BLoC pattern.
    - task_bloc.dart ⚙️: The main BLoC file that handles business logic for tasks.
    - task_event.dart ⚡: Defines all possible events the task BLoC can handle (add, update, delete, etc.).
    - task_state.dart 🚦: Defines all possible states of the task feature in your app
  - data Directory 🗄️: Contains all classes to handle data sources and data models.
    - data_sources Directory 🗃️: Source of the data (API, local db..).
      - remote_data_source.dart ☁️: Class which handles remote data from firebase firestore.
    - models Directory 🧩: Contains data models.
      - task_model.dart 📝: Model for a Task object.
    - repos Directory 🏪: Contains repository classes for accessing the data sources.
      - task_repository.dart 📜: Interface for task repository operations
      - task_repository_impl.dart ⚒️: Concrete Implementation of TaskRepository which uses RemoteDataSource
  - presentation Directory 📺: Contains UI components.
    - screens Directory 📱: Holds entire screens.
      - task_list_screen.dart 📋: The main task list screen.
    - widgets Directory 🧩: Contains reusable UI elements.
      - add_buttom_windes.dart 🪟: A custom window for adding task.
      - add_task_button.dart ➕: Floating button to open task adding window.
      - congratulations_dialog.dart 🎉: Dialog to show when a task is completed.
      - empty_task_list.dart 📭: UI showed when there is no tasks.
      - filter_chips.dart 🏷️: Chips to control tasks displayed (All/Completed/Not completed)
      - header_section.dart 🗣️: UI for the header with greeting and user image.
      - show_task_bottom_sheet.dart 📄: Displays task details and options in a bottom sheet.
      - show_task_dialog.dart 💬: Dialog to show task details and options.
      - task_form_validation.dart ✅: Class to handle task form validation
      - task_list.dart 📋: Listview to display all available tasks.
      - task_list_content.dart 🗄️: UI of content of the task_list.
      - task_list_item2.dart 🏷️: UI single item of the task_list.
- core Directory ⚙️: Contains the shared codebase.
  - constants Directory 📌: Holds app wide constants values.
    - constants.dart 🔢: Values to use in app code.
  - enums.dart 🔣: Holds project wide enums.
  - error Directory ❌: Contains classes related to failures in the project.
    - failure.dart ⚠️: Base Failure class and its implementations.
  - screens Directory 🖼️: Holds base screens.
    - splash_screen.dart 💫: The very first splash screen for app
  - services Directory 🧰: Contains classes for specific utility services that is used around the app
    - cache_helper.dart 🗄️: Service that helps to save and retrieve data to local storage.
    - calendar_service.dart 📆: Service to check if user grant permission for calendar access.
    - connectivity_helper.dart 🌐: Service to handle connectivity issues.
    - console_logger.dart 📝: Simple logger to view logs in console.
    - responsive_helper.dart 📐: Service to handle responsive view depending on device size
    - service_locator.dart 📍: Setup dependency injections.
    - snackbar_service.dart 📢: Service to display snackbar.
    - sound_service.dart 🔊: Service to play sounds.
    - transition_helper.dart 🎬: Service to handle transitions.
    - version_helper.dart ℹ️: Service to check app version.
    - window_helper.dart 🖥️: Helper class to manage window size.
  - themes Directory 🎨: Contains color themes and style guide
    - colors.dart 🌈: Class to manage colors in the app.
    - themes.dart 🎭: Class to manage themes of the app.
  - utils Directory 🛠️: Contains utility classes and functions.
    - date_formatter.dart 📅: Class to format date.
  - widgets Directory 🧩: Contains reusable widgets for the project.
    - connectivity_dialog.dart 🔌: Custom dialog for connection issues.
    - connectivity_status_icon.dart 📶: UI that displays connection status.
    - connectivity_wrapper.dart 🔗: Component to control displaying of the app based on connection.
    - custom_window_frame.dart 🖼️: Custom component to customize the frame for desktop app.
    - exit_confirmation_dialog.dart 🚪: Dialog to display when user tries to exit the desktop app.
- firebase_options.dart 🔥: Firebase configurations.
- list_files.dart 📑: List of generated files (not important).
- main.dart 🚦: The main entry point for the application.

III. Core Features Implementation ✨

- Bloc State Management 🧠:
  - Explain the flow of events, states, and the BLoC itself. 💠
  - Mention the use of the task_event.dart ⚡ and task_state.dart 🚦 files.
  - Explain how task_bloc.dart ⚙️ handles different scenarios (add, edit, delete).
- Firebase Firestore Integration 🔥:
  - Explain how data is fetched and pushed using Firestore through remote_data_source.dart ☁️.
  - Mention that Firebase is used as a backend 🗄️
  - Explain the usage of task_repository.dart 📜 to abstract data access.
- Offline Support 📴:
  - Describe how you handled offline data storage (cache_helper.dart 💾).
  - Explain that all operation works in offline mode until connection is detected to save the state.
  - Detail the synchronization logic with Firebase. 🔄
- Task Management Logic 🧠:
  - Explain the details of adding ➕, editing ✏️, and deleting 🗑️ tasks, including how data is persisted and retrieved.
  - How is validation handled ✅



UI IMPLEMENTATION DOCUMENTATION 📱

TASKLISTSCREEN 📃

- IMPLEMENTS THE MAIN TASK MANAGEMENT INTERFACE.
- USES BLOCBUILDER TO REBUILD THE UI BASED ON STATE CHANGES. 🔄
- CONTAINS THREE MAIN SECTIONS:
  - FILTER CHIPS AT THE TOP. 🏷️
  - SCROLLABLE TASK LIST IN THE MIDDLE. 📋
  - ADD TASK BUTTON AT THE BOTTOM. ➕
- MANAGES A RESPONSIVE LAYOUT FOR DIFFERENT SCREEN SIZES. 📐

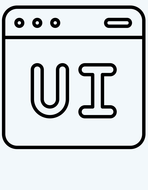
STATE MANAGEMENT & UI UPDATES 🔄

BLOC PATTERN IMPLEMENTATION 🧠

- TASKBLOC MANAGES STATE CHANGES AND BUSINESS LOGIC. ⚙️
- THE UI AUTOMATICALLY REBUILDS WHEN THE STATE CHANGES THROUGH BLOCBUILDER. ⚡
- STATE UPDATES FOLLOW A UNIDIRECTIONAL DATA FLOW: ➡️
  - USER ACTION TRIGGERS AN EVENT. 👆
  - THE BLOC PROCESSES THE EVENT. ⚙️
  - A NEW STATE IS EMITTED. 🚦
  - THE UI REBUILDS WITH NEW DATA. 📱

USER INTERACTION 👆

- USERS CAN:
  - ADD NEW TASKS VIA THE FLOATING ACTION BUTTON. ➕
  - FILTER TASKS USING CHIPS. 🏷️
  - MARK TASKS AS COMPLETE/INCOMPLETE. ✅
  - VIEW TASK DETAILS. 👁️
  - DELETE TASKS WITH SWIPE ACTIONS. 🗑️



COMMON COMPONENTS ❇️

ADDTASKBUTTON ➕

- A REUSABLE FAB COMPONENT.
- TRIGGERS A TASK CREATION DIALOG. 💬
- MAINTAINS CONSISTENT STYLING ACROSS SCREENS. 🎨

CONGRATULATIONSDIALOG 🎉

- SHOWN WHEN A MILESTONE IS ACHIEVED.
- FEATURES ANIMATED CELEBRATION EFFECTS. ✨
- HAS A CUSTOMIZABLE MESSAGE. 📝

FILTERCHIPS 🏷️

- PROVIDES HORIZONTAL SCROLLABLE FILTER OPTIONS.
- TOGGLES FILTERING BY CATEGORY/STATUS.
- OFFERS VISUAL FEEDBACK FOR ACTIVE FILTERS. 👁️

TASKLIST 📋

- A CORE LIST/GRID VIEW COMPONENT.
- HANDLES EMPTY STATES. 📭
- SUPPORTS DIFFERENT LAYOUTS BASED ON SCREEN SIZE. 📐

RESPONSIVE DESIGN 📐

SCREEN SIZE HANDLING 📱💻

- USES BREAKPOINT-BASED LAYOUTS (MOBILE < 600PX).
- MOBILE: SINGLE COLUMN LIST VIEW. 📱
- DESKTOP: GRID LAYOUT WITH MULTIPLE COLUMNS. 💻
- IMPLEMENTS DYNAMIC PADDING AND MARGINS. ↔️

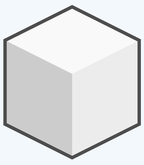
RESPONSIVEHELPER CLASS 🛠️

- CENTRALIZED RESPONSIVE UTILITIES. 🧰
- PROVIDES SCREEN SIZE BREAKPOINTS. 📏
- HELPER METHODS FOR:
  - DETERMINING THE DEVICE TYPE. 📱💻
  - CALCULATING ADAPTIVE DIMENSIONS. 📐
  - ADJUSTING LAYOUTS DYNAMICALLY. ↔️
  - MANAGING COMPONENT SIZES. 📏
  - SETTING APPROPRIATE SPACING. ↔️

THIS IMPLEMENTATION ENSURES A CONSISTENT UX ACROSS DEVICES WHILE MAINTAINING A CLEAN, MAINTAINABLE CODE STRUCTURE. ✅

CORE FEATURES IMPLEMENTATION DOCUMENTATION ✨

BLOC STATE MANAGEMENT 🧠



EVENT FLOW 🔄
- USER ACTION TRIGGERS AN EVENT. 👆
- EVENT IS PROCESSED BY THE BLOC. ⚙️
- THE BLOC EMITS A NEW STATE. 🚦
- THE UI REBUILDS BASED ON THE STATE. 📱

COMPONENTS 🧩
- TASKEVENT: BASE CLASS FOR ALL EVENTS. ⚡
  - ADDTASKEVENT ➕
  - EDITTASKEVENT ✏️
  - DELETETASKEVENT 🗑️
  - LOADTASKSEVENT 🚂
  - FILTERTASKSEVENT 🏷️
- TASKSTATE: REPRESENTS UI STATES. 🚦
  - TASKINITIAL ⚪
  - TASKLOADING ⏳
  - TASKLOADED ✅
  - TASKERROR ❌
- TASKBLOC: CORE BUSINESS LOGIC USING HYBRID APPROACH 🎛️
  - HANDLES EVENTS. ⚡
  - MANAGES STATE TRANSITIONS. 🔄
  - COORDINATES WITH REPOSITORY TO MANAGE LOCAL DATABASE AND FIRESTORE. 🗄️
  - USES LOCAL DATA BASE IF OFFLINE OTHERWISE CALLS REMOTE DATABASE.

FIREBASE INTEGRATION 🔥

REMOTE DATA SOURCE 📡
- HANDLES CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS WITH FIRESTORE. ☁️
- MANAGES REAL-TIME DATA SYNCING. 🔄
- HANDLES ERROR CASES AND TIMEOUTS. ⏳

REPOSITORY PATTERN 🍫
- ABSTRACTS DATA ACCESS. 🍊
- COORDINATES BETWEEN REMOTE AND LOCAL STORAGE. 🗄️
- PROVIDES A CLEAN API FOR THE BLOC. 📜

OFFLINE SUPPORT 📴

CACHE HELPER 💾
- LOCAL STORAGE USING HIVE/SQLITE. 🗄️
- STORES TASK DATA AND USER PREFERENCES. ⚙️
- MANAGES DATA VERSIONING. 🔄

SYNC STRATEGY ♻️
- SAVE OPERATIONS LOCALLY FIRST. 💾
- QUEUE CHANGES WHEN OFFLINE. 📴
- SYNC WITH FIREBASE WHEN ONLINE. 📶
- RESOLVE CONFLICTS USING TIMESTAMPS. ⏱️

TASK MANAGEMENT 🍊

CRUD OPERATIONS ⚙️
- CREATE: VALIDATION -> LOCAL SAVE -> REMOTE SYNC. ➕
- READ: CACHE FIRST -> REMOTE FETCH. 🚂
- UPDATE: OPTIMISTIC UI -> BACKGROUND SYNC. ✏️
- DELETE: SOFT DELETE -> REMOTE SYNC. 🗑️

DATA VALIDATION ✅
- REQUIRED FIELDS CHECKING. 📝
- DATE RANGE VALIDATION. 📅
- DUPLICATE DETECTION. ⚠️
- INPUT SANITIZATION. 🛡️

ERROR HANDLING ❌
- NETWORK ERRORS. 📶
- VALIDATION ERRORS. 📝
- SYNC CONFLICTS. ⚠️
- TIMEOUT HANDLING. ⏳

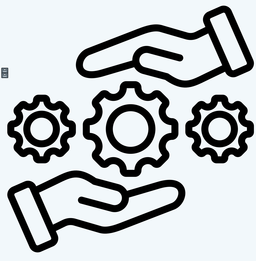
THIS ARCHITECTURE ENSURES:
- RESPONSIVE UI. 📱
- OFFLINE FUNCTIONALITY. 📴
- DATA CONSISTENCY. 🍊
- ERROR RESILIENCE. 🛡️
- CLEAN SEPARATION OF CONCERNS. ⚙️

SERVICES AND CORE COMPONENTS DOCUMENTATION 🛠️

STEP-BY-STEP IMPLEMENTATION PLAN: 📝
- DEFINE SERVICE LAYER ARCHITECTURE. ⚙️
- IMPLEMENT CORE HELPERS. 🛠️
- SETUP SERVICE LOCATOR. 📍
- CREATE BASE SCREENS. 🌄
- DESIGN THEME SYSTEM. 🎨
- BUILD CUSTOM WIDGETS. 🍀

SERVICE LAYER DOCUMENTATION 🧰

CACHEHELPER 💾
- MANAGES LOCAL DATA PERSISTENCE USING HIVE. 🗄️
- HANDLES:
  - TASK CACHING. 🍊
  - USER PREFERENCES. ⚙️
  - APP STATE PERSISTENCE. 🔄
  - DATA VERSIONING. 🔢



CONNECTIVITYHELPER 📶
- MONITORS NETWORK STATUS. 📡
- PROVIDES:
  - CONNECTION STATE STREAMS. 🌊
  - AUTO-RETRY MECHANISMS. ♻️
  - BACKGROUND SYNC TRIGGERS. ⏱️
  - OFFLINE MODE DETECTION. 📴

SERVICE LOCATOR (GETIT) 📍
- CENTRALIZE DEPENDENCY INJECTION FOR SERVICES 💉

CORE COMPONENTS ⚙️

BASE SCREENS 🌄

SPLASHSCREEN 💫
- INITIAL LOADING SCREEN. ⏳
- APP INITIALIZATION. 🚀
- SERVICE STARTUP. ⚙️
- ROUTE PREPARATION. 🗺️

THEME SYSTEM 🎨

THEMEDATA
- LIGHT/DARK MODE SUPPORT. ☀️🌙
- CUSTOM COLOR SCHEMES. 🌈
- TYPOGRAPHY DEFINITIONS. 🔤
- COMPONENT STYLING. 🖌️

CUSTOM WIDGETS 🍀

CONNECTIVITYDIALOG ⚠️
- SHOWS NETWORK STATUS. 📶
- RETRY OPTIONS. ♻️
- OFFLINE MODE INDICATOR. 📴
- USER GUIDANCE. ℹ️

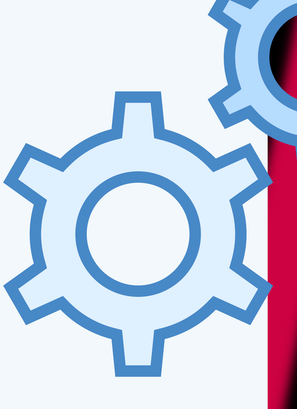
CONNECTIVITYWRAPPER 🔗
- NETWORK STATE MONITORING. 📡
- AUTO-RECONNECTION. 🔄
- WIDGET TREE PROTECTION. 🛡️
- ERROR BOUNDARY. ❌

CUSTOMWINDOWFRAME 🖼️
- DESKTOP WINDOW CONTROLS. 🖥️
- CUSTOM TITLEBAR. 🏷️
- DRAG REGIONS. 🖱️
- MINIMIZE/MAXIMIZE/CLOSE. ⬇️⬆️❌

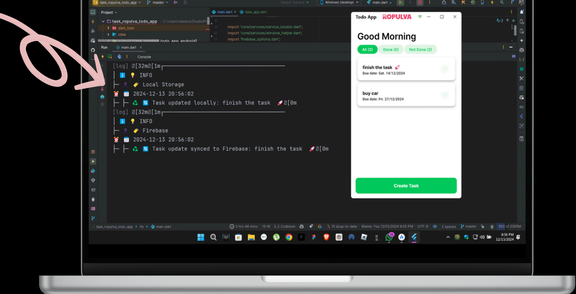
EXITCONFIRMATIONDIALOG 🚪
- UNSAVED CHANGES CHECK. 📝
- CLEAN APP SHUTDOWN. 🚪
- STATE PRESERVATION. 🔄
- USER CONFIRMATION. ✅

BENEFITS: ✅
- MODULAR ARCHITECTURE. 🍀
- REUSABLE COMPONENTS. ♻️
- CONSISTENT UI/UX. ✨
- ROBUST ERROR HANDLING. 🛡️
- CLEAN SERVICE SEPARATION. ⚙️

THIS STRUCTURE ENSURES A MAINTAINABLE, TESTABLE, AND SCALABLE APPLICATION ARCHITECTURE. 🚀



DEBUGER

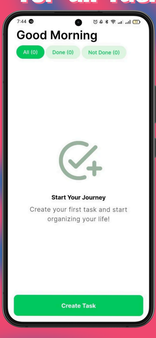


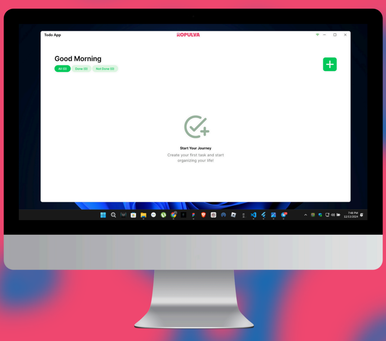
UI screen shots

splash screen

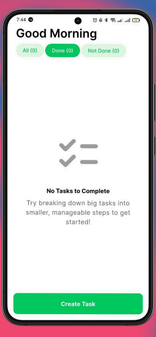
Home screen empty tasks

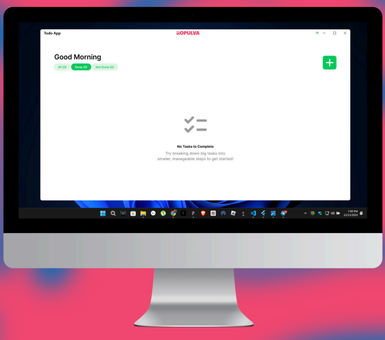
for all tasks



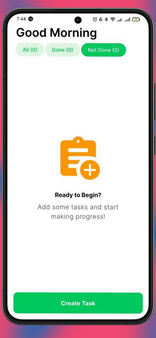


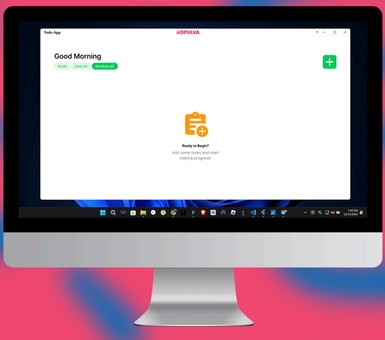
for done tasks



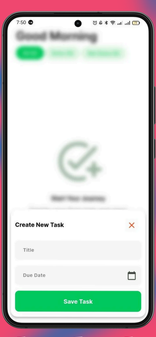


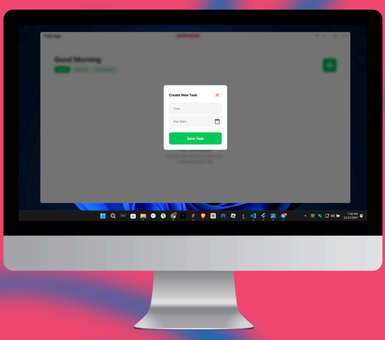
for not done tasks



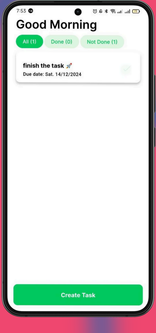


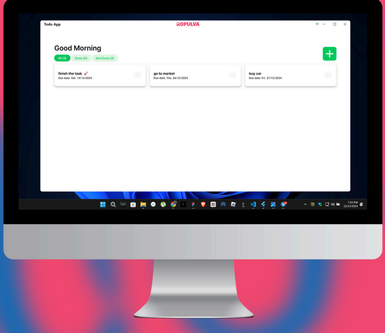
add task ui





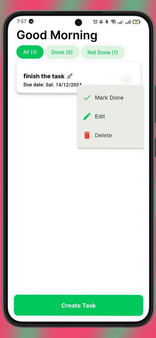
task item ui

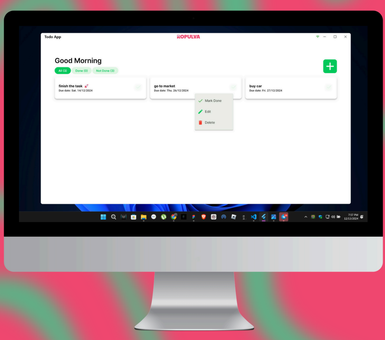




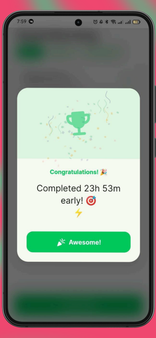
UI screen shots

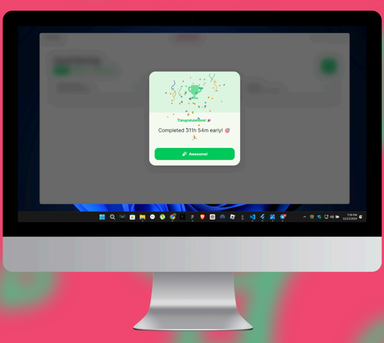
edit delelt and change states menu



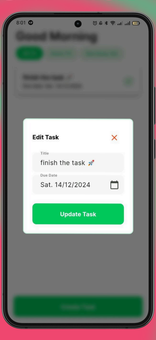


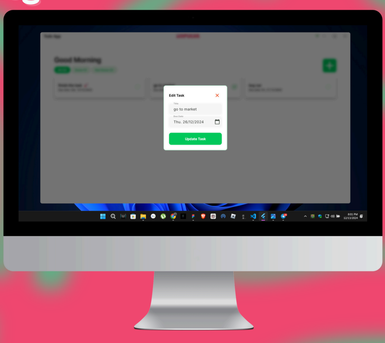
congratulations dialog when u done task 🎊



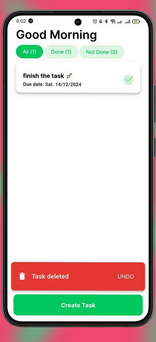


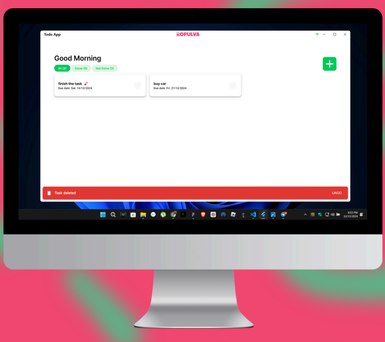
edit task dialog



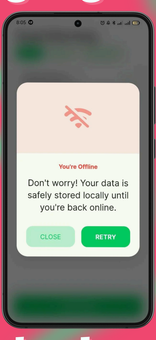


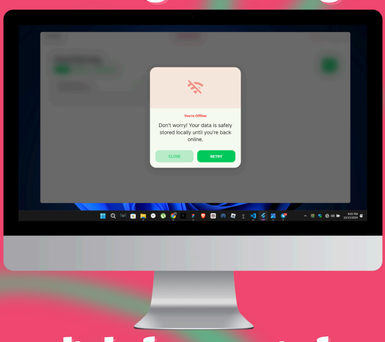
delet task snak bar with undo function



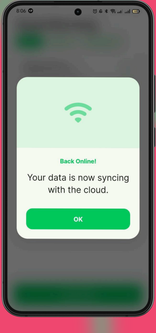


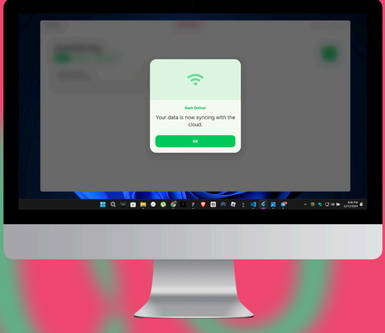
going offline dialog warming





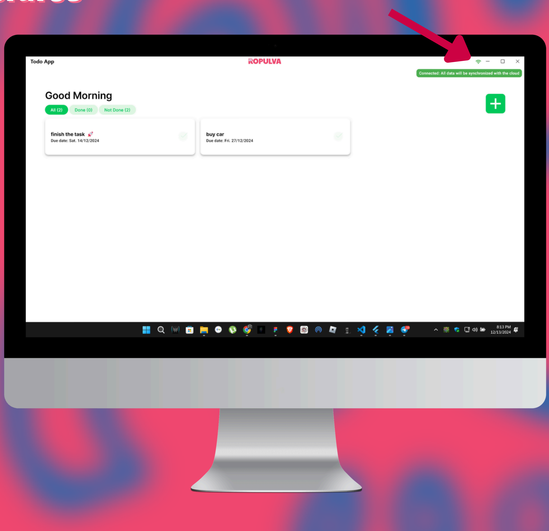
back online dialog warming



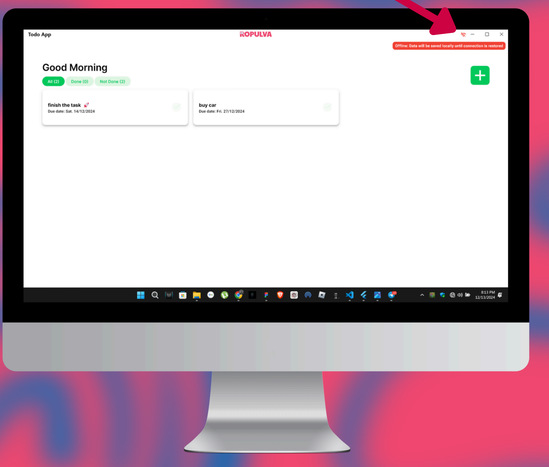


UI for windows only screen shots

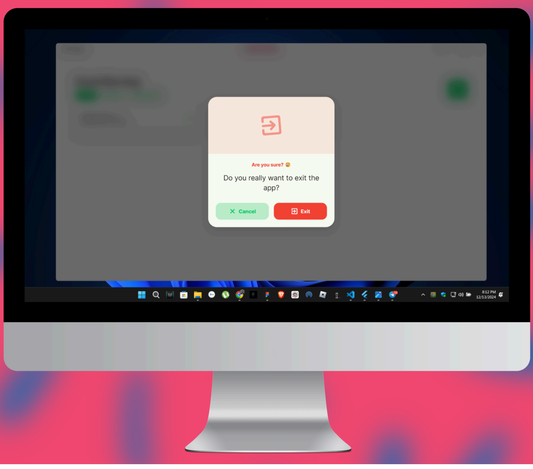
wifi icon for internet provid internet states



wifi icon for internet provid internet states



not so easy to exit the app dialog :D



TASK APP ANIMATION

Documentation

TASK LIST ANIMATIONS
- STAGGERED LIST ENTRY
  - WHEN: OPENING TASK LIST 📂
  - EFFECT: TASKS SLIDE IN ONE BY ONE 👉
  - PURPOSE: REDUCES VISUAL OVERWHELM, ADDS POLISH ✨
  - USER IMPACT: SMOOTHER TRANSITIONS BETWEEN SCREENS 📱

TASK CARD INTERACTIONS
- HOVER EFFECTS
  - WHEN: MOUSE OVER TASK CARDS 🖱️
  - EFFECT:SUBTLE ELEVATION INCREASE 🔼
  - PURPOSE: INDICATES INTERACTIVITY 👆
  - USER IMPACT: BETTER DESKTOP USABILITY 💻

- SWIPE ACTIONS
  - WHEN: SWIPING TASK CARDS 📲
  - EFFECT: REVEAL DELETE/COMPLETE ACTIONS ✅🗑️
  - PURPOSE: QUICK TASK MANAGEMENT ⚙️
  - USER IMPACT: FASTER TASK COMPLETION ⚡

- PROGRESS ANIMATIONS
  - CIRCULAR PROGRESS
    - WHEN: TASK COMPLETION UPDATES 🔄
    - EFFECT: SMOOTH PERCENTAGE CHANGES 📈
    - PURPOSE: VISUAL FEEDBACK 👀
    - USER IMPACT: CLEAR PROGRESS INDICATION ✅

- CELEBRATION ANIMATIONS
  - COMPLETION CONFETTI
    - WHEN: TASK/MILESTONE COMPLETION 🎉
    - EFFECT: PARTICLE BURST ANIMATION 🎊
    - PURPOSE: POSITIVE REINFORCEMENT 💪
    - USER IMPACT: INCREASED MOTIVATION 😊

- STATE TRANSITIONS
- LOADING STATES
- WHEN: DATA FETCHING ⏳
- EFFECT: SHIMMER LOADING EFFECT 🌫️
- PURPOSE: LOADING INDICATION ℹ️
- USER IMPACT: REDUCES PERCEIVED WAIT TIME 🕒
-

- ERROR STATES
- WHEN: ERROR OCCURS ⚠️
- EFFECT: SHAKE ANIMATION 🔀
- PURPOSE: ERROR FEEDBACK 🚨
- USER IMPACT: CLEAR ERROR INDICATION ❗

- DIALOG ANIMATIONS
- MODAL ENTRY/EXIT
- WHEN: OPENING/CLOSING DIALOGS 💬
- EFFECT: SCALE AND FADE 🌓
- PURPOSE: SMOOTH TRANSITIONS 👌
- USER IMPACT: PROFESSIONAL FEEL 💼

- FILTER ANIMATIONS
- CHIP SELECTION
- WHEN: SELECTING FILTERS 🏷️
- EFFECT: SMOOTH COLOR TRANSITION 🎨
- PURPOSE: SELECTION FEEDBACK 📌
- USER IMPACT: CLEAR ACTIVE STATE 👀

- ANIMATION PHILOSOPHY 🧠
- PURPOSEFUL: EACH ANIMATION SERVES A FUNCTION 🎯
- SUBTLE: NOT DISTRACTING FROM CORE TASKS 🤫
- CONSISTENT: SIMILAR ACTIONS = SIMILAR ANIMATIONS 🔄
- PERFORMANCE: OPTIMIZED FOR SMOOTH OPERATION 🚀
- ACCESSIBILITY: CAN BE REDUCED/DISABLED ♿

- THIS COMPREHENSIVE ANIMATION SYSTEM CREATES A POLISHED, PROFESSIONAL FEEL WHILE IMPROVING USABILITY AND USER ENGAGEMENT. ✨

task Statistics!

📊 FINAL STATISTICS:

📁 TOTAL FILES: 46

📝 TOTAL LINES: 4581

⚙️ TOTAL FUNCTIONS: 265

🔢 TOTAL VARIABLES: 218

FOR GIT HUB REPO

FOR ANDROID ABK

FOR DESKTOP EXE