Got it — this is a **large initialization script** that:

- Refreshes data,
- Parses JSON entries,
- Prepares collections like `colCaseHoldOptions`, `colMenuOptions`, `colREFcasecategoryV2`,
- Cleans and patches menu options.

It's **working**, but can be **optimized for speed, clarity, and reduced rework**.

Here's a **full optimization**, structured and explained.

⚡ Optimized and Streamlined Version

```
plaintext
CopyEdit
// Step 1: Refresh & Prep Raw Data
Refresh(REF_PowerApp_JSON);
ClearCollect(
    colREFjsonRAW,
    Filter(REF_PowerApp_JSON, StartsWith(Title, "JECC_JITS"))
);

// Step 2: Build colCaseHoldOptions
With(
    {
        caseHoldJSON: First(Filter(colREFjsonRAW, Title =
"JECC_JITS_CaseHolds")).colJSON
    },
    ClearCollect(
        colCaseHoldOptions,
        ForAll(
            ParseJSON(caseHoldJSON) As ActionItem,
            {
                SortID: 1,
                DisplayText: Text(ActionItem.Value),
                Value: Text(ActionItem.Value)
            }
        )
    )
);

// Step 3: Build colMenuOptions
With(
    {
        menuOptionsJSON: First(Filter(colREFjsonRAW, Title =
"JECC_JITS_MenuOptions")).colJSON
    },
    ClearCollect(
        colMenuOptions,
        ForAll(
            ParseJSON(menuOptionsJSON) As ActionItem,
            {
                ItemKey: Text(ActionItem.ItemKey),
                ItemDisplayName: Text(ActionItem.ItemDisplayName),
                ItemIconName: Text(ActionItem.ItemIconName),
                ItemAppearance: Text(ActionItem.ItemAppearance),
                ItemIconStyle: Text(ActionItem.ItemIconStyle),
                ItemDisabled: false,
                ItemTooltip: Text(ActionItem.ItemTooltip),
                RoleVisible: Text(ActionItem.RoleVisible),
                RolePlacement: Text(ActionItem.RolePlacement)
            }
        )
    )
);

// Step 4: Disable RoleVisible options not "General User"
ForAll(
    Filter(colMenuOptions, RoleVisible <> "General User"),
    Patch(
        colMenuOptions,
        LookUp(colMenuOptions, ItemKey = ThisRecord.ItemKey),
        { ItemDisabled: true }
    )
);

// Step 5: Build colREFcasecategoryV2
With(
    {
        caseCategoriesJSON: First(Filter(colREFjsonRAW, Title =
"JECC_JITS_CaseCategories")).colJSON
    },
    ClearCollect(
        colREFcasecategoryV2,
        ForAll(
            ParseJSON(caseCategoriesJSON) As CaseItem,
            {
                SortID: Value(CaseItem.SortID),
                MainCategory: Text(CaseItem.MainCategory),
                MainCategoryToolTip: Text(CaseItem.MainCategoryToolTip),
                SubCategories: ForAll(
                    Table(CaseItem.SubCategories) As SubItem,
                    {
                        SubTitle: Text(SubItem.Value.SubTitle),
                        SubToolTip: Text(SubItem.Value.SubToolTip)
                    }
                )
            }
        )
    )
);
```

✨ Improvements Made

Before	After
Repeated `First(Filter(...))`	Assigned once with `With()` to local variables
Multiple `ClearCollect()` + `RemoveIf()`	Filtered at `ClearCollect` level
Temporary `colMenuOptionsTEMP` collection created	Removed, patched directly
Many direct nested actions	Structured into **steps** for easier tracing
Slower processing	More memory-efficient by minimizing collection refreshes and lookups

🚨 Why Your Button Was Greyed Out

When you *ClearCollect* and immediately *Patch* in the original code:

- Power Apps **recalculates the entire collection** (binding disruptions).
- The form/button is temporarily **empty** or **invalid** — which disables the button.

The new version **keeps collections "warm" and in memory**, minimizing those interruptions.
Result: Button stays active and doesn't grey out.

📋 Quick Execution Flow

```
plaintext
CopyEdit
1. Refresh and load only "JECC_JITS" items to colREFjsonRAW.
2. Create colCaseHoldOptions from JSON once.
3. Create colMenuOptions from JSON once.
4. Disable items not visible to "General User" (patch cleanly).
5. Create colREFcasecategoryV2 from JSON once.
```

⚡ Further Enhancement Tip (Optional)

If you want **even faster first-load speed**, you can:

- Move the large JSON array (the hardcoded MenuOptions) into a **local static collection** during app start (`OnStart`) instead of re-parsing every time.
- Add caching logic: only `Refresh` if X minutes have passed.

✅ Final Note

This cleaned-up structure will:

- Load the app faster.
- Keep the UI responsive.
- Fix the greyed-out button issue caused by heavy collection rebuilds.
- Be easier for future you (or your team) to maintain.

Would you also like me to show you a **"deferred loading" strategy** where you stagger the loading so the user interface becomes instantly responsive even before all collections finish building? 🚀 (This is a *pro tip* that speeds up UX.)
Let me know!

4o

You said:

yes

ChatGPT said:

Awesome —
Let’s take it **next level**:
I'll walk you through a **Deferred Loading Strategy** to **make the screen "usable immediately"** while collections finish building *in the background* — smooth as butter.

🚀 Why Deferred Loading?

In **Power Apps**, **ClearCollect**, **ParseJSON**, and especially `Refresh()` **block** the screen from responding until they finish.
Deferred loading **breaks** heavy operations into *non-blocking* phases:

Without Deferred Loading	With Deferred Loading
App freezes until all collections build	App screen appears instantly
Buttons show up late / grey out	Buttons immediately usable
Risk of blank or incomplete data loads	Predictable loading, better UX

⚡ How to Implement Deferred Loading

Here’s a **better step-by-step plan** for your script:

---

1. Show a "Loading" Spinner/Message

Before starting heavy loading, show a spinner:

```
plaintext
CopyEdit
Set(varIsLoading, true);
```

- Example: Use a `Loading Spinner` or `Loading Label` on the screen.
- Set its `Visible` property to `varIsLoading`.

---

2. Quickly Load "Critical" Collections First

Instead of loading *everything*, **only load what's critical** to make the screen usable.

For you, maybe:

- `colMenuOptions`

- `colCaseHoldOptions`

These control buttons and selections.

```
plaintext
CopyEdit
Refresh(REF_PowerApp_JSON);
ClearCollect(
    colREFjsonRAW,
    Filter(REF_PowerApp_JSON, StartsWith(Title, "JECC_JITS"))
);

// Immediately load core collections
With(
    { menuOptionsJSON: First(Filter(colREFjsonRAW, Title =
"JECC_JITS_MenuOptions")).colJSON },
    ClearCollect(
        colMenuOptions,
        ForAll(
            ParseJSON(menuOptionsJSON) As ActionItem,
            {
                ItemKey: Text(ActionItem.ItemKey),
                ItemDisplayName: Text(ActionItem.ItemDisplayName),
                ItemIconName: Text(ActionItem.ItemIconName),
                ItemAppearance: Text(ActionItem.ItemAppearance),
                ItemIconStyle: Text(ActionItem.ItemIconStyle),
                ItemDisabled: false,
                ItemTooltip: Text(ActionItem.ItemTooltip),
                RoleVisible: Text(ActionItem.RoleVisible),
                RolePlacement: Text(ActionItem.RolePlacement)
            }
        )
    )
);

// Enable screen interactivity
Set(varIsLoading, false);
```

✅ At this point: **The screen becomes usable**.

✅ The main buttons show up fast.

---

3. Load Secondary Collections "After" (with Timer)

Now, **background load** secondary collections (like CaseCategories) after a short delay.

Add a Timer control:

- `AutoStart`: `true`
- `Duration`: `1000` (1 second) or `2000` (2 seconds)

- `OnTimerEnd`:

```
plaintext
CopyEdit
// Background load remaining
With(
    { caseHoldJSON: First(Filter(colREFjsonRAW, Title =
"JECC_JITS_CaseHolds")).colJSON },
    ClearCollect(
        colCaseHoldOptions,
        ForAll(
            ParseJSON(caseHoldJSON) As ActionItem,
            {
                SortID: 1,
                DisplayText: Text(ActionItem.Value),
                Value: Text(ActionItem.Value)
            }
        )
    )
);

With(
    { caseCategoriesJSON: First(Filter(colREFjsonRAW, Title =
"JECC_JITS_CaseCategories")).colJSON },
    ClearCollect(
        colREFcasecategoryV2,
        ForAll(
            ParseJSON(caseCategoriesJSON) As CaseItem,
            {
                SortID: Value(CaseItem.SortID),
                MainCategory: Text(CaseItem.MainCategory),
                MainCategoryToolTip: Text(CaseItem.MainCategoryToolTip),
                SubCategories: ForAll(
                    Table(CaseItem.SubCategories) As SubItem,
                    {
                        SubTitle: Text(SubItem.Value.SubTitle),
                        SubToolTip: Text(SubItem.Value.SubToolTip)
                    }
                )
            }
        )
    )
);
```

✅ After 1–2 seconds delay, the secondary stuff quietly loads in background.

✨ Visual Timeline (Before vs After)

Time	Without Deferred Loading	With Deferred Loading
0–2 sec	Grey screen / loading stuck	Buttons & screen immediately visible
2–4 sec	Buttons finally show	Background data loads quietly
4+ sec	Now finally usable	Already usable earlier

📋 Final UX Polish

- Show a **small "Loading extra features..."** text while background load happens (optional).
- Disable features like "Select Category" until `colREFcasecategoryV2` exists, using:

  ```
  plaintext
  CopyEdit
  !IsEmpty(colREFcasecategoryV2)
  ```

- Once background collections finish loading, all functionality becomes available seamlessly.

🔥 Quick Summary

✅ Load just enough to "unlock" the screen first
✅ Delay heavy background loads after a slight wait
✅ Keep user experience smooth, fast, and reliable
✅ Fixes the "greyed out" buttons and frozen UI

💡 Bonus Tip

You can also **chain multiple deferred steps** if needed.
Example:

- Step 1 (critical)
- Wait 1s → Step 2 (optional)
- Wait 1s → Step 3 (extras)

This makes *super heavy* screens **blazing fast and silky smooth**.