Alternatives to custom button URL "hacks" for Lightning Experience

Custom buttons with URL "hacks" to pre-populate fields on record create or edit pages are one of the most common things I see on Lightning readiness reports. In this post, I'll cover some solutions for recreating this functionality in Lightning Experience.

The Requirement

For Salesforce Administrators, Developers and Consultants this has to be one of the most commonly heard user stories:

> As a user, I want to be able to create a new related record from a record detail page and pre-populate the following fields using values from the current record...

A custom button with a URL "hack" is the most common solution to meet this requirement. For example, if you wanted a button on Account records that allowed you to create a new Contact with the mailing address populated using the values of the Account billing address, the URL for the button would look like this:

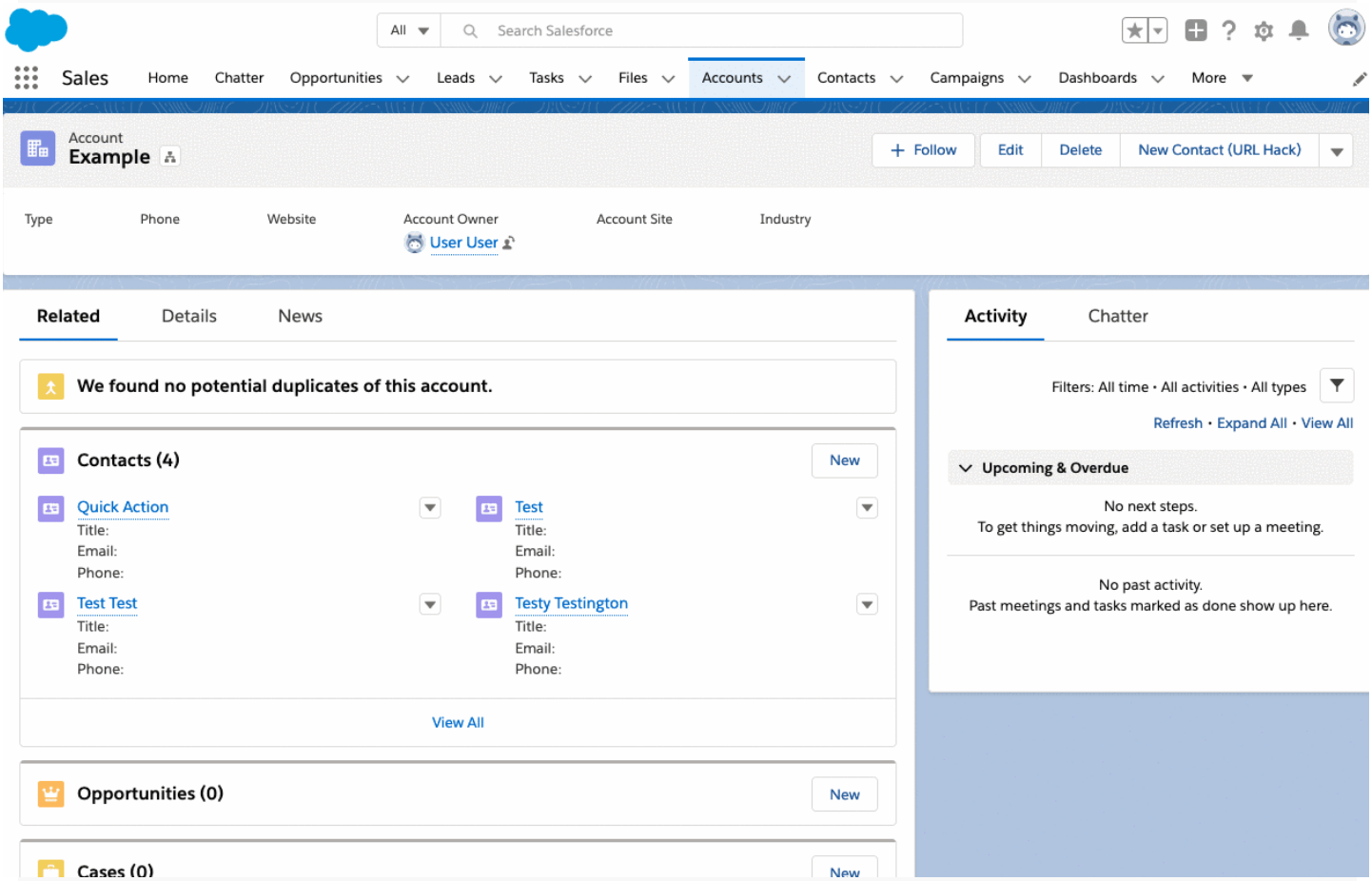
```
/setup/ui/recordtypeselect.jsp?ent=Contact&save_new_url=/003/e?con4_lkid=
{!Account.Id}&con4={!Account.Name}&con19street=
{!Account.BillingStreet}&con19city={!Account.BillingCity}&con19zip=
{!Account.BillingPostalCode}&con19state={!Account.BillingState}&con19country=
{!Account.BillingCountry}
```

Note: This will URL assumes there are record types defined for Contact. Remove everything before `/003` if you don't have any Contact record types or you want to always use the default.

This was *never* a solution officially supported by Salesforce. However, before Quick Actions came along, this was the only option available without writing code. Even after Quick Actions came along, URL "hacks" are still preferred by some because they support record type selection or using the default defined on the running user's profile.

Unfortunately, URL "hack" buttons will stop working in Lightning Experience. If you've taken some time to test them you'll find that although the buttons appear in Lightning, and even open the record

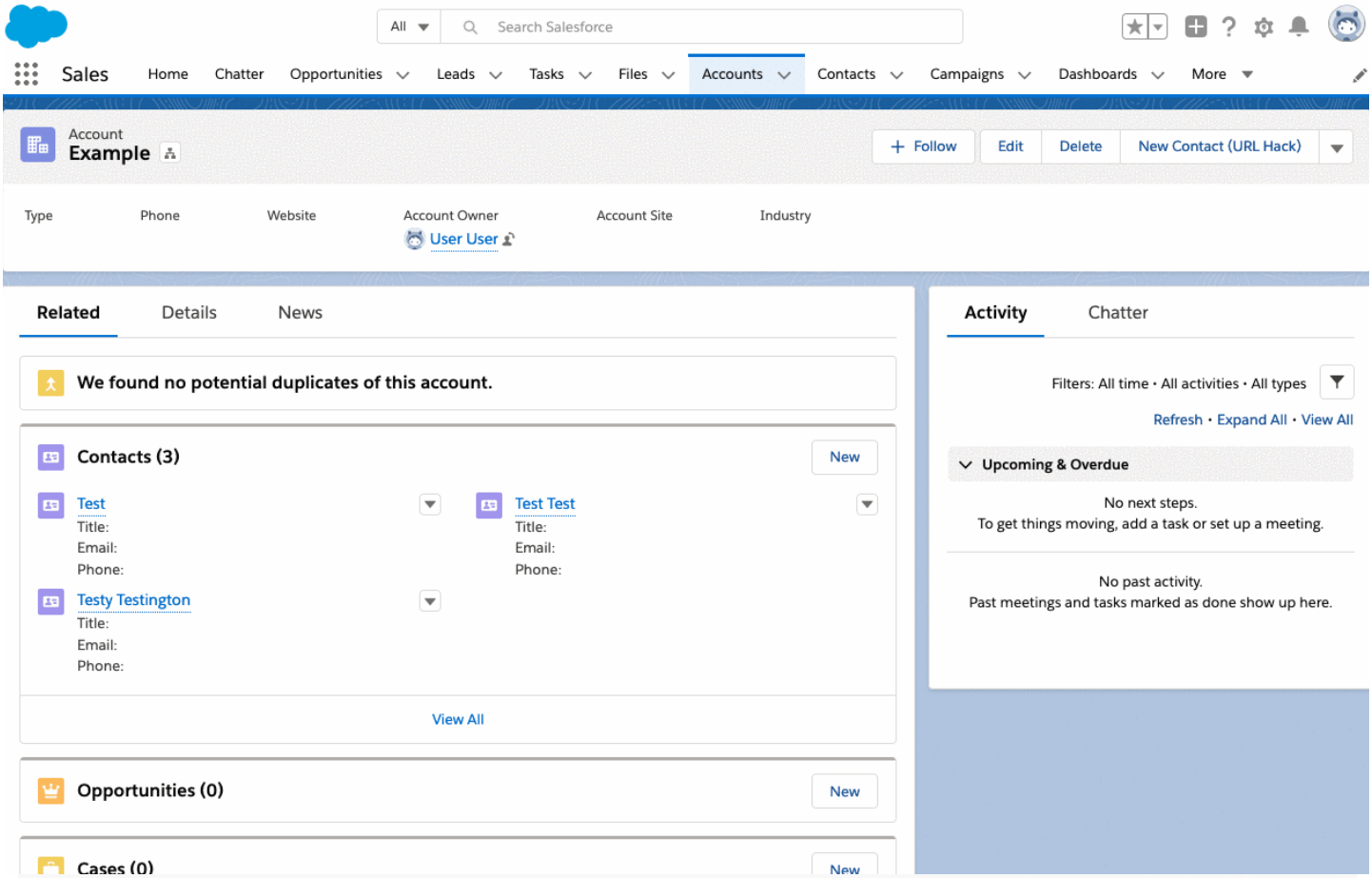
edit screen when clicked, the field mappings do not work. Note in this GIF, the Account Name field and the Mailing Address fields are blank.



So, with Lightning Experience auto-activation coming in to force in the Winter '20 release, let's take a look at what the Lightning ready alternatives to URL "hacks" look like.

Solution #1 — Quick Action

Create a quick action and use the predefined values feature to map field values to the new record.



Pros

- This is a simple "clicks not code" solution that a Salesforce Administrator can set up in a reasonably short amount of time.
- The quick action will be available in Classic, Lightning Experience *and* mobile.
- Errors generated from saving the record, such as validation rules firing, are handled gracefully by the quick action.
- Quick actions allow you to define a layout independent of the full page layout. This allows you to display a more relevant subset of fields for the action, while the full page layout remains available for general create and edit activity.

Cons

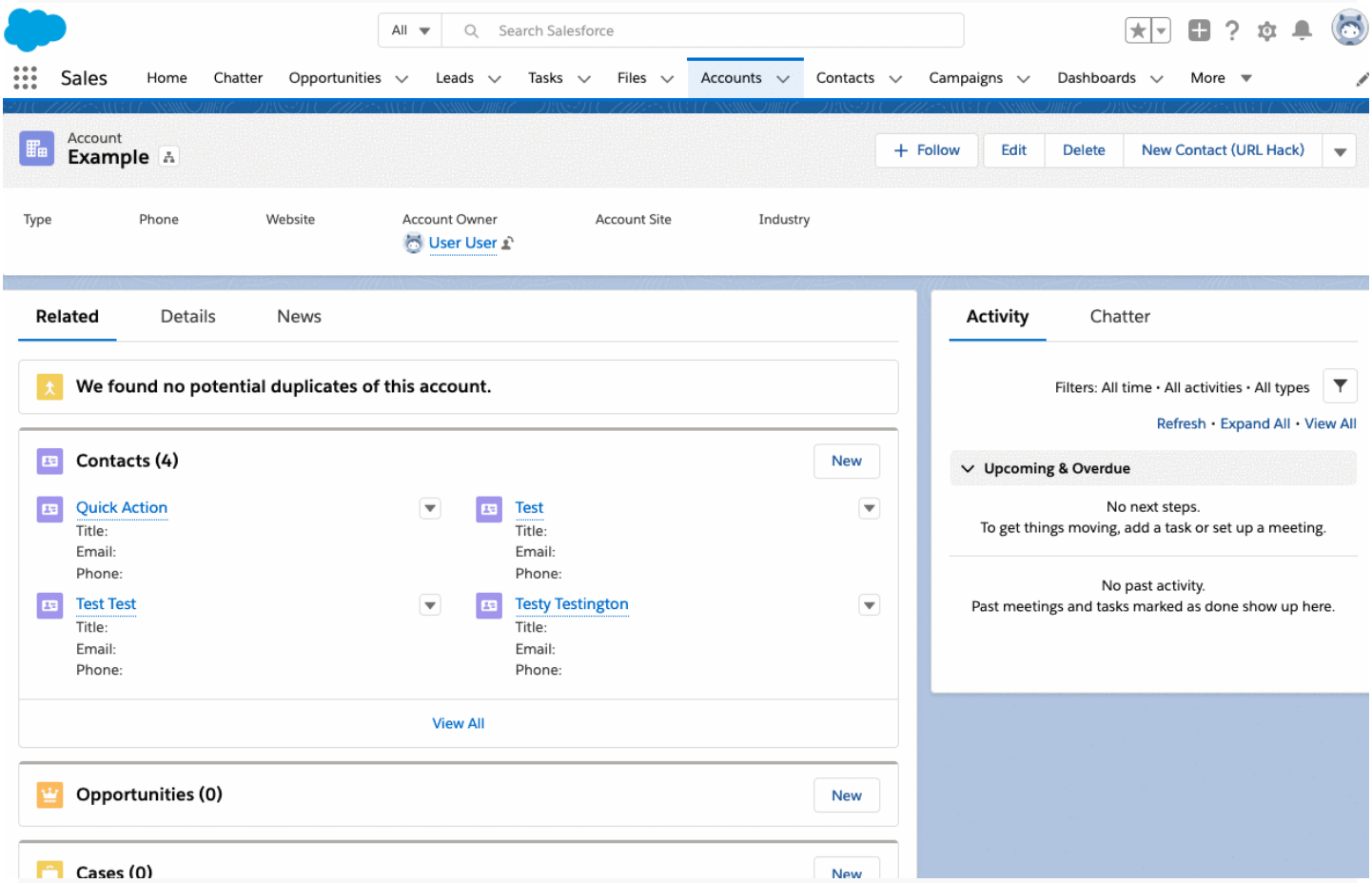
- If the target object has record types defined, you **must** specify a single record type for the quick action. At the time of writing, it's currently not possible to make record type selection part of the quick action, and it's not possible to use the default specified on the running user's profile.
- In Classic the quick action will show in the chatter publisher, whereas in Lightning Experience it will show as a button.
- As mentioned, quick action layouts are independent of page layouts, so if you don't want to display a subset of fields you'll need to add all fields to the action layout and manually keep it in sync with your page layout.

Verdict

If you don't have record types to worry about, this is the simplest and cleanest solution. If you had a small number of record types you could consider creating a quick action per record type.

Solution #2 — Quick Action with a Flow

Create a quick action that displays a visual flow. The flow will have a screen for record type selection, and subsequent screens for completing fields on the record with default values mapped from the current record.



Pros

- The flow can be created without any code, although with some drawbacks.
- The flow can easily query out and display a list of record types.
- The powerful capabilities of visual flows create plenty of opportunities to streamline business processes and make them more user-friendly.

Cons

- When a flow launched via a quick action finishes, it starts over again rather than closing. To handle the flow finish behaviour gracefully a small Aura component will be required. Thankfully, it's a relatively simple component that can be reused. Check out this example in the repo or follow the instructions in this knowledge article to build it.
- Classic isn't supported using this solution, the button simply won't show on the page layout for classic users.
- The flow will display **all** record types, regardless of whether or not they are assigned to the running user's profile. This is because the flow queries the `RecordType` object which at the time of writing doesn't seem to respect the available record types defined at the profile level. (This is possibly a known issue).
- Error handling can be added to the flow using a fault connector, but it's not particularly graceful as the fault message often includes error codes and other information that isn't user-friendly.

Verdict

This solution improves on a standard quick action by offering record type selection, but it's not perfect.

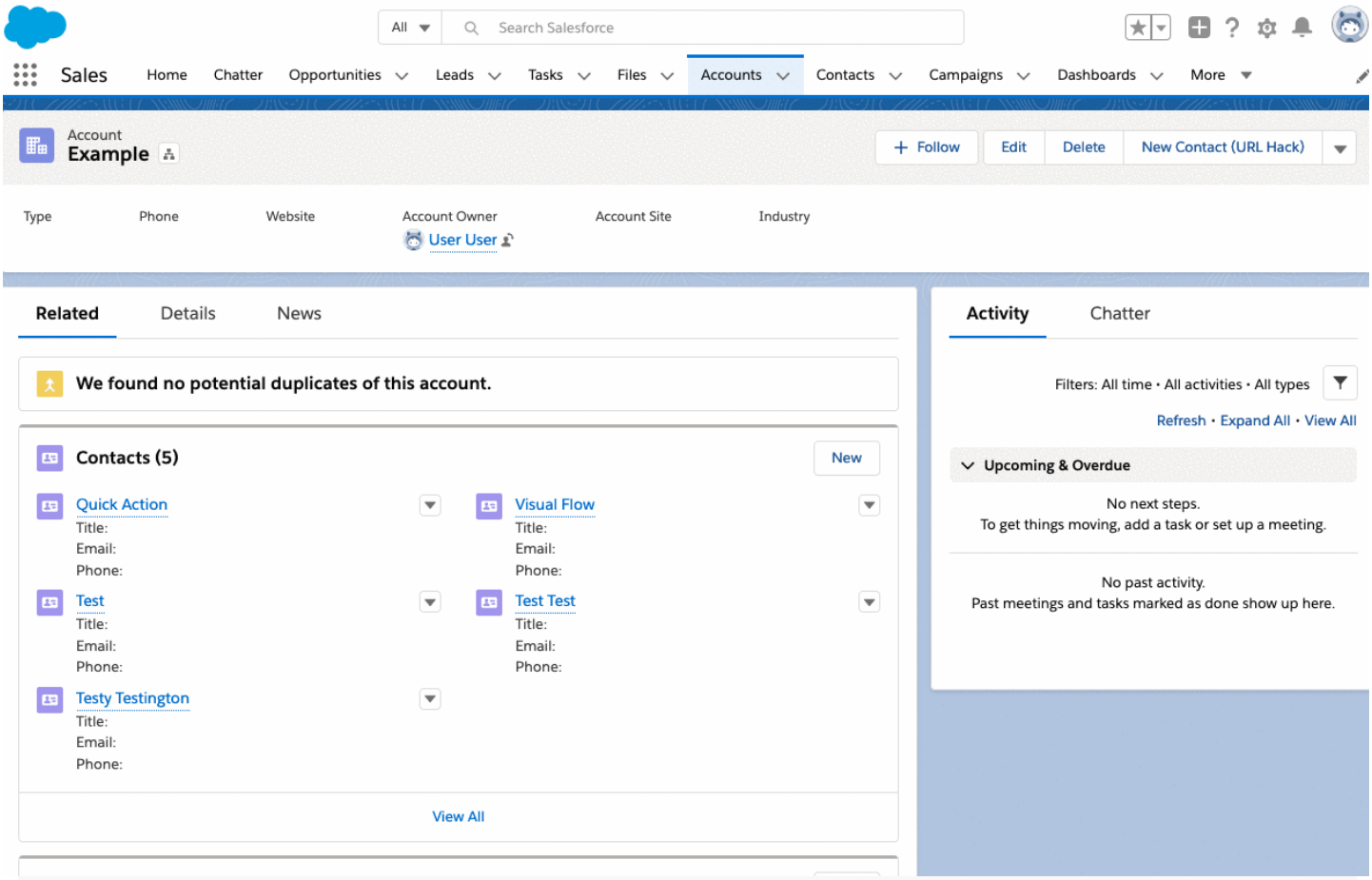
If you want to make sure users can only use record types they have access to i.e. assigned to their profile, you'll probably need to look at using APEX to retrieve the record types and return them to the flow using an `@InvocableMethod` .

If you want to make the flow available to Classic users, you could create a simple URL button that navigates to the flow url, but you'll also need to handle the finish behaviour using the `retURL` parameter. If you want to navigate to the newly created record in Classic, you'll need to embed the flow in a Visualforce page.

All of these workarounds add a lot of undesirable complexity to this solution.

Solution #3 — Custom Button with a Visualforce "Dispatcher" Page

An alternative to quick actions is to create a custom button that displays a Visualforce page. Visualforce pages displayed in Lightning Experience have access to the `sforce.one` Javascript object which can be used to fire Lightning navigation events from within a Visualforce page. What does this mean, and how can it be used? Using Javascript on the Visualforce page, check `UItheme.getUITheme()` to establish if the user is in Classic or Lightning Experience, and call either `sforce.one.createRecord()` for Lightning Experience or the original URL "hack" for Classic.



Here's what the code would look like for the New Contact example:

```
<apex:page standardController="Account" showHeader="true">
    <script>
        if (
            UITheme.getUITheme() === 'Theme4d' || // Lightning Experience
            UITheme.getUITheme() === 'Theme4u' || // Lightning Console
            UITheme.getUITheme() === 'Theme4t'    // Mobile
        ) {
            sforce.one.createRecord('Contact', null, {
                'AccountId': '{!Account.Id}',
                'MailingStreet': '{!Account.BillingStreet}',
                'MailingCity': '{!Account.BillingCity}',
                'MailingState': '{!Account.BillingState}',
                'MailingPostalCode': '{!Account.BillingPostalCode}',
                'MailingCountry': '{!Account.BillingCountry}'
            });
        }
        else {
            window.top.location = '/setup/ui/recordtypeselect.jsp?ent=Contact&s
        }
```

```
    </script>
</apex:page>
```

Note: Don't forget to set "Available for Lightning Experience, Lightning Communities, and the mobile app" to `true` for the page. You must also explicitly set `showHeader="true"` on the page to workaround this known issue.

Pros

- This is a relatively simple coded solution that works in both Classic, Lightning Experience and Mobile.
- The default record type for the running user's profile will be used if `null` is passed to `sforce.one.createRecord` instead of a `recordTypeId` .
- This solution will utilise existing page layouts.

Cons

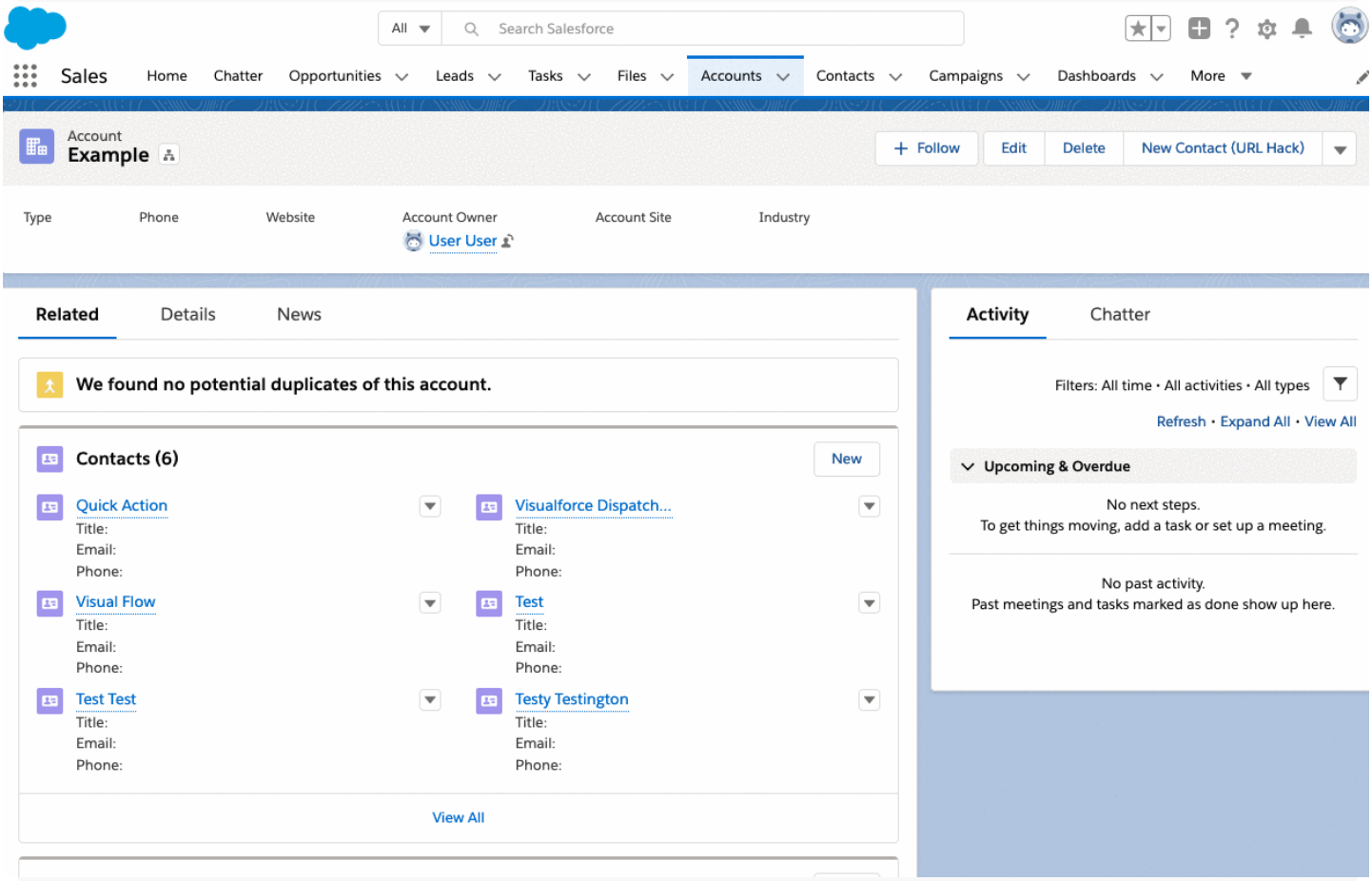
- If you need to support record type selection this solution will not work in Lightning Experience or Mobile without a significant amount of additional code. This is because in Classic the standard record type selection is available as part of the original URL "hack", but in Lightning Experience, there's currently no way of invoking the standard record type selection.
- In Lightning Experience and Mobile, if the user clicks "Cancel" instead of creating a new record, they will be left staring at the blank Visualforce page. You could put a link back to the original record on the page as a workaround, but it's not ideal.
- Creating a new Visualforce page to support Lightning feels a little dirty. Visualforce isn't going anywhere for a while, but in the longterm, it will most likely be retired so this solution will ultimately need to be replaced.

Verdict

If you need a solution that always uses the running user's default record type, without needing to offer record type selection this solution trumps Quick Actions. Using a Quick Action instead of a Custom Button to display the Visualforce page solves the blank page on Cancel problem in Lightning but not on Mobile. If you need to allow users to select a record type, other solutions are likely to be more suitable.

Solution #5 — Quick Action with a Lightning Component

If code is not a barrier, create a quick action that displays a custom Lightning component. The Lightning component, with the help of an APEX controller, can display record type options and then fire the create record event.



Pros

- The APEX controller can retrieve record types based on the running user's profile and even retrieve the default.
- The Lightning component could be used to display a custom UI for creating the new record, or it could simply fire the create record event. The latter would utilise the existing page layouts.

Cons

- This approach will not work in Classic, although the original URL "hack" button might be a suitable fallback.
- This is an entirely coded solution, which might be a barrier for some.
- When firing the create record event, any fields with default values specified **must** be visible on the page layout, otherwise, an internal server error is thrown.
- The Lightning Component will need to be built as an Aura Component rather than a Lightning Web Component (LWC). This is because it is currently not possible to use Lightning Web Components in quick actions without wrapping them with an Aura component that implements the `force:lightningQuickAction` interface. More importantly, it is not possible to specify default field values when using the `NavigationMixin` in LWC.

Verdict

This solution overcomes most of the drawbacks of the other solutions, but you will most likely need the support of a developer to deliver it.

Conclusion

Unfortunately, none of the current alternatives to URL "hack" custom buttons is perfect. The solution that works best to replace your URL "hacks" will very much depend on the requirements of the business process they are currently supporting. Standard quick actions may help you simplify the process by focusing on only the fields that are relevant to a particular business process. Whereas a visual flow may help you break a particularly complex business process into a much easier to use step by step process.

One thing is clear, if you need to support record type selection, and you want to only display the record types available to users based on the profile then the clear winner here is the Quick Action with a Lightning Component.

All of the solutions are available for free here under the terms of the MIT licence, try them out in a developer or scratch org.