

# Build an Application: Step-by-Step #8

Exercise to Accompany  
Interfaces 101

The Appian Step-by-Step series consists of 12 exercises that accompany the courses in the Appian Developer learning path. Exercises build upon each other. Complete exercises in order and keep the app and all objects until you are done with the project.

- 1 Welcome to the Appian Developer Learning Path
- 2 Create an Application
- 3 Manage Users and Groups
- 4 Expressions
- 5 Design Record Types
- 6 Sites
- 7 Query Your Data
- 8 Interfaces 101**
- 9 Process Modeling 101: Part 1
- 10 Process Modeling 101: Part 2
- 11 Reports
- 12 Task Report

<b>Exercise 8: Interfaces</b>	<b>3</b>
Update the Summary View Interface	4
Add a Vehicle Image	7
Update the Maintenance Section Layout	7
Edit the Create Vehicle Form	8
Update Fields and Labels	8
Configure the File Upload Component	11
Configure the Create Button	11
Create the Supervisor Approval Form	12
Create a Reusable Interface	15
<b>Troubleshooting Resources</b>	<b>17</b>

## Notice of Rights

This document was created by Appian Corporation, 7950 Jones Branch Dr, Tysons, Virginia 22102. Copyright 2025 by Appian Corporation. All rights reserved. Information in this document is subject to change. No part of this document may be reproduced or transmitted in any form by any means, without prior written permission of Appian Corporation. For more information on obtaining permission for reprints or excerpts, contact Appian Training at [academyonline@appian.com](mailto:academyonline@appian.com).

This exercise was developed for **Appian 25.3**. If Appian Community Edition is on a later Appian version, functionality might be different. Go to [academy.appian.com](https://academy.appian.com) to download the latest exercise.

## Exercise 8: Interfaces

In this exercise, you will modify or create the following interfaces:

1. W#SA\_VehicleSummary
2. W#SA\_CreateVehicle
3. W#SA\_SupervisorForm

**W#SA\_VehicleSummary** is used in the Summary view to display read-only vehicle information.

**W#SA\_CreateVehicle** is used to start the W#SA Create Vehicle process model to add a new vehicle to the fleet.

**W#SA\_SupervisorForm** is used by supervisors in the W#SA Add Vehicle process model to review a new vehicle.


To create efficient, intuitive, and attractive interfaces, it is important to follow UX design best practices and recommendations. Visit the SAIL Design System in [Appian Documentation](#) to learn more.

## Update the Summary View Interface

When you generated the vehicle summary view in an earlier exercise, the **W#SA\_VehicleSummary** interface was created for you. In this section, you will update this interface to better display the vehicle information. Your interface will look like the image below.

**Details**

**Image**



**Year**  
2021

**Make**  
Ford

**Model**  
F150

**Mileage**  
500

**Mileage Category**  
Low Mileage

**Next Maintenance Date**  
January 4, 2021

**Vehicle Category**  
Convertible

**Vehicle Condition**  
Like New

**Vehicle Status**  
Inactive

**Maintenance Requests**

**SEARCH**

Request Id

Vehicle Id

Is Scheduled

Status

Start Date

End Date

Cost

Assigned Mechanic

00001

1

✗

Completed

5/5/2020

5/8/2020

50

mike.mechanic

00042

1

✗

Completed

5/11/2020

5/18/2020

50

fred.fixit

00043

1

✗

Completed

5/11/2020

5/15/2020

200

mike.mechanic

00059

1

✗

Completed

5/12/2020

5/14/2020

900

mike.mechanic

00060

1

✓

Completed

6/1/2020


6/1/2020


70

mike.mechanic


< 1 - 5 of 6 >

**Contacts**

 Added By


 Modified By

**Event History**

 Add a comment

Subscribe

0

  
No events to display

Follow the steps below to update this interface.

1. In your application, open **W#SA\_VehicleSummary**.

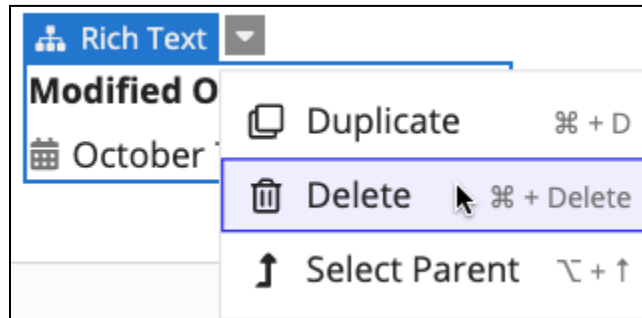
Appian Step-by-Step 25.3

© Appian Corporation, 2025

4

2. Configure the following:

- Delete the following fields: **Color, Last Maintenance Date, Image, Added On, Modified On, Added By, Modified By, Cost Sum, Count of Maintenance Requests, Min Event Timestamp, Max Event Timestamp, Duration, and Is Active.**



- Click the plus (+) icon to add a third column for each of the first three rows of columns.
- Rearrange the components by dragging and dropping them. Rearrange the fields to match the following layout:

### Details

Drop component here	Drop component here	Drop component here	+
<b>Year</b> 2021	<b>Make</b> Ford	<b>Model</b> F150	+
<b>Mileage</b> 500	<b>Mileage Category</b> Low Mileage	<div>Rich Text</div> <b>Next Maintenance Date</b> January 4, 2021	+
<b>Vehicle Category</b> Convertible	<b>Vehicle Condition</b> Like New	<b>Vehicle Status</b> Inactive	+
Drop component here	Drop component here	Drop component here	+
Drop component here	Drop component here	Drop component here	+

**NOTE:** When dragging components, you will notice a bold pink line. This indicates where the component will be placed.

- After rearranging the components, you will be left with several empty column layouts. Delete extra columns by clicking the trashcan icon to match the layout below. **You should be left with one empty column layout above year, make, and model.**

Drop component here	Drop component here	Drop component here	+
<b>Year</b> 2021	<b>Make</b> Ford	<b>Model</b> F150	+
<b>Mileage</b> 500	<b>Mileage Category</b> Low Mileage	<b>Next Maintenance Date</b> January 4, 2021	+
<b>Vehicle Category</b> Convertible	<b>Vehicle Condition</b> Like New	<b>Vehicle Status</b> Inactive	+

3. Click **SAVE**.

## Add a Vehicle Image

1. Drag and drop an **IMAGE** component into the first empty column layout at the top.
2. Click **+** in the **Image** component, and select **DOCUMENT IMAGE**.
3. In the **Component Configuration** pane, configure the following:
  - Under **Document**, click **a!EXAMPLE\_DOCUMENT\_IMAGE()**. Delete the expression, and enter:

```
if(
  a!isNullOrEmpty(
    ri!record[recordType!W#SA Vehicle.fields.image]
  ),
  a!EXAMPLE_DOCUMENT_IMAGE(),
  ri!record[recordType!W#SA Vehicle.fields.image]
)
```

Similar to how you configured the record list, this expression displays an example document image if a vehicle has no image.

- Click **OK**.
4. Under **Component Configuration**, click **Images**, then **Image** to return to go to the top-level configurations.
  5. Change **Label Position** to **Hidden**.
  6. Click **SAVE**.

## Update the Maintenance Section Layout

1. Select the Maintenance section layout.
2. Change the **Label** to `Maintenance Requests`.
3. Click **SAVE**.

**NOTE:** The Maintenance Requests grid is based on the current configuration of the W#SA Maintenance record list. If you want, you can update the grid directly in this interface. Updates to the grid in this interface will only be applied here.

4. In the **Build** view of your application, click the **W#SA Vehicle** record type. Next to the record type name, click **View Record List**. Click on a **VIN** to view the updated summary view interface.

## Edit the Create Vehicle Form

In this section, you will edit the generated W#SA\_CreateVehicle interface that starts the W#SA Create Vehicle process. Your goal is to improve the overall UX design of this form and to update a few fields so that they display the information correctly. When you are done, the interface will look like the image below.

The screenshot shows a web form titled "Add Vehicle" with the subtitle "Enter details for the vehicle". The form contains several input fields and dropdown menus. At the top, there are three input fields for "Year \*", "Make \*", and "Model \*". Below these is a single-line input field for "VIN \*". This is followed by a single-line input field for "Mileage \*", and another for "Color \*". Then there are two dropdown menus: "Vehicle Condition \*" with the placeholder text "Select a vehicle condition", and "Vehicle Category \*" with the placeholder text "Select a vehicle category". Below these are two date pickers: "Last Maintenance \*" and "Next Maintenance \*", both with the placeholder text "mm/dd/yyyy". At the bottom of the form is a "File Upload \*" section with an "UPLOAD" button and a dashed box containing the text "Drop or paste file here". At the very bottom of the form, there are two buttons: "CANCEL" on the left and "ADD VEHICLE" on the right.

Follow the steps below to update the Add Vehicle form.

### Update Fields and Labels

1. Open **W#SA\_CreateVehicle**
2. Change the **form** title to **Add Vehicle**.
3. Delete the following components: **Vehicle Status**, **Added By**, **Added On**, **Modified On**, and **Image**.
  - To delete, click the component, then click the dropdown arrow > **Delete**.



4. Remove the word **date** from the Last Maintenance and Next Maintenance field labels.
5. Drag and drop a **SIDE BY SIDE** layout above the **Make** field. It should be within the card layout.

**Add Vehicle**  
Enter details for the vehicle

Side By Side Layout ▼

Drop component here

**Make \***

**Model \***

**Color \***

**Vehicle Condition \***  
Select a vehicle condition ▼

6. Move the **Year**, **Make**, and **Model** fields inside of the Side By Side layout. The Side By Side layout will keep these fields next to each other on a narrower screen. For reference, use the image at the beginning of this section.

**NOTE:** On mobile devices, columns layouts are flattened into a single column. If you need certain fields to stay together, use side by side layouts.

7. Click on the **Make** text field.
8. Using the **Component Configuration** pane, change **Margin Above** to **Less**.
9. Move the **Vin** field below the Side By Side layout and change the label to **VIN**.
10. Move the **Mileage** field below **VIN**.
11. Drag and drop another **SIDE BY SIDE** layout below the **Next Maintenance** field.
12. Move the **Last Maintenance** and **Next Maintenance** date fields into the Side by Side layout.
13. Drag and drop a **FILE UPLOAD** component below the Side by Side layout.

At this point, your interface should resemble the image below.

**Add Vehicle**  
Enter details for the vehicle

Year\* Make\* Model\*

VIN\*

Mileage\*

Color\*

Vehicle Condition\*  
Select a vehicle condition

Vehicle Category\*  
Select a vehicle category

Last Maintenance\* Next Maintenance\*

mm/dd/yyyy mm/dd/yyyy

File Upload  
UPLOAD Drop or paste files here

CANCEL CREATE

14. Next, add a validation to the Next Maintenance field so that the next maintenance date entered must be after the last maintenance date.

- Click **Next Maintenance**.
- In the **Component Configuration** pane, find **Validations**. Next to **Validations**, click the **Edit as Expression** icon.
- In the Expression Editor, enter the following expression:

```
if(
  todate(
    ri!record[recordType!W#SA
      Vehicle.fields.nextMaintenanceDate]
  ) < todate(
    ri!record[recordType!W#SA
      Vehicle.fields.lastMaintenanceDate]
  ),
```

```
        "The next maintenance date must be after the last  
        maintenance date.",  
        null()  
    )
```

This expression checks whether the next maintenance date is before the last maintenance date. If it is, the field is invalid and displays an error message.

15. Click **OK**, then **SAVE**.

## Configure the File Upload Component

1. Click the **File Upload** component. Change the **Label** to Image.
2. In the **Component Configuration** pane, configure the following:
  - **Target:** Select **Document or folder** and enter **W#SA\_DOCUMENTS\_FOLDER\_POINTER**.
  - **Maximum Selections:** Enter 1.
  - **Selected Files** and **Save Files To:** Select **ri!record.image**.
  - Select the **Required** toggle



### Tip: Use the Show When Setting to Conditionally Show Interface Components

- In some cases, you might want to display a component based on a condition.
- To do this, use the **Show When** setting in the **Component Configuration** pane. Select **Only Show When** and enter an expression in the Expression Editor.
- For example, imagine that you want to display a Comments field if a vehicle's mileage is above 150,000 miles. You can enter the expression `tointeger(ri!record[recordType!W#SA Vehicle.fields.mileage]) > 150000`
- This expression determines whether the Comments component is displayed on the interface. When set to false, the component is hidden and not evaluated. The default is set to true.

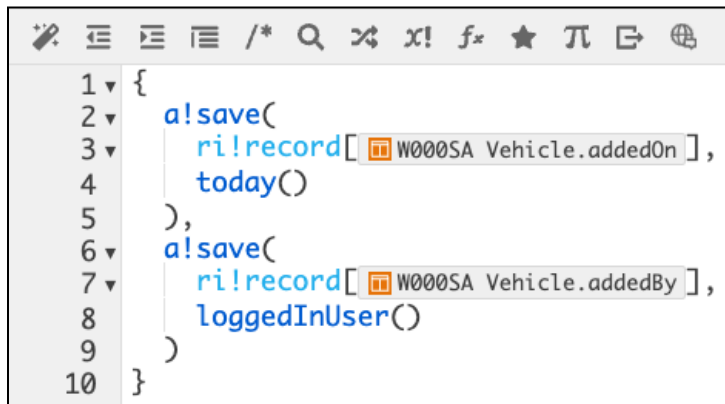
## Configure the Create Button

Next you will configure the Create button to save a vehicle's Added On value. You will also configure the Create button to save a vehicle's Added By value.

1. Click the **CREATE** button. In the **Component Configuration** pane, configure the following properties:
  - **Label:** Change the label to `Add Vehicle`.

- **Save Value To:** Click the **Edit as Expression** icon, and replace the existing expression with the following. As you are typing, use the suggested fields to ensure proper formatting.

```
{
  a!save(ri!record[recordType!W#SA Vehicle.fields.addedOn],
  today()),
  a!save(ri!record[recordType!W#SA Vehicle.fields.addedBy],
  loggedInUser())
}
```



In this expression, `ri!record[W#SA Vehicle.addedOn]` is the target and `today()` is the value being saved into the target. Similarly, `ri!record[W#SA Vehicle.addedBy]` is the target and `loggedInUser()` is the value being saved into the target.

2. Click **OK**, then **SAVE**.

## Create the Supervisor Approval Form

In this section, you will create an interface for the supervisor to review the new vehicles. After the registrar submits a new vehicle, the supervisor will receive a task to approve or reject it. Your interface will look like the image below:

Review Vehicle

Make

Ford

Model

F-150

VIN

2F2DE48C8N4309374

Year

2021

Mileage

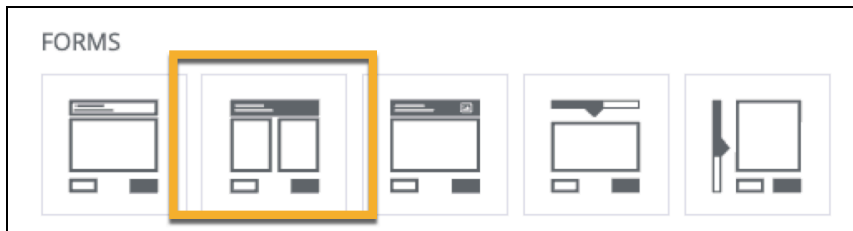
500

REJECT

APPROVE

Follow the steps below to create the supervisor approval form and configure the rule inputs.

1. In the **Build** view of your application, click **New > Interface**.
2. In the **Create Interface** dialog, configure the following properties:
  - **Name:** Enter `W#SA_SupervisorForm`.
  - **Description:** Enter `Displays read-only details about a selected vehicle for Supervisor approval.`
  - **Save In:** Select **W#SA Interfaces**.
3. Click **CREATE**.
4. Choose the **Form with Full Header** template:



5. In the **Rule Inputs** pane, click the **New Rule Input** icon. Create a rule input with the following properties:
  - **Name:** Enter `vehicle`.
  - **Description:** Enter a simple description.
  - **Type:** Select the **W#SA Vehicle** record type.
6. Click the **cancel** rule input, and configure the following properties:
  - **Name:** Change the name to `approvalDecision`.
  - **Description:** Enter a simple description.
  - **Type:** Keep **Boolean** as the type.

7. Click **SAVE**.

Next, update the layout and configure the components.

1. Rename the form **Review Vehicle**.
2. Delete the **Secondary Text**.
3. Delete the top **Section Layout**. To delete it, click this section, then click the dropdown arrow > **Delete**.
4. For the bottom section layout, delete the blue **Section** label.
5. Drag and drop **three TEXT** and **two INTEGER input components** into the first column.
6. For the text fields, change the labels to **Make, Model, VIN**. For the integer fields, change the labels to **Year** and **Mileage**. For each of these fields, make the following changes in the **Components Configuration** pane:
  - **Label Position**: Select **Adjacent**.
  - **Display Value**: Select the correct values. For example, for Make, select **ri!record.make**.
  - Select the **Read-only** toggle.
7. Drag and drop an **IMAGE** component into the second column. Delete the label **Image**, and click **+** in the Image field. Select **DOCUMENT IMAGE**.
8. In the **Component Configuration** pane, under **Document**, click **a!EXAMPLE\_DOCUMENT\_IMAGE()**. In the Expression Editor, delete all text, and enter the same expression you used in the summary interface to conditionally display a vehicle image:

```
if(
  a!isEmpty(
    ri!vehicle[recordType!W#SA_Vehicle.fields.image]
  ),
  a!EXAMPLE_DOCUMENT_IMAGE(),
  ri!vehicle[recordType!W#SA_Vehicle.fields.image]
)
```

9. Add test values to the interface.
  - Click **TEST**, and enter the following expression into the vehicle row:  
`rule!W#SA_QR_GetVehicleByID(vehicleId: 1)`
  - Click **SET AS DEFAULTS AND TEST**. The interface will display the sample data.

10. Click the **CANCEL** button, and rename it `Reject`.
11. In the **Component Configuration** pane, under **Value**, click `true` and change it to `false`. Under **Save Value To**, keep `ri!approvalDecision`.
12. Click the **SUBMIT** button, and rename it `Approve`.
13. Configure the Approve button to save values for the following four fields: Modified By, Modified On, Status, and Approval Decision.
  - In the **Component Configuration** pane, for **Save Value To**, click the **Edit as Expression** icon.
  - Enter the following expression:

```
{
    a!save(ri!vehicle[recordType!W#SA
Vehicle.fields.modifiedBy], loggedInUser()),
    a!save(ri!vehicle[recordType!W#SA
Vehicle.fields.modifiedOn], today()),
    a!save(ri!vehicle[recordType!W#SA
Vehicle.fields.statusId], 1),
    a!save(ri!approvalDecision, true())
}
```

- This expression saves:
  - The logged in user as the value for the Modified By field,
  - The current date as the value for the Modified On field,
  - The active vehicle status (which has a value of 1 in the `W#AA_VEHICLE_STATUS` lookup table) as the value for the Status Id field, and `true()` as the value for the Approval Decision rule input.

14. Click **SAVE**.

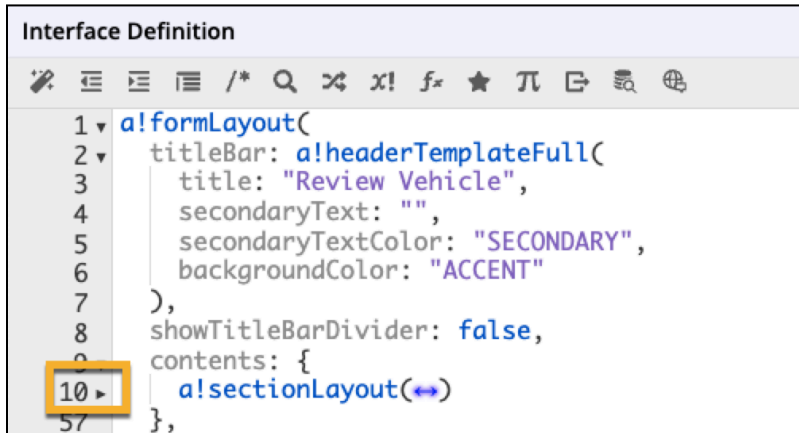
## Create a Reusable Interface

Keep in mind that interfaces can be built as reusable blocks. You can create an interface and then call it from other interfaces in your application. Try it out by creating a reusable interface from the Supervisor Review form.

1. Click **Expression Icon** to switch to Expression Mode.



2. In **line 10**, use the **arrow** to collapse the section layout:

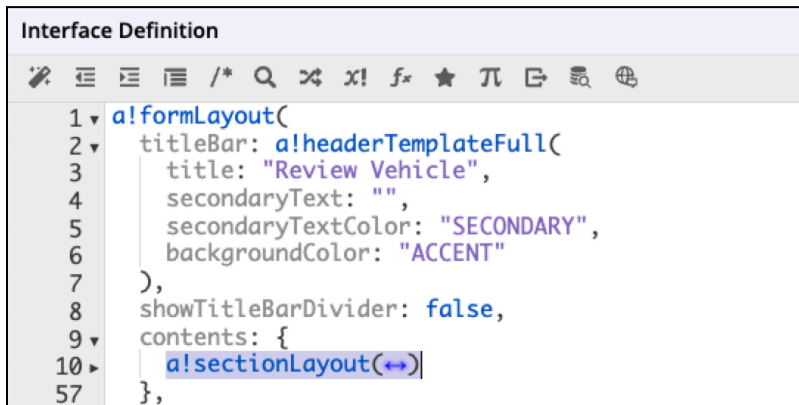


The screenshot shows the 'Interface Definition' editor with a toolbar at the top. The code is as follows:

```
1 a!formLayout(  
2   titleBar: a!headerTemplateFull(  
3     title: "Review Vehicle",  
4     secondaryText: "",  
5     secondaryTextColor: "SECONDARY",  
6     backgroundColor: "ACCENT"  
7   ),  
8   showTitleBarDivider: false,  
9   contents: {  
10    a!sectionLayout(↔)  
57  },
```

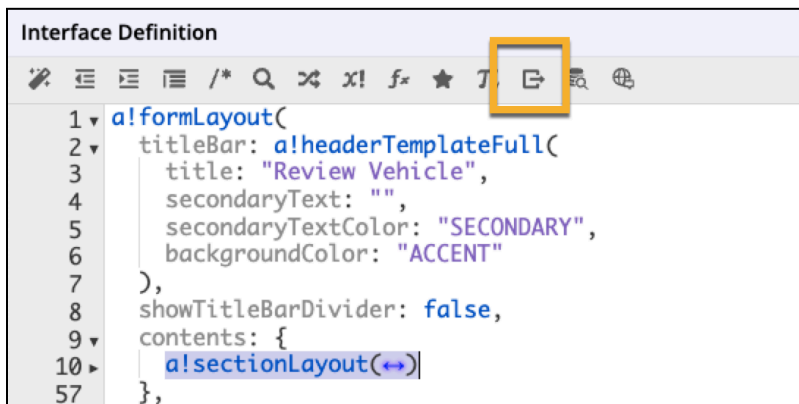
Line 10 is highlighted with a yellow box, and a yellow arrow points to the collapse button (↔) in the expression `a!sectionLayout(↔)`.

3. Highlight the section layout:



The screenshot shows the 'Interface Definition' editor with the same code as before. The expression `a!sectionLayout(↔)` on line 10 is now highlighted with a blue selection box.

4. Click **Save Selected Expression As** in the toolbar:



The screenshot shows the 'Interface Definition' editor with the same code. The 'Save Selected Expression As' button (represented by a document icon with a plus sign) in the toolbar is highlighted with a yellow box.

5. In the **Save Expression As** dialog, configure the following properties:

- **Save As:** Select **Interface**.
- **Name:** Enter `W#SA_VehicleDetailsView`.



- **Description:** Enter Read only interface to display vehicle details.
  - Click **SAVE**. The new interface will open.
6. In W#SA\_VehicleDetailsView, in the top right corner, click the **Gear** icon > **Properties**.
  7. Configure the following:
    - **Name:** W#SA\_VehicleDetailsView.
    - **Description:** Enter Reusable interface containing a read only view of vehicle details.
    - **Save In:** Leave as-is
    - **Include in the design library:** True
    - **Display Name:** Enter Vehicle Details View
  8. Click **OK**, then **SAVE**. Close the W#SA\_VehiclesDetailsView tab.
  9. Go to the W#SA\_SupervisorForm. You will notice the reusable interface has been added in place of the section layout and mapped to the appropriate rule input. Later, you will be able to reference this reusable interface anytime you want to display read-only vehicle details.
  10. Click **SAVE**.



#### **Tip: Build Reusable Interfaces for Reporting**

Creating reusable interfaces is useful for reporting interfaces. You can create each chart or grid as its own interface, and then combine them into different larger reporting interfaces for your business users.

## **Troubleshooting Resources**

Stuck on a step, or need help troubleshooting? Appian provides several support resources that you can use as you build:

1. **Acme Auto Solution Application** - The Acme Auto Solution Application (AS) is the solution to the exercises you are following in the Step-by-Steps. You can use the AS application as a reference tool. Review it to see how specific objects are configured, or test the application to see how the features work from a business user's perspective. This application is preloaded into your workspace. If you do not see it in the list of applications in your workspace, you can deploy it from the App Catalog. Refer to **Build an Application: Step-by-Step #1** for more information on how to use the App Catalog.

2. [Community Discussions for New Users](#) - Check out the **New to Appian** thread in Community. Join our community of experts to ask questions and find answers from past discussions.
3. [Appian Documentation](#) - Appian's product documentation will provide you with an overview of key Appian features, newest release information, additional tutorials, and helpful patterns and recipes to implement in your app.