Create apps by using Microsoft Power Apps (25–30%)

Create model-driven apps

* Create and configure multiple form types
* Use controls in the form designer
* Create and configure views
* Configure custom pages
* Configure modern commanding by using Power Fx
* Embed a canvas app in a model-driven app form
* Add Microsoft Power BI dashboards and reports in a model-driven app

Describe canvas apps

* Describe canvas app structure
* Describe form navigation, formulas, variables and collections, and error handling
* Describe how Microsoft Power Automate flows are called from a canvas app

Build Microsoft Power Pages

* Configure pages, forms, and navigation
* Describe advanced Power Pages features, including document management, search, multi-step forms, and Power BI
* Configure website security including web roles and page access
* Describe use cases for templates
* Describe authentication options

Canvas Apps vs Model-driven Apps

Application

Description automatically generated with low confidence

Canvas App

Three ways to start creating a canvas app. 1) Blank canvas 2) Start from dataset 3) Use preconfigured template example. We can built canvas app with or without Dataverse.

Components – for reusable purpose. We can create our own and put them into component libraries or use PCF components created by other developers. PCF components need to be imported as solution.

Model-driven Apps

Form Types – different types of forms are available for model-driven apps:

* 1. Main – main UI for viewing and interacting with table data
  2. Quick Create – basic form optimized for creating new records
  3. Quick View – appear within main form (e.g. display lookup column data from another table)
  4. Card – present data in compact format suitable for mobile UI

Form settings – can define security roles for the form controlling things like access, form order, and fallback forms. Additionally, Form settings provide us with the ability to check a particular security group's access to the form using “Form access checker” menu option.

Form elements – can control via form properties, business rule or scripting. If you use scripts, make sure that any element that might be hidden is hidden by default. Only show an element with scripts when your logic calls for it. That way it doesn't appear in presentations that don't support scripts.

Form event handlers - consists of a reference to a JavaScript web resource, and a function defined within that web resource that executes when the event occurs. Each element can have up to 50 separate event handlers configured. Add event handlers from the Events tab in the right side.

Form order - You can set the form order when using multiple forms. Form order defines the order in which a user sees the available forms, within the set of allowed forms for their security roles.

Access to forms - When users need to access common tables for different roles, it might be useful to have other forms available to tailor the user experience for that particular set of users. You can assign a security role (or collection of security roles) to control access to the form.

Grid controls - You can configure Grid controls for a Table or a Form as a read-only **subgrid**, an editable **grid**, or as a **Power Apps grid control (preview)**.A data browsing experience via infinite scrolling.

Display controls - Display controls provide you with the ability to display information in your app that isn't directly related to table data. Examples include the Calendar control, which allows users to view scheduled activities and their associated details in a calendar. Another display control is an embedded Canvas app, which enables rich data integration of various data sources right next to contextual data from the host model-driven form.

External website (iframe) - You can add iframes to the form by inserting an **External website** display control onto the form and inputting a URL. You can configure how it renders.

Timeline control - use the timeline control to capture activities like notes, appointments, emails, phone calls, and tasks. It ensures tracking and visibility of all interactions with the related table over time.

Create/Edit views - When you create a new custom table, you have a combination of public (active and inactive views of your table) and system views (including the four system views). Additionally, users can create custom public views. Users can also delete any custom public views they've created in an un-managed solution. However, any system-defined public views can't be deleted.

System views - System Views for your table aren't immediately shown in the view selector, and you can't use them in sublists in a form or as a list in a dashboard. You can't delete or deactivate these views. If you're a system administrator, you can edit these views just like you would a Public view. Just follow the steps for editing a public view.

Dashboard – two types, user dashboard and system dashboard. User can override the system dashboard to suit individual needs. Power App user can use interactive dashboard (Stream View) namely multi-stream and single stream.

Multi-stream - display data in real-time over multiple streams. There's no limit on how many streams you can configure on the dashboard. The data in a stream can be based on a single table, but each stream can be based on a different table. Include visual filters with the ability to toggle “Visual Filter” options.

Single-stream - visual filters already applied. They enable a more detailed view of a given table, and as such are good for monitoring smaller but more complex data. This dashboard type always displays tiles and places them on the right side of the screen.

# Use specialized components in a model-driven form

Apart from the regular controls, such as a tab, a section, and input controls, model-driven forms include specialized controls that help you manage relational data and cater to typical business requirements.

Business process flows - is a component of model-driven apps. These flows can guide users in implementing sequential work. A business process flow can streamline the organization's processes.

In a business process flow, you define a set of stages and steps. Each stage contains a group of steps. Each step has a field where you can enter data.

You can apply business process flows on form fields. **Business rules** or form scripts are automatically applied to the fields that are used in the business process flow. Workflows that are initiated by changes to fields in a business process flow will run when the data in the form is saved.

Creating a business process flow - You must select a base table to create a business process flow. The default table for all stages is the base table. However, from the second stage onward, you can change the table.

Tips: Business process flows can have up to 30 stages. Support both linear flow and branching (AND/OR).

Security features - Different users in an organization might follow different processes to run the same operation. That is, different conditions might be applied for different users to perform the same function. Because of this, you can have up to 10 active business process flows for each table (entity) to provide appropriate processes for different users and situations. You can associate business process flows with security roles so that only people who have a certain security role can view or use the related BPF.

Custom Connectors – a custom connector is a wrapper around a REST API that allows **Logic Apps**, **Power Automate**, or **Power Apps** to communicate with that REST or SOAP API.

Step 1 – Build your API (can be public or private, Logic Apps support SOAP)

Step 2 – Secure your API, i.e., use one of these standard authentication methods for your APIs and connectors.

Step 3 – Describe the API and define the custom connector. The stage defines request and response in the connector. You can use [OpenAPI definition](https://learn.microsoft.com/en-us/connectors/custom-connectors/define-openapi-definition) or [Postman collections](https://learn.microsoft.com/en-us/connectors/custom-connectors/define-postman-collection) to define your connector’s endpoints. It also allows to [create from scratch](https://learn.microsoft.com/en-us/connectors/custom-connectors/define-blank), i.e., entering request/response manually.

Step 4 – once a custom connector is created, we can start using in our power platform.

If you update (remove, add, or change) a field in the API, perform these steps:

1. Republish the connector so it looks at the updated Swagger for the API.
2. Remove any connection / data source in any app that used that connector.
3. Re-add the connection / data source for that connector back into the apps.