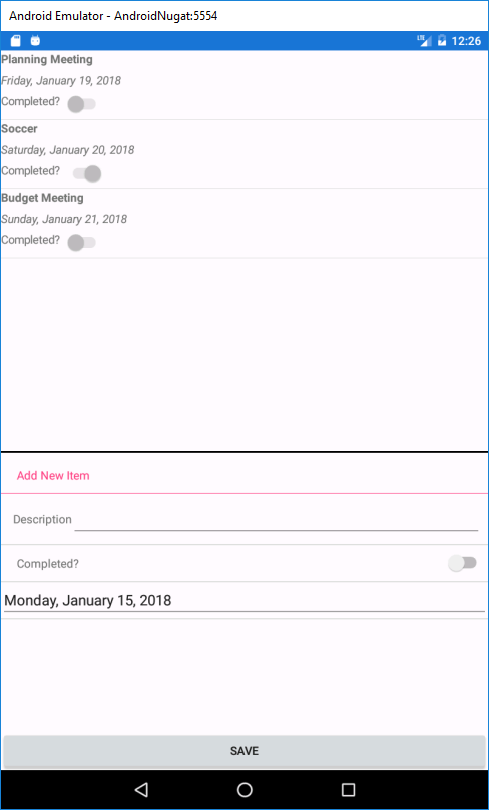
# Exercise: Table View and List View

In this exercise you will create a page that will contain a List View that shows a list of items and underneath that a Table View that allows editing or adding of new items to that list.

An example of what the app interface could like:



## ToDo Class

Create a class to store information on To Do list items. The class will store a description, a date, and a completed flag (Boolean). The class should also implement the INotifyPropertyChanged interface as was done in the example for the Fruit class.

To begin with you may want to hard code a few To Do items into a collection for your app so that you can work on it in stages.

## List View and View Cell

In your app class, define a List View that will show the description, date, and completed properties in a View Cell. Extend the View Cell class and define a cell that will contain three Labels and a Switch. A switch is a Xamarin Forms view that can be used to display Boolean information.

When using the Set Binding call, you can use the syntax below to set optional properties on the binding, the most common being the string format:

ControlName.SetBinding(ControlType.PropertyType, “BoundPropertyName”, stringFormat: “format”)

* String format codes for this take the form “{0:XYZ}” where XYZ is a format code.
* List of codes: <https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-date-and-time-format-strings>

Setting the string format is useful when showing dates or numbers and you want to control how they will be shown in a label, for instance.

When adding your list view to your main layout you may want to set the Height Request to something as a list view by default will not vertically expand as new items are added, so if there are very few items the list view will take up very little space and may look odd on the page.

Run your app with the hard coded items to see if your list view works before proceeding.

## Table View

For table view remember that there are a number of built in cell types that can be used. Use an Entry Cell for the description and a Switch Cell for the switch. For the date, you can wrap a date picker in a View Cell without having to completely extend your own View Cell. The ***View*** property of the View Cell can be assigned any View, including a Date Picker.

Add your button outside the table at the end, though if you want you can add a View Cell for it as well.

## Event Handlers

You’ll need an event handler for the button and the list view. You may want to get each working separately.

The list view tap should populate the table view with the relevant information from the selected item.

The button really has two behaviors, if nothing has been tapped it will save a new To Do item to the list. If a list item was previously tapped, it should update the selected list view item. It might be helpful to declare a private variable that will store the currently selected To Do item, though there are other approaches.