Technical task for iOS Developer

Requirements: Obj-C or Swift, iPhone, iOS 9+. You can use any 3rd party libraries if you need.

For example, you have a JSON:

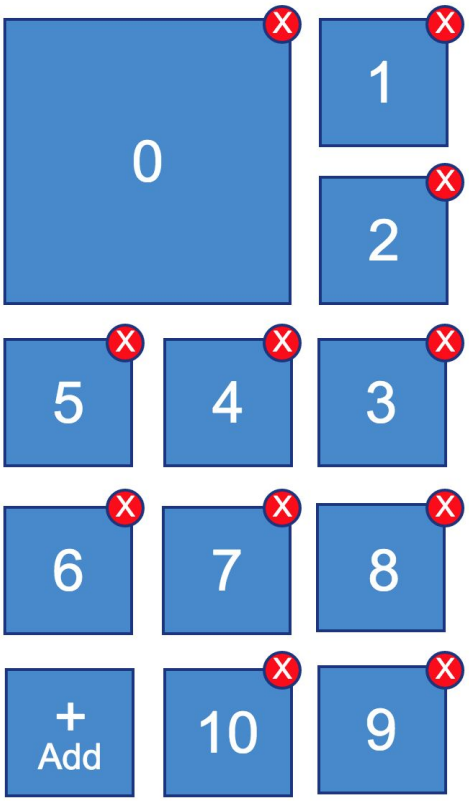

```
{
  "items": [
        {
        "uuid": "bef89c66-2318-46bb-b142-41f88ddf532f",
        "imageUrlString":
"https://vignette1.wikia.nocookie.net/disney/images/f/f6/MickeyArt.jpg/revision/latest/scale-to-width-down/250?cb=20130705054827"
        },
        {
        "uuid": "2813a61c-3d5e-409b-bece-7c67b60bb5a0",
        "imageUrlString":
"http://vignette2.wikia.nocookie.net/deathbattlefanon/images/2/2b/Bugs_Bunny.png/revision/latest?cb=20151206010607"
        },
        {
        "uuid": "9d3e9bff-1833-4b49-82cc-7847a935d4ec",
        "imageUrlString":
"http://vignette2.wikia.nocookie.net/tomandjerry/images/6/65/20140211071529%21Tom_Tom_and_Jerry.png/revision/latest?cb=20140529170742"
        },
        {
        "uuid": "588e2c7f-40de-4c4d-ac74-2884daf5ecc8",
        "imageUrlString":
"https://vignette2.wikia.nocookie.net/simpsons/images/1/11/Homersimpson.jpg/revision/latest?cb=20121229201104"
        },
        …
  ]
}
```


This JSON represents a response from a server. You will not use a real API in this task, just use this JSON as a stub. Instead of calling the API, just save the updated JSON data. You can use any saving mechanism you prefer (file, NSUserDefaults, NSCoding, etc).

Please consider that the real networking may come to the project later (This is not true for sure, but please pay attention on this when you will design your app architecture). All update operations should be performed asynchronously and return the success code. If the success code is false do not update the UI or cancel the operation (if you have already updated).

The JSON contains an array of items which you have to display on the screen. Each item has an identifier (**uuid**) and a link to the picture (**imageUrlString**). You can use your own links if you don't like our images :)

After parsing you should display these items on the screen like this:



This is an UICollectionView. The numbers in the cells explain the order of the items visible in the UICollectionView (You don't need to place these numbers into the cells!!!). Each item cell should instead display an **image** (downloaded from **imageUrlString**) and should have a **button to remove the item.** The zero cell is always bigger than the rest of the cells.

The last cell in the collection view does not represent an item - instead it has the button to add a new cell. After pressing the button the user should see a system alert with a text field where he can insert a new link - and create a new item by selecting "OK" or cancel the process by selecting "Cancel". You must generate the identifier by yourself. Next update the JSON and add a new cell to the collection view.

For example: You have 11 items and add a new item. The new item will have the **index 11** and the new cell will be added instead of "**Add**" cell. "**Add**" will move underneath the new cell, to the **12th** position. All further cells should respect the same layout rule of the shown diagram. If items take more than the screen height, the collection should become vertically scrollable.

Tapping the **delete** button should erase the item in the number sequence, deleting in both the collection view as well as the JSON. Before the action, the confirmation alert should be shown to user. The text should read: *"Do you really want to delete the item with identifier **UUID**?"* (Replace the word "UUID" with the real item identifier). If the user selects "Yes" then delete the item, if he selects "Cancel" - nothing happens, just close the alert.

For example: When clicking the "Delete" button on cell 7 in the number sequence, all of the following cells (8, 9, 10 etc.) should change according to cell 7's deletion, leaving no empty cells.

After **selecting a cell** by tapping on it, the user should see an alert with the text *"You have selected the item with identifier **UUID**"* Again, replace the "UUID" with the real identifier of the

item.

The user should be able to **reorder the items** with drag and drop, each reorder operation should affect the JSON as well. Please pay attention that the zero index cell should always be bigger than the others.

All collection modifications should perform in an animated way (performBatchUpdates will help you).

Please send us the complete project as a zip archive. If you choose to use third-party libraries - please mention them and explain why you used them. If a **.xcworkspace** file should be opened instead of a **.xcodeproj** file please mention this as well. Focus on performance, stability and a good application architecture. Good luck!