Implement Tab Bar Arrangement

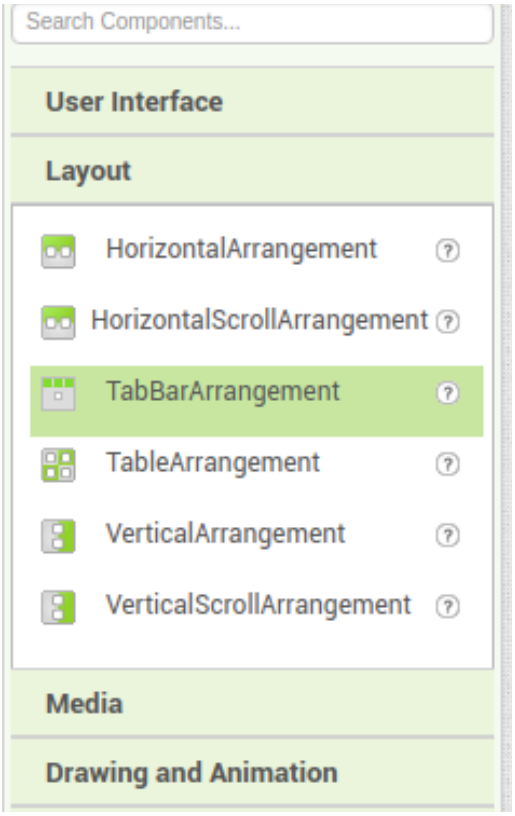
The purpose of this document is to ask for opinions from the users as well as the developer community of App Inventor on my GSoC project. Kindly feel free to drop a comment. Any suggestions/improvements are most welcome.

Overview

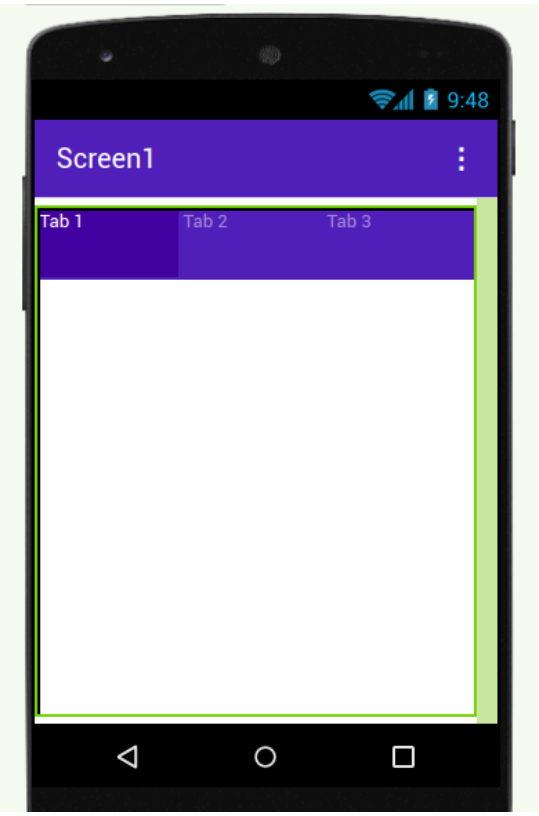
The aim of the project is to add a new component "**Tab Bar Arrangement**" to the existing App Inventor.

I plan to achieve this as follows:

- The new component will be added under the **layout category** of the design palette. (as shown in *Figure. 1*)



(Figure. 1)



(Figure. 2)

- As the user drags the component to the screen, a tab bar layout will be displayed on the mobile screen with height **FILL_PARENT** and width **MATCH_PARENT**. The expected outcome is shown in *Figure. 2.*

- Initially, there will be 2 tabs in the arrangement. However, the user can modify the number of tabs from the property.
- The component will have 3 properties:
    - **Number of tabs**: An input field which will take only integers (greater than or equal to 2) as input.
    - **Tab Label from String**: An input field in which the user can input a string containing Tab labels separated with a comma(,).
    - **Selected Tab Index**: An input field which will take only integers ranging from 1 to the number of tabs. Users can vary this number to select a particular tab.
- For implementing this, I'll add 2 Mock classes.
    - **MockTabBarLayout**
        - This class will extend the abstract class MockLayout.
        - It'll be responsible for displaying the layout similar to the tab layout on the screen in the designer window and keeping the record of children available under each tab.
        - Here, each tab can have only a single child. The vision behind this point is that the user will add the desired arrangement (Horizontal or Vertical Arrangement) inside each tab to place multiple children under a tab.
          We could possibly extend one of the arrangements to avoid this, but we shouldn't/can't restrict our users on having their desired orientation under each tab.
        - For keeping the record of the children, I'll add a property "**TabIndex**" for each component, which will hold the tab index of the respective component. This property will neither be visible in the designer window nor be available in the block editor.
        - If multiple children claim to be in the same tab, then only the last added child will be set visible.
    - **MockTabBarArrangement**
        - It will extend the abstract class MockContainer.
        - It will use the MockTabBarLayout class to set up the Tab Bar arrangement on the screen in the designer window.

- Similar to Mock classes, there will be 2 component classes:
    - **TabBarArrangement**:
        - It will extend the AndroidViewComponent and will implement the Component and ComponentContainer interfaces.
        - It will use the class TabBarLayout (discussed below) to set up a required arrangement in the application running on our Android devices.
        - It will contain the getter and setter methods of all the properties mentioned above.
    - **TabBarLayout**:
        - This class implements the Layout interface and is responsible for setting up the tab layout.
        - For this purpose, I'll use TabLayout and ViewPager.

Discussion

- For using ViewPager with TabLayout, I need to have an Adapter class extending **android.support.v4.app.FragmentPagerAdapter** which would display the **android.support.v4.app.Fragment** in the view pager. FragmentPagerAdapter requires **android.support.v4.app.FragmentManager** to display the Fragments whereas the FragmentManager given by the Activity class in App Inventor is **android.app.FragmentManager**.

  To solve this, either I can use the Fragment Pager Adapter from another support library: **android.support.v13.app.FragmentPagerAdapter** or I could rewrite the FragmentPagerAdapter class by copying the code from **android.support.v4.app.FragmentPagerAdapter**, and then changing the imported Fragment class to **android.app.Fragment**.

  In addition to this, I need to have another class **Tab** extending the **Fragment** class where I'll inflate the view of the respective component in the Fragment. (As mentioned above, a tab can contain only a single child, the view of which will be inflated in the Fragment.)
- Another workaround for this could be to add the TabBarLayout and a VerticalArrangement in the same container and then display the component of the selected tab in the vertical arrangement. (The only child of the selected tab

will be set visible, rest all of the children components will have their visibility GONE.)

I'm open to taking up any of the above-mentioned methods. Kindly highlight the pros and cons of each method. If you've any other workaround, please feel free to mention it here. Any suggestions would be much appreciated. Thanks in advance. :)

If you've any other queries/opinion/suggestions, kindly drop a comment here.