There are many different platforms for developing mobile apps that require either limited or no coding. This is done by using pre-developed objects called components, coupled with some block programming.

The Platform we will be using is called **Thunkable**. This service has a free (public) development section and a paid (private) app development section. For this camp we will be utilizing the free or public app development one. If you have an Android or iOS mobile device, you can download a live app viewer from the respective app store. If you do not have a mobile device do not worry as you can live preview your app directly in the web browser for most functions.

We will be developing a mobile app to work as a quiz based around the camp’s GenCyber key concepts. The app will consist of four screens with each screen allowing for input or selection of answers in a different format than the others.

This app demonstrates both **Availability** and the concept of **Keep It Simple**.

**Section: Account Creation / Access:**

If you have not created an account yet please go to: <https://x.thunkable.com/signup> , if you have created an account you can skip this section.

You will need to enter your email address into the box with the placeholder text “Enter your email” and then click the button “Email me the link”. Check your email and you should receive a link to click which will log you into the site on your device. If you do not click the logout button afterwards it should keep you logged in for 30 days and you can just revisit the site to continue working.

**Section: Account Access:**

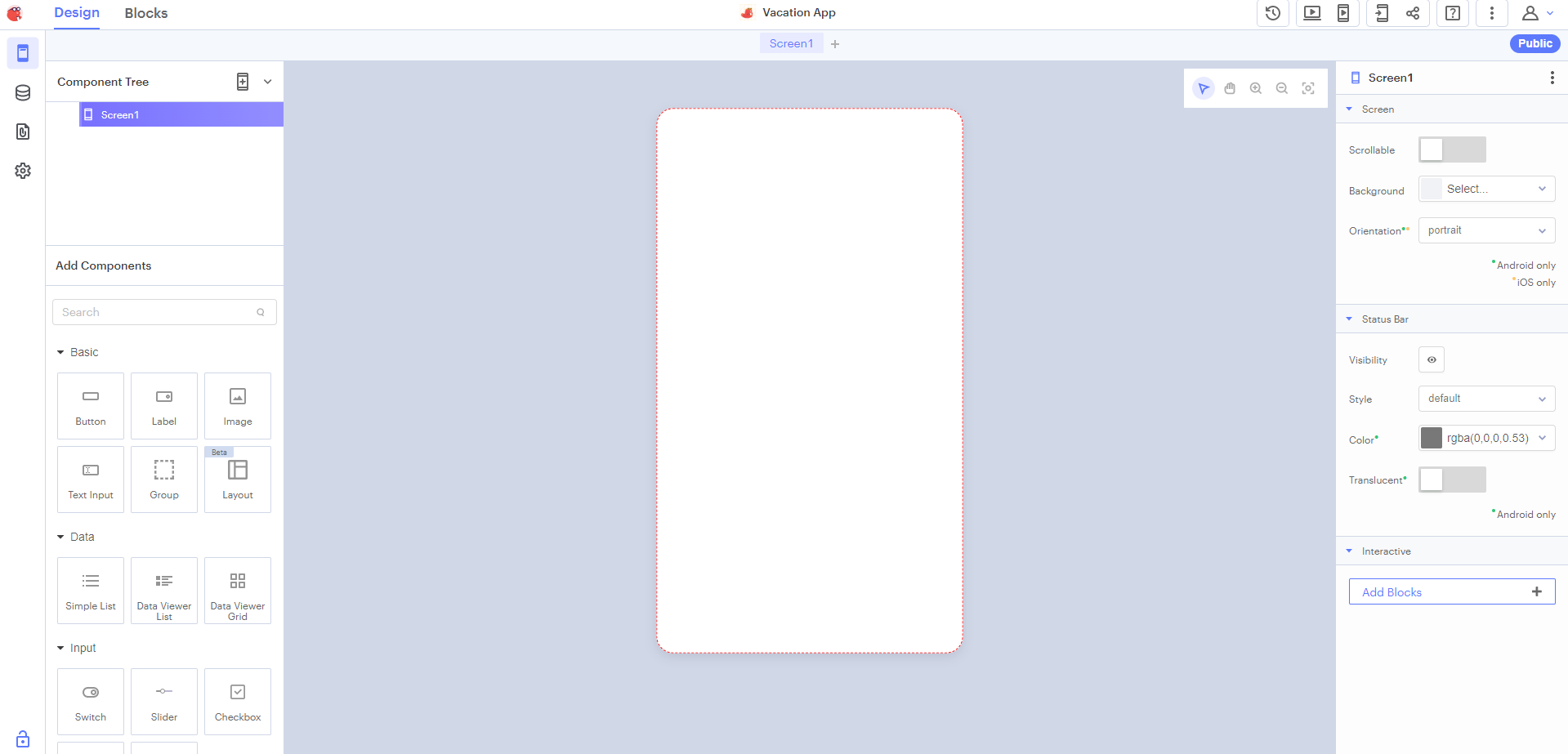
If when you visit the site again you find yourself logged out, you will need to go to <https://x.thunkable.com/login> and once again provide your email address and click “Email me the link”. Then log into your email and click the link to be automatically logged into the site.

**Section: Getting Started:**

|  |  |  |
| --- | --- | --- |
| 1. Click the Create New App Button     **OR**  **A blue rectangle with white text  Description automatically generated** | 1. Fill in the form identical to this:   A screenshot of a project  Description automatically generated | |
| Then Click the “Create” button | | Hide the tutorials section by clicking the purple tab with the left arrow on it |

**Section: The Design Process:**

1. At this point, your screen should look like the one below:



On the left, at the top, you will see 2 tabs, Design and Blocks. To begin we will be working on the design of the app so make sure that tab is selected. Below that is a section for components that are currently part of your app and further below that are components you can drag onto your app. On the right will be the individual properties of whatever the current selected component is. To begin it is showing the properties of the Screen1 component.

In the middle of the window at the top, above the screen display, you will see the current screen’s title highlighted which initially should be Screen1 along with a plus sign to the right. Press the plus sign 3 times so you have Screen1, Screen2, Screen3, and Screen4. Rename these screens in the Component Tree on the left of the window. When you hover over one of the screen names, two additional icons will show up being the Trash (delete) and the Pencil (edit) icons. Click the pencil for Screen1 and rename it “**Introduction**”, Screen2 should be named “**Fill in The Blank**”, Screen3 should be named “**Choose Correct**”, and Screen4 should be named “**Pick One**”.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Should Become:

**Screen 1 - Introduction**

A screenshot of a computer

Description automatically generatedA light bulb on a table

Description automatically generatedThe introduction screen when completed will look like the image on the right. To build this screen, you will be dragging a Button, Label, and an Image component, found under basic components, to the screens workspace. The components should be located to the left of the window below the Component Tree. Name these items as follows: Button: “btnStart”, Label: “lblTitle”, Image: “imgMain”.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedPlace the components in a similar spot on the screen as you see in the finished version. Once done, select the screen itself and set the background to black.

Next, Select the label and then set the text to “**GenCyber** **Quiz**”, font to “**Abril Fatface**” and font size to 36.

Then select the *paragraph alignment* button (top row of buttons on right) and select the “**center text**” option. Verify that your settings look like the image to the right.

Next you will be setting the image size and source. With the image object selected, click on source, and then upload image. You will be using the QUIZ.png file provided with this module. Use the handles of the image to shape it to fit the space as seen in the final image.

Finally, the button, set the text to “START QUIZ”, with font of “Lato”, size of 28, centered, with a background color of (hex) 274408.

**Code Blocks – Click the Blocks tab at the top of the window.**

The code blocks for this screen are relatively simplistic as they will only be needed to start the quiz itself which initially just navigates to another page. The first block you need can be found by selecting the **btnStart** object in the UI Components section on the left. The block you need should be the first one and it looks like the image to the right.

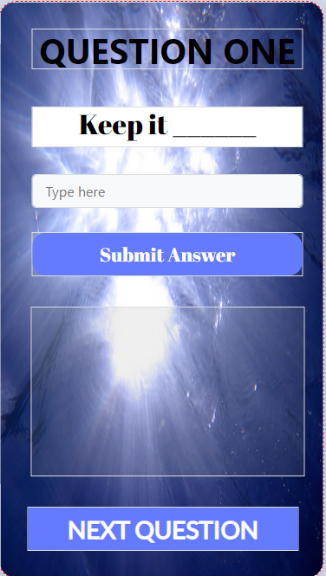
A yellow and white screen with black text

Description automatically generated with medium confidenceA yellow rectangle with black text

Description automatically generatedA yellow chat box with black text

Description automatically generatedThen we need the “**navigate to**” block to place at the “**do**” section of the click event block we just placed in workspace. To locate this block, on the left scroll down and select “**Control**” under the **Core** components. Then in the dropdown, select “**Fill in the Blank**”. That’s it for this screen.

**Screen 2 - Fill in the Blank**

The first type of question / answer quiz question we will be building and checking is a fill in the blank styled question. The user guesses by typing their answer in a text input element and then tapping a button. If what the user enters matches the answer, it will let the user know they got it correct, otherwise a message stating that the answer is incorrect will be displayed.

A screenshot of a computer

Description automatically generatedTo build this screen, you will be dragging three labels to the screen, 2 buttons, a text input component, and an image component. Name them as shown in the image. Make sure they are in this order as well. You can click and hold the mouse button down on an item to drag it to a new position.

A white square with black text

Description automatically generatedSet the text of each component to match what is shown. For the Labels, the Title Label should a font size of 40 and centered with a font type of your choice. The Question label should use “**Abril Fatface**” for a font of size **32**, **centered** as well. The third label, Response\_1, should have an empty string as it will show no text upon load. The make sure the size fills up the rest of the area as shown in the image.

Leave the default settings on the text input. For the buttons, The Submit Answer button should have a font size of 22, font of “**Abril Fatface**” with a text color of **#FFFFFF** and a background color of **rgba**(93,122,252, 100). The Next page button at the bottom of the screen should use **Lato** for the font, size of 28, color of **#FFFFFF** and the same background color as the last button you modified. Then select the screen background and set the image to the underwater.jpg file that has been provided to you by clicking upload and selecting the correct file name.

**Code Blocks – Click the Blocks tab at the top of the window.**

A yellow box with black text

Description automatically generatedThe code blocks for this section will be more complicated than the first page. You will need another one of the navigation blocks you built for the last page but with a different screen selected in the drop down.

A yellow puzzle piece with white text

Description automatically generatedA yellow box with black text

Description automatically generatedTo handle checking the value entered in the text input by the user, the next block will be more complicated. Like the navigation block, you will need to drag a when/do click event to the screen and make sure it matches the one to the right. You can find it by clicking the Submit\_1 button on the list of components in the left panel. Inside this block, you will need to embed an if/do/else block which is also found under the Control section of the Core components. You can see them combined in the image to the right.

A group of text boxes

Description automatically generatedThis block has 3 different sections to fill. The “do” and the “else” section will both be similar so let’s build that first. Select the Response\_1 object on the left panel in UI Components. Here some blocks that are relevant to that component appears. Drag 2 of the Set Response\_1’s Text to “Label” blocks to the workspace and connect them to the do and else sections. Then replace “Label” on the first block to “That is Correct!” with the label on the else section set to “That is not correct”.

All that is left for this screen is to build the conditional statement that we will be attaching to the If spot of the if/do/else block placed earlier. Select logic and then drag the top block to the screen. It should be the “blank” = “blank” block. Under the Text category of Core components, drag the “To Uppercase” block to the screen and place it within the first blank spot of the equals component that was last placed. Switch Uppercase to lowercase by clicking it and selecting the new option.

Next click the Answer\_1 component on the left toolbar and drop the light green block that says “Answer\_1’s Text” and place it within the label spot on the to lowercase block. The last thing to do is drag an empty string block, found under the Text category at the top of those blocks to the blanks spot left on the equal’s block. Fill the in the string with the word “simple” to complete the “Keep it Simple” question. When complete it should look like this:

A puzzle pieces with text

Description automatically generated

Test out your work by clicking the Web Preview button in the upper right set of icons. If it doesn’t work, go back over this section, and see where you might have mixed things up. When complete move on to the next screen, “Choose Correct

**Screen 3 – Choose Correct**

A screen shot of a hand with a sign

Description automatically generatedThe second type of question / answer quiz question we will be building and checking is a multiple-choice styled question. The user guesses by selecting the checkboxes next to the corresponding labels, and then tapping a button to submit. Depending on how many of the answers were correct, a corresponding message will appear, ending with “All Correct” when all options are selected correctly.

Most of the layout will seem similar and should be uniform across each screen to give an easy-to-handle UI (User Interface). The main difference here will be the 3 new checkbox components which are found inside the Input components further down the toolbar.

A screenshot of a computer

Description automatically generatedName the individual components to match this list. Remember you can rename them by hovering over them and then clicking the pencil icon.

Make sure the text matches the text shown in the finished image. Try to get it as close as you can to this layout by adjusting the different settings.

The background image used in this screen is the cyber.jpg file provided.

**Code Blocks** **– Click the Blocks tab at the top of the window.**

A yellow box with white text and black text

Description automatically generatedLike the previous screens, you will need to create a navigation block which will bring you to the next screen. The final block should look like this.

To make this page work correctly, we need to initialize an **app** level variable named **correct**, to the value of **0**. Inder the **Variables** category, drag the first block onto the workspace. Replace name, with the name “**correct**”. You can place this at the top of all your blocks to signify that it is global to this page.



The last click event block is going to be a large one. When the user clicks the submit button, we need to make sure the “correct” variable is reset to 0 in case it has been ran more than once. Then check each of the checkboxes to make sure the ones that should be selected are, whereas the ones that shouldn’t be selected are not.

To begin, select the **Submit\_2** UI Component from the selection on the left and drag out a when / do click event as shown below.

A yellow rectangular object with black text

Description automatically generated

Next under Variables, drag a “**set app variable correct to**” block and place it at the top of the click event block. Then under Math grab the top block that is just a 0 and place it to the right of “to”. It should look like this image at this point.

A yellow box with black text

Description automatically generated

A yellow puzzle piece with a purple and white logo

Description automatically generatedTo check each of the checkboxes, you need to drag 3 of the if/do blocks, found under the **Control** section. Stack these 3 blocks under the “**set app variable correct to 0**” block inside of the click event block. Under logic, drag a “not” block to the middle if/do block and place it on the IF part. Then under each of the checkbox components you dragged and named earlier you will find a block such as the one to the right that has the named of the checkbox then the keyword value. Drag these to the if spots with the chkCon being dragged to the not. This signifies in code that the checkbox to the Confidence option should not be checked to be correct. The do section of each of these are identical.

Under Variables drag 3 of the “**change app variable correct by 1**” blocks and place each one in a DO spot. The code block should now look like this:

A screenshot of a computer program

Description automatically generated

The last chunk of code need for this page determines the output that will show to the user based upon the choices they made. We can determine this solely off the value stored in the variable: correct. To start this last block, under the Control section, drag an **if/do/else** block to the workspace and attach it to the bottom of the previous 3 **if/do** blocks. To fill out the IF block, under **Logic**, drag a **“blank” = “blank”** block to the workspace. On the right side, drag a number block from the **Math** section (it was the 0 block we dragged before) and replace **0** with a **3**. Then on the left side of the equal sign, drag the “**app variable correct**” block from **Variables** and affix it there. At this point, this code block should look like this:

A screen shot of a computer

Description automatically generated

For the next two spots, select Result\_2 from the UI components on the left and drag out two blocks of the “set Result\_2’s Text to ‘Label’” Blocks to the workspace and attach them accordingly. Set the Label part of the first block to “All Correct!”.

A close-up of text boxes

Description automatically generated

A puzzle pieces with text

Description automatically generatedNow we need to handle if the user gets a score of anything other than 3. To simplify this, drag a join block from under Text and replace the “Label” spot of the else block. Replace “hello” with “ “Number of Choices Correct:” and then replace “word” with another “app variable correct” block. The final click event block should look like below, if so, you can continue to the next screen. Otherwise double check the blocks and design to make sure everything is correct.

A screenshot of a puzzle

Description automatically generated

A screenshot of a phone

Description automatically generated**Screen 4 – Pick One**

The final Question / Answer type is the “Pick One” screen. This type of question has a single correct answer and allows a user to click one of 2+ buttons to get the question right or not. The example for this page is visually lacking to allow you to customize as you feel fit. The basics are shown in the final image for the screen.

This screen requires 3 labels and three buttons. Name them as shown in this image.

A screenshot of a computer

Description automatically generated

Make sure to remove the text from the Response\_3 Label. Otherwise, the text should match what you see in the final screen example.

**Code Blocks – Click the Blocks tab at the top of the window.**

The code blocks for this page are probably some of the least complicated blocks for quiz questions. Drag a when/do click event from each of the buttons to the screen. Then under **Response\_3**, drag 3 of the “Set **Response\_3’s** Text to ‘label’” blocks and place one in each of the DO sections of the when/do blocks. Change the label for **btn\_DID** to **“That is NOT correct”**, the **btn\_avail** label to **“That is CORRECT!”** and **btn\_int** to **“That is NOT Correct”** as well. That is it for This application!

A puzzle pieces with text

Description automatically generated

**Section: Challenge & Explore:**

1. Create a global variable to hold total score, modify the module so that there is no Next Question buttons at all. Instead after the user guesses their answers and submits it, the page should still show the results and then after 3 seconds, navigate to the next page.
   1. Then add a final screen to your mobile app that shows the user’s total score.
2. Using the free sound assets provided in this module.
   1. Add different sounds to the button clicks.
   2. Add the quiz\_music.mp3 file to a background music that should play once the user starts the quiz, and end on the final page.

Note: The play and stop blocks can be found under “App Features > Sound” category under the Core components.

**Section: Additional Information:**

Thunkable Docs: <https://docs.thunkable.com/get-started/>

Sounds files: <https://freesound.org>

Images used: <https://pixabay.com>