

Thunkable Tutorial Food Court App Tutorial

Authors:

- Chris Kerslake, chris@xmodus.com

Versions:

- 1.0 - 2020-02-15 - Chris Kerslake - original



This material is licensed under the Creative Commons (CC-BY). This means you are free to copy, remix, create derivatives and even use this material for commercial use so long as attribution is given.

Contents

Thunkable.....	4
What is Thunkable?.....	4
Quick Thunkable tour.....	4
App Development Process.....	5
Food Court App.....	6
Step: Create the Home Screen.....	6
Step: Add a version number label to the Home Screen.....	7
Step: Add a “Change Colour Button” to the Home Screen.....	9
Step: Make “Change Colour Button” change the Home Screen background colour with code.....	10
Step: Make the background change to a random colour when “Change Colour Button” is clicked.....	11
Adding the “I’m Hungry Screen”.....	13
Step: adding a second screen, “I’m Hungry”.....	13
Step: adding a Home Screen button to navigate to the “I’m Hungry” screen.....	14
Step: add a second screen, “I’m Hungry”.....	14
Step: add a “Home Screen” button to the “I’m Hungry Screen”.....	15
List the available restaurants on the “I’m Hungry Screen”.....	17
Step: add a list component to the I’m Hungry Screen.....	17
Step: add numbers to our restaurant list using code.....	17
Step: replace list numbers with list names/text (strings).....	19
Choose a random restaurant on the I’m Hungry screen.....	20
Step: add a “Random Restaurant” button to the I’m Hungry Screen.....	20
Step: connect the Random Restaurant button to the label when the button is pressed.....	21
Step: randomly choose a restaurant when the Random Restaurant button is pressed.....	22
Using a variable for our list.....	25
Step: creating a variable named “Restaurant List”.....	25
Step: replace all copies of your list with your new “Restaurant List” variable.....	25
Adding a local DB and then loading it into a list.....	27
Step: add a “Local DB” (DB = database) to our project.....	27
Step: name the database “Restaurants DB”.....	27
Step: add data to the database.....	28
Step: reset the Restaurant List to empty when the I’m Hungry Screen starts.....	29

Thunkable Tutorial Food Court App Tutorial

Step: load our "Restaurant List" variable from our new "Restaurant DB"	29
How do I make an image clickable (like a button)?.....	34

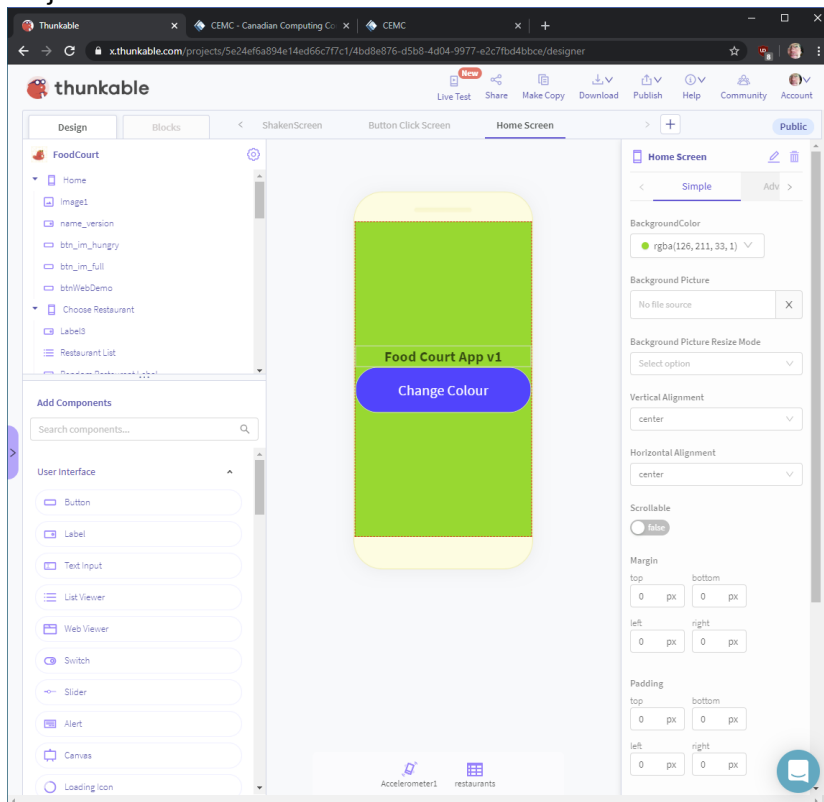
Thunkable

What is Thunkable?

1. Thunkable is a website that allows you to create applications that run on cell phones (called mobile devices).
2. Thunkable allows you to program your app by drawing screens and then adding functionality (reword) using blocks of code.

Quick Thunkable tour

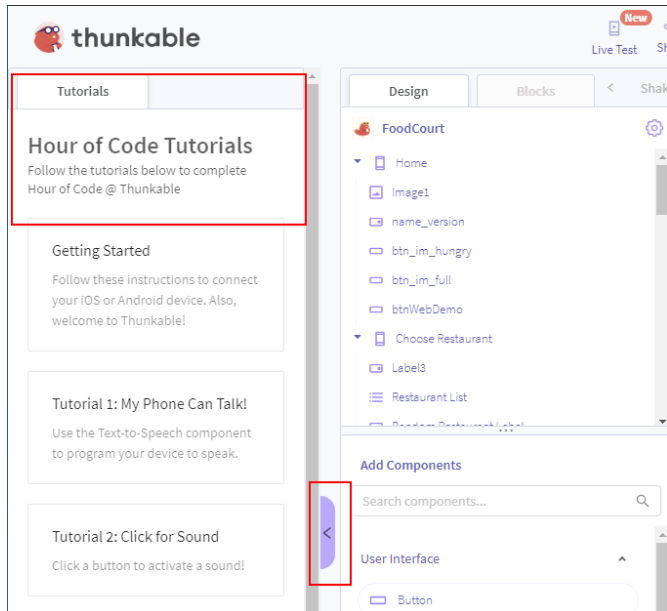
3. Login/account
4. Create New App / My projects / Top Community Programs / Public Gallery
5. Project Screen



- a. Design tab
 1. Project Components and Add Components on left-side
 2. Properties / Attributes on the right-side
 1. Name vs. Text
 2. Advanced vs Basic
- b. Blocks tab (per-block)
 1. w/ block groups on left-side
 2. trash can and zoom on the bottom right

Thunkable Tutorial Food Court App Tutorial

3. existing objects, like buttons, screens, etc. on the left-bottom
 4. mechanics – drag-drop & snap into place – beware of shape-shifting pieces
 5. Clicking vs Snapping
 6. collapsing, expanding and organizing blocks (for later in tutorial?)
- c. Screen list (top-middle)
6. Tutorials on the left (purple slider tab)



7. Live Test

- a. Must have the Thunkable app downloaded to your phone
- b. Login to your account on your phone
- c. Note on Apple iOS devices that Thunkable Live apps have a banner at the bottom.

According to Thunkable this is due to a requirement by Apple that apps like Thunkable cannot take up more than 80% of the screen.

“Apple has been a bit inconsistent on how they've been treating various live preview apps, including ours. Technically, they require that the "live" portion take up no more than 80% of the screen, so we have been doing that to stay compliant.” Source:

<https://community.thunkable.com/t/congrats-your-project-is-live-remove/384620/5> - as of Jan 13, 2020. This is for live-testing apps only and not for apps that you build and put into the Apple App Store.

App Development Process

8. The development lifecycle
 - a. Design (ideation & wireframes)
 - b. Build (code) * our focus today
 - c. Test * our focus today
 - d. Deploy – release to app store (not covered today)

Food Court App

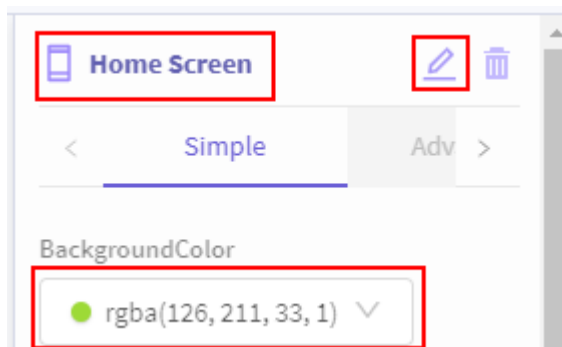
The Food Court app was built by me during my time taking evening classes at SFU Surrey. During our dinner break we would often go down to the food court in the mall and inevitably someone would ask, what are we going to eat tonight? I thought that instead of trying to come up with an answer each time that I could just shake my phone and it would make suggestions. I also thought that I could keep track of the number of times and experiences I had with each vendor so eventually the app could help me avoid bad places, reward good places and also help others feed into the decisions myself and my classmates were making each week.

Step: Create the Home Screen

Reason: TODO

Using the Home screen – reason: learn about screens and their properties

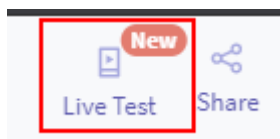
Rename the default starting page from Home to Home Screen via the properties section on the right side of the website page. Click the edit icon (the pencil with a line under it) to change the name.



While we are over in the screen properties, let's change the screen's background colour to your favourite colour! I'm going to choose a lime green.

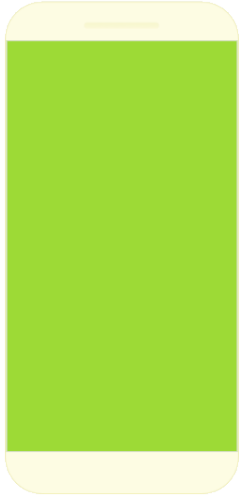
➤ Live Test app

Now that we have our first screen, let's test it out by clicking on the "Live Test" option at the top of the Thunkable website page, on the right-side.



You should get a preview screen like this, with just a blank screen and your colour, mine is lime green.

Thunkable Tutorial Food Court App Tutorial

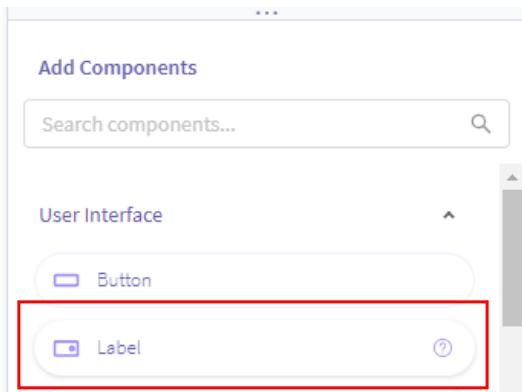


Step: Add a version number label to the Home Screen

Reason: Familiarize with User Interface controls (Thunkable calls them components), and add the version number that we will use later when live testing on our devices.

Let's add a title to the home page of our app and at the same time also add a version number that we will use later.

1. Start by locating the "Label" User Interface (UI) component on the left side of the "Design" tab.

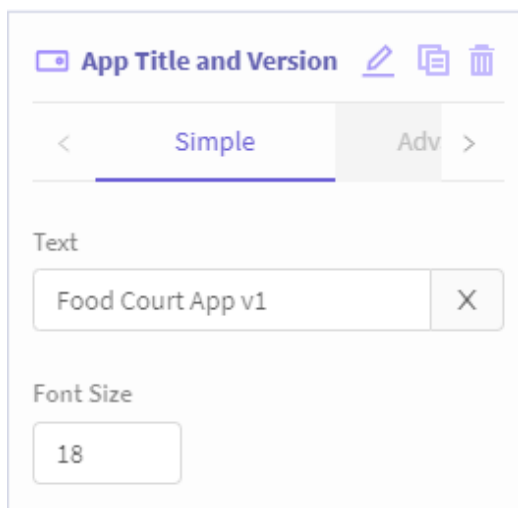


2. Click and drag the label onto your Home Screen

Thunkable Tutorial Food Court App Tutorial



3. Change the "Text" for the label to the name of your app, mine is called "Food Court" and add a version number at the end, starting with the letter v, mine will be v1.
4. Let's also change the name of this label from "Label1" to something meaningful like "App Title and Version".



5. You can also change the size, style, colour, placement, etc. of the font here. So go ahead and make your font bigger, change its colour, etc. Mine is Font Size: 24, Font Weight: Bold, Text Align: Center; Width: Fill Container



9. You will need to manually update this each time you are about to live test on the device.

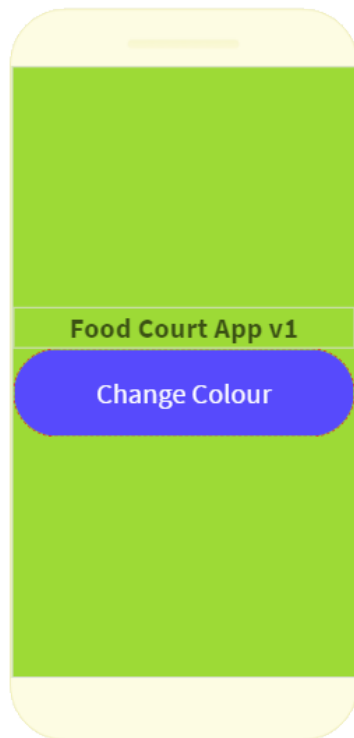
Thinkable Tutorial Food Court App Tutorial

Step: Add a “Change Colour Button” to the Home Screen

Reason: reason: to learn about controls, placement and control properties

- a. Add a button -
 1. name it “Change Colour Button”
 2. change its Text to “Change Colour” (or something else)
 3. change the text to “Change Colour” (I’ve chosen blue: #2962ff)
 4. stretch the button to fill the width of the container
 5. change the height of the button to 75 pixels tall
 6. change the Font Size to 24 of the button to make it easier to read
 7. Test App to make sure button appears - it doesn’t do anything yet, but it should be on your screen.

➤ Live Test app



You will notice that clicking the “Change Colour” button doesn’t do anything, we’ll fix that next.

Things to try:

10. Try changing the look of the button by removing or changing the rounded edges of the buttons so your button is squarer, for example.
11. Try changing the padding and margins of the button to see how that changes the look of your button.

Step: Make “Change Colour Button” change the Home Screen background colour with code

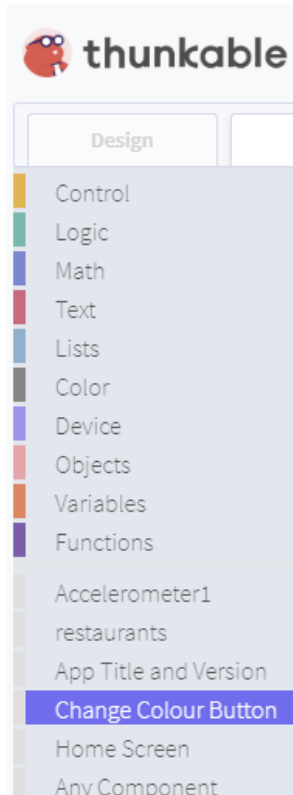
Reasons: to add code, test our app, learn about colours

Now that we have our button, we need to make the button actually do what it says it will do – change the background colour of our screen. To do this we will need to add some code to the button’s “Click” event.

12. Click on Blocks (top left corner of the Thunkable web page)

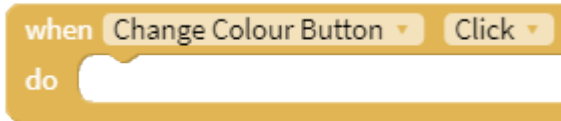


13. Locate and click on the “Change Colour Button” on the left, under the block categories

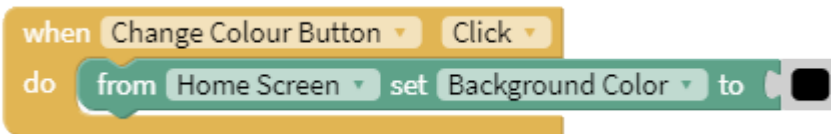


14. Choose “when ‘Change Colour Button’ ‘Click’ do” and drag it to workspace

Thunkable Tutorial Food Court App Tutorial



15. Click on the “Home Screen” component on the left-side of Block and choose “from ‘Home Screen’ set ‘Background Color’ to” block and snap it into the “do” section of “when ‘Change Colour Button’ ‘Click’”.



➤ Live Test app

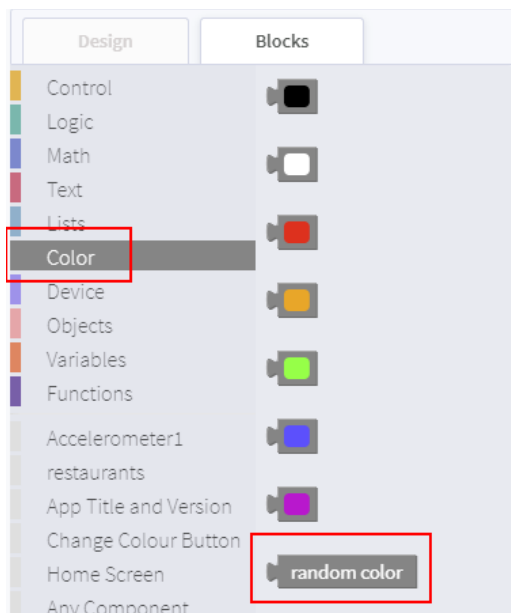
Test now... does the button change the background colour to ‘Black’ (the default colour)?

Step: Make the background change to a random colour when “Change Colour Button” is clicked.

Reason: introduce random

Okay, let’s change the background to a random colour – random means just pick any colour.

16. Return to Blocks and choose the “Color” category.

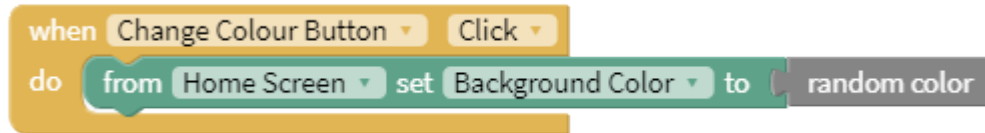


Thinkable Tutorial Food Court App Tutorial

17. Find the “random color” block



18. Drop the ‘random color’ block to the right code area and replace the default black colour block with the new ‘random color’ block.



➤ Live Test app

Test your app, click the ‘Change Colour Button’ and your background should change.

Things to try:

19. Try changing the colour of the buttons when the screen colour changes.
 - a. Try making them change to a random colour too.
 - b. Try making them change to a complementary colour (you’ll need to do some math for this). Source: <https://www.101computing.net/complementary-colours-algorithm/>

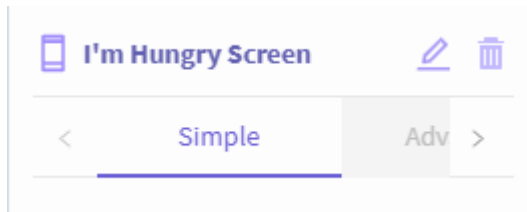
Adding the “I’m Hungry Screen”

Now that we have our “Home Screen” working, let’s add our “I’m Hungry” screen so we can get the app to help us choose a restaurant.

Step: adding a second screen, “I’m Hungry”

reason: adding new functionality with more screens

20. Adding a second screen to our app
 - a. add the new screen and give it a name: “I’m Hungry Screen”

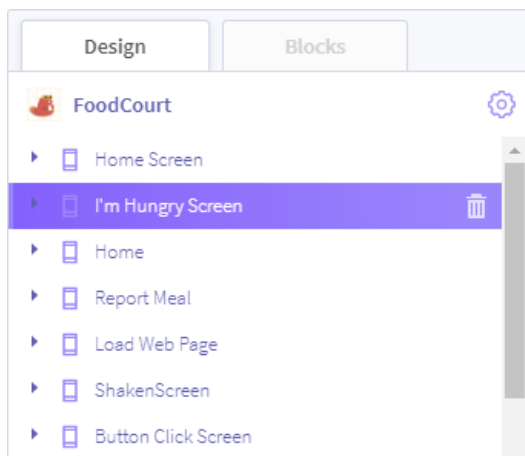


➤ Live Test app

Problem: there is no way to get to our “I’m Hungry” screen, instead our Home Screen appears.

Things to try:

21. If you want to make this the first screen in your app, instead of the Home Screen, you can drag this screen to the top of the list of screens on the left side of the Design screen. I found it easier to do if I collapsed the screens and their components to just the screen names and then I could drag and drop them in the order I wanted them in.



Thunkable Tutorial Food Court App Tutorial

Step: adding a Home Screen button to navigate to the “I’m Hungry” screen

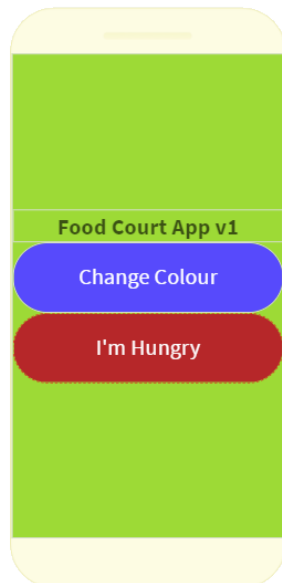
reason: screen navigation

To fix the issue of the Home Screen starting first with no way to get to the “I’m Hungry” screen, we’ll add a button to the Home Screen that navigates the user to the I’m Hungry screen.

- a. Open the Home Screen
- b. Add a new button, name it “I’m Hungry Button” (or Jump to I’m Hungry, etc), and change the text to “I’m Hungry”.
- c. You can also change the size, colour, etc.

➤ Live Test app

Problem: the button appears, but nothing happens when I click on it?! That’s because we need to add code to tell the app what to do when the button is clicked / pressed.



Things to try:

22. If the “I’m Hungry” button is in the wrong place, you can use the Component mechanism in the top-right corner of the Thinkable app to change the ordering of the buttons.

Step: add a second screen, “I’m Hungry”

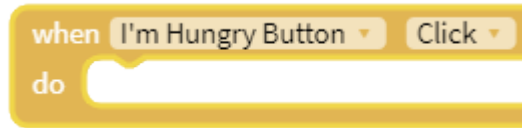
reason: screen navigation and user experience

We need to add a new “when ‘I’m Hungry Button’ ‘Click’ do” event.

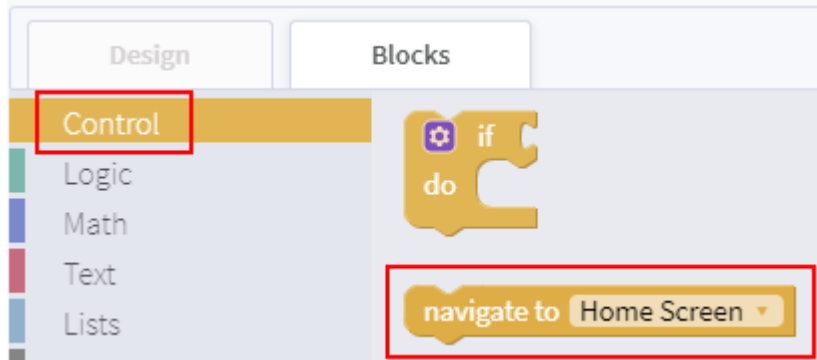
1. On the Home Screen, click on the Blocks tab to switch to block-coding mode.
2. On the left-side, locate the “I’m Hungry Button” component and click on it.

Thunkable Tutorial Food Court App Tutorial

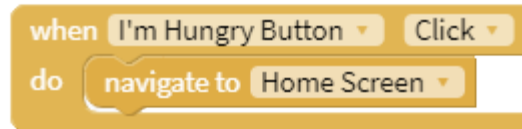
3. Find the new “when ‘I’m Hungry Button’ ‘Click’ do” event on the left side and drag it over to the block area in the middle of the screen.



4. Locate the “Control” section of Design on the left side and click on it to see the blocks available there. We are looking for the “navigate to ...” block.



5. Select the “navigate to ‘Home Screen’” block and drag it over and snap it into the “do” section of our “when ‘I’m Hungry Button’ ‘Click’” block.



6. Change the navigate target from “Home Screen” to “I’m Hungry Screen”

➤ Live Test app

Success, we can navigate to the I’m Hungry screen from the Home Screen.

Problem: no way to get back to the home page?!

Step: add a “Home Screen” button to the “I’m Hungry Screen”

Reason: complete the navigation cycle, user experience

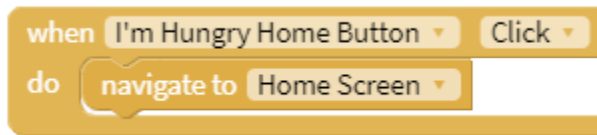
7. Go to the I’m Hungry Screen in Design and add a new button named, “I’m Hungry Home Button” and titled “Home Screen”.

Note that due to naming conflicts we can’t just call this “Home Button” because any other screen that also has a home button would not be able to use the same name. This is not the case with

Thunkable Tutorial Food Court App Tutorial

most programming languages and may change in Thinkable in the future, but for now we need to uniquely name each component.

8. Switch to Blocks mode.
9. Locate the “I’m Hungry Home Button”, click on it and find the new “when ‘I’m Hungry Home Button’ ‘Click’ do” component and drag it over to the block area.
10. Return to the block list on the left, click “Control”, and locate the “navigate to ‘Home Screen’” block. Drag this block over and snap it into the “when ‘I’m Hungry Home Button’ ‘Click’ do” component.



➤ Live Test app

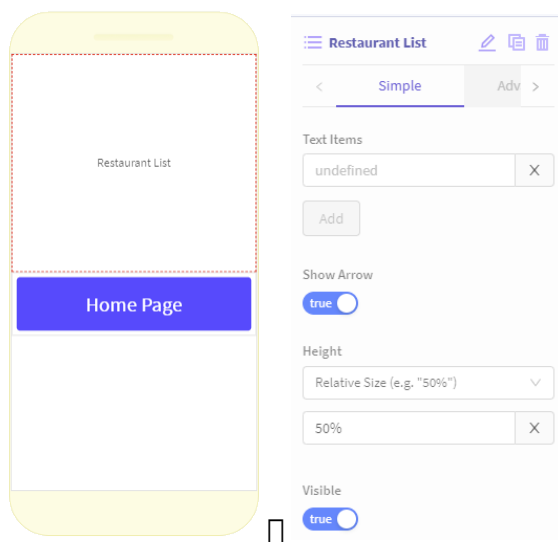
Success, we can now navigate from the Home Screen to the I’m Hungry screen and back!

List the available restaurants on the “I’m Hungry Screen”

Step: add a list component to the I’m Hungry Screen

Reason: try the list control, see visually, our list of restaurants

11. Navigate to the I’m Hungry Screen.
12. Add a list component to the I’m Hungry screen.
13. Name the new component “Restaurant List” and make it Relative Size of 50% or it will take over the entire screen.



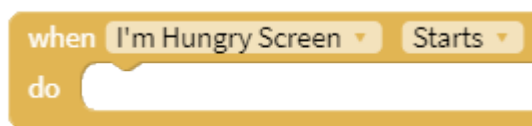
Note: you could add your restaurants here manually using the “Text Items” property but I want to do it using code so I’m skipping this step.

Step: add numbers to our restaurant list using code

Reason: adding list items using code

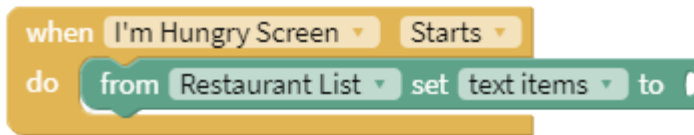
Now that we have our list of restaurants, we need to actually add items to our list. We could do this by manually adding them to the Restaurant List via the right-side properties but instead we will add them using code so we can control the list better later on.

14. Navigate to the I’m Hungry Screen and switch to Block mode.
15. In the list of components on the left, click on the “I’m Hungry Screen” and locate the new “when ‘I’m Hungry Screen’ ‘Starts’ do” control block and drag it over to the code area.

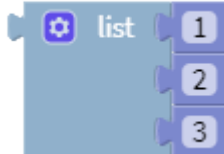


Thinkable Tutorial Food Court App Tutorial

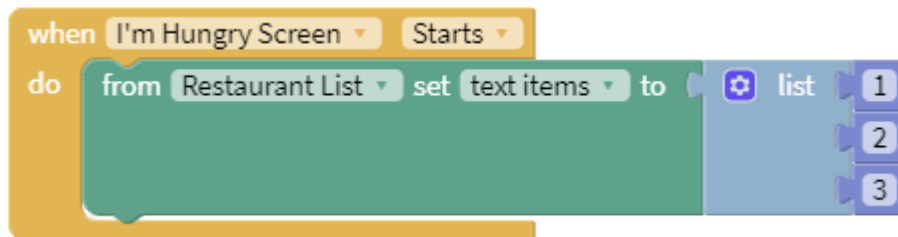
16. Next, we need the Restaurant List control, so again on the left side, locate the “Restaurant List” control, click on it and locate the “from ‘Restaurant List’ set ‘text items’ to” (green) block and drag it over and snap to the “when ‘I’m Hungry Screen’ ‘Starts’ do” block we just added.



17. Now we need to add a list of items, so return to the left-side list of blocks and click the “Lists” and choose the first item, which is a list of the numbers 1, 2, and 3.

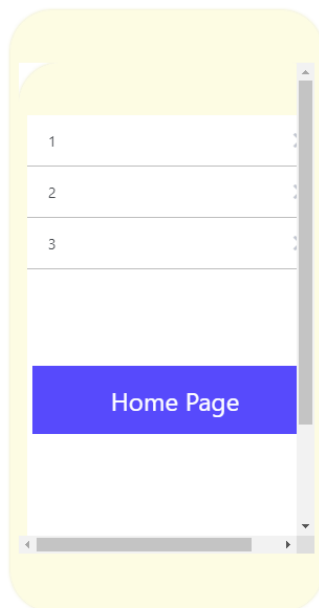


18. Drag the number list over and snap it to the right-side of the “from ‘Restaurant List’ set ‘text items’ to” block.



➤ Live Test app

Make sure the three numbers appear



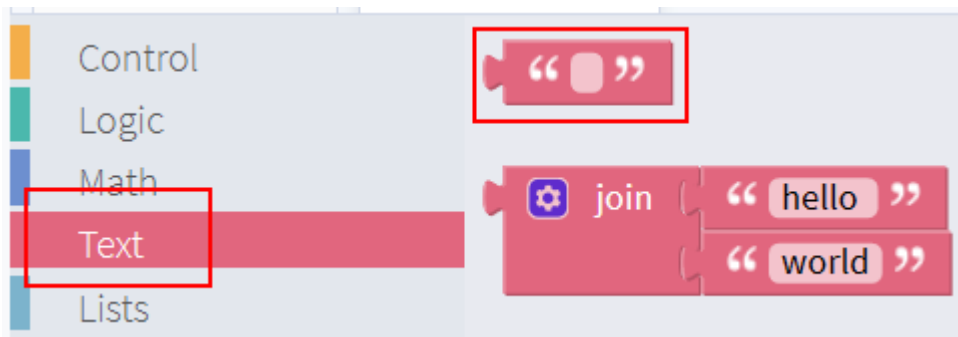
Thunkable Tutorial Food Court App Tutorial

NOTE: I found a bug when I just added the list items via blocks and the list would be empty!? Then I manually added some "Text Items" via the properties but left the Block-assignment logic and voila my Block-assignment worked.

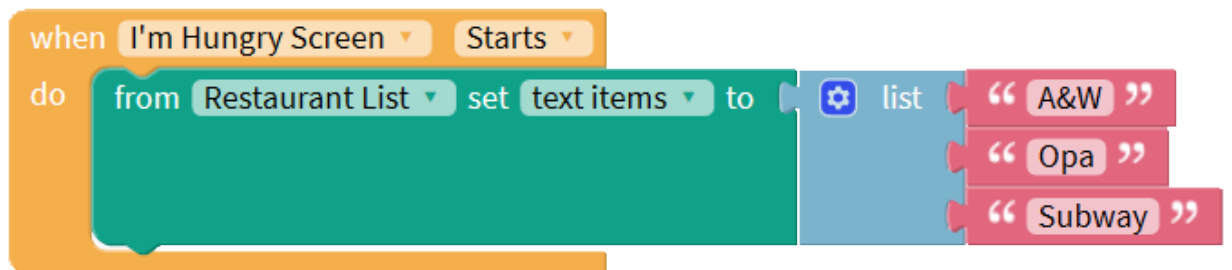
Step: replace list numbers with list names/text (strings)

Now we need to change from numbers to names – so we select the "Text" block group (pink) and replace our "1", "2", and "3" with names of restaurants (sample). NOTE the quotes around the text items, this is a nod to text-based programming.

19. Return to the Blocks mode again and on the left locate the "Text" blocks.
20. Locate the blank text block:

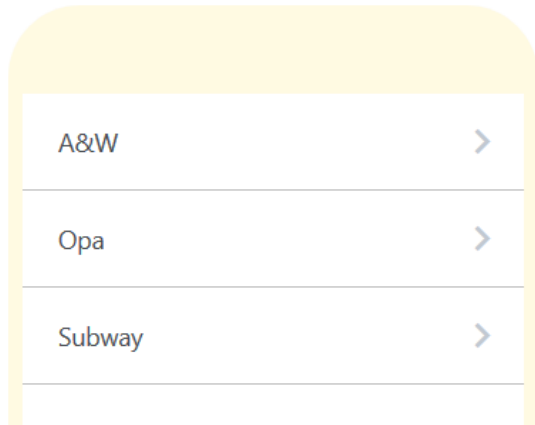


21. Drag the three empty text blocks over to the block area and replace each of the numbers blocks 1, 2 and 3 with the blank text.
22. Replace the blank text with three restaurant names, I've chosen A&W, Opa!, and Subway (because they are quick to type).



➤ Live Test app

Your text restaurant list should appear.



Choose a random restaurant on the I'm Hungry screen

Now that we have our list of restaurants we will add a button to randomly select one of our restaurants, i.e. let the app choose a restaurant for us.

Step: add a "Random Restaurant" button to the I'm Hungry Screen

reason: add an action button, randomly choose and continue with live test ability

1. Return to the "I'm Hungry Screen" in Design mode.
2. Add a new button and name it "Random Restaurant Button", set it's text to "Random Restaurant" (or "Choose Random Restaurant"), set the size to 24, height to 75 px, and width to fill container.



Thunkable Tutorial Food Court App Tutorial

- Next add a label, name it “Random Restaurant Label” and set its text to “Random Restaurant”. Place it below the list (TODO: may need to drag-and-drop ordering via the controls list on the top left side of the screen) TODO: text for the label?



➤ Live Test app

Make sure the text and button are shown, they won't do anything yet.

Step: connect the Random Restaurant button to the label when the button is pressed

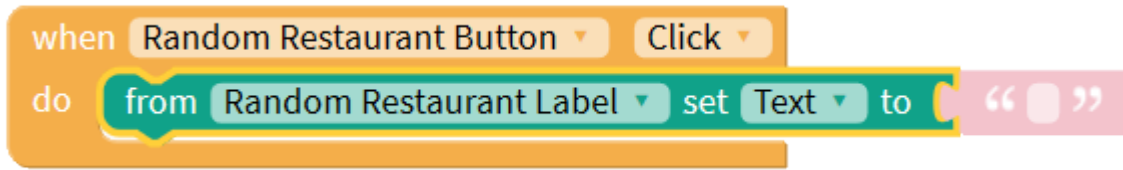
Reason: connect the button click event to changing the label action

- On the I'm Hungry Screen, switch to Blocks mode.
- On the left component area click on the “Random Restaurant Button”, select the “when'Random Restaurant Button' 'Click' Do” component, and drag it to the script area.



- Next, click on the “Random Restaurant Label” on the left and select the “from 'Random Restaurant Label' set 'Text' to ""” and snap this into the 'do' area that we just added to make this the action for when we click the “Random Restaurant Button”

Thinkable Tutorial Food Court App Tutorial

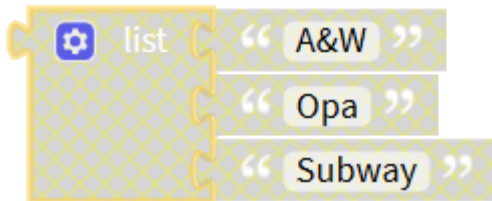


7.
 - a. Test App And verify that when we click the “Random Restaurant Button” that our “choices” label changes to blank - not random yet, but we know that clicking the button changes the label text.

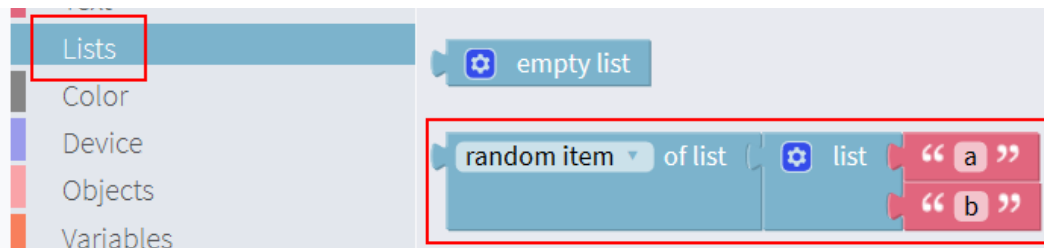
Step: randomly choose a restaurant when the Random Restaurant button is pressed

Reason: demonstrate how to randomly select a list item with a button click

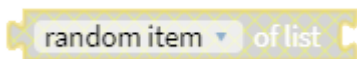
8. Return to the “I’m Hungry Screen” and then Blocks.
9. Right-click on our restaurant list and choose “Duplicate” to make a copy of our list.



10. Then return to the Lists group and drag over a “random item” of list combination block:



11. Disconnect the second half of this block, the list and the a & b, so you just have the “random item” of list section, like this:

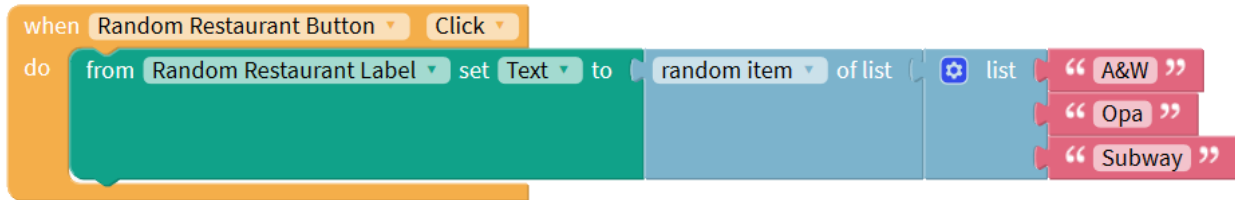


12. Next drag your duplicate and snap it into the right-hand side of the “random item” of list block, like this:

Thunkable Tutorial Food Court App Tutorial



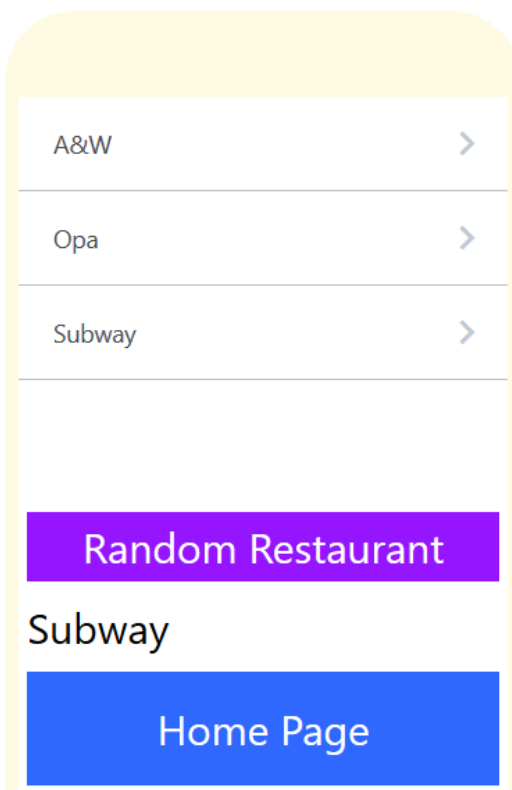
- Now drag your new “random item” combination block and snap it into the “from ‘Random Restaurant Label’ set ‘Text’ to” block, replacing the pink quotes block.



- Test your app and when you click the “Random Restaurant Button” it should choose one of your restaurants each time you click the button.

➤ Live Test app

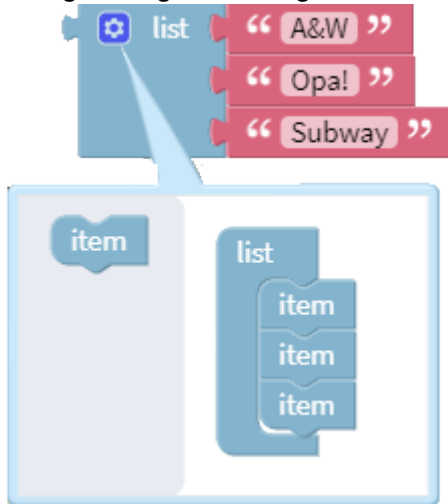
Now when you click the “Random Restaurant” button the app should randomly choose one of your restaurants!



Thunkable Tutorial Food Court App Tutorial

Things to try:

13. Add more restaurants to your list by expanding the number of list items for the “list” block by clicking the cog and adding more text slots.



14. Add more restaurants using the “make ‘list from text’



15. Try using the Accelerometer component on the I'm Hungry code to allow you to randomly select a restaurant when you shake the app. You might want to make the background colour also change when you shake so you know that the restaurant changed due to the shaking. Note that you cannot currently test the accelerometers via the web-based Live Test only on a device using Live Test.

Using a variable for our list

One of the issues with making and keeping duplicate copies of our list of restaurants is that if we want to change the list of restaurants, we have to change all copies of the list which is too much work and prone to errors. Instead we will create a single list of restaurants and put them into something called a variable and then we just use this variable wherever we need the entire list.

What is a variable?

A variable is just a container with a name and inside the container is a value like a name, number or list of items. You use the variable as input to something, like a list, or as a place to store output from something like a score, or the randomly chosen restaurant when you click your button, etc.

Step: creating a variable named “Restaurant List”

In the I’m Hungry screen we’ll add a new variable named “Restaurant List” and fill it up with our list of restaurants used previously.

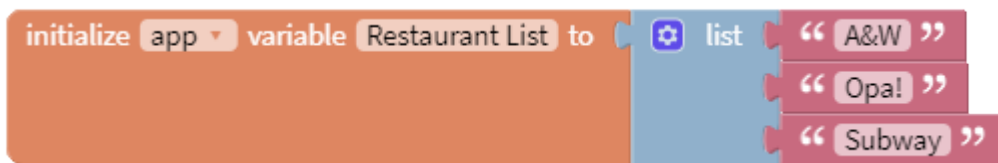
1. In the Blocks mode for the I’m Hungry screen click on the Variables blocks on the left side.
2. Click on the “initialize ‘app’ variable ‘name’ to” block and drag it over to the code area.



3. Set the name of the variable to “Restaurant List”



4. Disconnect one of the “list” variables that you already have that contains your list of restaurants and snap it to the right-side of this new variable.



Step: replace all copies of your list with your new “Restaurant List” variable

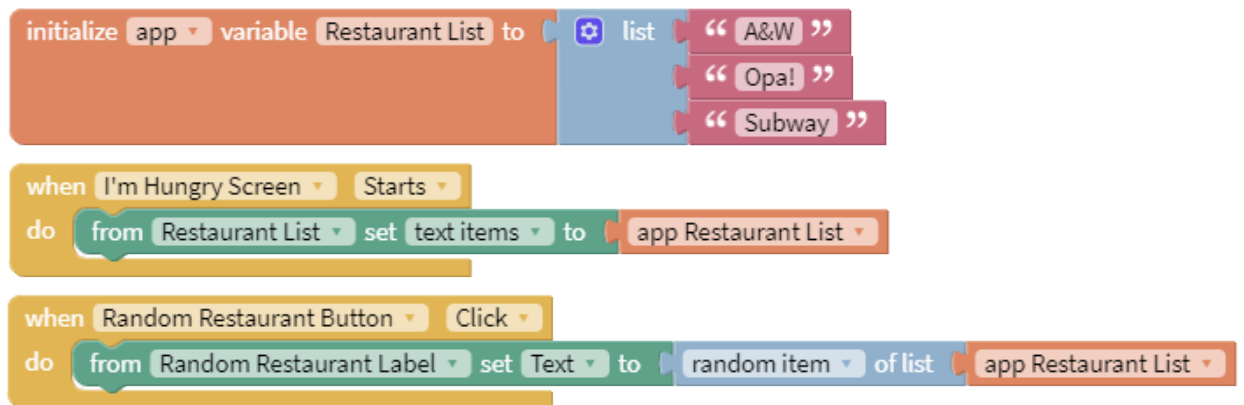
Now that we have our new Restaurant List variable we will remove our duplicate lists from before and replace them with the new variable instead.

5. Get a new copy of the variable “app Restaurant List”



Thunkable Tutorial Food Court App Tutorial

6. Replace all copies of your duplicated list, except for the one used to initialize the Restaurant List variable, with the new Restaurant List variable block. You can create copies of the variable as you need.



Note that all of the variables are “app” variables which means they live in the application only and are available anywhere in the application and thus are known as global variables.

Note that since we are initializing our “Restaurant List” variable here in our “I’m Hungry Screen”, if the user doesn’t come to this screen but we use this variable on another screen then this variable will be blank on that other screen until they visit this I’m Hungry screen. Therefore, you should consider creating your variables in your home screen (or whatever your first screen is) or creating variables only on the screens that need them.

Adding a local DB and then loading it into a list

Remember when we had our list of strings and we replaced it with a variable so we didn't have to duplicate our list, well now we want to add a new screen and we don't want to have to duplicate the list of restaurants from our previous screen to this screen. So, we will create a database and put all of our data in that storage and then create lists from that master record of data.

Note that your local DB will be reset each time you Live Demo or reload the application, so don't rely on this to save your data, at least during development. It will keep any data you manually add to it, but it will not keep data that is input via the app itself. This may work once the app is deployed but you are encouraged (by Thunkable) to use something like Firebase (they call it the Realtime DB) or Airtable. You will need to set up a free account.

Note that your application is public, unless you create a paid private account with Thunkable, so any cloud service you use, including Airtable or Firebase, will require your access key to be included in your application. This means that anyone who copies or edits your project will have access to your Airtable or Firebase cloud account too.

Step: add a “Local DB” (DB = database) to our project.

23. In Design mode, in the bottom left, there is a set of components called Data and one of them is called “Local DB” which is short for local database.



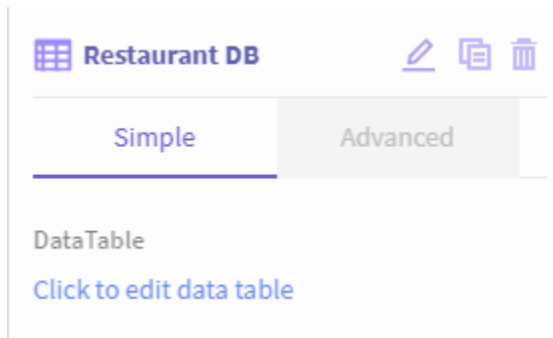
24. Drag a Local DB to your application but note that this is not a visual component, like for example a button, so the database component will appear at the bottom middle of the design window separate from the app screen.



Step: name the database “Restaurants DB”

1. Click on the Local DB and in the properties area (top right) change the name to “Restaurant DB”.

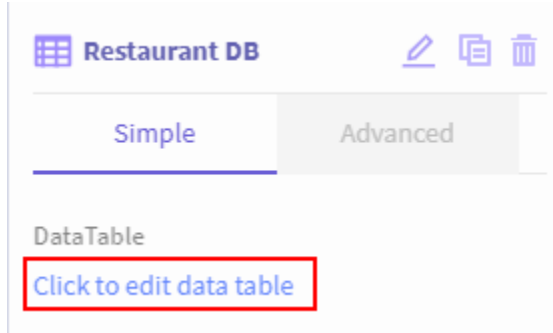
Thinkable Tutorial Food Court App Tutorial



Step: add data to the database

Now that we have our database, we need to add data to it.

1. Click the “Click to edit data table” link for the database.



2. I've added a few fields for future features but for the initial list of restaurants you can simply rename the first column to “name” (or whatever you want to call it) and input your list of restaurants as shown here.

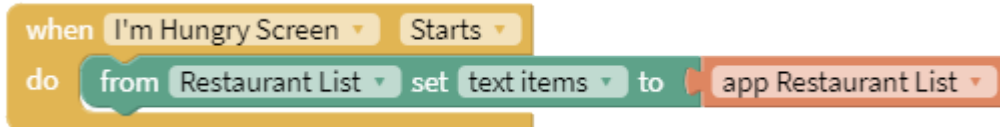


Thinkable Tutorial Food Court App Tutorial

Step: reset the Restaurant List to empty when the I'm Hungry Screen starts

We currently have a variable named "Restaurant List" that we load using a static list of restaurants when our application starts. We will replace this start-up value by loading the list from our new database when our I'm Hungry screen starts. Note that we should change this to when our Home Screen starts but to keep this simple, we will do it on this screen instead.

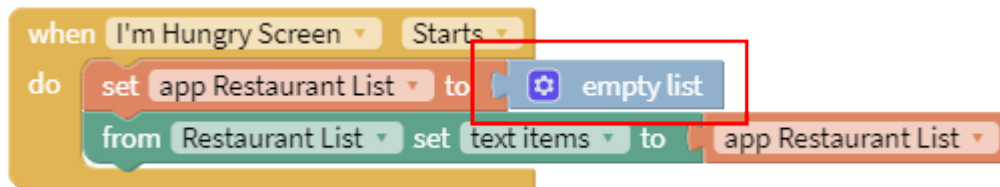
1. Return to the I'm Hungry screen and locate our existing "who "I'm Hungry Screen" "Starts" do"" block, which looks like this:



2. We first need to reset our Restaurant List back to an empty list, otherwise we will load our database as more items to the list instead of replacing the list items with the items from our database. So, click on the Variables blocks and choose the "set 'app Restaurant List' to" block and snap it into the top of our 'when "I'm Hungry Screen" "Starts" do' block, like shown below. Note that the new set-block is the first block now.



3. Now we have our 'set' block in place click on the Lists blocks and drag over the "empty list" block and snap it into the 'set "app Restaurant List"' block.



➤ Live Test app

Now when you click on the I'm Hungry screen the restaurant list should be blank because we reset the list to empty on page load. We will load our list of restaurants from the database next to fix this.

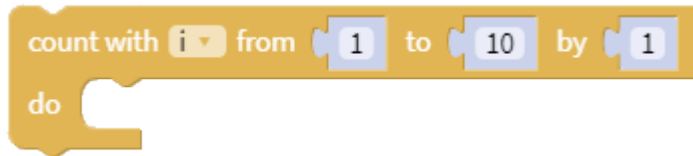
Step: load our "Restaurant List" variable from our new "Restaurant DB"

Reason: introduce a loop and retrieve data from a database

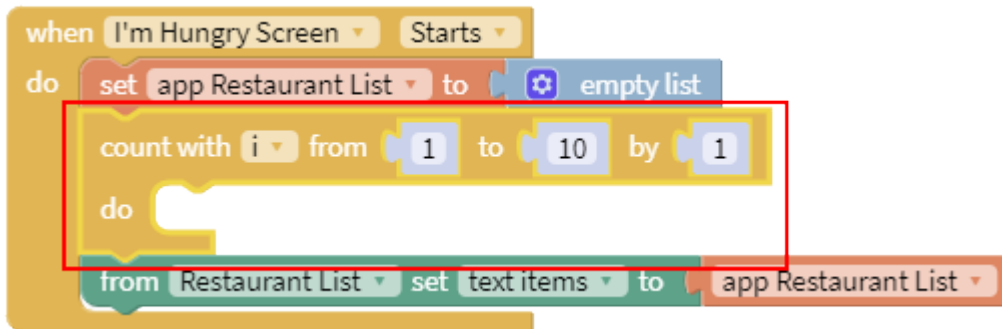
Now that we have our list empty

1. Next, we will add a new block called a count-with-do block, also known as a loop, that will allow us to read each of the rows of data in our database and load them into our Restaurant List variable. In the Controls blocks locate the "count with 'i' from '1' to '10' by '1' do" block:

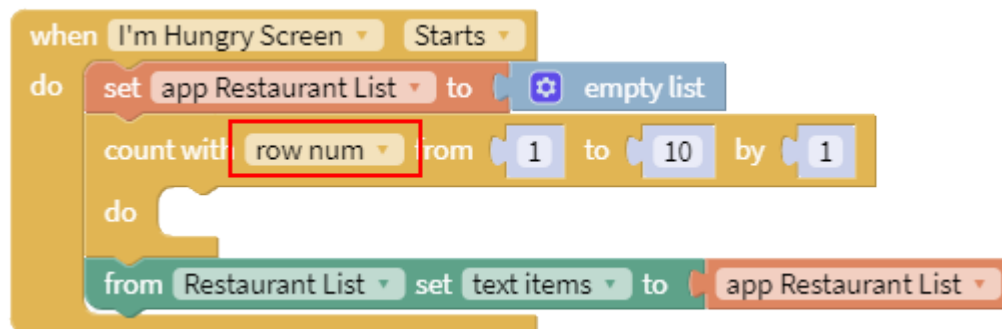
Thinkable Tutorial Food Court App Tutorial



2. Snap the count-with-do block between the 'set "app Restaurant List"' block and the 'from "Restaurant List" set "text items" to "app Restaurant List"' as shown here:



3. Our new count-with-do block has four things we can change and then we need to add something to do inside the "do" section of the block. We will start by changing the "i" to something more meaningful and in our case we want to loop through the rows of our database and so we will rename this to "row num" which is short for row number. Click on the "i" and choose the "Rename Variable" option and type in the new name:

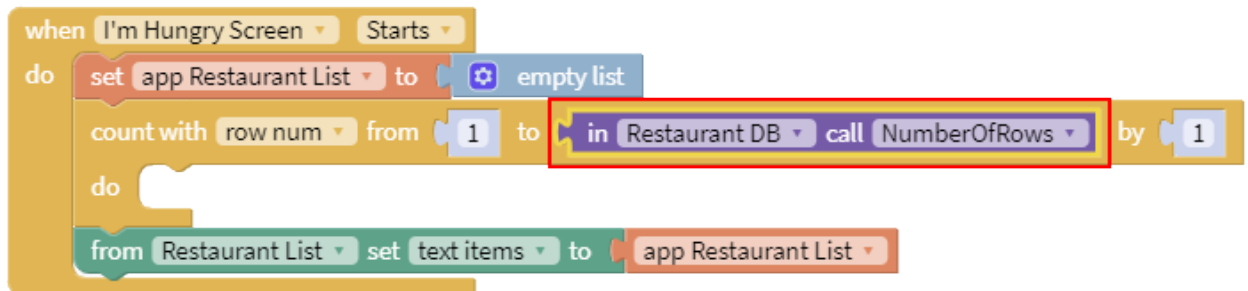


4. We will leave the first number one "1" as 1 because the rows of our database start with row #1.
5. We will change the second number, the ten "10", to the number of rows in our database. We can get the NumberOfRows by clicking on the Restaurant DB on the left-side of our list of blocks and then locating the "in 'Restaurant DB' call "NumberOfRows":

Thinkable Tutorial Food Court App Tutorial



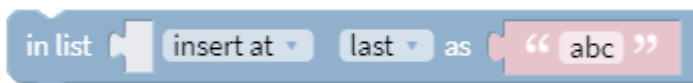
- Now, we seem to have a problem though, look at the shape of the “in ‘Restaurant DB” call “NumberOfRows” block, it’s not the right shape?! It turns out this is a shape-shifting block... and Thinkable unfortunately doesn’t make that clear here. We only discover this when we drag it to the right and attempt to snap it into the place of the number ten and the block magically shape-shifts – try it.



- Now that we have our loop, we need to get the name of each restaurant from the first column of each row. Remember what our database looks like:

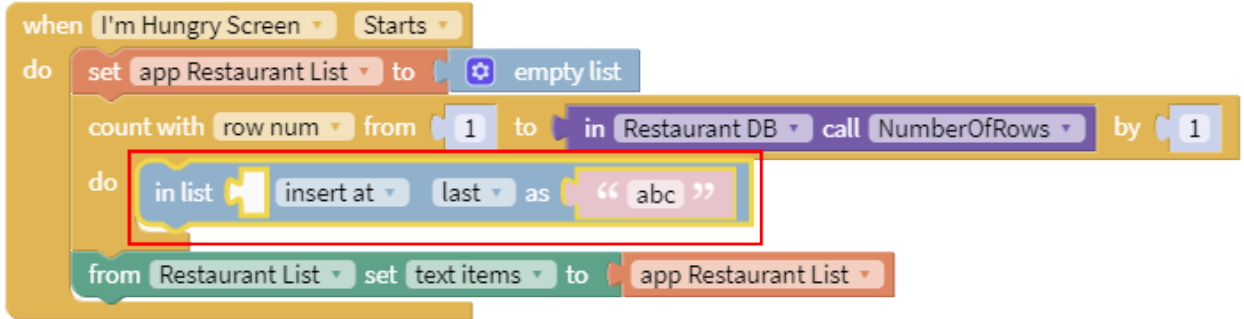
	(1) name	type	num_meals	ratings
1	Subway	sandwich	0	
2	Opa	wrap	0	
3	Steve's Poke	poke	0	
4	KFC	chicken	0	
5	A&W	burger	0	
6				

- Before we get the data to add to our list we need to get the list block that allows us to append (add to the end of our list), this is the “in list <name> ‘insert at’ ‘last’...” block:



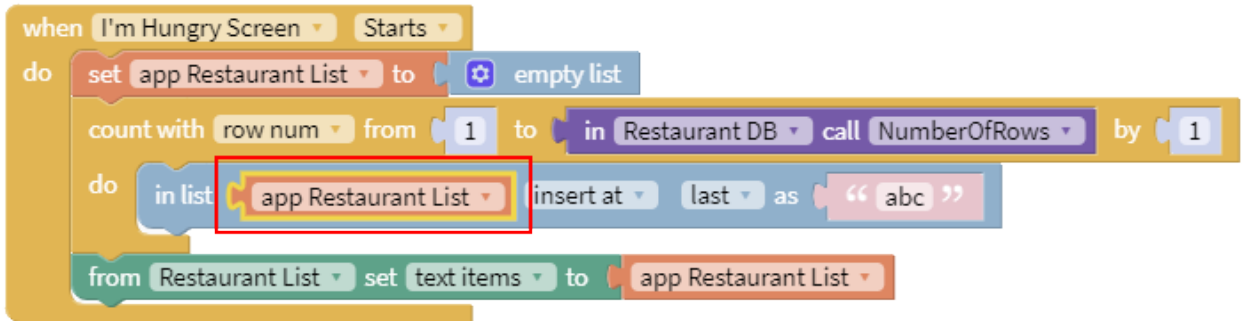
- We will snap this in-list-insert-at-last block into the “do” section of our count-with block:

Thinkable Tutorial Food Court App Tutorial



The code block shows a 'when I'm Hungry Screen Starts' event. It contains a 'do' loop with three blocks: 'set app Restaurant List to empty list', 'count with row num from 1 to in Restaurant DB call NumberOfRows by 1', and 'do in list insert at last as " abc "'. The 'do in list' block is highlighted with a red box.

10. We will replace the first list item in our new in-list block with our “app Restaurant List” variable, as shown here:



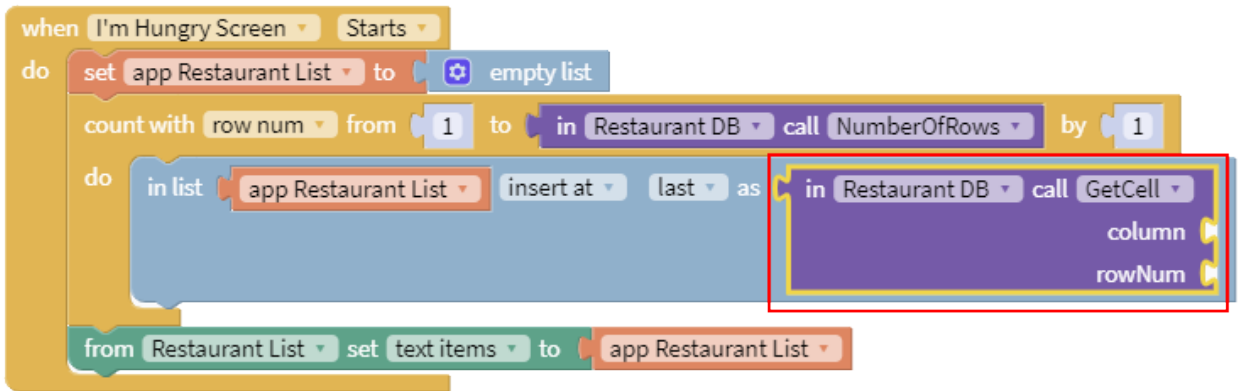
The code block is identical to the previous one, but the 'do in list' block now has 'app Restaurant List' as its first input, highlighted with a red box.

11. Finally, we need to get the first column of the current row in our Restaurant DB. To do this return to the Restaurant DB component blocks again and locate the “in “Restaurants DB” call “GetCell”” block:



A single 'in Restaurant DB call GetCell' block is shown, highlighted with a red box.

Note that this is the wrong shape again... or is it?! Drag the block over and try and snap it in the place of the current “abc” Text block.

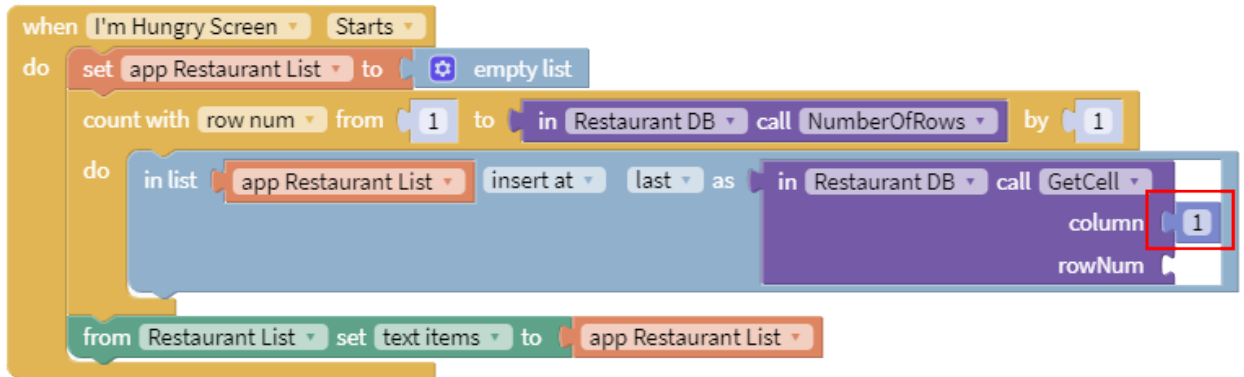


The code block is identical to the previous one, but the 'do in list' block now has 'app Restaurant List' as its first input and 'in Restaurant DB call GetCell' as its second input. The 'in Restaurant DB call GetCell' block is highlighted with a red box and has 'column' and 'rowNum' input fields on its right side.

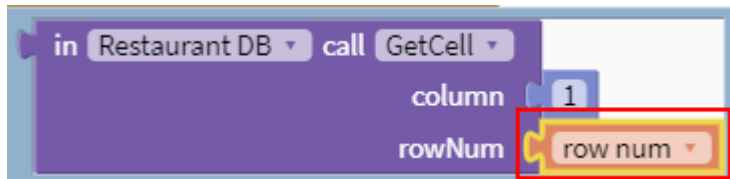
Note that the block not only changed shape but it also now has two input spaces on the right side, one named “column” and the other named “rowNum”.

12. We will grab a number 1 from the Math blocks (actually a number zero, but we’ll change the number to a 1).

Thunkable Tutorial Food Court App Tutorial



13. Next, we need to get each row as we count through the rows in our database, so we will go to the Variables blocks to find a new “row num” variable that we will snap into place in the rowNum slot.



➤ Live Test app

Whew, that was a lot of steps so let's test our screen now. Run the I'm Hungry screen and you should see the list of restaurants that you have put into your database, in the order they are in the database.

Things to try:

1. Try adding a list sorting block to the flow so that your list of restaurants is automatically sorted.

Miscellaneous How-To's

How do I make an image clickable (like a button)?

TODO: clickable images (or images as button)

Don't use an 'image' for a button as the 'image' objects are not clickable... however, buttons can have images and they are still buttons and function as buttons.

The "Button" text can be removed by deleting the value "Button" from the Text property for the button itself and so long as the width and height of the image are controlled and matched to the button size the image will overlay the button correctly, so the user is pressing the image.

