

MIT APP INVENTOR

DAY 1

INSTRUCTIONS

Welcome to the course on MIT App Inventor. We will be using Zoom Application for delivering this course. Please adhere to the following instructions during the presentation.

- All of the participates other than the host are requested to mute (Alt+A) their microphone unless otherwise specified.
- Please use the chat window to type in your doubts and response to questions/tasks.

OVERVIEW

DAY 1

- ❑ Introduction to MIT App Inventor.

- ❑ Familiarizing with the app components.

Two main editors: the design editor and the blocks editor.

- ❑ Build four apps :

- Talk To Me 1
- Talk To Me 2
- Ball Bounce
- Digital Doodle

Introduction

- ❑ MIT App Inventor is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology.
- ❑ It provides a web-based “What you see is what you get” editor for building mobile phone applications targeting the Android and iOS operating systems.
- ❑ It uses a block-based programming language built on Google Blockly.
- ❑ Inspired by languages such as StarLogo TNG and Scratch .

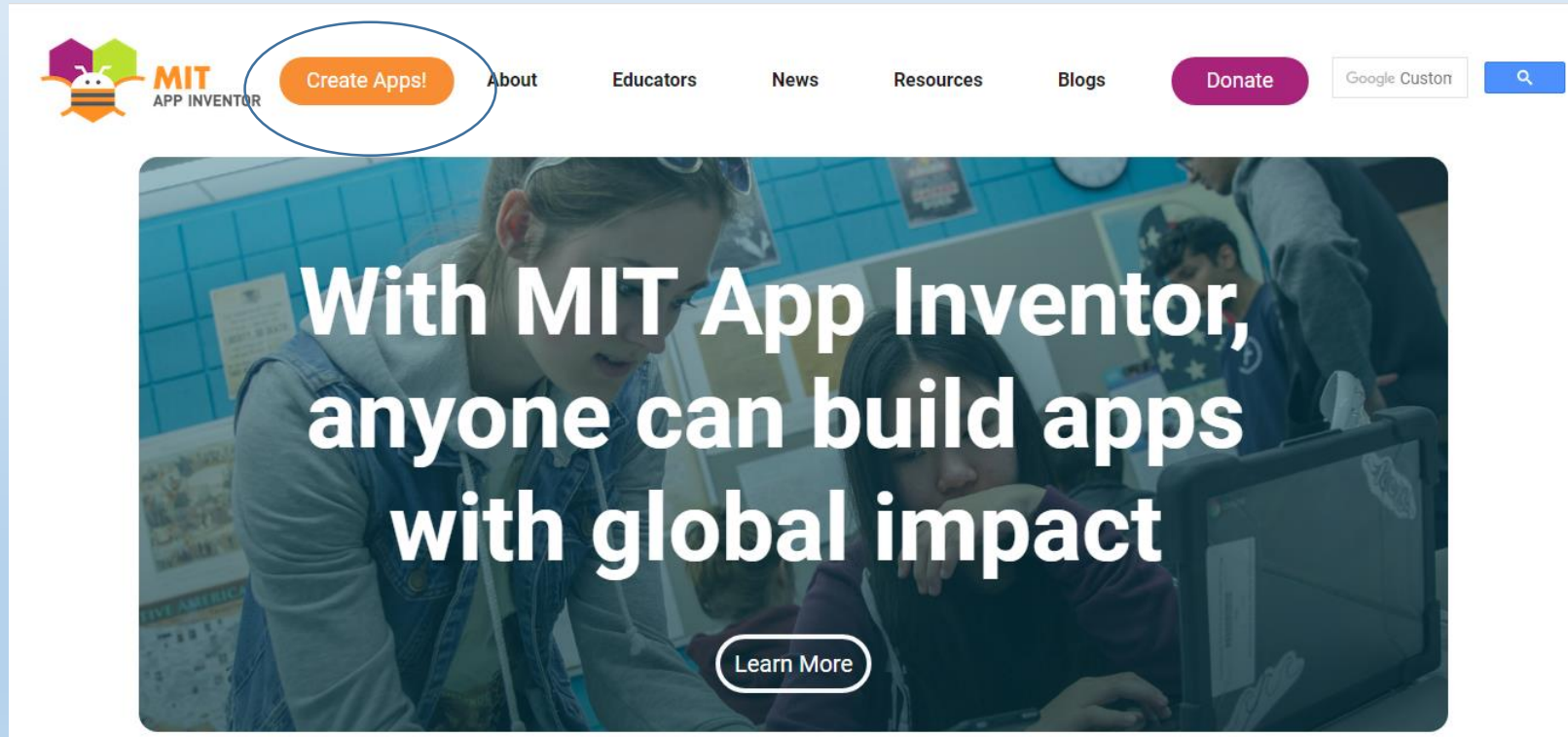
- ❑ People around the world use App Inventor to provide mobile solutions to real problems in their families, communities, and the world.
- ❑ MIT App Inventor is an online development platform that anyone can leverage to solve real-world problems.
- ❑ The smartphone is an information nexus in today's digital age, with access to a nearly infinite supply of content on the web, coupled with rich sensors and personal data.

Key Features

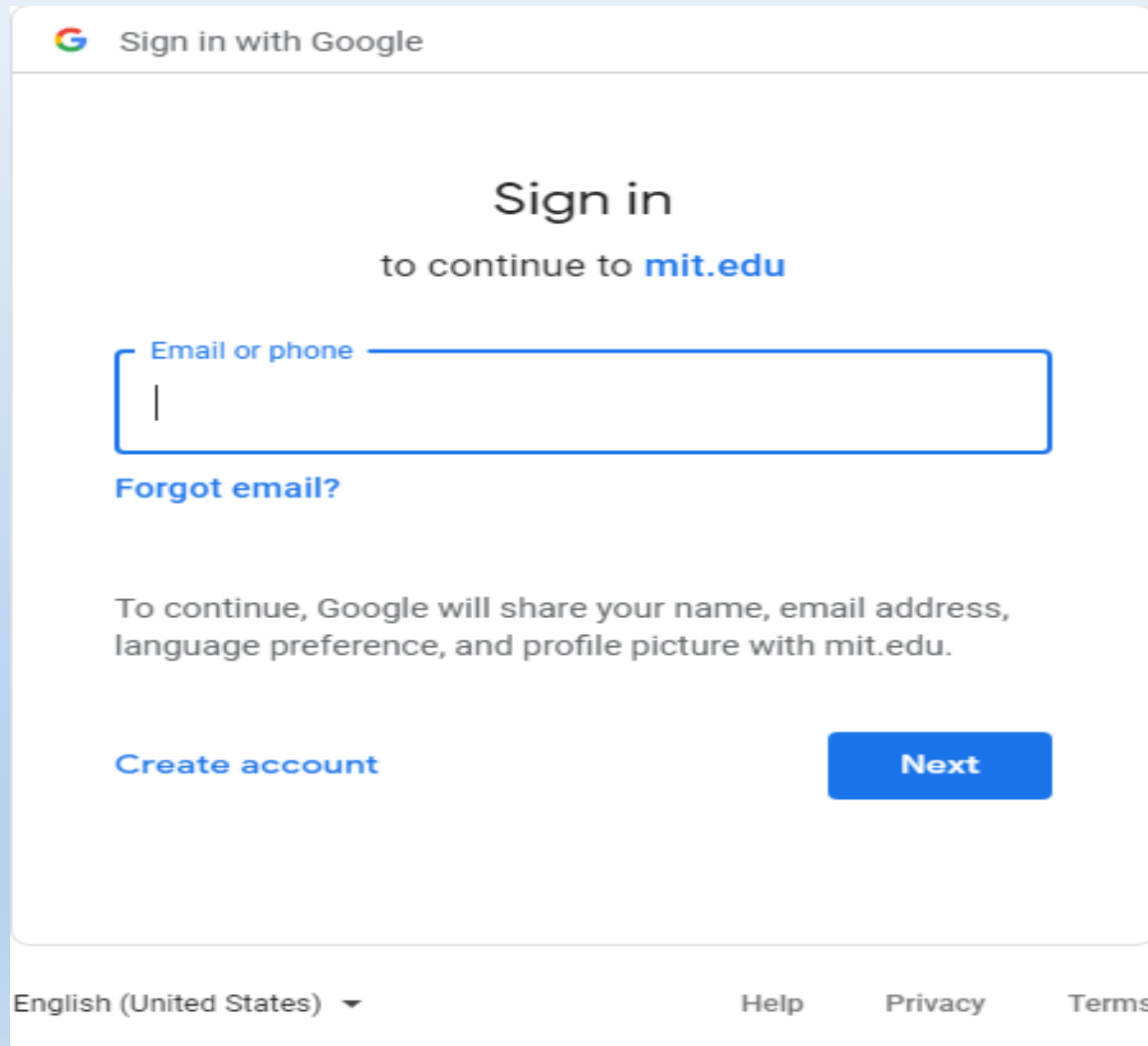
- ❑ MIT is a drag and drop interface to lay out the elements of the application's user interface (UI).
- ❑ It allows users to drag and drop visual objects to create an application that can run on mobile devices.
- ❑ MIT empowering anyone to build a mobile phone app to meet their need.

MIT App Inventor

- ❑ Go directly to <https://appinventor.mit.edu/> , and click the orange "Create" button from the App Inventor website.

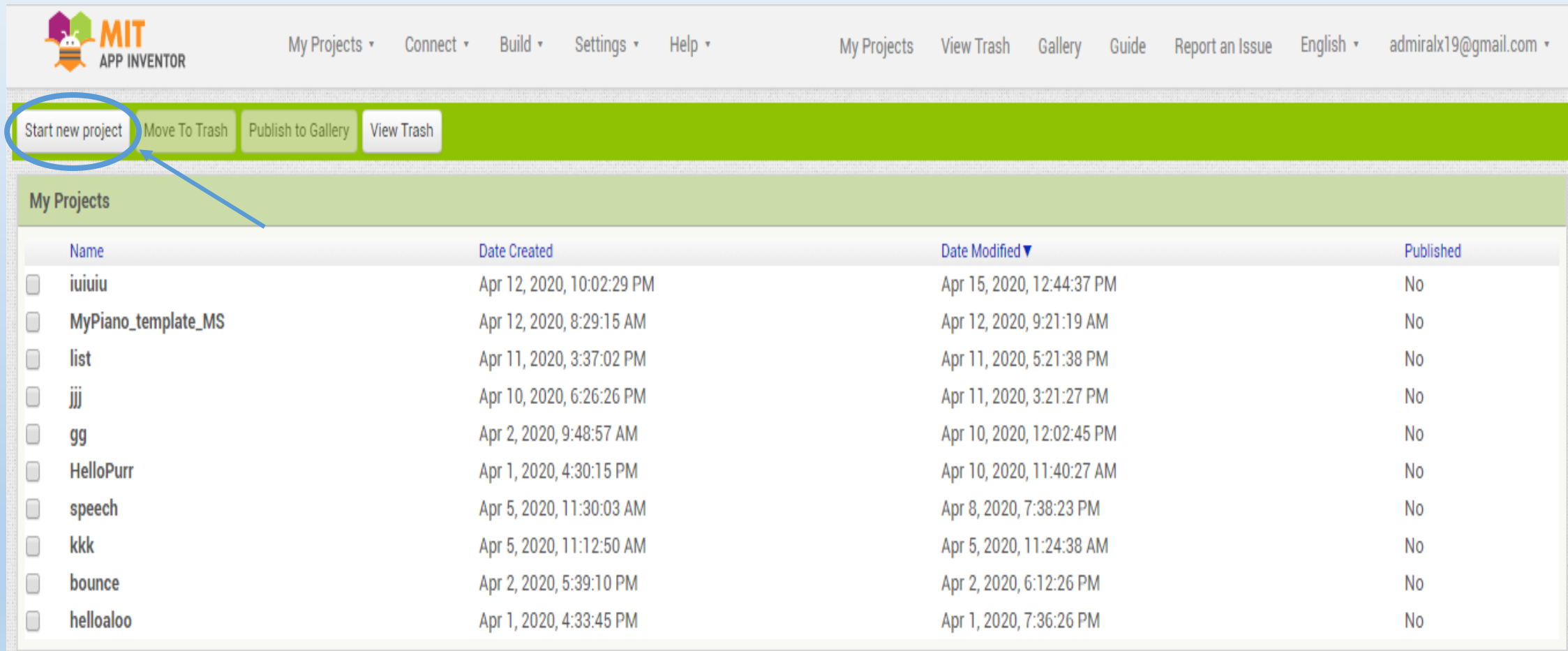


- ❑ Use an existing gmail account or school-based google account to log in to ai2.appinventor.mit.edu To set up a brand new gmail account, go to accounts.google.com/SignUp



The image shows a screenshot of the Google sign-in interface. At the top left, there is a Google logo and the text "Sign in with Google". The main heading is "Sign in" followed by "to continue to [mit.edu](#)". Below this is a text input field with the placeholder "Email or phone" and a vertical cursor. Underneath the input field is a link "Forgot email?". A paragraph of text states: "To continue, Google will share your name, email address, language preference, and profile picture with mit.edu." At the bottom left is a link "Create account" and at the bottom right is a blue button labeled "Next". The footer contains "English (United States)" with a dropdown arrow, and links for "Help", "Privacy", and "Terms".

☐ Start a new project.

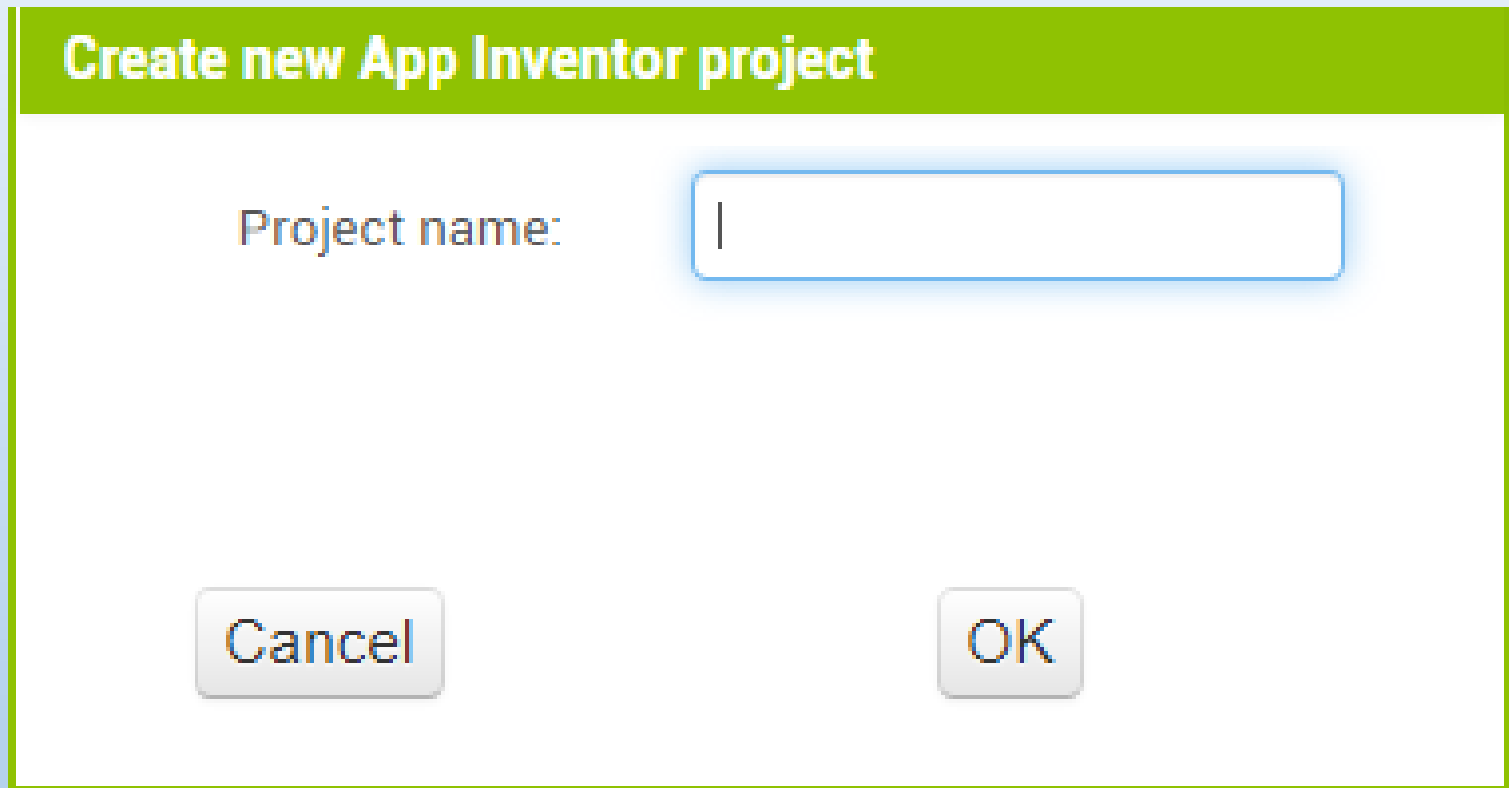


The screenshot shows the MIT App Inventor web interface. At the top left is the MIT App Inventor logo. The top navigation bar includes links for 'My Projects', 'Connect', 'Build', 'Settings', and 'Help'. On the right side of the navigation bar, there are links for 'My Projects', 'View Trash', 'Gallery', 'Guide', 'Report an Issue', 'English', and the user's email 'admiralx19@gmail.com'. Below the navigation bar is a green bar containing four buttons: 'Start new project', 'Move To Trash', 'Publish to Gallery', and 'View Trash'. The 'Start new project' button is circled in blue, and a blue arrow points from it to the 'My Projects' table below. The table has four columns: 'Name', 'Date Created', 'Date Modified', and 'Published'. It lists ten projects with their respective creation and modification dates and whether they are published.

Name	Date Created	Date Modified	Published
<input type="checkbox"/> iuiuiu	Apr 12, 2020, 10:02:29 PM	Apr 15, 2020, 12:44:37 PM	No
<input type="checkbox"/> MyPiano_template_MS	Apr 12, 2020, 8:29:15 AM	Apr 12, 2020, 9:21:19 AM	No
<input type="checkbox"/> list	Apr 11, 2020, 3:37:02 PM	Apr 11, 2020, 5:21:38 PM	No
<input type="checkbox"/> jjj	Apr 10, 2020, 6:26:26 PM	Apr 11, 2020, 3:21:27 PM	No
<input type="checkbox"/> gg	Apr 2, 2020, 9:48:57 AM	Apr 10, 2020, 12:02:45 PM	No
<input type="checkbox"/> HelloPurr	Apr 1, 2020, 4:30:15 PM	Apr 10, 2020, 11:40:27 AM	No
<input type="checkbox"/> speech	Apr 5, 2020, 11:30:03 AM	Apr 8, 2020, 7:38:23 PM	No
<input type="checkbox"/> kkk	Apr 5, 2020, 11:12:50 AM	Apr 5, 2020, 11:24:38 AM	No
<input type="checkbox"/> bounce	Apr 2, 2020, 5:39:10 PM	Apr 2, 2020, 6:12:26 PM	No
<input type="checkbox"/> helloaloo	Apr 1, 2020, 4:33:45 PM	Apr 1, 2020, 7:36:26 PM	No

❑ Name the project.

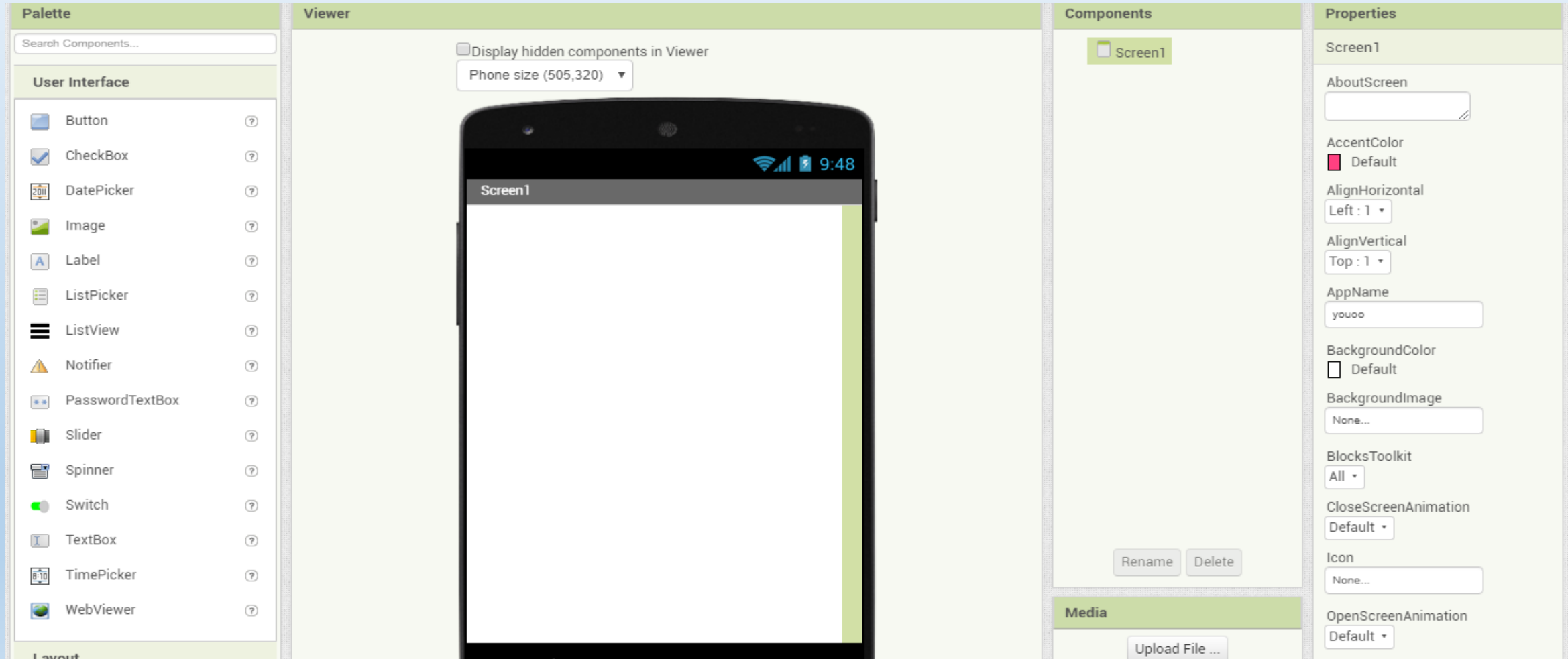
First app we'll call it "Talk to me"



The image shows a dialog box titled "Create new App Inventor project" with a green header. Below the header, the text "Project name:" is followed by a text input field containing a single vertical bar character "|". At the bottom of the dialog, there are two buttons: "Cancel" on the left and "OK" on the right.

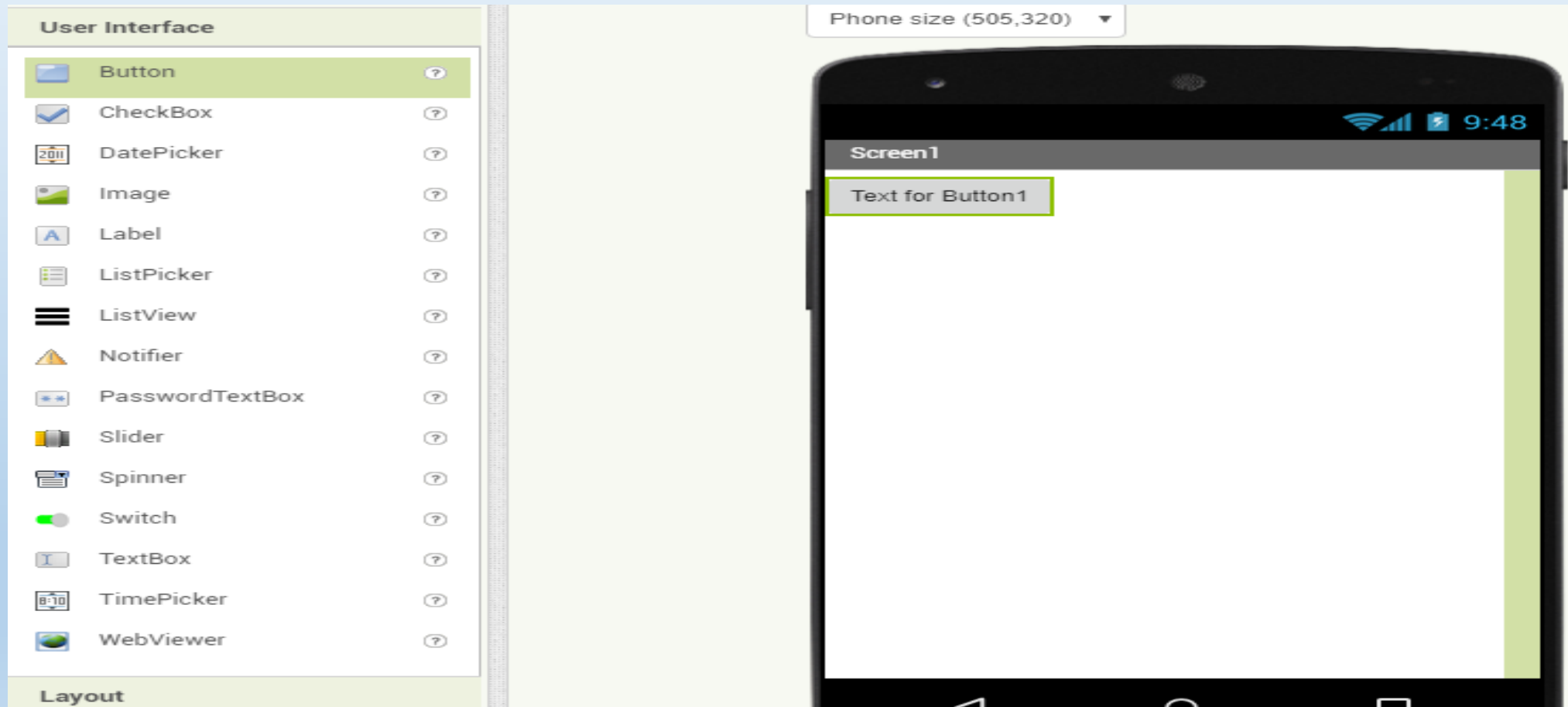
□ The Design Window

The Design Window, or simply "Designer" is where you lay out the look and feel of your app, and specify what functionalities it should have.



□ Add a Button

Our project needs a button. Click and hold on the word "Button" in the palette. Drag your mouse over to the Viewer. Drop the button and a new button will appear on the Viewer.



❑ Connect App Inventor to your phone for live testing

Test your app after each by connecting to your phone.



My Projects ▾

Connect ▾

Build ▾

Settings ▾

Help ▾

youoo

Screen1 ▾

Add

AI Companion

Emulator

USB

Refresh Companion Screen

Reset Connection

Hard Reset

Palette

Search Components...

User Interface

Button



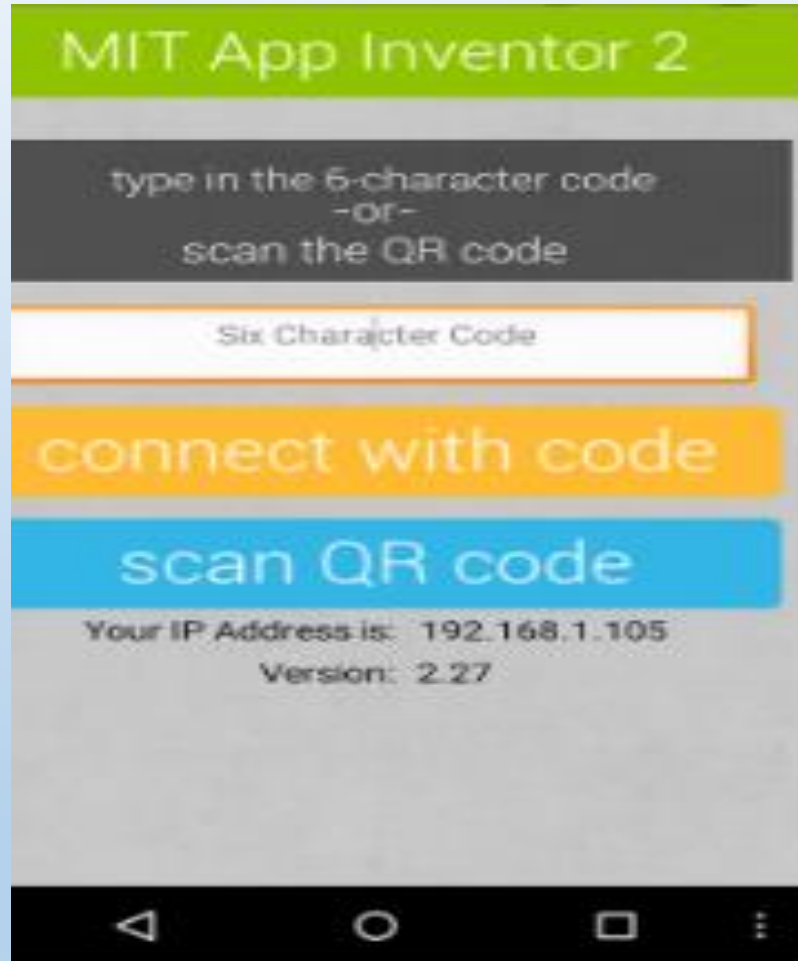
Viewer

Components in Viewer

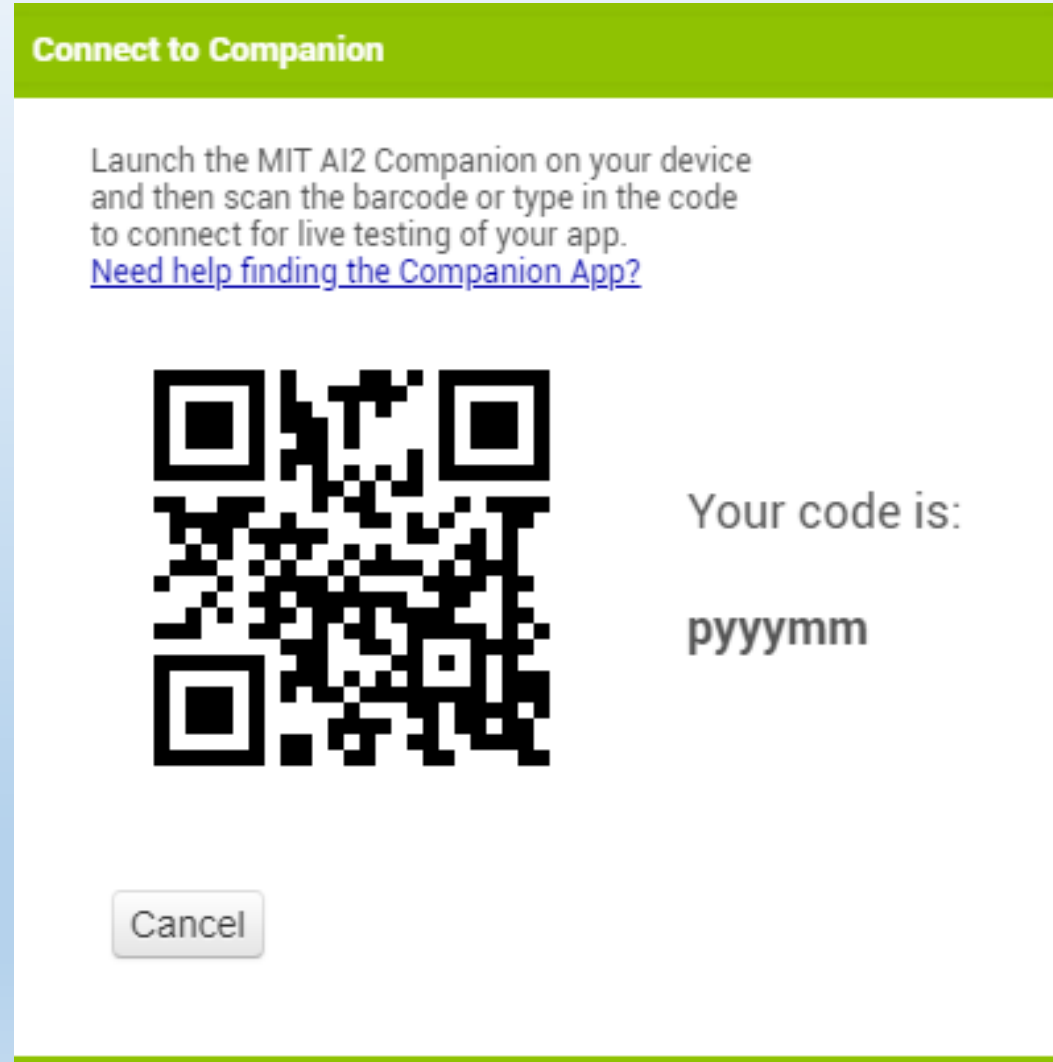
❑ Install MIT AI2 Companion from the Play Store

The image shows a screenshot of the Google Play Store interface. At the top, the Google Play logo is on the left, and a search bar is on the right. Below the logo, the word "Apps" is highlighted in a green bar. To the right of "Apps", there are navigation options: "Categories" with a dropdown arrow, "Home", "Top charts", and "New releases". On the left side, there is a vertical menu with options: "My apps", "Shop", "Games", "Family", "Editors' Choice", "Account", "Payment methods", "My subscriptions", "Redeem", "My wishlist", "My Play activity", and "Parent Guide". The main content area displays the app "MIT AI2 Companion" by "MIT Center for Mobile Learning" in the "Education" category. It has a 4.5-star rating from 21,928 reviews and is rated "PEGI 3". A green "Installed" button is visible. Below the app title, there are three screenshots of the app's interface. The first screenshot shows the app's main screen with a green header "MIT App Inventor 2", a grey box with the text "type in the 6-digit code -or- scan the QR code", a white input field labeled "Six Digit Code", an orange button "connect with code", and a blue button "scan QR code". At the bottom, it shows "Your IP Address is: 128.31.34.195" and "Version: 2.07nb7zx1". The second screenshot shows a close-up of a ginger cat's face with the text "Pet the Kitty!" and "This is a screen shot of the HelloPurr Tutorial loaded via the MIT AI2 Companion". The third screenshot is similar to the first, showing the same interface but with "Your IP Address is: 192.168.1.107" at the bottom.

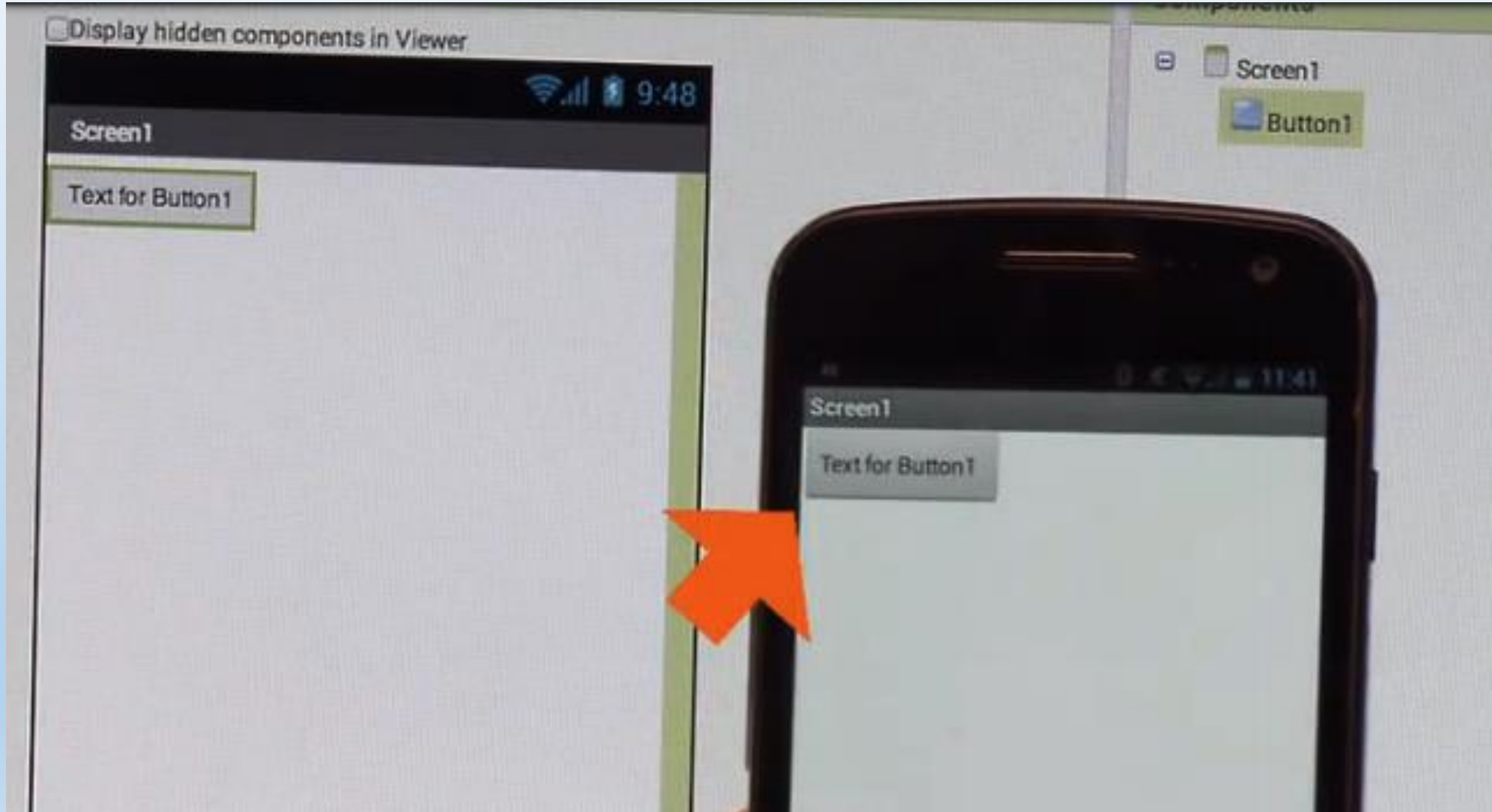
- ❑ Open AI2 Companion on your device



- ❑ Scan or type the QR code it into your Companion app



- ❑ See your app on your device.



❑ Add a Text-to-Speech

Go to the Media drawer and drag out a TextToSpeech component. Drop it onto the Viewer.

The image displays a software development interface with three main panels:

- Palette:** A sidebar on the left containing various component categories. The 'Media' category is expanded, showing a list of components including Camcorder, Camera, ImagePicker, Player, Sound, SoundRecorder, SpeechRecognizer, **TextToSpeech** (highlighted), VideoPlayer, and YandexTranslate.
- Viewer:** A central mobile device simulation showing a screen with a button labeled 'Text for Button1'. Below the viewer, a 'Non-visible components' section contains a 'TextToSpeech1' component icon.
- Components and Properties:** A right-hand panel. The 'Components' section shows a tree view with 'Screen1' containing 'Button1' and 'TextToSpeech1'. The 'Properties' section for 'TextToSpeech1' includes dropdown menus for 'Country' (Default) and 'Language' (Default), and input fields for 'Pitch' (1.0) and 'SpeechRate' (1.0). At the bottom of this panel are 'Rename' and 'Delete' buttons.

□ The Blocks Editor

The Blocks Editor is where we specify how our program should behave.



- 1- Click on the Button1 drawer. Click and hold the when Button1.Click do block.
- 2- Click on the TextToSpeech drawer. Click and hold the call TextToSpeech1.Speak block.

The screenshot displays a visual programming environment with two main panels: 'Blocks' on the left and 'Viewer' on the right. The 'Blocks' panel is organized into categories: 'Built-in' (Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, Procedures), 'Screen1' (Button1, TextToSpeech1), and 'Any component'. The 'TextToSpeech1' block is highlighted. The 'Viewer' panel shows a script with a 'when Button1 .Click' block containing a 'do' block with a 'call TextToSpeech1 .Speak message' block. At the bottom of the viewer, there are warning and error indicators: a yellow warning icon with the number '1' and a red error icon with the number '0', along with a 'Show Warnings' button.

□ Specify what the app should say

The image shows a visual programming environment with two main panels: 'Blocks' on the left and 'Viewer' on the right.

Blocks Panel:

- Expanded category: Built-in
 - Control
 - Logic
 - Math
 - Text** (highlighted)
 - Lists
 - Dictionaries
 - Colors
 - Variables
 - Procedures
- Screen1
 - Button1
 - TextToSpeech1
- Any component

Viewer Panel:

The viewer displays a single code block with the following structure:

```
when Button1 .Click  
do call TextToSpeech1 .Speak  
message "Hello World"
```

At the bottom of the viewer, there are two warning indicators (a yellow triangle and a red X) both showing a count of 0, and a 'Show Warnings' button.

Accelerometer Sensor

The image shows the Android Studio IDE interface with the AccelerometerSensor component selected. The interface is divided into several panels:

- Palette:** A search bar at the top is followed by categories: User Interface, Layout, Media, Drawing and Animation, Maps, and Sensors. Under the Sensors category, AccelerometerSensor is highlighted.
- Components:** A tree view showing the hierarchy of components on the screen: Screen1, Button1, TextToSpeech1, and AccelerometerSensor1 (highlighted).
- Properties:** A panel for AccelerometerSensor1 with the following settings:
 - Enabled:
 - LegacyMode:
 - MinimumInterval: 400
 - Sensitivity: moderate
- Non-visible components:** A panel at the bottom showing TextToSpeech1 and AccelerometerSensor1 (highlighted).

1- Click the AccelerometerSensor1 and Drag out the when accelerometerSensor1.Shaking do block.

2- Copy and paste the blocks that are currently inside the when Button1 then change the text to whatever you want.

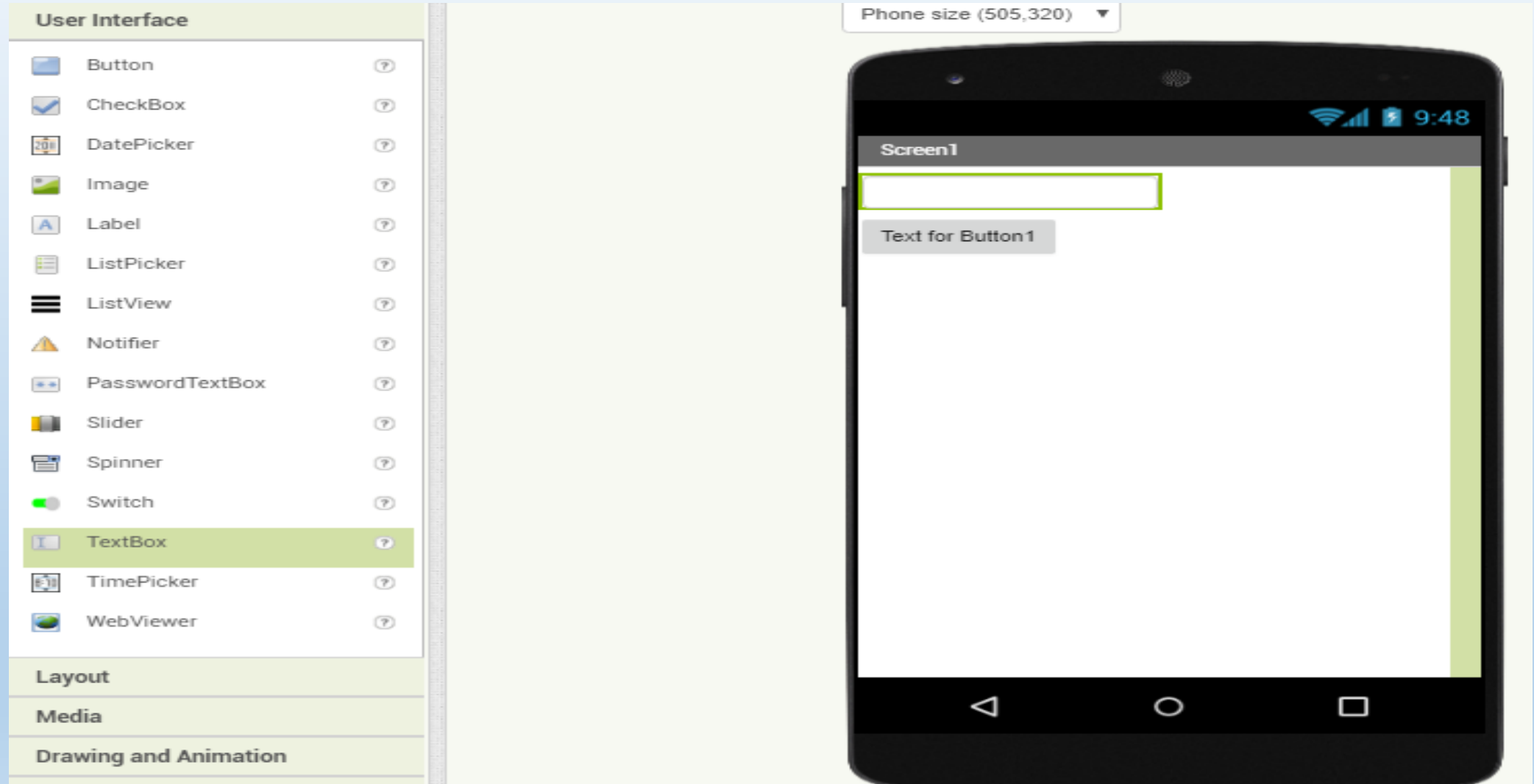
The screenshot displays a visual programming environment with two panes: 'Blocks' on the left and 'Viewer' on the right. The 'Blocks' pane shows a hierarchy of components including 'Screen1', 'Button1', 'TextToSpeech1', and 'AccelerometerSensor1'. The 'Viewer' pane shows two event-driven blocks:

- The top block is a 'when Button1 .Click' block containing a 'do' block with a 'call TextToSpeech1 .Speak' block and a 'message' block containing the text 'Hello World'.
- The bottom block is a 'when AccelerometerSensor1 .Shaking' block containing a 'do' block with a 'call TextToSpeech1 .Speak' block and a 'message' block containing the text 'Stop shaking'.

At the bottom of the Viewer pane, there are two warning icons (a yellow triangle with an exclamation mark and a red circle with an X) and a 'Show Warnings' button.

□ Text Box

Drag the text then click on Blocks



□ Get the text property of the TextBox1.

The green blocks in the TextBox1 drawer are the "getters" and "setters" for the TextBox1 component. Replace the getter box with the text message.

The screenshot displays a visual programming environment with two main panels: 'Blocks' on the left and 'Viewer' on the right.

Blocks Panel:

- Built-in:**
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Dictionaries
 - Colors
 - Variables
 - Procedures
- Screen1:**
 - TextBox1 (highlighted)
 - Button1
 - TextToSpeech1
 - AccelerometerSensor1
- Any component:**

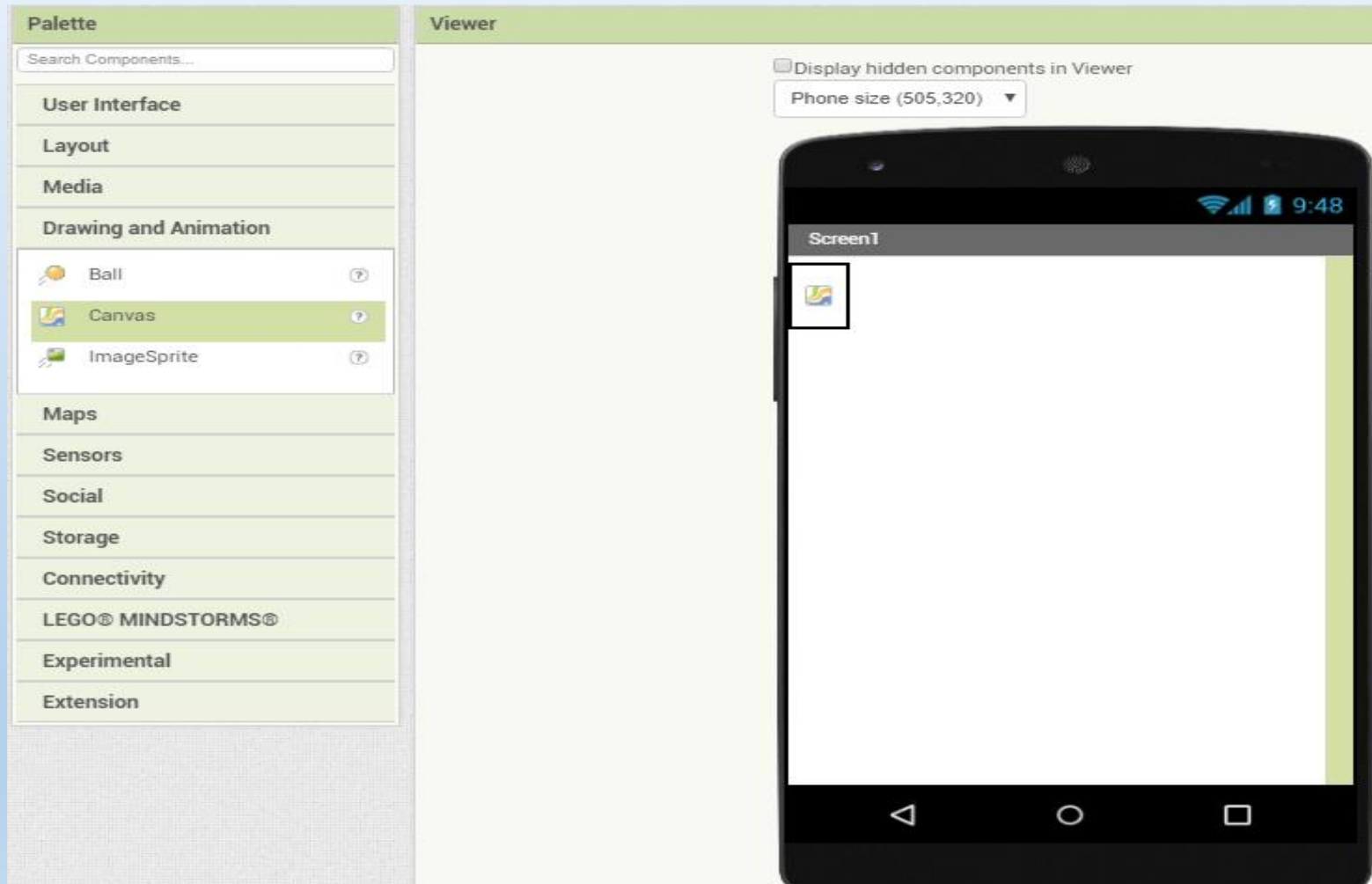
Viewer Panel:

- Block 1:** A 'when' block for 'Button1 .Click' containing a 'do' block with 'call TextToSpeech1 .Speak message' and a green 'TextBox1 . Text' block.
- Block 2:** A 'when' block for 'AccelerometerSensor1 .Shaking' containing a 'do' block with 'call TextToSpeech1 .Speak message' and a pink 'Stop shaking' block.

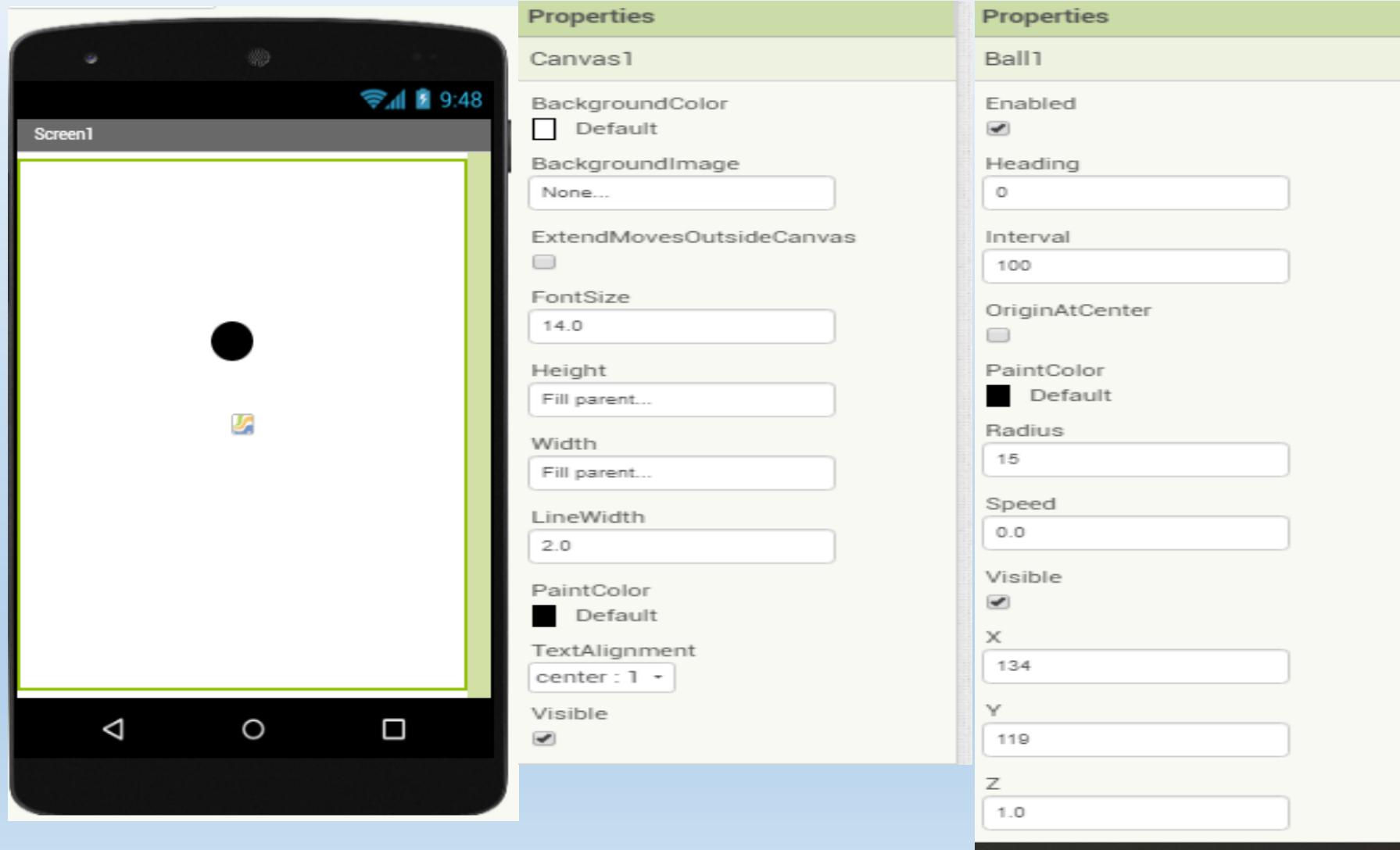
At the bottom of the Viewer panel, there are warning indicators (a yellow triangle with '0' and a red 'X' with '0') and a 'Show Warnings' button.

□ App2:Ball Bounce

From the Drawing and Animation drawer, drag out a Canvas component and drop it onto the viewer. Also uncheck scrollable so the screen doesn't scroll.



- 1- Set the Height property to "Fill Parent". Do the same with the Width property.
- 2- Drag out a Ball component and change its Radius property in the Properties pane.



The image shows a screenshot of the Android Studio interface. On the left is a mobile emulator displaying a screen with a black circle and a small icon. To the right are two 'Properties' panes. The first pane is for 'Canvas1' and the second is for 'Ball1'.

Canvas1 Properties:

- BackgroundColor: Default
- BackgroundImage: None...
- ExtendMovesOutsideCanvas:
- FontSize: 14.0
- Height: Fill parent...
- Width: Fill parent...
- LineWidth: 2.0
- PaintColor: Default
- TextAlignment: center : 1
- Visible:

Ball1 Properties:

- Enabled:
- Heading: 0
- Interval: 100
- OriginAtCenter:
- PaintColor: Default
- Radius: 15
- Speed: 0.0
- Visible:
- X: 134
- Y: 119
- Z: 1.0

- ❑ **Open the Ball1 Drawer.**
Drag out the Flung Event Handler

The image shows a software interface with two main panels: 'Blocks' and 'Viewer'. The 'Blocks' panel on the left is organized into categories: 'Built-in' (Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, Procedures), 'Screen1', 'Canvas1', and 'Any component'. Under 'Canvas1', there is a sub-category 'Ball1' which contains a ball icon. The 'Viewer' panel on the right displays a script block. The script starts with a 'when' block set to 'Ball1' and '.Flung'. Below this are six input fields labeled 'x', 'y', 'speed', 'heading', 'xvel', and 'yvel'. The script ends with a 'do' block, which is currently empty.

❑ Set the ball speed and heading :

- 1- Click the ball blocks drag out set Ball1.Heading and set Ball1.Speed blocks.
- 2- Mouse over the "speed" parameter and get speed block and plug to ball1.Speed.
- 3- Mouse over the heading parameter and get heading block and plug it to ball1.Heading.



□ Add an Edge

- 1- Drag out a when Ball1.EdgeReached do event
- 2- Scroll down Drag out Ball.Bounce block
- 3- Mouse over the heading parameter and drag get edge

The screenshot displays a programming environment with a 'Blocks' panel on the left and a 'Viewer' panel on the right. The 'Blocks' panel shows a hierarchy of categories: Built-in (Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, Procedures), Screen1, Canvas1, and Ball1. The 'Viewer' panel shows the following code blocks:

```
when Ball1 .Flung
  x y speed heading xvel yvel
do
  set Ball1 . Speed to get heading
  set Ball1 . Heading to get heading

when Ball1 .EdgeReached
  edge
do
  call Ball1 .Bounce
  edge
```

At the bottom of the viewer, there are two warning icons: a yellow triangle with an exclamation mark and the number '1', and a red circle with an 'X' and the number '0'. Below these icons is a button labeled 'Show Warnings'.

Final step.

The blocks should look like this three blocks below

The image shows a programming environment with two main panels: 'Blocks' on the left and 'Viewer' on the right.

Blocks Panel: A sidebar containing various block categories. Under 'Built-in', there are categories for Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, and Procedures. Below these are project-specific components: Screen1, Canvas1, and Ball1 (highlighted with a magnifying glass icon). At the bottom is 'Any component'.

Viewer Panel: Displays the code for the 'Ball1' object. It contains two event-driven code blocks:

- when Ball1 . Flung** (yellow block):
 - Inputs: x, y, speed, heading, xvel, yvel
 - do** (green block):
 - set Ball1 . Speed to** (green block) with input: **get heading** (orange block)
 - set Ball1 . Heading to** (green block) with input: **get heading** (orange block)
- when Ball1 . EdgeReached** (yellow block):
 - Input: edge
 - do** (green block):
 - call Ball1 . Bounce** (purple block) with input: **edge** (orange block) and **get edge** (orange block)

□ App3: Drawing App

- 1 - Drag out a Canvas and uncheck scrollable so the screen doesn't scroll
- 2 - Set the Height property to "Fill Parent". Do the same with the Width.
- 3 - Change LineWidth if you like. Also drag Accelerometer Sensor.

The screenshot displays the Android Studio IDE with the following components:

- Palette:** A list of components including User Interface, Layout, Media, Drawing and Animation, Maps, and Sensors. The AccelerometerSensor is selected.
- Viewer:** A central area showing a smartphone mockup. A canvas is drawn on the screen, and an AccelerometerSensor component is placed on it. The status bar shows the time as 9:48. Below the mockup, a section for "Non-visible components" shows the AccelerometerSensor1.
- Components:** A tree view on the right showing the hierarchy: Screen1 > Canvas1 > AccelerometerSensor1.
- Properties:** A panel on the right showing the properties for the selected Canvas1 component. The properties include: BackgroundColor (Default), BackgroundImage (None...), ExtendMovesOutsideCanvas (unchecked), FontSize (14.0), Height (Fill parent...), Width (Fill parent...), LineWidth (10), PaintColor (Default), TextAlignment (center : 1), and Visible (checked).

□ Get a Canvas.Dragged event block

- 1- Pull out the when Canvas1.Dragged event
- 2- Scroll and pull out Call Canvas1.DrawLine

The screenshot shows a software development environment with two main panels: 'Blocks' on the left and 'Viewer' on the right.

Blocks Panel: This panel contains a hierarchical list of components. Under the 'Built-in' category, there are several sub-categories: Control (orange), Logic (green), Math (blue), Text (pink), Lists (light blue), Dictionaries (dark blue), Colors (grey), Variables (orange), and Procedures (purple). Below these, there are project-specific components: 'Screen1' (grey), 'Canvas1' (green), and 'Ball1' (orange). At the bottom is 'Any component' (blue).

Viewer Panel: This panel displays a code block that has been assembled. It starts with a 'when Canvas1 .Dragged' event block (orange). Below this event block is a 'do' block (purple) containing a 'call Canvas1 .DrawLine' block. The 'call' block has four input fields: 'x1', 'y1', 'x2', and 'y2'. The event block also has several input fields: 'startX', 'startY', 'prevX', 'prevY', 'currentX', 'currentY', and 'draggedAnySprite'.

□ Use the get and set blocks to fill in the values for the Draw Line block.

1- Each time that Dragged event block is called, it will draw a small line between the previous location (prevX, prevY) of the finger to the new location (currentX, currentY).

2- . Mouse over the parameters of the Canvas1.Dragged block to pull out the get blocks that you need.

```
when Canvas1 .Dragged
  do
    call Canvas1 .DrawLine
      x1: get prevX
      y1: get prevY
      x2: get currentX
      y2: get currentY
```

❑ Clear the screen.

The image shows a visual programming environment with two main panels: 'Blocks' on the left and 'Viewer' on the right.

Blocks Panel:

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Dictionaries
 - Colors
 - Variables
 - Procedures
- Screen1
 - Canvas1
 - AccelerometerSensor1
- Any component

Viewer Panel:

The viewer displays two event-driven code blocks:

```
when Canvas1 .Dragged
  startX startY prevX prevY currentX currentY draggedAnySprite
do
  call Canvas1 .DrawLine
    x1 get prevX
    y1 get prevY
    x2 get currentX
    y2 get currentY

when AccelerometerSensor1 .Shaking
do
  call Canvas1 .Clear
```

NEXT DAY

❖ Build Speech Recognition App:

- Speech recognition.
- Get the Gold.

THANK YOU!