

# MIT APP INVENTOR

DAY 3

# 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

## ❖ **Build Piano board & To Do list:**

Concept of procedures.

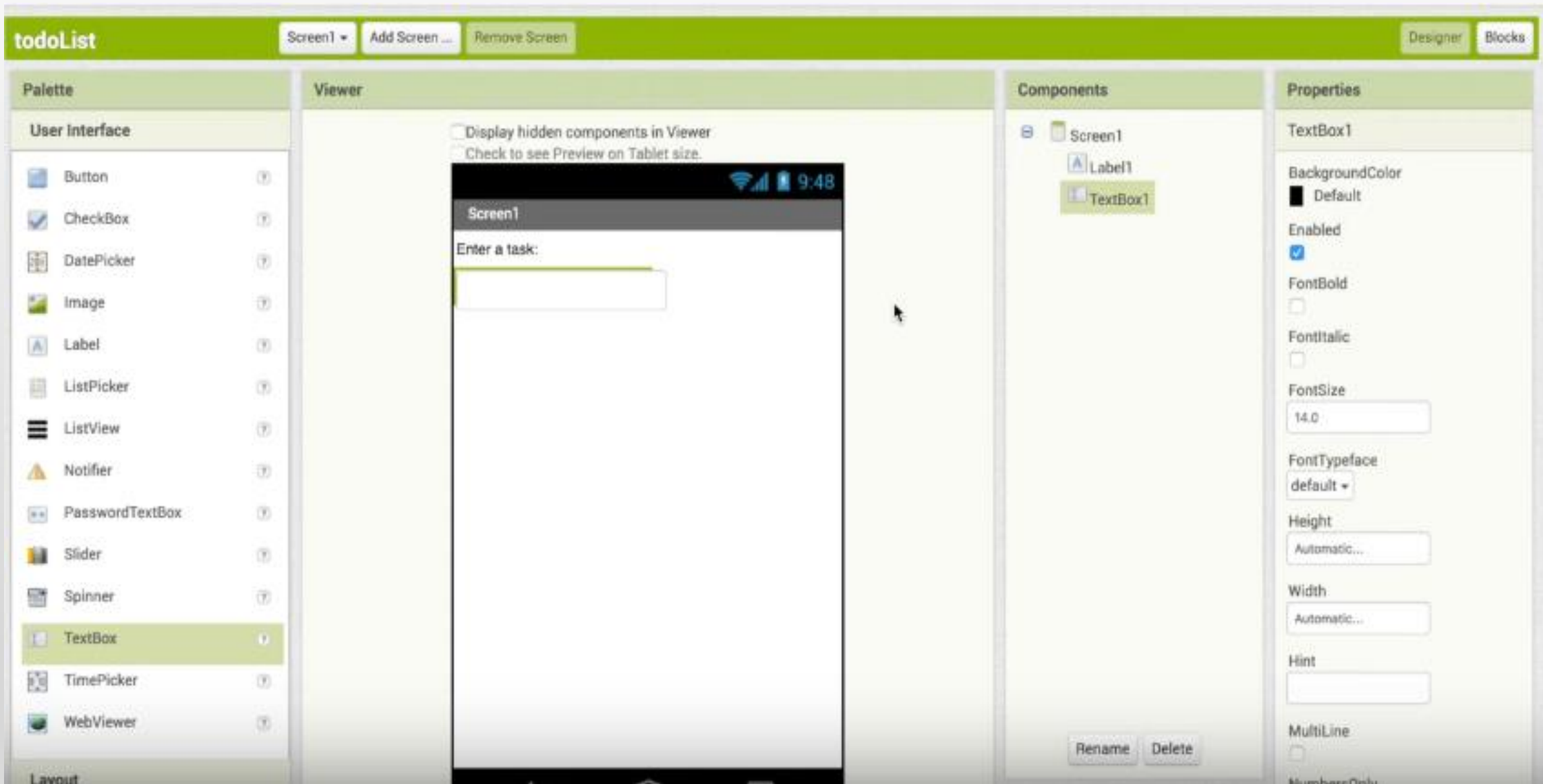
Build more advanced apps.

Build Piano board & To Do list app

## □ App 1 : Do list

The functionality of the app is to store list of things that you want to do.

- Drag a label to the screen to tell the user what to do .
- Rename the label .
- Pull out a text box from the Palette for the user to write on it.



## ❑ Add tinyDB

- Add button.
- Rename the button.
- Drag listview from user interface.
- Change its color if want.
- Click on storage from the Palette section and drag tinydb.


**todoList**    Screen1 ▾    Add Screen ...    Remove Screen    Designer    Blocks

**Palette**

- User Interface
- Layout
- Media
- Drawing and Animation
- Sensors
- Social
- Storage
  - File
  - FusiontablesControl
  - TinyDB**
  - TinyWebDB
- Connectivity
- LEGO® MINDSTORMS®
- Experimental

**Viewer**

Display hidden components in Viewer  
 Check to see Preview on Tablet size.



Screen1

Enter a task:

Submit

Screen1

**Components**

- Screen1
  - Label1
  - TextBox1
  - Button1
  - ListView1
  - TinyDB1**

**Properties**

TinyDB1

## ❑ Add task to task list

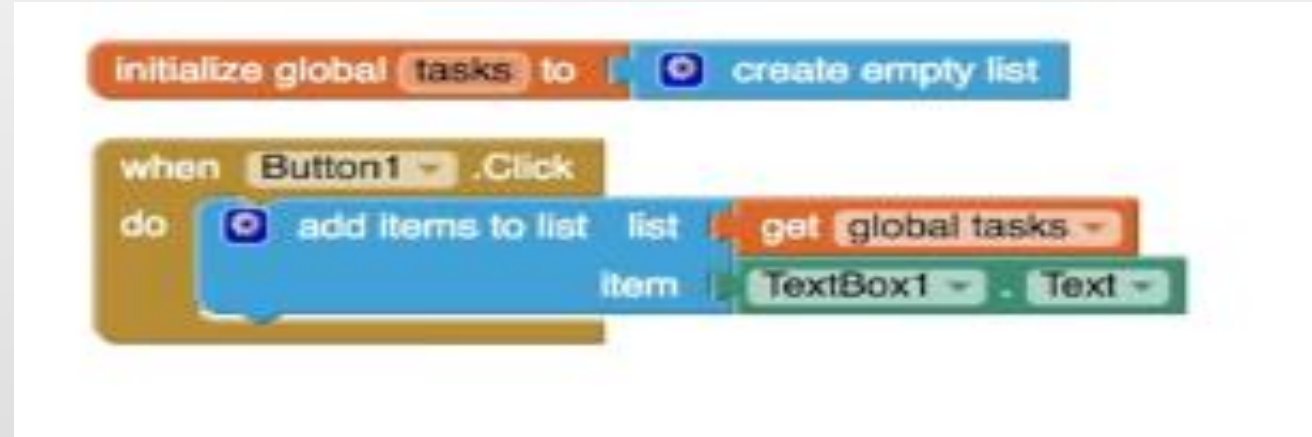
- Even though we have tinydb we need a variable to keep track of the tasks .
- Click variable, drag initialize global.
- Rename it .
- We need empty list, drag create empty list.





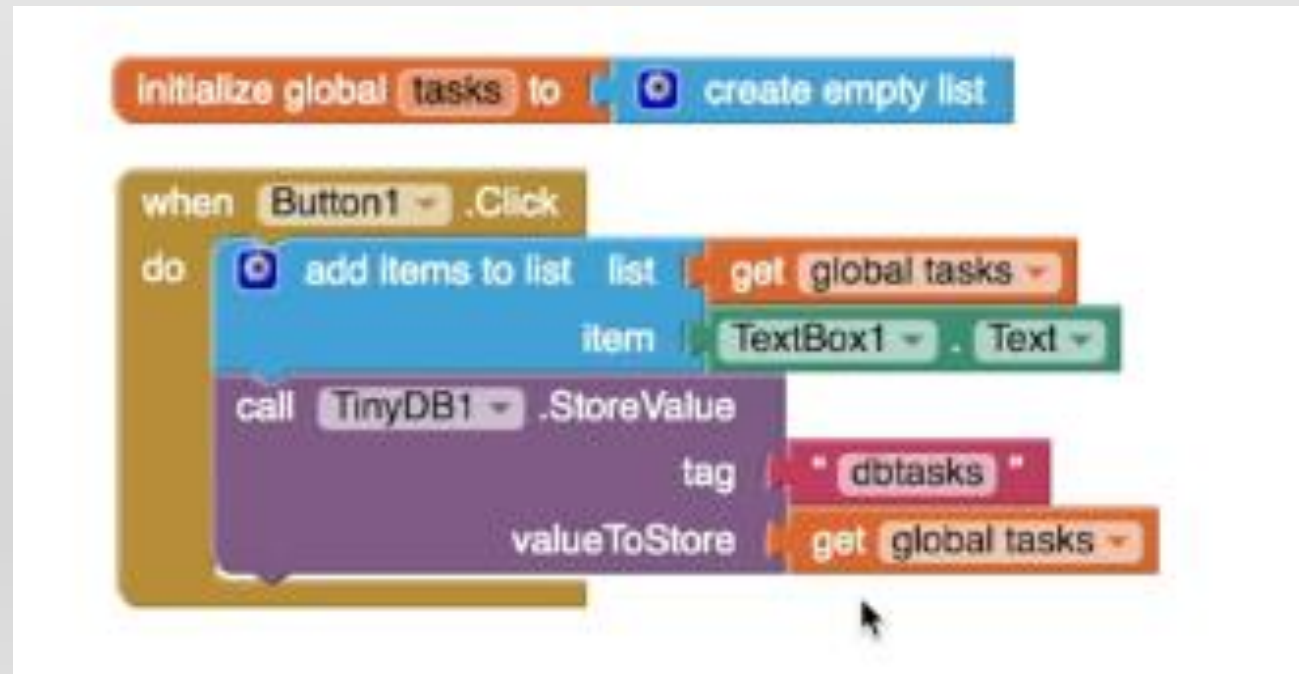
## ❑ Add the tasks to listview

- Drag event Handler.
- Click on list, drag add items.
- The lists will be tasks.
- Item is whatever you put in the text box.



## □ Update tinydb

- Drag call tinydb (it allows to store tinydb version of tasks)
- **Tag: It's a variable name that we'll be calling but we can retrieve it later.**
- Value to store: Task list



## □ Update listview

- Add event handler after the user added a list.
- Drag remove an list ,index : listview.selection
- Update tinydb with the new tasklist
- Update our listview.



```
initialize global tasks to create empty list

when Button1 .Click
do
  add items to list list get global tasks
  item TextBox1 . Text
  call TinyDB1 .StoreValue
  tag "dbtasks"
  valueToStore get global tasks
  set ListView1 . Elements to get global tasks
```

```
when ListView1 .AfterPicking
do
  remove list item list get global tasks
  index ListView1 . SelectionIndex
  call TinyDB1 .StoreValue
  tag "dbtasks"
  valueToStore get global tasks
  set ListView1 . Elements to get global tasks
```

## □ Last step

- The app should show what was the last list that have been added even after user got out of the app.
- Get an event handler from screen1.
- Hover above initialize global and drag set global tasks.
- Drag get tinydb value, tag:dbtasks , value: emptylist.
- Update listview so that the list that have been added before will appear in the list.

Initialize global `tasks` to `create empty list`



```
when Button1.Click
do
  add items to list list
    item TextBox1.Text
  call TinyDB1.StoreValue
    tag "dbtasks"
    valueToStore get global tasks
  set ListView1.Elements to get global tasks
```

```
when ListView1.AfterPicking
do
  remove list item list
    index ListView1.SelectedIndex
  call TinyDB1.StoreValue
    tag "dbtasks"
    valueToStore get global tasks
  set ListView1.Elements to get global tasks
```

```
when Screen1.Initialize
do
  set global tasks to call TinyDB1.GetValue
    tag "dbtasks"
    valueIfTagNotThere create empty list
  set ListView1.Elements to get global tasks
```

## □ App 2 : Piano board

- Download the files to your computer and import to MIT app inventor.

[https://drive.google.com/file/d/1XLD0BCxk\\_xut5uRKaQjvDzGLh8mFhGWy/view](https://drive.google.com/file/d/1XLD0BCxk_xut5uRKaQjvDzGLh8mFhGWy/view)

- Look for HorizontalArrangement1 in Components and change the height to 50 percent and width choose fill parent.
- To make this work we need buttons, so put 8 buttons and change their height: fill parent and width :10 percent .
- Change colors and rename the buttons same as media name .

Display hidden components in Viewer






## □ Add player

- Drag a button and a label to record the notes.
- Rename button for now and we'll do the label later .
- Drag player from media.

Viewer

Display hidden components in Viewer.



Screen1

C D E F G A B High C

Clear Notes

Non-visible components

NotePlayer

Components

- Screen1
  - HorizontalArrangement1
    - CNote
    - DNote
    - ENote
    - FNote
    - GNote
    - ANote
    - BNote
    - HighCNote
  - NotesLabel
  - ClearButton
  - NotePlayer

Rename Delete

## ❑ Blocks editor

- Lest start on the first note .
- When button is clicked sound must come out, player be informed what sound to play.
- Go to noteplayer and make the note start.
- The app must tell what note is playing.
- Do the same thing to for another note but change the note the player will play.

Blocks

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

- HorizontalArrangeme
  - CNote
  - DNote
  - ENote
  - FNote
  - GNote
  - ANote

Rename Delete

Viewer

```
when CNote Click
do
  set NotePlayer Source to CNote.wav
  call NotePlayer Start
  set NotesLabel Text to C

when DNote Click
do
  set NotePlayer Source to DNote.wav
  call NotePlayer Start
  set NotesLabel Text to D
```

Show Warnings

## □ Add the rest of the notes

- Drag procedure to add rest of notes.
- Instead of copy and paste, you can just call procedure to play the note.
- Now since the Cnote is empty, you can fill with call the procedure to play the note and choose what note you want to call.
- Do the same with the rest of 8 notes.
- Test your outcome.

```
when CNote .Click
do call PlayNote
   note "C"
```

```
when GNote .Click
do call PlayNote
   note "G"
```

```
to PlayNote note
do set NotePlayer .Source to join
   get note
   "Note.wav"
   call NotePlayer .Start
   set NotesLabel .Text to get note
```

```
when DNote .Click
do call PlayNote
   note "D"
```

```
when ANote .Click
do call PlayNote
   note "A"
```

```
when ENote .Click
do call PlayNote
   note "E"
```

```
when BNote .Click
do call PlayNote
   note "B"
```

```
when FNote .Click
do call PlayNote
   note "F"
```

```
when HighCNote .Click
do call PlayNote
   note "HighC"
```



## Clear and list notes

- You need to update the list using join block.
- Drag the event handler form clearnote .
- Add an empty liable to clear out the notes .
- Test the app .

```
when CNote .Click
do call PlayNote .
  note "C"
```

```
when GNote .Click
do call PlayNote .
  note "G"
```

```
when DNote .Click
do call PlayNote .
  note "D"
```

```
when ANote .Click
do call PlayNote .
  note "A"
```

```
when ENote .Click
do call PlayNote .
  note "E"
```

```
when BNote .Click
do call PlayNote .
  note "B"
```

```
when FNote .Click
do call PlayNote .
  note "F"
```

```
when HighCNote .Click
do call PlayNote .
  note "HighC"
```

```
to PlayNote note
do set NotePlayer . Source to join get note .
  "Note.wav"
  call NotePlayer . Start
  set NotesLabel . Text to join NotesLabel . Text .
  " "
  get note .
```

```
when ClearButton .Click
do set NotesLabel . Text to ""
```





# NEXT DAY

## ❖ Build two App games:

- Multiplication quiz.
- Mini Golf.

**THANK YOU!**