Contents

Setting up App Inventor ...................................................................................................................... 3
Creating the Interface: Main Menu ...................................................................................................... 4
Creating the Interface: Populate Quiz ................................................................................................ 6
Creating the Interface: Take Quiz ......................................................................................................... 8
Creating the Interface: Leader Board .................................................................................................... 10
Creating the Functionality: Main Menu ............................................................................................... 11
Creating the Functionality: Populate Quiz .......................................................................................... 13
Creating the Functionality: Take Quiz ................................................................................................. 17
Creating the Functionality: Leader Board ........................................................................................... 23
Bonus Activities ..................................................................................................................................... 24
Resources ............................................................................................................................................... 25
Setting up App Inventor

To start developing our app we need to open App Inventor and create a new project.

1. Go to the App Inventor website:

   [http://appinventor.mit.edu/explore/](http://appinventor.mit.edu/explore/)

2. Select the Create button in the top right corner of the page:

   ![Create button](image)

   This will open a new page and ask you to sign in with your Google account. If you don’t have a Google account, you will need to create one. To do this, go to [www.google.com.au](http://www.google.com.au) and select the Sign In option in the top right corner, then select the Create Account option.

   Once logged in, you will see the main App Inventor 2 account page.

3. Select the New Project... button near the top left corner of the page:

   ![New Project button](image)

4. Give your project a name, (e.g. Quiz) and press OK.

   This will open the application development section. This section is where you design the layout (Designer) for your app and make it functional (Blocks).
Creating the Interface: Main Menu

Our app starts as a completely blank screen. We will be adding a new screen that will be the *Main Menu* screen.

1. Select the **Add Screen**... button at the top of the screen and name the new screen **MainMenu**.

![Add Screen button](Image)

The *Designer* section will now be opened with a blank screen. On the left is the palette of user interface elements and some background functionality elements.

* Spend a small amount of time clicking through each of these sections and explore what’s available.

2. We will be adding three options in our main menu: Take Quiz, Populate Quiz and the Leader Board. First, we need to add a container to hold all of our elements. Select **Layout > VerticalArrangement** and drag this onto your screen. We will make our container to fill the width of the parent and be 400 pixels in height. The Properties should look like:

![VerticalArrangement Properties](Image)

3. The main elements of our main menu will be an image and three buttons.

Select **User Interface > Image** and drag this onto your container. Find an image that you like (e.g. using Google Image Search) and set this to be the **Picture** in the image. Set the Width and Height to 200 pixels. The Properties should look like:

![Image Properties](Image)
To add the buttons, select **User Interface > Button** and drag this underneath the image. Do this until you have **three** buttons. Rename the first button `btnTakeQuiz`, the second button `btnPopulateQuiz` and the third button `btnLeaderBoard`. For each of these buttons set the **Shape** to `rounded`, the **Text** to `Take Quiz`, `Populate Quiz` or `Leader Board`, and the **Width** to `Fill parent...`.

Your screen should now look similar to:
Creating the Interface: Populate Quiz

This screen is where we will be creating the quiz questions. To do this we will need two databases for storing the questions and answers and another database to store the quiz parameters (for e.g. the number of questions stored). We will also need elements to enter the question and answer and a way to submit the question.

1. Create a new screen by pressing the Add Screen… button and name this screen PopulateQuiz.

2. Select Layout > VerticalArrangement in the Palette and drag this onto your blank screen. This will hold all of the interface elements. Change the Width property to Fill Parent… and the Height property to 400 pixels… . The properties should look like:

3. Select Layout > TableArrangement in the Palette and drag this into the VerticalArrangement. This is where we will be putting the question and answer interface elements. Keep the Columns and Rows properties at 2 and change the Width property to Fill Parent… and the Height property to 200 pixels…. The Properties should look like:

Select User Interface > Label and drag this into the first and third cell of the TableArrangement. Change the Text property to ‘Question:’ and ‘Answer:’.

Select User Interface > TextBox and drag this into the second and fourth cell of the TableArrangement. Rename each of these TextBoxes to txtPQ_Q and txtPQ_A and change the Width property to 200 pixels....
Select **User Interface > Button** and drag two buttons into the VerticalArrangement layout, underneath the TableLayout. Rename these to *btnSubmitQ* and *btnDone* and the **Text** property to ‘Submit’ and ‘Done’. Set the **Shape** property to *rounded* and **Width** to *Fill Parent*… .

Your screen should look similar to:

![Image of a screen with buttons and a TableLayout]

4. We will need databases to store the questions, answers and application parameters. We will be using TinyDB to do this. TinyDB are databases that store data locally, whereas TinyWebDB store data on a server on the internet.

Select **Storage > TinyDB** and drag this onto your screen. You will notice the icon disappear and reappear underneath your screen. Do this until you have three databases. Rename the databases to *DBQuestions, DBAnswers* and *DBParameters*.

Your screen should now look similar to:

![Image of a screen with databases and non-visible components]
Creating the Interface: Take Quiz

This screen is where we will be taking the quiz. To do this we will need to access the databases created in the previous section. We will also need elements to display the question, to allow us to type out the answer and to submit the answer.

1. Select Screen1 from the Screen dropdown.

2. Select Layout > VerticalArrangement in the Palette and drag this onto your blank screen. This will hold all of the interface elements. Change the Width property to Fill Parent... and the Height property to 400 pixels... .

3. Select Layout > Label and drag this onto the VerticalArrangement. Rename this to 'lblQ' and set the Width property to Fill Parent... and the Text to Question.

4. Select Layout > TextBox and drag this underneath the Label in the VerticalArrangement. Rename this to 'txtA' and set the Width property to Fill Parent... .

5. Select Layout > Button and drag this underneath the TextBox in the VerticalArrangement. Rename this to 'btnSubmit' and set the Width property to Fill Parent..., the Shape property to rounded and the Text property to Submit.

6. We will be displaying a correct (tick) or incorrect (cross) image as the result of the question. We want to centre the image. To do this, select Layout > HorizontalArrangement and drag this underneath the Button in the VerticalArrangement. Set the AlignHorizontal property to Center, the AlignVertical property to Center, and the Width property to Fill Parent... .

   Select User Interface > Image and drag this into your HorizontalArrangement. Rename this to 'imgResult'. Find an image that you like for ‘correct’ and ‘incorrect’ (e.g. using Google Image Search) and upload this in the Media section. For example, I have used:

   ![Correct Image](image1.png) ![Incorrect Image](image2.png)

   Make sure ‘None...’ is selected in the Picture property for the image.

7. Select UserInterface > Button and drag this underneath the HorizontalArrangement in the VerticalArrangement. Rename this to ‘btnNextQ’ and set the Width property to Fill Parent... , the Shape property to rounded and the Text property to Next.

8. We need a way to quit the quiz if the user does not want to finish it. Select UserInterface > Button and drag this underneath the VerticalArrangement. Rename this to ‘btnQuit’. Set the Width property to Fill Parent... , the Shape property to rounded and the Text property to ‘Quit’. Set the Visible
9. To share the databases with the PopulateQuiz screen, we need to create TinyDB elements with the same names. Select Storage > TinyDB and drag three of these onto your screen. Rename these to ‘DBQuestions’, ‘DBAnswers’, and ‘DBParameters’.

10. We will be using a notifier to tell the user their final score. Select UserInterface > Notifier and drag this onto your screen. Rename this to ‘notifier’.

Your screen should look similar to:
Creating the Interface: Leader Board

This section is where we will display the score.

1. Create a new screen by pressing the Add Screen... button and name this screen LeaderBoard.

2. Select Layout > VerticalArrangement in the Palette and drag this onto your blank screen. This will hold all of the interface elements. Change the Width property to Fill Parent... and the Height property to 400 pixels...

3. Select User Interface > Label in the Palette and drag this onto the VerticalArrangement. Change the Text property to ‘Leader Board’, tick the FontBold property and change the FontSize property to 18.0.

4. Select Layout > TableArrangement in the Palette and drag this onto the VerticalArrangement. This will hold the name and score labels. Change the Width property to Fill Parent... and change the Rows property to 1.

5. Select User Interface > Label in the Palette and drag one into the first and second cell in the TableArrangement. Rename the first label to lblName and the second label to lblScore. Set the Width property of lblName to 150 pixels...

6. Select User Interface > Button in the Palette and drag this into the VerticalArrangement, underneath the TableArrangement. Rename this to btnBack, change the Text property to ‘<< Back’, the Shape property to rounded and the Width property to Fill Parent... .

7. To get the score from the quiz section, we will be sharing the parameters database. Select Storage > TinyDB in the Palette and drag this into your screen. Rename this to DBParameters.

Your screen should look similar to:
Creating the Functionality: Main Menu

In this section we will be making the main menu functional. To do this we need to change to the Blocks section. Select Blocks from the top right corner of the page:

On the left in Blocks you will see a list of built-in sections (e.g. Control, Logic, Math, etc) and the elements that you created for your interface (e.g. btnTakeQuiz, btnPopulateQuiz, etc). These sections are where you will find all of the programming options.

Take some time to click through the different sections and see what’s available.

1. We want to open the screen for taking the quiz when the user presses the ‘Take Quiz’ button. To do this, select the btnTakeQuiz > when btnTakeQuiz.Click block and drag this onto the viewer:

To open another screen, select the Control > open another screen screenName block and drag this in the btnTakeQuiz.click block:

We need to tell the block the name of the screen to open. Select the Text > “ “ block and place this at the end of the previous block.

Type ‘Screen1’ into the spot between the quotation marks. Your block should now look like:
2. Repeat this for the other two buttons. We want to open the PopulateQuiz screen when the btnPopulateQuiz button is pressed and the LeaderBoard screen when the btnLeaderBoard button is pressed.

Your Viewer should now look like:

```
when [btnPopulateQuiz] Click
  do [open another screen screenName = "PopulateQuiz"]

when [btnTakeQuiz] Click
  do [open another screen screenName = "Screen1"]

when [btnLeaderBoard] Click
  do [open another screen screenName = "LeaderBoard"]
```
Creating the Functionality: Populate Quiz

In this section we will be creating the functionality for populating the quiz questions. We will be storing the questions in the question database and the corresponding answers in the answer database. We will also be keeping track of how many questions were entered.

1. Select the PopulateQuiz screen from the screen dropdown list and ensure you are in the Blocks mode:

2. To keep track of the number of questions we need to create a variable. This will store the number of questions. Select Variables > initialize global name to from the Blocks list and drag this onto your blank viewer. We will be calling this variable questionCounter and setting it to 0. You will find the number option in the Math section. Your block should look like:

   ![Initialize global questionCounter to 0]

3. We want to store the question in the question database and the answer in the answer database when the user clicks the Submit button. Select the btnSubmitQ > when btnSubmitQ.Click block and drag this onto the Viewer.

   We will be putting all of the blocks for storing the questions and answers inside this block.

4. To store the question select the DBQuestions > call DBQuestions.StoreValue block and place this inside the button block:

   ![StoreValue block]

   We will need to give the question a unique tag. This identifies the question for later use. Our tags will be ‘Q1’, ‘Q2’, etc, for each added question. To do this we need to join the Q with the current question number. Select the Text > join block and attach this to the tag area in the database block:

   ![Join block]

   Select Text > “ ” and drag this into the first join slot. Type in Q. Select the Variables > get name block and drag this into the second join slot. Select global questionCounter from the dropdown menu in the get name block. Your block should look like:
We now need to store the question that was typed in the text box. Select the `txtPQ_Q > txtPQ_Q.Text` block and drag this into the `valueToStore` slot in the database block:

5. Repeat the same procedure for storing the answer in the `answer` database. Your block should look like:

   ![Diagram of storing answer in database]

Remember to select the `DBAnswers` database for the answers, and use an A in the tag!

6. Since we have stored a question, we need to increment the counter. To do this we will be setting the counter to itself + 1. Select the `Variables > set name to` block and drag this underneath the database block in the click block. Set the `name` to `global questionCounter`. To add 1 to the counter select the `Math > [ ] + [ ]` block and drag this into the `set global questionCounter to` block. This should now look like:

   ![Diagram of incrementing question counter]

Select the `Variables > get name` block and drag this into the first blank square in the addition block. Select `global questionCounter` as the name. Select the `Math > 0` block and drag this into the second addition block. Change the value to 1. Your block should look like:
7. The final step in this block is to clear the question and answer text boxes. Use the set text and Text blocks to do this.

Your final block should look like:

8. The final block in this section is for the Done button. This will be selected when the user is finished populating the questions and answers and will redirect the user to the main menu.

Select the btnDone > when btnDone.Click block and drag this into the Viewer.

Before we redirect the user, we need to store the number of questions. We will be storing this in the parameters database. Select the DBParameters > call DBParameters.StoreValue block and drag this into the button click block. This should look like:

Set the tag to ‘NoQs’ and the value to the global questionCounter variable:

The final step is to redirect the user back to the main screen. Select the Control > open another screen screenName block and drag this into the btnDone.Click block, under the database block. Set the screenName to “MainMenu” by using the Text > “ ” block. The final block should look like:
9. We have now finished the functionality for the PopulateQuiz screen!

Your final blocks for this screen should look like:

![Diagram of PopulateQuiz block logic]

You can now put some questions and answers in your quiz!
Creating the Functionality: Take Quiz

In this section we will be creating the functionality for taking the quiz. We will be getting a random question from the question database and checking the users answer to the answer stored in the answer database. We will also be keeping track of the users score.

1. Select Screen1 from the screen list and ensure that Blocks is selected:

![Screen1 selection]

2. We will be using three variables in this screen to store the score, the current question number and the number of questions seen. Create three variables and call them score, QNo and NoQuestions. Set score and QNo to 0 and NoQuestions to 1:

![Variable initialization]

If you are unsure where to find these blocks, ask a tutor!

3. Because we will be displaying the question when the screen opens and when we go to the next question, we will have duplicate code. Whenever this happens we should create a procedure that we can call later. Select Procedures > to procedure do block, drag this onto the Viewer and name it displayQuestion:

![Procedure creation]

To randomly select a question from the question database we need to generate a random number. Select the Variables > set name to block, drag this into the displayQuestion procedure and change the name to global QNo. We will be setting this to a random integer, found in the Math blocks section:

![Random integer selection]

The random number needs to be between 1 and the number of questions in the database. Previously, we set and stored this in the Parameters database.

Set the first block to 1 (found in Math). Set the second block to DBParameters > call DBParameters.GetValue with the tag ‘NoQs’ and the valueIfTagNotThere to ‘1’:
To set the question label to the random question, select the `lblQ > set lblQ.Text to` block and drag this underneath the `set global QNo to` block in the procedure. Set this to the random Question by using the `call DBQuestions.GetValue` block.

The tag needs to be the random number we previously generated. Use the `join` block in the Text section with the global QNo variable. Your block should look like:

The final step in the procedure is to clear the image and the answer text. Use the `set ImageResult.Picture to` and the `set txtA.Text to` blocks to do this.

Your final procedure should look like:

4. We want the first question to appear when the screen appears. Select the `Screen1 > when Screen1.Initialize do` block and drag this into the Viewer.

We only want to display a question if the database has already been populated. To do this we will use an `if` control block. Select the `Control > if then else` block and drag this into the screen initialize block:

An if block checks a condition and if it is true, it runs the blocks in the `then` section, otherwise it runs the blocks in the `else` section. We want to check if the variable `NoQs` has been created and stored in the Parameters database (this should have been created when a question was added).
Select the Math > [≠] block and drag this next to the if. Select the DBParameters > call DBParameters.GetValue block and drag this into the first blank of the math block. Check for the NoQs tag and set the valueIfTagNotThere to 0. Compare this to 0:

If the statement is true, then there are questions in our database and we want to display the first one. We can do this by calling our procedure for displaying the question (found in Procedures > call displayQuestion)!

If the statement is not true, then our database has no questions in it and we want to notify the user and exit to the main menu. We will be using our notifier to do this. Select the notifier > call notifier.ShowMessageDialog message title buttonText block and drag this into the else section. Type out a message, title and button caption. Use the close screen block to exit to the main menu.

Your final block should look like:

5. We need to check whether the submitted answer is right or not. This will happen when the user clicks the Submit button. We will be displaying a tick if the answer is right or a cross if the answer is wrong.

Select the btnSubmit > when btnSubmit.Click block and drag this onto your viewer. First we will hide the keyboard and hide the submit button from the user. Use the call txtA.HideKeyboard block (found in txtA) to hide the keyboard and the set btnSubmit.Visible to block (found in btnSubmit) and the false block (found in Logic) to hide the submit button:

Use the blocks in the Text section to compare the text in txtA.Text to the answer in the DBAnswers database with the tag for the current QNo. This should look like:
If the answer is correct we want to add one to the score variable and set the image to the tick. If the answer is wrong we want to set the image to the cross.

Set the visibility of the btnNextQ button to true so the user can go to the next question.

Your final block should look like:

6. When the user presses the Next Question button, we need to check if we have reached the maximum number of questions (we will use 5 for this). If 5 questions have been answered, the user will be notified, the score will be stored and the screen will change to the main menu. If there are more questions, the next one will be displayed.

Use the if then block in the Control section to test whether 5 questions have been reached in the when btnNextQ.Click do block:

Use the notifier and Text blocks to notify the user of their score:
Use the **DBParameters** blocks to store the score and the **close screen** block in the **Control** section to send the user back to the main menu:

After the **if then** block, we want to ask another question and increment the question counter. Use our **displayQuestion** procedure and the **Math** blocks to add one to the **global NoQuestions** variable. Set the **btnSubmit** button visibility to **true** and the **btnNextQ** visibility to **false**.

Your final block should look like:

7. The final block in this screen will close the screen if the **Quit** button is pressed:
8. We have now finished the functionality for the Take Quiz screen!

Your final blocks for this screen should look like:
Creating the Functionality: Leader Board

In this section we will be creating the functionality for the LeaderBoard screen. This is where we will be displaying the score from the quiz.

1. We want to display the score when the screen initializes. Use the **when LeaderBoard.Initialize do** block in the **LeaderBoard** section:

```
when LeaderBoard.Initialize do
set lblName.Text to "Person"
set lblScore.Text to call DBParameters.GetValue(tag, "score")
```

Change the text of the **lblName** label to ‘Person’ and the text of the **lblScore** to the score stored in the Parameters database.

2. We want to go back to the main menu when the user presses the Back button. Use the **when btnBack.Click do** block in the **btnBack** section and the **close screen** block in the **Control** section:

```
when btnBack.Click do
    close screen
```

Your final screen should look like:
**Bonus Activities**

If you have finished the previous sections try some of the following activities:

1. Change the look of the screens in Designer mode until you are happy with the interface, colours, fonts and images. 🌟

2. Change the code so that only questions that haven’t been asked already are selected from the database 🌟🌟🌟

3a. Use a *Clock* to time how long it takes the user to finish the questions. 🌟🌟

   b. Add the clock time to the Leader Board. 🌟

4a. Create a screen where the user can save their name with their score. 🌟🌟🌟

   b. Alter the LeaderBoard screen to show multiple users (*hint: this may have to be done in the program with blocks*). 🌟🌟🌟🌟
Resources

There are many resources online for App Inventor. Most can be found on the MIT App Inventor website:

http://appinventor.mit.edu/explore/

You will find documents about references, tips and troubleshooting at:

The App Inventor team have taken time to integrate some of the tutorials with the curriculum. You will find these details at:

Note: this may be the curriculum for America, since the team are based in America. However, there still may be some interesting techniques that could be used in Australia.

There are a lot of tutorials for App Inventor, ranging from games to storage. These can be found at: