



Technovation iridescent
CANADA

App Inventor Workshop 2

Objectives

- ➔ Create a registration app
- ➔ Use lists to store data
- ➔ Use a web db to save your users
- ➔ Use maps to show where your users are

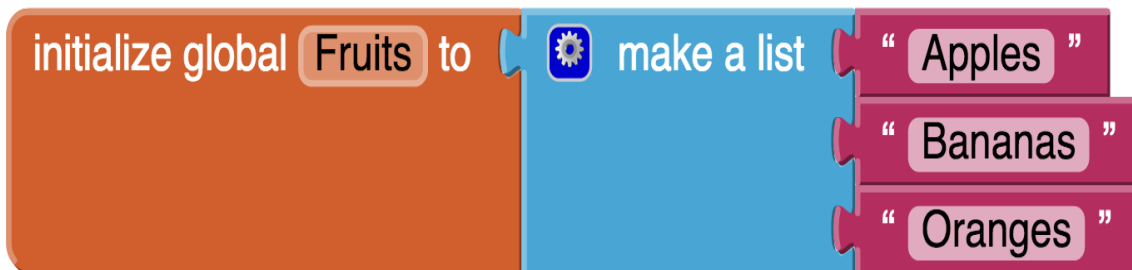
Types of Databases

Types of databases:

- [TinyDB](#) stores data directly on the phone and will only be used by that phone
 - Uses tag, value pairs to store the data
 - Tag to get the data back
- [TinyWebDB](#) stores data on a web database that can be shared among phones
 - Uses tag, value pairs to store the data
 - Tag to get the data back
- [FusionDB](#) stores data on a web database that can be retrieved
 - Uses columns and rows to store the data, like Excel
 - Flexible retrieval

Organizing data

- ➔ Lists can hold multiple pieces of data and they're easy to get data from.
- ➔ You may have made a to-do list or a grocery list before, and lists in programming are very similar.



List Name: Fruits

- Apples (Index = 1)
- Bananas (Index = 2)
- Oranges (Index = 3)

Using Lists in a Database

Non-visible components



TinyDB1

```
call TinyDB1 .StoreValue
  tag "Food"
  valueToStore get global Fruits
```

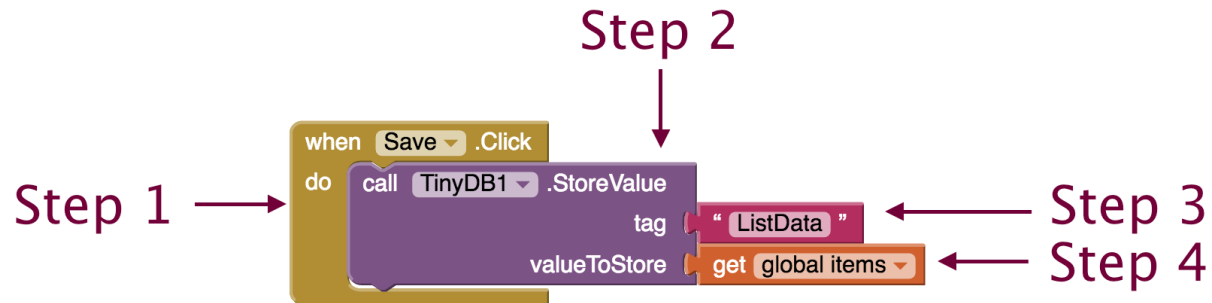
```
call TinyDB1 .GetValue
  tag "Food"
  valueIfTagNotThere ""
```

```
call TinyDB1 .StoreValue
  tag "MyAge"
  valueToStore 16
```

```
call TinyDB1 .StoreValue
  tag "Favorites"
  valueToStore make a list
    "Learn how to Code"
    "Visit Family"
    "Listen to Music"
```

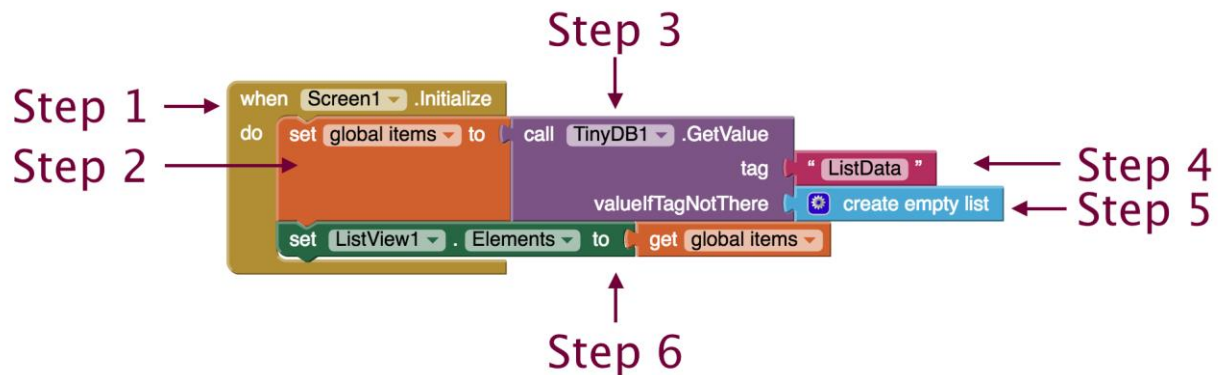
Saving the list for next time

1. Grab the when “when button.Click” block for your save button
2. Add the “call TinyDB1.StoreValue” block
3. Add a “tag” name
4. Add your *items* variable to “valueToStore”



Using a saved list

1. Grab the “when Screen1.Initialize” block
2. Grab a “set ‘variable name’ to” block and set the variable to be your *items* list
3. Call the database using “call TinyDB1.GetValue” block
4. Enter the tag name you used to save the list for the “tag”
5. Put the “create empty list” block for “valueIfTagNotThere”
6. To view your list “set ListView1.Elements to” block and attach it to your *items* var



TinyWebDB

- ➔ TinyWebDB is like TinyDB but in the cloud
- ➔ Default TinyWebDB is shared by everyone
- ➔ <https://appinvtinywebdb.appspot.com/>
- ➔ Good for developing and testing your app
- ➔ If you want your own TinyWebDB
 - <http://appinventor.mit.edu/explore/content/custom-tinywebdb-service.html>

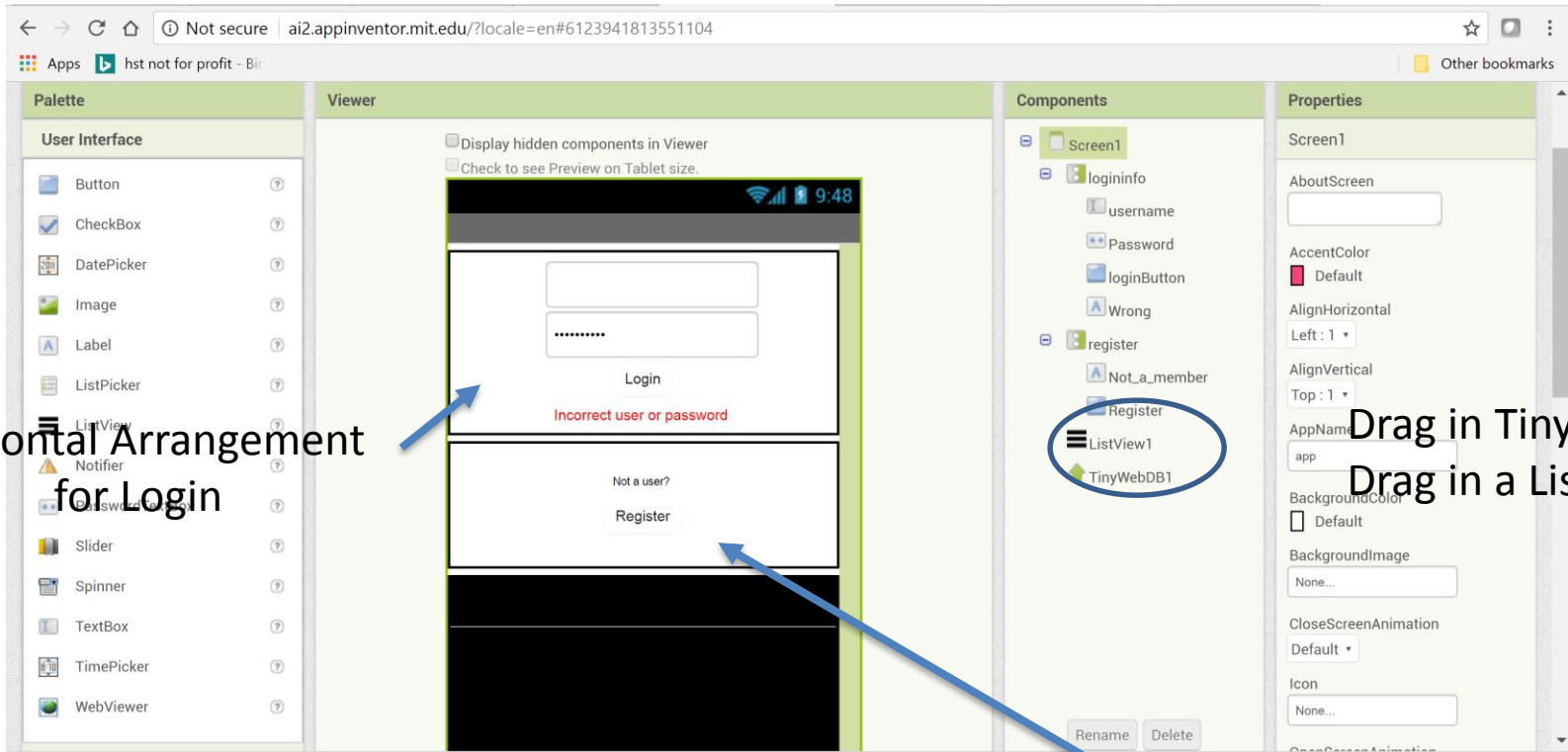
Building our app

- ➔ Register new users
- ➔ Login for existing users
- ➔ Map where the user is
 - Show directions from current location

Components we will use

- ➔ TinyWebDB to store our registered users
- ➔ List to store the information about our users
- ➔ Maps to show where our users are
 - Activity starter to call the google maps app
- ➔ Notifier to show messages

Creating a Login Screen



Horizontal Arrangement
for Login

Drag in TinyWebDB
Drag in a List Viewer

Horizontal Arrangement
for Register

Login blocks

The screenshot displays a software development environment for a project named "RegistrationExample". The interface is divided into several sections:

- Top Bar:** Contains the project name "RegistrationExample", a dropdown menu for "Screen1", and buttons for "Add Screen ..." and "Remove Screen". On the right, there are tabs for "Designer" and "Blocks".
- Blocks Panel (Left):** A sidebar showing a hierarchy of components. Under "Built-in", there are categories: Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under "Screen1", there is a "logininfo" component with sub-items "username" and "Password".
- Viewer (Center):** The main workspace showing a sequence of blocks:
 - Five "initialize global" blocks, each setting a variable to an empty string: "password", "firstname", "lastname", "address", and "email".
 - A "when Screen1 .Initialize" block containing a "do" block that sets "Wrong . Visible" to "false".
- Bottom Left:** A status bar showing "0" warnings and "0" errors, with a "Show Warnings" button.
- Right Side:** A vertical toolbar with icons for a folder, a target, a plus sign, a minus sign, and a trash can.

Initialize our variables and Start with the error message turned off

Logging the user in



If they are logging in, get the value from tinywebdb

Check that the value is a list, set our variables, and make sure they used the right password

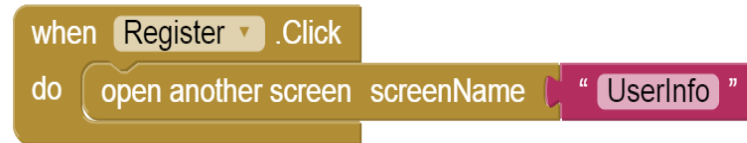
Show the list

```
when Register .Click
do
  open another screen screenName "UserInfo"

when loginButton .Click
do
  call TinyWebDB1 .GetValue
  tag username . Text

when TinyWebDB1 .GotValue
tagFromWebDB valueFromWebDB
do
  if is a list? thing get valueFromWebDB
  then
    set ListView1 . Elements to get valueFromWebDB
    set global email to username . Text
    set global password to select list item list get valueFromWebDB
                        index 1
    set global firstname to select list item list get valueFromWebDB
                        index 2
    set global lastname to select list item list get valueFromWebDB
                        index 3
    set global address to select list item list get valueFromWebDB
                        index 4
    if compare texts get global password = Password . Text
    then
      set ListView1 . Elements to get valueFromWebDB
    else
      set Wrong . Visible to true
  else
    set Wrong . Visible to true
```

Registering Users



- ➔ Call the UserInfo screen to register
- ➔ Decide what information you want from the user
- ➔ We will use a list to keep the information – the list will be the value we store
- ➔ We will use their email as the tag

MapMyUser

- ➔ Call the mapping screen passing in the user's email

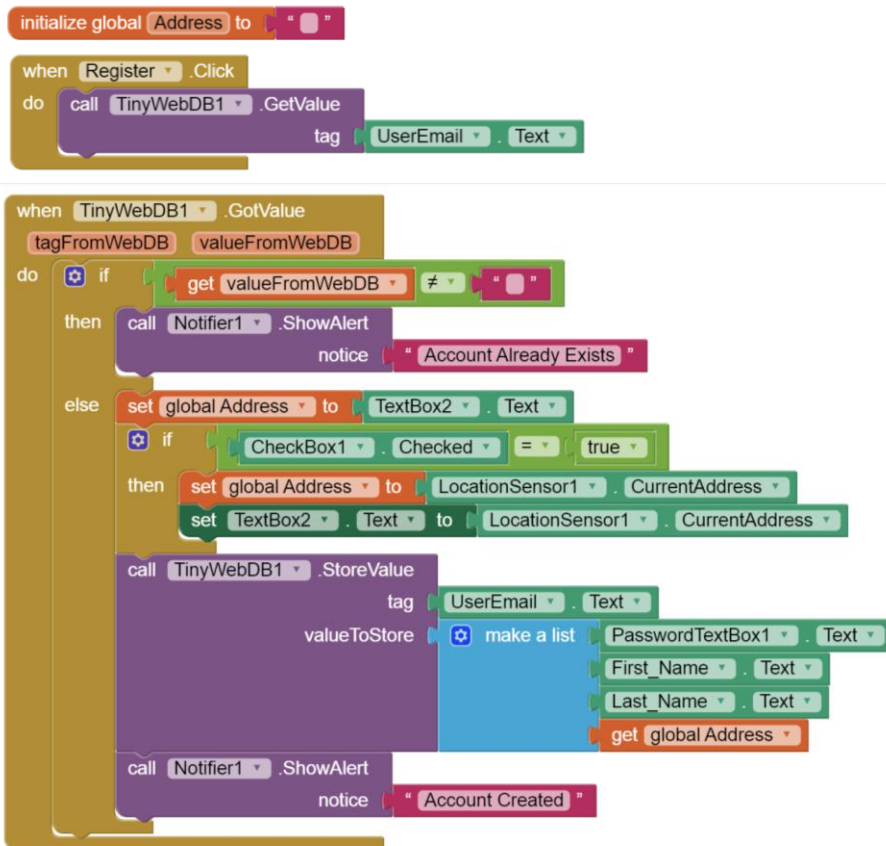
```
when MapMyUser .Click
do
  open another screen with start value screenName " MapMyUser "
  startValue get global email
```

Creating a Registration Screen

The screenshot displays the App Inventor web interface for creating a mobile application. The browser address bar shows the URL `ai2.appinventor.mit.edu/?locale=en#6123941813551104`. The interface is divided into several sections:

- Component Palette (Left):** A list of UI components including Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, TextBox, TimePicker, and WebViewer. Below this are categories for Layout, Media, Drawing and Animation, and Maps.
- Design Canvas (Center):** A preview of the registration screen. It features a title bar labeled "userinfo" with a back arrow. Below the title bar are input fields for "User Name", "Address" (with a "Use my current location" button), "Email", and "Password" (with a masked input). A "Register" button is positioned at the bottom of the form. The status bar at the top shows the time as 9:48.
- Non-visible components (Bottom Center):** A row of icons representing hidden components: TinyWebDB1, TinyDB1, Notifier1, and LocationSensor1.
- Properties Panel (Right):** A list of UI components with their respective properties. The "AboutScreen" properties are visible, including:
 - AlignHorizontal: Left : 1
 - AlignVertical: Top : 1
 - BackgroundColor: Default
 - BackgroundImage: None...
 - CloseScreenAnimation: Default
 - OpenScreenAnimation: Default
 - ScreenOrientation: Unspecified
 - Scrollable:
 - ShowStatusBar:
 - Title: userinfo
 - TitleVisible:

Registration blocks



```
initialize global Address to ""

when Register.Click
do
  call TinyWebDB1.GetValue
  tag userEmail.Text

when TinyWebDB1.GotValue
  tagFromWebDB valueFromWebDB
  do
    if (get valueFromWebDB != "")
    then
      call Notifier1.ShowAlert
      notice "Account Already Exists"
    else
      set global Address to TextBox2.Text
      if (CheckBox1.Checked == true)
      then
        set global Address to LocationSensor1.CurrentAddress
        set TextBox2.Text to LocationSensor1.CurrentAddress

      call TinyWebDB1.StoreValue
      tag userEmail.Text
      valueToStore make a list
      PasswordTextBox1.Text
      First_Name.Text
      Last_Name.Text
      get global Address

      call Notifier1.ShowAlert
      notice "Account Created"
```

- ➔ Check that it is a new user
- ➔ Set their address if they picked current location
- ➔ Store the user information in TinyWebDB

Storing in TinyWebDB

```
when TinyWebDB1 .GotValue
tagFromWebDB valueFromWebDB
do
  if
  then
    call Notifier1 .ShowAlert
    notice "Account Already Exists"
  else
    set global Address to TextBox2 . Text
    if
    then
      set global Address to LocationSensor1 . CurrentAddress
      set TextBox2 . Text to LocationSensor1 . CurrentAddress
    call TinyWebDB1 .StoreValue
    tag UserEmail . Text
    valueToStore
    make a list
    PasswordTextBox1 . Text
    First_Name . Text
    Last_Name . Text
    get global Address
    call Notifier1 .ShowAlert
    notice "Account Created"
```

- Store the user info in a list
- The userEmail is the tag
- Password is list item 1
- First name is list item 2
- Last name is list item 3
- Address is list item 3
- On Screen1 and MapMyUser use those list items to get the values back

Creating a Map Screen

The screenshot displays the Android Studio IDE interface for a project named "RegistrationExample". The top bar shows the current screen is "MapMyUser" and includes buttons for "Add Screen ..." and "Remove Screen". The interface is divided into several panels:

- Palette:** A list of UI components under the "User Interface" category, including Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, TextBox, TimePicker, and WebViewer. Below this are sections for "Layout", "Media", "Drawing and Animation", and "Maps".
- Viewer:** The central workspace showing a mobile app preview. The app has a title bar "MapMyUser" and a status bar showing signal strength, Wi-Fi, and the time "9:48". The main content area contains two text input fields, a "ShowMap" button, a "Show Directions" button (highlighted in yellow), and a "Return" button. Below the preview, there are checkboxes for "Display hidden components in Viewer" and "Check to see Preview on Tablet size.". At the bottom of the viewer, a "Non-visible components" section lists TinyWebDB1, LocationSensor1, Notifier1, and ActivityStarter1.
- Components:** A tree view showing the hierarchy of components on the screen: MapMyUser (root), Email_for_user, Address, Wrong, ShowMap, ShowDirections (selected), Return, TinyWebDB1, LocationSensor1, Notifier1, and ActivityStarter1. Buttons for "Rename" and "Delete" are visible at the bottom.
- Properties:** A panel for configuring the selected "ShowDirections" component. It includes properties such as ShowDirections, BackgroundColor (Default), Enabled (checked), FontBold, FontItalic, FontSize (14.0), FontTypeface (default), Height (Automatic...), Width (Automatic...), Image (None...), Shape (default), ShowFeedback (checked), and Text (Show Directions).

Mapping blocks

- ➔ Getting the user info – the starting value for the email was passed in from the first screen

```
when MapMyUser .Initialize
do
  set global useremail to get start value
  set Email_for_user .Text to get global useremail
  call TinyWebDB1 .GetValue
  tag get global useremail
```

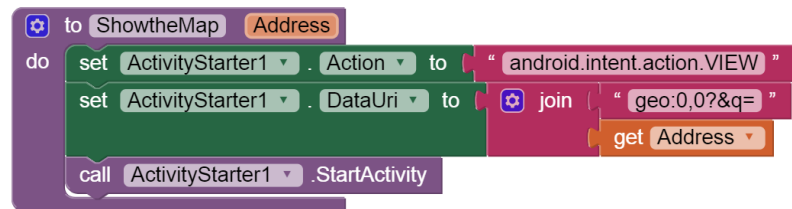
Get the User's info from TinyWebDB

```
when TinyWebDB1 .GotValue
  tagFromWebDB valueFromWebDB
do
  if is a list? thing get valueFromWebDB
  then
    set global password to select list item list get valueFromWebDB
    index 1
    set global address to select list item list get valueFromWebDB
    index 4
    set Address .Text to get global address
  else
    set Wrong .Visible to true
```

Remember the address was the Fourth item in the list

Map the users location

- ➔ When the user clicks to show the map call the activity starter to start the map.
- ➔ We need to create a procedure for the map
- ➔ Pass in the users address and start the map with that address



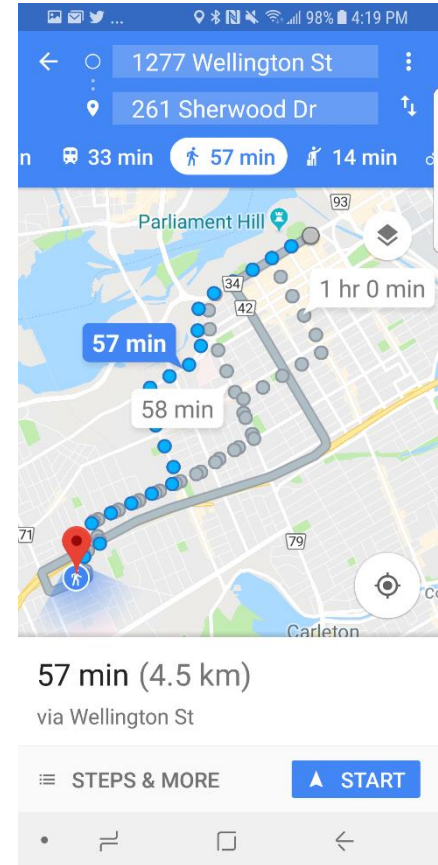
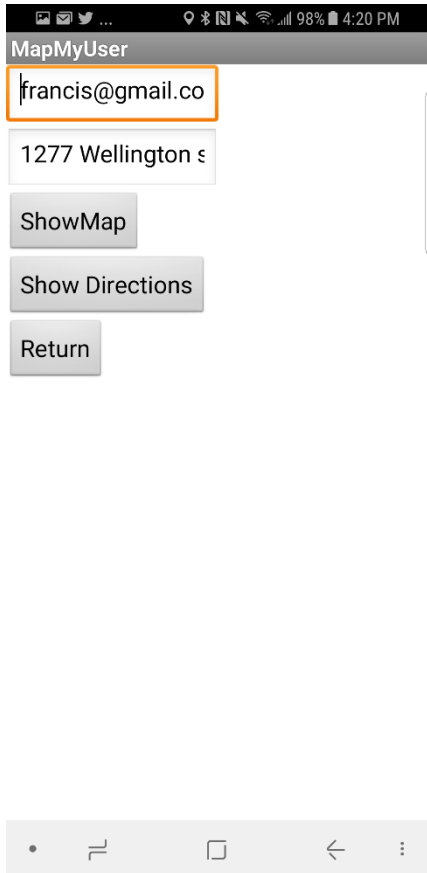
Show Directions

- ➔ When the user clicks directions you need to pass the start and destination addresses
- ➔ saddr is the start, daddr is the destination
- ➔ Call the map to go from their address to the current location

```
when ShowDirections .Click
do
  call ShowtheMap2
  Address get global address
```

```
to ShowtheMap2 Address
do
  set ActivityStarter1 . Action to " android.intent.action.VIEW "
  set ActivityStarter1 . DataUri to join ( " http://maps.google.com/maps?saddr= "
  get Address
  " &daddr= "
  LocationSensor1 . CurrentAddress
  call ActivityStarter1 .StartActivity
```

Your completed app



More with maps

- ➔ Letting the user choose which address to map
 - Want to show where a job opportunity is
 - Want to show where a sports event is
 - Want to show where to drop off charity items ...
- ➔ Create a list with the addresses
- ➔ Use the list picker to choose which one to map

Creating an address list

```
to appendNewAddress
do
  set global listLocations to ListPicker1 . Elements
  add items to list list
  item get global listLocations
  item upcase EnterAddressText . Text
  call TinyDB1 .StoreValue
  tag get global tagAddress
  valueToStore get global listLocations

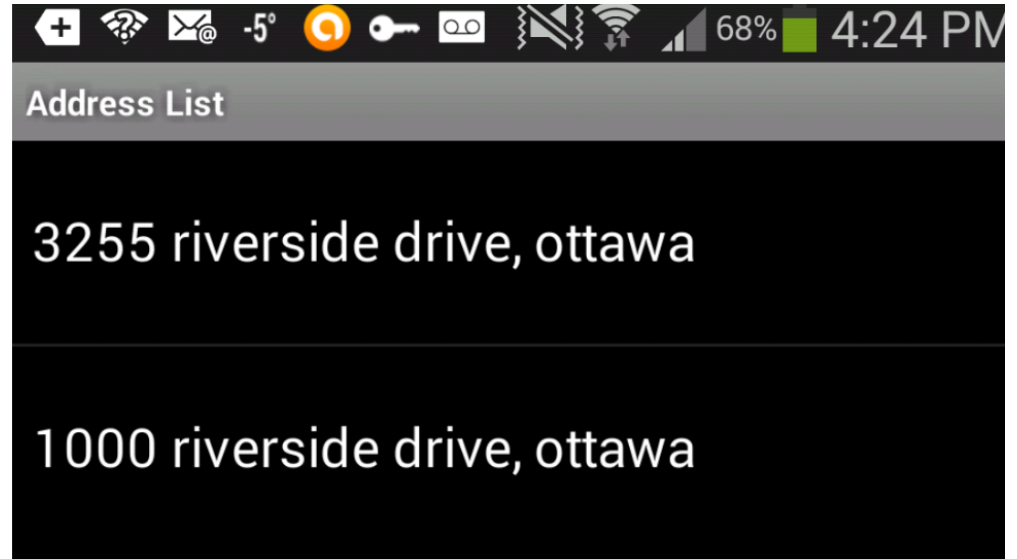
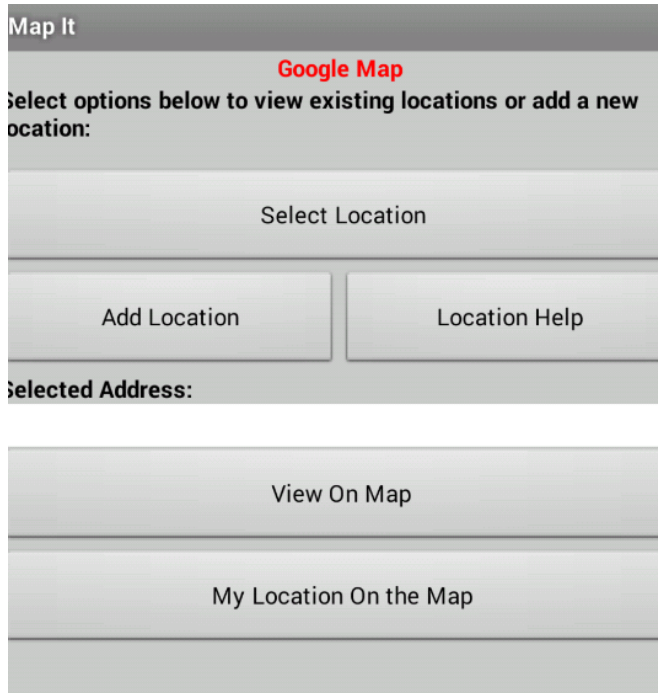
when SubmitButton .Click
do
  if is empty trim EnterAddressText . Text
  then
    call Notifier1 .ShowMessageDialog
    message " No address has been entered. Please enter address to submit. "
    title " Info "
    buttonText " Okay "
  else
    call appendNewAddress
    set EnterAddressText . Text to " "
    set VerticalArrangement1 . Visible to false
    call Notifier1 .ShowAlert
    notice " Address was added! "
    set AddLocationButton . Enabled to true
    set ListPicker1 . Enabled to true
```

Retrieving an address list

```
when Screen1.Initialize
do
  call initData
  set ListPicker1.Enabled to false

to initData
do
  set LocationSensor1.Enabled to true
  set ListPicker1.Title to "Address List"
  set VerticalArrangement1.Visible to false
  set global listLocations to call TinyDB1.GetValue
    tag get global tagAddress
    valueIfTagNotThere get global listLocations
```

Selecting an address



When the user clicks on ListPicker1 'Select Location' and selects an address, this action calls the blocks below:



Exploring maps

- ➔ <http://appinventor.mit.edu/explore/ai2/android-wheres-my-car.html>
- ➔ <http://appinventor.mit.edu/explore/displaying-g-maps.html>