# Clicker App with CloudDB Screenshot (69).png

# Getting Started

Start App Inventor with [Clicker App Template](http://ai2.appinventor.mit.edu/?repo=templates.appinventor.mit.edu/trincoll/csp/unit7/templates/ClickerAppCloudDB/ClickerCloudDBtemplate.asc). Once the project opens use Save As to rename your project *CloudDBstudent.*

# User Interface

| **UI Component** | **Name** | **Properties** |
| --- | --- | --- |
| CloudDB | CloudDB1 |  |
| Notifier | Notifier |  |

# You may also change the Question text.

Note: CloudDB sometimes has connection problems due to server overload. If you get a socket connection error, switch to using the Experimental/FirebaseDB and its associated blocks instead of CloudDB in this tutorial!

# Coding the App

| **Variables** | | **Values** |
| --- | --- | --- |
| agreeCount | | 0 |
| disagreeCount | | 0 |

# 

| **Event Handlers** | | **Algorithms** |
| --- | --- | --- |
| Screen1.Initialize | | Call a new procedure *getDBvalues*. |
| ButtonAgree.Click | | Add 1 to agreeCount. Call a new procedure *storeDBValues*. |
| ButtonDisagree.Click | | Add 1 to disagreeCount. Call procedure *storeDBValues*. |
| ButtonReset.Click | | Set agreeCount to 0. Set disagreeCount to 0. Call procedure *storeDBValues*. |
| CloudDB1.GotValue | | Call a new procedure *gotData* with 2 parameters tag and value. |
| CloudDB1.DataChanged | | Call a new procedure *gotData* with 2 parameters tag and value. |
| CloudDB1.CloudDBError | | Call Notifier.ShowAlert with the message. |

| **Abstraction: Procedures** | | **Algorithms** |
| --- | --- | --- |
| getDBvalues | | Calls CloudDB1.getValue for the tag “agree” and 0 for the valueIfTagNotThere.  Calls CloudDB1.getValue for the tag “disagree” and 0 for the valueIfTagNotThere. |
| storeDBvalues | | Call CloudDB.storeValue with tag “agree” and value agreeCount.  Call CloudDB.storeValue with tag “disagree” and value disagreeCount. |
| gotData(tag, value) | | If the tag is “Agrees” then set agreeCount to the value returned from the database.  Else if the tag is “Disagrees” then set disagreeCount to the value returned from the database.  Call new procedure updateDisplay. |
| updateDisplay | | Set labelAgree to “Agree:” joined with agreeCount.  Set labelDisagree to “Disagree:” joined with disagreeCount. |

# Testing the App

This app is best tested by forming a group of students where everyone in the group loads one student's app using **Build/App (provide QR code for apk).** Or the whole class could load 1 student’s app projected on the screen. When one of student in your group votes, the latest data should update on everyone’s screen. Because this app is more easily tested using .apk files, we recommend it be built (and tested) on Android devices until iOS .apk files become available in App Inventor.

| **Inputs** | | **Expected Outputs** | **Actual Outputs** |
| --- | --- | --- | --- |
| Test clicking agree and disagree buttons with other people in your group. | | All apps in the group should update whenever you click on a button and show the agree and disagree counts. | ? |

# Enhancements

# Enhancement #1: Create a Bar Chart Using the Thumb Switches

Read [this documentation](http://ai2.appinventor.mit.edu/reference/components/userinterface.html#Slider) and watch this [video](https://www.youtube.com/watch?v=cm2-kVcWTuw&feature=youtu.be) on sliders. Sliders or thumb switches are most frequently used to allow the user to set the value of some property by moving their thumb on a sliding scale. For our Clicker app, we will be using this component in reverse - to create a ***bar chart*** based on the ratio of “Agree” and “Disagree” votes recorded by the app.

# Enhancement #2: Allow Users to Vote Only Once

Modify the app so that the Clicker only allows the user to vote once (hint: there is an Enabled property for buttons). The updating feature should still continue to work with the timer. Add re-enabling the voting buttons when the user hits reset. You may want to consider turning off this feature when it comes time for you to demonstrate your app to your instructor.

# Enhancement #3: Build a Teacher Version of the App

Add a feature that will allow a special version of the app, the “Teacher” version, to update the question displayed on the screen in real time. First in the student app,

* Change the student version of the app to accept new questions while the app is running. This will involve adding code to the ***CloudDB.DataChanged*** event handler to see if the question was changed in the database (use the “question” tag) and changing the question label accordingly and re-enabling the voting buttons. Note that the Question data will consist of a string, whereas the agree and disagree data were numbers.
* Remove the ***RESET*** button from the UI of the student side so that only the teacher can reset the counters.

Build a separate version of the app called "ClickerTeacher" (use File/Save As). Allow only this version to change the questions. Note that when you use File/Save As, the CloudDB ***token*** and ***ProjectID*** will both stay the same, so the student app and the teacher app can share the same database. Also, when testing the app, it may be easier to use QR codes to load the two versions of the app instead of trying to use the Companion.

* Replace the Question Label in the teacher version of the app with a TextBox to allow the teacher to update the question field in real time.
* Add an “Update Question” button to the teacher app that will store the new question into the CloudDB database from where it will get pushed to all the users. Remember the tag name you used! Also, reset the counters and store them in the database too.
* Test with your group with one student using the teacher app and the rest using the corresponding student apps.