Ask a Friend (AFF) device side

Lawrence Botley  
Friday, 14 September 2007

This document contains the following sections:

1. Ask a Question
2. View the responses
3. Web Services

Core functionality

* Disk I/O for browsing images
* Image resizing – images should be resized to 320x240 and sent to server as JPEGS
* Use camera viewfinder and allow users to snap photos - images should be resized to 320x240 and sent to server as JPEG files.
* Use web service (SOAP) to post text data and photos – This is an ASP.NET web service. The address of the server and access to the WSDL is available at <http://12.206.33.18:66/AskAFriend.asmx>

More details on the web service can be found at the end of this document

**1.0 Ask a Question**

Process overview:

1. The user on the device will Select the AAF (Ask A Friend) option
2. They will either have the photo(s) on the device already or wish to snap the photo(s) now. Therefore they are presented with the option to either use the inbuilt photo viewfinder or browse the flash disk for the photo(s).
3. Once they have selected up to three photos they are forwarded to the question screen where they will write their question and then select the appropriate response type (YES/NO, Image select, rating, etc).
4. The user will then decide the duration of long the question will remain live for. (5, 10, 60 minutes or 24 hours)
5. Finally the user will submit the question. As the question is being transported to the server they are presented with an advert that will be downloaded asynchronously after the actual text of the question is sent. (this will allow the advert to be relevant to the question they are asking and should only take 5-10 seconds to get the advert text/image to the device)

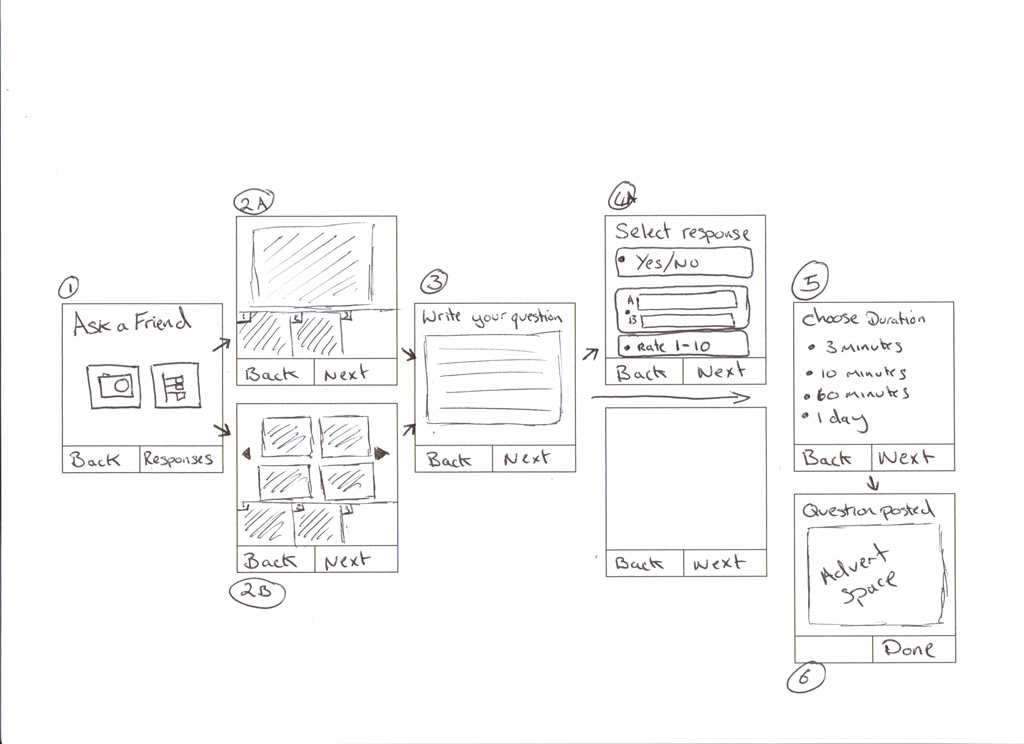


Diagram 1.1 – The 7 different screens of the AAF question posting wizard

1. This is the opening splash screen for the AAF feature. The main part of the screen contains an image with the AAF text. This simple denotes that they have started the AAF wizard. This screen contains two options
   1. Use Camera - 2A
   2. Select Image – 2B
   3. 2A – Selecting to use the camera presents the user with the camera viewfinder. Upon taking a photo they will be asked if they wish to use the photo. Once a photo has been taken the user is able to click the right soft key (Next) and go to the next screen. However, if they desire the user may up to 3 photos moving to the next screen. Each time a photo is snapped and ok’ed it will appear at the bottom of the screen. (three boxes are placed in a row at the bottom of the screen to accommodate this)
   4. 2B – Much like 2A, the user can browser their devices for up to 3 images stored on the flash disk. The photo browser will display photos in blocks of 4 (showing the latest photos first). The arrow keys will be used to either highlight and select an image or to select the forward or backward key to navigate to another block of 4 images. Once 1 photo has been taken the user is able to click the right soft key (Next) and go to the next screen. However, if they desire the user may up to 3 photos moving to the next screen. Each time a photo is snapped and ok’ed it will appear at the bottom of the screen. (three boxes are placed in a row at the bottom of the screen to accommodate this)

The developer should choose a suitable method to allow the user to browse that fits in accordance of the device controls that are available. (in other words.. don’t create a custom image browser if necessary, use the controls available in the J2ME/Symbian control set).

1. Screen 3 allows the user to enter their question text. The textbox will allow the user to enter a question of up to 255 characters in length. Since this screen and the next few screens require very little resources, it would be a good idea to execute the image resizing functionality upon this screen loading. This way the photo(s) will be already resized into the 512 width (and relative height) resolution for the final upload and will reduce overall waiting time. Photo resizing should be threaded so as not to cause a slowdown to the interface.
2. Screen 4 is an optional screen that will only appear if the user has selected a single photo in screen 2A/2B. If more than one photo has been chosen then the response is assumed that the question requests the responder to respond by selecting one of the photos IE: *which car looks the coolest?*. In these cases screen 3 will jump straight to screen 5 when the user presses the *Next* key.

In the event that only a single photo was selected, the user may choose how they want the question to be answered. Responses fall into one of the three following categories

* 1. Yes/No
  2. Custom A/B answer (denoted by two textboxes that allow the user to enter their response)
  3. Rating value of 1-10

1. Screen 5 asks the duration that the question will appear for. These options are:
   1. 3 Minutes
   2. 15 Minutes
   3. 60 Minutes
   4. 1 day
2. Screen 6 (The final screen in the AAF wizard) shows a confirmation message denoting that the question is being submitted. This is where an advert will eventually be placed. For the purpose of the initial beta the advert need not be shown.

Through every option of the AAF the left soft key will display the *Back* button and the right soft-key will contain a “Next” option. The only exception to this is the first screen where the right button will be labelled “Responses” where this option will allow the user to view the responses to their previously asked questions. Finally, for the last screen the right soft-key will be labelled “Done” (and return the user to the first AAF screen). The left softkey on the last page will be blank.

**2.0 View Question Responses**

The AAF homepage allows users to access all the previously asked questions to view responses.

Once the Responses soft-key has been pressed on the AAF home screen the user will be presented with a list of all the questions that they have previously asked. The question responses are obtained using the N2F web service. The list can be refreshed by pressing the right soft-key. Users may use the up and down key to select to view a result from a question in the list. Any questions that have not yet finished (pending) and greyed out, are non-selectable and have a number in minutes and seconds until the poll has finished.

Users will also be given the option to view any optional comments that users may have posted with their replies. By using the right soft-key it is possible to toggle back and forth between the questions vote responses and comments.

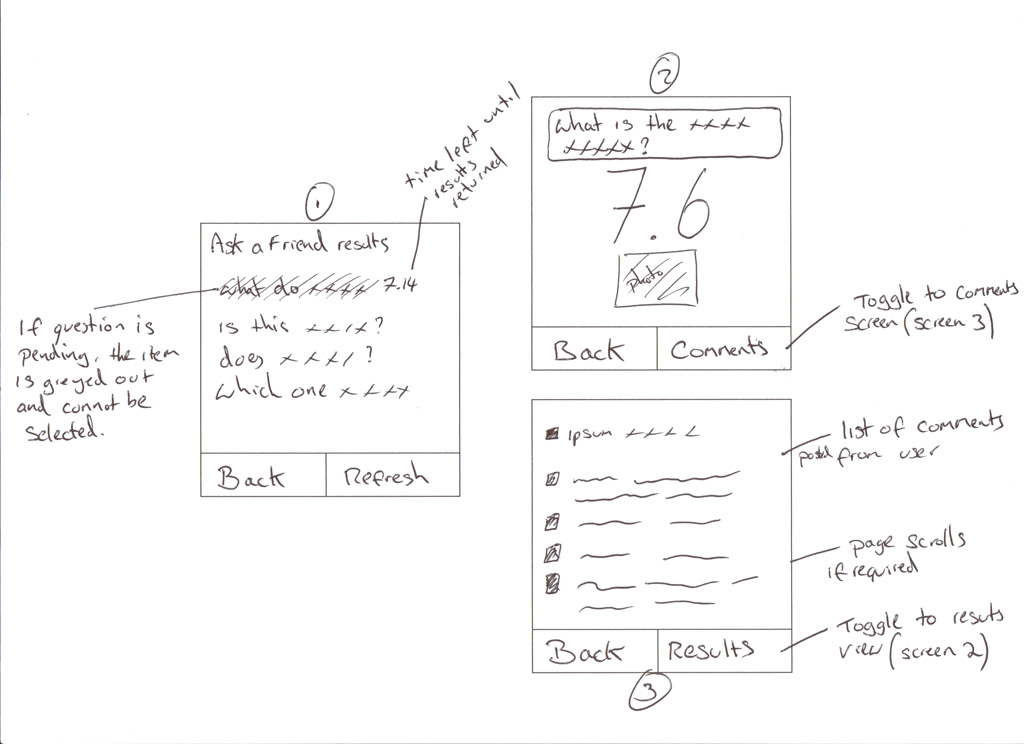


Figure 2.1 – The 3 screens of question responses

Diagram 2.1 shows the 3 main screens of the response sections and can be broken down as follows:

1. A list of all questions asked. This is a scrollable list that when an item is selected they are taken to screen 2.
2. The main results page for the question. There are 4 variations of this screen that can display the different result sets. These will be included at a later date. The right soft-key will display *Comments.* When this is pressed the user will be forwarded to screen 3.
3. This screen lists all the comments made by users for this question. The right soft-key (*results*) will allow the user to toggle back to the main results screen (Screen 2).

A response will take form in the following:

1. An average value of 1 – 10 (result from a rating)
2. A 2-3 set results of values where users have voted for a particular photo. (this can be displayed in a simple bar graph)
3. A pair of results where users have selected Yes or No (this can be displayed in a simple bar graph)
4. A pair of results where users have selected the custom A or B option (this can be displayed in a simple bar graph)

**3.0 Web services**

**3.1 Using web services in J2ME**

A full overview of how SOAP is utilised in J2ME can be found at <http://developers.sun.com/mobility/apis/articles/wsa/>

A tool for creating java classes from WSDL long with detailed information on SOAP can be found at <http://ws.apache.org/axis/java/user-guide.html#WhatIsSOAP>

**3.2 Using web services in Symbian**

Although there may be a number of tools available for SOAP and web services, one such tool by Borland is a port of GSOAP and can at <http://www.cs.fsu.edu/~engelen/soap.html>

**3.3 Ask a Question**

There are 3 web methods to call in order to invoke and complete an AFF request. The reason why this process is split up into three methods is because of the following:

* The initial request posts the actual text question to the server, from this the user will be returned a relevant ad based on the content of the question. This is displayed to the user while their photos are uploaded to the server (which could be a minute or so). The SubmitQuestion methos returns an object , this will contain an jpeg image file (as a byte array), a unique identifier (to identify what question the subsequest images are part of) and finally a URL if the user decides to click through on the advert .
* Should connectivity be lost while photos are being uploaded, the user will be able to continue and pick up where they left off instead of uploading all the photos again.

**1 SubmitQuestion()**

Begins the AAF question posting process

/// <summary>

/// Call this method to start a question. only once the NumberOfPhotos has equal value

/// to the number of Attached photos can CompleteQuestion() be called

/// </summary>

/// <param name="WebMemberID">The members unique ID</param>

/// <param name="Password">The members password</param>

/// <param name="Question">The questions to be submitted</param>

/// <param name="NumberOfPhotos">Total number of photos to be submitted</param>

/// <param name="ResponseType">What type of question to ask</param>

/// <param name="CustomResponses">Optional values of Response A and Response B</param>

/// <param name="Duration">enumeration of how long the question is open for</param>

/// <param name="IsPrivate">Is the question indended for Friends only</param>

/// <returns></returns>

[WebMethod]

public AskAFriendConfirm SubmitQuestion(string WebMemberID, string Password, string Question,

int NumberOfPhotos, int ResponseType, string[] CustomResponses, int Duration, bool IsPrivate)

**2 AttachPhoto()**

Attaches the photos to the question. In IndexOrder parameter denotes whether the photo file was image 1, 2 or 3.

/// <summary>

/// Attaches a single photo to an AskAFriend Question

/// </summary>

/// <param name="WebMemberID">The members unique ID</param>

/// <param name="Password">The members password</param>

/// <param name="WebAskAFriendID">The WebAskAFriendID to attach the photo to</param>

/// <param name="IndexOrder">if there are more than 1 photo, this order in which they will appear.

/// This is used to attach responses from multiple select</param>

/// <param name="PhotoBase64String">The photo file as a Base64Binary string</param>

/// <returns>true if the photo was sucessfully attached. an Exception is thrown otherwise</returns>

[WebMethod]

public bool AttachPhoto(string WebMemberID, string Password, string WebAskAFriendID, int IndexOrder, string PhotoBase64String)

**3 CompleteQuestion()**

Finalises the question and promotes it to live

/// <summary>

/// Completes an AAF question and promotes it to live

/// </summary>

/// <param name="WebMemberID">The members unique ID</param>

/// <param name="Password">The members password</param>

/// <param name="WebAskAFriendID">The WebAskAFriendID of the AskAFriend that has been completed</param>

/// <returns>true if the command was successul. an Exception is thrown otherwise</returns>

[WebMethod]

public bool CompleteQuestion(string WebMemberID, string Password, string WebAskAFriendID)

**4 GetPrivateAAFQuestions()**

Returns an array of strings containing all the new AAF private questions. The strings are URLS and are to be passed to the device web browser.

/// <summary>

/// Gets all the new AAF Questions

/// </summary>

/// <param name="WebMemberID">The members unique ID</param>

/// <param name="Password">The members password</param>

/// <returns>An Array of strings containing URLs for the device browser to launch</returns>

[WebMethod]

public string[] GetPrivateAAFQuestion(string WebMemberID, string Password)

**6 GetMyAAFQuestions()**

Gets all the members AAF questions so they can select to view individual responses

/// <summary>

/// Gets the the members AAF questions. The object returned is a data structure containing

/// the WebAskAFriendID and question text

/// </summary>

/// <param name="WebMemberID">The members unique ID</param>

/// <param name="Password">The members password</param>

/// <returns>A AskAFriendQuestion array is returned. The AskAFriendQuestion data structure

/// contain the text question and a WebAskAFriendID</returns>

[WebMethod]

public AskAFriendQuestion[] GetMyAAFQuestions(string WebMemberID, string Password)

**7 GetAAFResponses()**

Gets the response for a question that the member has asked

/// <summary>

/// Gets the response for a question that a member has asked

/// </summary>

/// <param name="WebMemberID">The members unique ID</param>

/// <param name="Password">The members password</param>

/// <param name="WebAskAFriendID"></param>

/// <returns>A MobileAskAFriendResponse object</returns>

[WebMethod]

public MobileAskAFriendResponse GetAAFResponse(string WebMemberID, string Password, string WebAskAFriendID)

In order to view a detailed view of all the parameters in the web services, a web browser can be pointed at <http://12.206.33.18:66/AskAFriendWS.asmx>

**Test WebMemberID and passwords**

Password always = “password”

**WebMemberID**

ZDNjNzYzYjkzYWU5NDVjMz  
MTM5NjhiZjIxZjVjNDZkMD  
MzY3MTE5ODJjY2VlNDA0OD  
Mzk5OWEwN2Y5ZDg5NDg3Mz  
YmFmNzYwNGU4NzVkNGM0ZW  
NjgyYzFhODYzMDMwNGZkOD