Experience Sampling

Remote survey system

This application and backend interface allows researchers to conduct surveys remotely using the mobile phone on Android and iOS.

This app is developed by BOSONIC.design in assignment of the department of Human-Technology Interaction @ Eindhoven, University of Technology.

info@bosonic.design - http://www.bosonic.design/

hti@tue.nl - https://www.tue.nl/en/university/departments/industrial-engineering-innovation-sciences/research/research-groups/human-technology-interaction/

App customization

In “www/js/func.js” you find the most important function that you might want to edit to customize the app to your needs. This file includes for example the “renderQuestion” function in which you can provide logic to show certain question depending on specific parameters. For example, you can hide a question in the weekend (Saturday and Sunday) and show only throughout workdays (example provided). You can also show specific question only in the morning, or the evening.

Furthermore, this file allows you to schedule local notifications, to remind your participants to fill in the questionnaire on specific times. This function is called “scheduleNotifications”. This part is based on the local-notifications plugin of Cordova, so more information about how to set-up notifications is found here: https://github.com/katzer/cordova-plugin-local-notifications .

Lastly, you might want to edit is “www/js/pages.js”. This file contains the navigation logic of the app, allow specific function to fire when you enter or leave a specific page. The app is based on Framework7 framework, so check out https://v1.framework7.io/docs/page-callbacks.html to see what navigation functions are possible. You can also look at the general documentation to see what else is possible: https://v1.framework7.io/docs/get-started.html

In order to change the styling of the app, change the CSS in “www/css/style.css” accordingly.

Modules

In the folder “www/js/modules” you can find files containing additional questions types and you can manage them. For example, “basics.js” contain the logic for most basic question types, like input fields or likert scales. “camera.js”, “geo.js” and “microphone.js” contain more specialized question types that include extra functionalities. “module.example.js” shows an example of how to include your own question types to the system. Read this file carefully to understand how it works.

Make sure that at the bottom of “www/index.html” you add or delete the paths to the module files to manage what files you are using in your application.

Additional functions

In the folder “www/js/functions” additional files can be found that contain functions used within the system. For example, functions related to registering and logging in as well as the messaging system.

Questions Type Per Module

Basics.js

Type	Description	Label	Label Example
ShortText	Single line text input	-	
LongText	Multi line text input	-	
Select	Select a single option	Write down individual option, separated by ';'. Write '-s' behind an option to automatically select it.	Mouse;Duck-s;Bee **Note that 'Duck' will be automatically selected.**
MultiSelect	Select multiple option	Write down individual option, separated by ';'.	Mouse;Duck;Bee
Likert	Likertscale (similar as Select, however the options will be displayed horizontally).	Write individually cases like Select. However, one can also write only '5', '7' or '9' to automatically generate likert scale with corresponding options.	No;Neutral-s;Yes **Note that 'Neutral' will be automatically selected.** OR 5 **Note that automatically a 5-point Likert scale will be generated labelled '1' to '5'.**
Slider	Select a value between a maximum and minimum number.	Write in the form of "minimum-selected-maximum", were minimum is the lowest value, selected the default value and maximum the maximum value.	0<50<100
Dropdown	Select a option from a dropdown list	Write down individual option, separated by ';'.	Mouse;Duck;Bee
Date	Select a date	-	
DateText	Generates 3 specific fields for date input in the form dd/mm/yyyy	-	
Time	Select a time	-	

Camera.js

Type	Description	Label	Label Example
Photo	Select photo from phone or take a photo with the camera	-	

Geo.js

Type	Description	Label	Label Example
ShareLocation	Generates a checkbox, for accepting to share the device's current location and saves the latitude and longitude.	-	
ChooseLocation	Allows the user to select a location on a map from Google. Saves latitude and longitude of the specified location	-	

Microphone.js

Type	Description	Label	Label Example
Recording	Record audio through the microphone	-	

Messaging to participants

The system is equipped with a messaging system to communicate with your users. To send message you need to open phpMyAdmin and search for the messages table. If you insert a new record you have access to the following properties:

- **Mid:** message ID – keep empty, this will be generated automatically
- **Date:** time the message should be fetched by application and shown to the user.
- **Content:** your message (supports html, but it is recommended not to use the tag as this can mess up the page).
- **Uid:** can be used to send a message to a single specific user, but is optional.
- **AllUsers:** set to '1' to send this message to all available users.
- **RetrievedBy:** this are the users ID's that received the message – keep empty, this will be automatically filled
- **ReadBy:** this are the users ID's that actually read the message – keep empty, this will be automatically filled

A downside of the current implementation is that the messages will only be checked when the users opens the menu page, but no push notification will be send to make the user aware of this. Instead, you can use firebase, as complementary feedback, to send notifications at the same time (time set in the property **Date**) as that the message should appear to the user.

Setting up the system

Web-interface set-up

phpMyAdmin

1. Navigate to your phpMyAdmin environment, databases and import a new database. In the main folder you can find ExperienceSampler.sql, that has to be imported.

2. Navigate to the new created database and select the questions table. Edit here the questions according to the table found under “Survey Questions” in this document.

Web-interface files

1. In the Sampling-WebInterface folder open the file “php/Config.php”.
2. In this document give up credentials that will be used by the app to identify itself (remember these, as you will need them for the app), as well as the credentials to access the database.
3. That is all! Upload all the files to your host.
4. !- important, make sure you also place the .htaccess files on your host and make sure they are working. These files are important to prevent access to your data and files from outside your host environment.

General app set-up

1. Navigate to “Mobile-Experience-Sampling-Master/Sampling-App” and open the config.xml file.
2. On line 2, give your app an appropriate package name, in the widget’s attribute id (default value is “com.tue.experienceSampler”) and app name (default value is “Experience Sampling”).
3. You might also want to edit additional parameters like description and author information found on line 4 to 8.
4. On line 18 and 25 you find the access tag with default value “http://yourweb.com”, change this value to your own host were you uploaded the web-interface of the app.
5. Next, open the file config.js found in “www/js”.
6. In this file adjust the parameters found on line 18 to 20, with the credentials you set in the WebInterface.

Android Specific

1. Go to firebase (https://console.firebase.google.com/), create a project and download the “google-services.json” file for the android SDK.
2. Place this file in the root folder of the app (“Mobile-Experience-Sampling-Master/Sampling-App”).
3. Go to Google maps Android SDK (https://developers.google.com/maps/documentation/android-api/) and get your google maps key.
4. Using command line navigate to the root folder of the app, and execute the following code:
    a. Cordova plugin add cordova-plugin-firebase
    b. Cordova plugin add cordova-plugin-googlemaps --variable API_KEY_FOR_ANDROID="<insert google maps key here>"
    c. Cordova platform add android@6.3.0
5. Run the following code in the command line: “Cordova build android” to create your apk file for testing purposes or upload the apk file directly to your phone over USB with the Android Debug Bridge (ADB). Look at cordova’s documentation for more information about building a release version of the app.

iOS Specific

1. !- important, you need a **PAID** Apple Developer account in order to make push notifications work.
2. Go to firebase (https://console.firebase.google.com/), create a project and download the “GoogleService-Info.plist” file for the iOS SDK.

3. Place this file in the root folder of the app ("Mobile-Experience-Sampling-Master/Sampling-App").
4. Go to Google maps iOS SDK (https://developers.google.com/maps/documentation/ios-sdk/) and get your google maps key.
5. Using command line navigate to the root folder of the app, and execute the following code:
    a. Cordova plugin add cordova-plugin-firebase
    b. Cordova plugin add cordova-plugin-googlemaps --variable API_KEY_FOR_IOS="<insert google maps key here>"
    c. Cordova platform add ios
6. Then open the file <app_name>.xcodeproj in the folder "platforms/ios" and follow the Xcode process to push the app to your device.
7. Make sure you turn on "inter-app audio" and "push notifications" to make certain plugins work.

Firebase Usage

Go to grow > notifications to start sending push notifications.

Under "message text" write the text that is visible in the push notification itself; note that this text will not be visible in the app.

For sending a notification to everyone, use for target topic with the text 'Announcement'.
For sending it to a specific user us for target single device and place a token from the database (Users table) in the field.

Then open the advanced options. Under "Title" you can write the title of your push notification that appears; note that this text will also not be visible in the text.

Under "custom data" add the following to make a message appear in the app: for the field key use 'content' and for value write the message that you wish to appear on the message page. Note that you can use HTML-tags here to change the appearance of the message (MAKE sure you close them though, as there is no code that checks this, and could therefore mess up the page).

If you do this correctly, you will receive a push notifications and/or a message on the message page.

Deployment Essentials

When deploying your app to the 'wild' make sure you add https://github.com/tkyaji/cordova-plugin-crypt-file to your application to encrypt your app and increase safety. This will prevent people that have access to your app, to look at the JavaScript code from Chrome's remote devices functionalities. This can be harmful, as one can see the location of your back-end database.

Plugins versions

This are the plugin versions which should work with this system:

- cordova-plugin-badge 0.8.7 "Badge"
- cordova-plugin-camera 4.0.2 "Camera"
- cordova-plugin-device 2.0.1 "Device"
- cordova-plugin-file 6.0.1 "File"
- cordova-plugin-file-transfer 1.7.1 "File Transfer"
- cordova-plugin-firebase 0.1.25 "Google Firebase Plugin"
- cordova-plugin-geolocation 4.0.1 "Geolocation"
- cordova-plugin-googlemaps 2.2.5 "cordova-plugin-googlemaps"
- cordova-plugin-local-notification 0.9.0-beta.3 "LocalNotification"
- cordova-plugin-media 5.0.2 "Media"
- cordova-plugin-network-information 2.0.1 "Network Information"
- cordova-plugin-whitelist 1.3.3 "Whitelist"