# The Mashlever App – Crowdsourcing Citizenship

It’s 11:30 a.m. in the morning. Your company’s vice president of sales would like to show a prototype application to a major customer at 12:30p.m. The prototype application doesn’t exist. Yet!

The proposal is as follows. The client would like an app that runs on Android, iPhone and Windows Phone which operates much like popular ‘instant response’ polling software. In the existing methodology, in-person focus groups are monitored for their reaction to political debates. In the proposed app, viewers will watch the debate in real time from their homes and enter their reactions through a mobile application that you will build. Your VP indicates that winning this business will be a great accomplishment for the company.



Figure 1 Screen Capture of an Existing Instant Response Focus Group

The prototype will:

* Have 3 screens (layouts for the screens is shown in the accompanying document):
  + a main screen with two buttons, one for registration and one for entering the polling screen. The main screen will also contain an image, which will be supplied. You must include this image.
  + a registration screen, with entry fields for name, email and other demographic information, as specified in accompanying image
  + a polling screen with buttons for agreeing and disagreeing with the prevailing event
* Provide an alert pop-up when the user selects *Agree* or *Disagree*, containing the users name, their sentiment and a timestamp reflecting the time of the action (see included image for an example).
* Allow the user to response with simple two-button response to a prevailing event. The user can enter as many agree / disagree elections as the event progresses.
* Be implemented with Cordova
* Integrate the data collected in ‘My Profile’ when the alert is displayed

The prototype need not:

* Implement any of the backend data aggregation. This will be done at a later time if the client agrees to move forward.
* Perform any validation of the data entered.
* Perform any registration. The form will simply collect the users input and store it temporarily in the ‘My Profile’ form’s fields.
* Look exactly like the sample screens provided
* Use jQuery or jQuery Mobile

# Rubric

*Main* Screen

5 Header with *MashLever* title

5 Subheading with *Crowdsourced Citizenship*

5 Buttons centered and operate properly

5 vote-button.png image present, scaled and centered

*My Profile* screen

5 Contains header, subheader, first name, last name and email labels with corresponding input fields present

5 radio button selector for Liberal/Moderate/Conservative

5 register button, redirects back to *Main* Screen

*Let’s Go!* Screen

5 Contain header, subheader and instructions paragraph

5 Contains *Agree* and *Disagree* Buttons

10 generates a javascript alert when *Agree* or *Disagree* is clicked

15 integrates last name, first name, email and timestamp into alert, as shown in accompanying document

5 General

Good coding practices followed, submission submitted properly and on time

# Hints

* Use an incremental programming approach when possible. Make small changes and validate successful code execution as often as possible.
* Be sure your html tags are balanced (for example, does every <div> tag have a corresponding </div> tag?).
* Be sure to save all of your files before testing
* Be sure to test all of your screens and functions before zipping and making your final submission
* If you are not successful with all of the screens and functionality, submit as much as you can. Comment out those features that aren’t operational and submit something that compiles and runs. You can place comments near any problematic areas to indicate what you were thinking, so that I can award partial credit where possible.
* The example screens use jquery’s radio button controls, described at: <http://demos.jquerymobile.com/1.2.0/docs/forms/radiobuttons/>
* No plugins are necessary, using javascript’s alert() function is adequate.
* Scaling for the ‘vote’ image in the demo screens is 40%
* Use console.log(...) for debugging. Once you’ve added console.log() statements, you will want to look at *adb logcat* output for insight into the underlying problems and errors.
* generating a timestamp with *new Date()* is fine, there is no need for any additional formatting
* although not required, you can achieve the color scheme shown in the accompanying document by adding *data-theme=”b”* attribute to the page elements.

# Submission

Zip your entire cordova project by zipping the folder containing the entire project.

# Screen Layouts

(also see the accompanying png file for screen flow)

 

 