Udeep Shakya

Using a Framework to develop Client-Side App

A Javascript Framework for cross-platform application

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme in Media Engineering

Bachelor’s Thesis

6 November 2014

Author Title Number of Pages Date	Udeep Shakya Using a Framework to develop Client-Side App 40 pages + 1 appendix 6 November 2014
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Specialisation option	JAVA and .NET Application Development
Instructors	Kari Salo, Project Manager/Principal Lecturer Petri Vesikivi, Principal Lecturer

This project aims to study the comfort of using a framework to develop client side applications based on Hypertext Markup Language 5 (HTML5), Cascading Style Sheets (CSS) and JavaScript technology. The application tends to serve both as a web client application and a mobile client application for multiple platforms. A survey answering application which fetches questions (texts) from an Application Programming Interface (API) in the application server and uploads text, sound, video and picture answer to the server was built to test the framework on.

The application serves as a client application for Contextual Activity Sampling System (CASS) developed by Metropolia University of Applied Sciences for the Doctoral Students of Department of Psychology of Helsinki University. The Doctoral students use the System to study Human behavior for different researches. Helsinki University is also a contractor for CASS client (CASS-Q) project.

AngularJS, an open source JavaScript framework, maintained by Google Incorporation is the chosen framework. It was released in 2009 and has evolved with many modules developed by the community. The results of this study suggests that the easy to learn framework can be a very good choice to solve global variable scope issue present in applications developed using most of the other JavaScript libraries.

The Doctoral students sensed a great benefit of cross-platform CASS client as compared to that of the only Android Native Client they had earlier. Though the web client developed during this project could only upload text answers, they found it already usable for their researches which required only text answers. The mobile device application versions, which has not been released yet, will able them to receive media files too.

Keywords	JavaScript framework, HTML5, CSS, AngularJS, CASS, CASS-Q

Contents

1 Introduction

Contextual Activity Sampling System (CASS) is a system for collecting data involving a process of repeated sampling of activities. It is being used by doctoral students of Helsinki University to devise a design practice by investigating human activity, social interaction, and changes in location as well as the emotional dimensions of their experiences. A PHP powered CASS admin console was built in 2008 as a backbone for the system. However, this thesis is not about the PHP CASS admin console but it is about the CASS-Q client side web application, a cross platform JavaScript application. Nevertheless, this document would be incomplete if the CASS System is not introduced. This paper documents the process of building the CASS-Q application using the AngularJS framework. [1]

CASS admin console allows user to create surveys which would be delivered to experimental subjects at selected times of the days for specified duration. Before CASS-Q web application was developed, the subjects could obtain the survey only on their mobile phones. The android client application was working fairly well, unlike iOS and Windows phone client application. Having several client applications for different platforms is costly and time consuming when the application needs to be upgraded or amended. Further, it cannot be guaranteed that a technician has the knowledge of all the different platforms. Over and above, there wasn't any client application for the desktop environment. So, it was decided to address this issue by developing a client web (HTML5) application, CASS-Q, which could be executed in any browser in the desktop environment and the mobile domain. More than that, the application would also be coupled with *phonegap* to produce native application for phones with different platforms. [2]

Hypertext Markup Language (HTML5), JavaScript, and CSS were the main technologies used to build the CASS-Q application. AngularJS, the JavaScript framework used in the project and its architecture were the main subject to study for this thesis. On the side, the capability of HTML5 was also examined.

2 JavaScript Frameworks

2.1 JavaScript, Libraries and Frameworks

JavaScript is one of the most extensively used components in web browsers today. Not only for the client side data validation, but it is being used even for designing the web application. [3,1] In fact, a whole stand-alone application can be built with JavaScript and HTML which can run on browsers even offline. Client-Side Scripting Languages like JavaScript is interpreted in the user's device (client) rather than in the server computer. It means, once the Script is downloaded to client, the execution is faster and may not need connection to server anymore.

As the JavaScript became popular, libraries and frameworks of JavaScript started developing too. The libraries consist of predefined objects or functions that can be used repeatedly. Using library features saves time as there is no need code from scratch. JavaScript libraries can also be used by server side web application to perform actions in client side. There are numerous JavaScript libraries available associated with Graphical User Interaction, Visualization, Unit Test, Template Systems, even Server Side, DOM (Document Object models) handling, and web application development [4].

A framework is not too different from libraries. An important difference between them is that the framework provides the skeleton codes for an application. It defines technical structure of the application. [2, 336] The libraries contain just reusable codes and have nothing to do with the structure of main application. The code written by developer uses the library functions. On the other side, the backbone codes of a framework use the code written by developer to build an application. A framework defines ways of coding while library is just a tool. A framework may also contain reusable features just like the libraries do. Using a JavaScript library functions to use client's computer processing could be a good choice when developing a server side application but when a whole client side application is being built, it is worth to choose a JavaScript framework.

2.2 JavaScript Framework Selection

One of the serious problems with JavaScript libraries is the lack of modular structure. The objectives of CASS-Q project was to have a modular, organized, extensible and maintainable codebase. Different frameworks address this challenge in different ways.

It is also possible that, a certain framework is better suited for a project of certain type. Of the many JavaScript frameworks; AngularJS, Ember and Knockout were the frameworks that were considered for selection before the implementation. The comparison among them is shown in Table 1.

Table 1. Comparison of frameworks [2,340]

Features	AngularJS	Knockout.JS	Ember
Learning curve and documentation	Medium learning curve and Well documented	Easy learning curve and well documented	High learning curve and
Framework/Patterns followed	Model-View-Controller, Model-View-ViewModel	Model-View-ViewModel	Pure MVC
2 Way data binding	Yes	Yes	Yes
Modularity/Code organization	Yes	Yes	Yes
Routing/Bookmarking capability	Yes	Yes	Yes
Directives	Yes	No	No
Opinionated	Yes	No	Very much
Ability to work with other libraries	Yes	Yes	Not easy
Browser support	Yes and goes back till IE8	Yes and goes back till IE6	Yes but has some issues with IE8, IE9
Maturity and size of the community	14, 000 Git Hub Watchers	4, 153 Git Hub Watchers	8, 154 Git Hub Watchers
Templating mechanism	DOM Based Templating	DOM or String-Based Templating	Uses Handlebars
Testability	5	4	4
Leadership	Google	Steve Sanderson	TILDE INC.

As suggested by the results in Table 1, the AngularJS was chosen considering its advantage in component organization, data binding ability, modularity, extensibility, browser support, easy learning curve and documentation, routing capability, ability to work with other libraries, maturity and size of the community, template mechanism, support for localization and testability. It can also be seen from the table that it offers almost all the features that other frameworks do.

3 Technologies used

3.1 HTML5

The latest version of Hypertext Markup Language (HTML5) standardizes many new structural elements including those for presenting videos and drawings in the World Wide Web. It also defines new Document Object Model (DOM) APIs (Application Programming Interface) for events like server communication and mouse events. Improved web form, Persistent Local Storage, Geolocation, Microdata, Interactivity and animation are other new features of HTML5. [5] This not only has eased developers from depending upon Third Party plugins but also assisted browsers to include open standard features to deliver more desktop like application in their interface [6,xv]. A few new features of HTML5 are introduced in this section.

3.1.1 Semantic Structure Elements

New semantic elements like `<footer>`, `<content>`, `<header>`, `<logo>`, `<container>`, `<nav>` have given more concrete structure to HTML compared to <div> element in previous HTML version. These elements have allowed repurposing the contents, for example when the web page is rendered in various sized screens. They allow easier page design and also enhance, search engine optimization for a site. Besides new elements, HTML5 also allows developers to add custom data attributes to the elements, the values of which can be accessed using JavaScript APIs or CSS. [6,1-9;7] Figure 1 shows the use of new elements to describe contents.

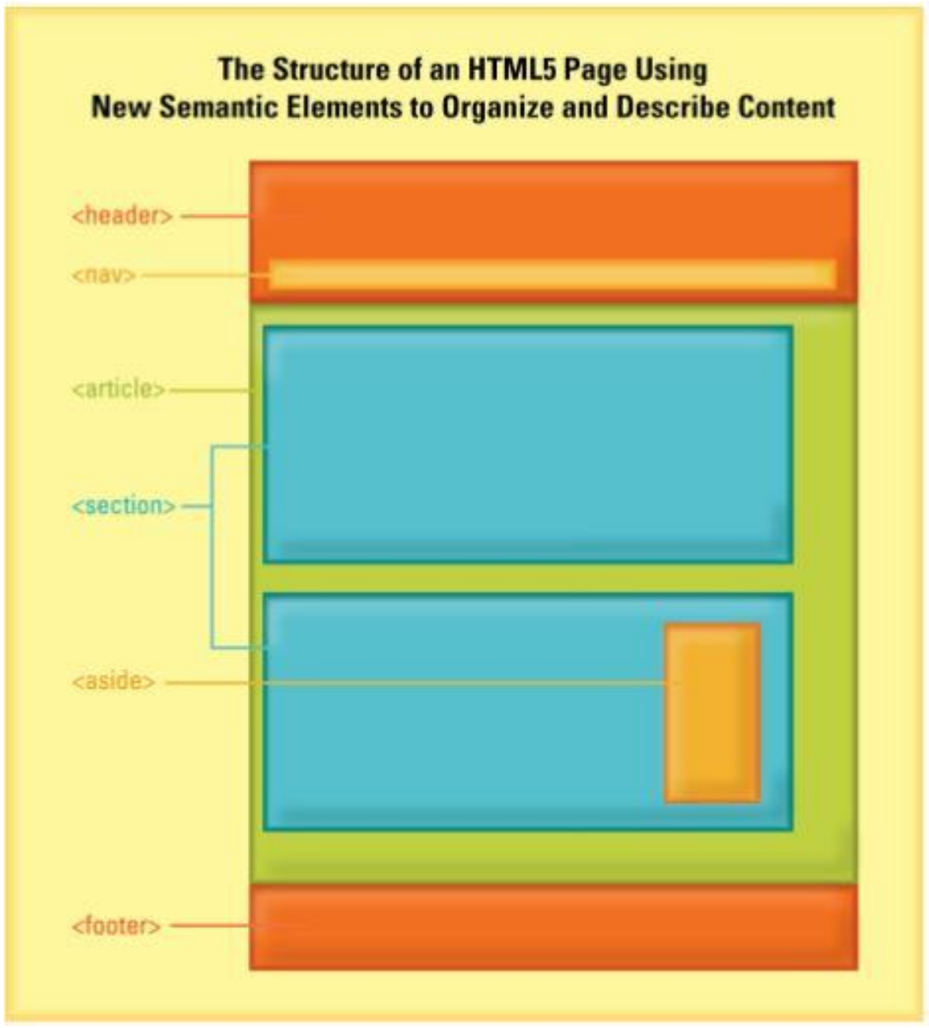

The Structure of an HTML5 Page Using
New Semantic Elements to Organize and Describe Content
<header>
<nav>
<article>
<section>
<aside>
<footer>

Figure 1. New semantic HTML5 elements. [8,10]

An HTML tag can have different meaning when it is placed inside different semantic elements depending on the hierarchy of elements shown in Figure 1 [8,48].

3.1.2 New Form Tools

Input verification which used to be done earlier with JavaScript or Server Side Script is no longer required for several new input types in HTML5. *datetime, date, month, week, time, number, range, email, url, datalists* are few of the new types that comes with the with the validation rules in the HTML5 Web Forms 2.0. In addition, custom validation criteria can also be specified. Frequently used scripted properties such as placeholder, autofocus, required are included as attributes in the HTML5. [9] As the HTML5 features takes place in the browser itself, they execute faster than traditional JavaScript. Delivering a native feel even in different devices (especially in mobile devices) is another advantage of the HTML5 form elements. However, there are compatibility issues with few browsers regarding few of the new elements. [8,120-129]

3.1.3 Video and Audio

As smart phones evolved, mobile browsing boomed too. Media player plugins that used to present audio and video in the desktop and laptop browsers for about a decade couldn't be a solution for mobile browsers. HTML5 offers native video and audio features which plays without any plugin and this eliminates the hassle of updating media player plugin frequently. [8,145] MPEG4 files with H.264 video codec and AAC audio codec is most commonly used video format for HTML5, however, the old version of most of the browsers do not support this feature. Regarding audio, *mp3* formats are played by most of the browsers along with *ogg* and *wav* formats. The <*audio*> and <*video*> tags come with attributes such as autoplay, autobuffer, loop, preload which looks after the presentation of the media. Apart from that, events generated from the player such as *play, pause, abort, error, volumechange* can be used by JavaScript to control functionalities. [10]

3.1.4 Data Storage

Session Storage and local storage are two new storages introduced at client side in HTML5. Earlier versions had cookies as client side data storage which is limited to size of 4KB. Cookies also slow down the application and makes the communication vulnerable as it is included in every HTTP (Hypertext Transfer Protocol) request which is unencrypted. [11] The size of new data storages depend on the browsers. A limit of five megabytes for each storage area is recommended by W3C which is quite large for texts only. [12] Session storage stores data for only one session while local storage is persistent until the user or the script deletes the data explicitly. This development is especially very useful in cases where users are not connected to internet all the time when they are using the browsers. Especially in case of mobile phones, the connection cannot be always reliable. Local storage has also allowed users to browse web application even when they are offline. There are already HTML5 games which can be played offline in the browsers. [13,278]

3.2 AngularJS and JavaScript libraries.

AngularJS is the framework technology used in the project. It uses the scripting language JavaScript which fills in with the HTML technology to build client side web applications. The HTML is a strong declarative language but it lacks the capability of creating applications. Google Inc, the developer of the framework boasts AngularJS as what the HTML would have been if the technology was designed for applications. Using AngularJS, HTML’s syntax can be extended to articulate components of an application. Angular’s data binding capability and dependency injection feature helps reducing lots of codes. AngularJS can be considered a complete client side solution. [14]

A few components of other JavaScript libraries like jQuery, number polyfill, Cordova (PhoneGap framework) are also used. The designing technology, Cascading Style Sheet (CSS) is used for presenting elements in web. The CSS and JavaScript of the Bootstrap library is used for responsive design in this project. A version control system, Git is used for managing code versions during the development. Bitbucket (www.bitbucket.org) was used to save the project in the cloud.

AngularJS and APIs used from other libraries is discussed in detail in the later section of this document.

4 The Project

4.1 The Architecture

This section discusses the technical aspect of the Cass-Q project. Cass-Q application communicates with CASS backend application to download survey questions and upload text answers using Hypertext Tranfer Protocol (HTTP). The data is transferred in XML format. The CASS backend is hosted in Amazon server which can be accessed using web browsers in a local machine. The CASS-Q client app can be accessed in the same way but in addition, it also has a hybrid version for mobile devices. The system architecture is shown in Figure 2.

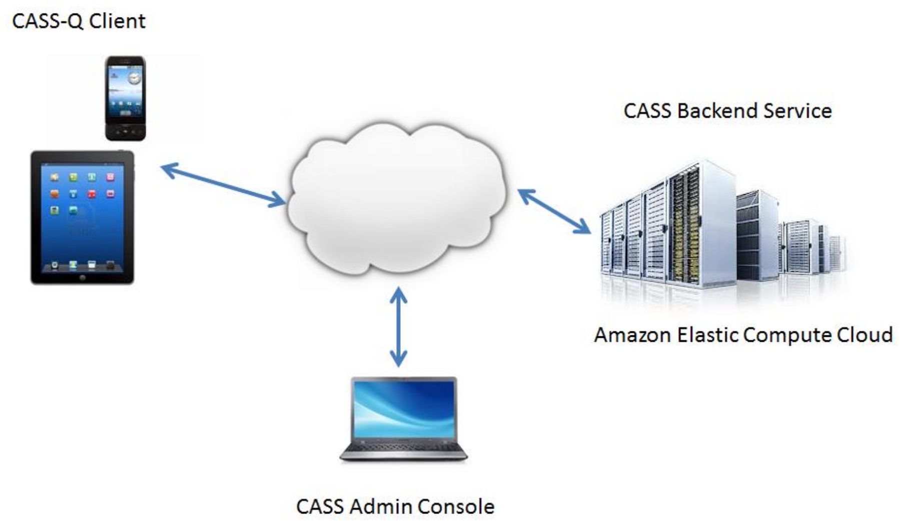


Figure 2. CASS architecture [2,335].

The hybrid apps produced by connecting web app with PhoneGap library are sometimes referred as native apps in some articles [15,17]. Figure 3 shows the plan of hybrid CASS-Q system.

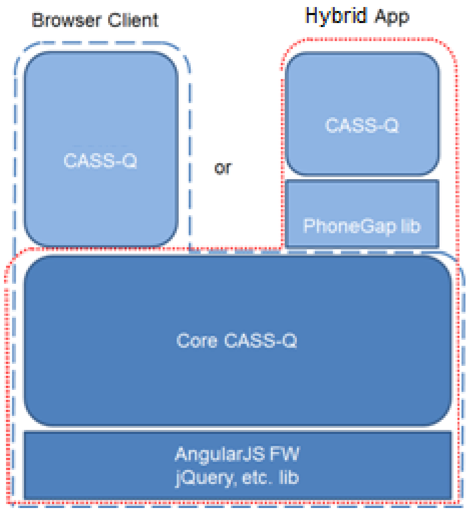


Figure 3. CASS-Q plan [2,342]

The PhoneGap library contains the APIs which can be used by the web-application to access the device features.

4.2 Data models

The customers of the CASS application wish the client application to run offline even for several days. At the moment, the current version of backend does not provide enough information to develop such client. At present, the application must connect online every time to get a survey and the answers should be sent to the server before fetching another survey. The persistent storage is not used for survey questions and answer at this stage.

The data model is designed considering a possibility that the client application can develop further into an application which could connect to different question server depending on client’s choice, fetch more than one survey (queries) at a time and upload answers for more than one survey in batch. With more information from the backend and more processing at the client-end, the application can be transformed into better offline application in the future. The use of persistent storage like IndexedDB or local storage for survey information would be required for that purpose. The data model looks like as it is shown in Figure 4.

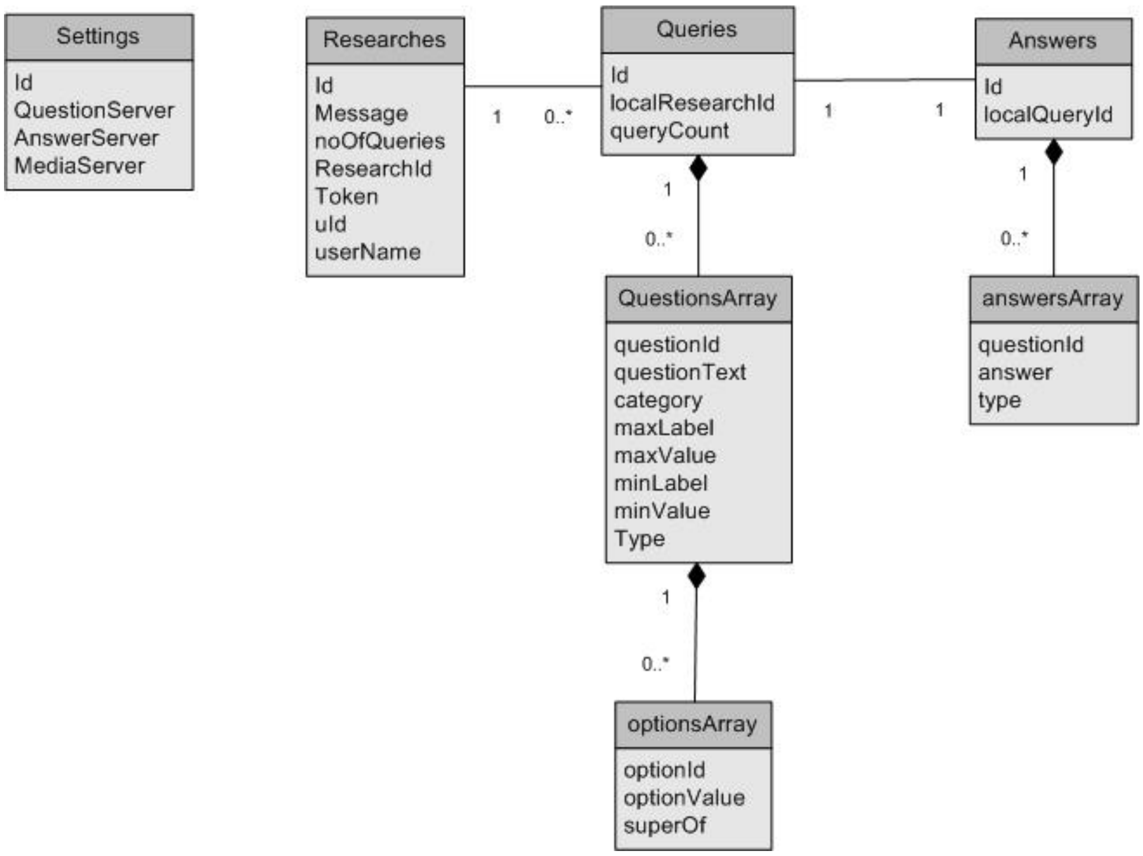


Figure 4. Data models for CASS-Q.

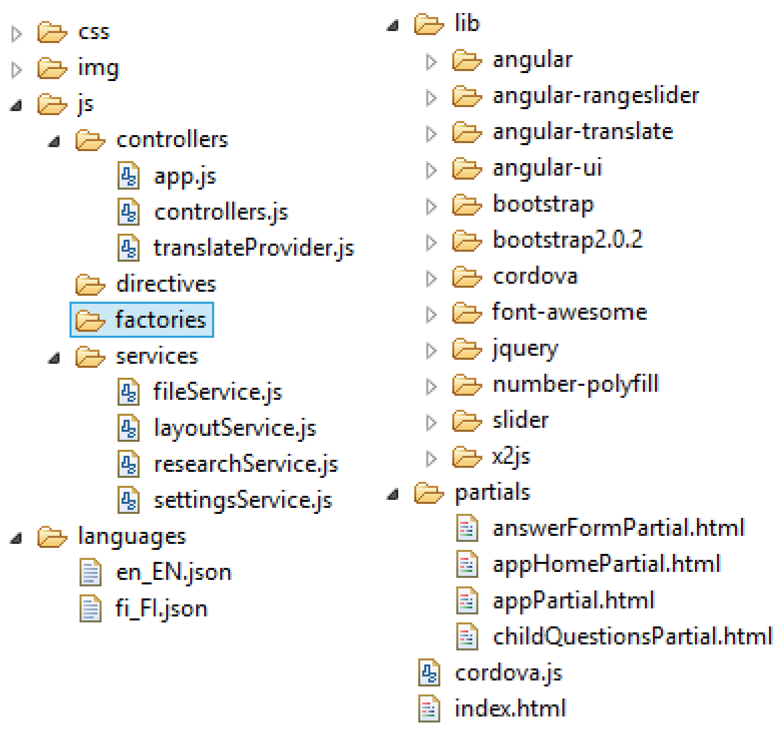
The data is modeled as JavaScript objects in this project. The entities shown in the Figure 4 are objects containing array or collection of objects. The settings object contains server information. At present, only one setting is hardcoded in the application as there is only one instance of server the researchers in Helsinki University are using. If they use more than one instance of CASS backend in the same or different server in future, the settings model can be used to store information of multiple setting models.

Research object in the collection Researches is associated to Query object by “Id” key in Research object which is mapped as *“localResearchId”* in Query object. The Query object contains Array of Questions which further contains Array of options. Answers are stored as Answer object in the collection Answers. It is associated with a Query object by *“Id”* key of Query object and *“localQueryId”* in Answer object. The answer object contains array of answers for the questions in a query.

5 The AngularJS in the project

5.1 Directory Structure

As AngularJS is a structural framework [16]. The directories in an AngularJS project can be arranged in a very comprehensible way. Figure 5 shows the directory structure of the CASS-Q application.



```
▷ css
▷ img
◢ js
    ◢ controllers
          app.js
          controllers.js
          translateProvider.js
      directives
      factories
    ◢ services
          fileService.js
          layoutService.js
          researchService.js
          settingsService.js
◢ languages
      en_EN.json
      fi_FI.json
◢ lib
    ▷ angular
    ▷ angular-rangeslider
    ▷ angular-translate
    ▷ angular-ui
    ▷ bootstrap
    ▷ bootstrap2.0.2
    ▷ cordova
    ▷ font-awesome
    ▷ jquery
    ▷ number-polyfill
    ▷ slider
    ▷ x2js
◢ partials
      answerFormPartial.html
      appHomePartial.html
      appPartial.html
      childQuestionsPartial.html
  cordova.js
  index.html
```

Figure 5. CASS-Q directory structure.

In an Angular application, it is very convenient to locate custom functions such as controllers, directories, factories, services, filters in the *'js'* folder. These codes are related to the CASS-Q application and are used as fillings by the Angular Framework codes which are contained in the folder *'lib'*. The *'lib'* folder is also home to library files and modules developed by other developers.

Folder *"Partials"* contains the HTML templates which are displayed to the clients. A template and its respective controller along with its dependencies are used by the application controller to present a view to a client when a user request certain route in the application. All the templates in “Partials” folder are partial views for the main html file *“index.html”* in the root folder.

5.2 Application module

A module specifies how an application should be loaded or how an application should function. It helps to keep the global context clean as the data as the functions defined inside a module cannot be accessed from global namespace. It makes writing tests easier. Modules are reusable. It can be shared between applications. [17,18;18] A module can contain various parts or functions as shown in figure 6.

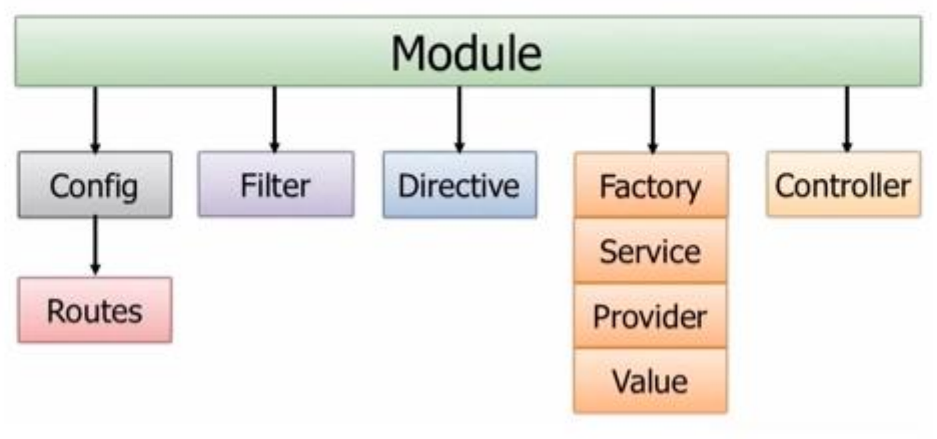


Figure 6. Module as a container of functions [19,57]

The parts of a module shown in Figure 6 are introduced in next section of this document but it would be sensible to mention *ng-app* directive of angular framework because the use of this directive bootstraps the module to a part of HTML document.

> Directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's HTML compiler (`$compile`) to attach a specified behaviour to that DOM element or even transform the DOM element and its children. [20]

ng-app directive is a built-in Angular directive which makes the web application an Angular application. A custom module can also be connected with *ng-app* directive. As shown in Figure 6, this custom module could then contain configuration for routes, custom filters, directives or functions like Factory, Service, Provider, and Value which could serve data from different sources. [19,57] An application module is defined in the way shown below for this project.

```
var cassApp = angular.module('cassApp', ['ngRoute','ngResource','ui-rangeSlider','pascalprecht.translate']);
```

cassApp mentioned between the quotes in above code is the name of the module and the ones mentioned in the square brackets are the dependency modules for cassApp. The HTML file is coupled to angularJS by the following code in 'index.html' file in this project.

```
<html lang="en" ng-app='cassApp'>
```

Angular uses spinal-case for its custom attributes but corresponding directives which implement them are named using camelCase. Only the portion inside the DOM element where *ng-app* directive is defined is treated as Angular application. This coupling loads the module and allows the segment of the HTML to be manipulated by the functions and properties. [21]

5.3 Routing

Config function of app module is used to create routes in the app as in the code below.

```
cassApp.config(function($routeProvider) {
    $routeProvider.
            when('/',
                    {
                        templateUrl: 'partials/appHomePartial.html',
                        controller: 'AppController'
                    }).
            when('/questions',
                    {
                        templateUrl: 'partials/appPartial.html',
                        controller: 'AppController'
                    }).
            when('/answerform',
                    {
                        templateUrl: 'partials/appPartial.html',
                        controller: 'AppController'
                    }).
            otherwise({
                templateUrl: 'partials/appHomePartial.html',
                controller: 'AppController'
            });
});
```

In the above configuration, the ‘when’ method is used to add specific routes, while ‘otherwise’ method points to the default route. The first parameter taken by the when method is the path that appears on the URL address. The second parameter is an object which contains the template (a view) the application should access and the configuration object which can be properties like a *controller, template, templateURL, resolve or redirectTo* and *reloadOnSearch*. [17,138] The DOM element, which would contain the partial template, should contain the directive ‘*ng-view’*.

```
<div ng-view class="container slide {{layout}}" id="cass-content-
container">
```

The routes can be changed in the controller using *$location* object. For instance, the code below directs the application to the URL containing *‘/questions’* to the root address.

```
                    $location.path('/questions');
```

Routing therefore is one of important settings in an Angular app as it connects the several properties of Angular App. Figure 7 shows a big picture of how the various functions and properties are related in a module.

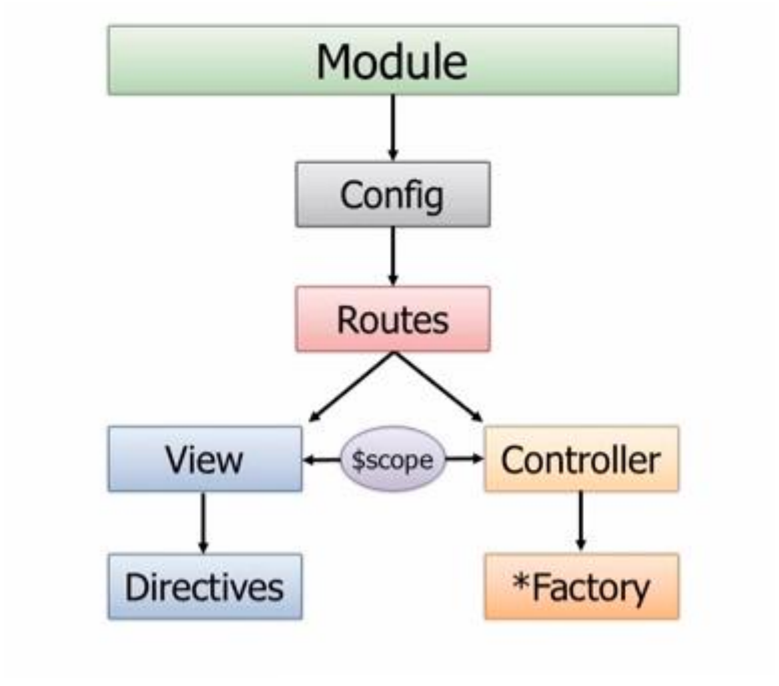


Figure 7. Big picture of an Angular Module [19,55]

A module can contain configuration function which defines the routes. A route specifies a view and a controller for a path. The View and Controller uses the *$scope* object for two way data binding. *$scope* object is the View Model. A View uses a Directive or Filter to access an angular function or property in the View. The Controller uses factories, services, provider or value to access data from data sources.

5.4 Model, View and Controller

Model, View and Controller (MVC) architecture allows the developer to keep the presentation logic, business logic and database interaction separate. A view is a projection of HTML Template which contains presentation logic. Models are the data displayed in the Views. The Controller is an instance of JavaScript object which contains the business logic. It loads Models from different data sources and provides them to the View. AngularJS offers automatic data binding, meaning that any change in the model happening in the controller would be automatically reflected in the View. The feature also works the other way from the view to the database interaction logic via the controller. $scope object or simply called Scope, is glue between View and Controller and plays an important role in data binding. It is also referred to as ViewModel by some developers. [17,11-13;19,46]

There are two controllers defined in the cassApp module – the application controller and the language controller.

```
cassApp.controller('AppController',function($scope, $route, $http,
$window, $location, layoutService, settingsService, researchService,
fileService, $filter) {
            $scope.changeToken = function() {
                if ($scope.Token != null && $scope.Token != "")
                {
                    researchService.setCurrentToken($scope.Token);
                    $scope.fetchXmlMakeObjNavigate();
                }
            };
```

In the above code, a controller function called '*AppController*' is defined in the '*cassApp*' module. Necessary dependencies are listed as parameters while defining controller function. In above case, they are custom data providers (*layoutService, settingsService, researchService, fileService*), angular services (*$route, $http, $window, $location, $filter*). The scope object must be listed as one of the parameters if the controller function needs to augment the ViewModel. As shown in above code, the controller function can create or change properties (a function or value) in Scope object which can be used by View Template.

Controller can be assigned to a DOM section of the HTML using ng-controller directive. An angular application can use more than one controller.

```
<body ng-controller="AppController"  >
<div ng-controller="langController"  style="margin-bottom: 42px; text-
align: center; position: fixed; bottom: 0">
 <div id="contentarea" class={{layout}}>
 <form ng-submit="changeToken()"
  <input type="text" ng-model="Token"  >
  <span ng-if='currentToken' >Current Token = {{currentToken}}
  </span>
</div>
</body>
```

If the Scope properties needs to be passed to non angular features, for instance to be viewed in the HTML page or to DOM attributes, they are enclosed by double curly braces. Curly braces denote bindings. In above example, `{{layout}}` and `{{currentToken}}` are properties of scope object and are bound to reflect to any changes that would occur anywhere in the module. A view can access properties of scope object using any angular or custom directive such as *ng-submit, ng-if, ng-click*. A view can create or change a property of the scope by using *ng-model* directive. In above snippet of code, a text input field is bound to Scope property “Token” by the *ng-model* directive.

While discussing about scope, it should be noted that it is not just the controller which can have its scope. *ng-app* directive and *ng-repeat* directive also creates their own scope. Figure 8 shows scope inheritance in an Angular application.

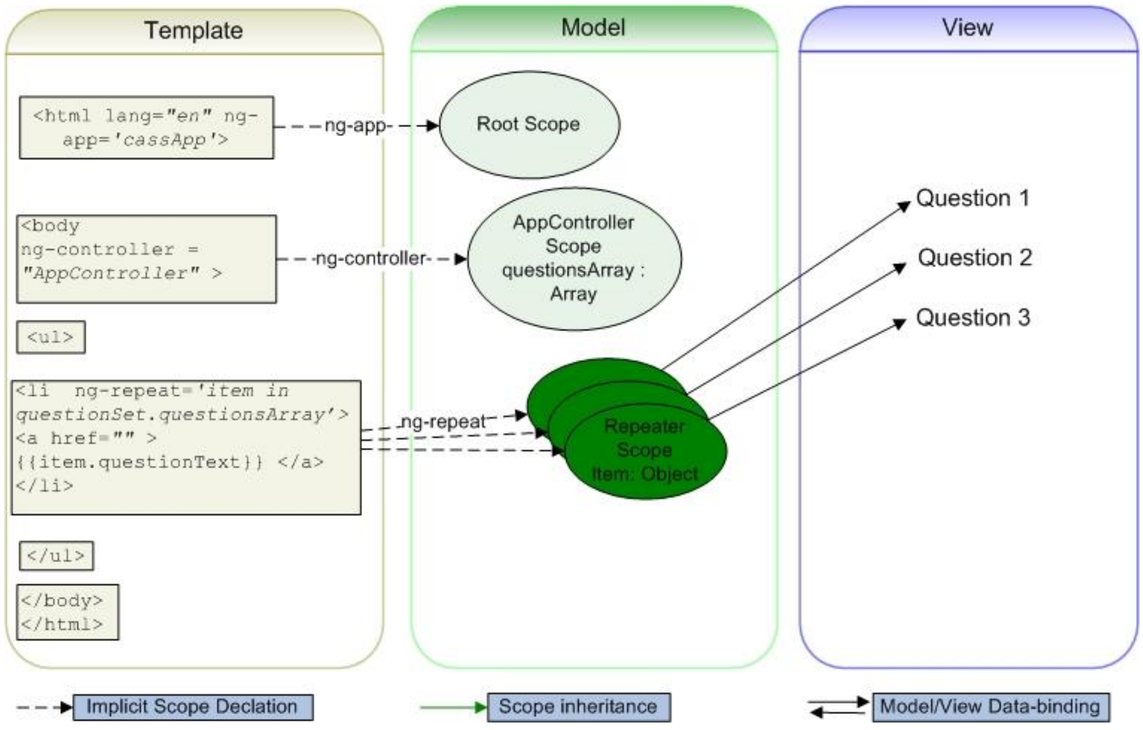

Template
Model
View
<html lang="en" ng-app='cassApp'>
ng-app
Root Scope
<body ng-controller = "AppController" >
ng-controller
AppController Scope questionsArray : Array
<ul>
<li ng-repeat='item in questionSet.questionsArray'> <a href="" > {{item.questionText}} </a> </li>
ng-repeat
Repeater Scope Item: Object
Question 1
Question 2
Question 3
</ul>
</body> </html>
Implicit Scope Declation
Scope inheritance
Model/View Data-binding

Figure 8. Scope inheritance [22]

Every angular application has a single root scope created by *ng-app* directive. The properties in root scope can be created or accessed by using *$rootscope* from any of its child scopes. However, using *$rootscope* as a viewmodel is not recommended as it pollutes the JavaScript global scope with too much data and logic. All other scope descends from the application scope. In the above figure, repeater scopes are child scopes of the controller scope. A property '*questionsArray'* is an array in the Controller Scope but each array element is projected as an object 'item' in corresponding repeater scope. The properties of Controller Scope can be accessed in repeater scope by using keyword '*parent'*. However, there is no way to access properties defined in a child scope from a parent scope. [17,21;23,24]

The other controller defined in the *cassApp* module is the language controller, created mainly to make *'$translate'* filter change the language to the one chosen by a user. More about this feature is explained in localization section of this document.

```
cassApp.controller('langController', function($scope,$translate){
    $scope.changeLanguage = function(lang_code){
        $translate.use(lang_code);
    };
});
```

5.5 Recipe

Factory, Service, Provider and Value encapsulates data functionality and are called Recipes in AngularJS. They can be injected as a parameter when controllers or other Recipes are created. And this is how they provide data to multiple functions. The Recipes are singleton objects that are created just once and are cached for all future needs. Recipes might be interacting with a database or a XML/JSON request or a file containing XML/JSON as a data source. Factory, Service and Provider serves common functionality but the way they create the object that fetches the data are different. [19,18; 25]

- In case of the **factory**, an object is created inside it and is passed to the controller. A factory can have dependencies to other Recipes.
- In case of the **service**, the service itself is an object. The standard functions and properties are defined in the service using **this** keyword.
- In the **Provider**, a $get property is used to get the object that provides the data.
- A **value** is a way of defining a constant. Unlike the previous three, it can’t contain any functions.

Four services are used in this project.

```
cassApp.service('settingsService', function() {
});
```

settingsService contains functions to insert, delete and provide a setting that would contain URLs to question Server, answer Server and media Server that the application will communicate to.

```
cassApp.service('layoutService', function() {
});
```

layoutService checks whether the application is running in a browser or as a hybrid application and returns a value to the controller to change to the application layout according to the screen size.

```
cassApp.service('researchService', function($http, $window, set-
tingsService, fileService, $location, layoutService, $filter) {
});
```

researchService is the main recipe of the application which contains the main logic like making objects of the xml data received from the Question Server. It also contains functions which saves answers to the object and upload answer to the answer server.

```
cassApp.service('fileService', function(settingsService) {
});
```

fileService contains functions which create names for media files, add files to an object, and upload the media files to the Media server.

5.6 Communicating with the Backend

The client-side CASS-Q application contains logic to show questions contained in a query and obtain the input as answers for them. It cannot save the questions or the answers at the client side. It has to communicate to Backend every time to fetch questions and upload answers to save the answers in the Backend. Among the several methods offered by AngularJS to communicate with a remote server, a simple service $http is used for this project.

5.6.1 $http Service

In a simple HTML/JavaScript web application, browser supported XMLHttpRequest (XHR) object would be used to upload or download data to remote server. *$http* is a core angular service which is just a wrapper function around the XHR object. It is an asysnchronous method, meaning that it doesn’t wait for the result from server to execute the next code of the application. *$http* service takes an argument which should be a configuration object that would be used to produce an HTTP request. The configuration object may contain keys like *method, url, params, data, headers, timeout, reponse type* depending upon the method and necessity of the request. If the HTTP request is fulfilled, a function in Success method executes. If the HTTP request fails to connect to a server then the error method is returned. [17,173;26]

```
$http({
method: "GET",
url: settingsService.getSetting().questionServer + token //URL
}).success(function(data) {
     researchService.makeObjects(data);
     if ($scope.isQuestionPage == true) {
     $route.reload();
     }
     else
     $location.path('/questions');
}).error(function(data, status, headers, config) {
     $window.alert($filter('translate')('CONNECTION_ERROR_INFO'));
});
```

Above code from CASS-Q application sends a GET request to the backend to download a query. The query is downloaded as a parameter '*data'* to the success function. To send answers to the server, a POST method of *$http* service is used as follows.

```
$http({
    url: settingsService.getSetting().answerServer,
    method: "POST",
    data: dat,
    headers: {'Content-Type': 'text/xml; charset=ISO-8859-1'}
  }).success(function(response, status, headers, config) {
    $window.alert($filter('translate')('ANSWER_SENT_ALERT'));
    $location.path('/');
  }).error(function(data, status, headers, config) {
    this.msgfromServer = "Error - " + status;
    $window.alert($filter('translate')('SUBMIT_ERROR') + status);
  });
```

5.6.2 XML Handling

Had the backend used JSON for data transfer, the JavaScript would automatically load the JSON to JSON object when the XHR request succeeds and Angular would resolve it to corresponding JavaScript objects automatically. But the CASS backend sends the data in an XML format and Angular requires JavaScript objects to work with. Therefore, and external XML parser is required. [17,217-218] An open-source library "X2JS" library [27] is used to parse XML to JavaScript in this project.

```
        <script src="lib/x2js/xml2json.js"></script>
```

The XML sent from the backend contains all the data which is delivered in single XHR request. Shown below is one of such XML response.

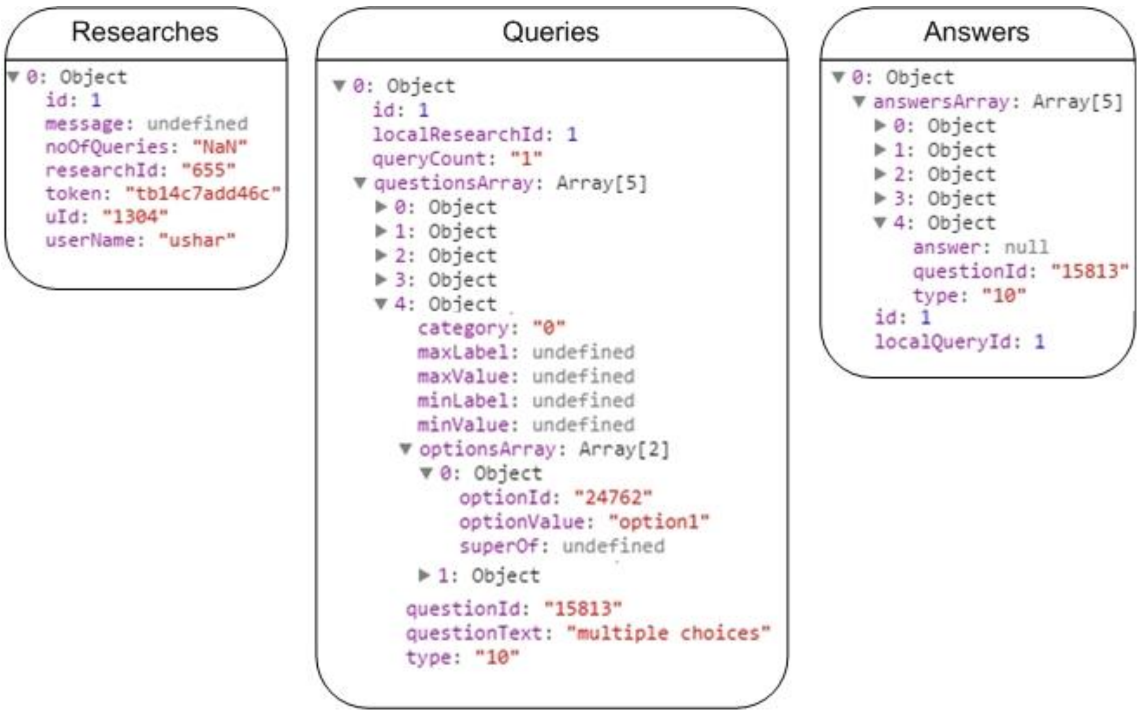
```
<survey username="ushar" uid="1304" surveyId="655"
surveyCount="1" surveyTotal="NaN">
<item category="0" type="1" q_id="15525">open text question</item>
<item category="0" type="2" q_id="15526" min="5" max="9">open num-
ber</item>
```

```
<item category="0" type="9" q_id="15527"
min="1" max="10" minlabel="cold" maxlabel="hot">slider question</item>
<item category="0" type="7" q_id="15528">add a picture</item>
<item category="0" type="10" q_id="15813">
multiple choices
<option value="option1" o_id="24762"/>
<option value="option2" o_id="24763"/>
</item>
</survey>
```

The X2JS library parses the above xml to one JavaScript object.

```
var x2js = new X2JS();
var wholeObject = (x2js.xml_str2json(decodeURIComponent(xml))).survey;
```

The angular way of doing this would be creating x2js object in a separate factory or service dedicated only for parsing xml [17,218]. The JavaScript object created by the parser is then split to three objects (*Researches, Queries, Answers*) as proposed by the data models for this application. The objects are shown in Figure 9.



Researches

```
▼ 0: Object
    id: 1
    message: undefined
    noOfQueries: "NaN"
    researchId: "655"
    token: "tb14c7add46c"
    uId: "1304"
    userName: "ushar"
```

Queries

```
▼ 0: Object
    id: 1
    localResearchId: 1
    queryCount: "1"
  ▼ questionsArray: Array[5]
    ▶ 0: Object
    ▶ 1: Object
    ▶ 2: Object
    ▶ 3: Object
    ▼ 4: Object
        category: "0"
        maxLabel: undefined
        maxValue: undefined
        minLabel: undefined
        minValue: undefined
      ▼ optionsArray: Array[2]
        ▼ 0: Object
            optionId: "24762"
            optionValue: "option1"
            superOf: undefined
        ▶ 1: Object
      questionId: "15813"
      questionText: "multiple choices"
      type: "10"
```

Answers

```
▼ 0: Object
  ▼ answersArray: Array[5]
    ▶ 0: Object
    ▶ 1: Object
    ▶ 2: Object
    ▶ 3: Object
    ▼ 4: Object
        answer: null
        questionId: "15813"
        type: "10"
    id: 1
    localQueryId: 1
```

Figure 9. Three JavaScript object stores constructed from XML.

Just like the download format, the upload format for CASS backend is also an XML. A custom XML maker is created to generate XML text from JavaScript object store ‘*Answers*’.

5.6.3 Cross Origin Resource Sharing (CORS)

The client side CASS-Q application is independent of Backend CASS application except for the exchange of data which requires running a script at the backend. It should be possible to host the client application from any domain (Internet Protocol address) and communicate with Backend which can be originating from other domain. However, normally the web browsers do not allow fetching and executing scripts on cross domains unlike in the same origin policy. The same-origin policy allows the browsers to run scripts on pages that are hosted from same domain. The CASS-Q web application is at the moment hosted from same origin as the Backend application, so no CORS functionality is implemented. However, it is worth to have an idea of applying CORS methodology in AngularJS in case the hosting environment changes. [17,211]

The CORS specification allows JavaScript to make cross-domain XHR calls. It actually sends a preflight to the foreign server and asks for the permission to send the request. The server can accept or reject any request from all servers, a select server, or set of servers during the preflight. The client and server application both needs to have coordinating preflight settings to transfer data to either side. In an Angular app, cross domain (XDomain) feature is set to true to in the *config()* method of the application module so that the application can use the feature.[17,211-213] An example is shown in code below.

```
angular.module('myApp')
.config(function($httpProvider) {
$httpProvider.defaults.useXDomain = true;
delete $httpProvider.defaults.headers
.common['X-Requested-With'];
});
```

Code example in ng-book, page 213

In the above code the configuration setting makes sure that X-Requested-With header is removed from all XHR calls. It has been removed from the common header defaults, but it is recommended to ensure in the configuration to make CORS possible.

A simple setup is also required at server side too to support the CORS. The file at the server which the client app is communicating to, must respond with few access control headers to the preflight request as shown below. [17,214-216]

```
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: GET,POST");
header("Access-Control-Allow-Headers: X-Requested-With, Content-Type\n");
```

Code example in CASS Backend PHP application

In the above code, The `Access-Control-Allow-Origin` header is set to `*` which means it allows any kind of requests from any origin. If it needs to allow a particular foreign domain only, then the domain is mentioned instead of `*`. The value of `Access-Control-Allow-Headers` should be same as the header type in the XHR request made by the client side. The server must respond with `Access-Control-Allow-Methods` header if client makes request with `Access-Control-Request-Headers` header. It lists the allowed HTTP methods. Use of this header allows the request to be cached in the client which makes preflight not required for future requests. [17,214-216]

5.7 Presentation

5.7.1 Localization

An AngularJS module, angular-translate [28] allows to switch languages in the webpage on the fly in the runtime, without a need to refresh the page or load different URL. It reduces the work of building new templates for different languages. The module is not contained in core angular framework, so it should be embedded in the HTML document manually. [17,482]

```
<script
src="lib/angular-translate/angular-translate.min.js"></script>
```

The angular-translate module is declared as pascalprecht.translate in its definition file. Pascal Precht is an active member of the AngularJS community and the developer of the module. Angular-translate library must be mentioned as dependency module to be loaded in the application module.

```
var cassApp = angular.module('cassApp', ['ngRoute','ngResource','ui-
rangeSlider','pascalprecht.translate']);
```

The application is taught a new language by injecting *$translateProvider* in the *config()* function of the app. *$translateProvider* is a Provider in angular-translate module which is used to configure translation data. There are different ways to load translation data but the one that has been used in the CASS-Q app is shown below.

```
cassApp.config(['$translateProvider', function($translateProvider) {
        $translateProvider.preferredLanguage('en_EN');
        $translateProvider.useStaticFilesLoader({
            prefix: 'languages/',
            suffix: '.json'
        });
```

Code example in CASS-Q application

$translateProvider can load a separate file which contains translation table using `useS-taticFilesLoader` function as in the case in above code. This helps to keep the translations for each language in a separate file. The prefix and suffix attributes passed to the file loader function forms the path to the file containing translation table in JSON format as shown in below.

JSON data in the file *languages/en_EN. json*

```
{
        "GET_SURVEY_BUTTON": "Get Survey",
        "WELCOME_TEXT": "Welcome to CASSQ",
        "FETCH_SURVEY_TEXT": "Enter a token to fetch a survey"
}
```

JSON data in the file *languages/fi_FI. json*

```
{
        "GET_SURVEY_BUTTON" : "Hae kysely",
        "WELCOME_TEXT" : "Tervetuloa CASS-Q:n",
        "FETCH_SURVEY_TEXT" : "Anna Token kyselyn hakua varten"
}
```

The key in the JSON represents translation ID and the value represents the translated text for a particular language. Translation ID and `translate` filter that comes within the angular-translate module is used in the view template to show the concrete texts.

```
               <div >
                {{ 'WELCOME_TEXT' | translate }}
                <br/>
                {{ 'FETCH_SURVEY_TEXT' | translate }}

            </div>
```

Figure 10 shows CASS-Q application using translation texts to present the application in Finnish language.

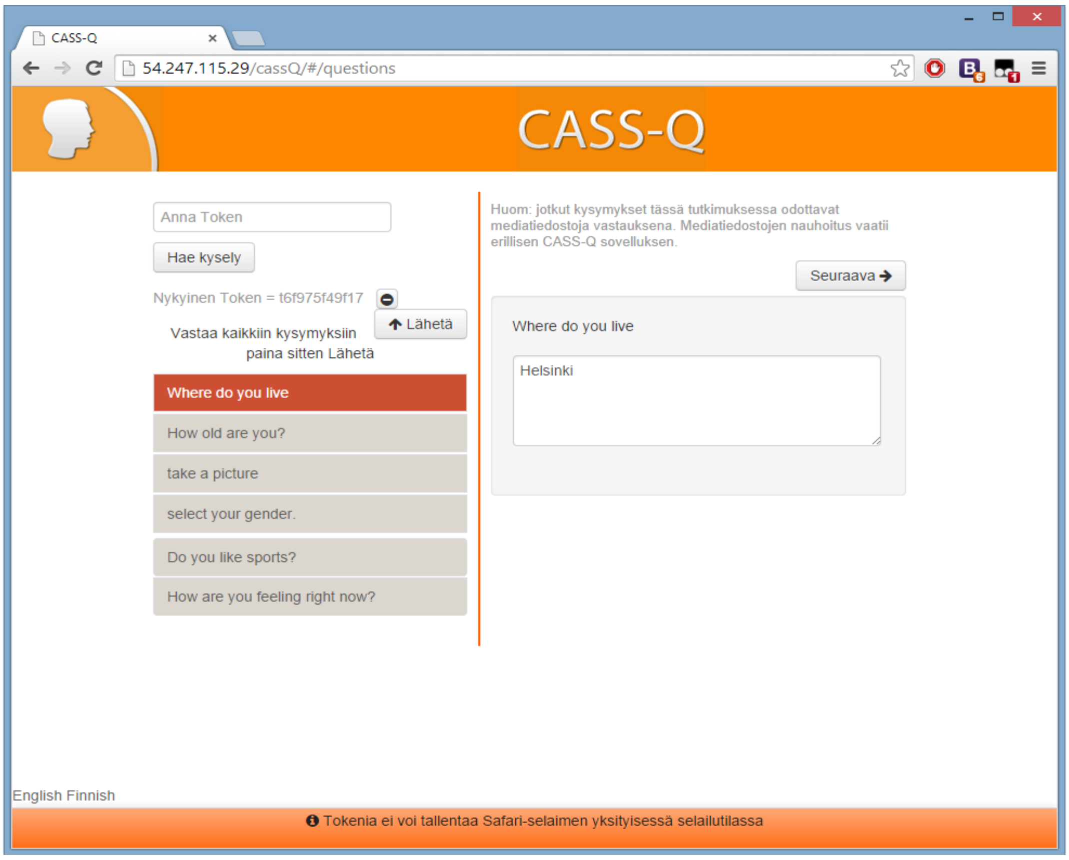

CASS-Q
54.247.115.29/cassQ/#/questions
CASS-Q
Anna Token
Hae kysely
Nykyinen Token = t6f975f49f17
Lähetä
Vastaa kaikkiin kysymyksiin paina sitten Lähetä
Where do you live
How old are you?
take a picture
select your gender.
Do you like sports?
How are you feeling right now?
Huom: jotkut kysymykset tässä tutkimuksessa odottavat mediatiedostoja vastauksena. Mediatiedostojen nauhoitus vaatii erillisen CASS-Q sovelluksen.
Seuraava
Where do you live
Helsinki
English Finnish
Tokenia ei voi tallentaa Safari-selaimen yksityisessä selailutilassa

Figure 10. Angular-translate used to change language in the application

Switching the language in the run time requires a function to be implemented in a controller. It is actually *use()* method of $translate service of angular-translate module which performs the action.

```
cassApp.controller('langController', function($scope,$translate){
    $scope.changeLanguage = function(lang_code){
        $translate.use(lang_code);
    };

});
```

The function can then be called from a view when a user clicks a button or a link. A language code is passed to the function when it is called as shown in the code below. [17,482-488]

```
<div ng-controller="langController"  ><div ng-cloak>

  <a href="" ng-click="changeLanguage('en_EN')">English</a>
  <a href="" ng-click="changeLanguage('fi_FI')">Finnish</a>
</div>
```

The raw Angular html template can be briefly displayed while switching to a different language in the browser. This happens when the compilation of JavaScript is slower than the compilation of HTML. *ngCloak* directive of AngularJS solves this problem by hiding the raw content when the transition takes place [29].

5.7.2 Polyfills

HTML5 has considerably evolved compared to its predecessor. However, a few new elements of HTML5 are not supported in old browser. At the time of this project implementation, it was noticed that the new elements didn’t work as it was supposed to work even in few latest browsers. The delivery of the features is important for the mobile browsers in the touch devices for the CASS-Q application. The performance of the HTML5 elements that are needed for CASS-Q application was tested in several browsers during the project and the result is summarized in Table 2.

Table 2. HTML5 element test results. [2,343]

HTML5 element	Device/OS	Browser	Test result	Notes
Number Input	Windows PC	Firefox 26.0	Failed	HTML5 number type delivered as normal text box
	Windows PC	Chrome 32.0	OK	
	Windows PC	IE 10.0	Failed	HTML5 number type delivered as normal text box
	iMAC	Safari 6.0.5	OK	
	MacBook Pro	Safari 7.0.1	OK	
	Samsung Galaxy S4	Native browser	OK	
	Nexus 7	Chrome32.0	OK	
	iPhone 5s iOS 7.0.5	Safari	Failed	HTML5 number type delivered as normal text box
	iPad mini iOS 7.0.5	Safari	Failed	HTML5 number type delivered as normal text box
	Nokia Lumia 920 win 8	Internet Explorer	Failed	HTML5 number type delivered as normal text box
Range Input	Windows PC	Firefox 26.0	OK	
	Windows PC	Chrome 32.0	OK	
	Windows PC	IE 10.0	Failed	Input delivered by user is not acknowledged
	iMAC	Safari 6.0.5	OK	
	MacBook Pro	Safari 7.0.1	OK	
	Samsung Galaxy S4	Native browser	OK	
	Nexus 7	Chrome32.0	OK	
	iPhone 5s iOS 7.0.5	Safari	Partial	The slider in range input doesn't respond touch readily
	iPad mini iOS 7.0.5	Safari	OK	
	Nokia Lumia 920 win 8	Internet Explorer	Failed	Input delivered by user is not acknowledged

Polyfill fills the gap between the browser and the elements not supported by it, meaning that it helps to deliver the functionality that the browser is supposed to deliver. It is just a piece of codes (usually JavaScript/CSS) that could be added in a web application. An ideal polyfill would take care of backward compatibility in browsers in a quiet way without the developer needing to work around. There are many open source lightweight polyfills available for a particular input element. [30;31,258;32]. This sort of polyfill is exactly suitable for CASS-Q application.

An open source Number Polyfill written by Jonathan Stipe available in GitHub system [33] is used in the CASS-Q project. It is very simple to use. The HTML page just needs to include `number-polyfill.js` file as its helper file and `<input type="number" />` element can be used without any bother. The polyfill comes with a default CSS file which can be edited to style the looks.

Polyfills are easy to use but AngularJS provides another tool too to augment the HTML elements. A directive called *angular-rangeslider* is used to style the Range input in the application. The module is developed by Daniel Crisp and should be separately installed. This directive actually makes `<div>` element to show a range slider and not `<input type =”range”>` element. The *angular-rangeslider* can have 2 handles form minimum and maximum values. The values can be attached to modal in the angular scope. The slider can be displayed in vertical or horizontal orientation. Having the slider orientation in vertical orientation is mainly useful in touch devices as the activity doesn’t collide with the swipe activity. The HTML page should include a file *angular.rangeSlider.js* and the application module must list range-slider module as its dependency when it is loaded. jQuery library is also required. [34] The codes used for range-slider directive is shown below and its output is shown in Figure 11.

```
<div range-slider min="selecteditem.minValue"
max="selecteditem.maxValue" orientation= "vertical" decimal-places="1"
model-max="selecteditem.answer" pin-handle="min" >
</div>
```
Code example from CASS-Q application

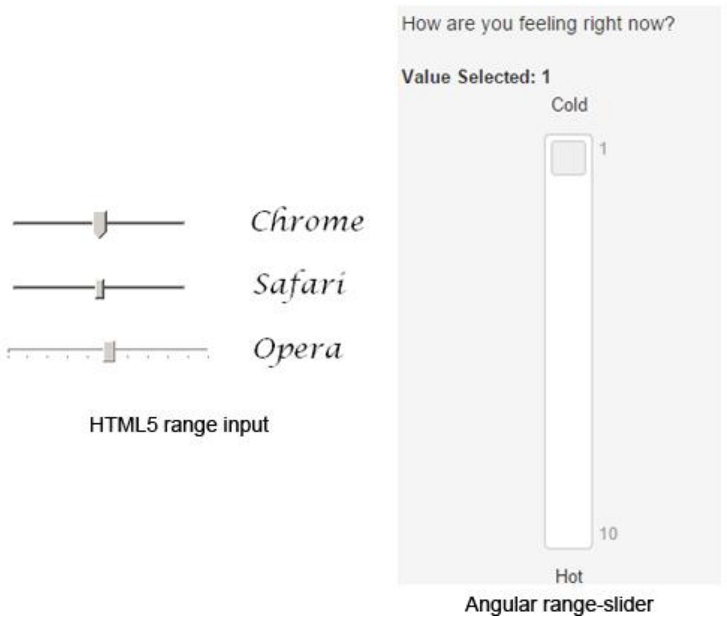


Figure 11. HTML5 range input and Angular range-slider

5.7.3 Display Layout and Swipe feature

Cascading Style Sheets from Bootstrap (http://getbootstrap.com/) is used to make the application responsive to mobile, iPad and desktops. The layout designed for large screen shows question list and answers form in the same web page giving a user an overall view of the application. While, in the small screen devices, like phone and tablet, the layout is different. The welcome page, question list and answer forms are put in three different pages so that the elements appear large enough for users to understand easily. The layout changes on the fly during the runtime when the browser is resized.

The application also supports swipe functionality in touch devices to traverse through the questions. Angular has an extension module Angular-Touch that provides touch event for touch-enabled devices. [35] In fact, the swipe feature of angular-touch module also extends to mouse drag events in non-touch devices. The feature isn't a merit for CASS-Q application as it might conflict with other mouse activity like dragging handle in a slider. jQuery *Touchwipe* plugin came very handy for this case. It is interesting to find out how a jQuery callback function could be defined inside a controller of Angular module and access other functions of the Angular Scope. It is shown in the code below.

```
$("#question-form").touchwipe({
   wipeLeft: function() {
        if (angular.isDefined(researchService.getCurrentToken()))
            $scope.showNext();
   },
   wipeRight: function() {
        if (angular.isDefined(researchService.getCurrentToken()))
            $scope.showPrevious();
   },
   preventDefaultEvents: false
});
```

Code example from CASS-Q application

6 Mobile Application

6.1 PhoneGap

PhoneGap is a framework that is used to develop mobile applications using web technologies. Mainly User Interface, application logic and exchange of data to servers are based on web technology (HTML, CSS, and JavaScript) but the component of the application that handles the device features is based on the native language for that platform. The device handling JavaScript API is provided by the PhoneGap. The native component of PhoneGap framework works behind the scene and so a developer can concentrate on business logic and User Interface worrying less about accessing device capabilities. [15,17-19] Figure 12 shows the Architecture of a PhoneGap Application.

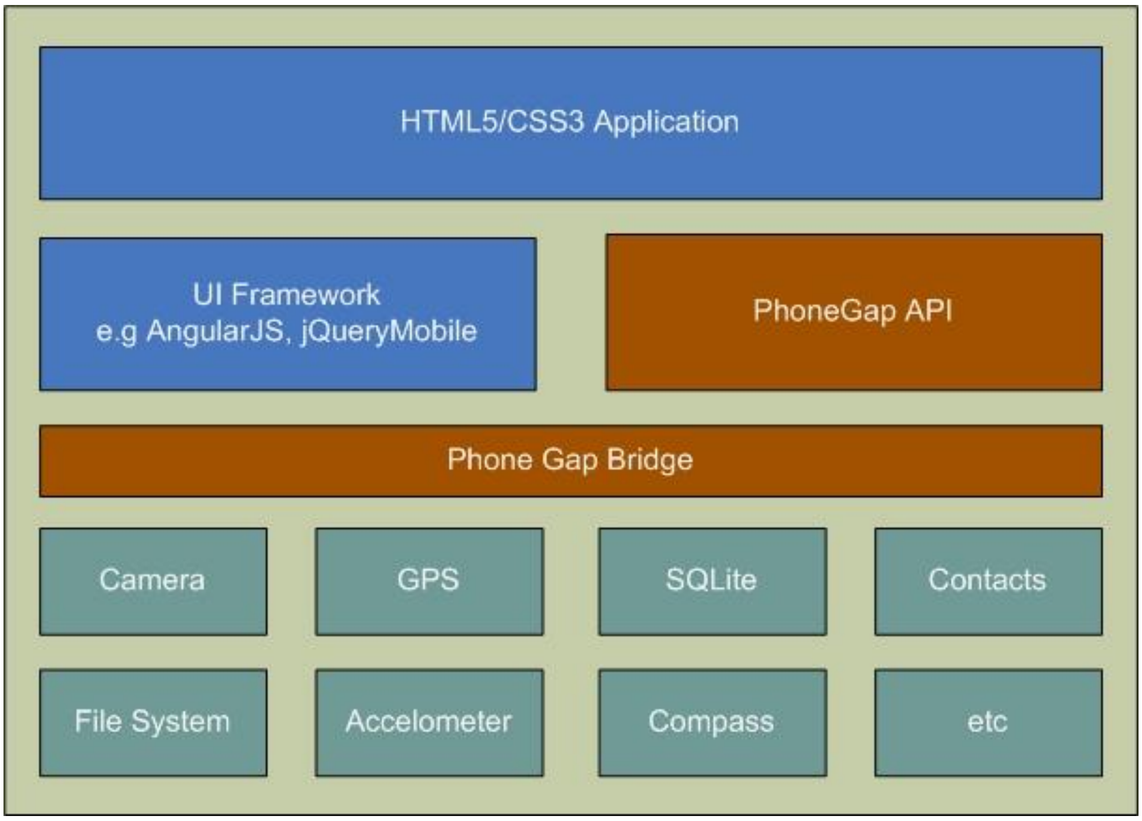


Figure 12. PhoneGap Application Architecture [15,18]

The PhoneGap Bridge is device specific meaning that there are separate PhoneGap frameworks for different mobile platforms but the same web application can be used to build mobile application for different platforms. [15,17-19]

6.2 Converting to Android application

PhoneGap is a just a framework. It doesn't provide any development environments. A developer has to setup mobile platform related development environment by himself. Xcode Integrated Development Environment is needed to develop an iPhone application whereas android application is developed in Eclipse. Eclipse requires several other plugins to be able to develop an Android application. They are listed below.

1. Java Development Kit (JDK)
2. Android Software Development Kit (SDK)
3. Android Development Tools (ADT) plugins for Eclipse
4. Android Virtual Device (AVD)
5. PhoneGap Software Development Kit (for hybrid applications only)

It is recommended to have the latest version of plugins that are needed. Fortunately, there is no need to setup above plugins separately anymore. Android Development community provides ADT Bundle which contains the following tools as a set. [36]

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- A version of the Android platform
- A version of the Android system image for the emulator

Java Development Kit [37] needs to be installed separately. Eclipse application can be run readily just after extracting the bundle and an application can be developed for the Android version that comes with the bundle without needing any other tools. To develop an application for other Android versions, ADT Plugin and AVD of the same version is needed. This can be done by running Android SDK manager to update Android SDK Tools and Android SDK Platforms-tools and then by selecting the required version of Android. Android Virtual Device is easy to create by using Android Virtual Device Manager that comes with the bundle. [15,19-27]

The PhoneGap [38] framework can be downloaded from http://phonegap.com/install/ . The package contains an example and framework for android, blackberry, iOS, and windows platform. To develop a PhoneGap application, a normal android project is created in the Eclipse and the PhoneGap is injected by copying files from PhoneGap framework for android to the project folder in Eclipse. The web application is developed in *assets/www* folder of PhoneGap framework. Then, in the main activity of the android application, the URL to the web application should be loaded when the activity is created.

```
import android.os.Bundle;
import org.apache.cordova.*;
public class MainActivity extends DroidGap {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        super.loadUrl("file:///android_asset/www/index.html");
    }
}
```

The Cordova library that comes with PhoneGap framework must be imported to the *MainActivity*. It contains the APIs which communicate with device features. *MainActivity* must extend *DroidGap* class. *DroidGap* is a class defined in Cordova package.

6.3 Accessing device feature

Cordova library of PhoneGap provides limited Plugin APIs for accessing device features. Battery Status, Camera, Console, Contacts, Device information, Accelerometer, Device Orientation, Visual Device notifications, File System, File Transfer, Geolocation, Globalization, Media recording and playback, Vibration are among them. For other functionalities, PhoneGap recommends other third party plugins. Camera and File Transfer plugin are used in the CASS-Q project and a general idea of using the plugins is discussed in this section. [39]

By default, the function *camera.getPicture* of Cordova library opens the default camera application of the device. The camera application closes when the photo is taken and the main application restores. If *Camera.sourceType* is set to *Photolibrary* or *Savedphotoalbum*, a photo chooser dialog appears instead of the camera. [40]

```
$scope.launchPictureTask = function() {
  navigator.camera.getPicture( function(imageURI) {
      document.getElementById("pic").src = imageURI;
      var pictureScope = angular.
       element(document.getElementById('pic')).scope();
      pictureScope.nameAndAddPictureFile(imageURI);
       },function(message) {
       alert('Failed because: ' + message);
       },{
       quality : 70,
       destinationType : Camera.DestinationType.FILE_URI,
       sourceType : navigator.camera.PictureSourceType.CAMERA
       });
   };
```

If camera capture is a success, the first function parameter of *naviga-tor.camera.getPicture* is called. *imageURI* is the captured image and can be assigned as source for image DOM element. To pass the image to an angular scope function, first the scope object should be identified. It can be done by using *angu-lar.element(DOM).scope()* function on the any DOM element inside targeted scope. Then, it is possible to call the function of the scope and pass values to it. The second function is called if there is an Error and the third object is for configuring camera op-tions.

File Transfer is an object in File API in PhoneGap that allows the application to upload files to a server using an HTTP multi-part POST request. In iOS and Android, it also supports downloading a file from a server and saving it. Function *upload()* of File Transfer object is used to upload files. It takes five parameters- filePath, server, *suc-cessCallback* function, *errorCallback* functions and *options*. *options* is an object which contains parameters such as *fileKey, filename, mimeType.* [41]

7 Conclusion

The prime objective of this project was to explore the usage of a Javascript framework to build client side application. The known available JavaScript frameworks were compared and AngularJS was chosen for the project. As it was expected, AngularJS was found to be learnt easily. AngularJS is not just suitable for developing Single Page Application (SPA) for which it is praised, but it was found during the project that the framework can perfectly utilize back buttons in web-browser to go to previous pages giving a feel of familiar handling in the smart phones with back button. The two-way data binding capability of Angular helps to reduce the amount of codes and keep the building procedure focused towards more serious logic. By presenting the changes on the fly, the data binding ability makes the application's performance appear swifter.

In my personal opinion, the Angular framework is very light and fast executing. The MVC structure and the modular structure is easy to understand. The Developer Guide and API references provided by Angular Team in www.angularjs.org are very helpful to understand the concept. However, I sensed a need for other learning resources, like a book to teach a new developer how to structure functions in an angular way, especially when creating custom dependency services. Unfortunately, there were not many books written on AngularJS [16] when the project was taking place and those too, were not very easy to find in market. Discussion rooms in the internet broke the deadlocks sometimes but it could always be felt that better or more solutions to the problem could be lying somewhere else. However, the lack of materials on AngularJS did not cloud the advantages of using a framework to build a client side application. The use of framework reduced a significant amount of thinking and code to write for me during this project which resulted in quick development of the application.

The open source framework, AngularJS has a big community and it offers many extensions. Some of the external modules, for instance, angular-translate, is recognized by the Angular Team itself [16]. The offerings from the community certainly made my work a lot easier when it came to implementing localization and substituting incompatible HTML5 feature. I had not realized the benefit of a big community before running into trouble myself. The ability of AngularJS to work along well with non-AngularJS libraries also helped this project to include extra features. Swipe feature is one of them.

The client side application, CASS-Q developed during the project is hosted at http://54.247.115.29/cassQ. The application is being tested by the Doctoral Students of Department of Psychology in Helsinki University. Thanks to HTML technology, CASS-Q cross platform web application provided them with wide range of client devices, almost all the devices which has a web browser installed in it. With this development at the client side, they can now invite more test subjects to their research and carry out more surveys. The frequency of tests is crucial for a developing system like Contextual Activity Sampling System because more tests help realize the needed features at the Admin Console quicker.

The integration of the CASS-Q application with PhoneGap framework to produce a hybrid application for mobile devices has also been studied. Angular structure works well in the Cordova library of PhoneGap in Android app and the CASS-Q application is already successful to communicate with camera and upload picture to the CASS Backend. However, in the short time of three and half months, the audio-video recording features required for CASS were not looked into. The remaining features are aimed to be dealt with during future development. The granular structure of the framework is easier to understand. Also, as modular nature of framework does not let extended features mess with the business logic of the main application, I believe, the succeeding developers will grab the application concept effortlessly in short time.

References

1 Muukkonen H, Inkinen M, Kosonen K, Hakkarainen K, Vesikivi P, Lachmann H, Karlgren K. Research on knowledge practices with the Contextual Activity Sampling System. In: O'Malley C, Suthers D, Reimann P, Dimitracopoulou A, editors. Proceedings of the 9th international conference on Computer supported collaborative learning - Volume 1. International Society of the Learning Sciences; 2009. p. 385-394.

2 Salo K, Shakya U, Damena M. Device Agnostic CASS Client. In: Marcus A, editor. Design, User Experience, and Usability: User Experience Design for Diverse Interaction Platforms and Enviroments. Switzerland: Springer International Publishing; 2014. p. 334-344.

3 Zakas NC. Professional JavaScipt for Web Developers. Indianapolis, Indiana: Wiley Publishing Inc; 2005.

4 Muhaddisoglu U. 40 Useful JavaScript Libraries. [online]. Smashing Magazine. 2 March 2009.
URL: http://www.smashingmagazine.com/2009/03/02/40-stand-alone-javascript-libraries-for-specific-purposes/
Accessed 9 October 2014.

5 Tutorials Point. HTML5 - Quick Guide. [online]. Website.
URL: http://www.tutorialspoint.com/html5/html5_quick_guide.htm
Accessed 9 October 2014.

6 Lawson B, Sharp R. Introducing HTML5. USA: Peachpit; 2012.

7 Tutorials Point. HTML5 Attributes. [online]. Website.
URL: http://www.tutorialspoint.com/html5/html5_attributes.htm
Accessed 9 October 2014.

8 Karlins D. HTML5 and CSS3 For Dummies. Somerset, NJ, USA: John Wiley & Sons, 2013.

9 Tutorials Point. HTML5 - Web Forms 2.0. [online]. Website.
URL: http://www.tutorialspoint.com/html5/html5_web_forms2.htm
Accessed 9 October 2014.

10 Tutorials Point. HTML5 - Audio & Video. [online]. Website.
URL: http://www.tutorialspoint.com/html5/html5_audio_video.htm
Accessed 9 October 2014.

11 Tutorials Point. HTML5 - Web Storage. [online]. Website.
URL: http://www.tutorialspoint.com/html5/html5_web_storage.htm
Accessed 9 October 2014.

12 Hickson I. Web Storage. W3C Recommendation.[online]. www.w3.org; 2013.
URL: http://www.w3.org/TR/webstorage/#disk-space
Accessed 9 October 2014.

13 Lowery JW, Fletcher M. HTML5 24-Hour Trainer. Hoboken, NJ, USA: John Wiley & Sons, 2011.

14 AngularJS.org . Introduction. [online]. Website.
URL: https://docs.angularjs.org/guide/introduction
Accessed 9 October 2014.

15 Ghatol R, Patel Y. Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5. Apress; 2012.

16 AngularJS.org. Developer Guide. [online]. Website.
URL: https://docs.angularjs.org/guide/
Accessed 9 October 2014.

17 Lerner A. ng-book: The Complete Book on AngularJS. USA: Fullstack.io; 2013.

18 AngularJS.org . Modules. [online]. Website.
URL: https://docs.angularjs.org/guide/module
Accessed 9 October 2014.

19 Wahlin D. AngularJS in 60 Minutes. [online]. 2013.
URL: http://weblogs.asp.net/dwahlin/angularjs-in-60-ish-minutes-the-ebook
Accessed 9 October 2014.

20 AngularJS.org . Directives. [online]. Website.
URL: https://docs.angularjs.org/guide/directive
Accessed 9 October 2014.

21 AngularJS.org . Bootstrapping. [online]. Website.
URL: https://docs.angularjs.org/tutorial/step_00
Accessed 9 October 2014.

22 AngularJS.org . Angular Templates. [online]. Website.
URL: https://docs.angularjs.org/tutorial/step_02
Accessed 9 October 2014.

23 AngularJS.org . $rootScope. [online]. Website.
URL: https://docs.angularjs.org/api/ng/service/$rootScope
Accessed 9 October 2014.

24 AngularJS.org . $rootScope.Scope. [online]. Website.
URL: https://docs.angularjs.org/api/ng/type/$rootScope.Scope
Accessed 9 October 2014.

25 AngularJS.org . Providers. [online]. Website.
URL: https://docs.angularjs.org/guide/providers
Accessed 9 October 2014.

26 AngularJS.org . $http. [online]. Website.
URL: https://docs.angularjs.org/api/ng/service/$http
Accessed 9 October 2014.

27 x2js. [online].
URL: https://code.google.com/p/x2js/
Accessed 9 October 2014.

28 Angular Translate. [online].
URL: http://angular-translate.github.io/
Accessed 9 October 2014.

29 AngularJS.org . ngCloak. [online]. Website.
URL: https://docs.angularjs.org/api/ng/directive/ngCloak
Accessed 9 October 2014.

30 Sharp R. What is a Polyfill? [online].
URL: https://remysharp.com/2010/10/08/what-is-a-polyfill
Accessed 9 October 2014.

31 Frain B. Responsive Web Design with HTML5 and CSS3. Olton, Birmingham, GBR: Packt Publishing, 2012.

32 Isrih P. HTML5 Cross Browser Polyfills. [online].
URL: https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills
Accessed 9 October 2014.

33 Stipe J. Number polyfill. [online].
URL: https://github.com/jonstipe/number-polyfill
Accessed 9 October 2014.

34 Crisp D. AngularJS RangeSlider. [online].
URL: http://danielcrisp.github.io/angular-rangeslider/
Accessed 9 October 2014.

35 AngularJS.org . ngTouch. [online]. Website.
URL: https://docs.angularjs.org/api/ngTouch
Accessed 9 October 2014.

36 Android Developers Guide. Get the Android SDK. [online].
URL: https://developer.android.com/sdk/index.html
Accessed 9 October 2014.

37 Oracle. Java SE Development Kit 7 Downloads. [online].
URL: http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
Accessed 9 October 2014.

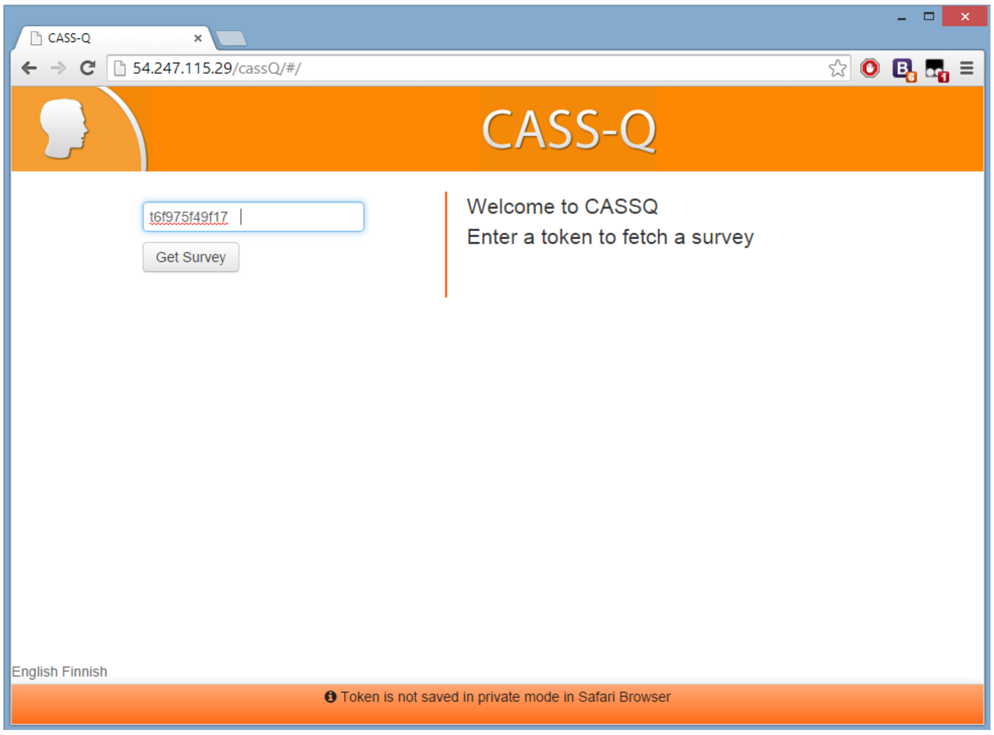
38 PhoneGap. [online].
URL: http://phonegap.com/
Accessed 9 October 2014.

39 Plugin APIs. PhoneGap Documentaion. [online].
URL:http://docs.phonegap.com/en/edge/cordova_plugins_pluginapis.md.html#Plugin%20APIs
Accessed 9 October 2014.

40 Camera. PhoneGap Documentaion. [online].
URL:http://docs.phonegap.com/en/2.0.0/cordova_camera_camera.md.html#camera.getPicture
Accessed 9 October 2014.

41 File. PhoneGap Documentaion. [online].
URL: http://docs.phonegap.com/en/2.0.0/cordova_file_file.md.html#FileTransfer
Accessed 9 October 2014.

Screen shots of main views

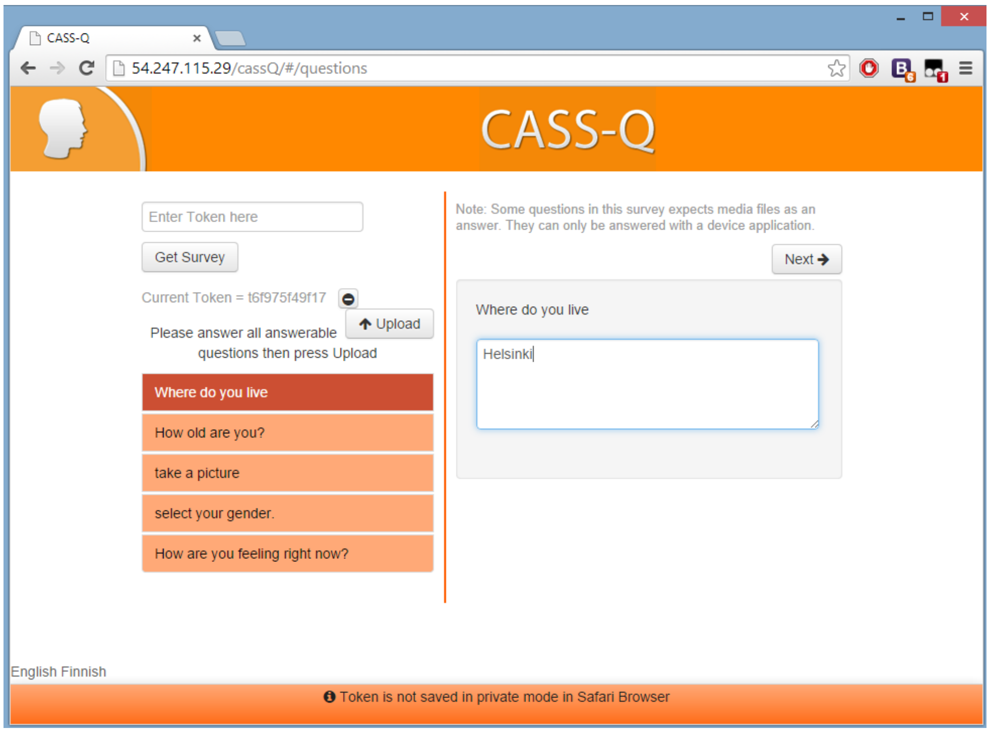


Welcome to CASSQ
Enter a token to fetch a survey

English Finnish

Token is not saved in private mode in Safari Browser

Figure 13. Home Page Implementation of CASS-Q web application in a desktop



Current Token = t6f975f49f17

Please answer all answerable questions then press Upload

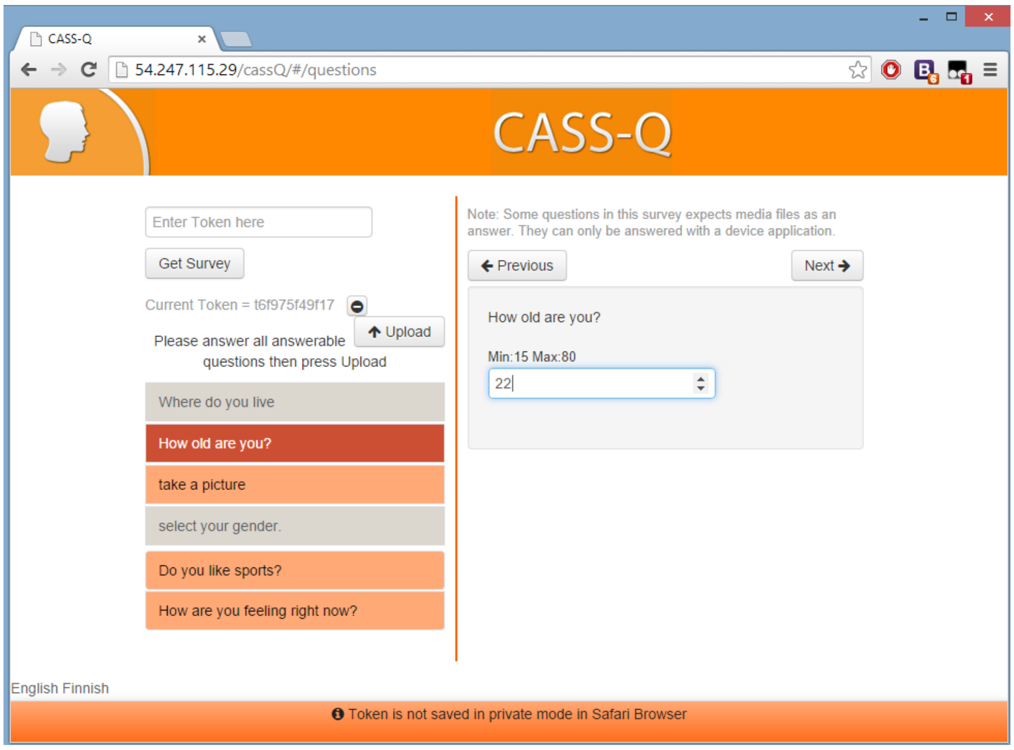
Note: Some questions in this survey expects media files as an answer. They can only be answered with a device application.

Where do you live

English Finnish

Token is not saved in private mode in Safari Browser

Figure 14. Questions downloaded from CASS backend.



CASS-Q

54.247.115.29/cassQ/#/questions

Enter Token here

Get Survey

Current Token = t6f975f49f17

Upload

Please answer all answerable questions then press Upload

- Where do you live
- How old are you?
- take a picture
- select your gender.
- Do you like sports?
- How are you feeling right now?

Note: Some questions in this survey expects media files as an answer. They can only be answered with a device application.

Previous | Next

How old are you?

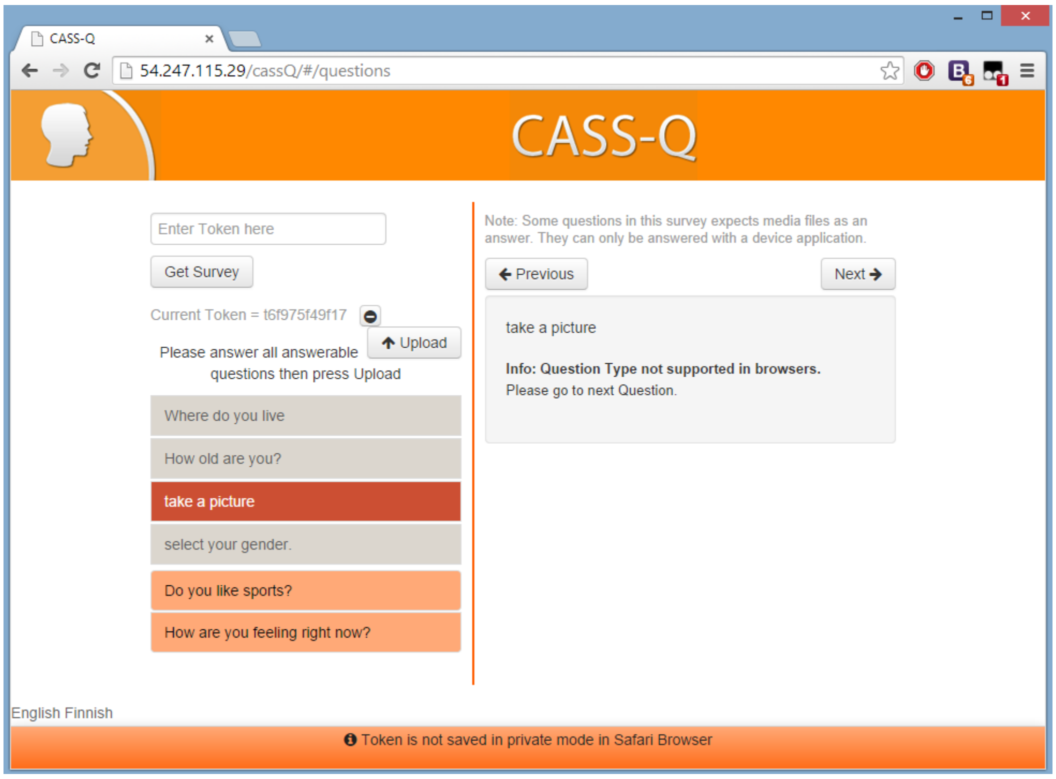
Min:15 Max:80

22

English Finnish

Token is not saved in private mode in Safari Browser

Figure 15. HTML 5 input number type used for open number answer.



CASS-Q

54.247.115.29/cassQ/#/questions

Enter Token here

Get Survey

Current Token = t6f975f49f17

Upload

Please answer all answerable questions then press Upload

- Where do you live
- How old are you?
- take a picture
- select your gender.
- Do you like sports?
- How are you feeling right now?

Note: Some questions in this survey expects media files as an answer. They can only be answered with a device application.

Previous | Next

take a picture

Info: Question Type not supported in browsers.
Please go to next Question.

English Finnish

Token is not saved in private mode in Safari Browser

Figure 16. Media capturing not implemented in browsers.

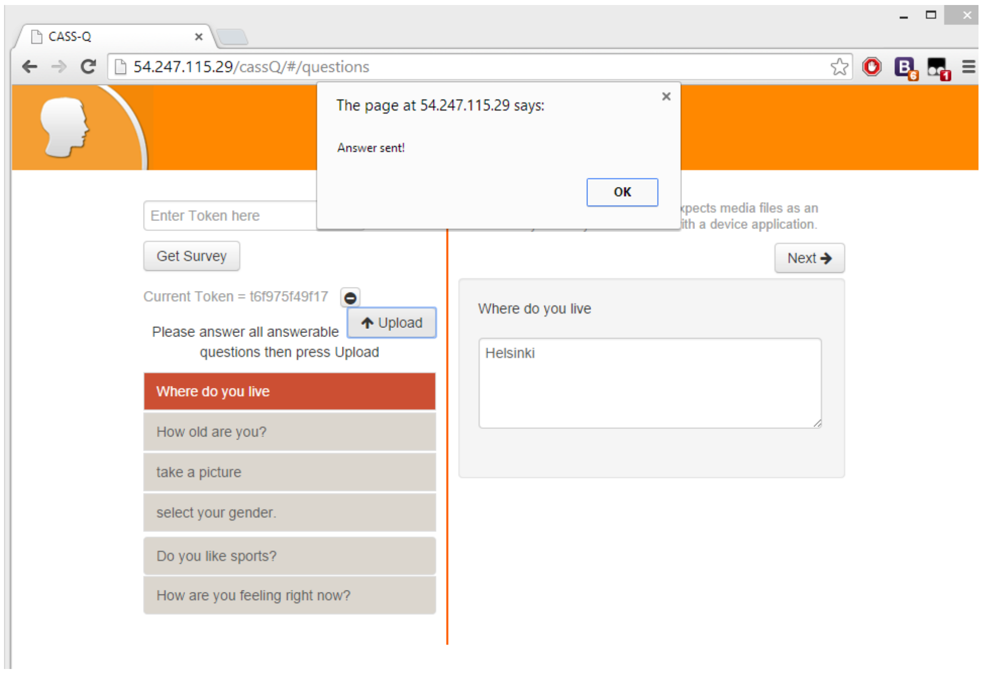

CASS-Q
54.247.115.29/cassQ/#/questions
The page at 54.247.115.29 says:
Answer sent!
OK
Enter Token here
Get Survey
Current Token = t6f975f49f17
Upload
Please answer all answerable questions then press Upload
Where do you live
How old are you?
take a picture
select your gender.
Do you like sports?
How are you feeling right now?
Next
Where do you live
Helsinki

Figure 17. $http service of AngularJS uses POST method to send answers to backend.

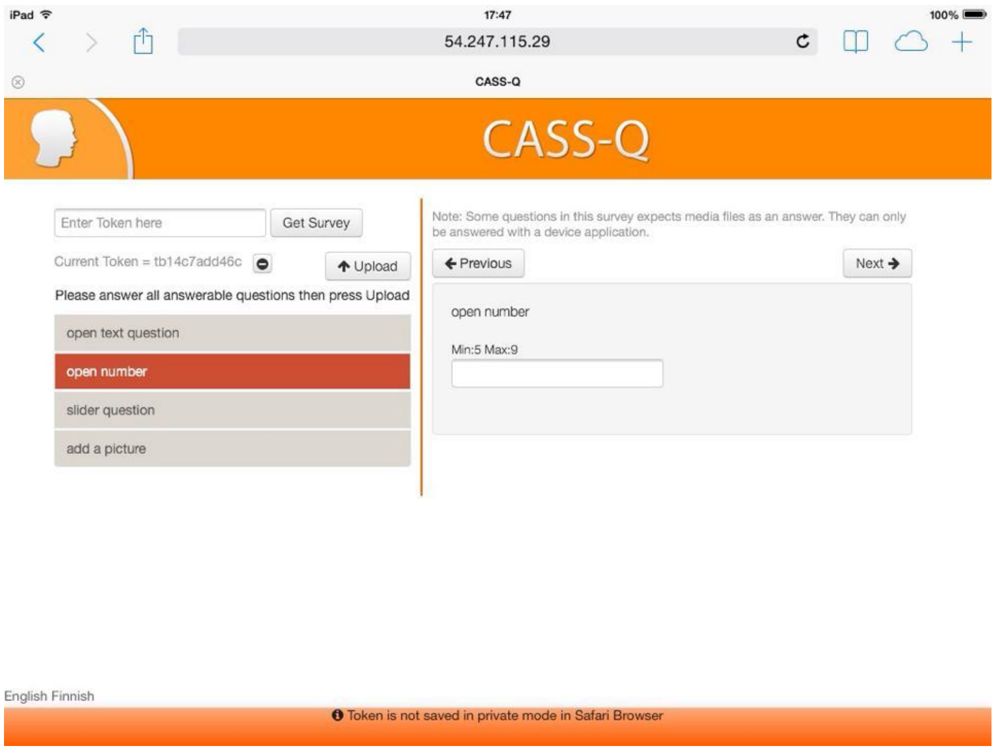
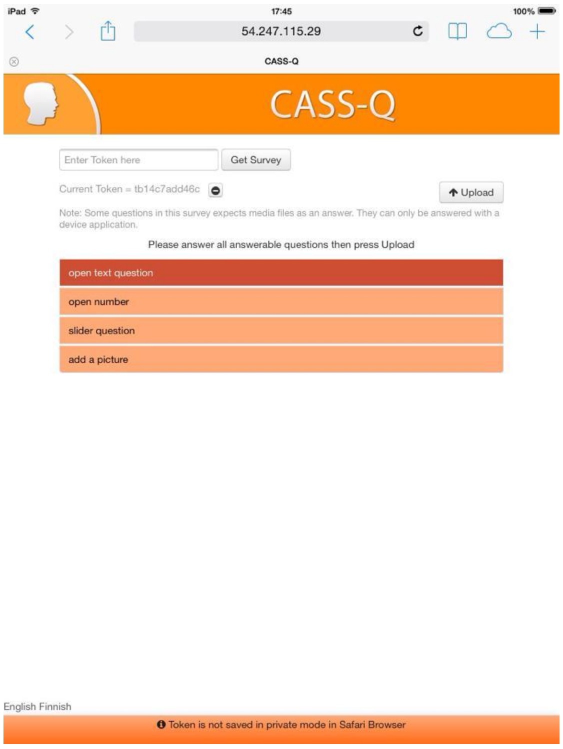

iPad
17:47
100%
54.247.115.29
CASS-Q
CASS-Q
Enter Token here
Get Survey
Current Token = tb14c7add46c
Upload
Please answer all answerable questions then press Upload
open text question
open number
slider question
add a picture
Note: Some questions in this survey expects media files as an answer. They can only be answered with a device application.
Previous
Next
open number
Min:5 Max:9
English Finnish
Token is not saved in private mode in Safari Browser

Figure 18. Application layout in iPad (horizontal mode).


iPad 17:45 100%

54.247.115.29

CASS-Q

CASS-Q

Enter Token here | Get Survey

Current Token = tb14c7add46c

Upload

Note: Some questions in this survey expects media files as an answer. They can only be answered with a device application.

Please answer all answerable questions then press Upload

open text question

open number

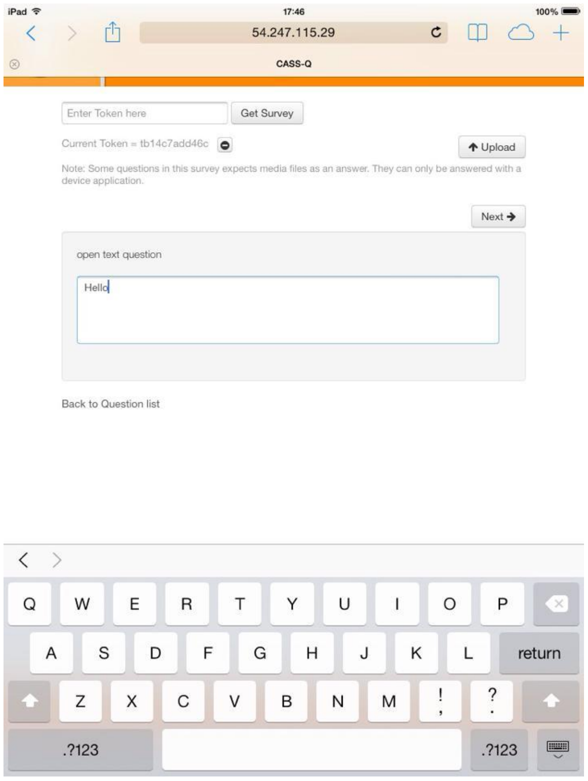
slider question

add a picture

English Finnish

Token is not saved in private mode in Safari Browser

Figure 19. Application layout in iPad (vertical mode).


iPad 17:46 100%

54.247.115.29

CASS-Q

Enter Token here | Get Survey

Current Token = tb14c7add46c

Upload

Note: Some questions in this survey expects media files as an answer. They can only be answered with a device application.

Next

open text question

Hello

Back to Question list

Figure 20. Separate answer page in vertical mode

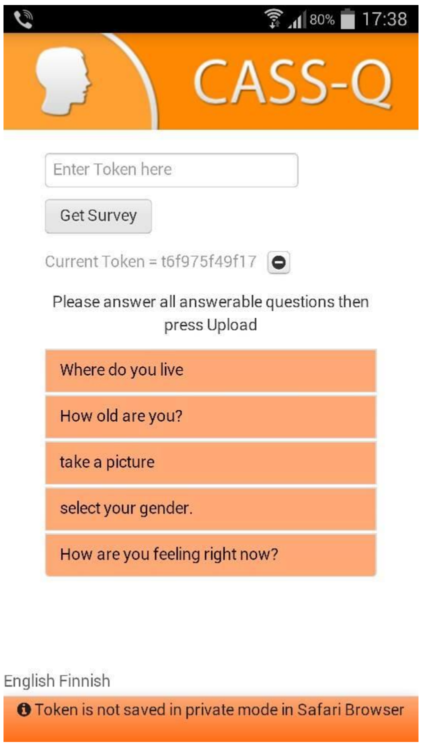

Enter Token here
Get Survey
Current Token = t6f975f49f17
Please answer all answerable questions then press Upload
Where do you live
How old are you?
take a picture
select your gender.
How are you feeling right now?
English Finnish
Token is not saved in private mode in Safari Browser

Figure 21. Layout in Hybrid Application in Android

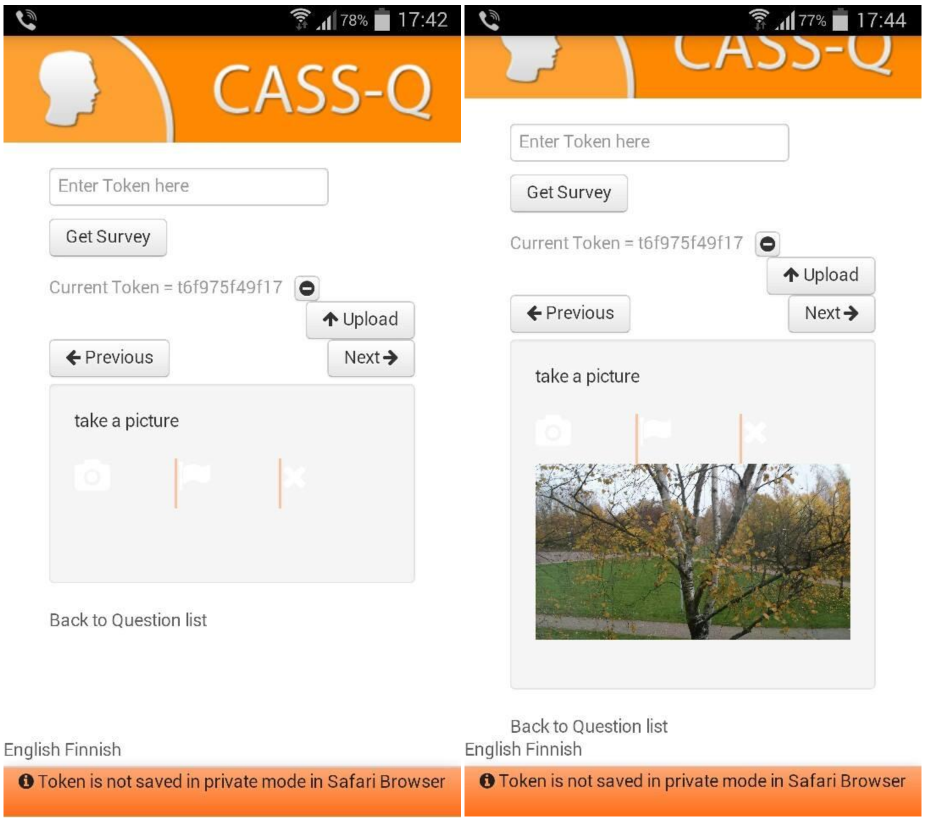

Enter Token here
Get Survey
Current Token = t6f975f49f17
Upload
Previous
Next
take a picture
Back to Question list
English Finnish
Token is not saved in private mode in Safari Browser

Figure 22. Taking a picture with the hybrid application in Android.