Abstract

Task 6.5 aims at laying the foundations of a coherent and intuitive user experience. There are two main objectives of this task. The first is to design guidelines for use by project internal and third party developers based on existing best practice design standards and technology specific considerations to guide the development of the platform UI and 3rd party service UIs for the target UI technologies and end user devices.

The second one is to produce a prototype implementation of a unified UI to enable users to interact with, configure and experience the features of the SOCIETIES platform.
Disclaimer

This document contains material, which is the copyright of certain SOCIETIES consortium parties, and may not be reproduced or copied without permission.

**In case of Public (PU):**
All SOCIETIES consortium parties have agreed to full publication of this document.

**In case of Restricted to Programme (PP):**
All SOCIETIES consortium parties have agreed to make this document available on request to other framework programme participants.

**In case of Restricted to Group (RE):**
All SOCIETIES consortium parties have agreed to full publication of this document. However this document is written for being used by <organisation / other project / company etc.> as <a contribution to standardisation / material for consideration in product development etc.>.

**In case of Consortium confidential (CO):**
The information contained in this document is the proprietary confidential information of the SOCIETIES consortium and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the SOCIETIES consortium as a whole, nor a certain party of the SOCIETIES consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

---

**Impressum**

[Full project title] Self Orchestrating Community Ambient Intelligence Spaces
[Short project title] SOCIETIES
[Number and title of work-package] WP6, Third party services
[Document title] D6.6: Platform UUI Prototype for 2nd user trials
[Editor: Name, company] Mei Yii Lim HWU
[Work-package leader: Name, company] Yiorgos Bouloudis, AMITEC
[Estimation of PM spent on the Deliverable] 6PMs

**Copyright notice**

© 2013 Participants in project SOCIETIES

*Optionally list of organisations jointly holding the Copyright on this document*
Executive summary

The goal of T6.5 is to provide a unified and consistent look and feel of the user interface for the trial users based on best practice design standards. This interface will enable users to interact with, configure and experience the features of the SOCIETIES platform. The technology specific details will produce guidelines for both the internal as well as the 3rd party services developers, for implementing the necessary user interfaces. This deliverable extends the first UUI prototype. It contains descriptions of further user interface elements beyond those delivered in the last release. It focuses mostly on the SOCIETIES Web Application with a brief explanation of the feedback plugin on the SOCIETIES Android Application.

Note that this document was originally scheduled as D6.8 but the content of deliverable D6.7 was subsumed within the earlier D6.6 deliverable so this document has been renumbered as D6.7.
## List of authors

<table>
<thead>
<tr>
<th>Company</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWU</td>
<td>Mei Yi Lim</td>
</tr>
<tr>
<td></td>
<td>Patrick Skillen</td>
</tr>
</tbody>
</table>
A slider which is a jQuery plugin has also been added. Its styling is specified in the slider.css file. This plugin depends on widget.css, jquery.ui.core.js, jquery.ui.widget.js, jquery.ui.mouse.js and jquery.ui.slider.js.

The default range is 0-100. A separate document included in the DOCUMENTS/Workpackages/WP6/T6.5/Webapp Additional Features Styleguide folder SVN called “slider.html” provides the necessary information on how to use it and the different functions available.

Another new element is the notification component. Its interface styling and functions are specified in the notification.css and notification.js respectively. This component consists of a notification bubble which appears on the top menu as shown in the diagram below and a notification box which contains all the different notifications messages.

A loading icon “loading_solid_48x48.png” is now available in the images folder.

A logout button consistent with the login button style has been added at the top of all pages.

4 JSF and PrimeFaces Components
4.1 Introduction
4.1.1 PrimeFaces Components
4.1.2 Close Up View of the UUI Features
4.1.3 Adding Pages to the WebApp
4.1.4 Creating a Controller for your Page

5 Web User Interface Menu

6 Vibration and Sound Feedback on Android

7 Conclusion
1 Introduction

This document provides descriptions of additional UI features and pages for the Societies Web User Interface. It also touches on the feedback plugin on the Societies Android App which includes vibration and beeping functionalities. It is divided into seven sections, with the first one being the current introduction. In Section 2, the requirements gathering process and implementation code location is provided. This is followed by the styleguide details of the additional WebApp features in Section 3. Section 4 provides instruction to the developers on how to integrate the relevant PrimeFaces components in their service UIs. Section 5 presents the current Web User Interface navigation menu while Section 6 details the feedback plugin on the Societies Android App. Finally, Section 7 provides a conclusion.
2  Gathering requirements

2.1  Discussion with Developers

The process of gathering requirements for the additional features was less demanding than the initial requirements gathering phase. It mainly involved discussion with developers about which elements are still missing from the existing User Interface which are required by their services.

2.2  Implementation Code


Corresponding PrimeFaces components following the proposed styleguide have been created and integrated into the WebApp package at [https://github.com/societies/SOCIETIES-Platform/tree/development/platform-infrastructure/web-app](https://github.com/societies/SOCIETIES-Platform/tree/development/platform-infrastructure/web-app).

3 Styleguide of Additional Features

3.1 Introduction

All new features conform to the existing style of the Societies User Interface defined in D6.6. Please note that this section provides only the styleguide mentioning the relevant CSS, jQuery, JavaScript and HTML files. These features cannot be used directly in the new WebApp which is utilising JSF and PrimeFaces components. The instructions on how to integrate them with the new WebApp are provided in Section 4.

3.1.1 A Star Rating Element

Figure 1: Star Rating Element

The star rating component is made up of a list of 5 stars. The style is defined in `star-rating.css`. In order to use this element, include the `star-rating.css` in the `<head>` section of the specific html page where it will appear. Then, simply add it to the html page body as below:

```
<ul class="star-rating">
    <li><a href="#" class="one-star"></a></li>
    <li><a href="#" class="two-stars"></a></li>
    <li><a href="#" class="three-stars"></a></li>
    <li><a href="#" class="four-stars"></a></li>
    <li><a href="#" class="five-stars"></a></li>
</ul>
```

A current rating can be added with the line below:

```
<li class="current-rating" style="width:60%;">Currently 3/5 Stars.</li>
```

After the list of stars has been added, each star can be linked to the appropriate page or JavaScript functions can be written for it to perform specific actions.
3.1.2 A slider

A slider which is a jQuery plugin has also been added. Its styling is specified in the slider.css file. This plugin depends on widget.css,jquery.ui.core.js, jquery.ui.widget.js, jquery.ui.mouse.js and jquery.ui.slider.js.

The default range is 0-100. A separate document included in the DOCUMENTS/Workpackages/WP6/T6.5/Webapp Additional Features Styleguide folder SVN called “slider.html” provides the necessary information on how to use it and the different functions available.

![Obfuscation level](image)

Figure 2: A slider

To insert it to your HTML page after all the necessary CSS and JavaScript files are included, add a <div> tag with the id “slider” as below:

```
<div id="slider"></div>
```

3.1.3 Notification Components

Another new element is the notification component. Its interface styling and functions are specified in the notification.css and notification.js respectively. This component consists of a notification bubble which appears on the top menu as shown in the diagram below and a notification box which contains all the different notifications messages.

![Notification bubble](image)

Figure 3: Notification Bubble

The notification box uses the same interface design as the login pop-up. You can add and delete notification from this box. When a notification is added, the notification bubble on the top menu will be updated automatically to show the number of new notifications available. When the user clicks on the notification icon, this number will be reset automatically to zero.

Each notification should be independent from another notification and is contained in a separate panel with a specific ID (given by you). Hence, to add a notification, always remember to include your element inside a <div> tag of the class “notification-panel” and name the panel.

```
<div class="notification-panel" id="panelName">
  …
</div>
```

For example, if you would like to add a panel with some text and buttons, you have to call the addPanel function in notifications.js with your panel attributes:

```
<div class="notification-panel" id="locationPanel">
  <span>Do you want to share your location with John?</span>
</div>
```
Figure 4 shows the result of adding the above panel.

![Notification Box]

Figure 4: Notification Box

The code to add radio buttons will look like this:

```html
<div class=notification-panel id=radioPanel>
  <div class=notification-option-group radio>
    <input type=radio id=radio1 /><label for=radio1>Radio1</label>
    <input type=radio id=radio2 /><label for=radio2>Radio2</label>
  </div>
</div>
```

To add a slider, call the addSlider function in notifications.js.

```html
<div class=notification-panel id=sliderPanel>
  <label id=Obsfuscation level>
    <span>Obsfuscation level</span>
    <div id=slider></div>
    <input class=slider-text type=text id=value />
  </label>
</div>
```

For more elements, please refer to the notification.css file.
As more and more notifications are added, a scroll bar will appear to the right of the notification box as presented in Figure 5.

![Figure 5: Scrollable Notification Box](image)

In order to delete a notification panel, call the `removePanel(panelID)` function in `notification.js` by passing in the panel ID you would like to remove.

### 3.1.4 Loading Icon

A loading icon “loading_solid_48x48.png” is now available in the images folder.

![Figure 6: Loading Icon](image)

### 3.1.5 Logout Button

A logout button consistent with the login button style has been added at the top of all pages.

![Figure 7: Logout Button](image)
4 JSF and PrimeFaces Components

4.1 Introduction

The SOCIETIES Web Application is a bundle running in the container that will allow the CSS user access to their core services. Developers can extend this bundle to provide a web user interface to their services, displaying data and receiving input. The new WebApp has been modified to serve JSF (Java Server Faces) pages instead of JSP. JSF provides a higher level of abstraction than JSP, and takes care of handling requests, AJAX, etc, without the developer having to implement these each time. Additionally, the new WebApp uses pre-built PrimeFaces Components. As a result, the additional features as described in Section 3 cannot be used directly due to compatibility issues between PrimeFaces and jQuery.

To allow developer to take advantage of the UUI features, a corresponding PrimeFaces theme has been generated based on the existing styleguide. This theme called theme.css can be found under the web-app/scr/main/webapp/resources/primefaces-societies-theme folder. Applying this theme, PrimeFaces components with similar look and feel to the features already mentioned in D6.6 as well as those described in Section 3 of this document are created and integrated with the new WebApp. Please note that only the user interface is integrated, developers will be required to develop the underlying features.

4.1.1 PrimeFaces Components

The styleguide.xhtml page which can be found in the WebApp package contains all the different SOCIETIES UUI features as presented in the diagram below. For additional PrimeFaces components that are not included, please refer to http://www.primefaces.org/showcase/ui/home.jsf

Figure 8 : Styleguide.xhtml
4.1.2 Close Up View of the UUI Features

Figure 9: Buttons

Figure 10: Text Boxes

Figure 11: Check Boxes

Figure 12: Radio Buttons
Figure 13: List

Figure 14: Hyperlink

Figure 15: Star Rating Component

Figure 16: Slider

Figure 17: Spinner

Figure 18: Tab Menu

Figure 19: Accordion Menu
In the new WebApp, notifications are automatically added/removed by the UserFeedback component in response to UF events. When the user has responded to these notifications, the number in the bubble is automatically decremented and the bubble removed when the count reaches zero.
4.1.3 Adding Pages to the WebApp

JSF allows the definition of a template file with "regions" which can be filled by the implementing pages. The template file includes all links to header, menu and footer ensuring that they are consistent across all pages. The SOCIETIES WebApp main template is called `main_template.xhtml` which can be found in `web-app/scr/main/webapp/templates` folder of the package. A snapshot of the code is presented below. More information can be found at [https://redmine.ict-societies.eu/projects/sp/wiki/SocietiesWebapp](https://redmine.ict-societies.eu/projects/sp/wiki/SocietiesWebapp).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<head>
  <style>/* Template: HTML HEAD CONTENT GOES HERE IN THE IMPLEMENTING PAGE */</style>
</head>

<body>
  <!-- Template: BODY CONTENT GOES HERE IN THE IMPLEMENTING PAGE -->
</body>
</html>
```

© SOCIETIES consortium 2013
The template contains `ui:insert` tags which have been highlighted in red. In order to add new web page to the WebApp the developer should create a file under `src/main/webapp`, called `[page_name].xhtml`, and create two `ui:define` tags to correspond with the template’s `ui:insert` tags, as shown below.

To view the webpages, browse to `http://localhost:8080/societies/page_name.xhtml`

For example, the `index.xhtml` of the SOCIETIES WebApp after the replacements will look like this.

```html
<!-- disable redirect to "index.xhtml" if we’re not logged in -->
<ui:param name="noRedirect" value="true"/>

<!-- DEVELOPER's HEAD CONTENT GOES HERE -->
<ui:define name="html_head">
  <title>SOCIETIES</title>
</ui:define>

<!-- DEVELOPER's BODY CONTENT GOES HERE -->
<ui:define name="body">
  <!--Main Content START -->
  <section id="featured" class="clearfix grid_12">
    <div>
      <article><img src="images/societies-v1.png" alt="Where Pervasive Meets Social" width="960" height="400" class="landing_img"/></article>
    </div>
    <div class="hr grid_12 clearfix boxhr"></div>
    <section class="homebox_entries grid_12">
      <!-- FIRST ROW -->
      <header>
        <h4 class="box_title">DISCOVER, CONNECT &amp; ORGANISE WITH SOCIETIES</h4>
      </header>
      <div class="hr grid_12 clearfix boxhr"></div>
      <section class="homeboxgrid box" href="your_societies_friends_list.xhtml" title="Friends">
        <a class="homeboxgrid box " href="your_societies_friends_list.xhtml" title="Friends">
          <img src="images/home_friends.png" width="270" height="150" alt="Friends"/>
        </a>
      </section>
      <section class="homeboxgrid box" href="your_installed_apps.xhtml" title="Apps">
        <a class="homeboxgrid box " href="your_installed_apps.xhtml" title="Apps">
          <img src="images/home_apps.png" width="270" height="150" alt="Apps"/>
        </a>
      </section>
      <section class="homeboxgrid box" href="your_communities_list.xhtml" title="Communities">
        <a class="homeboxgrid box " href="your_communities_list.xhtml" title="Communities">
          <img src="images/home_communities.png" width="270" height="150" alt="Communities"/>
        </a>
      </section>
    </section>
  </section>
</ui:define>
</ui:composition>
</html>
```
4.1.4 Creating a Controller for your Page

The WebApp follows the classic Model-View-Controller design paradigm. Controllers are Java files stored under `web-app/src/main/java/org/societies/webapp/controller`. The controller receives the requests and processes the responses, the model contains data submitted from web forms and the view contains the formatted HTML within JSF pages. Each controller class should look like this.

```java
@ManagedBean(name = "exampleController")
@RequestScoped
public class ExampleController extends BasePageController {

    @ManagedProperty(value = "#userService")
    private UserService userService; // NB: MUST include public getter/setter

    public ExampleController () {
        // controller constructor - called every time this page is requested!
    }

    public void showMeAMessage() {
        addGlobalMessage("You asked for it, \"Now here's your message!\", FacesMessage.SEVERITY_INFO);
    }
}
```

Page controllers should extend the BasePageController class so that they can inherit some global behaviour (e.g. a logger, the ability to pass JSF Faces Messages). Controllers are generally request scoped, as defined by the `@RequestScoped` annotation (also available are `@SessionScoped` and `@ApplicationScoped`, but their use is discouraged). The `@ManagedBean(...)` annotation indicates that this class is a JSF bean which JSF views can access.

After controllers are implemented, PrimeFaces components will be able to call the respective methods to perform the necessary actions. For example, when a button is pressed, it will call whichever method is pointed to by its `actionListener` property. The button in the code below will call the `showMeAMessage` method in the exampleController above.

```html
<html xmlns="http://www.w3.org/1999/xhtml">
  <ui:composition xmlns="http://www.w3.org/1999/xhtml"
                  xmlns:ui="http://java.sun.com/jsf/facelets"
                  xmlns:p="http://primefaces.org/ui"
                  template="templates/main_template.xhtml">
    <ui:define name="html_head">
      <title>SOCIEITES</title>
    </ui:define>

    <ui:define name="body">
      <p:commandButton value="Message!"
                        actionListener="#\{exampleController.showMeAMessage\}"/>
    </ui:define>
  </ui:composition>
</html>
```
This is just a small example of component functionality implementation. For more information about writing controllers, please refer to https://redmine.ict-societies.eu/projects/sp/wiki/PlatformWebappMVC. Example controllers can also be found in the controller folder of the WebApp package.
5  Web User Interface Menu

Some restructuring has been done to the existing main menu navigation. The tree below presents the new navigation flow diagram. The pages with * are new.

![Diagram of Web User Interface Menu]

Figure 24: Template for the new Social Network page
6 Vibration and Sound Feedback on Android

In order to improve the responsiveness of the SOCIETIES Android App, a feedback plugin has been implemented. This plugin includes beeping and vibration features. The vibration feature will provide haptic feedback to the user, for example, during login and button press to reflect the fact that an input has been received or saved successfully. The beeping feature on the other hand, can act as an alert to user when a new message or notification is received when the user is not looking at the screen.

The plugin is a Cordova plugin which bridges the functionality between the web view powering the SOCIETIES Android App and the native platform the application is running on. Hence, it is composed of a single JavaScript interface which can be used across all platforms and native implementations following platform-specific requirements that the JavaScript will call into. The Javascript side consists of cordova.exec(successCallback, failureCallback, 'plugin name', 'method name', [parameters]) methods while the Java side contains a method PluginResult execute(String methodName, JSONArray arguments, String callbackId).

The JavaScript side of this feedback_plugin.js can be found in asset/www/javascript/plugins folder of the SocietiesAndroidApp. Its content is presented below

```javascript
/**
 * Provides an API for notifications
 *
 * @namespace SocietiesFeedback
 */

var SocietiesFeedback = {

    beepFeedback: function (success, fail, count)
    {
        console.log("Beeping");
        return cordova.exec (success, fail, "PluginFeedback", "beepFeedback", [count]);
    },

    vibrateFeedback: function (success, fail, time)
    {
        console.log("Vibrating");
        return cordova.exec (success, fail, "PluginFeedback", "vibrateFeedback", [time]);
    },

    onSuccess: function(data) {
        console.log("Feedback Success");
    },

    onFailure: function(e) {
        console.log("Feedback Error");
        console.log(e);
    }
};
```

The corresponding native component which performs the vibration or beeping is a Java Android file PluginFeedback.java which can be found in src/org/societies/android/platform/phonegap folder. It has the following content.

```java
public class PluginFeedback extends Plugin {

    public static final String BEEP_FEEDBACK = "beepFeedback";
    public static final String VIBRATE_FEEDBACK = "vibrateFeedback";

    @Override
    public PluginResult execute(String action, JSONArray args, String callbackContext) {
        // TODO Auto-generated method stub
        PluginResult result = null;

        if (action.equals(BEEP_FEEDBACK)){
```
try {
    this.beep(args.getInt(0));
    result = new PluginResult(Status.OK);
} catch (JSONException e) {
    e.printStackTrace();
}

else if (action.equals(VIBRATE_FEEDBACK)) {
    try {
        this.vibrate(args.getInt(0));
    } catch (JSONException e) {
        e.printStackTrace();
    }
    result = new PluginResult(Status.OK);
} else {
    result = new PluginResult(Status.INVALID_ACTION);
}
return result;

/**
 * Plays the default ringtone
 * @param count Number of times to play notification
 */
public void beep (int count) {
    Uri ringtone = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    Ringtone notification = RingtoneManager.getRingtone(this.ctx.getContext(), ringtone);
    // If phone is not set to silent mode
    if (notification != null) {
        for (int i = 0; i < count; ++i) {
            notification.play();
            int timeout = 500;
            while (notification.isPlaying() && (timeout > 0)) {
                timeout = timeout - 100;
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                }
            }
        }
    }
}

/**
 * Vibrates the device for the specified amount of time
 * @param time Time to vibrate in ms
 */
public void vibrate (int time) {
    // Start the vibration, defaults to half a second
    if (time ==0){
        time = 500;
    }
    Vibrator vibrator = (Vibrator) this.ctx.getSystemService(Context.VIBRATOR_SERVICE);
    vibrator.vibrate(time);
}

In order to apply vibration in the application, the code below needs to be added at the appropriate places.

    window.plugins.SocietiesFeedback.vibrateFeedback(window.plugins.SocietiesFeedback.onSuccess,
    window.plugins.SocietiesFeedback.onSuccess,500);

An example call during user login can be found in mainpage.js under the asset/www/javascript/pagespecific folder. The number 500 denotes the length of the vibration in milliseconds. For the beep functionality, the last parameter represents the number of beep required instead of time length. If necessary, this feature can be
extended to represent different beep patterns using different combinations of time length and number of beeps.
7 Conclusion

Task 6.5 provided guidelines for developers to follow when developing user interfaces for their services to ensure a consistent look, feel and experience. This consistency will enable trial users to have an improved interaction experience with the features of the SOCIETIES platform.