WP7 – Multi-Device Media Representation and Interaction

D7.9.2: Multi-Device Media Representation and Interaction Public Report (Second Version)

Deliverable Lead: TPVI

Contributing Partners: TPVI, ASC, TALK, TIE, NTUA

Delivery Date: 2015-12

Dissemination Level: Public

Final

This deliverable, the second of four deliverables, is the description of the second prototype implementations of the Tasks T7.1 SAM multi-device content and media representation, T7.2 SAM 2nd Screen media interaction, T7.3 SAM 1st Screen media interaction and T7.4 SAM multi-device dashboard. The deliverable covers the descriptions of the software deliverables in WP7. This document is a living document that is enhanced with each delivery of the different iterations of the WP7 prototypes.

Document Status	
Deliverable Lead	TP Vision
Internal Reviewer 1	TIE: Vadim Chepegin; V0.8; 10.2015
Internal Reviewer 2	UOR: Marco Tiemann, V.09; 11.2015
Type	Deliverable
Work Package	WP7 – Multi-Device Media Representation and Interaction
ID	D7.9.2: Multi-Device Media Representation and Interaction Public Report
Due Date	10.2015
Delivery Date	12.2015
Status	Final

Document History	
Versions	V0.1: TPVI, Initial version, starting from D7.9.1 with updated document structure and general WP7 items. V0.2: TPVI, All: Merge of all initial Partner input. V0.3 – V0.7: TPVI: Intermediate versions to fix Word Heading numbering problems V0.8: TPVI, All: Internal rework and merging of all partner inputs V0.9: Rework of Review 1 comments V0.9.1: Rework of Review 2 comments V1.0: Ready for Approval V1.1: Rework of “Before Approval” comments from TIE
Contributions	TPVI: Koen Cooreman – Document creation, structure, contribution to all sections Sushil Jha – Contributions to Sections 5, 6 and 7 Lukasz Kreft – Contributions to Sections 5, 6 and 7 ASC: Norman Wessel – Contributions to Section 4 TALK: Apostolos Apostolidis – Contributions to Sections 3 and 4 TIE: Fran Rodriguez – Contributions to Section 6 NTUA: Andreas Menychtas – Contributions to Section 6

Disclaimer

Project Partners

TIE Nederland B.V., The Netherlands

Ascora GmbH, Germany

Talkamatic AB, Sweden

TP Vision Belgium NV, Belgium

Institute of Communication and Computer Systems, National Technical University of Athens, Greece

The University of Reading, UK

Universidad de Alicante, Spain

Deutsche Welle, Germany

Bibliographic Data Services Limited, UK

Executive Summary

This public document describes the status and results of the different WP7 prototypes without exposing confidential information such as prototypes access information and source code. This confidential information can be found in the related Programme Participants Prototypes (D7.1.2, D7.2.2, D7.3.2 and D7.4.2).

This document is iteratively updated coinciding with the delivery of the different related software prototypes of WP7:

- SAM Multi-Device Content and Media Representation (T7.1)
- SAM 2nd Screen Media Interaction (T7.2)
- SAM 1st Screen Media Interaction (T7.3)
- SAM Multi-Device Dashboard (T7.4)

Each part is being developed iteratively with 3 prototypes, except for T7.4, which will have 4 prototypes, and will produce a software deliverable and an update of this public documentation (in the series D7.9.1 - .2 - .3 - .4).

As such, this second version of this deliverable (D7.9.2) describes the developments and results of the second iteration of the 4 prototypes in WP7. Although in the DOW, T7.4 (SAM multi-device dashboard) was planned to start in M20, it was decided, and also indicated in D7.9.1, to start it in M14 because it has common parts to be used by both the 1st and 2nd Screen.

At this point in time the deliverable contains the following software prototype information:

Task	Component	Software Prototype Deliverable	Due	Section
T7.1	SAM Multi-Device Content and Media Representation	D7.1.2	M25	3
T7.2	SAM 2nd Screen Media Interaction	D7.2.2	M25	4
T7.3	SAM 1st Screen Media Interaction	D7.3.2	M25	5
T7.4	SAM Multi-Device Dashboard	D7.4.2	M25	6

Figure 1: Overview of Tasks and Software Deliverables

For each of the prototypes, the following information is presented:

- **Scope and Relationship:** Describes the scope of the prototypes implementation, its purpose and the main relationships with other modules being implemented in SAM
- **Requirements and Preparations:** Introduces the information needed to deal with the prototype, in terms of technical and non-technical requirements, software to be installed, etc.
- **Installation:** Describes the steps needed to install the software, and how to build it from source code
- **Execution and Usage**: Presents the different screens and actions implemented in the prototype itself, how to access it and how to test the different implemented options
- **Limitations and Further developments:** Depicts the current prototypes limitations, and the expected improvements in the next iteration
- **Summary:** Describes the conclusions of the implementation of the first prototype

Table of Contents

1 Introduction

SAM – Dynamic Social and Media Content Syndication for 2nd Screen – is a project funded by the Seventh Framework Programme of the European Commission under Grant Agreement No. 611312. It provides a content delivery platform for syndicated data to be consumed in a contextualised social way through 2nd Screen devices.

1.1 SAM Project Overview

The current generation of Internet-connected devices has changed the way users interact with media. Previously, users were restricted to being passive and unidirectional consumers; now, they are proactive and interactive media users. They can comment on and rate a television show or film and search for related information regarding cast and crew, facts and trivia or even filming locations. They do this with both friends and wider social communities through the so-called "2nd Screen".

Another related phenomenon is "Content Syndication", which is a field of marketing where digital content is created once and delivered to consumers through various different marketing channels (devices, markets and stakeholders) simultaneously, enabling efficient content control, delivery and feedback.

However, the 2nd Screen phenomenon has grown in a disorderly manner. Tools supplied by the media provider companies (e.g. as mobile or tablet apps) limit the potential outreach and, as a result, users are not enjoying relevant contextual syndicated information. European enterprises wishing to provide services have limited methods of receiving feedback, restricting the business intelligence that can be extracted and applied in order to profit from and enrich this growing market.

SAM is reshaping the current disorganised 2nd Screen ecosystem by developing an advanced social media delivery platform based on 2nd Screen and Content Syndication within a social media context. This is achieved by providing open and standardised means of characterising, discovering and syndicating media assets interactively. Users will be able to consume and prosume digital assets from different syndicated sources and synchronised devices (e.g. connected televisions), creating more fulfilling experiences around the original media assets.

The SAM vision that is now becoming reality sees the former, out-dated system of users searching for the information they desire replaced with a new approach where information reaches users on their 2nd Screen using content syndication. This is enriched through the creation of dynamic social communities related to the user and digital asset context (e.g. profiles, preferences and devices connected). These are continuously evolving social spaces where people share interests, socialise and build virtual communities. SAM will enable syndication of comments, ratings, facts, recommendations and new information that will enrich and energise the virtual community as well as enhance personalised knowledge and satisfaction.

1.2 Deliverable Purpose, Scope and Context

The purpose of this deliverable is to accompany the software prototypes of WP7 tasks T7.1 SAM multi-device content and media representation, T7.2 SAM 2nd Screen media interaction, T7.3 SAM 1st Screen media interaction and T7.4 SAM multi-device dashboard. Each task will contribute different components to the SAM architecture that are developed

iteratively in 3 phases as per milestones 3/4/5 at M19/25/31 for tasks T7.1, T7.2 and T7.3 and in 4 phases as per milestones 3/4/5/6 at M19/25/31/37 for T7.4 and will produce a software deliverable and an update of this public documentation (D7.9.x) with the timings indicated in the following table:

Deliverable	Date
D7.9.1	M19
D7.9.2	M25
D7.9.3	M31
D7.9.4	M37

Figure 2: Deliverable Schedule

As the main focus of the tasks is the development of the software itself, this accompanying document focuses on providing a short summary of the main functionalities and on serving as user guide for the current status of the development.

1.3 Document Status and Target Audience

This document is the second iteration of the D7.9.x series and is listed in the DOW as public. It is primarily aimed at the project partners as a user guide but it also presents status information of the prototypes of the software components of WP7 to the interested public.

1.4 Abbreviations and Glossary

A definition of common terms and roles related to the realisation of SAM, as well as a list of abbreviations, are available at http://wiki.socialisingaroundmedia.com/index.php/Glossary

1.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 1 (Introduction):** Provides an overview of the entire document and the related pilot implementation, describing the main objectives, constraints and status
- **Section 2 (WP7 Introduction):** Provides an overview of WP7 goals and the WP7 prototypes
- **Section 3 (SAM Multi-Device Content and Media Representation):** Describes the latest software deliverable developed in T7.1
- **Section 4 (SAM 2nd Screen Media Interaction):** Describes the latest software deliverable developed in T7.2
- **Section 5 (SAM 1st Screen Media Interaction):** Describes the latest software deliverable developed in T7.3
- **Section 6 (SAM Multi-Device Dashboard):** Describes the latest software deliverable developed in T7.4
- **Section 7 (Document Summary):** Briefly summarises the work presented at the deliverable, as well as the overall WP7 status

In Sections 3 to 6, for each component in the SAM Architecture, the following subsections are provided:

- **Scope and Relationship:** Describes the scope of the prototype implementation, its purpose and the main relationships with other modules implemented in SAM.
- **Requirements and Preparations:** Introduces the information needed to deal with the prototype in terms of technical and non-technical requirements, software to be installed, etc.
- **Installation:** Describes the steps needed to install the software, and how to build it from source code.
- **Execution and Usage**: Presents de different screens and actions implemented at the prototype itself, how to access it, and how to test the different implemented options.
- **Limitations and Further developments:** Depicts the current prototype limitations and the expected improvements.
- **Summary:** Describes the conclusions of the implementation of the first prototype.

1.6 External Annexes and Supporting Documents

- D7.1.2: Multi-Device Content & Media Representation (2nd Prototype)
- D7.2.2: 2nd Screen Media Interaction (2nd Prototype)
- D7.3.2: 1st Screen Media Interaction (2nd Prototype)
- D7.4.2: Multi-Device Dashboard (2nd Prototype)

2 WP7 Introduction

WP7 is concerned with the multi-device representation and interaction with asset content taking into account the wide spectrum of devices with different specifications in the market.

Specific objectives of this WP include:

- To implement a framework for multi-device based media representation (T7.1)
- To provide implementations of advanced user interaction using voice recognition, Inter-Widget-Communication (IWC) and 1st Screen component detection (T7.2)
- To produce a multi-device dashboard supporting media representation and advanced user interaction (T7.3, T7.4)

The results of WP7 are a set of related applications where the consumer/user can experience all the other SAM RTD WPs. Each component is developed iteratively in 3 phases as per milestones 3/4/5 (for T7.1/2/3) and 3/4/5/6 (for 7.4) at M19/25/31/37 and will produce a software deliverable and an update of the public documentation (D7.9.x – this document series).

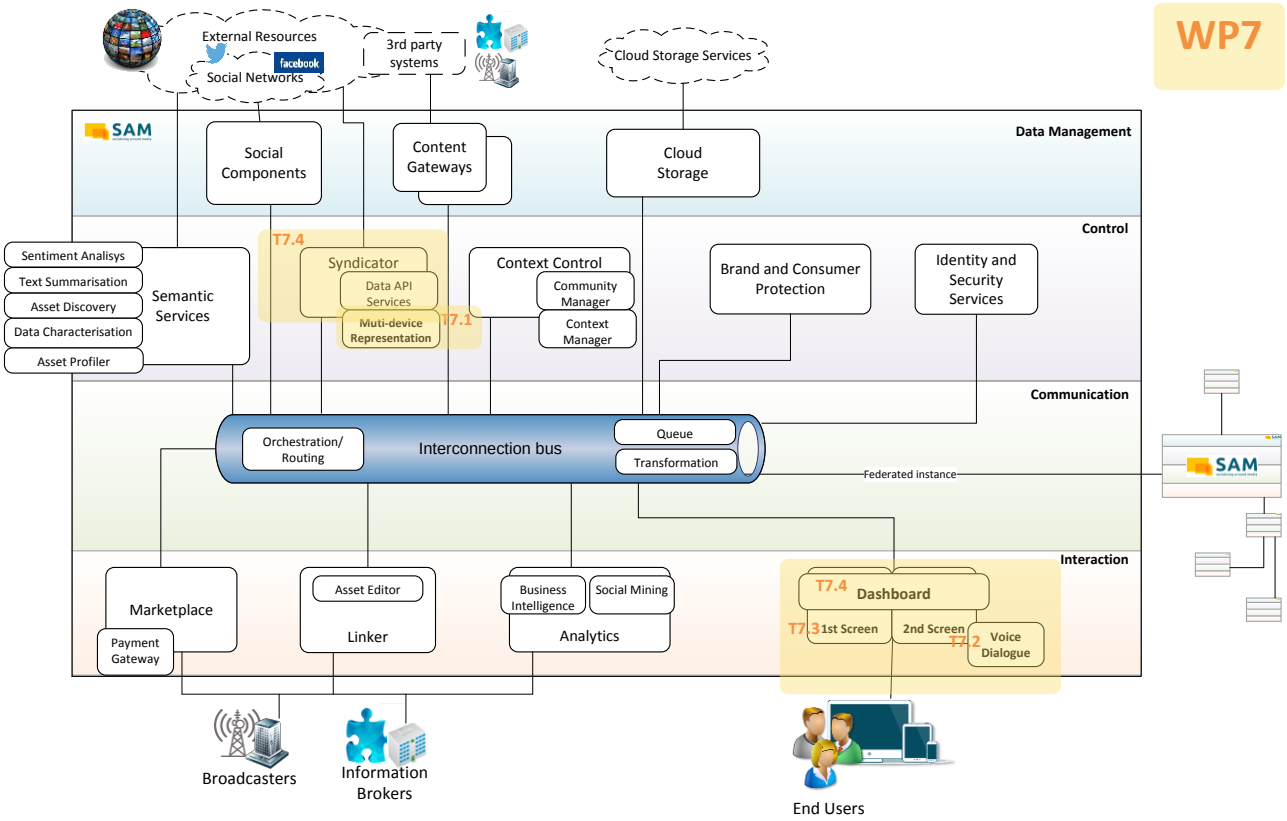


Figure 3: WP7 Components Contribution to SAM Architecture

The work in WP7 (as well as in the other development WPs) is managed by using the agile SCRUM methodology. For that purpose, a dedicated WP7 SCRUM board has been created in the SAM Jira task management system, with a representative of the WP Lead (TPVI) as the Scrum Master.

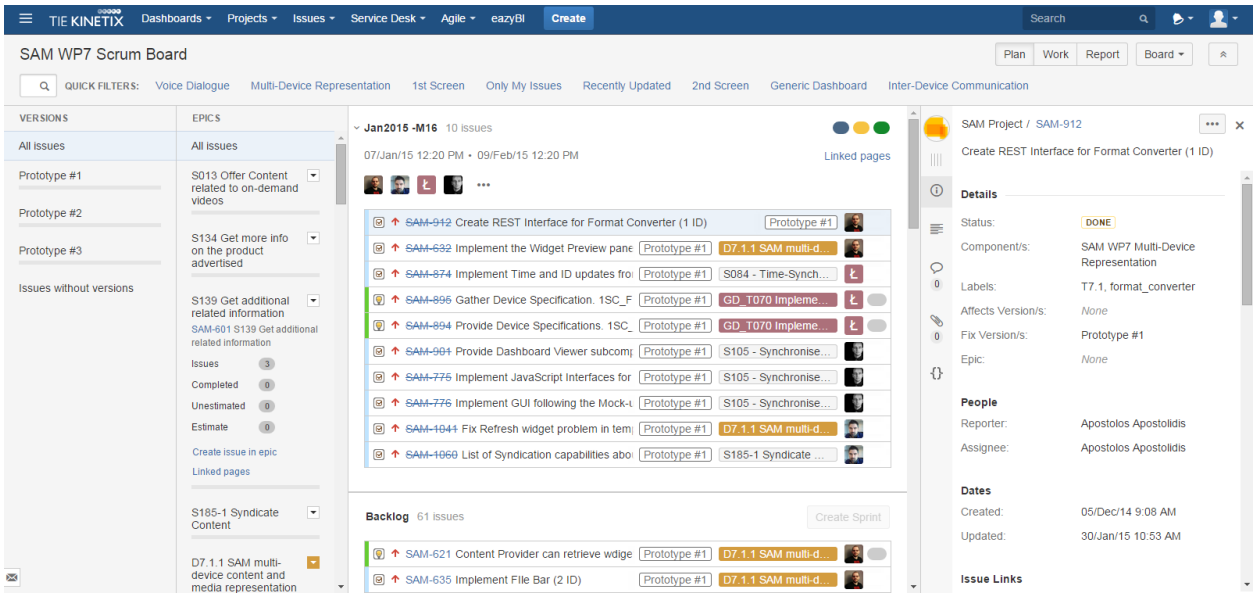


Figure 4: WP7 Scrum Board

Sprints are planned and executed monthly, and every story or task is linked to one or more specific requirements as expressed in D2.3 (User Stories and Requirements).

A Planning meeting is scheduled at the beginning of each sprint in order to plan the next monthly sprint and discuss the priorities or reschedule the unfinished work from the previous one. A Retrospective or Review meeting is also scheduled at the end of each sprint in order to discuss the work done during the sprint and find ways of improving (if necessary) the way of working.

During the development for the 2nd Prototype, two technical meetings were held with all technology partners in whom much progress was made and many of the integration and stability issues were tackled.

3 SAM Multi-Device Content and Media Representation

This section describes the software deliverable D7.1.2, which is the second prototype release of the SAM Multi-Device Content and Media Representation component.

3.1 Scope and Relationship

The Multi-Device Media Representation (MDR) component consists of two subcomponents: the Graphical Editor and the Format Converter. The former is active during Production Time while the latter during Prosumption.

Production Time is defined as the phase where Content Providers and SAM Administrators curate and manage content that will be available to the End Users. This content is syndicated to all end-users presented in the form of widgets. Each piece of information will be wrapped in an appropriate widget type.

The Prosumption is the phase where the End Users consume and interact with the data that are supplied to them.

The Graphical Editor aims to enable both the SAM Administrator and Content Owner to modify the style and functionality of the widgets that will be used to syndicate information during the Prosumption.

The Format Converter is queried by any SAM component, which needs information about widget types and their content.

Figure 5 shows the two subcomponents of MDR and the logical connections that have been established between them.

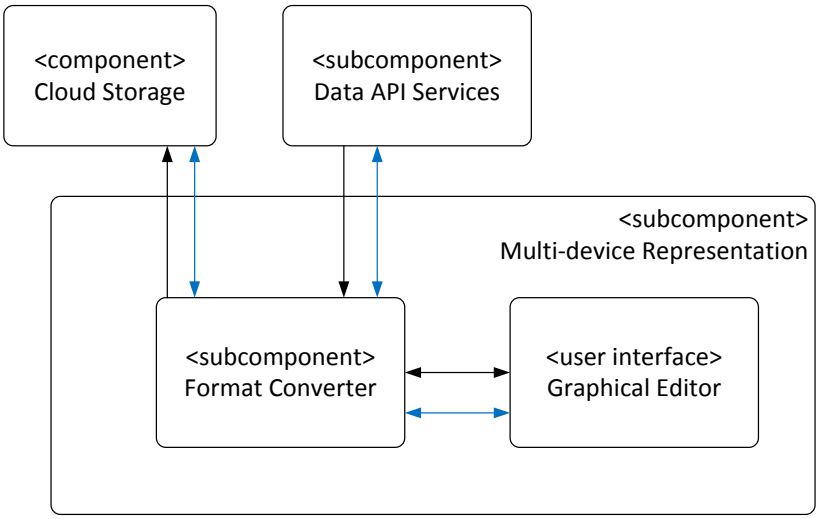


Figure 5: Multi-Device Representation (MDR) Architecture

For further descriptions of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.4.2 (Architecture), D3.2.2 Section 4.5.2 (Functional Specification) or D3.3.1 Section 3.5.2 (Technical Specification).

The 1st Format Converter prototype implemented a first approach to a mechanism that can provide HTML documents based on the asset data it is supplied with. No information on how each specific widget should be handled was embedded in this prototype.

The Graphical Editor was provided in the 1st prototype with limited editing capabilities of widget styles. The user could create new style templates for any given widget type and the

dashboard or edit existing ones. These could be created and edited with the following ways:

- The first way is to use the embedded editor which, in the first prototype, was limited in adjusting the colour values for various parts of a widget or the dashboard
- The other way of editing a style template is to upload a CSS file and thus be able to define anything that can be included in such a file

Finally, all changes could be saved in the cloud.

Apart from the various widget types, the Generic Dashboard editor offered colour configuration only for the header and the main body parts.

The custom CSS files that could be uploaded to the service were automatically applied to the currently selected widget style template. A preview area had also been implemented which displayed the current state of the style template. Finally, the Widget Gallery showed all available style templates stored in the Cloud Storage for a given widget type.

A summary of the tasks carried out for each subcomponent of the first version of the prototype is shown in the following table:

Subcomponent	Task
Format Converter	The base logic of the app has been implemented. Mock asset data can be converted to a widget according to a mock style.
Graphical Editor	• Style template preview (displaying changes in style) • Widget gallery (displaying all stored style templates) • Custom CSS upload • Colour settings editing • Style export (all changes exported to a single CSS file) • Cloud Storage integration (implemented saving & deleting style templates in the Cloud Storage) • Implemented Generic Dashboard specific style settings

Table 1: Tasks carried out during Prototype I implementation

The 2nd Format Converter prototype presents some alterations compared to the first one. It no longer injects Asset Data into widgets but rather provides the Syndicator with CSS and HTML files for it to combine them with the Asset data. Specifically, the 2nd prototype provides the following improvements:

- Provide the Syndicator with the currently active CSS style for a specific widget type
- Provide Syndicator the HTML style for a specific widget type
- Provide Widget Editor and Linker with a list of all widget types
- Provide Widget Editor with the ID of the default CSS style for a specific widget type
- Provide Syndicator and the Linker with Asset field – Widget part relations

As it can be seen in the list above, the Syndicator can retrieve from the Format Converter the CSS and HTML for any given widget type in order to be able to send it for displaying on the 1st and 2nd Screen. Likewise, Linker and the Widget Editor need to retrieve a list with all the active widget types that a user can configure.

Finally, the Syndicator and the Linker can retrieve information about which part of a widget will be used to display a specific Asset field (for example the Header widget part can display the Title Asset field). This relational list of Asset fields corresponding to Widget parts is called Asset Definitions.

The Graphical Editor in the 2nd prototype incorporates various editing capabilities of widgets and dashboard styles.

The user can select any widget type or the Dashboard and configure the colour values for specific parts of the widget / Dashboard. Also, the URL of the main image of a widget can be configured. Alternatively, the user can upload a CSS file for more advanced style editing and finally export and the current configuration of a template into a CSS file and download it. A different set of options for colour configuration is available for the Dashboard.

Any style template that is edited can be set as the default one, that is, as the one that will be assigned to the widget each time this widget needs to be displayed. A user can store multiple styles templates in the Cloud Storage, access them through the Template Gallery and set one of them as the default/active one. The Asset Definitions described above can be configured by a dropdown list next to each widget part in Widget Editor.

The functionality of the Widget can also be configured by uploading HTML code. The uploaded HTML will be first displayed in a Preview pane and upon saving it will be injected into the Dashboard every time it needs to display the specific widget.

A summary of the tasks carried out for each subcomponent of the second version of the prototype is shown in the following table:

Subcomponent	Task
Format Converter	• Get HTML (the HTML code for a specific widget is retrieved) • Get Widget types (a list of all the defined widget types is retrieved) • Get Asset definitions (a relation between a specific widget's parts and an asset's fields is retrieved)
Graphical Editor	• Widget HTML code editing • Communication with other SAM component services through TSB (and not directly with them) • Instant preview for all the CSS or HTML changes (instead of having to press a refresh button) • Asset definitions configuration through a drop-down menu next to each widget part • Extension of CSS editing (image configuration) • Improved and more robust network operations • Improved overall User Experience, including better error handling and reporting, a nicer notifications system and more intuitively designed menus and tabs

Figure 6: Tasks carried out during Prototype II implementation

3.2 Requirements and Preparations

This section provides information on technical and non-technical requirements for users as well as for developers.

3.2.1 For Users

The Graphical Editor, being a web application, is intended to be used via any web browser without any special preparations.

3.2.2 For Developers

The Format Converter is written in Python. The Graphical Editor is built with AngularJS and is mostly built with plugins and widgets taken from the common SAM template.

3.3 Installation (Deployment)

Currently, the deployment is carried out by the Jenkins Continuous Integration Server[1] provided by the SAM consortium. For the MDR component, a Jenkins integration project has been created and configured to build and deploy each subcomponent in Apache Tomcat 8.

The Graphical Editor resides in the Administration Tool and the Marketplace while the Format Converter runs as a backend service on the SAM Server.

3.4 Execution and Usage

In the following subsections the execution and usage of the Graphical Editor and Format Converter subcomponents will be explained. The “For Users” section will contain only information about the Graphical Editor, because it is a user interface available through the Administration Tool and the Marketplace. And the “For Developers” section will explain the Format Converter because it is a background service which is available to other components.

3.4.1 For Users

This section describes the steps required for a user to see all available templates for a given widget type (e.g. Facebook), create a new template for it and save it and adjust its colour settings. Also, it is shown how to import and export a CSS file. Regarding HTML, the importing process is explained. Finally, the Asset Definition configuration is explained.

3.4.1.1 See All Available Templates

In order to access this interface, the user needs to be registered with the SAM Platform and logged into the Marketplace. The Widgets option, on the left-hand side menu, provides access to the Graphical Editor, as shown in Figure 7.

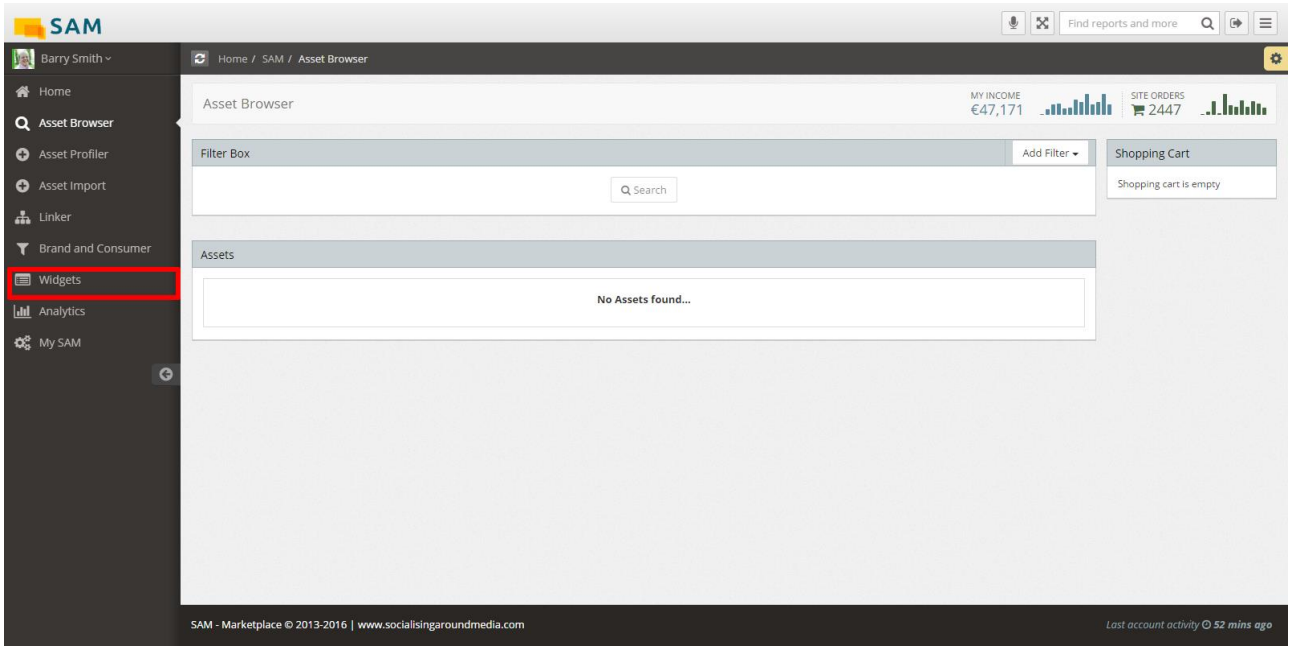


Figure 7: SAM Marketplace Home Screen

[1] http://jenkins-ci.org/

In the widgets screen, the dropdown menu at the top right corner allows to select the Widget Type (Figure 8). When the user selects one of the Widget Types (e.g., Facebook), the default style for this widget type is displayed in the HTML Preview, which is below the dropdown menu.

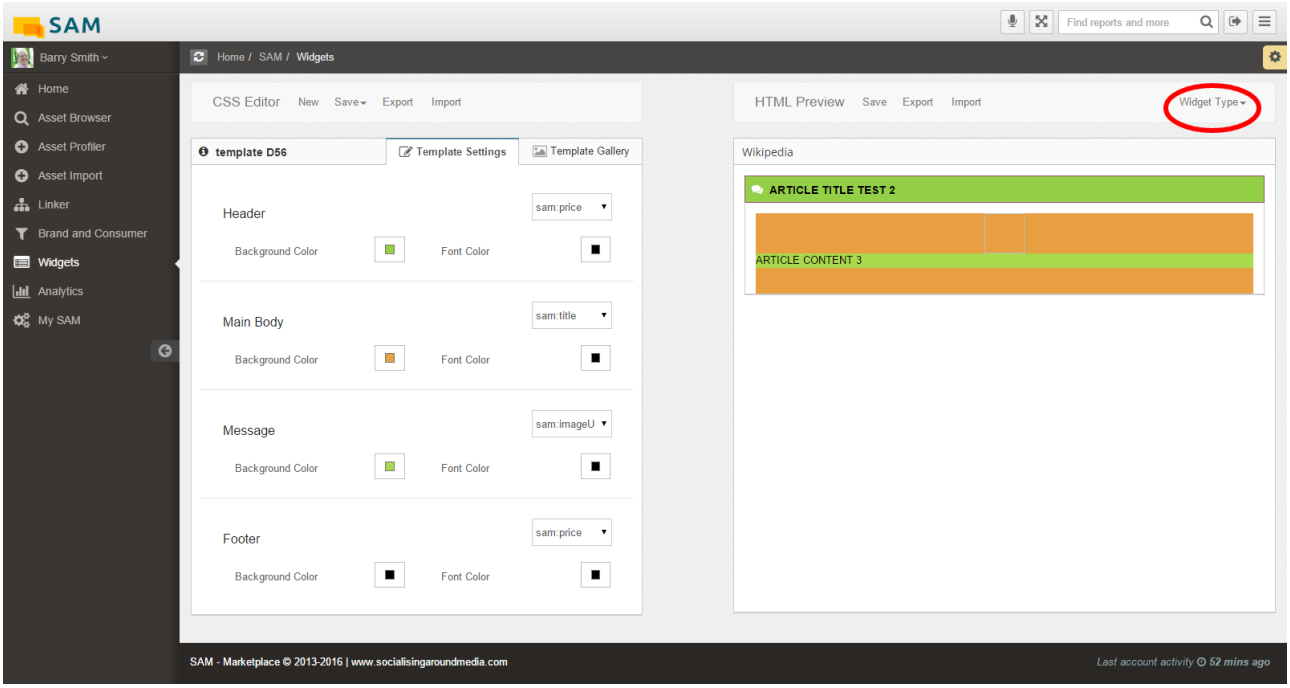


Figure 8: Widgets Component

A user can carry out the following tasks:

- Edit the widget style in the Template Settings tab
- Switch to the Template Gallery tab (Figure 9) and
  - See all available templates
  - Edit, delete or preview (unavailable for the first prototype) a template

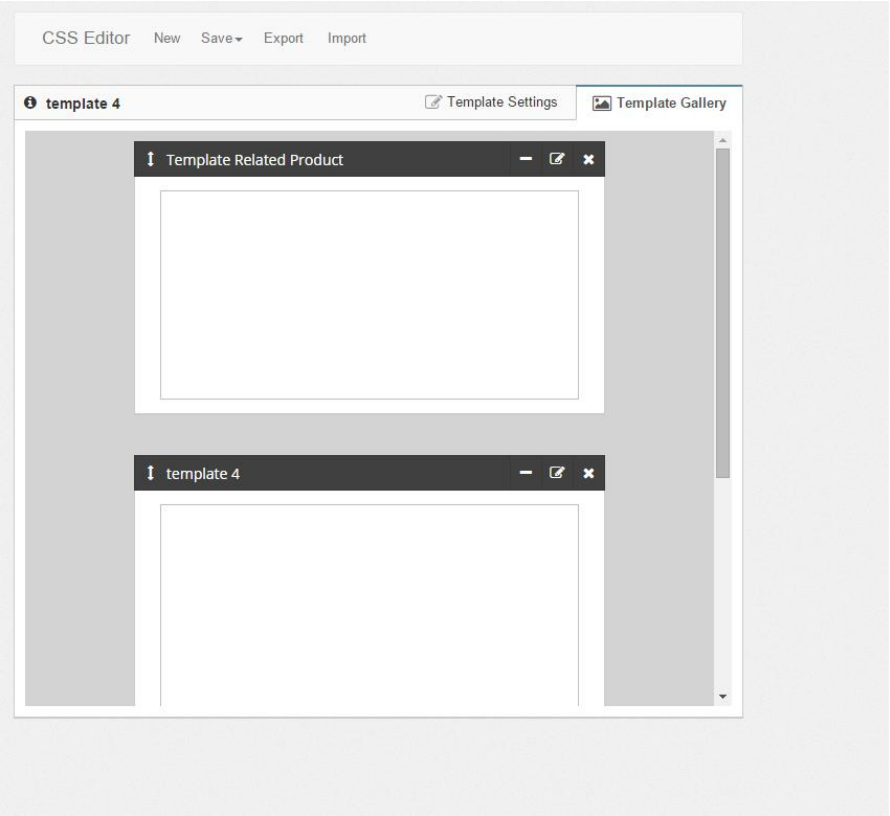


Figure 9: Template Gallery

3.4.1.2 Create a New Style

The user can create a new style by clicking the “New” button. The default style settings will be loaded and the user can select the new configuration. Once the user finishes the configuration, the style can be saved as template. This can be achieved by pressing the “Save” button and selecting the “Save As” sub-option. A popup will request a name and the saving of the template will take place (Figure 10).

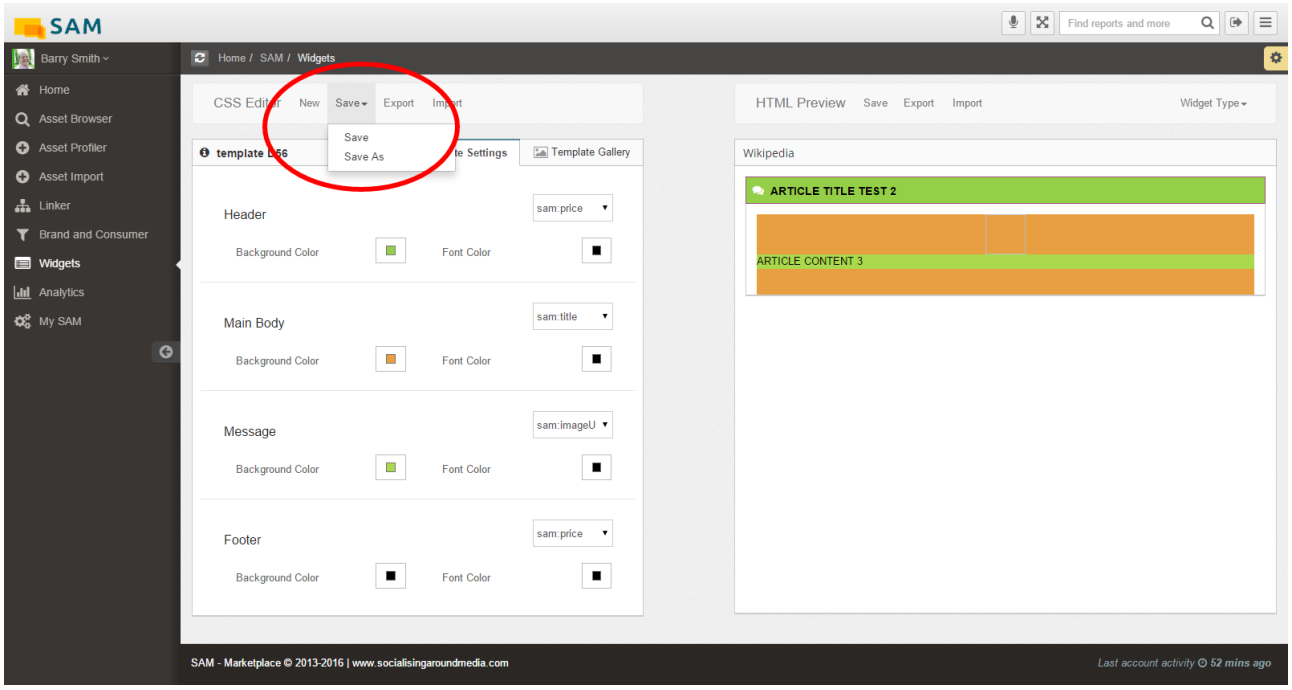


Figure 10: Create New Template and Save it

3.4.1.3 Settings Configuration

For the second prototype, colour and main picture configuration is included in the settings. Also, an Assets Definition dropdown menu is included where the user assigns the specific widget part to an Asset field. All changes made instantly appear on the Preview window.

3.4.1.4 Import/ Export CSS

At any point the user can export the current template configuration (even if it is not saved in the Cloud Storage) to a CSS file. This is done by pressing the Export button (Figure 11).

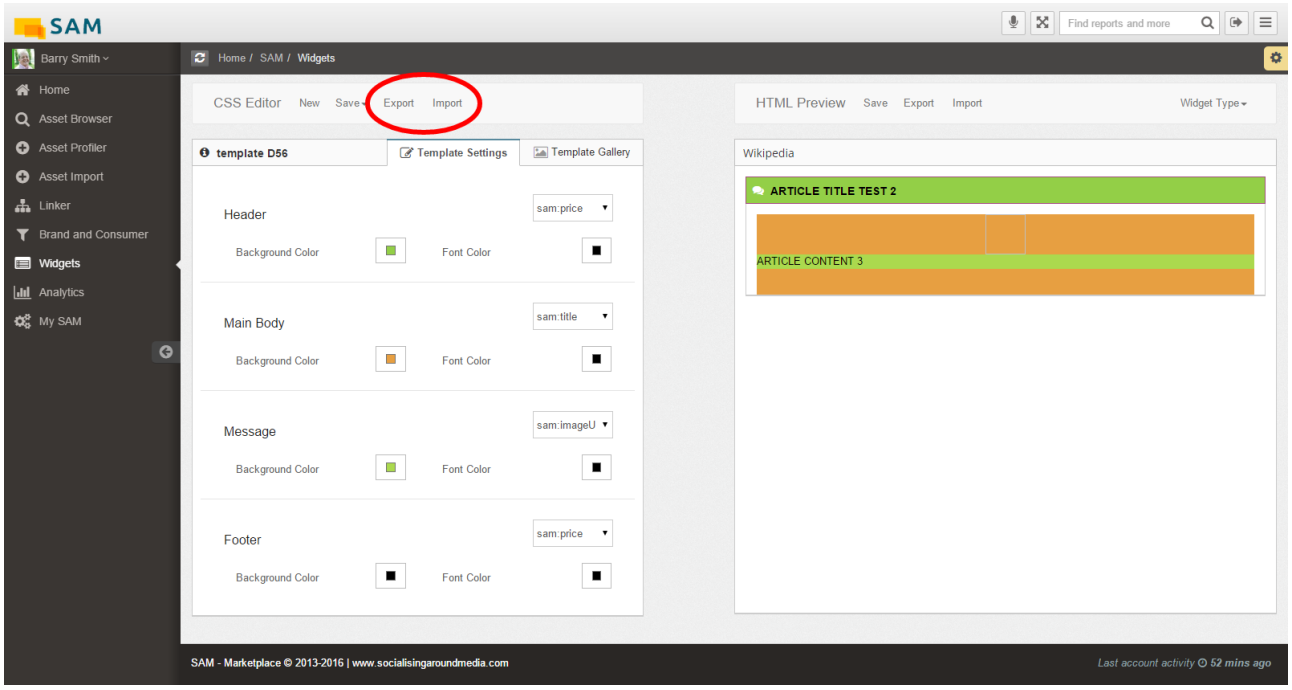


Figure 11: CSS Import/ Export

A custom CSS can be uploaded by pressing “Import”. A browser window will pop up and a CSS file can be chosen for uploading (Figure 11). Its content will be directly applied to the Preview area and the configuration settings.

3.4.1.5 Import / Export HTML

The user can update the HTML code of a Widget type and thus define its functionality. The user can update the current HTML by pressing the “Import” (Figure 12). They can also export the current HTML code (even if it is not saved) by pressing the “Export” button (Figure 12).

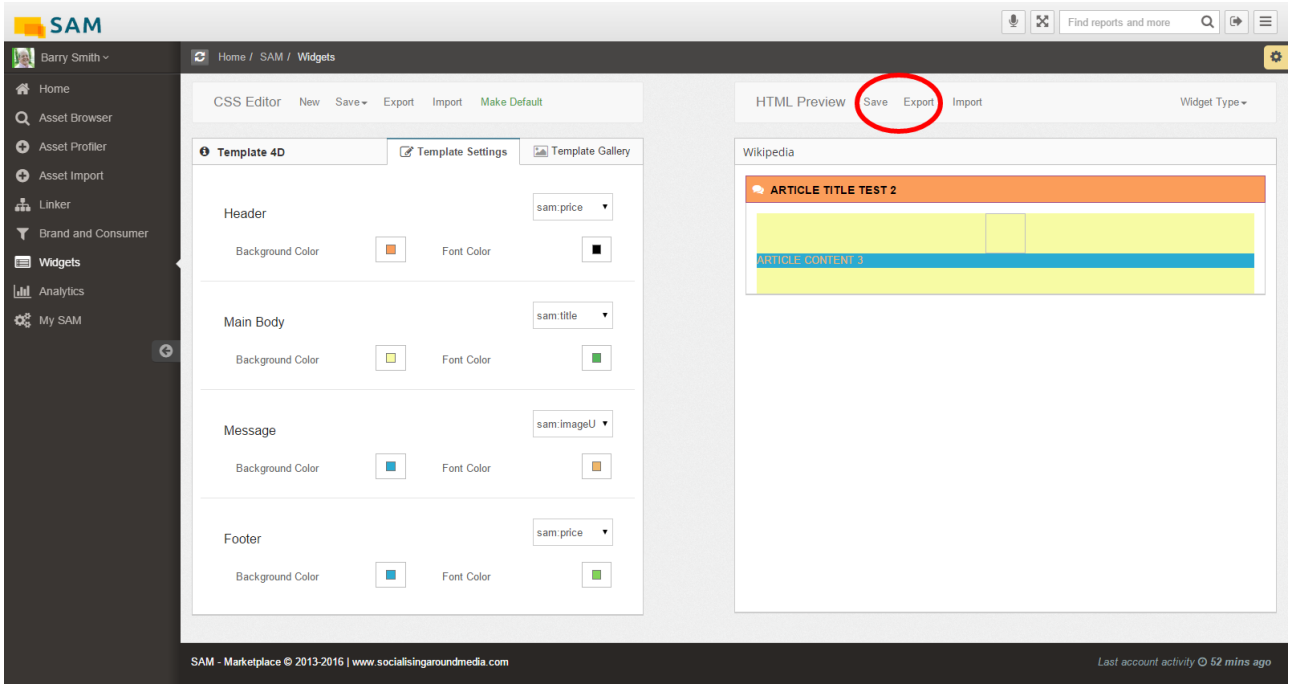


Figure 12: HTML Import/ Export

3.4.1.6 Configure Asset Definitions

The Widget Editor also allows for configuration of Asset Definitions meaning the assignment of Asset fields (e.g. title, description) on a specific widget part (e.g. main body, header). This is facilitated by selecting the desired Asset field from the dropdown menu next to each widget part (Figure 13).

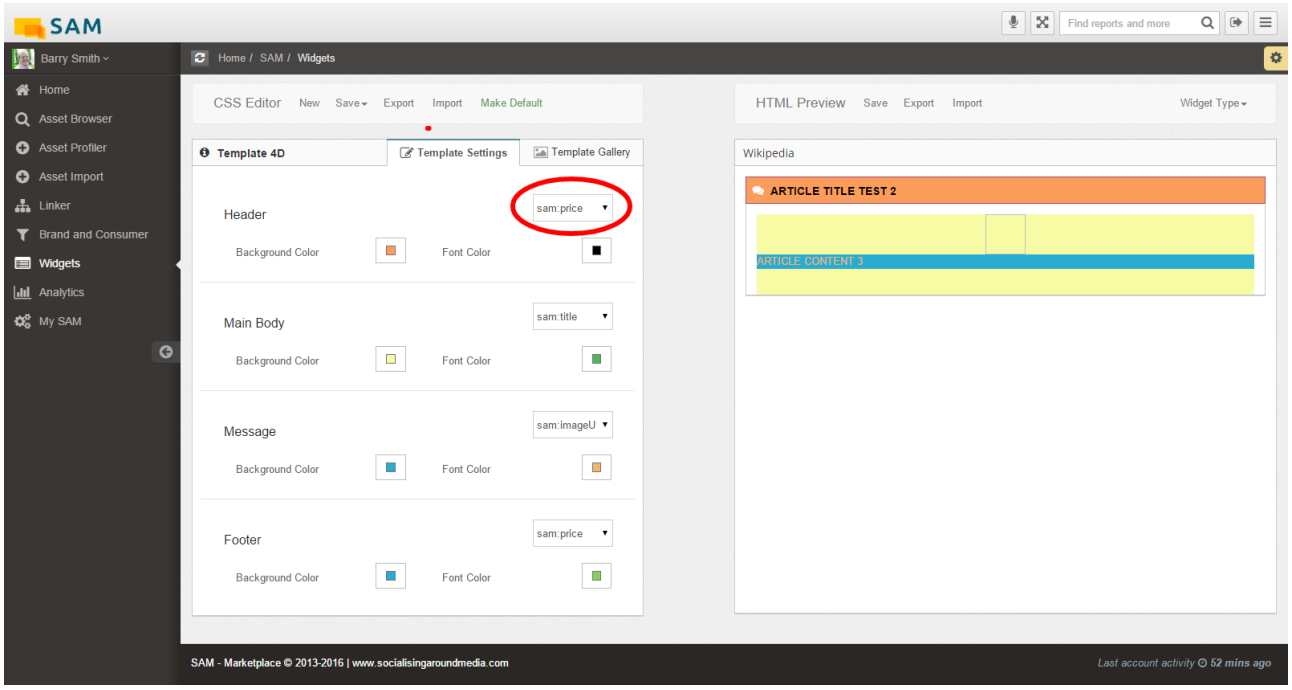


Figure 13: Configuring Asset Definitions

3.4.2 For Developers

The Graphical Editor is a graphical interface and thus not programmatically available. The Format Converter operates as a REST service available for other subcomponents. It can be queried with the REST methods described below.

3.4.2.1 Request CSS

Request CSS		
Description	Requests a CSS string with the default style for a specific widget type	
Request		
Request URL	**POST** http://localhost:9098/get_css	
HTTP Parameters	widget_type_id Required String	ID of the Widget type that the CSS will be about Example: "*Wikipedia*"
Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found

	400	Bad Request
Body	String	A string containing the requested CSSA s

Figure 14: Request CSS method

3.4.2.2 Request HTML

Request HTML		
Description	Requests an HTML string containing the code of a specific widget type	
Request		
Request URL	**POST** http://localhost:9098/get_html	
HTTP Parameters	html_id Required String	ID of the Widget type that the HTML will be about Example: "*Wikipedia*"
Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found
	400	Bad Request
Body	String	A string containing the requested HTML

Figure 15: Request HTML method

3.4.2.3 Request Widget List

Request Widget List		
Description	Requests a list with all the widgets in use	
Request		
Request URL	**POST** http://localhost:9098/get_widget_types	
Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found
	400	Bad Request
JSON Attributes	JSON Object	An array containing IDs of all widget types in use

Figure 16: Request Widget List method

3.4.2.4 Request Asset Definitions

Request Asset Definitions		
Description	Requests all the asset definitions for a given widget type	
Request		
Request URL	**POST** http://localhost:9098/get_asset_definitions	
HTTP Parameters	widget_type_id Required String	ID of the Widget type that the CSS will be about Example: "*Wikipedia"*
Response		
HTTP Status Code	Value	Description
	200	Object found
	204	Object not found
	400	Bad Request
Body	String	A string containing the requested CSSAs

Figure 17: Request Asset Definitions method

3.5 Limitations and Further Developments

The Graphical Editor has not been integrated with user account functionality. This is due to the fact this functionality was built in time for it to be integrated with the Graphical component.

3.5.1 Prototype 3 – Planned Tasks

Regarding the next steps in the MDR component, the summary of the tasks planned for the second prototype (to be delivered in M25) is as follows:

Subcomponent	Task
Format Converter	• Increase robustness (extend error handling and logging, handle connection failures) • Provide new service that will fetch all available style templates for a given widget type or Dashboard
Graphical Editor	• Extend Administrator mode • Implement user account functionality • Increase robustness (extend error handling/ logging) • Improvements based on user feedback • HTML template gallery (A gallery where previews to stock HTML codes / widgets will be hosted) • Create new widget type

Figure 18: Tasks Planned for the Third Prototype of MDR

3.6 Research Background

The following figure provides the research publications that helped to design Format Converter and the Graphical Editor.

Apart from that, considerable time was dedicated to experiment with various techniques in order to tackle specific obstacles in the development of the Graphical Editor. This experimentation consisted of reading official documentations of technologies in use and eventually adjusting code appropriately. In cases where this did not have the expected results, alternative frameworks were experimentally applied and where the results were satisfactory these frameworks were integrated in the code.

Source	Subcomponent	Description
[WDJ11] Wilson, S., Daniel, F., Jugel, U., Soi, S. (2011). Orchestrated User Interface Mashups Using W3C Widgets. Proceedings of Composable Web 2011.	Format Converter & Graphical Editor	Describes the basic principles and philosophy behind using widgets as the primary place of End User interaction.
[BWS13] Balasubramanee, V. Wimalasena, C. Singh, R. Pierce, M. (2013) Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development	Format Converter & Graphical Editor	This poster describes the experiences of Science Gateways developers of using Twitter Bootstrap and AngularJS frameworks in order to address the balance between design and implementation. It is described how using these tools eventually empowered them to create better styled and easily maintainable websites
[KAA07] Christian Kaar (2007), An introduction to Widgets with particular emphasis on Mobile Widgets	Format Converter & Graphical Editor	Mobile widgets provide an elegant way of delivering personalized web content and especially Web 2.0 services to mobile devices. This paper introduces the concept of widgets and general principles of widget development. The main section deals with the characteristics of mobile widgets and outlines differences to desktop widgets and traditional mobile application development platforms.

3.7 Target Performance

This section contains the key performance indicators (KPI) (see Section 4.7.1) and user experience measurement tasks (see Section 4.7.2) for this component.

3.7.1 Component KPIs

The Key Performance Indicators (KPIs) for this component are defined in Figure 19. The results vary depending on the indicator.

Topic	Description	Target KPI
Responsiveness	When a user is interacting with the Graphical Editor, it needs to be responsive.	Initial loading time of the Graphical Editor should be less than 5 sec. Editing actions which include cloud storage communication should be executed in less than 2 sec.
Ease of use	The ease of use is an important aspect of the Graphical Editor. Users use its UI in order to style widgets, configure their behaviour and organize these configurations in the Cloud Storage. A user-friendly environment is a necessity in order to take advantage of all these include functionalities.	Based on a short feedback questionnaire from the overall component owner the following metrics are taken under consideration. • Completion rate • Usability problems • Task time • Errors
Accessibility	The Graphical Editor should work in the most important browsers and it should be well formed XHTML markup and CSS markup.	Graphical Editor should work in Chrome, Firefox, IE and Safari. Pass the W3C validations.

Figure 19: Target Performance MDR

Regarding the Responsiveness indicator, Graphical Editor meets all requirements excluding the fetching of Asset Characterisations which occasionally may exceed the 2 seconds limit.

Regarding the Ease of use, a questionnaire has not been distributed as the component has been in heavy development. Such an activity is planned to take place during the 3rd prototype phase.

Regarding Accessibility, there has been no W3C validation yet as it is planned for the 3rd prototype phase. Occasional uses of the Graphical Editor with a browser other that the development one (Chrome) has showed no problems.

3.7.2 User Experience Measurements Tasks

Additional to the KPIs in Section 4, this work package provides user tasks, which form an input for measuring the subjective user experience in a uniform way. For each of the tasks below the task-specific KPIs, defined in

References

[WDJ11] Wilson, S., Daniel, F., Jugel, U., Soi, S. (2011). Orchestrated User Interface Mashups Using W3C Widgets. Proceedings of Composable Web 2011.

[BWS13] Balasubramanee, V. Wimalasena, C. Singh, R. Pierce, M. (2013) Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development

[KAA07] Christian Kaar (2007), An introduction to Widgets with particular emphasis on Mobile Widgets

[LAR02] Larsson, S., (2002). Issue-based Dialogue Management, Gothenburg Monographs in Linguistic

[BCL05] B. Bringert, R. Cooper, P. Ljunglöf, A. Ranta, Multimodal Dialogue System Grammars. Proceedings of DIALOR'05, Ninth Workshop on the Semantics and Pragmatics of Dialogue, Nancy, France, June 9-11, 2005, 2005

[BRI07] B. Bringert. Speech Recognition Grammar Compilation in Grammatical Framework SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing, June 29, 2007, Prague. 2007.

[ZIE13] Ziegler, C., "Second screen for HbbTV — Automatic application launch and app-to-app communication enabling novel TV programme related second-screen scenarios," Consumer Electronics Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on, vol., no., pp.1, 5, 9-11 Sept. 2013

[VAN12] Vanattenhoven, J., Geerts, D. (2012). Second-Screen Use in the Home: An Ethnographic Study. In Proceedings 3rd International Workshop on Future Television, EuroITV 2012 (p. 12).

[VAN14] Vanattenhoven, J., Geerts, D., De Grooff, D. (2014). Television Experience Insights from HbbTV. In proceedings of the International Workshop on Interactive Content Consumption, Newcastle upon Tyne

[BAS13] Bassbouss, L.; Tritschler, M.; Steglich, S.; Tanaka, K.; Miyazaki, Y., "Towards a Multi-screen Application Model for the Web," Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual , vol., no., pp.528,533, 22-26 July 2013

(2015-10) ETSI TS 102 796 V1.3.1 / HbbTV 2.0

[WCJ14] Lee, Wei-Po, Che Kaoli, and Jhih-Yuan Huang. "A smart TV system with body-gesture control, tag-based rating and context-aware recommendation." Knowledge-Based Systems 56 (2014): 167-178.

[MAY14] Maynard, G. HbbTV and Multi-Screen Strategy. HbbTV Symposium Asia. Singapore 18 June 2014.

Annex A: User Research, will be measured.

Task	Description
Customize default template	The user must find the default template of a specific widget type she wants to edit and then perform a number of alterations to the template's settings.
Manage templates	The user must promote another template to the default status, delete a template, create a new one and rename one.
Import and export template	The user must import and export templates.

Figure 20: Target Performance MDR – Task-Specific Measurements

3.8 Summary

This section provides a description of the second prototype of the Multi Device Representation component developed in task T7.1 Multi-Device Content and Media Representation. The main outcomes of this task are two pieces of software: the Graphical Editor and the Format Converter. This prototype is the second of the three iterations planned for this component.

The prerequisites necessary for both users and developers to install and use the Graphical Editor and Format Converter have been described.

Regarding the Format Converter, services for retrieving styles and code from any widget type have been implemented. For the Graphical Editor, the component is now using TSB, supports HTML editing and has improved overall user experience.

The last section has been dedicated to describing the limitations of the current prototype and describing the next steps considered for the third version of the component, which should be delivered in M31.

4 SAM 2nd Screen Media Interaction

This section describes the software deliverable D7.2.2, which is the second prototype release of the SAM 2nd Screen component.

4.1 Scope and Relationship

The 2nd Screen component enables the prosumption of content corresponding to the current video, which the end user is viewing. Figure 21 shows the different subcomponents of the 2nd Screen component, the logical connections that have been established between them and the relations with other components and actors in the SAM platform.

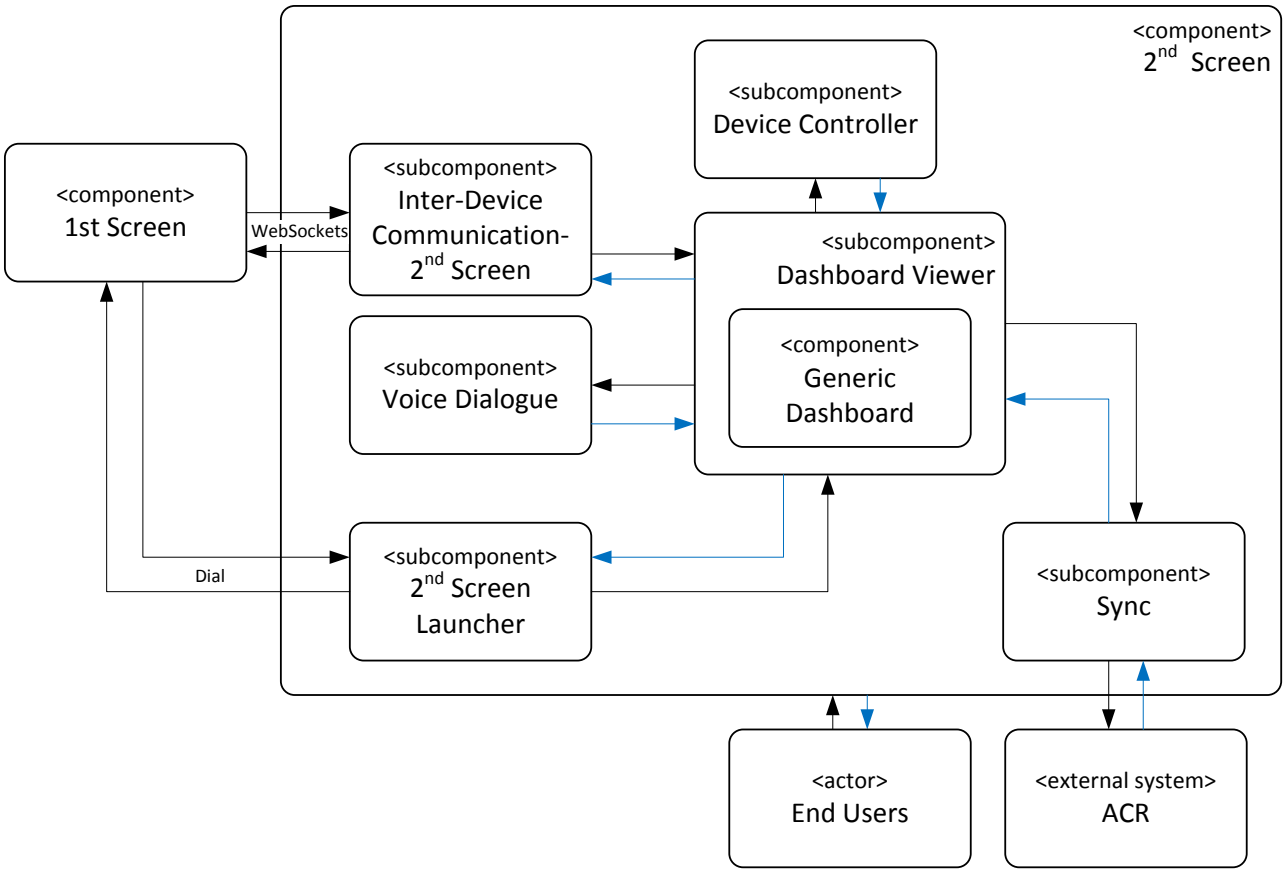


Figure 21: 2nd Screen Subcomponents and its Relationships

For further descriptions of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.13.4 (Architecture), D3.2.2 Section 4.14.3 (Functional Specification) or D3.3.1 Section 3.14.3 (Technical Specification).

The first prototype of T7.2 provided the basic integration of the Generic Dashboard component using the Dashboard Viewer subcomponent. The Dashboard Viewer subcomponent provided JavaScript interfaces which provide device specific information by using the Device Controller subcomponent. A basic remote control function had also been implemented which allows the 2nd Screen Android application user to start, stop and pause the video element displayed on the 1st Screen.

Additionally, a first integration of the Voice Dialogue subcomponent and the Talkamatic Dialogue Manager (TDM) backend had been carried out to support voice interaction using

the 2nd Screen Android application. In the first prototype, the Voice Dialogue only receives user input and gives mock feedback, not interacting with the widgets on the 2nd Screen.

A summary of the tasks carried out for the different subcomponents of the first version of the prototype is shown in the following table:

Subcomponent	Task
Voice Dialogue	Basic integration into the Android application and setup of the TDM back end
Dashboard Viewer	Display and remote control support of the Generic Dashboard component, Provide JavaScript interfaces for communication
Device Controller	Provision of device specifications and GPS data to the Generic Dashboard component using the Dashboard Viewer subcomponent

Figure 22: Tasks completed for the first prototype of T7.2

The second prototype of T7.2 adds the integration of the Identity and Security services component, which enables end users to log into the 2nd Screen application. This process is based on OpenID. Additionally, a local connection between 1st Screen and 2nd Screen devices has been implemented. The search for local 1st Screen devices has been realised by Android's "Network Service Discovery" (NSD) feature. Also, the user interface for a remote control has been updated to match the Android design recommendations and to provide a common look and feel.

On the side of the Voice Dialogue (VD) support, the Twitter use case has been implemented. This enables the end user to tweet by using voice commands. This feature is based on the dictation functionality, which also has been integrated in the second prototype.

VD has been converted to support a new integration model developed by Talkamatic. This model allows the GUI, in the 2nd Screen case, to decide which actions and data it will share with VD. Previously the 2nd Screen app executed actions upon request from VD, while now the 2nd screen decides whether and how certain events dispatched from VD will be processed. This allows for a more flexible implementation of the VD component by the parent app. Currently, VD supports launching popups, ASR and TTS events and simple actions e.g. posting on Twitter

A summary of the tasks completed for the different subcomponents of the second version of the prototype are shown in the following table:

Subcomponent	Task
2nd Screen Launcher	• Implement NSD support to discover local 1st Screen devices
Dashboard Viewer	• Provide integration of Identity and Security Services component to provide login functionality based on OpenID • Replace existing remote control implementation with a new approach based on Android design recommendations
Voice Dialogue	• Implementation of the new integration model • Implementation of dictation • Support for the Twitter use case

Figure 23: Tasks completed for the second prototype of T7.2

4.2 Requirements and Preparations

This section provides information on technical and non-technical requirements for users as well as for developers.

4.2.1 For Users

The user needs to provide an Android device (smartphone or tablet) running Android 4.4 or newer[2]. Since the download of the 2nd Screen Android application is not available on Google Play[3] yet, the following steps have to be undertaken to retrieve the installation file and install the application on an Android device:

1. Download the installation file of the application from the Jenkins Continuous Integration server[4] provided by the SAM consortium
2. Start your Android device
3. Open the Settings
4. Go to the Security section
5. Activate the option “Unknown Sources”
6. Move the installation file to the device
7. Start the installation process

Further information about the retrieval of the 2nd Screen application can be found in the D7.2.2 deliverable.

4.2.2 For Developers

For developers it is strongly recommended to use the Android Studio[5] IDE for the implementation. Besides of the usage of a physical device, developers can also use an Android Virtual Device (AVD), which is provided by Android SDK.

4.3 Installation (Deployment)

This section provides guidelines on how to install the app (see Section 4.3.1) and to retrieve the source code of the 2nd Screen application (see Section 4.3.2).

4.3.1 For Users

After downloading the 2nd Screen application and moving it to the device’s internal or external storage, the user has to execute the *.apk file. As the first step, the user will be requested for the acceptance of the access rights (see Figure 24). After the installation, the 2nd Screen application can be started using the icon in the app overview.

[2] http://www.android.com/versions/lollipop-5-0/
[3] https://play.google.com/store
[4] https://jenkins-ci.org/
[5] http://developer.android.com/sdk/index.html

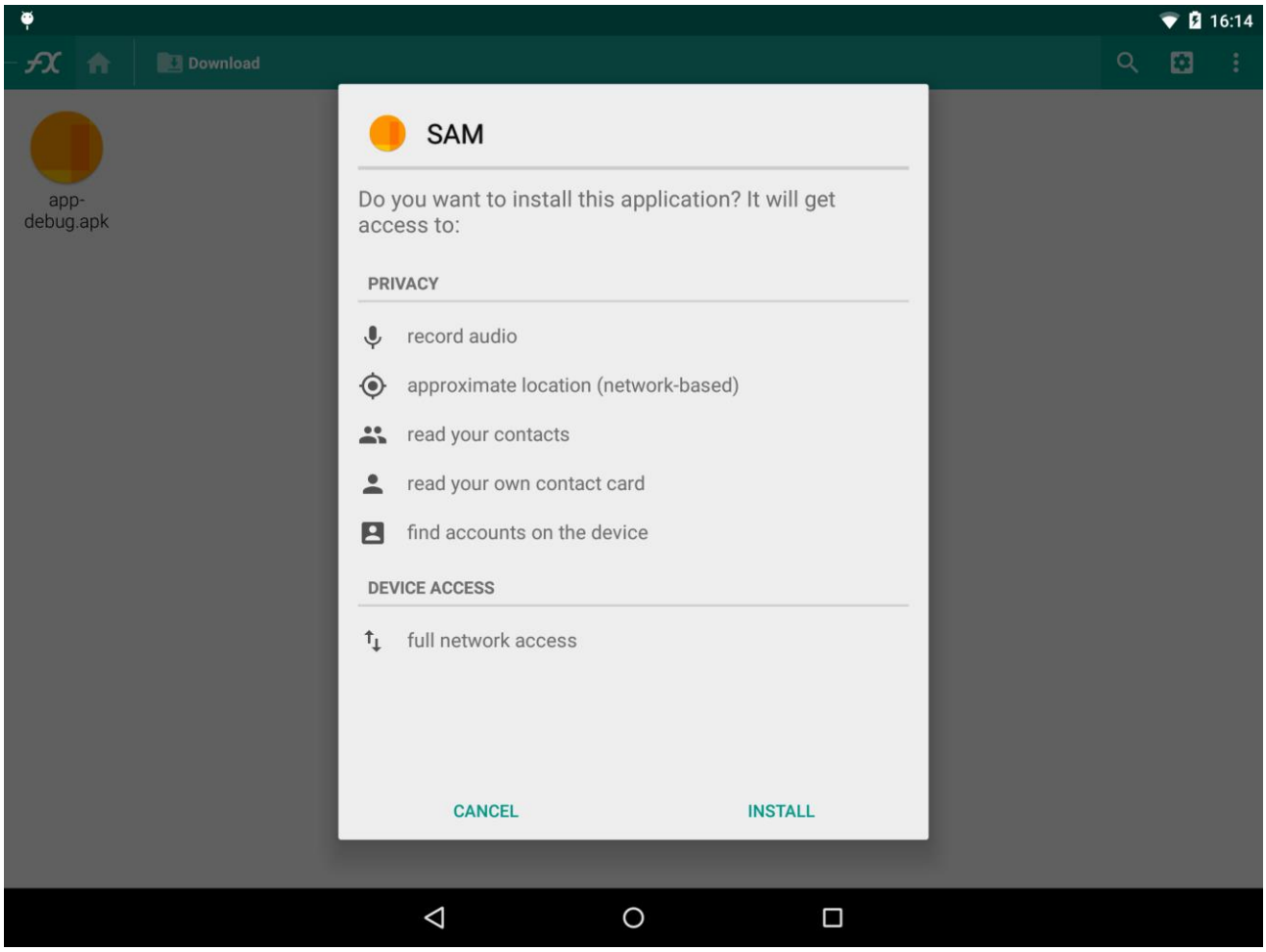


Figure 24: Installation of SAM 2nd Screen Application

4.3.2 For Developers

Developers have to download the source code first (Deliverable D7.2.2 contains the required information). The source code of the 2nd Screen application is an Android Studio project and it can be opened with Android Studio after downloading is finished (due to the fast development pace of Android Studio, adaptations can be necessary to start the project). Figure 25 shows the 2nd Screen project loaded in Android Studio.

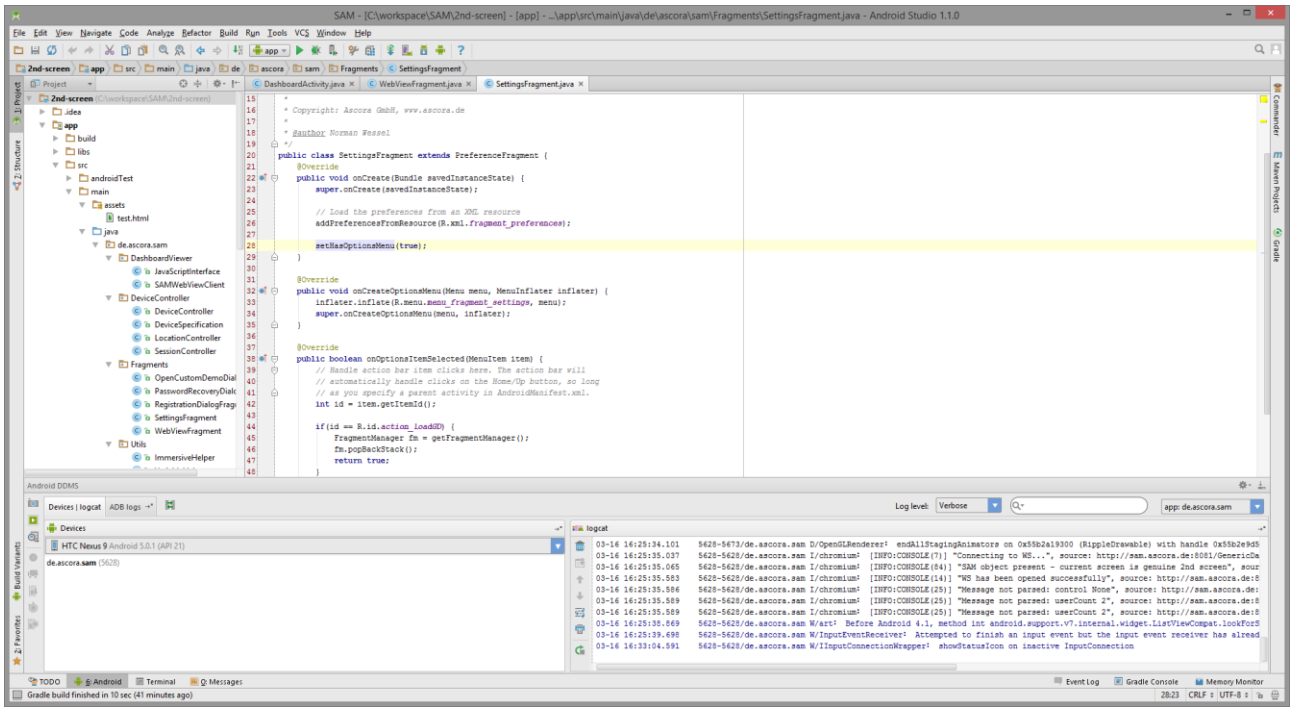


Figure 25: Android Studio IDE

4.4 Execution and Usage

This section describes how to use the prototype both as user or developer.

4.4.1 For Users

The following subsections present the different views of the 2nd Screen application.

4.4.1.1 Login Mask

To use the application, the user has to have a user account. If the user is not logged in, the 2nd Screen application shows a login screen (see Figure 26). The login screen consists of two form fields required for the login. Besides the implemented login process, there are two more buttons which allow the user to register (see Figure 27) or to start the password recovery process (see Figure 28). These two processes are still to be implemented.

Figure 26: 2nd Screen Application – Login Screen

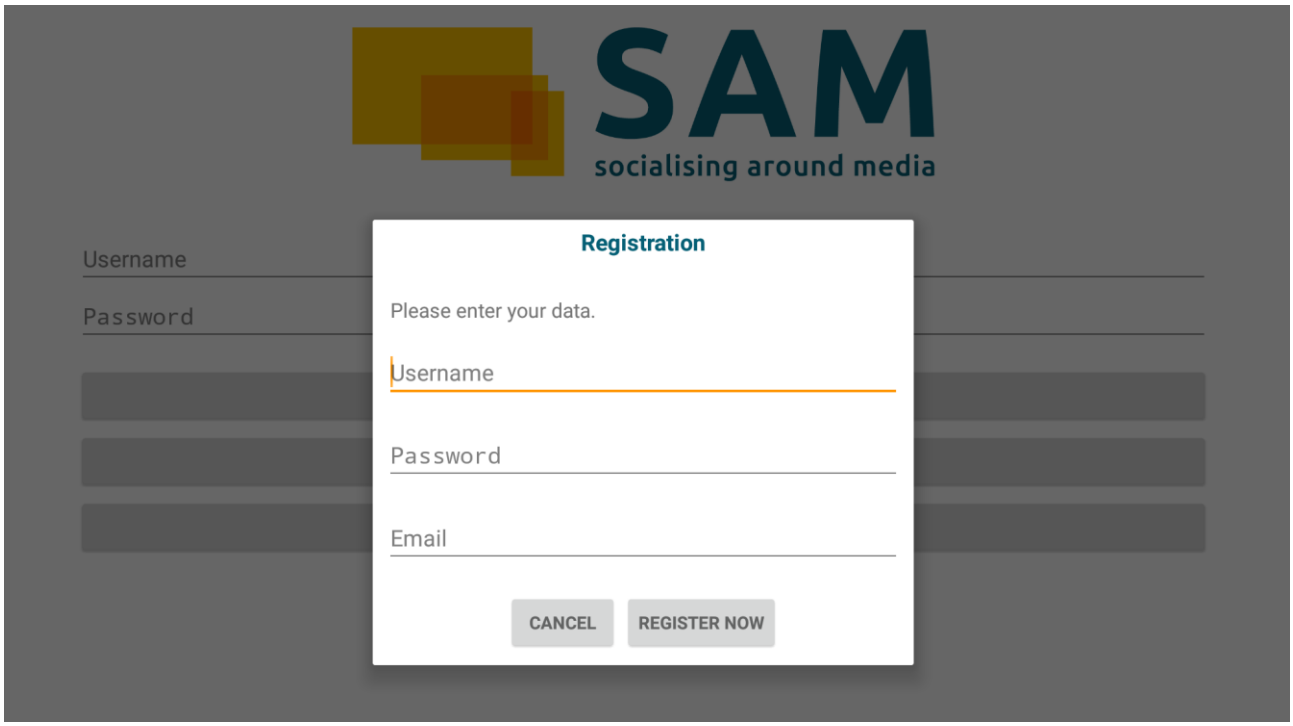


Figure 27: 2nd Screen Application – Registration Screen

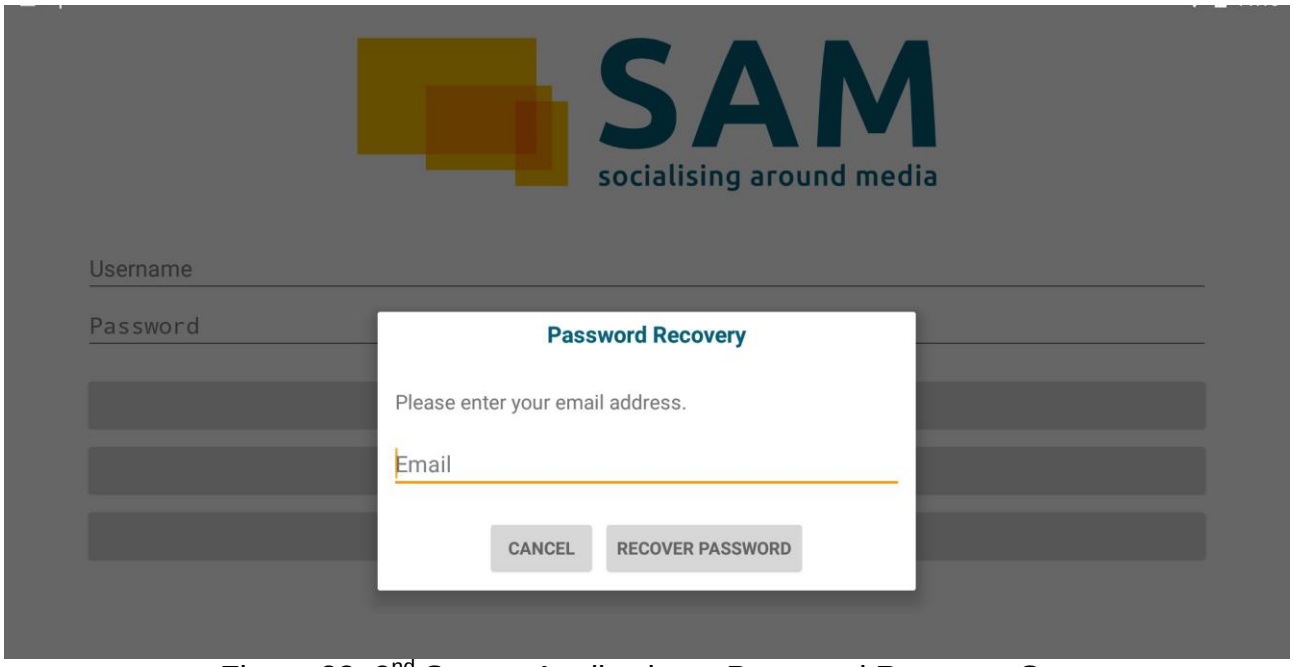


Figure 28: 2nd Screen Application – Password Recovery Screen

4.4.1.2 Dashboard

After the user successfully signed in, the Generic Dashboard is displayed (see Figure 29). In the action bar at the top of the Generic Dashboard the remote control functions are present (see Figure 30). The remote control contains three buttons: “Play”, “Pause”, and “Stop”.

Additionally, settings are available in the preferences screen by selecting the options menu (⋮) in the upper right corner.

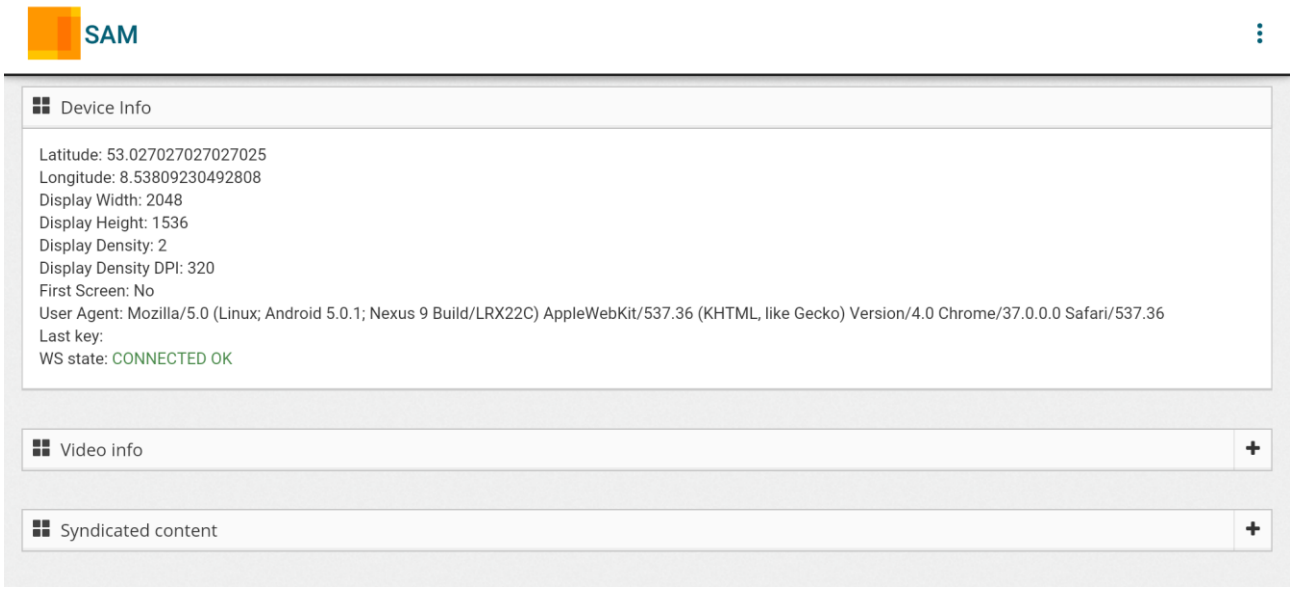


Figure 29: 2nd Screen Application – Generic Dashboard Screen

Figure 30: 2nd Screen Application – Remote Control

4.4.1.3 1st Screen Discovery

Since this second iteration the 2nd Screen application supports the discovery of existing 1st Screen devices in the same local network (see Figure 31). This feature uses the Network Service Discovery technology provided by the Android SDK.

To search the local network for 1st Screen devices, the user has to tap on the button shown in Figure 31. This will trigger a progress bar while the search is on-going. After a short time a popup with existing 1st Screen devices will be shown (Figure 32). By clicking on a listed 1st Screen, a local connection will be established which is used for synchronisation and control means.

Figure 31: 2nd Screen Application – 1st Screen Discovery Button

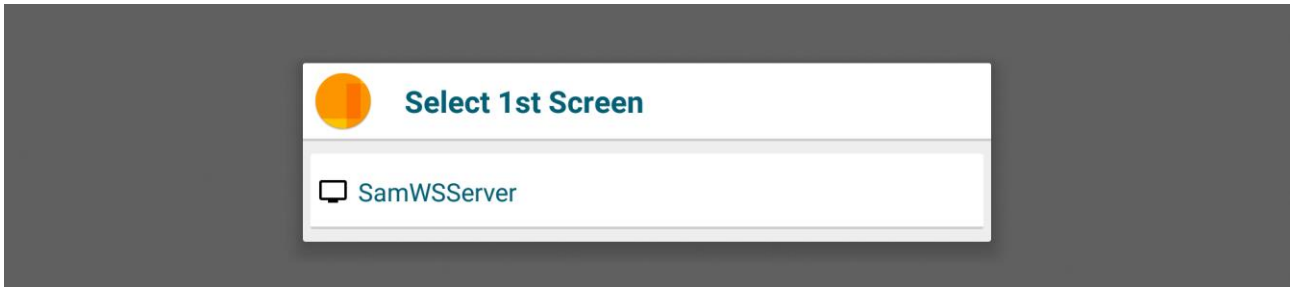


Figure 32: 2nd Screen Application – 1st Screen Discovery Popup

4.4.1.4 Voice Dialogue

To start interacting with the Voice Dialogue subcomponent, the user has to touch the microphone icon from the menu at the top right corner. The system introduces itself verbally as soon as the application is launched. Then, whenever the users desire to provide input they have to press the microphone button (Figure 33).

Figure 33: Voice Dialogue

In the second prototype only the Twitter interaction is supported, that is the user can dictate a tweet and have it posted to Twitter (Figure 34):

SAM> What would you like to do?

User> share my film on twitter

Sam> what comment should I post?

User> I think I really like Casino Royale

SAM> Media shared with comment I think I really like Casino Royale!

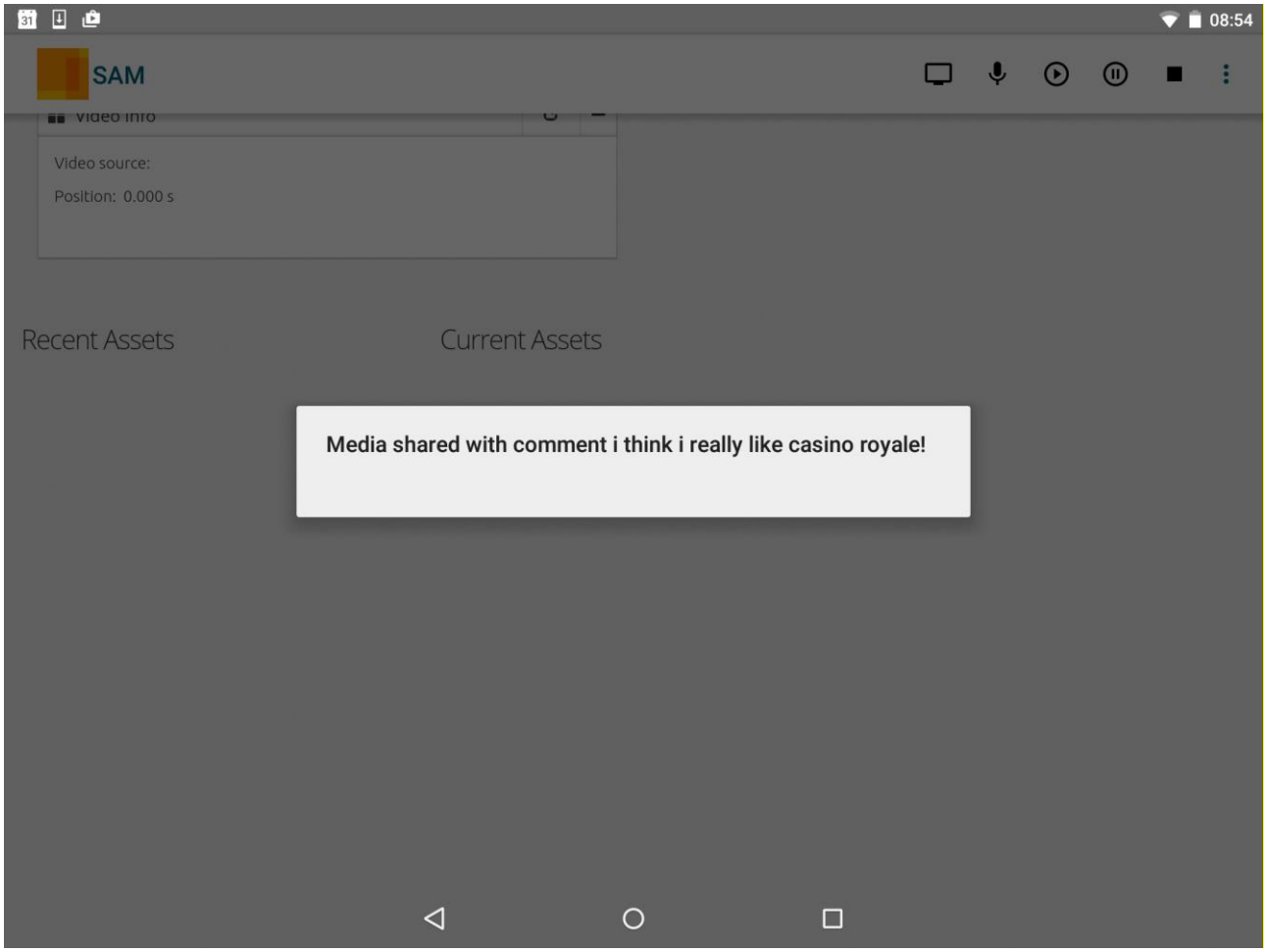


Figure 34: Media Sharing Confirmation with Dictation Results

4.4.2 For Developers

The communication between the Generic Dashboard component and the Dashboard Viewer subcomponent is based on JavaScript. The WebView element of the Dashboard Viewer supports customised JavaScript interfaces, which can be injected into the website displayed in the WebView and therefore enables the communication between the Generic Dashboard component and the Dashboard Viewer subcomponent. The JavaScript methods documented in Section 4.4.2.1 have been implemented and can be used by the Generic Dashboard component. To test these JavaScript interfaces a HTML test page has been set up. To show this test page, the user has to enable the Debug options in the App Settings screen of the 2nd Screen Android application.

Figure 35: 2nd Screen – Test Page Screenshot

4.4.2.1 JavaScript Interfaces

This section provides the documentation of the JavaScript interfaces required for the communication between the Dashboard Viewer subcomponent and the within presented website. The JavaScript interfaces are injected by the Dashboard Viewer subcomponent and are available using the namespace “SAM”.

4.4.2.1.1 Get GPS Coordinates - Latitude

Get GPS Coordinates – Latitude	
Description	Returns the latitude value of the GPS sensor. Availability depends on the settings and accuracy depends on the device
Method Header	SAM.getGPSCoordinatesLatitude()
Parameters	None
Return Value	Double
Error Handling	In case of an error, 0.0 is returned
Remarks	Only available on GPS-enabled devices

Figure 36: 2nd Screen JavaScript Interface – Get GPS Coordinates – Latitude

4.4.2.1.2 Get GPS Coordinates - Longitude

Get GPS Coordinates – Longitude	
Description	Returns the longitude value of the GPS sensor. Availability depends on the settings and accuracy depends on the device
Method Header	SAM.getGPSCoordinatesLongitude()
Parameters	None
Return Value	Double
Error Handling	In case of an error, 0.0 is returned
Remarks	Only available on GPS-enabled devices

Figure 37: 2nd Screen JavaScript Interface – Get GPS Coordinates – Longitude

4.4.2.1.3 Get Display Width

Get Display Width	
Description	Returns the width of the display in the current orientation (Landscape/Portrait)
Method Header	SAM.getDisplayWidth()
Parameters	None
Return Value	Integer
Error Handling	None
Remarks	None

Figure 38: 2nd Screen JavaScript Interface – Get Display Width

4.4.2.1.4 Get Display Height

Get Display Height	
Description	Returns the height of the display in the current orientation (Landscape/Portrait)
Method Header	SAM.getDisplayHeight()
Parameters	None
Return Value	Integer
Error Handling	None
Remarks	None

Figure 39: 2nd Screen JavaScript Interface – Get Display Height

4.4.2.1.5 Get Display Density

Get Display Density	
Description	Returns the density value of the display as an integer. ”For simplicity, Android groups all actual screen densities into six generalized densities: low, medium, high, extra-high, extra-extra-high, and extra-extra-extra-high.”[6]
Method Header	SAM.getDisplayDensity()
Parameters	None
Return Value	Integer
Error Handling	None
Remarks	None

Figure 40: 2nd Screen JavaScript Interface – Get Display Density

[6] http://developer.android.com/guide/practices/screens_support.html#terms

4.4.2.1.6 Get Display Density in DPI

Get Display Density DPI	
Description	Returns the density of the display in DPI (Dots per Inch)
Method Header	SAM.getDisplayDensityDpi()
Parameters	None
Return Value	Integer
Error Handling	None
Remarks	None

Figure 41: 2nd Screen JavaScript Interface – Get Display Density DPI

4.4.2.1.7 Start Voice Interaction

Start Voice Interaction	
Description	Calls the voice interaction method in the Voice Dialogue subcomponent
Method Header	SAM.handlePTTPushed()
Parameters	None
Return Value	None
Error Handling	None
Remarks	None

Figure 42: 2nd Screen JavaScript Interface – Start Voice Interaction

4.4.2.1.8 Logout

Logout	
Description	Logs out the user and invalidates the token
Method Header	SAM.logout()
Parameters	None
Return Value	None
Error Handling	None
Remarks	None

Figure 43: 2nd Screen JavaScript Interface – Logout

4.4.2.1.9 Get User Token

Get User Token	
Description	Returns the current user token
Method Header	SAM.getUserToken()
Parameters	None
Return Value	String
Error Handling	In case of an error, null is returned
Remarks	None

Figure 44: 2nd Screen JavaScript Interface – Get User Token

4.4.2.1.10 Get Logged-In Username

Get Logged-In Username	
Description	Returns the name of the current user
Method Header	SAM.getLoggedInUserName()
Parameters	None
Return Value	String
Error Handling	In case of an error, null is returned
Remarks	None

Figure 45: 2nd Screen JavaScript Interface – Get Logged-In Username

4.4.2.1.11 Get SAM ID

Get SAM ID	
Description	Returns the SAM ID of the current user
Method Header	SAM.getSAMID()
Parameters	None
Return Value	String
Error Handling	In case of an error, null is returned
Remarks	None

Figure 46: 2nd Screen JavaScript Interface – Get SAM ID

4.5 Limitations and Further Developments

Development for the second prototype has been concentrating on the further development of the Android application and the integration and communication of the Identity and Security Services component and the enabling of a local connection between 1st Screen and 2nd Screen devices. Due to that, the following views/functions have been provided as mock-ups for later development:

- **Register User Dialog:** The register user dialog will enable the user to create a SAM user account by adding personal information
- **Password Recovery User Dialog:** The password recovery user dialog will enable the user to retrieve a lost password by resetting the password and sending it back to the user

Additionally, the Voice Dialogue (VD) interaction is currently limited in two ways:

- **Widget Functionalities:** No other interaction except the Twitter interaction is supported
- **Graphical Interaction:** Apart from launching popups VD cannot instruct the Dashboard for any other visual change (e.g. focus on widgets, launch menus)

4.5.1 Prototype 3 – Planned Tasks

Regarding the next steps in the 2nd Screen component, Figure 47 and the following subsections summarise the tasks planned for the third and final prototype (to be delivered in M31).

Subcomponent	Task
2nd Screen Launcher	Implementation of 1st Screen notifications visible on the 2nd Screen Android application
2nd Screen Launcher	Provide implementation of “Register” and “Password Recovery” dialogs
Voice Dialogue	• Extension of VD in order to work with the rest of the widgets. • Extension of the New Integration Model (support for additional data exchange) • Increase robustness (e.g. decrease rate of misinterpreted actions, solve connection stability problems). • Investigate and implement ways for more natural interaction • Asynchronous execution of actions • Investigate and implement ways for multi-modal interaction (graphical feedback for voice actions)

Figure 47: Tasks Planned for the Third Prototype of the 2nd Screen

4.6 Research Background

Due to the nature of the component, no research background is available except of the Voice Dialogue subcomponent, where papers taken into consideration are provided in Figure 48. These papers point to research that led and influenced the creation of Talkamatic Dialogue Manager (TDM) upon which the Voice Dialogue component is built.

The integration of VD with the SAM application is done in a way that highlights TDM’s capabilities and thus exhibits the concepts, which exist in TDM.

Source	Subcomponent	Description
[LAR02] Larsson, S., (2002). Issue-based Dialogue Management, Gothenburg Monographs in Linguistic	Voice Dialogue	Dissertation thesis describing dialogue design principles on which TDM was built
[BCL05] B. Bringert, R. Cooper, P. Ljunglöf, A. Ranta, Multimodal Dialogue System Grammars. Proceedings of DIALOR'05, Ninth Workshop on the Semantics and Pragmatics of Dialogue, Nancy, France, June 9-11, 2005, 2005	Voice Dialogue	Shows how mouse clicks can be integrated in GF grammars alongside with speech input
[BRI07] B. Bringert. Speech Recognition Grammar Compilation in Grammatical Framework SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing, June 29, 2007, Prague. 2007.	Voice Dialogue	Generation of speech recognition language models from GF in several formats: GSL (Nuance), SRGS, JSGF, and HTK SLF, with embedded semantic interpretation.

Figure 48: Research Background 2nd Screen

4.7 Target Performance

This section contains the key performance indicators (KPI) (see Section 4.7.1) and user experience measurement tasks (see Section 4.7.2) for this component.

4.7.1 Component KPIs

For this component the following KPIs have been defined:

Topic	Description	Target KPI
Discovery duration	The 2nd Screen application will enable the detection of 1st Screen devices in the local network. A pairing with available 1st Screen devices can then done afterwards to receive additional content regarding to the video element running on the 1st Screen device.	The discovery of all available 1st Screen devices should be achieved in less than 10 seconds.
Connection success rate	When a 1st Screen discovers a 2nd Screen (or vice versa), setting up a connection should always be successful.	The discovery of all available 1st Screen devices should be achieved in 90% of the test cases.
Interpretation error rate (IER)	IER is defined as the difference between correct and actual interpretations for a given corpus on the semantic level. This metric covers the whole chain from speech recognition, via contextual processing, to semantic interpretation. In the best of worlds, IER approaches 0%. However, such a target would not be realistic. Even humans mishear and misunderstand each other, and currently systems even more so. Optionally, one could settle for a higher accepted value of IER, e.g. 30%. However, setting an acceptance value for IER is problematic, since IER depends on the performance of the automatic speech recognizer. Voice Dialogue is expected to use a speech recognizer which performance as such cannot be easily isolated or controlled. Instead, it is more relevant to measure the relative decrease of IER compared to a baseline value. For example, if initial results form a baseline IER of 0.50 (meaning that half of the interpretations were correct), and if contextual interpretation and other improvements decrease IER to 0.45, then the relative IER decrease is 10%.	What would be a reasonable target for relative IER decrease? The planned improvements are highly exploratory in nature, and predictions of their success are difficult to make. With these uncertainties in mind, a target value of 20% decrease of interpretation errors is set as goal. It should be emphasized, that this goal is quite ambitious given the exploratory and experimental nature of the planned improvements.

Figure 49: Target Performance 2nd Screen

These performance targets will be reviewed against the performance achieved in the third prototype. The following results have been achieved at the end of the second iteration:

- **Discovery duration:** The average discovery time has been 278,8 ms
- **Connection success rate:** The discovery achievement rate is 100%
- **Interpretation error rate:** Results for this KPI will be gathered in task T8.3 “Use Case: Social Consumption”

4.7.2 User Experience Measurements Tasks

Additional to the KPIs in Section 4.7.1, this work package provides user tasks, which are input for measuring the subjective user experience in a uniform way. For each of the tasks below the task-specific KPIs, defined in

References

[WDJ11] Wilson, S., Daniel, F., Jugel, U., Soi, S. (2011). Orchestrated User Interface Mashups Using W3C Widgets. Proceedings of Composable Web 2011.

[BWS13] Balasubramanee, V. Wimalasena, C. Singh, R. Pierce, M. (2013) Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development

[KAA07] Christian Kaar (2007), An introduction to Widgets with particular emphasis on Mobile Widgets

[LAR02] Larsson, S., (2002). Issue-based Dialogue Management, Gothenburg Monographs in Linguistic

[BCL05] B. Bringert, R. Cooper, P. Ljunglöf, A. Ranta, Multimodal Dialogue System Grammars. Proceedings of DIALOR'05, Ninth Workshop on the Semantics and Pragmatics of Dialogue, Nancy, France, June 9-11, 2005, 2005

[BRI07] B. Bringert. Speech Recognition Grammar Compilation in Grammatical Framework SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing, June 29, 2007, Prague. 2007.

[ZIE13] Ziegler, C., "Second screen for HbbTV — Automatic application launch and app-to-app communication enabling novel TV programme related second-screen scenarios," Consumer Electronics Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on, vol., no., pp.1, 5, 9-11 Sept. 2013

[VAN12] Vanattenhoven, J., Geerts, D. (2012). Second-Screen Use in the Home: An Ethnographic Study. In Proceedings 3rd International Workshop on Future Television, EuroITV 2012 (p. 12).

[VAN14] Vanattenhoven, J., Geerts, D., De Grooff, D. (2014). Television Experience Insights from HbbTV. In proceedings of the International Workshop on Interactive Content Consumption, Newcastle upon Tyne

[BAS13] Bassbouss, L.; Tritschler, M.; Steglich, S.; Tanaka, K.; Miyazaki, Y., "Towards a Multi-screen Application Model for the Web," Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual , vol., no., pp.528,533, 22-26 July 2013

(2015-10) ETSI TS 102 796 V1.3.1 / HbbTV 2.0

[WCJ14] Lee, Wei-Po, Che Kaoli, and Jhih-Yuan Huang. "A smart TV system with body-gesture control, tag-based rating and context-aware recommendation." Knowledge-Based Systems 56 (2014): 167-178.

[MAY14] Maynard, G. HbbTV and Multi-Screen Strategy. HbbTV Symposium Asia. Singapore 18 June 2014.

Annex A: User Research, will be measured.

Task	Description
Pairing	The user has to successfully pair the 2nd Screen device with the 1st Screen device.
Control Video Element	The user has to successfully pause, and then shift to a specific time in the video element and then start the video element again.
Enable GPS	The user has to successfully enable the GPS location feature in the settings menu.
Enable Notifications	The user has to successfully enable the notifications feature in the settings menu.
Login	The user has to successfully reset the password by using the recovery feature and login to the SAM application.
Post on Twitter using Voice Dialogue	The user has to successfully post a Tweet on Twitter using the Voice Dialogue feature
Control Video Element using Voice Dialogue	The user has to successfully pause, and then start the video element again using the Voice Dialogue feature.

Figure 50: Target Performance 2nd Screen – Task-Specific Measurements

4.8 Summary

This section provides a description of the second prototype of the 2nd Screen component developed in task T7.2 2nd Screen Media Interaction. The main outcome of this task is the 2nd Screen Android application and the integrated Voice Dialogue subcomponent including the connected backend system. This prototype is the second of three iterations planned for this component and the goal is to cover 60% of the requirements of the component (see Section B 1.3.3.7 of the DoW for additional information on the effort distribution for this component in the lifespan of the project).

The most important goals reached during this second prototype have been:

- Integration of the Identity and Security Services component by implementing OpenID usage and a login mask
- Support of the Android “Network Service Discovery” (NSD) feature to provide easy discovery of and connection to local 1st Screen devices
- Updated Remote control user interface
- Implementation of the VD’s New Integration Model and adding voice interaction onto the Twitter widget

The requirements necessary for both users and developers have been presented including installation and installation instructions. The last section has been dedicated to describing the limitations of the current prototype and describing the next steps considered for the third version of the component, which will be delivered in M31.

5 SAM 1st Screen Media Interaction

This section describes the software deliverable D.7.3.2, which is the second prototype release of the SAM 1st Screen Media Interaction functionalities.

5.1 Scope and Relationship

The 1st Screen component embeds the Generic Dashboard and enables video streaming for the End User view. The End User can in some cases also consume related content displayed inside widgets on the 1st Screen.

Figure 51 shows the different subcomponents of the 1st Screen component, the logical connections that have been established between them and the relations with other components and actors in the SAM platform.

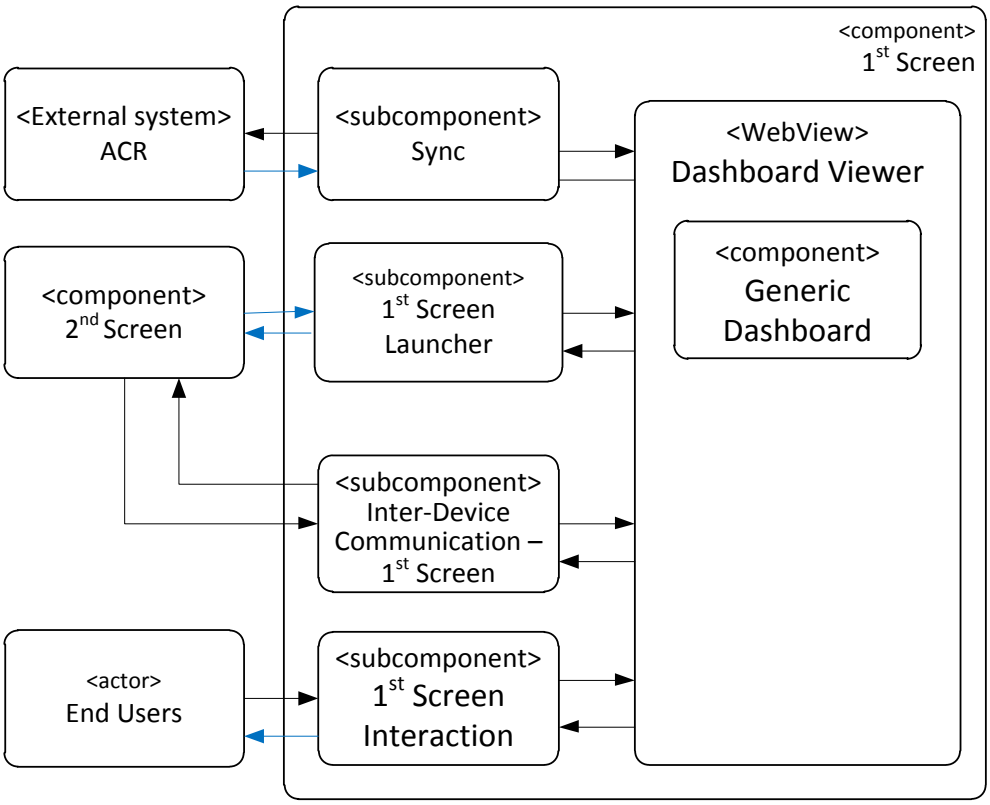


Figure 51: 1st Screen Subcomponents and Its Relationships

For further descriptions of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.13.3 (Architecture), D3.2.2 Section 4.14.2 (Functional Specification) or D3.3.1 Section 3.14.2 (Technical Specification).

The first prototype of T7.3 provided the basic integration of the Generic Dashboard component using the Dashboard Viewer subcomponent, which provided the display. Basic remote control functions had also been implemented using 1st Screen interaction component, which allowed the End User of the 1st Screen application to navigate through the screens, select a video stream, start/pause/stop the video and toggle full screen mode.

A summary of the tasks carried out for each subcomponent of the first version of the prototype is shown in the following table:

Subcomponent	Task
1st Screen Interaction	Remote control support.
Inter-Device-Communication 1st Screen	Simple communication between 1st and 2nd Screen devices using centralised Web Socket server.
Dashboard Viewer	Display of the Generic Dashboard component. Provide 1st Screen device specifications.
Sync	No tasks planned for the first prototype.

Figure 52: Tasks Carried Out for the First Prototype of T7.3

The second prototype of T7.3 provides the further integration of the Generic Dashboard component using the Dashboard Viewer subcomponent, which provides the display. The embedded Web Socket server as a part of 1st Screen Inter-Device Communication was also implemented, which enables more efficient local communication between 1st and 2nd Screen devices. Furthermore, to add the possibility of automatic 1st Screen discovery, an Android NSD protocol implementation was put in place.

A summary of the tasks carried out by each subcomponent of the second version of the prototype is shown in the following table:

Subcomponent	Task
1st Screen Interaction	No tasks planned for the second prototype
Inter-Device-Communication 1st Screen	• Implementation of local Web Socket Server in the 1st Screen allowing automatic synchronization between 1st Screen and 2nd Screens over a local network. • Advertisement of the 1st Screen application using Android NSD protocol
Dashboard Viewer	• Implementation of local HTTP Web Server in the 1st Screen allowing for local storage and access to Generic Dashboard
Sync	No tasks planned for the second prototype

Figure 53: Tasks Carried Out for the Second Prototype of T7.3

5.1.1 1st and 2nd Screen interaction

To facilitate discovery and interaction between 1st Screen and 2nd Screen, some standard protocols needs to be implemented. For SAM it is been decided to use Network Service Discovery (NSD) for discovery mechanism between 1st Screen and 2nd Screens (See section 5.1.1.1). Also a design decision was made to implement Web Socket Server on 1st Screen to host Generic Dashboard application (See section 5.1.1.2). This decision was made to improve the performance of the system by host the application in local network rather than running on remote network (See section 5.1.1.3).

5.1.1.1 Network Service Discovery (NSD)

NSD[7] is a simple protocol, which added to the app allows the users of it to discover other devices on the local network that support the services the app requests. This is useful for a variety of peer-to-peer applications such as communication between 2nd Screen and 1st Screen for exchange of information. Android's NSD API simplifies the effort required to implement such features.

[7] http://developer.android.com/training/connect-devices-wirelessly/nsd.html

5.1.1.2 Web Socket Server

A design decision was made to move the Web Socket Server implementation from Generic Dashboard server to 1st Screen application on the TV. Web Socket server on the TV is triggered from inside the 1st Screen application when the application is started.

The Web Socket server and client implementations are written in Java 1.7 version. The underlying classes are implemented using *java.nio*, which allows for a non-blocking event-driven model (similar to the Web Socket API for web browsers).

Implemented Web Socket protocol versions are:

- RFC 6455[8]
- Hybi 17[9]
- Hybi 10[10]
- Hixie 76[11]
- Hixie 75[12]

The *org.java_websocket.server.WebSocketServer* abstract class implements the server-side of the Web Socket Protocol[13]. A Web Socket server by itself does not do anything except establish socket connections though HTTP. After that it's up to the implementing subclass to add business logic.

The *org.java_websocket.client.WebSocketClient* abstract class can connect to valid Web Socket servers. The constructor expects a valid *ws://* URI to connect to. Important events *onOpen, onClose, onMessage* and *onIOError* get fired throughout the life of the Web Socket Client, and must be implemented in the relevant subclass.

5.1.1.3 Web Server and Hosting Generic Dashboard in 1st Screen

A design decision was made to move Generic Dashboard hosting from the remote server to the 1st Screen application on TV.

To host the files and related resources, a webserver is needed to be in place. The webserver needs to take care of following functionalities:

- Host source code files and related resource files, images, icons etc.
- Process clients' requests made by 2nd Screen applications and feed them processed data accordingly

There are different variants of webserver based of the type of protocol used for implementation. As the Generic Dashboard was previously hosted on a HTTP based server, it was decided to create the same variant of web server using HTTP protocol in the 1st screen application. With this in place, all the components which need to interact with the webserver continue to work in the same way without any modification required on their side.

[8] http://tools.ietf.org/html/rfc6455
[9] http://tools.ietf.org/id/draft-ietf-hybi-thewebsocketprotocol-17.txt
[10] http://tools.ietf.org/id/draft-ietf-hybi-thewebsocketprotocol-10.txt
[11] http://tools.ietf.org/id/draft-hixie-thewebsocketprotocol-76.txt
[12] http://tools.ietf.org/id/draft-hixie-thewebsocketprotocol-75.txt
[13] http://www.whatwg.org/specs/web-socket-protocol/

5.2 Requirements and Preparations

This section provides information on technical and non-technical requirements for users as well as for developers.

5.2.1 For Users

The user needs to provide an Android Smart TV device running Android version 5.1 and supporting SmartTV specification version 3.0[14]. The Android application is not available on Google Play[15]. Due to the fact that Philips 2k15 TV's do not allow apps coming from other sources than the Google Play Store or the Philips SmartTV Portal, the *.apk file can only be installed on a Philips Development TV model. On these models the app can be installed through the usage of android debug bridge (where x.x.x.x is the IP address of the TV):

```
adb connect x.x.x.x:5555
adb install 1st-screen.apk
```

5.2.2 For Developers

For developers it is strongly recommended to use the Android Studio IDE[16] for the implementation. Going forward, Google has deprecated the support for Eclipse for Android app development which was used before and is pushing all to migrate to Android Studio IDE. Also, Android Studio IDE has much better support for Android application development.

On the other hand, Java-WebSocket is known to work with:

- Java 1.7 (SE 7)
- Android 1.6 (API 4)

Some other Java versions may also be used for this implementation but have not been tested during this implementation.

To create a typical webserver, following steps are required:

1. Initially a server socket needs to be created which should listen to the desired port. To create a server socket the *java.net.ServerSocket* class is used. The constructor of this class accepts a port number to listen for the incoming connections.
2. Once the *ServerSocket* object is created, it is needed to accept the incoming connection using *ServerSocket.accept()* method. This method is blocking, so a separate thread is created to accept and process the incoming connections. The *accept()* method returns a *java.net.Socket* object which represents the accepted connection.
3. Once the connection is established, the next HTTP request needs to be processed. This can be done using *org.apache.http.protocol.HttpService* class. This class provides a server side implementation which can handle minimal HTTP processing requests. In the SAM implementation, to handle different HTTP requests, handler map design is used which is based on URI patterns.

[14] https://developers.smarttv-alliance.org/specification
[15] http://play.google.com/
[16] http://developer.android.com/tools/studio/index.html

An implementation of a local webserver to serve Generic Dashboard directly from the local network was decided to be part of prototype 3 but has been implemented in prototype 2. This decision was made to improve the performance of the system and to make a better demo able product in places with weak Internet connections.

5.3 Installation and Deployment

This section describes the Installation and deployment process to use the app and source code for Users and Developers:

5.3.1 For Users

After downloading the 1st Screen application and moving it to the device's internal or external storage, the user has to execute the *.apk file. As the first step, the user will be prompted for the acceptance of the access rights. After the installation, the 1st Screen application can be started using the icon in the app overview.

5.3.2 For Developers

Developers have to download the source code first (Deliverable D7.3.2 contains the required information). The source code of the 1st Screen application is an Android Studio project and can be opened with Android Studio after downloading.

5.4 Execution and Usage

This section describes how to use the different subcomponents of the prototype.

5.4.1 For Users

The following subsections present the different views of the 1st Screen application.

5.4.1.1 Generic dashboard viewer

When the application is started, it will launch a SmartTV capable browser to view the Generic Dashboard. Depending on the network bandwidth this can take a few seconds during the first start-up of the application. The home screen of the Generic Dashboard is shown in Figure 54.

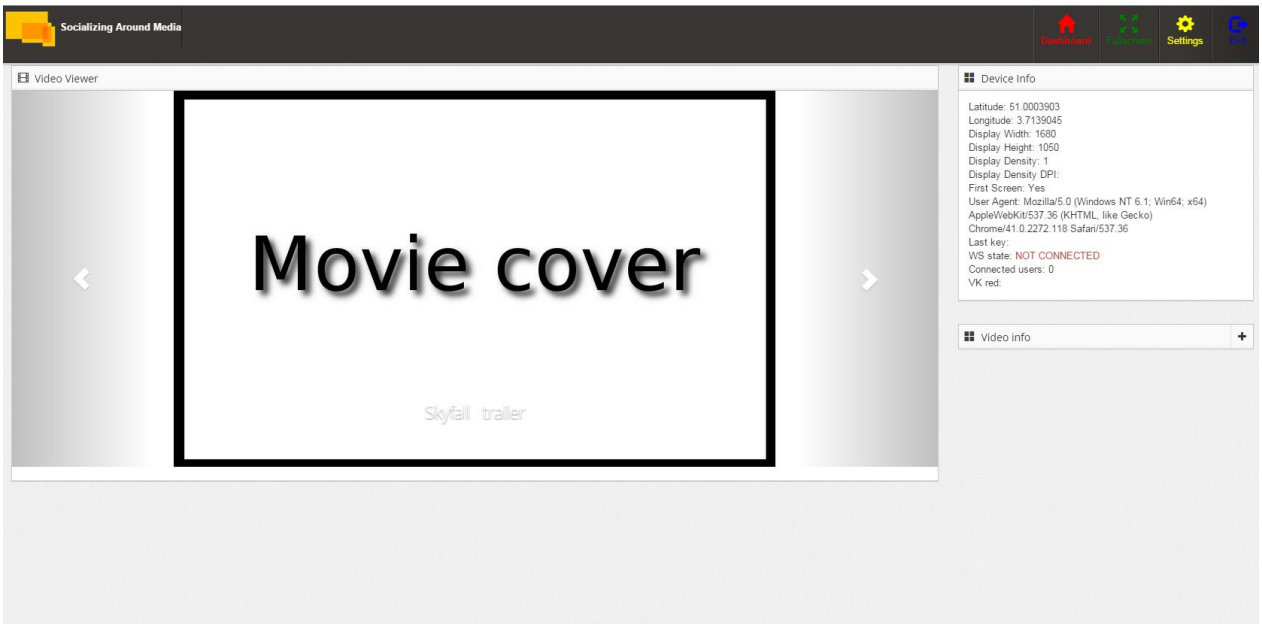


Figure 54: 1st Screen Application Generic Dashboard View.

The home screen consists of 4 navigation buttons and 3 Widgets. The End User can employ her/his remote control to navigate between the buttons and Widgets using remote's navigation buttons (up / down/ left / right). When the End User navigates the Generic Dashboard, a blue cursor frame is visible to indicate currently focused element. Pressing OK will execute requested action. Colour keys on the remote can also be used to quickly select access the functionality behind the four navigation buttons:

Remote button	On screen button	Function
	Dashboard	View the home screen of Generic Dashboard
	Fullscreen	Toggle full screen video player
	Settings	View SAM settings page
	Exit	Exit the 1st screen application

The Widgets presented on the Generic Dashboard are: Video Viewer, Device Info and Video Info. The Device Information Widget displays, amongst other information, the display size and density of the device and its GPS position (if known). Video Viewer displays currently available video streams for End User to browse and view. Once a selection has been made, the video is buffered and then played. The End User can use the play/pause/stop buttons on the remote to control the player. Video Info Widget contains the stream name and current position of streamed video.

5.4.1.2 SAM settings page

After selecting the Settings button (or pressing yellow key on the remote), the End User is presented with SAM settings page (see Figure 55). On this page, the user can toggle the state of various SAM-related settings (i.e. advertising, statistics, geo-location).

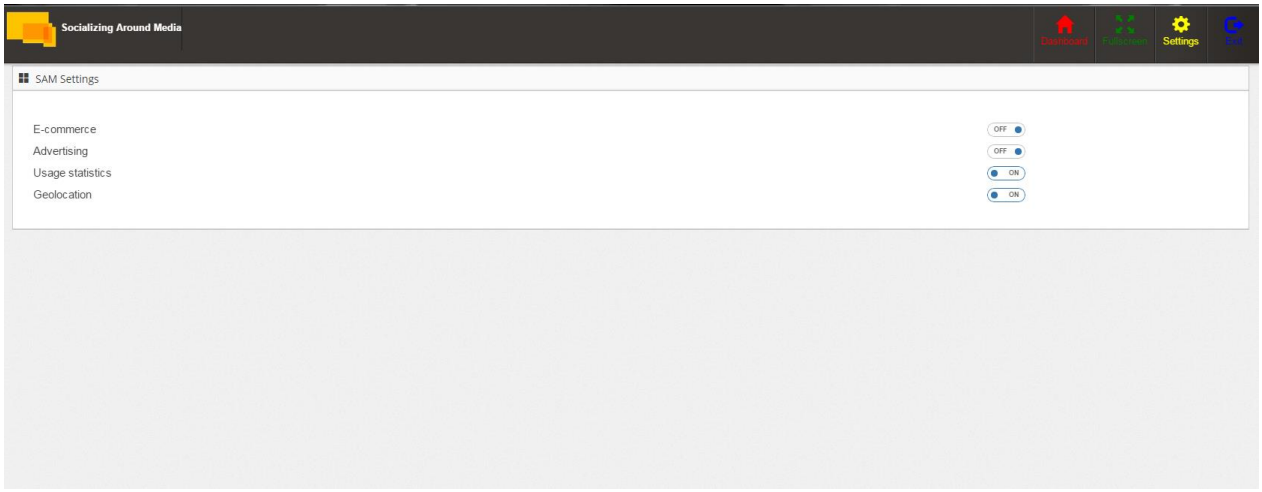


Figure 55: SAM Settings Page of 1st Screen Application

Due to the Smart TV application nature, the rest of 1st Screen components have been included in Generic Dashboard source code. Please, for further details refer to the Generic Dashboard Section 6.4.2.

5.4.2 For Developers

Once the source code is downloaded and opened using Android Studio, the application can be compiled and executed using Android Studio only. Android Studio has built-in tools to support compilation and installation of the app on to the device. Also, Android Studio has built-in tools which support debugging the application.

5.5 Limitations and Further Developments

Development of the second prototype has been concentrating on the further development of the 1st Screen Android application and the integration and communication with the Generic Dashboard component. Due to that, the following views/functions have been provided as mock-ups for later development:

- **1st Screen Launcher:** Currently the second prototype does not include any remote functionality for 1st Screen Launcher, thus it is needed to manually start 1st Screen app before connecting any of the 2nd Screen devices.

5.5.1 Prototype 3 – Planned Tasks

Regarding the next steps in the 1st Screen component, Figure 56 summarises the tasks planned for the third prototype (to be delivered in M31).

Subcomponent	Task
1st Screen Launcher	Implementation of 1st Screen launch functionality based on HbbTV or a DIAL-like approach

Figure 56: Tasks Planned for the Third Prototype of the 1st Screen

5.6 Research Background

For the current prototype implementation and the overall approach, the following related research work was taken into consideration.

Source	Subcomponent	Description
[ZIE13] Ziegler, C., "Second screen for HbbTV — Automatic application launch and app-to-app communication enabling novel TV programme related second-screen scenarios," Consumer Electronics Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on , vol., no., pp.1,5, 9-11 Sept. 2013	Inter-Device-Communication 1st Screen	This paper has been considered to implement communication protocols according to the HbbTV standard
[VAN12] Vanattenhoven, J., Geerts, D. (2012). Second-Screen Use in the Home: An Ethnographic Study. In Proceedings 3rd International Workshop on Future Television, EuroITV 2012 (p. 12).	Inter-Device-Communication 1st Screen	Input on 2nd screen users as background information on profiling.
[VAN14]] Vanattenhoven, J., Geerts, D., De Grooff, D. (2014). Television Experience Insights from HbbTV. In proceedings of the International Workshop on Interactive Content Consumption, Newcastle upon Tyne	Inter-Device-Communication 1st Screen	Highlights results from EU project Hbb-NEXT, in particular applied to recommenders. See http://www.hbb-next.eu/index.php/documents

5.7 Target Performance

The performance measurement of the 1st Screen prototype will be measured according to the defined KPIs (described in Section 5.7.1) with additional End User experience measurements (described in Section 5.7.2).

5.7.1 Component KPIs

For this component the following Key Performance Indicators (KPIs) have been defined:

Topic	Description	Target KPI	Current KPI
Responsiveness	When a user is interacting with his 1st Screen device, it needs to be very responsive.	Initial loading time of the 1st screen app should be less than 10 sec. Navigation actions should execute in less than 3 sec.	Current loading time of 1st screen app is around 5-8 secs. Navigation actions execute in less than 2 secs.
Connection success rate	When a 1st Screen discovers a 2nd Screen (or vice versa), setting up a connection should always be successful.	The discovery of all available 2nd Screen devices should be achieved in 90% of the test cases.	The discovery succeeds in 90% test cases.
Connection setup time	The time it takes to setup a connection between a 1st Screen and a 2nd Screen device needs to be low.	The discovery of all available 2nd Screen devices should be achieved in less than 10 seconds.	The discovery time is around 5-7 secs.

Figure 57: Target Performance 1st Screen

5.7.2 User Experience Measurements Tasks

Additional to the KPIs in Section 4.7.1, this work package provides user tasks, which are input for measuring the subjective user experience in a uniform way. For each of the tasks below the task-specific KPIs, defined in

References

[WDJ11] Wilson, S., Daniel, F., Jugel, U., Soi, S. (2011). Orchestrated User Interface Mashups Using W3C Widgets. Proceedings of Composable Web 2011.

[BWS13] Balasubramanee, V. Wimalasena, C. Singh, R. Pierce, M. (2013) Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development

[KAA07] Christian Kaar (2007), An introduction to Widgets with particular emphasis on Mobile Widgets

[LAR02] Larsson, S., (2002). Issue-based Dialogue Management, Gothenburg Monographs in Linguistic

[BCL05] B. Bringert, R. Cooper, P. Ljunglöf, A. Ranta, Multimodal Dialogue System Grammars. Proceedings of DIALOR'05, Ninth Workshop on the Semantics and Pragmatics of Dialogue, Nancy, France, June 9-11, 2005, 2005

[BRI07] B. Bringert. Speech Recognition Grammar Compilation in Grammatical Framework SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing, June 29, 2007, Prague. 2007.

[ZIE13] Ziegler, C., "Second screen for HbbTV — Automatic application launch and app-to-app communication enabling novel TV programme related second-screen scenarios," Consumer Electronics Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on, vol., no., pp.1, 5, 9-11 Sept. 2013

[VAN12] Vanattenhoven, J., Geerts, D. (2012). Second-Screen Use in the Home: An Ethnographic Study. In Proceedings 3rd International Workshop on Future Television, EuroITV 2012 (p. 12).

[VAN14] Vanattenhoven, J., Geerts, D., De Grooff, D. (2014). Television Experience Insights from HbbTV. In proceedings of the International Workshop on Interactive Content Consumption, Newcastle upon Tyne

[BAS13] Bassbouss, L.; Tritschler, M.; Steglich, S.; Tanaka, K.; Miyazaki, Y., "Towards a Multi-screen Application Model for the Web," Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual , vol., no., pp.528,533, 22-26 July 2013

(2015-10) ETSI TS 102 796 V1.3.1 / HbbTV 2.0

[WCJ14] Lee, Wei-Po, Che Kaoli, and Jhih-Yuan Huang. "A smart TV system with body-gesture control, tag-based rating and context-aware recommendation." Knowledge-Based Systems 56 (2014): 167-178.

[MAY14] Maynard, G. HbbTV and Multi-Screen Strategy. HbbTV Symposium Asia. Singapore 18 June 2014.

Annex A: User Research, will be measured.

Task	Description
Pairing	The user has to pair the 1st Screen device with the 2nd Screen device.
Controlling the Video Element	The user has to stop, to shift to a specific time in the video element and start the video element again.
Making video full screen	The user has to make the video full screen and later get back to the non-full screen mode again.

Figure 58: Target Performance 1st Screen – Task-Specific Measurements

5.8 Summary

This section provides a description of the second prototype of the 1st Screen component developed in task T7.3 1st Screen Media Interaction. The main outcome of this task is the software of the 1st Screen component. This prototype is the second of three iterations planned for this component and the goal of this prototype is to cover 30% of the requirements of the component (see Section B 1.3.3.7 of the DoW for additional information on the effort distribution for this component in the lifespan of the project).

The most important goals reached during this second prototype have been:

- Further extension of 1st Screen Android application including an internal WebSocket server for inter device communication over local network
- Implementation of Android Network Service Discovery protocol to enable automatic discovery of the 1st Screen by the 2nd Screen devices

The requirements necessary for both users and developers have been presented including installation instructions. The last section has been dedicated to describe the limitations of the current prototype, also describing the next steps considered for the third version of the component, which should be delivered in M31.

6 SAM Multi-Device Dashboard

This section describes the software deliverable D7.4.2, which is the second prototype release of the SAM Multi-Device Dashboard component.

6.1 Scope and Relationship

The Generic Dashboard provides the End User with a Graphical User Interface for the interaction with the SAM platform. The Generic Dashboard is presented on both 1st Screen and 2nd Screen devices, on top of which each device adds its specific subcomponents. Figure 59 shows the different subcomponents of the Generic Dashboard, the logical connections that have been established between them and the relations with other components and actors of SAM platform.

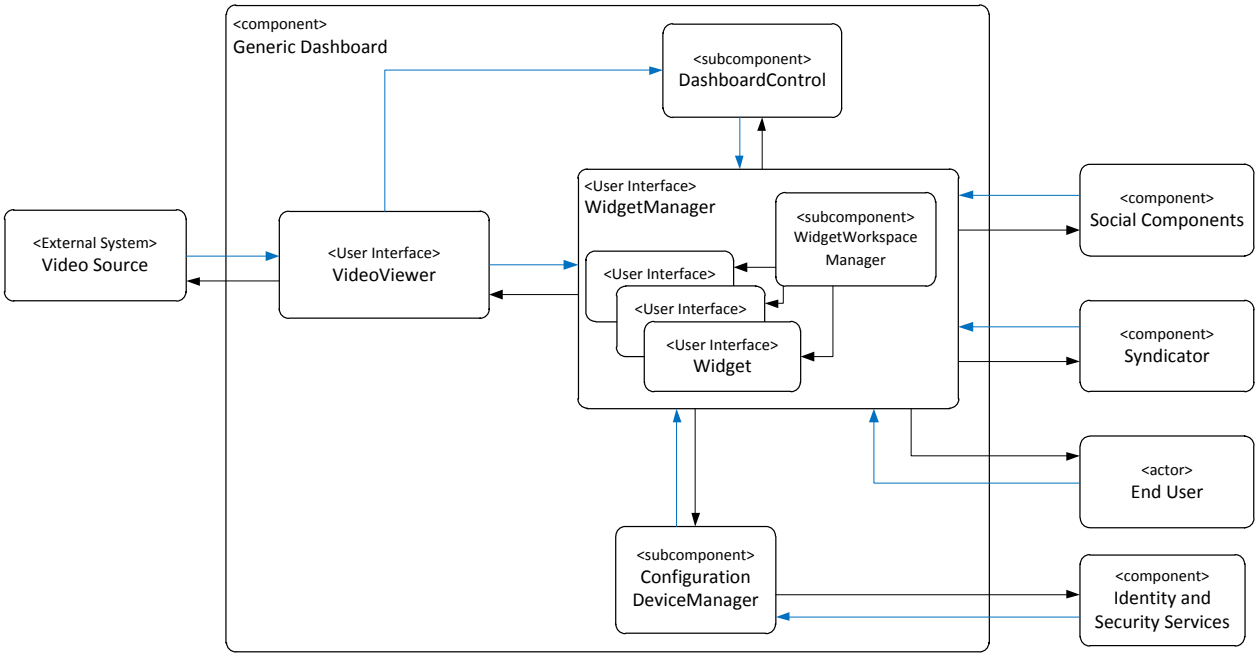


Figure 59 Generic Dashboard Subcomponents and its Relationships

For further description of the functional and technical foundations of these subcomponents, please revisit documents D3.2.1 Section 4.13.2 (Architecture), D3.2.2 Section 4.14.1 (Functional Specification) or D3.3.1 Section 3.14.1 (Technical Specification).

First prototype of the Generic Dashboard focused on providing basic functionalities to the End User, such as listing the available video streams and playback of the selected one. The Dashboard itself had been populated with several Widgets presenting basic functionalities and sample static content.

A summary of the tasks carried out for each subcomponent of the first version of the prototype is shown in the following table:

Subcomponent	Task
Video Viewer	Obtain a list of available streams from video source, select and stream video
Dashboard Control	Provide JavaScript interfaces for communication
Widget	Display and control user interface widgets on the Dashboard
Widget Manager	Provide JavaScript interfaces for updating widgets content
Widget Workspace Manager	Display and control support of the Widgets. Display SAM settings
Configuration Device Manager	Provide JavaScript interfaces for device information

Figure 60: Tasks Carried Out for the First Prototype of T7.4.

In the second prototype, the Generic Dashboard was extended with a link to the Syndicator component, which allows displaying dynamic Widgets. The number of Widget types was extended with Facebook, Twitter, Related Products and Wikipedia page Widgets allowing providing the End User with a more immersive and rich experience. Adding the interfaces to authenticate the user based allowed for gathering contextual information about user interactions with the Dashboard and consumed assets.

A summary of the tasks carried out for each subcomponent of the second version of the prototype is shown in the following table:

Subcomponent	Task
Video Viewer	No tasks planned for this prototype
Dashboard Control	Improved messaging mechanism between widgets, allowing for more reliable communication between 1st and 2nd Screen
Widget	Implemented particular widgets for various functionalities, in particular: • Widgets showing relevant social information from Facebook and Twitter related to presented video, allowing to post new posts / tweets • Related Product Widget offering information about possible related marketing product End User can buy • Wikipedia Widget displaying to the End User information from Wikipedia on related topic to currently played video • Related Assets widget displaying past Widgets that are moved away from the central point of the Dashboard, allowing the End User to browse its contents
Widget Manager	• Integration with Syndicator to receive and update Widgets with related content to currently viewed video • Integration with Social Components to feed Twitter and Facebook Widgets. This subcomponent is in charge of implementing the communication with the Data API services component. • Upgrades in Generic Dashboard services responsible for providing the list of assets related to the main video, obtaining the Widgets html and providing the list of videos to show in the 1st Screen. • Implementation of caching mechanism in order to improve the performance of the communication between Generic Dashboard and other SAM components.
Widget Workspace Manager	• Updating the Widgets accordingly to scheduled timeline obtained from Syndicator component • Advanced Widget handling responsible for prefetching of Widget HTML code & moving it afterwards to Related Assets Widget
Configuration Device Manager	Obtaining user token from 2nd Screen devices and making it available to all widgets that need this information to track End User preferences and/or activities on the Generic Dashboard on the 1st and 2nd Screen.
Context Listener	• 1st implementation of an AngularJS client for communicating with the Context Control services. Through this client, contextual information (e.g. location and consumed assets) about the users who interact with the dashboard are pushed to Context Control component. • In addition, information on the users' profiles and their behaviour in the SAM environment is available (e.g. consumed assets), which for the proof of concept of the current prototype, provides the Top10 consumed assets.

Figure 61: Tasks Carried Out for the Second Prototype of T7.4.

6.2 Requirements and Preparations

This section provides information on technical and non-technical requirements for users as well as for developers. The Dashboard has dependencies with the Data API services (Syndicator). For this reason it is necessary to check that the services of the Syndicator component are working properly. The necessary documentation to check the status of these services is D5.9.2 section 6.2.

6.2.1 For Users

The Generic Dashboard is an embedded subcomponent in the 1st and 2nd Screen applications. For specific requirements for running 1st Screen application see Section 5.2.1, for 2nd Screen application see Section 4.

6.2.2 For Developers

For developers it is strongly recommended to use Apache HTTP server[17] for hosting the web application. Additionally, using the devices listed in sections 4.2.2 and 5.2.2 is advised, but not a necessity.

6.3 Installation (Deployment)

This section describes the Installation process of Generic Dashboard for Users and Developers:

6.3.1 For Users

It is not necessary to separately install the Generic Dashboard, because it is embedded into 1st and 2nd Screen applications. The installation process of those applications is described in sections 4.3.1 and 5.3.1 respectively.

6.3.2 For Developers

Developers have to download the source code first (Deliverable D7.4.2 contains the required information). The downloaded source code must be placed in the web root of the Apache HTTP server. The source code of the Generic Dashboard application contains HTML and JavaScript files and can be edited using any text editor.

6.4 Execution and Usage

This section describes how to use the different subcomponents of the prototype.

6.4.1 For Users

The usage of Generic Dashboard depends on the type of the device. Please consult the corresponding sections describing usage of the 1st Screen (Section 0) and the 2nd Screen (Section 4) devices.

6.4.2 For Developers

The Generic Dashboard subcomponents have been implemented as AngularJS services[18] and controllers[19]. Widgets can use these services for messaging, receiving content and social components communication. The communication between Widgets and Syndicator or Social Components is handled by the Widget Manager service. The communication between Widgets, present either on the same Generic Dashboard or across 1st and 2nd Screen devices, is handled by the Dashboard Control service. The following JavaScript methods can be used from Widgets to specify the type of message they would like to

[17] http://httpd.apache.org/
[18] AngularJS Services: https://docs.angularjs.org/guide/services
[19] AngularJS Controllers: https://docs.angularjs.org/guide/controller

receive and define the call back function to receive and process them. The same holds for content update messages. Each Widget can then use the “sendMessage” function to send the message of appropriate type to all subscribed Widgets, irrespectively which Screen they are placed on.

6.4.2.1 Dashboard Control service

Subscribe For Message	
Description	A widget can subscribe to receive messages of a certain type
Method Header	dashboardControl.subscribeForMessage(widget, messageType)
Parameters	widget – subscribing widget object reference messageType – type of the message the widget is subscribing for
Return Value	None
Error Handling	None
Remarks	When a message of the messageType is received, the widget controller method receiveMessage is called.

Figure 62: Dashboard Control Service Interface – Subscribe For Message

Send Message	
Description	A widget can send message to other widgets (also other widgets present on other devices)
Method Header	dashboardControl.sendMessage(messageType, message)
Parameters	messageType – type of the message message – message to be send
Return Value	None
Error Handling	None
Remarks	None

Figure 63: Dashboard Control Service Interface – Send Message

6.4.2.2 Widget Manager Service

Subscribe for Content	
Description	A widget can subscribe to receive content of a certain type
Method Header	widgetManager.subscribeForContent(widget, contentType, frequency)
Parameters	widget – subscribing widget reference contentType – type of the content the widget is subscribing for frequency – how often the widget wants to receive the updates (in seconds); 0 = immediately
Return Value	None
Error Handling	None

Remarks	When content of the contentType is received, the widget controller method updateNotification is called

Figure 64: Widget Manager Service Interface – Subscribe For Content

6.4.2.3 Widget controller

Widget controller itself must implement following methods:

Receive Message	
Description	Widget receive message from other widgets (also other widgets present on other devices)
Method Header	widget.receiveMessage(messageType, message)
Parameters	messageType – type of the message message – message received
Return Value	None
Error Handling	None
Remarks	None

Figure 65: Widget Controller Interface – Receive Message

Receive Content	
Description	Widget receive content from Syndicator
Method Header	widget.updateNotification(contentType, content)
Parameters	contentType – type of the content content – content received
Return Value	None
Error Handling	None
Remarks	None

Figure 66: Widget Controller Interface – Receive Content

6.4.2.4 1st Screen components

Due to the nature of SmartTV applications, the Generic Dashboard also includes several subcomponents of the 1st Screen. The Dashboard Viewer of the 1st Screen invokes the SmartTV browser in order to display the Generic Dashboard component on the 1st Screen device. The SmartTV browser communicates with the Inter-Device-Communication and 1st Screen Interaction subcomponents of the 1st Screen (included in the Generic Dashboard) and adds the needed interfaces, as illustrated on Figure 66.

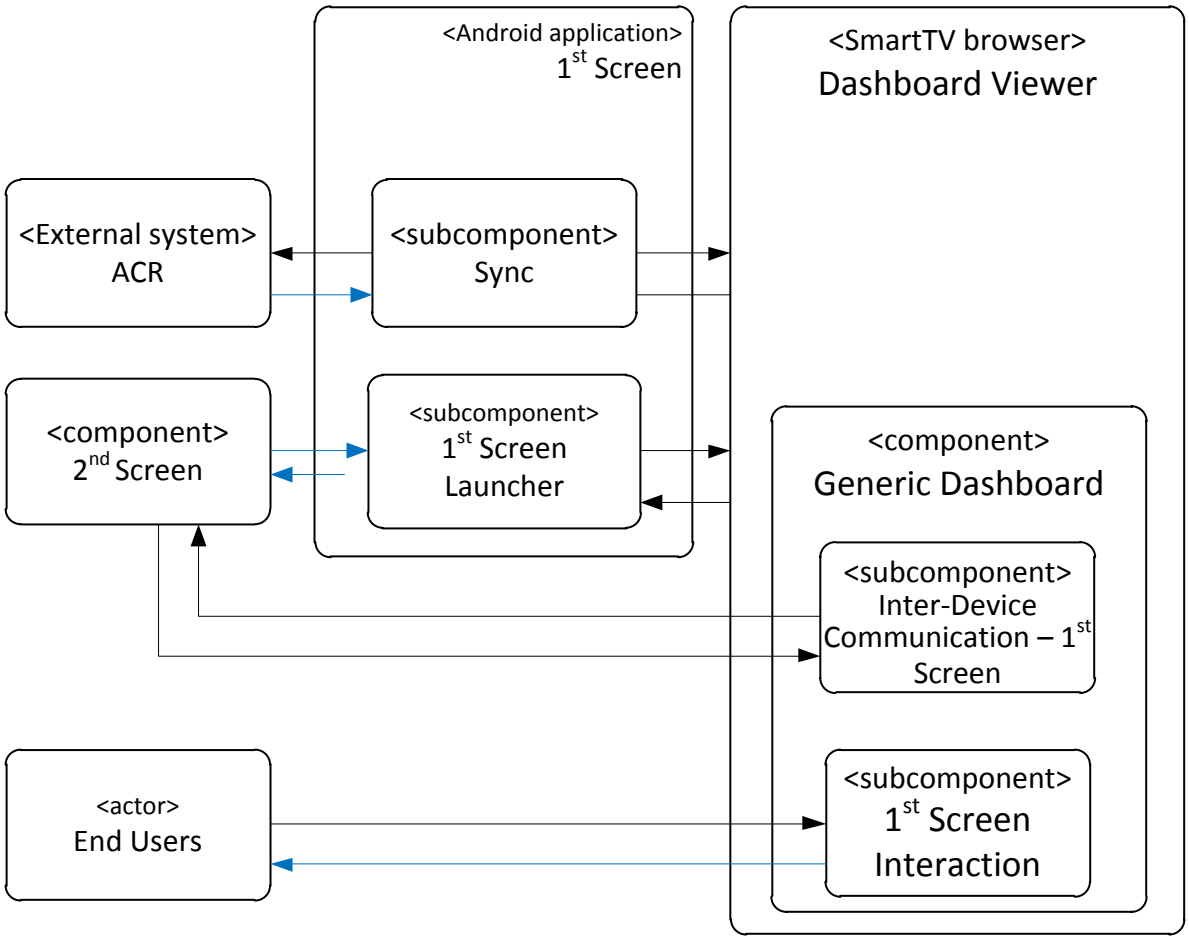


Figure 67: SmartTV application architecture

The Inter-Device-Communication is implemented as an AngularJS service. It is responsible for managing the connection to WebSocket server and exchange broadcast messages between all connected instances. Currently WebSocket is a locally embedded server and integrated into the Inter-Device-Communication component, as mentioned in Section 5.5.1

Each Widget subcomponent that is displayed on the Generic Dashboard is built from two parts: an HTML5 template and its AngularJS Controller (described above). The sample implementation of both is provided in the source code and can be found under the path in the source tree specified in Figure 68 below.

```
<source-root>/html/app/dashboard/widgets/widget.html
<source-root>/html/app/dashboard/widgets/widget.js
```

Figure 68: Sample Widget implementation source paths.

Those template files are loaded for each widget by the Widget Workspace Manager during the loading process of the Generic Dashboard. By default, the Widget controller is collapsing the Widget on the Generic Dashboard after 2 seconds from the last activity. This timeout can be set per individual Widget inside his controller.

The Video Viewer subcomponent is a specialised Widget, containing an HTML5 video element, used to display videos to the End User and to obtain various information about the video stream. Most importantly, the video ID and current timestamp are obtained via

HTML5 DOM and shared with the Widget Managers of the connected 1st and 2nd Screens using Inter-Device-Communication.

6.4.2.5 Context Control

The Generic Dashboard is integrated with the Context Control component of SAM in order to provide information regarding the users' behaviour during their interaction with SAM system, using 1st or 2nd screens. Currently, this information refers mainly on the assets that are consumed by the users and their location however, it will be further extended in order to acquire and relay to the context control system, information that is likely to provide more insight (implicit or explicit) about the user profile and their preferences such as the device type, the widgets that are open/closed, etc. In that sense, the context control will be able to provide more accurate information about the users, their profiles and their behaviour within the SAM environment. In the current prototype, the sample user related information is provided from the context control (e.g. assets consumed by users nearby) and is presented in a widget created by NTUA. For the communication with the context control and providing or consuming contextual information, the respective REST API can be used. However, for the Dashboard and the Dashboard components a JS library has been implemented, which provides access to the API through an AngularJS service.

The syndicator provides 3 services in order to allow the communication with the Generic Dashboard. The description about their Execution and Usage is detailed in D5.9.2 section 6.2.4. These services are the following:

- Get List of Videos: This service provides the list of Videos to show in the 1st Screen
- Get List of Assets Related to a particular video: This service provides the list of assets linked in the configuration of the 2nd Screen experience
- Get the HTML to show in a particular widget: This service provides the widget HTML to show in the Dashboard

6.5 Limitations and Further Developments

The second prototype has been concentrating on further integration and communication with 1st and 2nd Screen components as well as other SAM components, e.g. Syndicator. Hence, the following views/functions have been provided as mock-ups for later development:

- **Video Viewer:** currently the Video IDs used for syndication purposes is hard-coded into the Generic Dashboard. In the next prototype, the Syndicator API should be used to obtain a complete list of syndicated video and streams.

6.5.1 Prototype 3 – Planned Tasks

Regarding the next steps in the Generic Dashboard component, Figure 69 summarises the tasks planned for the third prototype (to be delivered in M31).

Subcomponent	Task
Video Viewer	Implement communication with SAM Syndicator for obtaining video lists
Widget Manager	• Implement widget cache and prefetching to improve user experience • Implement mechanism for inviting other people to join
Widget Workspace Manager	Improve the overall look & feel of the Generic Dashboard
Context Listener	Collect additional data for the users' behaviour while interacting with the dashboard and provide information and knowledge for the users, their profiles and preferences. Access to the user profile will be also available through the Context Control client.
Syndicator	• Finalise the engine to register all the user interaction in the 2nd Screen • Finalise the communication with the Dashboard in order to implement the change of the context in Dynamic Communities • Any update needed to finalise the integration with Generic Dashboard

Figure 69: Tasks Planned for the Third Prototype of the Generic Dashboard Screen

6.6 Research Background

For the current prototype implementation and the overall approach, the following related research work was taken into consideration:

Source	Subcomponent	Description
[BAS13] Bassbouss, L.; Tritschler, M.; Steglich, S.; Tanaka, K.; Miyazaki, Y., "Towards a Multi-screen Application Model for the Web," Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual , vol., no., pp.528,533, 22-26 July 2013	General	This paper has been considered to implement Generic Dashboard using HTML5
(2015-10) ETSI TS 102 796 V1.3.1 / HbbTV 2.0	General	HbbTV specification http://www.etsi.org/deliver/etsi_ts%5C102700_102799%5C102796%5C01.03.01_60%5Cts_102796v010301p.pdf
[WCJ14] Lee, Wei-Po, Che Kaoli, and Jhih-Yuan Huang. "A smart TV system with body-gesture control, tag-based rating and context-aware recommendation." Knowledge-Based Systems 56 (2014): 167-178.	Context Control	Related work on the acquisition of contextual information and its usage
[MAY14] Maynard, G. HbbTV and Multi-Screen Strategy. HbbTV Symposium Asia. Singapore 18 June 2014.	Dashboard	Dashboard principles and examples http://www.ipnetcom.at/upload/News/multiscreen.pdf

6.7 Target Performance

The performance measurement of the 1st Screen prototype will be measured accordingly to the defined KPIs (described in section 6.7.1) with additional End User experience measurements (described in section 6.7.2).

6.7.1 Component KPIs

For this component the following Key Performance Indicators (KPIs) have been defined:

Topic	Description	Target KPI
Timeliness of related content	The time it takes for widgets to be updated and for the Inter-Widget Communication will affect the user's experience and thus needs to be low.	Time difference between video trigger and appearance of related content should be less than 5 sec.
Reusability	The Generic Dashboard component will be used on both 1st and 2nd Screen devices, so it needs to be easy to integrate in these components.	70% of the common components of the 1st and 2nd Screen should be implemented in the Multi-Device Dashboard

Figure 70: Target Performance Generic Dashboard

6.7.2 User Experience Measurements Tasks

Additional to the KPIs in Section 6.7.1, this work package provides user tasks, which are input for measuring the subjective user experience in a uniform way. For each of the tasks below the task-specific KPIs, defined in

References

[WDJ11] Wilson, S., Daniel, F., Jugel, U., Soi, S. (2011). Orchestrated User Interface Mashups Using W3C Widgets. Proceedings of Composable Web 2011.

[BWS13] Balasubramanee, V. Wimalasena, C. Singh, R. Pierce, M. (2013) Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development

[KAA07] Christian Kaar (2007), An introduction to Widgets with particular emphasis on Mobile Widgets

[LAR02] Larsson, S., (2002). Issue-based Dialogue Management, Gothenburg Monographs in Linguistic

[BCL05] B. Bringert, R. Cooper, P. Ljunglöf, A. Ranta, Multimodal Dialogue System Grammars. Proceedings of DIALOR'05, Ninth Workshop on the Semantics and Pragmatics of Dialogue, Nancy, France, June 9-11, 2005, 2005

[BRI07] B. Bringert. Speech Recognition Grammar Compilation in Grammatical Framework SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing, June 29, 2007, Prague. 2007.

[ZIE13] Ziegler, C., "Second screen for HbbTV — Automatic application launch and app-to-app communication enabling novel TV programme related second-screen scenarios," Consumer Electronics Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on, vol., no., pp.1, 5, 9-11 Sept. 2013

[VAN12] Vanattenhoven, J., Geerts, D. (2012). Second-Screen Use in the Home: An Ethnographic Study. In Proceedings 3rd International Workshop on Future Television, EuroITV 2012 (p. 12).

[VAN14] Vanattenhoven, J., Geerts, D., De Grooff, D. (2014). Television Experience Insights from HbbTV. In proceedings of the International Workshop on Interactive Content Consumption, Newcastle upon Tyne

[BAS13] Bassbouss, L.; Tritschler, M.; Steglich, S.; Tanaka, K.; Miyazaki, Y., "Towards a Multi-screen Application Model for the Web," Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual , vol., no., pp.528,533, 22-26 July 2013

(2015-10) ETSI TS 102 796 V1.3.1 / HbbTV 2.0

[WCJ14] Lee, Wei-Po, Che Kaoli, and Jhih-Yuan Huang. "A smart TV system with body-gesture control, tag-based rating and context-aware recommendation." Knowledge-Based Systems 56 (2014): 167-178.

[MAY14] Maynard, G. HbbTV and Multi-Screen Strategy. HbbTV Symposium Asia. Singapore 18 June 2014.

Annex A: User Research, will be measured.

Task	Description
Collapse widgets	The user needs to collapse a widget
Check related information	The user needs to point out where he can find related information about the video.
Send tweet	The user needs to send a tweet about the show he is watching in the Dynamic Social Community
Control a Widget	The user has to perform a Widget function (e.g. share the current film on Facebook) successfully.

Figure 71: Target Performance Multi-Device Dashboard – Task-Specific Measurements

6.8 Summary

This section provides a description of the second prototype of the Generic Dashboard component developed in task T.7.4 Multi-Device Dashboard. The main outcome of this task is the software of the Generic Dashboard. This prototype of T7.4 provides the basic user interface elements and their functionalities. This prototype is the second of four iterations planned for this component and the goal of this prototype is to cover 50% of the requirements of the component (see Deliverable D7.4.2 for additional information for this component in the lifespan of the project).

The most important goals reached during this second prototype have been:

- Showing to the End User dynamically loaded content from the Syndicator on the defined timeline to the related video
- Dynamically link with Multi Device Representation Component through the Syndicator to obtain each widgets contents and style allowing for adaptation of the look and feel of presented content
- Display real-time updated content from Social Components (Facebook / Tweeter) with the possibility to post or tweet while watching the video

The requirements necessary for both users and developers have been presented including installation instructions. The last section has been dedicated to describing the limitations of the current prototype and describing the next steps considered for the third version of the component, which should be delivered in M31.

7 Document Summary

This document, D7.9.2 is the second release of the deliverable series D7.9.x to provide early insight of the prototypes of software deliverables D7.1.2, D7.2.2, D7.3.2 and D7.4.2. This document provides information about the running tasks in Work Package 7:

- T7.1 Multi-Device Content and Media Representation (Section 3)
- T7.2 2nd Screen Media Interaction (Section 4)
- T7.3 1st Screen Media Interaction (Section 5)
- T7.4 Multi-Device Dashboard (Section 6)

The information for each task provided contains:

- Scope of the pilot implementation, its purpose and the main relationships with other modules implemented in SAM in the first year of development
- Information needed to deal with the pilot in terms of technical and non-technical requirements, software to be installed, etc.
- Steps needed to install the pilot software and process to build it from source code
- Different screens and actions implemented at the pilot itself, ways to access it, and to test the different implemented options
- Current pilot limitations and the expected improvements
- Papers and other scientific information considered
- Key Performance Indicators (KPI's) for the SAM component
- Conclusion of the implementation of the first and second prototype

References

[WDJ11] Wilson, S., Daniel, F., Jugel, U., Soi, S. (2011). Orchestrated User Interface Mashups Using W3C Widgets. Proceedings of Composable Web 2011.

[BWS13] Balasubramanee, V. Wimalasena, C. Singh, R. Pierce, M. (2013) Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development

[KAA07] Christian Kaar (2007), An introduction to Widgets with particular emphasis on Mobile Widgets

[LAR02] Larsson, S., (2002). Issue-based Dialogue Management, Gothenburg Monographs in Linguistic

[BCL05] B. Bringert, R. Cooper, P. Ljunglöf, A. Ranta, Multimodal Dialogue System Grammars. Proceedings of DIALOR'05, Ninth Workshop on the Semantics and Pragmatics of Dialogue, Nancy, France, June 9-11, 2005, 2005

[BRI07] B. Bringert. Speech Recognition Grammar Compilation in Grammatical Framework SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing, June 29, 2007, Prague. 2007.

[ZIE13] Ziegler, C., "Second screen for HbbTV — Automatic application launch and app-to-app communication enabling novel TV programme related second-screen scenarios," Consumer Electronics Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference on, vol., no., pp.1, 5, 9-11 Sept. 2013

[VAN12] Vanattenhoven, J., Geerts, D. (2012). Second-Screen Use in the Home: An Ethnographic Study. In Proceedings 3rd International Workshop on Future Television, EuroITV 2012 (p. 12).

[VAN14] Vanattenhoven, J., Geerts, D., De Grooff, D. (2014). Television Experience Insights from HbbTV. In proceedings of the International Workshop on Interactive Content Consumption, Newcastle upon Tyne

[BAS13] Bassbouss, L.; Tritschler, M.; Steglich, S.; Tanaka, K.; Miyazaki, Y., "Towards a Multi-screen Application Model for the Web," Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual , vol., no., pp.528,533, 22-26 July 2013

(2015-10) ETSI TS 102 796 V1.3.1 / HbbTV 2.0

[WCJ14] Lee, Wei-Po, Che Kaoli, and Jhih-Yuan Huang. "A smart TV system with body-gesture control, tag-based rating and context-aware recommendation." Knowledge-Based Systems 56 (2014): 167-178.

[MAY14] Maynard, G. HbbTV and Multi-Screen Strategy. HbbTV Symposium Asia. Singapore 18 June 2014.

Annex A: User Research

Generic Target Performance KPI's

For assessing the target performance of different components of the Multi-Device Media Representation and Interaction work package (Sections 4 to 6.7.2), 5 Target KPI's were defined as below. Every task defined in Section 4, 4.7.2, 5.7.2 and 6.7.2 will be used to measure the 3 task-specific target KPI's. Next to that, 2 general usability KPI's were defined in order to measure the overall user experience.

Type	Type	Description	Target KPI
Task-specific	First Click score	After a limited amount of usage (familiarisation time, described below), did the user click the correct button for the optimal path to complete the requested task?	80% correctness for "top tasks" should be reached, 65% for other tasks
Task-specific	Task Completion	Was the user able to complete the task within a reasonable time? (A reasonable threshold depends on the task and on the product – it's best for someone who knows it well to set this)	80% correctness for "top tasks" should be reached, 65% for other tasks
Task-specific	Task Confidence	How confident is the user that s/he completed the task requested?	Average score of 5 or above (7-point scale)
Overall Usability	System Usability Scale (SUS)[20]	This is a survey with 11 statements with an associated 7-point scale. The procedure for executing the SUS is described on the website in footnote 20.	Composite of 70 or higher (100-point scale)
Overall Usability	Usability Adjective Scale	This is a single question that is usually added to the end of the SUS that gives a scale of adjectives to describe whatever system is being measured from "worst imaginable" to "best imaginable"	4 or higher (7-point scale)

Figure 72: Generic Target Performance KPI's

Testing procedure

A very short description of the testing procedure to follow is described below. The details can be found on the website in footnote 20.

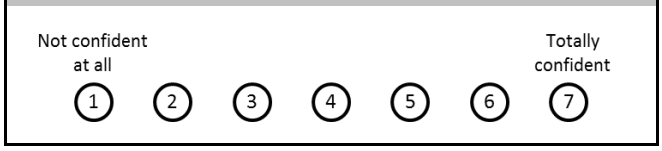
1. Recruit at least 10 research participants with little to no knowledge of the system.
2. Give each of these participants a brief (less than 5 minute) introduction to the feature under test. If possible, use the actual materials that a consumer would receive.
3. Foresee for each participant a short amount of time to familiarize himself with the feature (usually less than 5 minutes for a consumer is good).
4. The moderator then asks each participant to complete a series of tasks. These tasks should be the most common actions users would take when working with the feature. (For SAM, this would include widget configuration, TV-side navigation, and 2nd Screen app usage.) Within each of those tasks there may be more than one sub-task for the

[20] http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

participant. Of these sub-tasks, identify, by observing the behaviour of the participants, the sub-tasks that are absolutely crucial for using the feature.

As a general guideline, the moderators should not guide or assist the participants in any way – they are there only to take notes and ensure the participants attempt each task and fill out the survey.

After each task, each participant answers a question about the ease of the task and how confident he is that he successfully completed the task. These statements are graded on a 5 or 7-point scale.

Task instruction	Ease of task / Confidence
Please collapse the video info widget	
How confident are you that you successfully completed the task?	

The moderator should note whether the participant made the correct first click and whether the participant successfully completed the task.

After all tasks are completed, the participants are then asked to complete the SUS[21] including the Usability Adjective Scale. This is a survey with 11 statements with an associated 7-point scale.

[21] http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html