Middle East Technical University
Department of Computer Engineering
CENG490 Computer Engineering Design
Fall 2010

Initial Software Design Document for

watch & touch

*by **DialecTech***
Giray Havur (1630870)
Melike Ercan (1560135)
Utku Şirin (1560838)
Yaman Umuroğlu (1560614)

Table of Contents

1. Introduction

Initial Software Design Document of Watch & Touch provides the necessary definitions to conceptualize and further formalize the design of the software, of which its requirements and functionalities were summarized in the previous requirements analysis report. The aim is to provide a guide to a design that could be easily implemented by any designer reading this report.

1.1. Problem Definition

The aim of Watch & Touch is to create a complete, free & open source, multi-platform, interactive whiteboard system which supports multi-touch gestures, complemented by an in-classroom collaborative working client to increase student participation in classroom activities. Hardware and software solutions targeting these have existed for many years, especially as an integrated part of "smart class" solutions, but deployment is not widespread (only 2 smart classes in METU) due to their high cost (around a thousand dollars[2] for interactive whiteboard units not including the projector, and much higher costs for complete smart class solutions). A review of existing solutions can be found in the *1.4. User and Literature Survey* section of the Watch & Touch SRS[2] . Considering existing solutions or methods, it is desired to create a **complete**, **free & open source**, **multi-platform**, **interactive blackboard system** that can be utilized at the relatively low cost of its available off-the-shelf hardware components, as well as an **in-classroom collaborative working environment** to complement the classroom activity when the necessary hardware is available.

1.2. Purpose

In the Watch & Touch SRS[2] document, Watch & Touch's desired features and requirements were stated. This SDD is intended to provide a software system design which will satisfy the given functional and non-functional requirements in line with the provided assumptions and constraints. As the design of the system is the core of its development, the document is intended to be viewed primarily by the DialecTech team which will be developing Watch & Touch, and will be presented to the Department of Computer Engineering faculty as part of the Senior Computer Engineering Design (CENG491) course.

1.3. Scope

Scope of ISDD is to provide information about design of the project. This document covers the architectural design, data design, procedural design, design constraints, development schedule. Also hardware and software requirements and working environment will be explained.

1.4. Overview

This document provides information about Watch & Touch software system.

As it is explained in section 2, the system consists of two separate but interconnected software applications, named the Interactive Whiteboard Client (IWBC) and the Collaboration Client (CBC). This document explains these two components in detail. In section 3, design issues such as assumptions, dependencies and constraints, design goals and guidelines that are used is explained. In section 4, information about data design and data entities in the system is provided. Section 5 is about the system's overall architecture; the architectural design of the software applications and detailed description of modules are explicated. Section 6 provides design details on the user interfaces. Section 7 includes the libraries and tools that will be used during software development. In section 8, time planning and scheduling issues will be demonstrated by a Gannt Chart. The document is ended with a conclusion that will be given in section 9. Additional UML diagrams are provided for further clarification in section 10.

1.5. Definitions, Acronyms and Abbreviations

Definitions, Acronyms and Abbreviations in Watch & Touch Software are explained in the following table.

Wiimote	The hardware controller designed by Nintendo for use with the Nintendo Wii gaming console, that offers the tracking of movements of up to 4 infrared points
Watch & Touch	The software system hereby documented, including the IWBC and CBC clients
machine	A computing device, used either by the instructor or one of the students, that fulfills the Watch & Touch requirements
IWB	Interactive Whiteboard, the interactive projected display created using the Watch & Touch IWBC
IWBC	Interactive Whiteboard Client, the piece of software installed on the instructor's machine which allows interaction on the projected display using the IR pen and the Wiimote, and provides annotation features.
CBC	The Watch & Touch Collaboration Client, the piece of software installed on student machines that allows them to work collaboratively
collaborative working	A process in which several users are able to simultaneously make modifications to a common document and see the modifications done by others
collaboration hub	The machine which will serve as a hub for in-classroom collaboration, which is determined as the instructor's machine
annotation	The ability to create user content such as hand-made drawings, shapes or text on top of other content on the screen
multi-touch gesture	A predefined movement involving multiple points of interaction

pinch gesture	A multi-touch gesture in which two points along a line move towards, or away from, their center
rotation gesture	A multi-touch gesture in which two points rotate clockwise or counterclockwise along a fixed axis
swipe gesture	A multi-touch gesture in which two points move towards the same horizontal direction
IR pen	Infrared pen, a pen-shaped object whose tip radiates Wiimote-detectable infrared light when desired
IR ring	An IR pen that can be worn on a finger, several can be used multi-touch gestures
content	Visual content in presentation, video or webpage form.
SDD	Software Design Document
SVG	Scalable Vector Graphics

1.6. References

[1]
IEEE Std 1016-1998: IEEE Recommended Practice for Software Design Descriptions

A common software engineering standard to provide some guidance and recommended approaches for specifying software design descriptions

[2]
Software Requirements Specification for Watch & Touch

The SRS document for Watch & Touch, prepared according to the *IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications*

[3]
The KISS Principle in design process
http://en.wikipedia.org/wiki/KISS_principle

[4]
Designing QT-Style C++ APIs Guideline
http://doc.trolltech.com/qq/qq13-apis.html

[5]
Comparison of screencasting software
http://en.wikipedia.org/wiki/Comparison_of_screencasting_software

[6]
Free and Open Source software (FoSS)
http://en.wikipedia.org/wiki/Free_and_open_source_software

[7]
Cornell University Ergonomic Guidelines for User-Interface Design
http://ergo.human.cornell.edu/ahtutorials/interface.html

[8]
The Scalable Vector Graphics file format
http://en.wikipedia.org/wiki/Svg

[9]
Message-Digest Algorithm 5
http://en.wikipedia.org/wiki/MD5

[10]
The Poppler Qt4 interface library
http://people.freedesktop.org/~aacid/docs/qt4/

[11]
Phonon multimedia framework overview
http://doc.qt.nokia.com/latest/phonon-overview.html

[12]
WebKit in Qt
http://doc.qt.nokia.com/4.7-snapshot/qtwebkit.html#details

2. System Overview

Watch & Touch consists mainly of two separate but interconnected software applications, named the **Interactive Whiteboard Client (IWBC)** and the **Collaboration Client (CBC)**.

2.1. The Interactive Whiteboard Client

The IWBC is an application that is capable of turning any projected surface into a multi-touch-input whiteboard. In terms of providing **interaction**, it calculates the position of the infrared pen on the surface (by utilizing the input from a Wiimote) in terms of screen coordinates and generates mouse events accordingly to provide interaction on the selected point, as well as interpret the movements of multiple IR sources as **multi-touch gestures** to enhance the general user experience. An important consequence of this capability is the **annotation** features provided - the instructor is able to display presentations, web pages and videos from inside the IWBC and take handwritten notes on top of these visual content when desired. The IWBC also allows the instructor to actively partake in the classroom **collaboration** activities, in a way similar to the facilities provided to the students by the CBC.

2.2. The Collaboration Client

The CBC is a smaller application intended for use by students on their own machines, and allows them to engage in classroom **collaboration** by creating collaborative drawings. The CBC establishes collaborative links to other machines which also have . The collaborative link exists not only between students but also between students and the instructor; students can see the information on the instructor's projected screen on their own machines, and (with the instructor's permission) make drawings on the content seen on the instructor's screen, to ask questions or express themselves by visually emphasizing or annotating the currently displayed content.

3. Design Considerations

Special design issues which need to be addressed or resolved before attempting to devise a complete design solution are noted here.

3.1. Design Assumptions, Dependencies and Constraints

Watch & Touch is assumed to operate under the presence of certain factors with regard to hardware, system software and general properties of the operating environment. These factors are stated, separately for the IWBC and the CBC, in the following two subsections 3.1.1. and 3.1.2.

3.1.1. Interactive Whiteboard Client

- An operational Wiimote is assumed to be present and connected via Bluetooth to the instructor's machine
- The IWBC must be installed and successfully launching on the instructor's machine
- At least one IR pen with the necessary characteristics for being detected by the Wiimote must be present and functional. For multi-touch gestures, at least two such IR sources are required.
- A flat projected surface is required to be turned into an interactive whiteboard.

- The Wiimote must be positioned in such a way that all four corners of the projected display are within the line of sight of the Wiimote infrared camera.

3.1.2. Collaborativity Client:

- At least several student machines should be present. The ideal would be one machine per student, however this is not a requirement.
- The Watch & Touch collaborativity client must be installed and successfully launching on all the machines intended for collaboration
- The machines intended for collaboration must be connected by a Local Area Network, including the instructor's machine.
- The machines must have the necessary two-dimensional input hardware interface (such as a mouse or touch surface) that gives them the capability to provide the input for drawing

3.2. Design Goals and Guidelines

- Both software applications in the system (IWBC and CBC) must be capable of working with full functionality on multiple platforms and operating systems. Towards this end, the Qt framework will be used in the development of the project. Microsoft Windows XP® and Ubuntu 10.04 have been chosen as the testing platforms due to their widespread use[2].
- Since Watch & Touch is heavily centered on its user interfaces:
    - the user interfaces must be consistent, clean and easy to use in general
    - the Cornell University Ergonomic Guidelines for User-Interface Design[7] will serve as the guideline during the user interface design process
    - existing Qt widgets providing UI functionality should be preferred as much as possible instead of re-creating user interface components, as this will consume more time and be less stable
    - where possible, a smaller degree of abstraction should be preferred between the underlying business logic and the user interface, to keep a simpler software structure
- During the design process, the KISS principle will serve as the primary guideline for Watch & Touch. The KISS principle[3] states that simplicity should be a key goal in design, and that unnecessary complexity should be avoided.
- During design and development, the principles mentioned in the Designing Qt Style C++ APIs [4] document will be followed to obtain source code which is easier to understand for developers, and more suitable for Qt's style.
- To keep Watch & Touch as free and open[6] as possible, use of external libraries and tools which do not offer LGPL or GPL should be avoided.
- The Scalable Vector Graphics (SVG) format[8] should be used to store annotation and drawing data, for the following reasons:
    - Qt has good built-in support for SVG
    - vector graphics allows for easier processing of individual annotation elements

- scaling the annotations does not cause any loss of quality
- format is suitable for extensions and metadata storage since it is XML-based

4. Data Design

4.1. Data Description

This section contains a description of the information domain for Watch & Touch. First, a description of the structure of defined data entities is provided, followed by two sections providing an overview of how the annotation storage and content display work.

Watch & Touch does not make extensive use of relational data, thus no databases are utilized. In fact the only persistent data is for the annotations (whose storage is described in section 4.1.2) and the configuration data (4.1.1.8).

4.1.1. Description of data entities

4.1.1.1. AnnotationBase data entity

The AnnotationBase data entity describes the basic content of an annotation and forms the base of the other *Annotation data entities.

Field Name	Data Type	Description
ParentID	String	The MD5 hash of the parent content
Geometry	Rectangle	Position and size of annotation
LayerCount	Integer	Number of items in the Layers field
Layers	Array of SVGData	Annotation SVG data is contained in these items

4.1.1.2. PresentationAnnotation data entity

The PresentationAnnotation data entity describes the annotation on a single presentation slide.

Field Name	Data Type	Description
SlideNumber	Integer	Which slide the annotation belongs to
Data	AnnotationBase	Base annotation data entity

4.1.1.3. WebpageAnnotation data entity

The WebpageAnnotation data entity describes the annotation on a single webpage anchor.

Field Name	Data Type	Description
AnchorName	String	A string identifying the webpage anchor on which the annotation was made
Data	AnnotationBase	Base annotation data entity

4.1.1.4. VideoAnnotation data entity

The VideoAnnotation data entity describes the annotation on a single video frame.

Field Name	Data Type	Description
TimelinePosition	String	The position in the video timeline to which the annotation belongs (hh:mm:ss)
Data	AnnotationBase	Base annotation data entity

4.1.1.5. CollaborativeSession data entity

The CollaborativeSession data entity describes a group of participants working on a collaborative drawing.

Field Name	Data Type	Description
ParticipantCount	Integer	Number of participants in the session
Participants	Array of CollaborativeParticipant	Information on participants in the session
SessionName	String	The name of the session, visible to all
SessionPassword	String	MD5 hash of session password
TheDrawing	CollaborativeDrawing	The collaborative drawing being created by this session
CreationDate	DateTime	When the session was created

4.1.1.6. CollaborativeParticipant data entity

The CollaborativeParticipant data entity describes a participant in the collaborative drawing session.

Field Name	Data Type	Description
NameSurname	String	Publically visible name and surname of participant
EMail	String	E-mail address of the participant
IsInstructor	Boolean	Whether the participant has instructor privileges

4.1.1.7. CollaborativeDrawing data entity

The CollaborativeDrawing data entity describes a single collaborative drawing.

Field Name	Data Type	Description
Session	CollaborativeSession	The collaborative drawing session which is creating this drawing.
LastModified	DateTime	When the drawing was last modified
LayerCount	Integer	Number of layers in the drawing
Layers	Array of SVGData	Actual drawing SVG data in layers

4.1.1.8. Configuration data entity

The Configuration data entity describes the current configuration of the IWBC. It is made persistent in XML form, where the field names serve as tag names.

Field Name	Data Type	Description
WiimoteData	WiimoteConfig	Information pertaining to the currently connected Wiimote.
ScreencastingBackend	String	Which screencasting backend is to be used (“ffmpeg” or “vlc”)
GestureData	Array of GestureMap	The matching of gestures to actions

4.1.1.9. WiimoteConfig data entity

The Configuration data entity describes the current configuration of the Wiimote controller. It is made persistent in XML form, where the field names serve as tag names.

Field Name	Data Type	Description
UniqueID	Integer	The unique identifier assigned to the Wiimote by the wiiuse library
CalibrationData	Array of (Float, Float)	The raw IR coordinates of the four IR calibration points received

4.1.1.10. GestureMap data entity

The GestureMap data entity describes a single gesture-action mapping. It is made persistent in XML form, where the field names serve as tag names.

Field Name	Data Type	Description
GestureID	GestureEnum	The identifier of the predefined gesture
ActionID	ActionEnum	The identifier of the predefined action

4.1.2. Annotation data storage

This section gives a general description of how the annotations made on visual content are stored and retrieved.

4.1.2.1. Storage logic

The matching of content and annotations is made through a unique identifier. Each visual content is has a unique identifier, which represents the current state of the content file. This identifier is generated by running the MD5 algorithm[9] on the content file. This process ensures that the identifier is not modified by trivial changes such as changing the content filename, and prevents mismatches between the annotation and the content file in case the content changes.

All annotations created by the IWBC are stored under the annotations/ directory, located under the IWBC application directory. For each content which is annotated by the user, a sub-directory under annotations/ is created. This sub-directory is named as the unique identifier of the annotated content, and inside it are all the annotation data files which belong to this content.

Each annotation data file contains the annotation made in a different part of the content, and is named according to the following convention:

annotation_%subcontextid_%layerno.wta

The *%subcontextid* identifies which part of the content the annotation is related to. It is generated differently for different content types:

- *Presentations:* slide number, starting from 0
- *Webpages:* anchor name (such as #section1) if exists, nothing otherwise
- *Videos:* timeline position (hh:mm:ss) in the video

The *%layerno* identifies the layer number of the annotation. Currently, all annotation data is stored in layer number 0; if future expansion layers are needed (which will be numbered 1, 2, 3...) they will be stacked in front of layer 0.

4.1.2.2. File format

All annotations will be stored using the Scalable Vector Graphics (SVG) format, an XML-based file format for describing two-dimensional vector graphics[8]. See section 3.2 for the reasons why SVG was chosen as the annotation storage format.

Since the actual vector data will be generated using the Qt's QSvgGenerator and rendered using QSvgRenderer, no further details will be provided here as to how the annotation vector data is generated.

4.1.3. Supported content file formats

This section provides a list of the supported content file formats for presentations, videos and webpages. As long as the general content format is preserved (numbered slides for presentations, vertical scrolling for webpages and a hour/minute/second partitioned timeline for videos), support for other file formats can be added simply by adding the corresponding viewer code.

4.1.3.1. Presentations

Portable Document Format (PDF) will be the primary supported format for presentations. These files will be displayed using the Poppler Qt4 interface library[10]. Since the viewing is done by this external library, no further details on the PDF file format structure are provided here.

Additionally, support for Microsoft PowerPoint and OpenDocument Presentation formats should also be provided if time allows. This support can be provided by converting these formats to PDF via some command line tool.

4.1.3.2. Videos

The video playback functionality will be provided by the Phonon multimedia framework[11] which is a part of Qt. All seek-enabled video formats which is supported by the current Phonon backend are supported. Since the video display is handled by Phonon, no further details on the video file format structures are provided here.

4.1.3.3. Webpages

The displaying of webpages will be done using the integrated WebKit support[12] in Qt; thus, all HTML subformats supported by WebKit are also supported. The webpages are assumed to be only vertical-scrolling (horizontal scrolling is not supported) and static in terms of geometry (interacting with links etc. does not change the width/height of the webpage). Since the webpage display is handled by WebKit, no further details on the HTML file format structure are provided here.

4.2. Data Dictionary

Name	Type	Refer to Section
AnnotationBase	data entity	4.1.1.1
AnnotationWidget	component	5.2.3
BaseDrawingWidget	component	5.2.3
ClassroomSesson	component	5.2.4
CollaborativeDrawing	data entity	4.1.1.7
CollaborativeDrawingTask	component	5.2.1
CollaborativeDrawingWidget	component	5.2.3
CollaborativeParticipant	data entity	4.1.1.6
CollaborativeSession	data entity	4.1.1.5
Configuration	data entity	4.1.1.8
Configuration	module	5.2.6
ConfigurationData	component	5.2.6
ConfigurationManager	component	5.2.6
ConfigurationTask	component	5.2.1

ContentDisplay	module	5.2.2
ContentDisplay	component	5.2.2
ContentDisplayTask	component	5.2.1
ContentMatcher	component	5.2.3
ContentSelector	component	5.2.2
ContextRecognizer	component	5.2.3
Drawing	module	5.2.3
DrawingData	component	5.2.3
EmailSender component	component	5.2.4
EventGenerator	component	5.2.5
FileSystemBrowser	component	5.2.2
GestureMap	data entity	4.1.1.10
GestureMapper	component	5.2.6
GoogleDocsAccess	component	5.2.2
IContentSelection	interface	5.2.2
IDisplayedSelection	interface	5.2.2
InputCalibration	component	5.2.5
InputReceiver	component	5.2.5
ITask	interface	5.2.1
MessageTransciever	component	5.2.4
PresentationAnnotation	data entity	4.1.1.2
PresentationDisplay	component	5.2.2
PresentationDisplayTask	component	5.2.1
RecentlyUsed	component	5.2.2
ScreenCapture	component	5.2.3

Screencasting	component	5.2.2
Session	component	5.2.4
SessionManagementTask	component	5.2.1
SessionManager	component	5.2.4
Sessions	module	5.2.4
SessionUser	component	5.2.4
SketchingTask	component	5.2.1
SketchingWidget	component	5.2.3
SnapshotRetriever	component	5.2.4
Task	component	5.2.1
TaskManagement	module	5.2.1
TaskManager	component	5.2.1
VideoAnnotation	data entity	4.1.1.4
VideoDisplay	component	5.2.2
VideoDisplayTask	component	5.2.1
WebpageAnnotation	data entity	4.1.1.3
WebpageDisplay	component	5.2.2
WebPageDisplayTask	component	5.2.1
WiimoteConfig	data entity	4.1.1.9
WiimoteInput	module	5.2.5
WiimoteManager	component	5.2.5
wiiuse	component	5.2.5

5. System Architecture

A general description of the Watch & Touch software system architecture is presented in the following sections, in a top-down manner.

5.1. Architectural Design

Watch & Touch will include two separate but related software applications, the Interactive Whiteboard Client and the Collaboration Client whose functional descriptions and requirements are provided in the sections 2.2 and 3.2.1 of Watch & Touch SRS[2]. The deployment diagram below illustrates the topmost level view of Watch & Touch architecture by presenting a view of how the applications will be deployed in the usage environment.

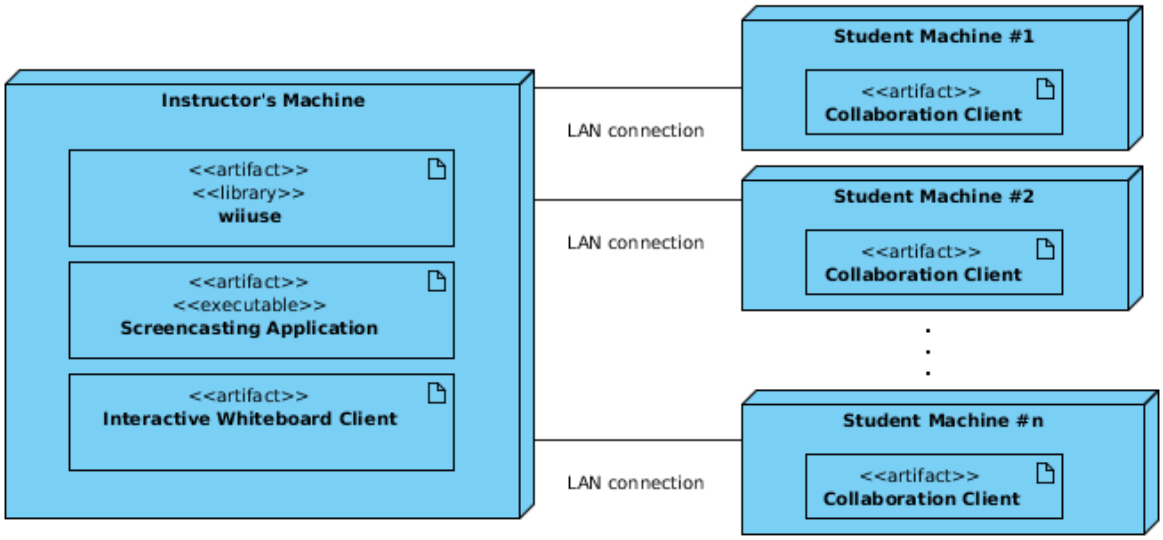


Figure 5.1.1: Deployment diagram for Watch & Touch, showing the physical positioning of the IWB and CBC applications on instructor and student machines

5.1.1. Architecture of Interactive Whiteboard Client

The IWBC, which provides the primary interactive whiteboard-related functionality such as content display and annotation, as well as collaboration-related functionality, will be composed of six top-level modules, each formed of components grouped according to their functional similarity. In the following subsections of this section, a brief overview of the responsibilities of each of these modules is provided. A simplified overview of the IWBC architecture *(simplified: not all components and associations inside the modules are shown - consult the following subsections for more detailed views of the internal module structures)* intending to show how these modules are connected together is presented in the following component diagram.

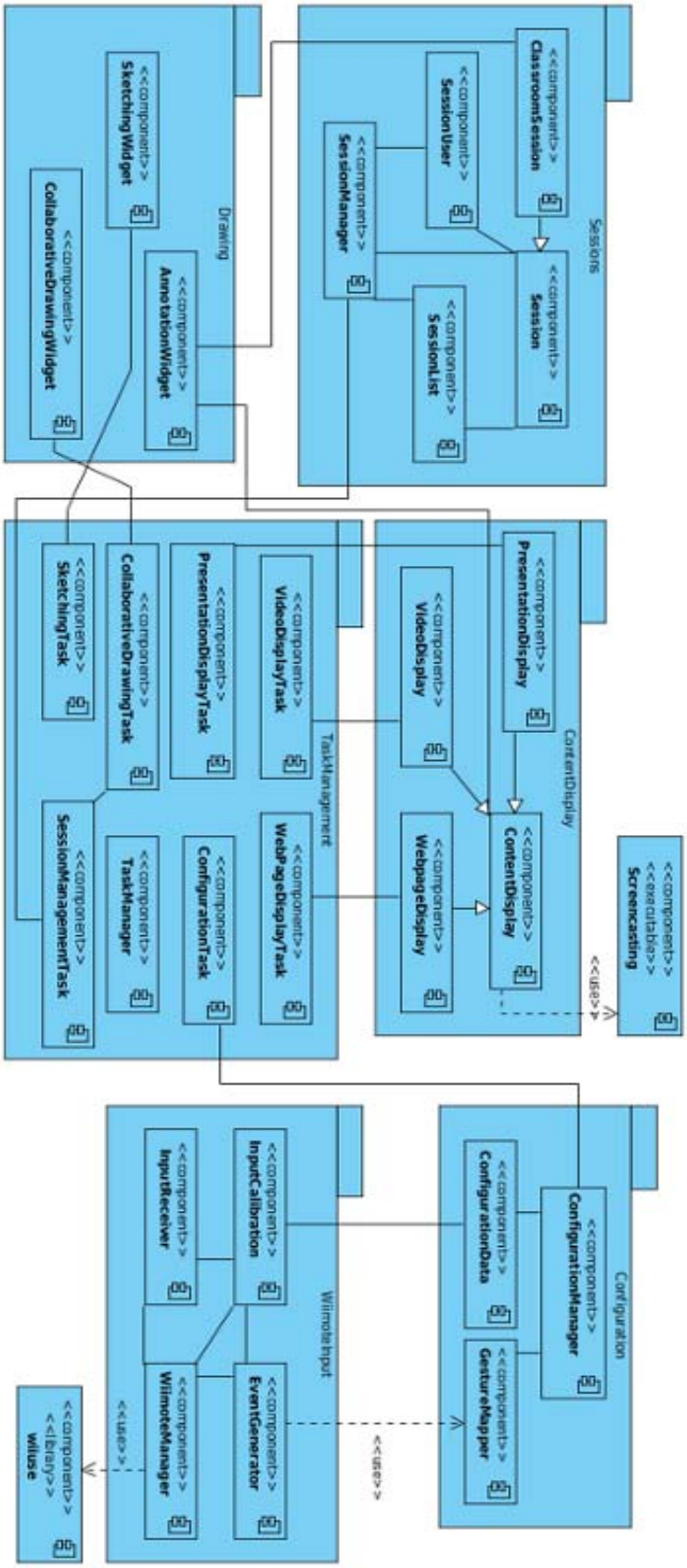


Figure 5.1.1.1: Simplified component diagram for the Watch & Touch Interactive Whiteboard Client.

5.1.2. Architecture of Collaboration Client

The CBC, which basically provides collaboration-related functionality, will be composed of three top-level modules, each formed of components grouped according to their functional similarity. In the following subsections of this section, a brief overview of the responsibilities of each of these modules is provided. A simplified overview of the CBC architecture *(simplified: not all components and associations inside the modules are shown - consult the following subsections for more detailed views of the internal module structures)* intending to show how these modules are connected together is presented in the following component diagram.

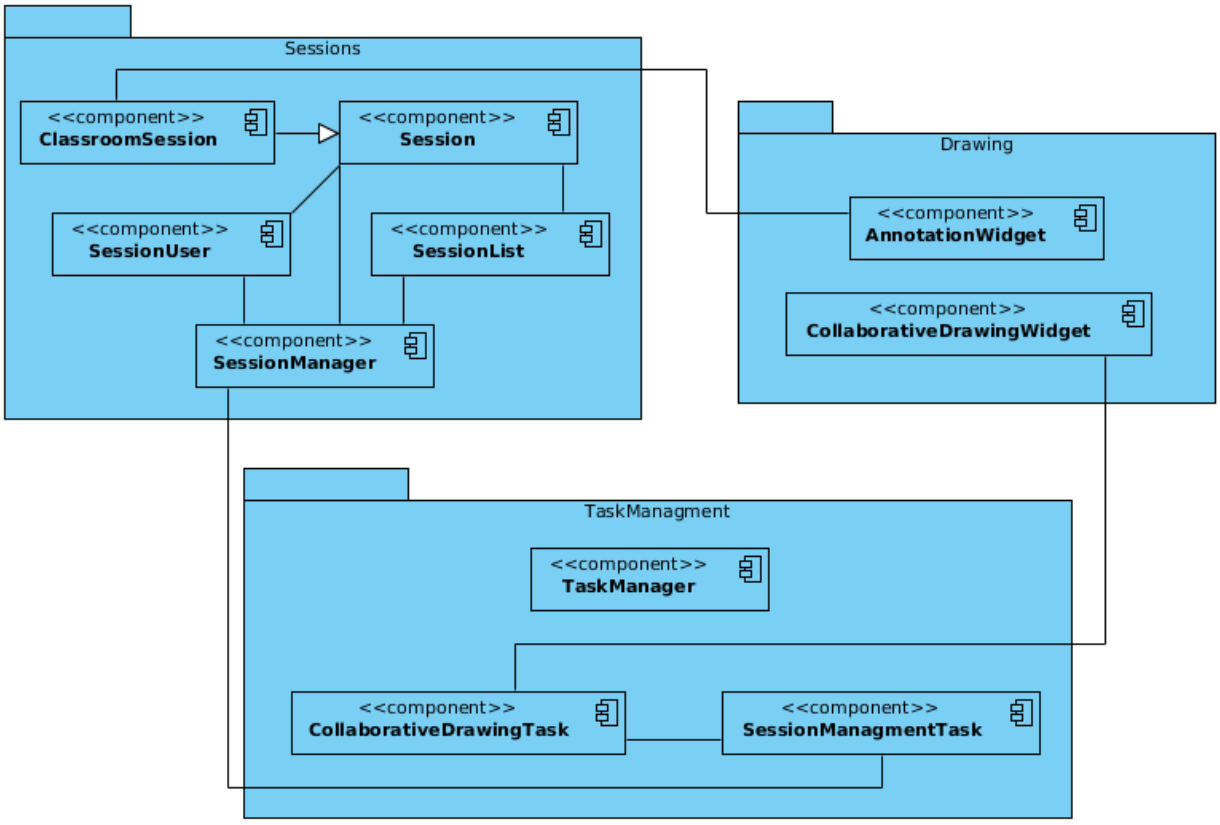


Figure 5.2.1.1: Simplified component diagram for the Watch and Touch Collaboration Client

As can be noted from the above diagram, the CBC architecture is essentially a minimized subset of the IWBC architecture - the same components are reused with a lesser degree of functionality, thus a separate module description for the CBC is not provided.

5.2. Description of Modules

5.2.1. TaskManagement module

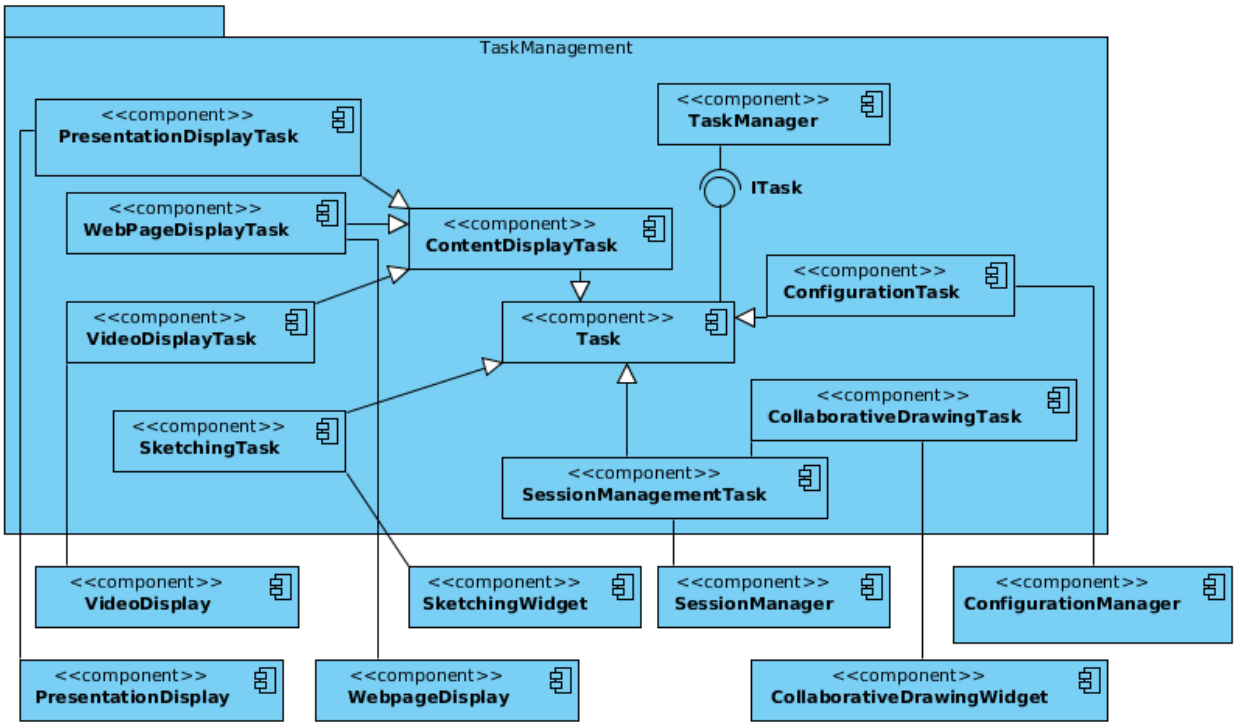


Figure 5.2.1.a: Component diagram for the TaskManagement module

5.2.1.1. Processing narrative

The TaskManagement module is responsible for the logical and visual management of the instances of multiple tasks which the IWBC is capable of - creating new task instances, switching between and terminating existing instances, and forming a basis for the user interface elements related with these tasks - the **TaskManager** is responsible for the IWBC Main Menu and each **QWidget**-derived ***Task** component is responsible for its own user interface elements.

5.2.1.2. Interface description

ITask: allows performing the user interface related tasks (set window geometry, minimize, maximize, show/hide...) and is realized by the **QWidget** set of member functions with which the component is associated.

5.2.1.3. Processing detail

The **TaskManager** component creates new ***Task** instances, stores the existing instances in an internal structure, and performs the necessary operations on them when requested through the user interface.

5.2.1.4. Dynamic behavior

Each ***Task** component is associated with a **QWidget**-derived component, which realizes the user interface and specific functionality for that particular task. The mapping is as follows:

- **PresentationDisplayTask** to **ContentDisplay::PresentationDisplay**
- **VideoDisplayTask** to **ContentDisplay::VideoDisplay**
- **WebpageDisplayTask** to **ContentDisplay::WebpageDisplay**
- **SketchingTask** to **Drawing::SketchingWidget**
- **SessionManagementTask** to **Sessions::SessionManager**
- **CollaborativeDrawingTask** to **Drawing::CollaborativeDrawingWidget**
- **ConfigurationTask** to **Configuration::ConfigurationManager**

5.2.2. ContentDisplay module

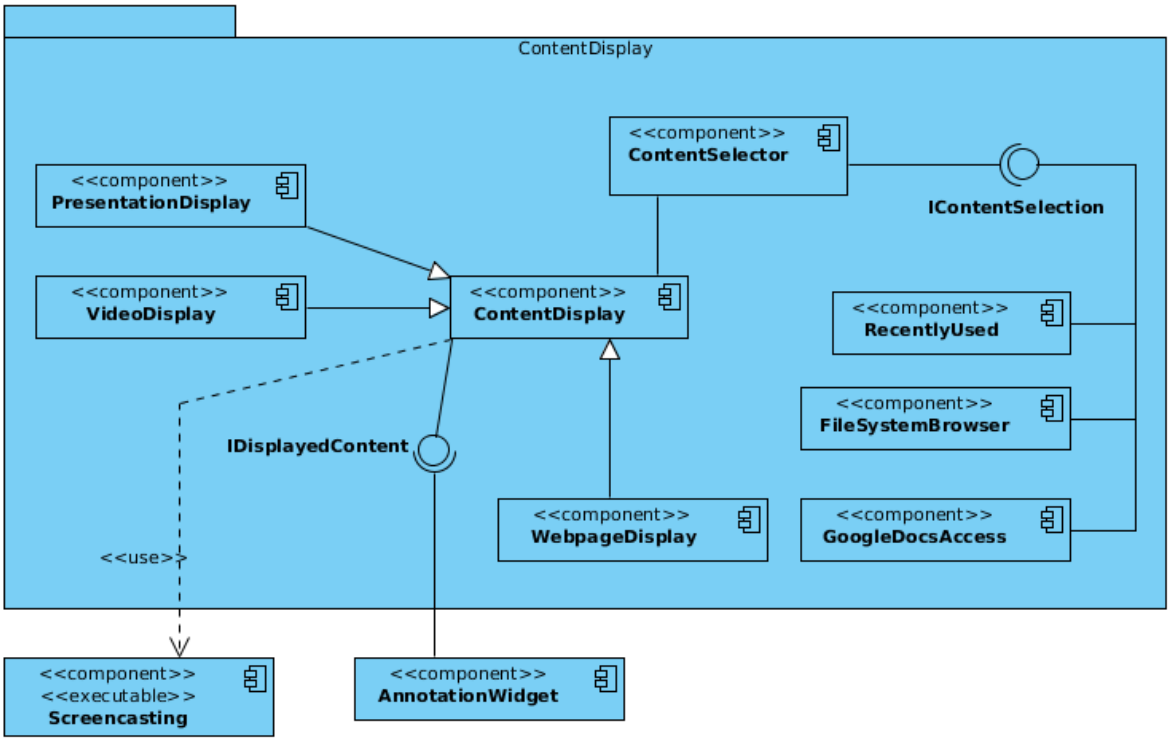


Figure 5.2.2.a: Component diagram for the ContentDisplay module

5.2.2.1. Processing narrative

The ContentDisplay module provides the necessary functionality for displaying the three primary kinds of visual content the IWBC supports: presentations (**PresentationDisplay**), videos (**VideoDisplay**) and webpages (**WebpageDisplay**). These display-related components provide the graphical basis for annotation, and they can access the external **Screencasting** component to start/stop the screencasting operation. The business logic for choosing which particular file will be displayed is also handled here, by the **RecentlyUsed**, **FileSystemBrowser** and **GoogleDocsAccess** components.

5.2.2.2. Interface description

IContentSelection: allows placing type restrictions on the selectable files and retrieving the selected file name.
IDisplayedContent: provides info the current status of the displayed content (current slide number for presentations, timeline location for videos, scroll position for webpages), used for recognizing the annotation context

5.2.2.3. Processing detail

The **ContentSelector** component retrieves the selected filename from one of the components implementing the *IContentSelection* interface, and supplies the preferred **ContentDisplay**-derived display component (one of **PresentationDisplay**, **VideoDisplay** and **WebpageDisplay**) with the selected file. The display component then renders and displays the selected file, enabling the user to browse through the content (next/previous slide for presentations, playback control for videos, scrolling and link following for webpages) through its own user interface.

5.2.2.4. Dynamic behavior

- The **ContentDisplay** component sends the current status of the displayed content to **Drawing::AnnotationWidget** so it can display the relevant annotations [Sequence Diagram 10.1.3]
- When requested, the display component can access the external **Screencasting** component to start/stop the screencasting operation.

5.2.3. Drawing module

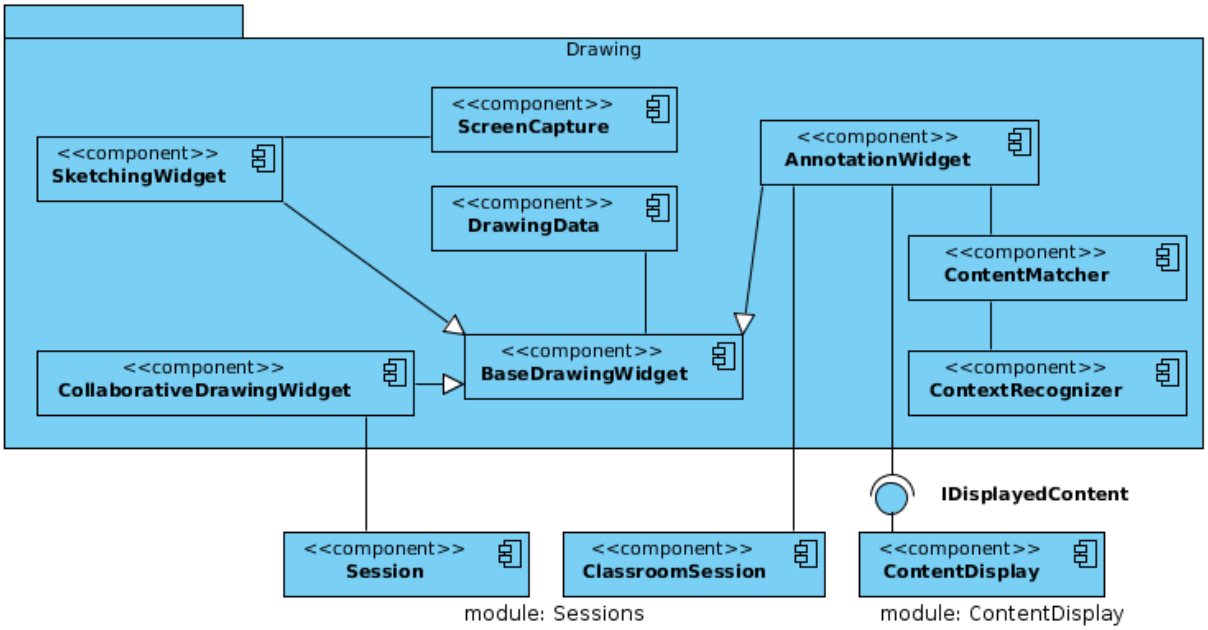


Figure 5.2.3.a: Component diagram for the Drawing module

5.2.3.1. Processing narrative

The Drawing module contains the components responsible for realizing the drawing operations which play a prime part in Watch & Touch's functionality. The three components **AnnotationWidget**, **SketchingWidget** and **CollaborativeDrawingWidget** offer annotation, sketching (on blank pages) and collaborative drawing capabilities, respectively. The **DrawingData** component is responsible for storing the drawing data and importing/exporting into files when requested. The **ScreenCapture** component offers user-directed screen capturing capabilities, whose raster forms can be used as sketching backgrounds later on. The **ContentMatcher** and **ContentRecognizer** components are responsible for recognizing the current state of the displayed content and fetch the relevant drawing data for annotations.

5.2.3.2. Interface description

IDisplayedContent: see section 5.2.2.2

All other inter-component links are associations, meaning that mutual access to both connected components' public properties and functions are possible.

5.2.3.3. Processing detail

Drawing is performed on the user interface of the **BaseDrawingWidget** and derived components, and the drawing data is kept in the **DrawingData** component which can export or import this data as SVG images when requested. In addition to these basic drawing operations:

- The **AnnotationWidget** uses the **ContextRecognizer** to identify the current status of the displayed content, and then the **ContentMatcher** component to retrieve the annotations which were previously made in this display state of the content, and display these annotations.
- The **SketchingWidget** can use the **ScreenCapture** component to retrieve a snapshot of the desired state of the screen, and use this as a background overlay to draw on [Sequence Diagram 10.1.4]
- The **CollaborativeDrawingWidget** updates the displayed drawing when other users in the collaboration session make changes to their own image.

5.2.3.4. Dynamic behavior

- The **AnnotationWidget** negotiates the permission status of a student's IWB access via their own CBC with the **Sessions::ClassroomSession** component, allows modification on the current annotation state when permission is granted, and provides the current state of the displayed content and its annotation to the **Sessions::ClassroomSession** [Sequence Diagram 10.1.6]
- The **AnnotationWidget** receives the status of the currently displayed content from the **ContentDisplay::ContentDisplay** component [Sequence Diagram 10.1.3]
- The **CollaborativeDrawingWidget** gets updated by the **Sessions::Session** component when change are made by other users in the session, and the **Session** is also notified when the user makes modifications to their own drawing [Sequence Diagram 10.1.5]

5.2.4. Sessions module

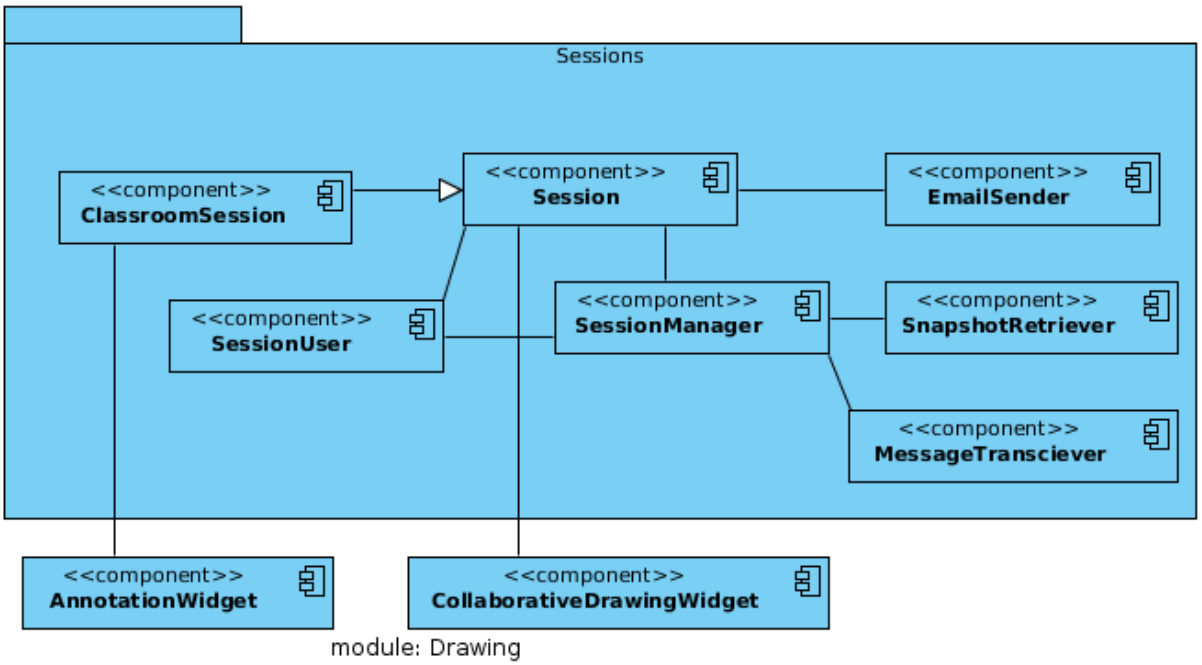


Figure 5.2.4.a: Component diagram for the Sessions module

5.2.4.1. Processing narrative

The Sessions module includes the components related to the administration and functioning of the collaborative drawing sessions. The **MessageTransciever** component handles the sending and receiving of network messages which allow the communication of IWBC and CBCs through the calls of other Sessions components. The **SessionManager** is the façade component for the Sessions module and handles the creation and administration of sessions and users, thus allowing the IWBC to function as the collaboration hub. **SessionUser**s represent the participants of **Session**s, and the **EmailSender** component is responsible for delivering e-mail copies of the collaborative drawing to the members of the session.

5.2.4.2. Interface description

No specific or constrained interfaces are defined; the **SessionManager** component is the façade component for the module and thus serves as the access interface.

5.2.4.3. Processing detail

When new users log in to the collaboration system and join or leave sessions the **SessionManager** is notified by the **MessageTransciever** component. The **SessionManager** then creates and modifies the corresponding **Session** or **SessionUser** objects as needed. When users enter or exit sessions or modify the collaborative drawing, the other CBCs in the session are notified by the **MessageTransciever** which is also capable of sending network messages. The **SnapshotRetriever** saves local copies of the current state of

all collaborative drawings when requested, and **EmailSender** sends the local state of the collaborative drawing to all session members.

5.2.4.4. Dynamic behavior

- The **Drawing::AnnotationWidget** negotiates the permission status of a student's IWB access via their own CBC with the **ClassroomSession** component, allows modification on the current annotation state when permission is granted, and provides the current state of the displayed content and its annotation to the **ClassroomSession** [Sequence Diagram 10.1.6]
- The **Drawing::CollaborativeDrawingWidget** gets updated by the **Session** component when change are made by other users in the session, and the **Session** is also notified when the user makes modifications to their own drawing [Sequence Diagram 10.1.5]

5.2.5. WiimoteInput module

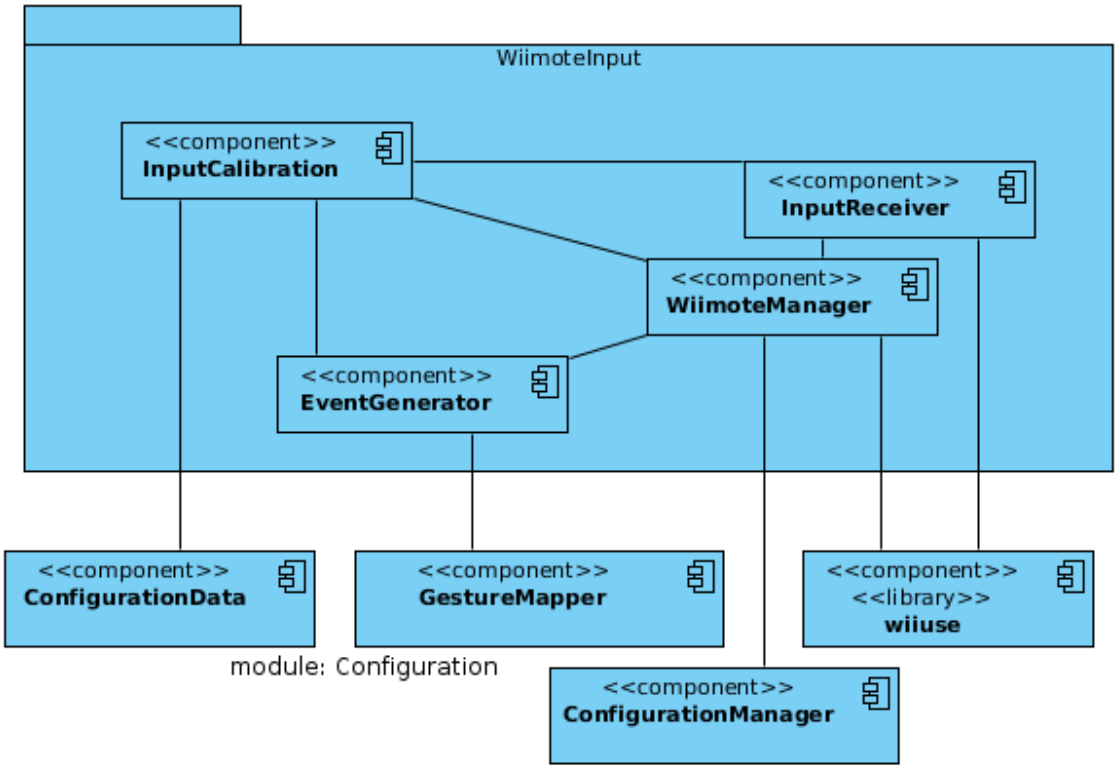


Figure 5.2.5.a: Component diagram for the WiimoteInput module

5.2.5.1. Processing narrative

The WiimoteInput module forms the core of the IR-pen input functionality of the IWBC; it communicates with the Wiimote via the **wiiuse** library to receive the changes of the IR point position via the **InputReceiver** component, which forwards this to the **InputCalibration** component that generates screen

coordinates for the IR point coordinates. The **EventGenerator** component generates the system mouse events from the received coordinates, and if movements of multiple points are involved, the corresponding multi-touch gesture events as well. The **WiimoteManager** is the façade component for the module and provides access to the functionality of the other components, as well handling the initial connection operation to the Wiimote.

5.2.5.2. Interface description

No specific or constrained interfaces are defined; the **WiimoteManager** component is the façade component for the module and thus serves as the access interface.

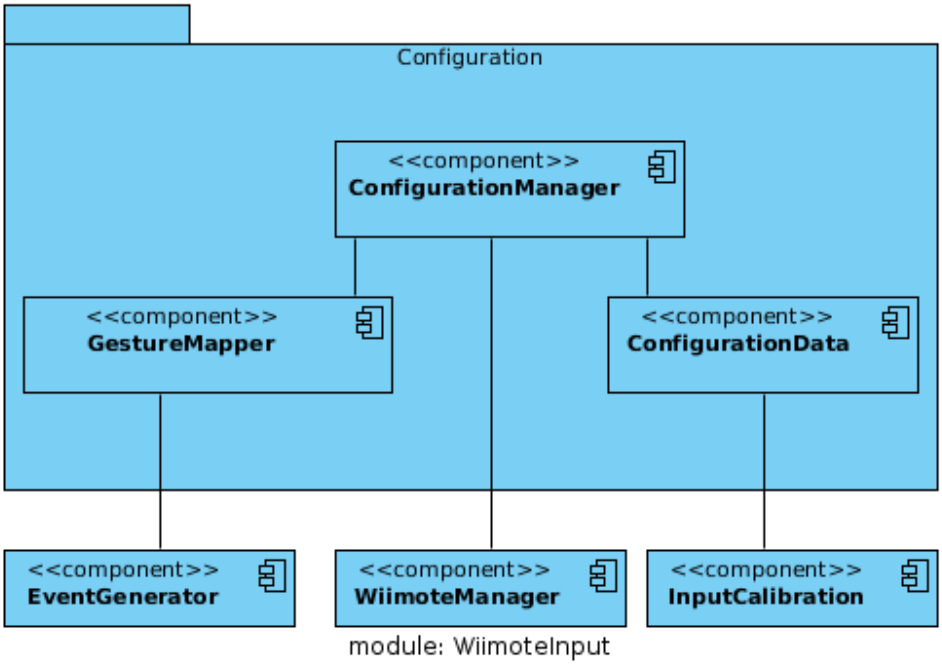
5.2.5.3. Processing detail

Initially, the **WiimoteManager** uses the **wiiuse** library to find and connect to the Wiimote. Once the connection is established, the four-point calibration data as provided by the user interface is given to the **InputCalibration** component. When the IR source changes position, the **InputReceiver** gets the current position info via a callback function from **wiiuse**, and forwards the pure IR coordinates to the **InputCalibration** component, which translates them to screen coordinates. Finally, the **EventGenerator** creates mouse events based on the information received - including multi-touch gesture events if multiple points are detected to be forming a gesture.

5.2.5.4. Dynamic behavior

- The **InputCalibration** component saves and loads calibration data using the **Configuration::ConfigurationData** component.
- The **EventGenerator** passes multi-touch coordinates to **Configuration::GestureMapper**, which recognizes any multi-touch gestures and sends info about them back to the **EventGenerator**.
- The **WiimoteManager** sends battery status and calibration info to the **Configuration::ConfigurationManager** when requested.
- The **InputReceiver** and **WiimoteManager** use the **wiiuse** library's API to access the Wiimote.
- see also: [Sequence Diagram 10.1.1] [Sequence Diagram 10.1.2]

5.2.6. Configuration module



5.2.6.a: Component diagram for the Configuration module

5.2.6.1. Processing narrative

The Configuration module is responsible for keeping the user preferences regarding the functioning of the IWBC. The **ConfigurationData** module keeps all the system configuration, including the Wiimote calibration data, screencasting preferences, and the mapping of gestures and actions. The **GestureMapper** component handles the recognition of multi-touch gestures using the Wiimote input data. The **ConfigurationManager** component is the façade component for the module and provides access to the other components; enabling the changing of configuration through its user interface, as well as displaying information on the currently connected Wiimote.

5.2.6.2. Interface description

No specific or constrained interfaces are defined; the **ConfigurationManager** component is the façade component for the module and thus serves as the access interface.

5.2.6.3. Processing detail

Through its graphical user interface, The **ConfigurationManager** displays and modifies the current configuration information stored in the **ConfigurationData**. The **GestureMapper** keeps the current mapping of gestures and actions, and performs gesture recognition and identifying the mapped action when presented with multi-touch point data.

5.2.6.4. Dynamic behavior

- The **ConfigurationData** component receives updated Wiimote calibration data from **WiimoteInput::InputCalibration** when calibration is performed.
- The **GestureMapper** recognizes any multi-touch gestures and sends info about them to the **WiimoteInput::EventGenerator** when requested.
- The **ConfigurationManager** receives Wiimote calibration and battery info from **WiimoteInput::WiimoteManager** when requested.

6. User Interface Design

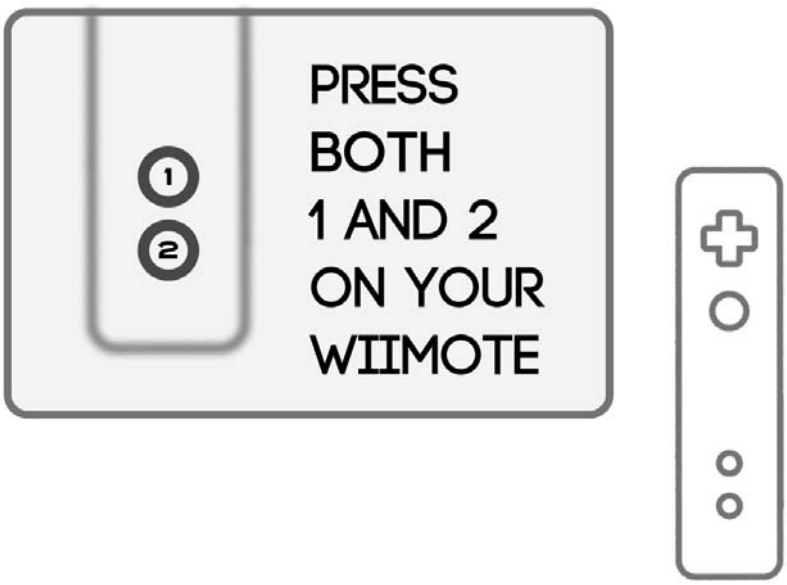
6.1. Overview of User Interface

Describe the functionality of the system from the user's perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

6.1.1. User Interfaces for the IWBC

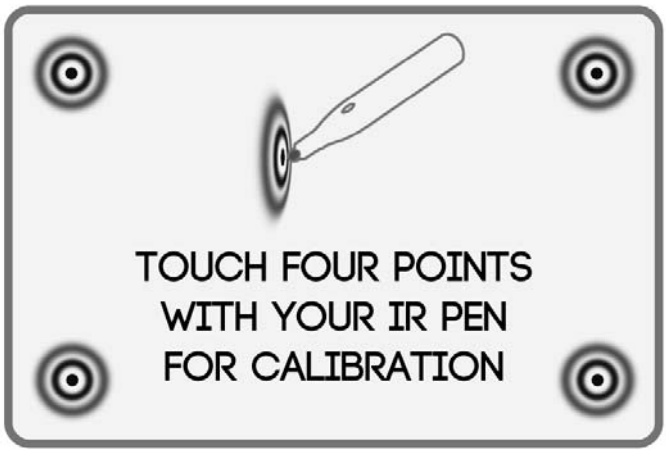
This section describes the graphical user interfaces offered by the Watch & Touch IWBC.

1. **Connect to Wiimote:** The user will be prompted to put Wiimote in the discoverable mode by simultaneously pressing 1+2 on the Wiimote, and will be informed about the progress of the connection.



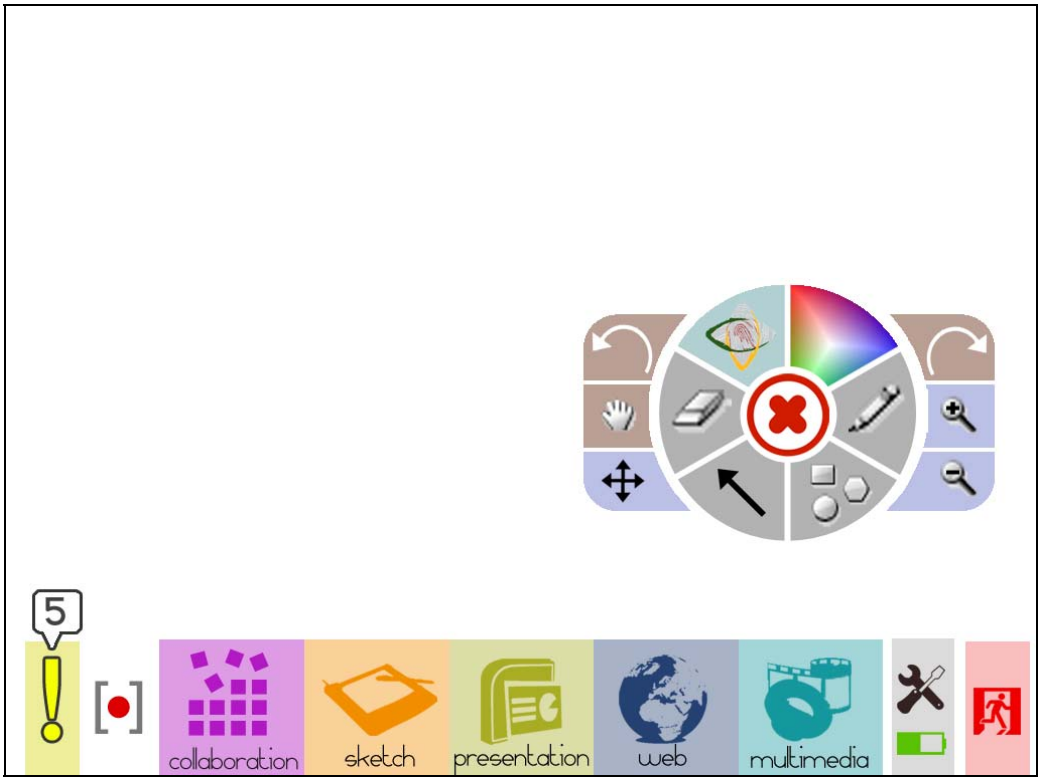
6.1.1: Connect to Wiimote

2. **Calibration:** Four circles on the four corners of the screen will be displayed, each revealing the next one as it is touched. Once all four are touched, the user will be able to test the new calibration settings on the screen by scribbling around the page, and indicating whether to keep the settings or to repeat the calibration by a button press.

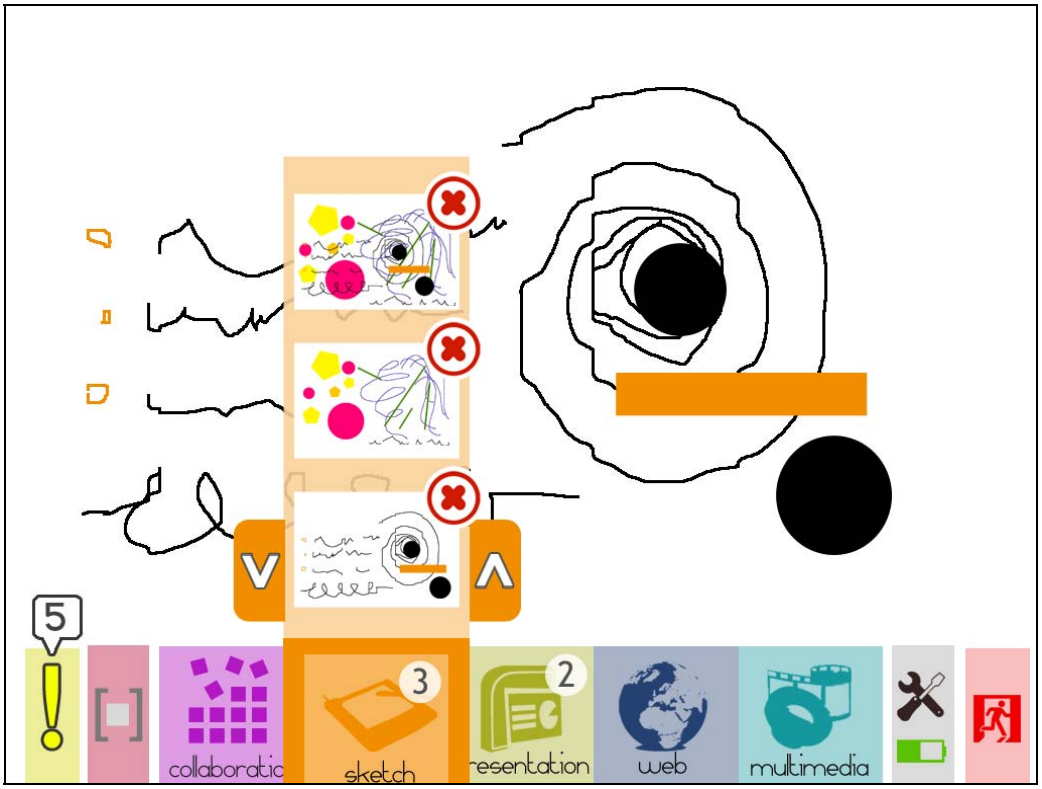


6.1.2:Calibration

3. **IWBC Main Menu:** The main menu will be following a modified version of the desktop metaphor to allow the instructor to launch the various tasks offered by the IWBC. Several independent instances of the same task can be launched for certain types of tasks, while for others only a single instance will be allowed. Only a single task can be active at the same time, all tasks take up maximal area and cannot be resized. Switching between tasks and instances, or closing them will be also done via the main menu interface. A toolbar will provide access to the functionality listed below:
    a. Display Presentation - launch new instance of presentation viewer
    b. Display Webpage - launch new instance of webpage viewer
    c. Display Video - launch new instance of video viewer
    d. Sketch - launch new instance of sketching application
    e. Collaboration - open the single-instance collaboration application
    f. Configuration - open the single-instance configuration application
    g. Exit - exit the IWBC

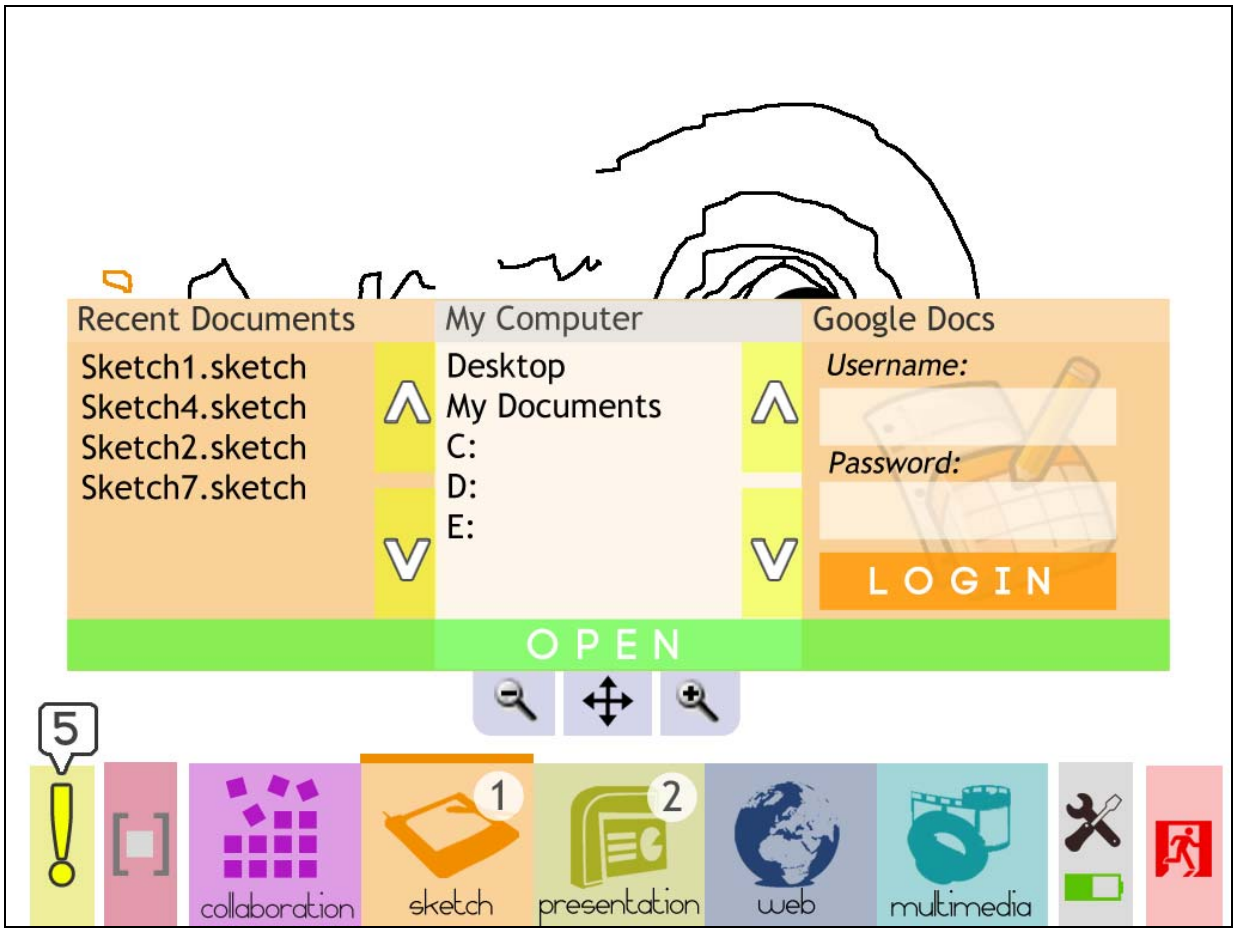


6.1.3.a: IWBC Main Menu – Annotation Menu On



6.1.3.b: IWBC Main Menu – Task Manager

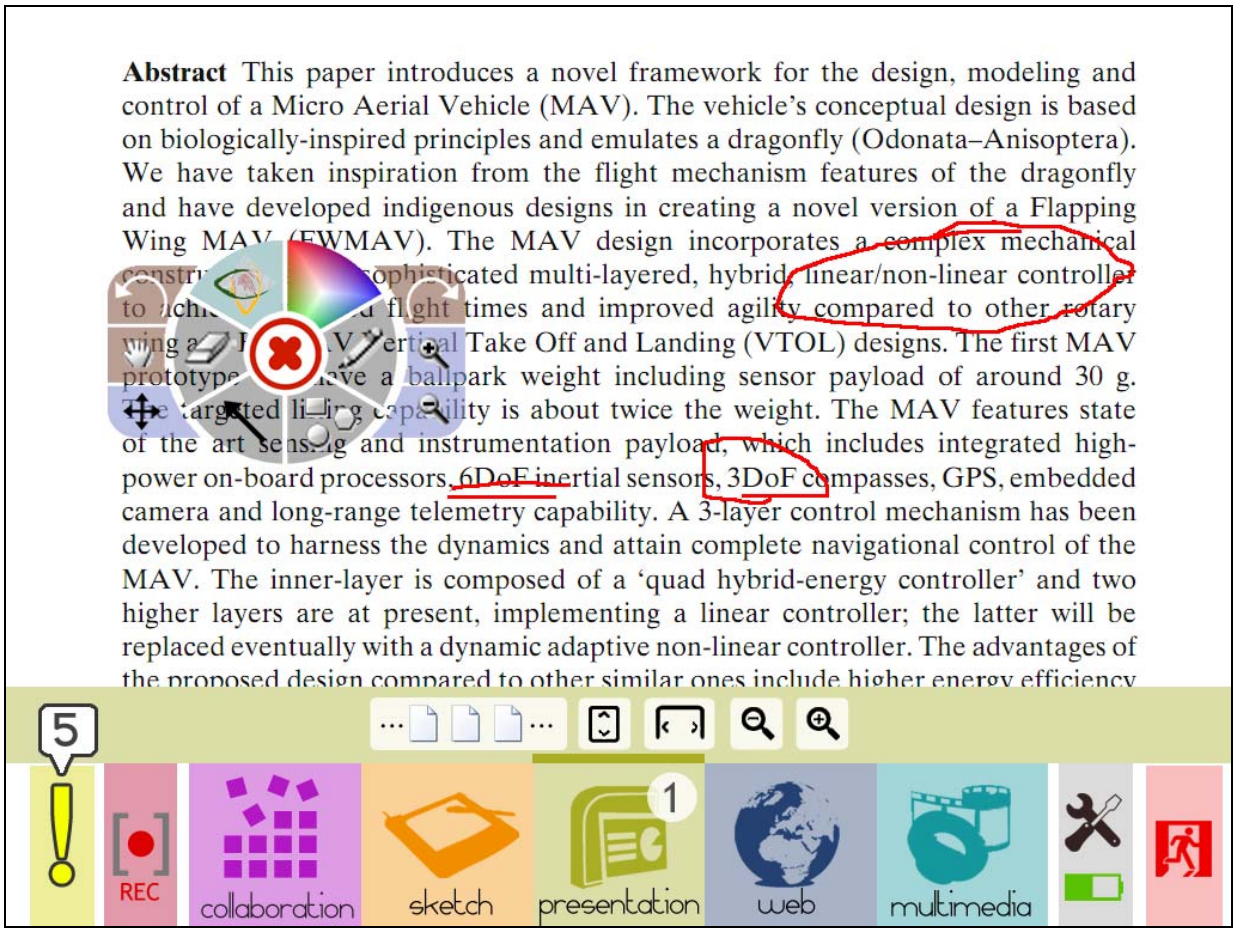
4. **File Selection:** The IWBC applications that work with user content will require the selection of a file to be displayed (presentations, webpages, videos, sketches). This interface will provide three options to make this selection: the native file selection dialogue from the operating system, a list of recently used files in this category, and an option to select the file from the instructor's Google Docs account. The user will be prompted to login to Google Docs if not already logged in. Cancelling the file selection operation to terminate the task instance and go back to the IWBC Main Menu is also possible.



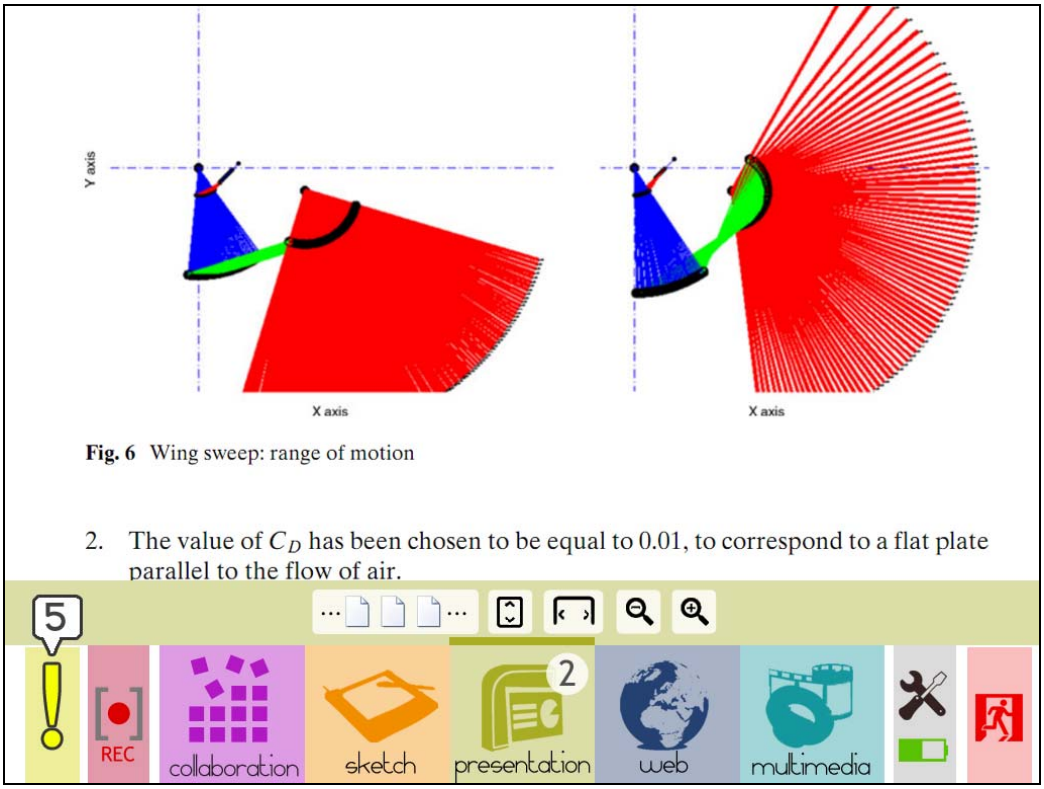
6.1.4.: File Selection

5. **Display and Annotate Content:** The main interface for displaying presentations, webpages and videos, it will be offering the standard display control interface options (next/prev buttons for presentations, a browser bar for the webpages, playback control for videos) for each kind of content. If the loaded content had been annotated before, the annotation info will appear in the previously determined location (screen position for presentations and webpages, timeline position for videos). The user will also be able to annotate the content further by simply "writing" on the content when annotation is enabled from the annotation menu, or exit the interface to go back to the IWBC main menu. The CBC may ask to be granted permission to annotate the IWB, in

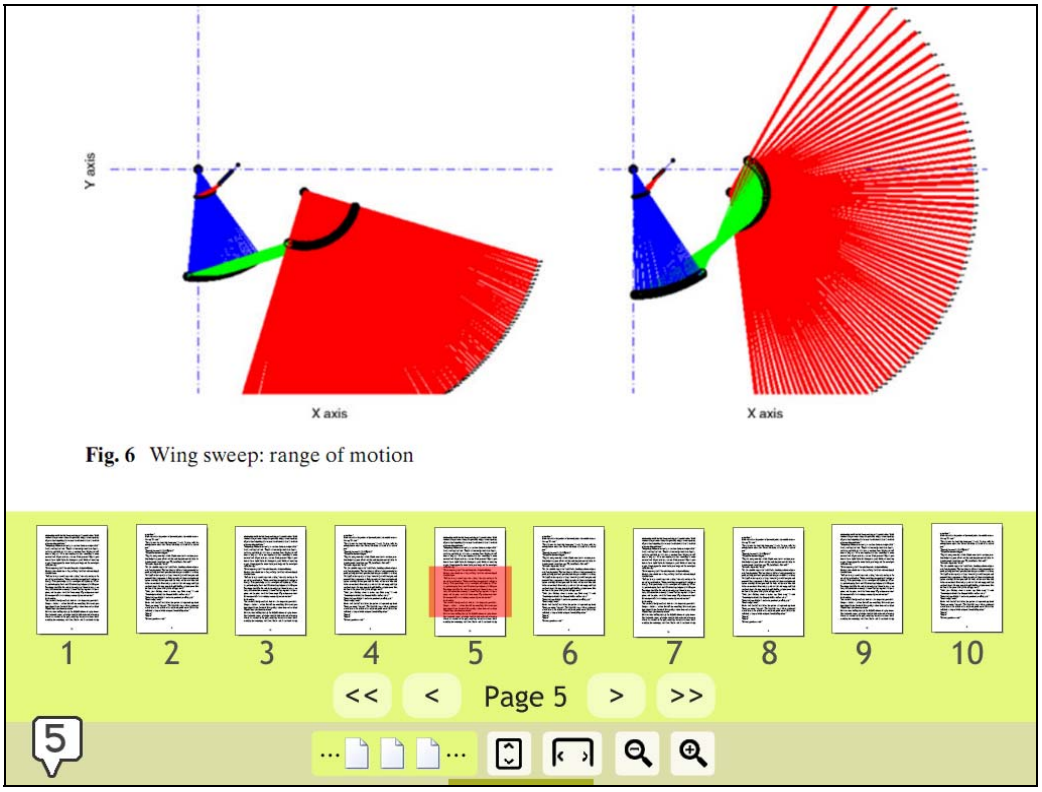
such a case a notification will be displayed on the lower left hand corner. The user can choose to deny or grant this permission. Once the permission is granted, a small window will be visible on the lower left hand corner, which can be used to revoke the permission of the CBC anytime.



6.1.5.a: IWBC Presentation

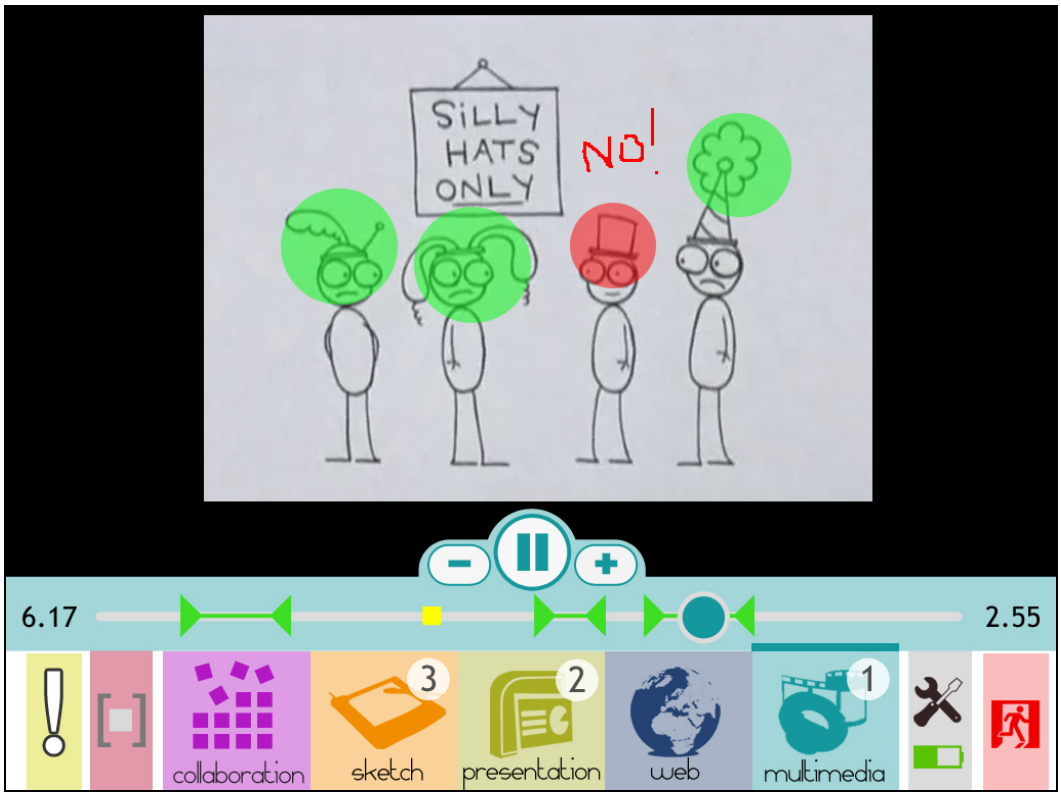


6.1.5.b: IWBC Presentation II



6.1.5.c: IWBC Presentation – Page Slider

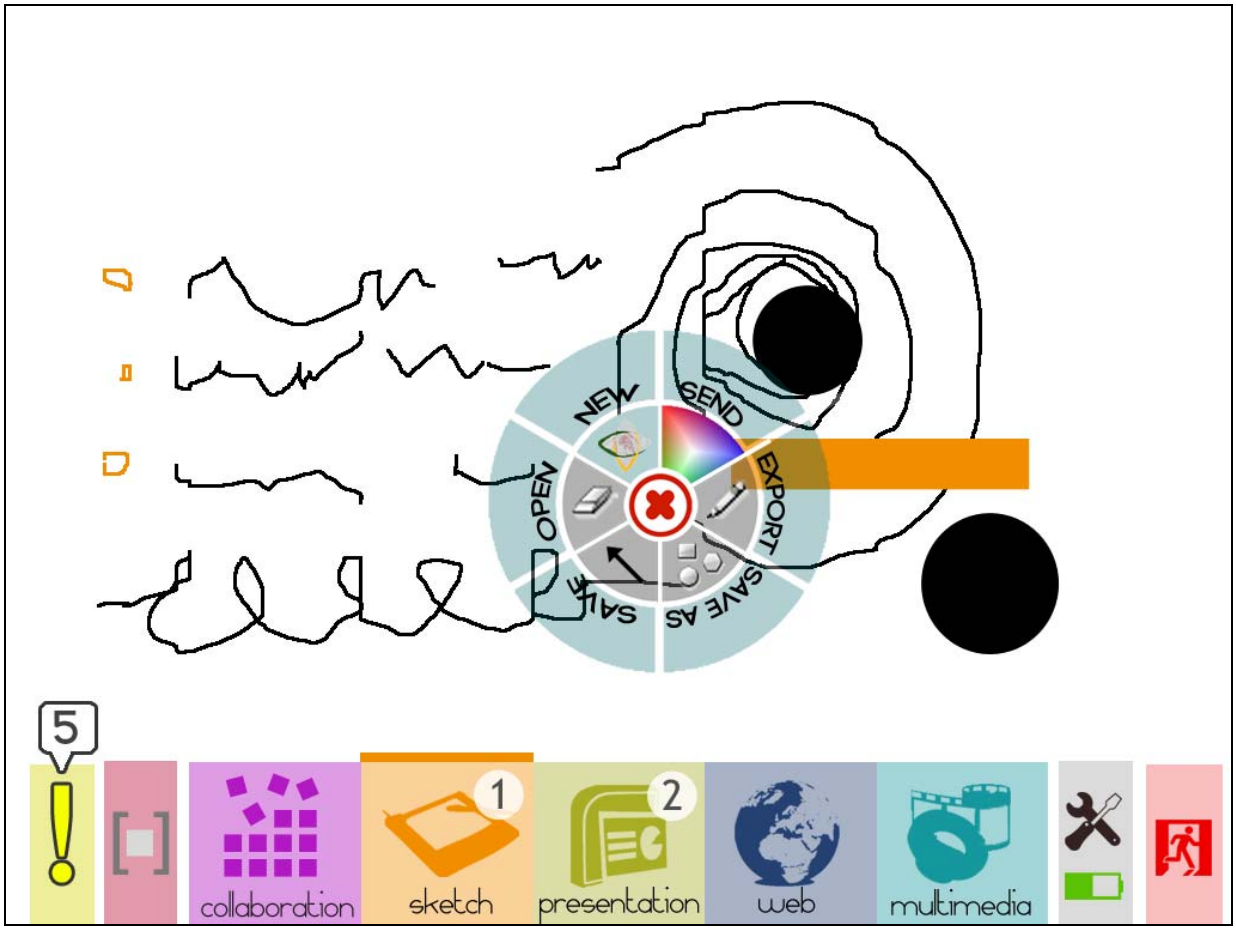
6.1.5.d: IWBC Web



6.1.5.e: IWBC Multimedia

6. **Annotation Menu:** Contains context-dependent annotation facilities: enabling/disabling annotation, changing pen style and color, eraser, exporting the annotated content, clearing all annotations and starting / stopping the screencast operation. The menu can be reached by performing the Open Context Menu multi-touch gesture determined in the Configuration section, or by long-pressing the IR pen (which is equivalent to a right click). (See 6.1.1.3 for the visual)

7. **Sketch:** Provides general drawing facilities similar to those found in painting applications. A pre-existing image can be loaded into the sketching area, and created sketches can be saved as images. Alternatively, the user can ask to work on a capture of the screen - in this case the IWBC will be minimized until the user presses the "Print Screen" key, which will capture the current screen image and bring it to the Sketch application as an underlay to be worked on.

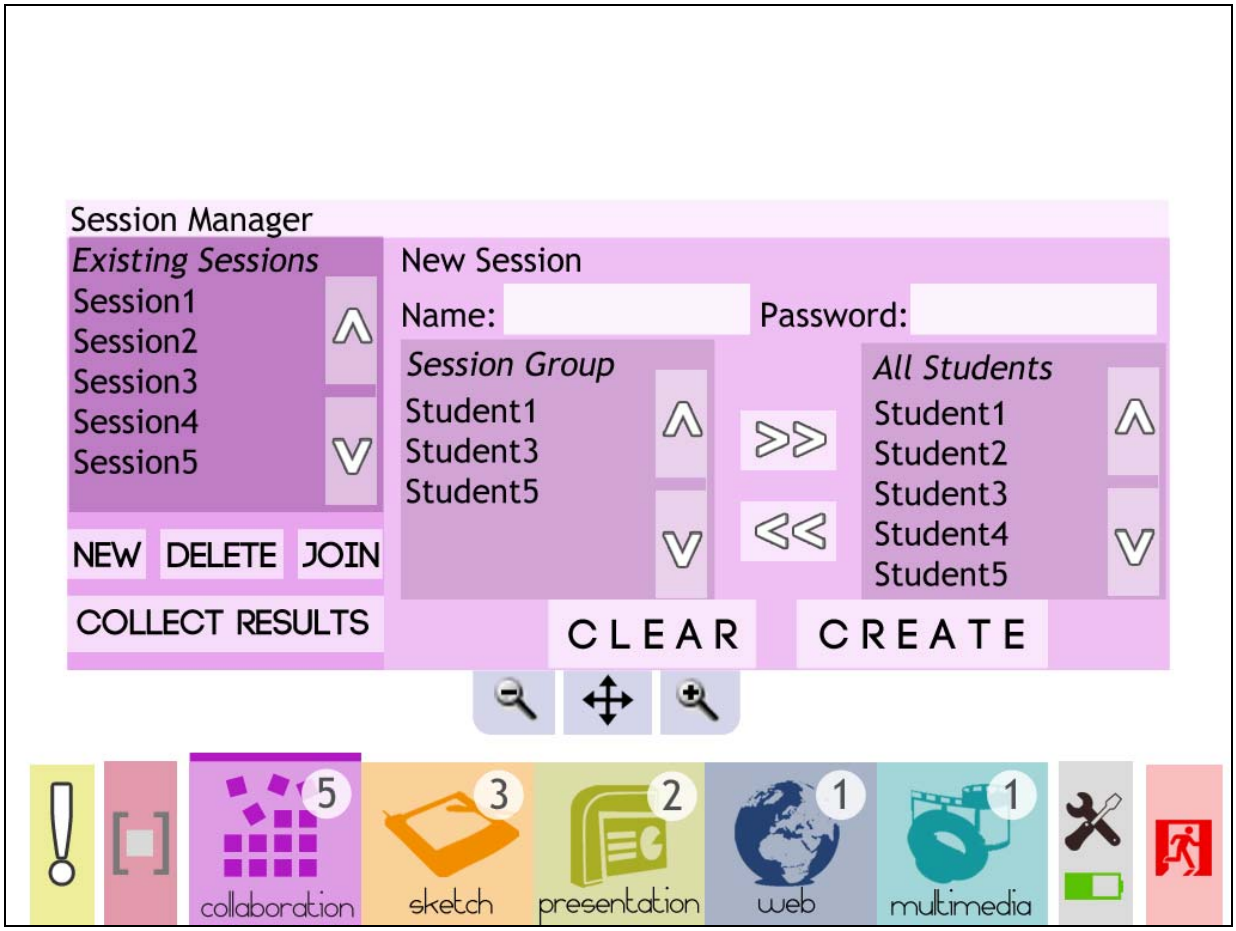


6.1.7: IWBC Sketch

8. **Sketching Menu:** Contains a variety of sketching-related tools: changing pen style and color, eraser, drawing elementary geometric shapes, saving/loading the image and capturing the screen. The menu can be reached by performing the Open Context Menu multi-touch gesture

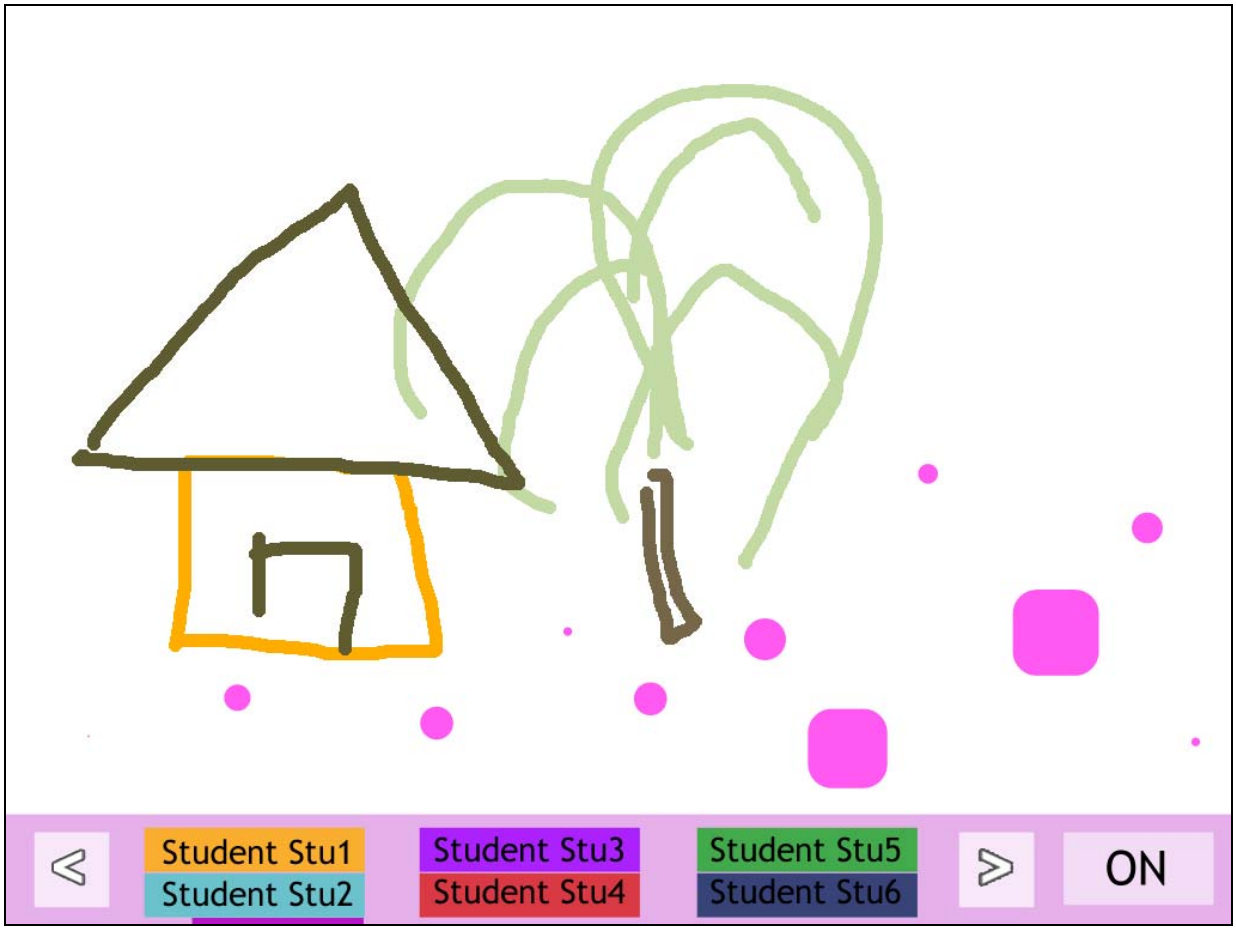
determined in the Configuration section, or by long-pressing the IR pen (which is equivalent to a right click). (See 6.1.1.3, 6.1.1.8 for the visual)

9. **Session Manager:** The portal for collaboration on the IWBC side, this interface provides the user with a list of currently existing collaborative drawing sessions. The instructor will be able to view the names of participants in each list. The user will also be able to create a new session by specifying a name and a password, or join an existing session (without providing the session password - this is a privilege of the IWBC user). Another feature accessible from this interface is the option to gather snapshots of all current collaborative drawings.



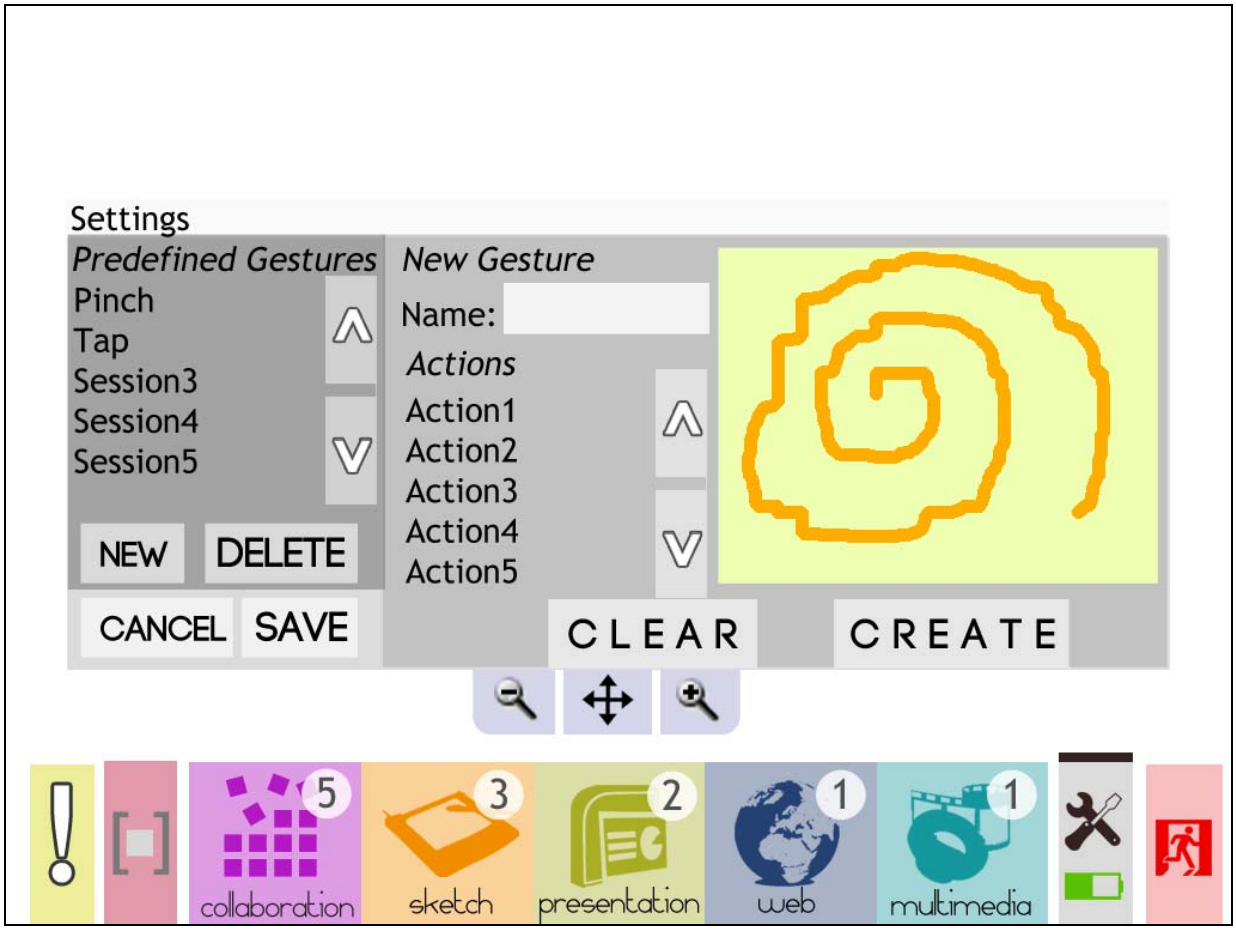
6.1.9: IWBC Session Manager

10. **Collaborative Drawing:** In this interface, the user will be able to create drawings together with the other members of the collaborative drawing session. Each user will be able to modify the drawing and see the modifications done by others as soon as the modifications are made, see the list of session members and change the currently used drawing tools, export or e-mail the current state of the drawing, or import a previously made drawing into the current session via a toolbar. Making local (not collaborative, not accessible to others) drawings is possible on a separate tab of the interface.



6.1.10: IWBC - Collaborative Drawing

11. **Configuration:** This interface will allow the user to change the mapping of predetermined multi-touch gestures to actions, re-calibrate the Wiimote and see the current battery level and calibration status of the Wiimote.



6.1.11: IWBC Configuration

6.1.2 User Interfaces for the Collaborativity Client:

1. **Login:** To be able to access the collaborative working system, the user must provide his name and surname, (optionally e-mail address and student number). This simple interface allows the user to provide these credidentals and login to the system.

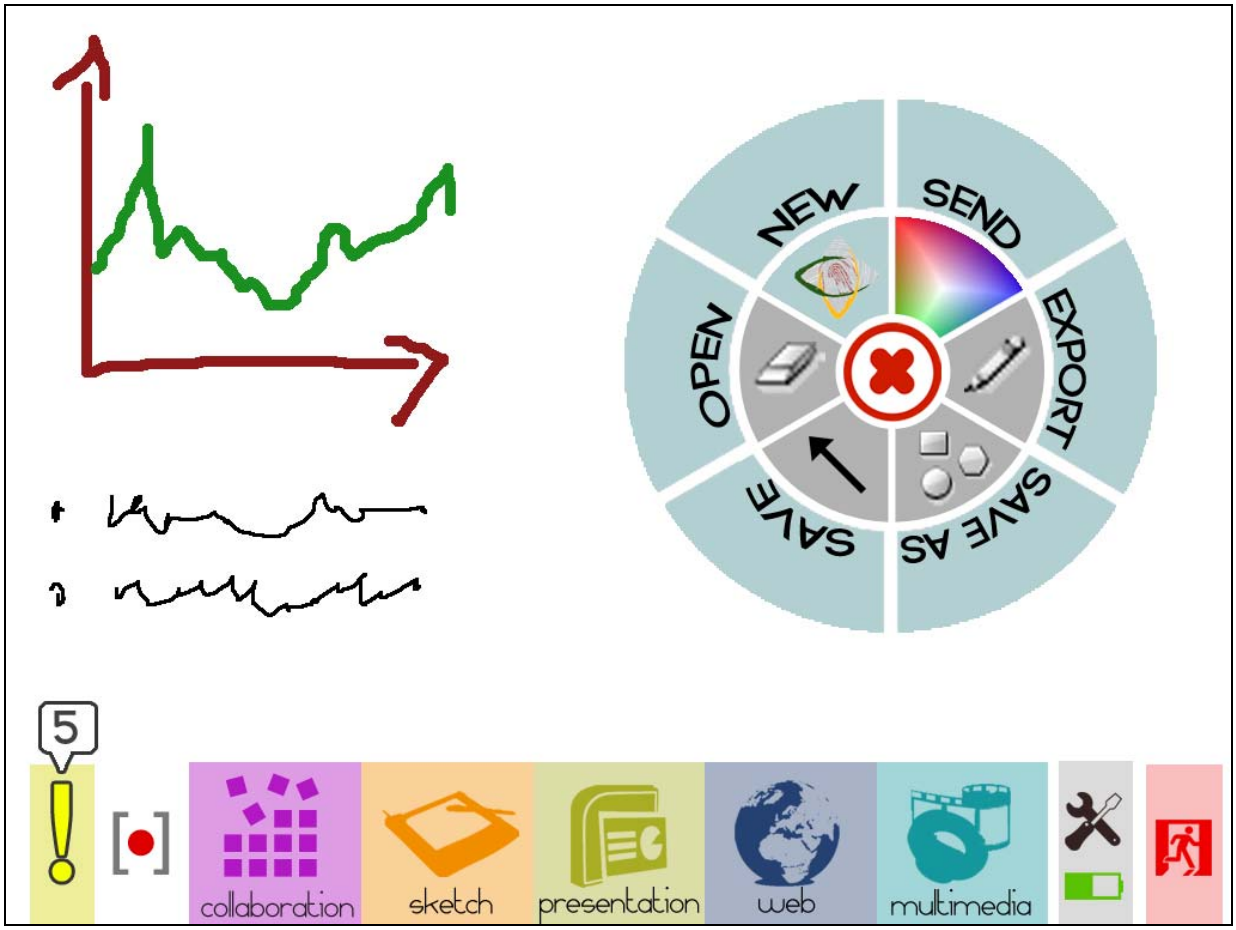
2. **CBC Main Menu:** The portal for collaboration on the CBC side, this interface provides the user with a list of currently existing collaborative drawing sessions. The user will also be able to create a new session by specifying a name and a password, or join an existing session by providing the session password. On the CBC side, the user is able to join the special Classroom Session directly from this interface (no password required).

3. **Collaborative Drawing:** *identical to the IWBC Collaborative Drawing interface, see section 3.1.1.10 for description*

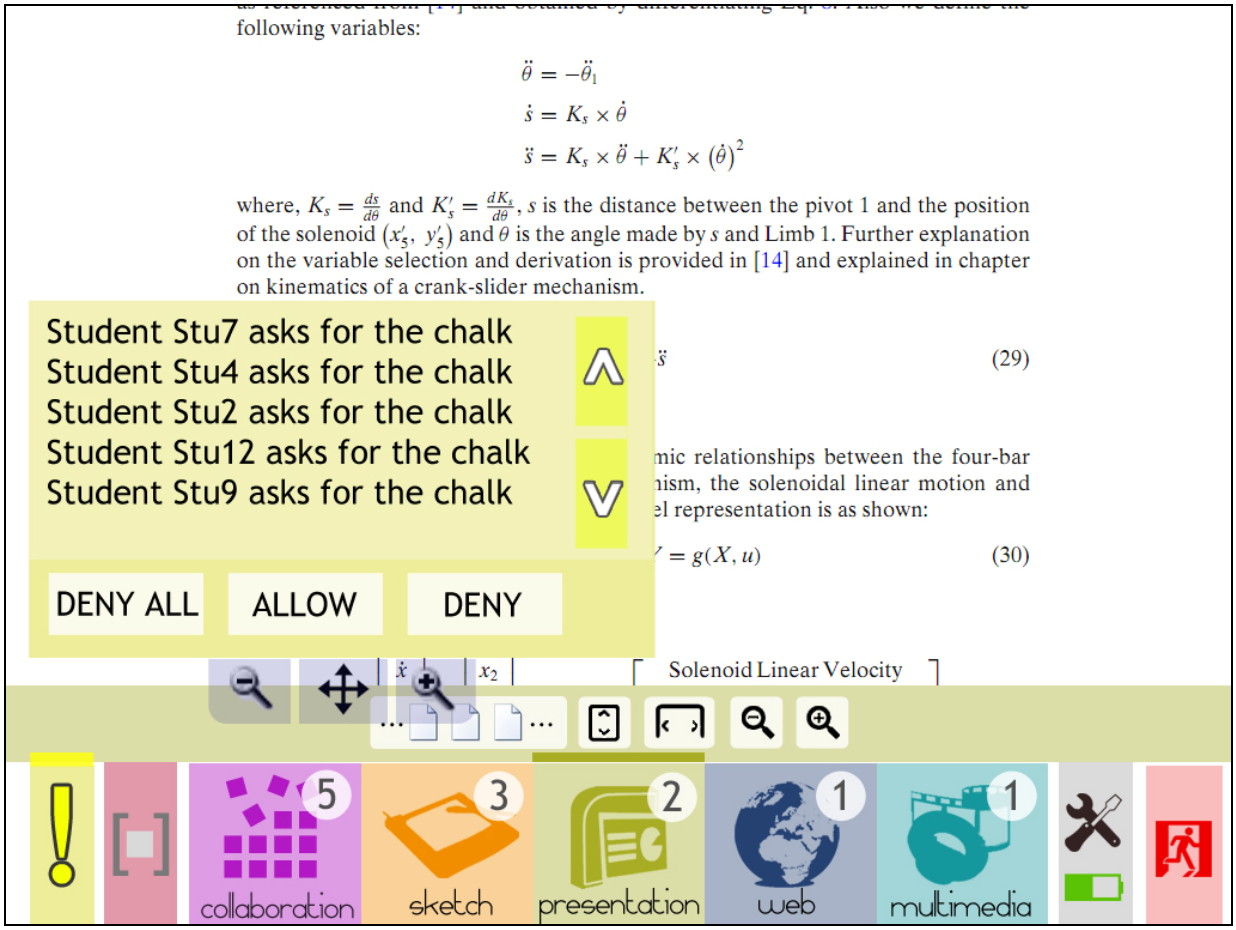
4. **Classroom Session:** This interface will be identical to the Collaborative Drawing interface in appearance, with some extra features. The background of the drawing will always reflect the

current content displayed by the IWB, and the drawings that the CBC user makes on top will not be reflected on the IWB unless the CBC asks for special permission via a button on the toolbar.

6.2. Screen Images



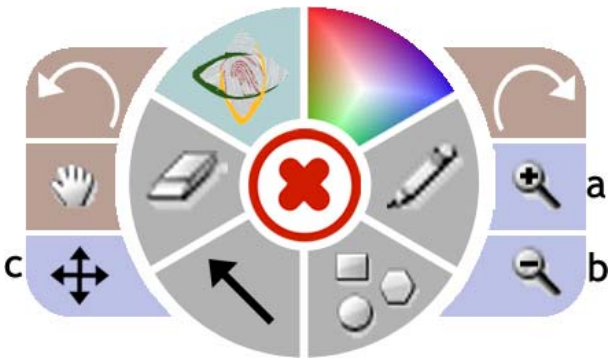
6.2.a: IWBC – Annotation Menu, File Operations



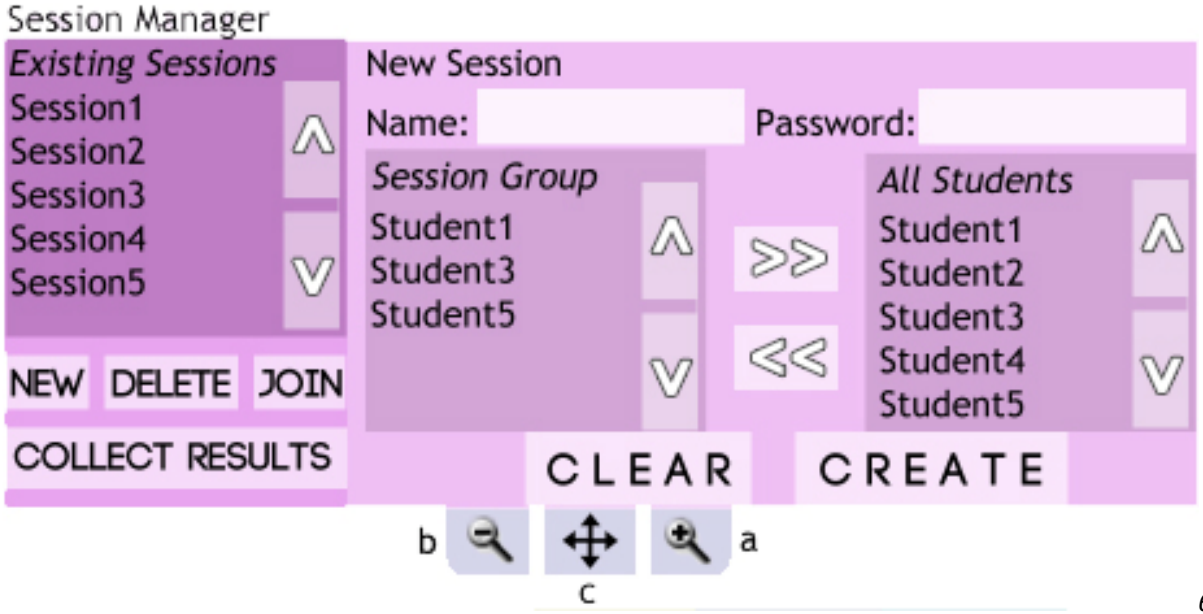
6.2.b: IWBC - Notifications

6.3. Screen Objects and Actions

1. **Menu Size and Menu Location Manipulation:**



6.3.1.a: Annotation Menu



6.

3.1.b: Session Manager Menu

User can both zoom-in, zoom-out menu and changes its location on the screen if it has a small light-blue submenu.

a. Zoom-In menu
b. Zoom-Out menu
c. Change location of the menu

2. Multimedia Control Panel

3.1.2: Session Manager Menu

a. Continuous multimedia annotation
While the time indicator is on the green pieces, a continuous annotation is shown along with the multimedia content.

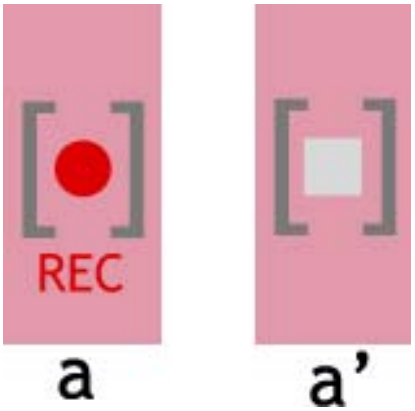
b. Point multimedia annotation
When the time indicator is on the yellow piece, a instantaneous annotation is shown for a while along with the multimedia content.

c. Time Indicator and Manipulator
A small circle indicates the

d. **Decrease time scale**
   By pressing ‘d’, one zooms out the timeline.

e. **Increase time scale**
   By pressing ‘e’, one zooms in the timeline.

3. **Screencasting Indicator**



3.1.3: Session Manager Menu

If screen casting is started, ‘a’ is shown on the bottom left side of the screen. It turns to a’ if screen casting is stopped.

7. Libraries and Tools

Watch & Touch will make use of several existing software libraries and tools, both during development to achieve a faster production cycle and during runtime to obtain extra functionality. These libraries and tools are described in the following subsections.

7.1 Qt

7.1.1. Description

Qt is a cross-platform application framework that is widely used for developing application software with graphical user interface (GUI) (in which case Qt is referred to as a widget toolkit when used as such), and also used for developing non-GUI programs such as command-line tools and consoles for servers. Qt is most notably used in Autodesk, Google Earth, KDE, Adobe Photoshop Album, the European Space Agency, OPIE, Skype, VLC media player, Samsung, Philips, Panasonic and VirtualBox. It is produced by Nokia's Qt Development Frameworks division, which came into being after Nokia's acquisition of the Norwegian company Trolltech, the original producer of Qt. Qt uses standard C++ but makes extensive use

of a special code generator (called the Meta Object Compiler, or moc) together with several macros to enrich the language. Qt can also be used in several other programming languages via language bindings. It runs on all major platforms and has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, thread management, network support, and a unified cross-platform API for file handling. Distributed under the terms of the GNU Lesser General Public License (among others), Qt is free and open source software. All editions support a wide range of compilers, including the GCC C++ compiler and the Visual Studio suite.

7.1.2. Usage in Watch & Touch

By the nature of its primary feature set (annotation, multitouch gesture manipulation, collaborative drawing) Watch & Touch is quite user interface centric. This heavy emphasis on graphical user interfaces makes their ease of construction a primary factor in design. Its feature-rich and diverse library of UI elements (including a multimedia framework for viewing videos, an HTML renderer for webpages and SVG support), multitouch gesture capabilities and the Qt Creator IDE make Qt an excellent choice of framework for Watch & Touch. Another important factor is the framework's cross-platform support; with Qt, building applications for a range of operating systems is often a simple matter of recompiling.

7.2 wiiuse

7.2.1. Description

Wiiuse is a library written in C that connects with several Nintendo Wii remotes. Supports motion sensing, IR tracking, nunchuk, classic controller, and the Guitar Hero 3 controller. Single threaded and nonblocking makes a light weight and clean API.

7.2.2. Usage in Watch & Touch

The wiiuse library is used in Watch & Touch as a layer of abstraction between the IWBC and the Wiimote; it handles the Bluetooth communication between the Wiimote and the instructor's machine and exposes a number of methods for accessing the Wiimote's functionality without going into the details of the communication protocol.

7.3 Google Documents API

7.3.1. Description

The Google Documents List API allows client applications to programmatically access and manipulate user data stored with Google Documents. Here are some of the things you can do with the API:

- Discovery: Retrieve documents that match specific keywords, categories, or metadata.
- Download: export documents in common formats such as pdf, rtf, doc, xls, ppt, and more.

- Sharing (ACLs): Modify the sharing permissions of documents and folders. The API allows sharing to individuals, group emails, or across an entire Google Apps domain.
- Create/upload/copy documents: Create online backups of local word processor documents, spreadsheets, presentations, and PDFs.
- Revisions: Review, download, or publish a document's complete revision history.
- File documents: Create folders and move documents/folders in and out of folders.
- Spreadsheets: While the Documents List API can be used to create and retrieve a list of Google Spreadsheets, it cannot be used to modify the data within a Spreadsheet. For that, you can use the Google Spreadsheets API.

7.3.2. Usage in Watch & Touch

In the Watch & Touch IWBC, the Google Documents List API is used to provide an alternative input source for presentations - instead of using local files, the user can provide Google Account login details to access the Google Docs presentations stored in this account. The ClientLogin authorization method is used to gain access to the account, followed by API calls that retrieve the list of presentations and download a given presentation in the PDF format. The details of the API are abstracted from the rest of the application by the GoogleDocsAccess component, which provides direct methods for logging in, getting the list of presentations and downloading a given presentation as PDF.

7.4 Screencasting tools

7.4.1. Description

A screencast is a digital recording of computer screen output, also known as a video screen capture, often containing audio narration. The term screencast compares with the related term screenshot; whereas screenshot is a picture of a computer screen, a screencast is essentially a movie of the changes over time that a user sees on a computer screen, enhanced with audio narration. Screencasting tools are the software tools that allow the recording of such screen activity videos.

7.4.2. Usage in Watch & Touch

A variety of screencasting applications[5] exist; however, keeping in mind that Watch & Touch aims to be open-source and cross-platform compatible, programs which fulfill these two constraints would be more fitting, so the two suitable candidates have been determined as VLC and FFmpeg. The user is able to specify which of these tools will be used during the screencasting operation in the IWBC Configuration screen.

7.5. Poppler

7.5.1. Description

Poppler (or libpoppler) is a free software library used to render PDF documents. It is used by the PDF viewers of the open source GNOME and KDE desktop environments, and its development is supported by freedesktop.org. The project was started by Kristian Høgsberg with two goals in mind: To provide PDF rendering functionality as a shared library, in order to centralize maintenance effort, and to go beyond the goals of Xpdf, and integrate with functionality provided by modern operating systems. Poppler itself is a fork of the Xpdf-3.0 PDF viewer developed by Derek Noonburg of Glyph and Cog, LLC.

7.5.2. Usage in Watch & Touch

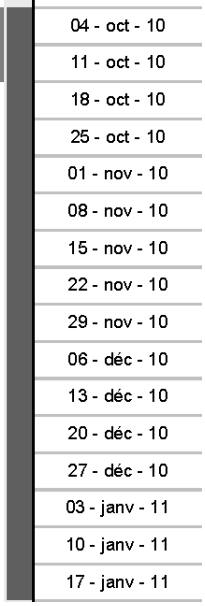
Watch & Touch uses the libpoppler Qt4 interface, which allows rendering of PDF files directly ath the desired resolution (as well as numerous other operations) from inside Qt to display PDF content. The following code segment illustrates how easily this can be accomplished:

```
Poppler::Document* document = Poppler::Document::load(filename);
Poppler::Page* pdfPage = document->page(pageNumber);

// Generate a QImage of the rendered page
QImage image = pdfPage->renderToImage(xres, yres, x, y, width, height);

// ... use image …
```

8. Time Planning (Gannt Chart)

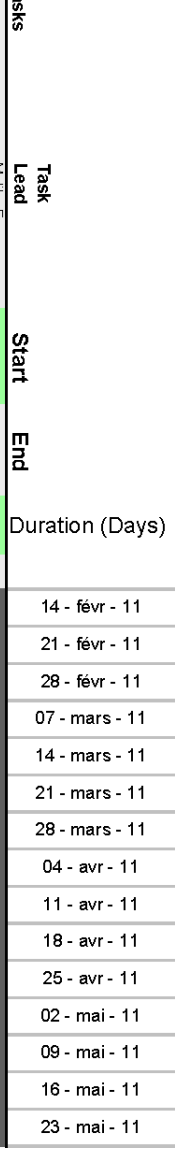
8.1. Term 1 Gannt Chart

WBS	Tasks	Task Lead	Start	End	Duration (Days)
1	**Project Management**	Melike Ercan	10/04/10	1/23/11	112
1.1	Proposal Report		10/04/10	10/17/10	14
1.2	Requirements Analysis Report		10/22/10	12/09/10	49
1.3	Initial Design Report		12/07/10	12/27/10	21
1.4	Final Design Report		1/02/11	1/22/11	21
1.5	Web Page		10/04/10	1/23/11	112
2	**Development Tools Analysis**	Ulku Şirin Yaman Umuroğlu	11/22/10	12/19/10	28
2.1	Qt Framework Analysis		11/25/10	12/08/10	14
2.2	Wiimote Communication Analysis		12/05/10	12/11/10	7
2.3	Google Docs API Analysis		12/06/10	12/12/10	7
2.4	Multitouch Framework Analysis		12/13/10	12/19/10	7
3	**Hardware Aspects Analysis**	Yaman Umuroğlu	11/29/10	12/26/10	28
3.1	Design of IR pen		12/01/10	12/07/10	7
3.2	Wiimote holder		12/08/10	12/14/10	7
3.3	Environment testing		12/15/10	12/21/10	7
3.4	Design of IR rings		12/20/10	12/26/10	7
4	**Component Design**	Melike Ercan Ulku Şirin Yaman Umuroğlu	12/06/10	1/23/11	49
4.1	Annotation and Sketching Component		12/06/10	1/23/11	49
4.2	Session Management Component		12/13/10	1/23/11	42
4.3	Collaborative Design Component		1/03/11	1/23/11	21
4.4	Video Display Component		12/27/10	1/23/11	28
4.5	Web Display Component		1/03/11	1/23/11	21
4.6	Presentation Display Component		1/10/11	1/23/11	14
5	**Interface Design**	Giray Havur Yaman Umuroğlu	12/06/10	1/23/11	49
5.1	Brand Design		12/06/10	12/19/10	14
5.2	Graphical Components		12/20/10	1/23/11	35
5.3	Human Factors and Ergonomics Study		12/13/10	12/26/10	14
5.4	Main Menu Design		12/27/10	1/23/11	28
5.5	Annotation Menu Design		12/20/10	1/23/11	35
5.6	Navigation Specification		12/27/10	1/23/11	28
5.7	Collaboration Interaction		1/03/11	1/23/11	21



8.2. Term 2 Gannt Chart

WBS	Tasks	Task Lead	Start	End	Duration (Days)
1	**Project Management**	Melike Ercan	2/14/11	6/12/11	119
1.1	Web Page		2/14/11	6/12/11	119
2	**Component Design**	Melike Ercan Ulku Şirin Yaman Unuroğlu	2/14/11	5/08/11	84
2.1	Annotation and Sketching Component		2/14/11	4/03/11	49
2.2	Session Management Component		2/14/11	2/27/11	14
2.3	Collaborative Design Component		2/14/11	5/08/11	84
2.4	Video Display Component		2/14/11	4/10/11	56
2.5	Web Display Component		2/14/11	4/17/11	63
2.6	Presentation Display Component		2/14/11	4/24/11	70
3	**Component Implementation**	Melike Ercan Ulku Şirin Giray Havur Yaman Unuroğlu	2/14/11	5/29/11	105
3.1	Annotation and Sketching Component		2/14/11	3/20/11	35
3.2	Session Management Component		3/21/11	3/27/11	7
3.3	Collaborative Design Component		3/28/11	4/17/11	21
3.4	Video Display Component		4/18/11	4/24/11	7
3.5	Web Display Component		4/25/11	5/01/11	7
3.6	Presentation Display Component		5/02/11	5/08/11	7
3.7	Integration		5/09/11	5/29/11	21
4	**Interface Design**	Giray Havur Yaman Unuroğlu	2/14/11	3/13/11	28
4.1	Graphical Components		2/14/11	2/27/11	14
4.2	Annotation Menu Design		2/14/11	3/06/11	21
4.3	Navigation Specification		2/14/11	3/13/11	28
4.4	Collaboration Interaction		2/14/11	2/27/11	14
5	**Finalization and Testing**	Melike Ercan Ulku Şirin Giray Havur Yaman Unuroğlu	5/30/11	6/12/11	14
5.1	User Manual Preparation		5/30/11	6/12/11	14
5.2	Testing		5/30/11	6/12/11	14
5.3	Final Presentation		5/30/11	6/05/11	7



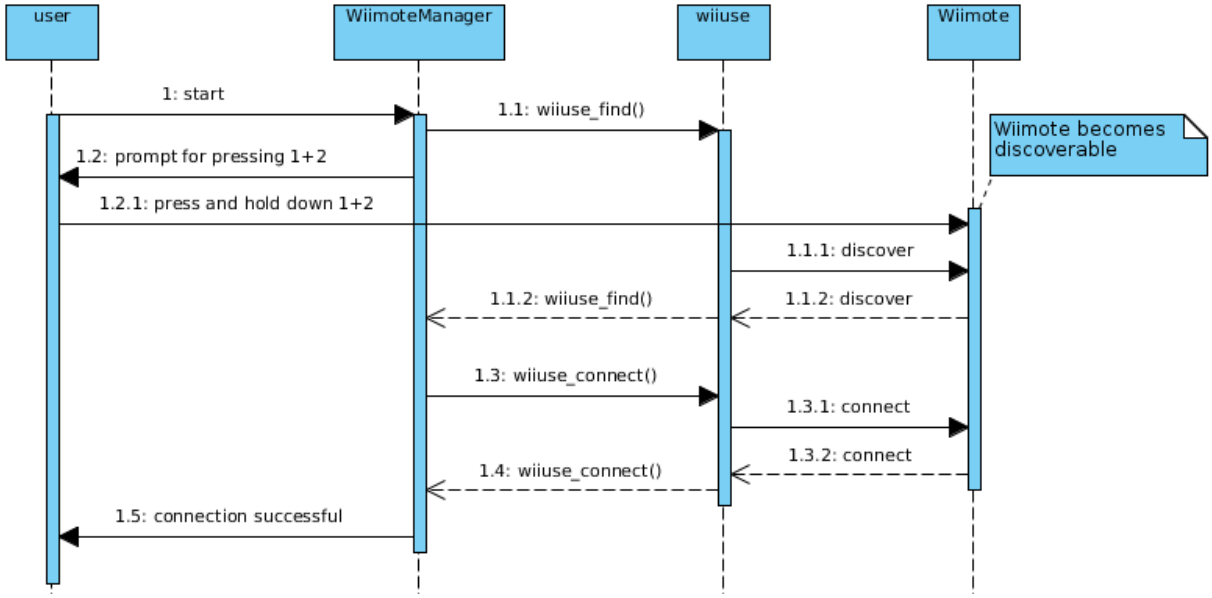
9. Conclusion

This document states the design level approach taken by DialecTech team for the Watch & Touch project. In this document, a fair amount of elaboration has been done on the project scenario pointing out most of the important details. The goals for the final product has become more apparent as the scenario and the desired user interface is visually explained. Additionally,
this document is the first document that explains somewhat deep technical details. The architecture of the system is discussed with an overview, and illustrated with component diagrams. Further information on the technical design is given with detailed explanations of the modules which are supported with and sequence diagrams. Finally, the progress made by the project team has been summarized.

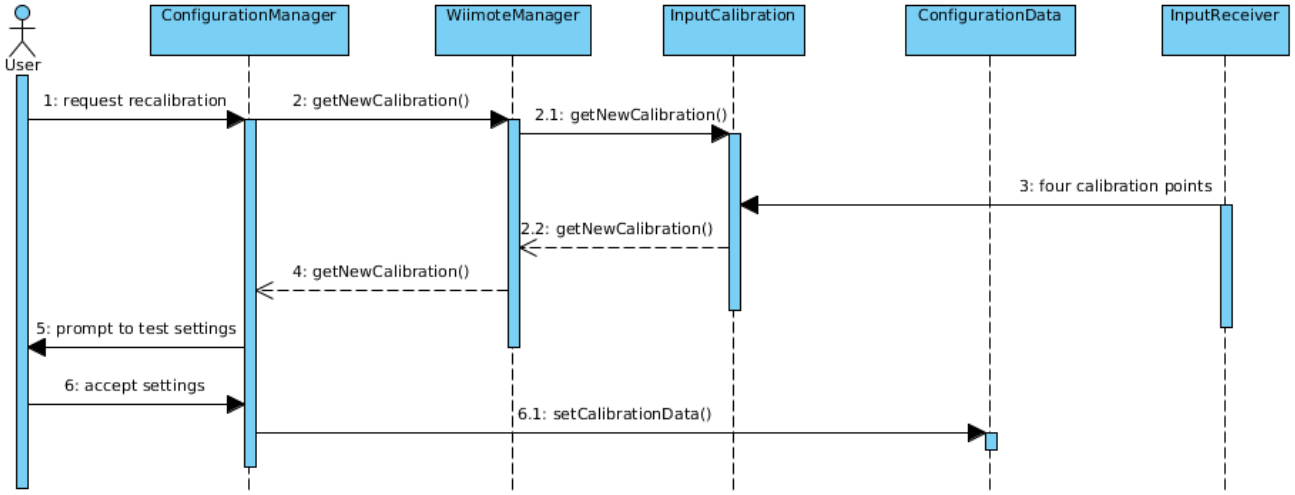
10. Appendix

10.1. Sequence Diagrams

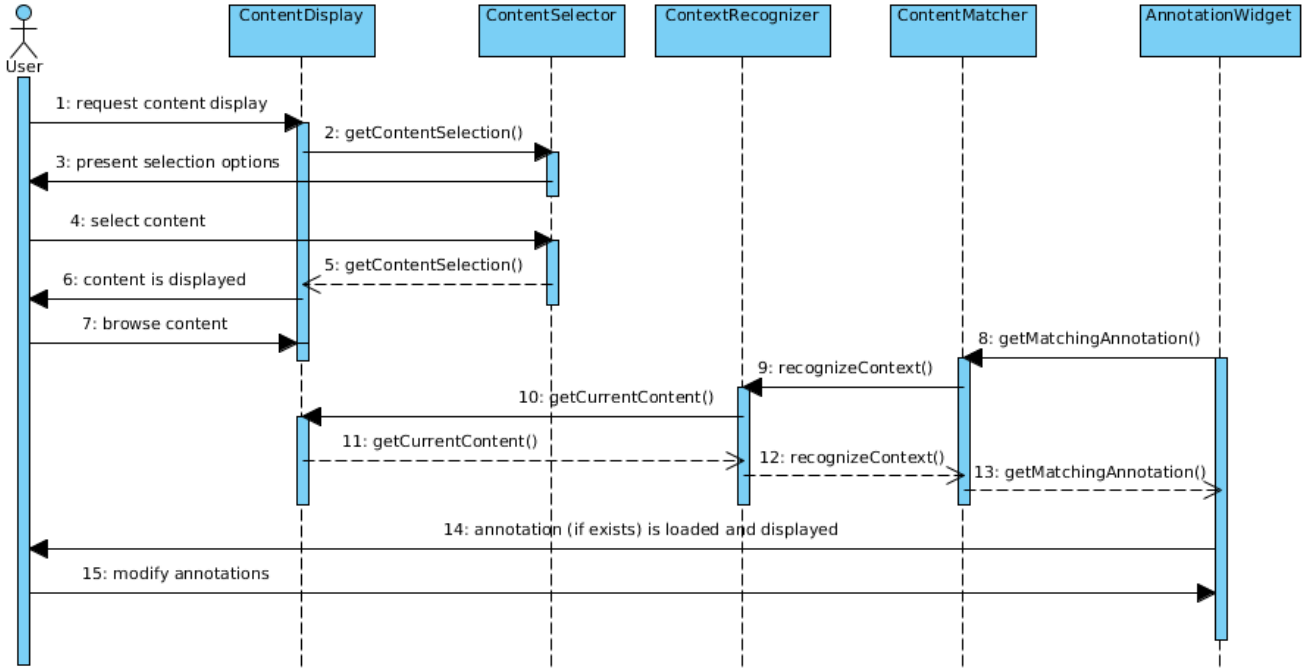
10.1.1. Wiimote Initialization sequence diagram



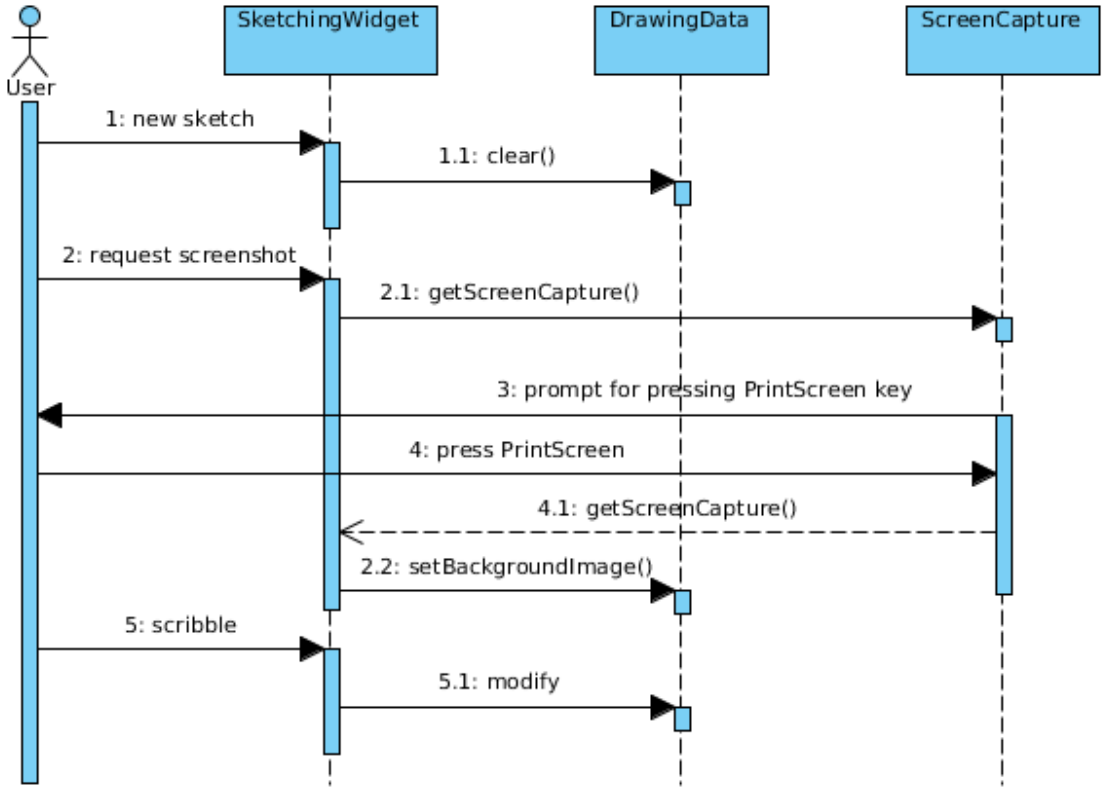
10.1.2. Wiimote Calibration sequence diagram



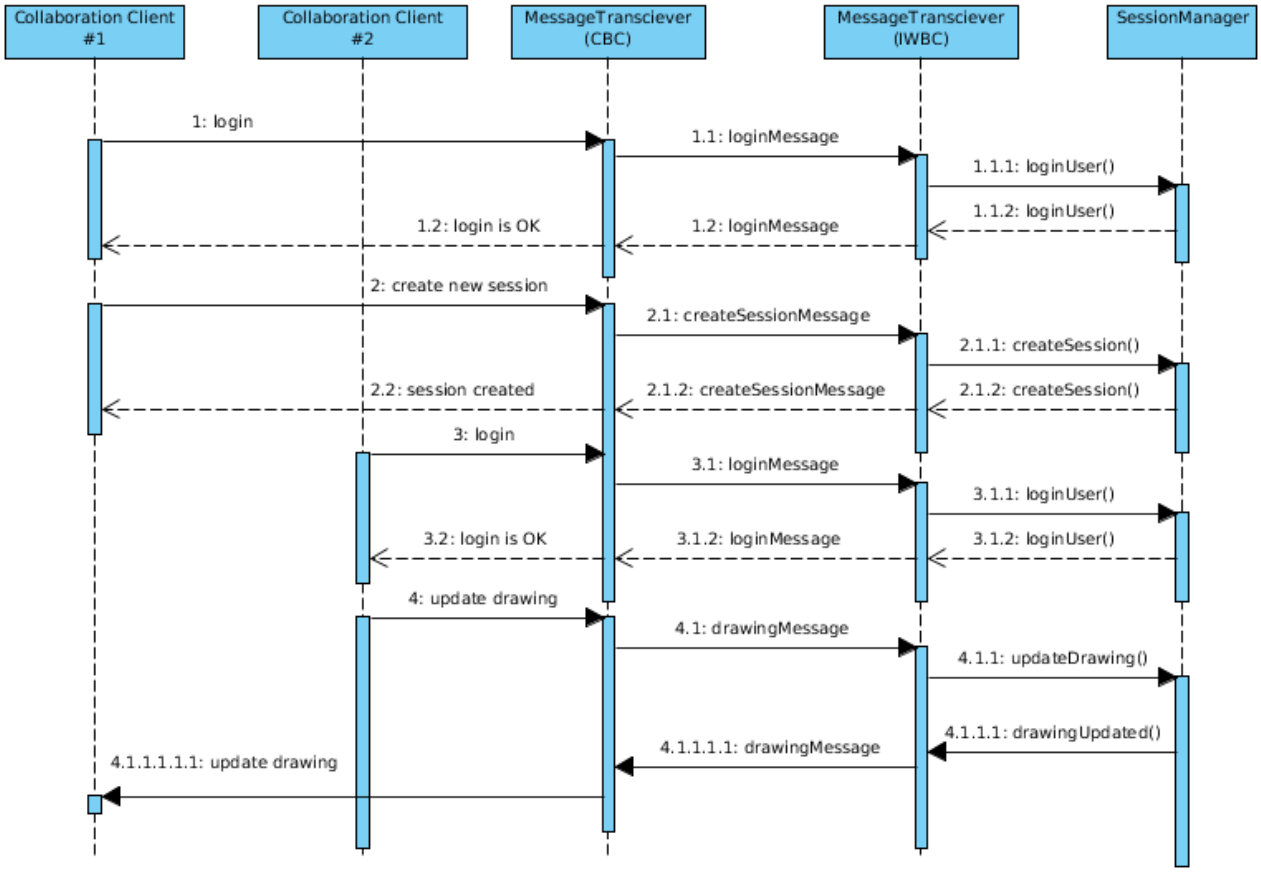
10.1.3. Content Display and Annotation sequence diagram



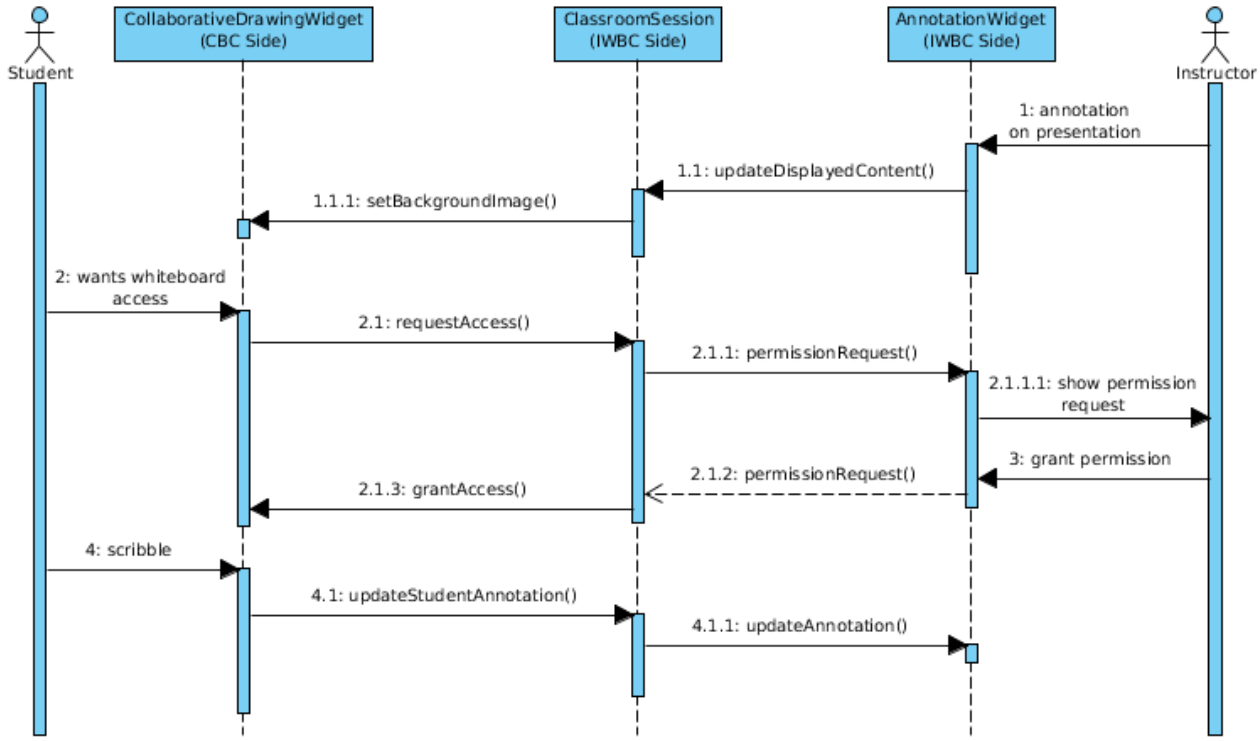
10.1.4. Sketching sequence diagram



10.1.5. Collaborative Drawing overview sequence diagram

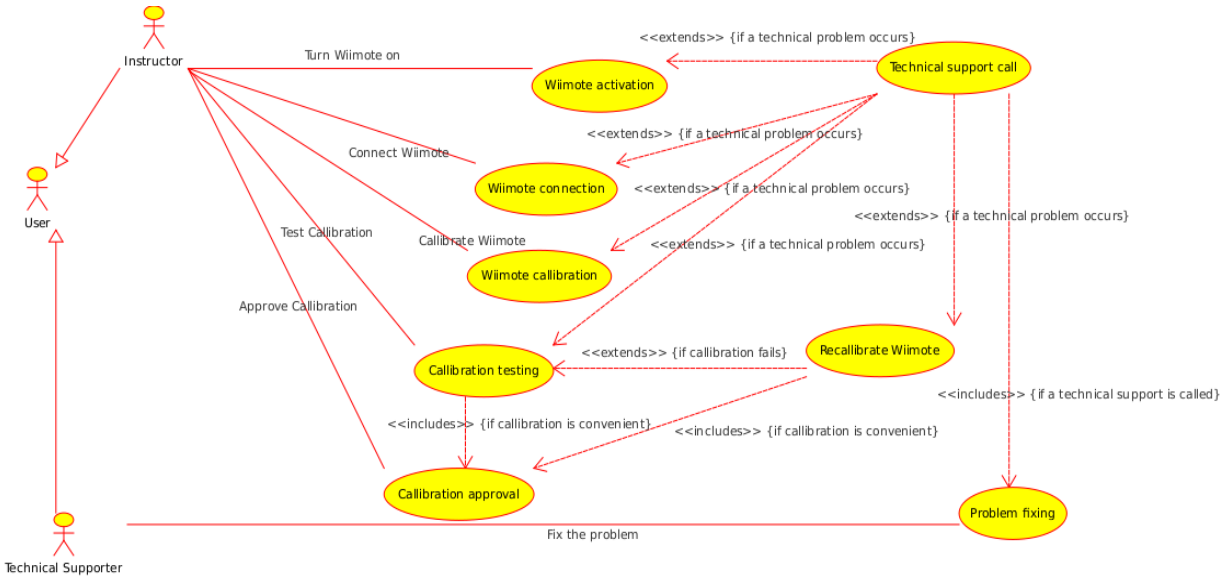


10.1.6. Classroom Session sequence diagram

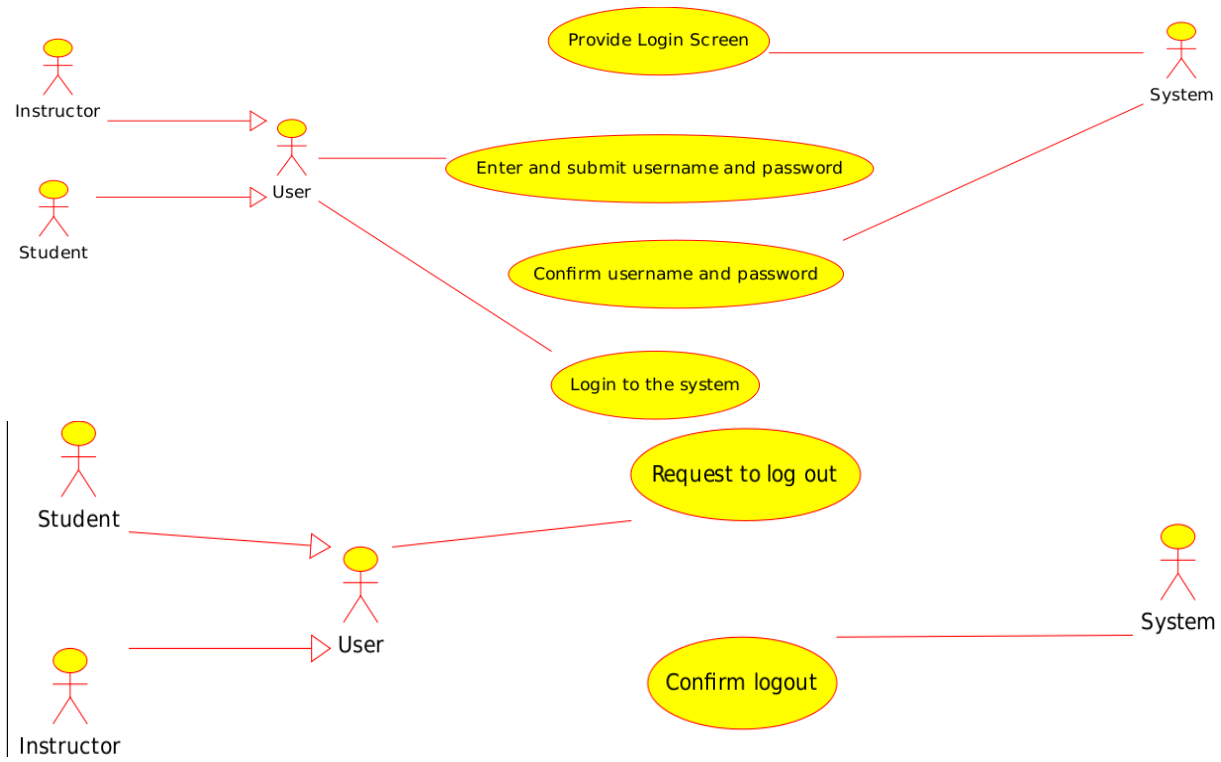


10.2. Detailed Use-Case Diagrams

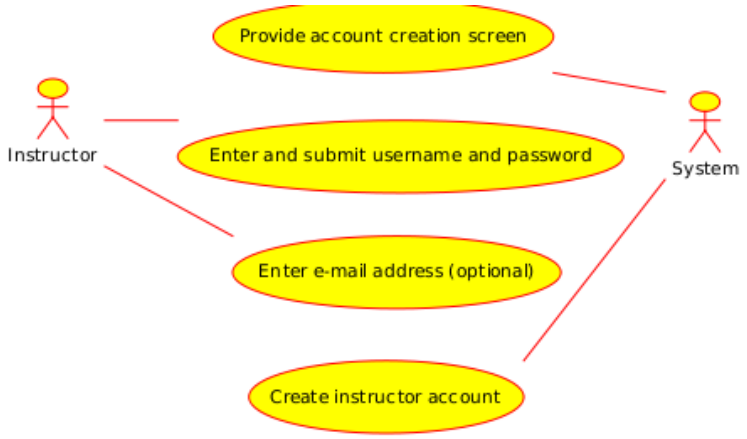
10.2.1. Calibration use-case diagram

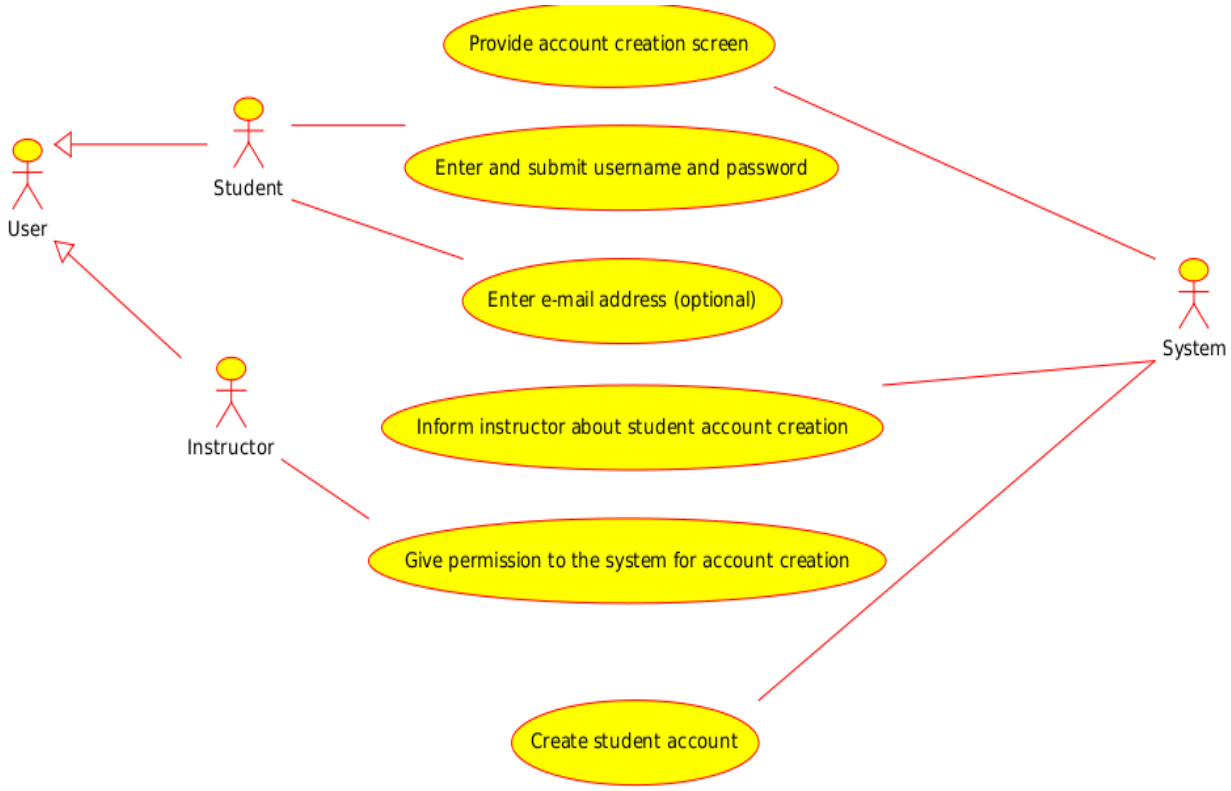


10.2.2. Log in and Log out use-case diagrams



10.2.3. Create Student Account and Instructor Account use-case diagrams



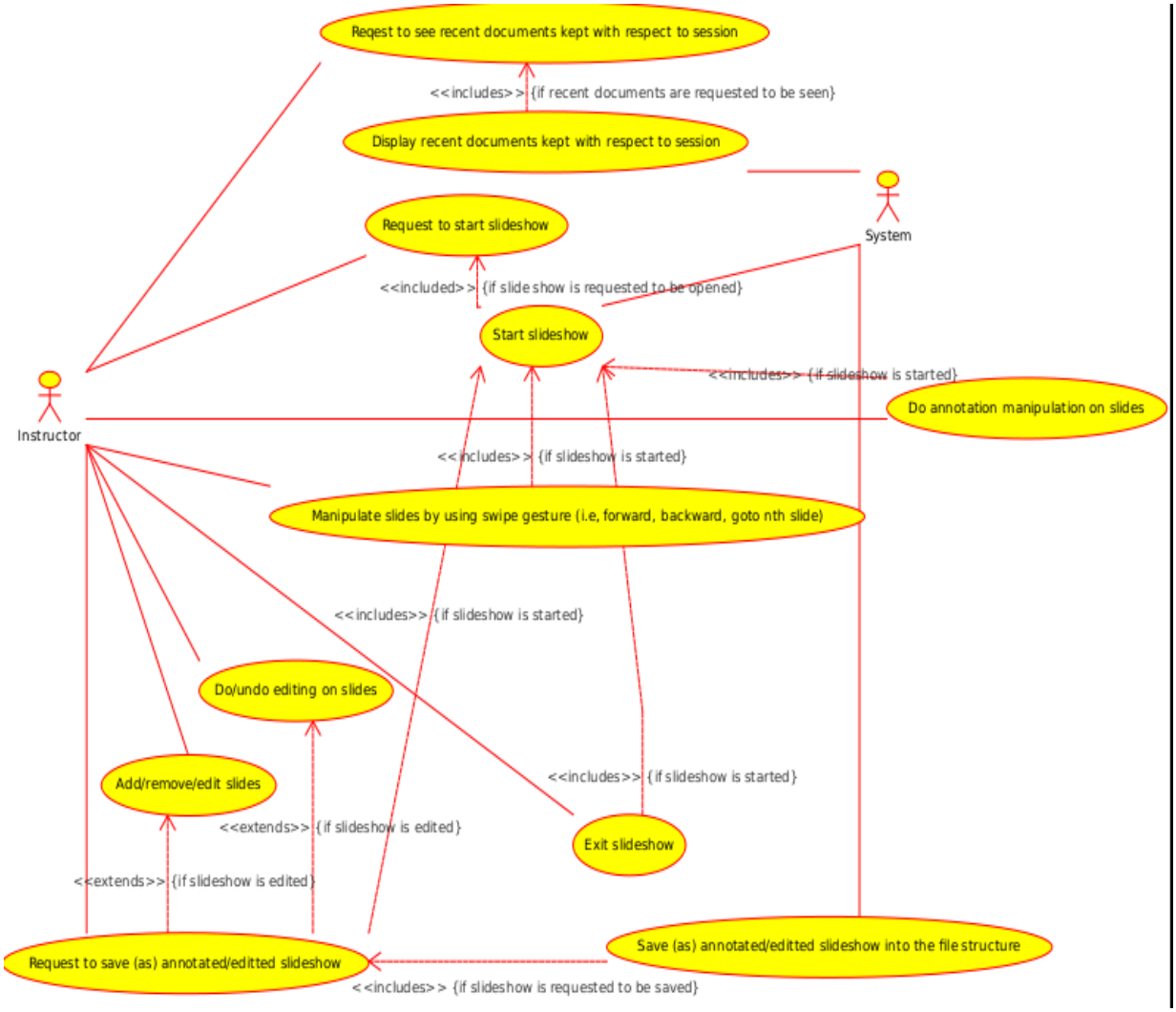

Provide account creation screen
Enter and submit username and password
Enter e-mail address (optional)
Inform instructor about student account creation
Give permission to the system for account creation
Create student account
User
Student
Instructor
System

10.2.4. Annotation use-case diagram

10.2.5. Collaboration use-case diagram

10.2.6. File Manipulation use-case diagram

10.2.7. Presentation Manipulation use-case diagram



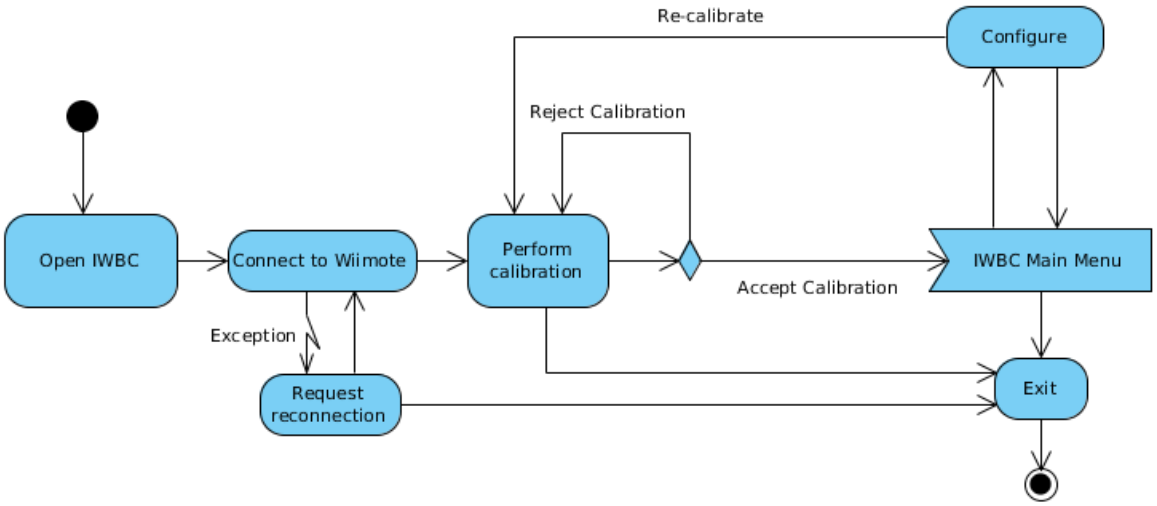
10.2.8. Video Manipulation use-case diagram

10.2.9. Webpage Manipulation

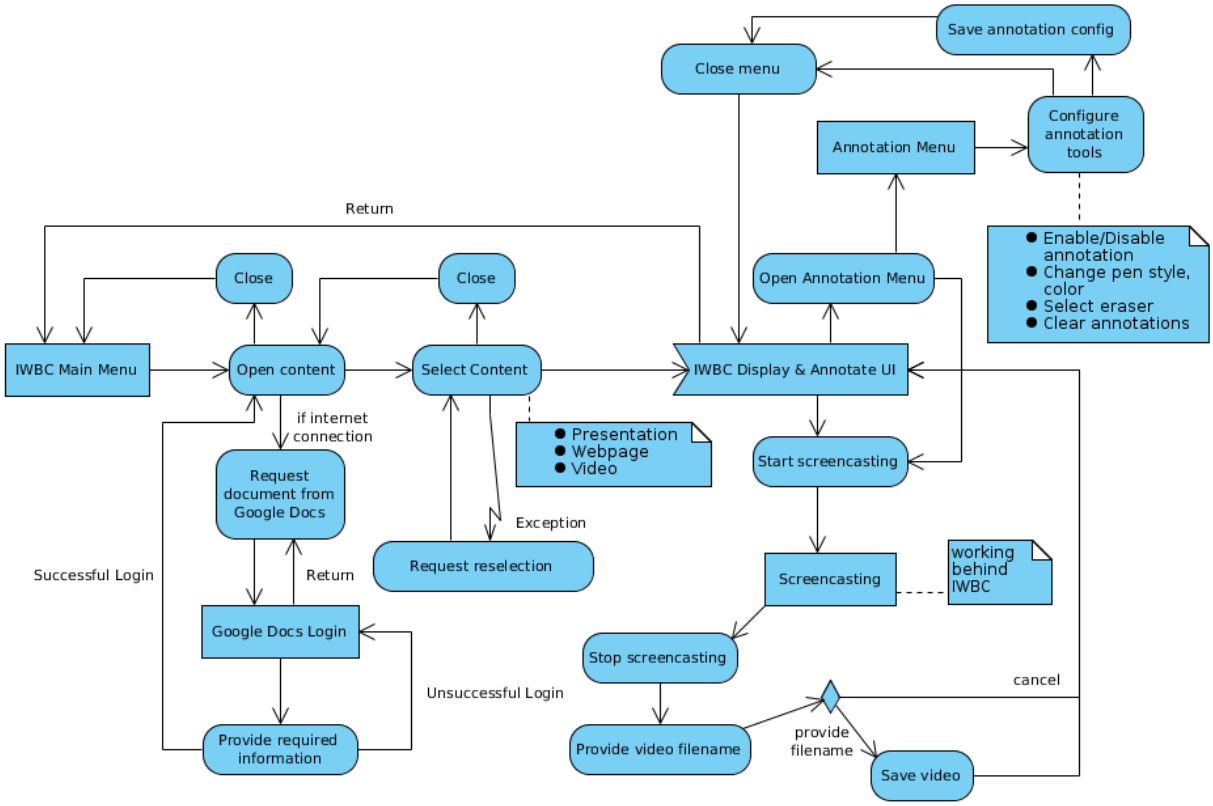
10.3. Activity diagrams

10.3.1. IWBC activity diagrams

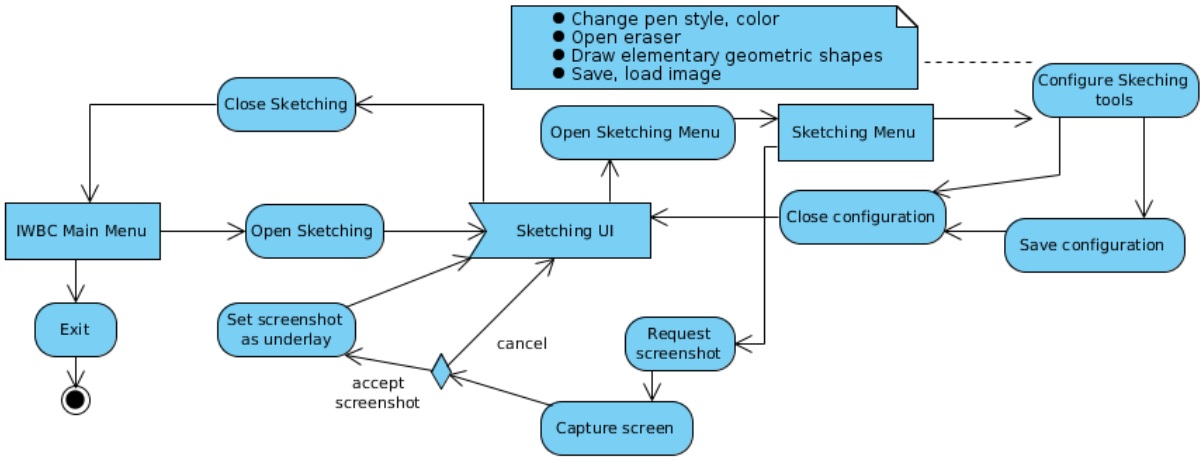
10.3.1.1. Initialization and configuration



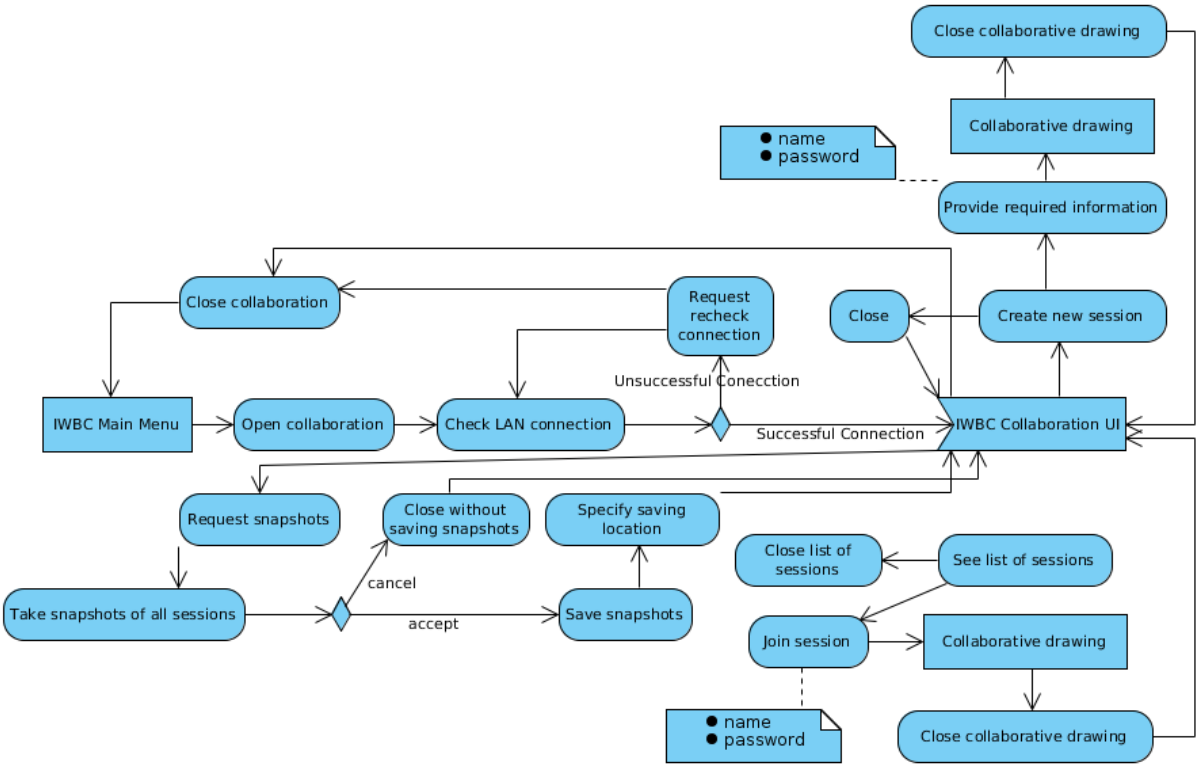
10.3.1.2. Content display and annotation



10.3.1.3. Sketching

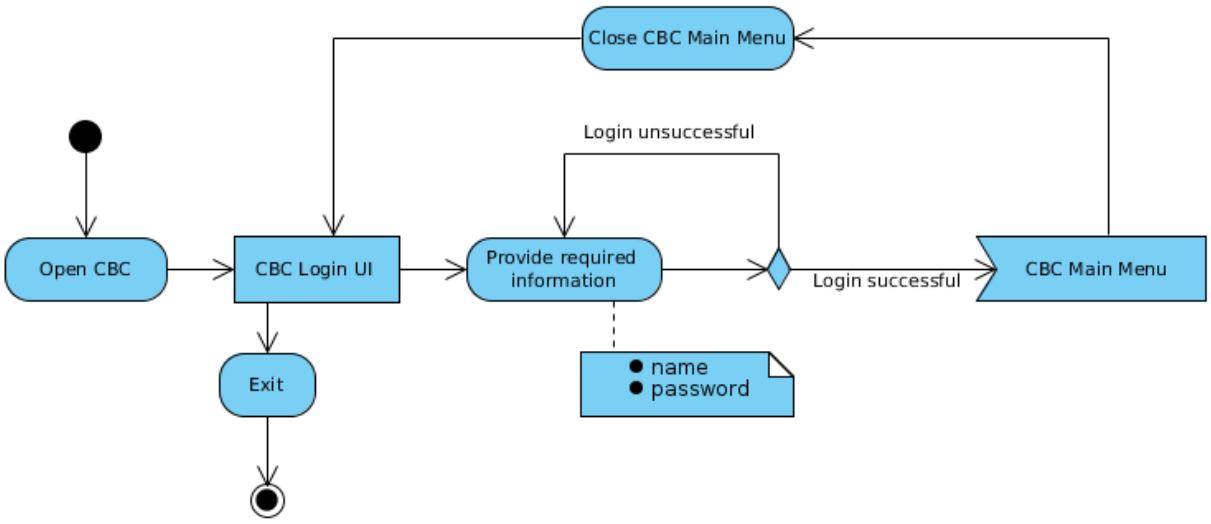


10.3.1.4. Collaboration



10.3.2. CBC activity diagrams

10.3.2.1. CBC login



10.3.2.2. CBC collaboration