Karlsruhe Institute of Technology

MORR
(Medical Online Research Recorder)

PSE-Project
Functional Specification Document

Irma Suppes, Karl Rubel,

Mingyi Li, Niklas Bülow, Sönke Jendral

December 5, 2019

Supervisors:
Paula Breitling, Dr. Till Riedel, Tobias Röddiger

Document Version History

Revision	Date	Description
1.0.0	22.11.2019	Creation and release of document.

Contents

1 Introduction

1.1 Problem statement

Artificial intelligence (AI) is a field of computer science which enables machines to autonomously learn from experience and adjust to new inputs. It is nowadays common when researching online to have to browse multiple websites in complex sequences. This problem can be addressed by optimizing the online research process with the prediction of potential destination pages using AI approaches. Such approaches require data enriched with information about user interactions with web pages, which the project aims to provide.

1.2 Project definition

The project MORR (Medical Online Research Recorder) aims at developing an application which collects information about user behavior during online research, in particular in the medical field. The application enables recording of user interactions, such as mouse, keyboard and web browser events. The recorded data is stored and can later on be extracted upon request.

The following chapters of the document specify in detail the application requirements.

2 Definition of Goals

The purpose of this project is to create an application which records the online research process of users. The focus of the application is to collect data and make it available to data scientists. This chapter gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the following chapters.

2.1 Mandatory Criteria

MC10 **Data recording**
User interactions during the online research must be recorded.

MC20 **User controllability**
The recording session is controlled by the user. It must be started and stopped by the user.

MC30 **Start notification**
Start notification informs the user about the start of the recording session and must be displayed to the user before the start of the recording session.

MC40 **User awareness**
A user must be aware of the recording process at any time during the session.

MC50 **Confirmation of the saving of the recorded data**
A user is asked to confirm if the recorded session is to be saved.

MC60 **Past recording sessions**
A user must be able to view the recordings of the past sessions.

MC70 **Data provisioning**
A data scientist must be able to extract the recorded data.

2.2 Optional Criteria

OC10 **Session duration**
Minimal/maximal recording duration of the session can be set.

OC20 **Inactivity**
A timer can be set in order to stop the recording of an event after a certain period of inactivity.

OC30 **Data visualization**
The recorded data can be visualized (e.g histograms, heatmap).

OC40 **Website-specific event type filtering**
The types of the recorded events can be filtered for any website.

2.3 Delimiting Criteria

DC10 **Data interpretation**
The application does not interpret the recorded data.

DC20 **Spying on the user**
The application does not secretly monitor and gather information about the user.

3 Project Scope

3.1 Scope

The application is used to record cancer or genetic disorder related researches conducted by a doctor and gather data for data scientists who use the data to predict the target pages in such researches.

3.2 Target Audience

The application is aimed at two major audiences:

1. Doctors who conduct cancer or genetic disorder related researches in certain online databases.

2. Data scientists who use the gathered data to predict the target pages in such researches.

3.3 Operating Conditions

The application will be used in an office or at home during research.

4 Project Environment

The product runs on a personal computer.

4.1 Software

- Operating System: Microsoft Windows 10 (minimum build: 1809)
- Web browsers:
  - Google Chrome (minimum version: 57)
  - Mozilla Firefox (minimum version: 57)
- Development Tools:
  - Programming Languages:
    C# (Core Application), JavaScript (WebExtensions)
  - Integrated Development Environment (IDE): Microsoft Visual Studio 2019
  - Documents: PDF (created using LaTeX)
  - Version Control: Git (GitLab)

4.2 Hardware

The application runs on personal computers with stable internet connection and average or better performance.

5 Functional Specifications

This chapter is meant to organize and describe the specifications that dictate the way the application functions for different actors.

5.1 High-level specification

The user interface components mentioned in this section are specified in 5.3.

FS10 **Identifiable sessions**
Actor: User
Precondition: The application is running. The configuration file is valid.
Action: When the user starts a session-recording by pressing the button labeled "Start Recording", a unique session-ID gets created.
Postcondition: The recording-managing component holds a session-ID which all events recorded during the current session will get associated with.

FS20 **Event timings**
Actor: User
Precondition: A session is currently being recorded.
Action: When the user triggers an event specified in chapter 6, the respective module (see 5.4) generates a data-structure (as defined in 6.3) associated with the event and adds the timestamp to it.
Postcondition: The data structure associated with the event is sent to the data-managing component and contains a timestamp describing its time of occurence.

FS30 **Event types**
Actor: User
Precondition: A session is currently being recorded.
Action: When the user triggers an event specified in chapter 6, the respective module (see 5.4) generates a data-structure (as defined in 6.3) associated with the event and adds the event-type to it.

Postcondition: The data structure associated with the event is sent to the data-managing component and contains a field describing its event-type.

FS40 **Event processing**
Actor: User
Precondition: A session is currently being recorded.
Action: When the user triggers an event specified in chapter 6, the data structure corresponding to this event is processed and filtered by modules (see 5.4 and 10.1). If the configuration does not specify to discard the event, the event data structure is sent to the data-managing component by the respective module.
Postcondition: The event data structure is either sent to the data-managing component for serialization or deleted based on a configured rule.

FS50 **Configurability**
Actor: Administrator
Precondition: No session is currently being recorded.
Action: An administrator opens the configuration file and adds/modifies/removes rules affecting some of the configurable behavior.
Postcondition: The configuration-managing component will use the configuration as specified in the changed configuration file the next time the application is launched.

FS51 **Configuring active modules**
Actor: Administrator
Precondition: No session is currently being recorded.
Action: An administrator opens the configuration file and changes the modules the application is allowed to use.
Postcondition: The module-managing component will manage the modules as specified in the changed configuration file the next time the application is launched.

FS52 **Configuring video recording**
Actor: Administrator
Precondition: No session is currently being recorded.
Action: An administrator opens the configuration file and changes the framerate, bitrate or resolution of the recorded video stream.
Postcondition: The video-recording-managing component will record the video stream as specified in the changed configuration file the next time the application is launched.

FS53 **Configuring audio recording**
Actor: Administrator

Precondition: No session is currently being recorded.
Action: An administrator opens the configuration file and enables or disables the recording of audio.
Postcondition: The audio-recording-managing component will record or not record audio as specified in the changed configuration file the next time the application is launched.

FS54 **Configuring saving path**
Actor: Administrator
Precondition: No session is currently being recorded.
Action: An administrator opens the configuration file and changes the system path where the recording of the current user will be stored.
Postcondition: The recording-managing component will store the recordings at the location as specified in the changed configuration file the next time the application is launched.

FS60 **Error-detection**
Actor: User
Precondition: The configuration file is invalid.
Action: The user starts the application. The configuration-managing component checks the configuration file and detects syntactic errors, therefore the application interrupts the startup procedure.
Postcondition: The application has not regularly started and is therefore not ready for recording. For UI details, see Error dialog.

FS70 **User controllability (start)**
Actor: User
Precondition: The application is running. No session is currently being recorded.
Action: The user clicks on the button labeled "Start Recording". The recording-managing component starts a new session-recording once the information dialog (see FS80) is confirmed by the user.
Postcondition: A session is currently being recorded.

FS80 **User instruction**
Actor: User
Precondition: The application is running. No session is currently being recorded. The configuration file is valid.
Action: The user clicks on the button labeled "Start Recording". The user-interface-managing component opens an information dialog.
Postcondition: An information dialog is being displayed, advising the user not to enter any personal or confidential business information.

FS90 **User controllability (stop)**
Actor: User
Precondition: A session is currently being recorded.
Action: The user clicks on the button labeled "Stop recording". The recording-managing component will stop the current session-recording.
Postcondition: No session is currently being recorded. The user-interface-managing component displays the save dialog to the user, providing the options to store or discard the recorded session. For UI details, see Save dialog.

FS100 **User controllability (quit)**
Actor: User
Precondition: The application is running. No session is currently being recorded.
Action: The user chooses the "Quit" option. The application terminates.
Postcondition: The application is no longer running.

FS110 **Quit while recording**
Actor: User
Precondition: The application is running. A session is currently being recorded.
Action: The user chooses the "Quit" option while a session is being recorded. The user-interface-managing-component opens the save dialog.
Postcondition: The save dialog is displayed to the user. The application terminates immediately if the user decided to discard the recording or terminates after the recording has been saved by the recording-managing component if the user decided to save the recording.

FS120 **Save prompt**
Actor: User
Precondition: A session-recording has been stopped and the save dialog is displayed.
Action: The user chooses whether to save or discard the recorded session.
Postcondition: If the user chose to save the recording, the recording is stored by the recording-managing component at the location specified in the configuration file. If the user chose the discard-option, the recording is not stored.

FS130 **User reviewability**
Actor: User
Precondition: One or more sessions have been recorded and not deleted.
Action: A user clicks on the button labeled "Open recordings folder". The user-interface-managing component opens the recordings folder in the Windows File Explorer, displaying the files in which this user's past recordings have been stored.
Postcondition: The folder containing the user's recordings is opened in the Windows File Explorer. The user can view the video stream contained in

these files by opening them with a compatible video-player, e.g. Windows Media Player.

5.2 Core application specification

The core application has to cover the following responsibilities:

FS200 **Video stream recording**
Actor: User
Precondition: A session is currently being recorded.
Action: While a session is being recorded, the video-recording-managing component constantly captures the device's primary screen's content into a video stream which the recording-managing component then serializes. The video stream forms the basis for every recording.
Postcondition: The session recording contains a video stream containing the device's primary screen's content.

FS210 **System module invocation**
Actor: User
Precondition: The application is running.
Action: The user clicks on the button labeled "Start Recording". The system-module-managing component invokes the necessary system modules according to the configuration.
Postcondition: All system modules that should be active according to the configuration are active and listening for their respective events.

FS220 **System module configuration**
Actor: User
Precondition: The configuration file is valid.
Action: The user starts the application. The system-module-managing component configures the system modules according to the configuration.
Postcondition: The system modules are configured according to the configuration and the application is ready for recording.

FS230 **On-demand module registration**
Actor: User
Precondition: A session is currently being recorded.
Action: The user starts an application (e.g. a web browser) which contains an on-demand module. The on-demand module notifies the on-demand-module-managing component about its availability. The on-demand-module managing component reacts by opening a communication port to the on-demand

module to allow for reception of event-data. It also configures the on-demand module according to the configuration.
Postcondition: The on-demand module is ready to send filled-in events to the data-managing component.

FS240 **On-demand module initial registration**
Actor: User
Precondition: The application is running. An application containing an On-demand module is already running.
Action: The user clicks on the button labeled "Start Recording". The on-demand-module managing component registers the available On-demand module and reacts by opening a communication port to the on-demand-module to allow for reception of event-data. It also configures the on-demand module according to the configuration.
Postcondition: The on-demand module is ready to send filled-in events to the data-managing component.

FS250 **Data serialization**
Actor: User
Precondition: A session is currently being recorded.
Action: The user triggers an event which is noticed by a module. The module sends the filled-in event to the data-managing component. The data-managing component collects events from multiple modules and the recording-managing component stores them in the same file (see 6.1).
Postcondition: The event-data received from all active modules is written to the same file.

FS260 **Combinability of modules**
Actor: User
Precondition: The configuration file is valid. All modules required by the configuration are installed.
Action: The user starts the application and starts a session-recording. The module-managing component sets up the internal processing-pipeline described in 10.1 in such a way that modules that receive events as input are connected to those that generate events as output and the modules adhere to the configuration.
Postcondition: The application is ready to process the events in the pipeline as configured.

FS270 **Context management**
Actor: User
Precondition: A session is currently being recorded.
Action: After interacting with an application **A**, the user starts interacting with another application **B**. The context-managing component registers this

change (see 6.2) and informs its modules where necessary.
Postcondition: The context-managing component holds information on the current context and the modules that need to be informed about changes to the context have been informed.

5.3 User interface specification

FS300 **Recording indicator**
Actor: User
Precondition: The application is running. No session is currently being recorded. The configuration file is valid.
Action: The user clicks on the button labeled "Start Recording". The user-interface-managing component will start displaying a yellow border along the screen edges.
Postcondition: A yellow border is being drawn along the screen edges as long as the recording is active.

FS310 **Error dialog**
Actor: User
Precondition: The configuration file is invalid.
Action: The user starts the application. The configuration-managing component will detect syntactic errors in the configuration file and the user-interface-managing component will display an error-message.
Postcondition: An error dialog is displayed, informing that the current configuration is invalid. When the user clicks the button labeled "OK" displayed in this dialog, the application will terminate.

FS320 **Save dialog**
Actor: User
Precondition: A session is currently being recorded.
Action: The user stops the recording by pressing the button labeled "Stop recording" or the button labeled "Quit". The user-interface-managing component opens a new save dialog.
Postcondition: A dialog window is displayed to the user, providing the options "Save recording" and "Discard recording".

FS330 **Event-Data extraction**
Actor: Data scientist
Precondition: One or more sessions have been recorded and not deleted.
Action: A data scientist starts the supplied command-line-tool and supplies the paths of one or multiple recordings as well as an output path as parame-

ters. The tool will start extracting the stored event-data from the recording.
Postcondition: The event-data has been extracted from the specified recordings and stored in CSV-format (comma separated values) at the location specified as output path. The recordings specified as input files have not been altered.

FS340 **Tray-Icon**
Actor: User
Precondition: The configuration file is valid.
Action: The user starts the application. The user-interface-managing component adds a new icon to the tray, indicating that the software has been launched successfully.
Postcondition: An additional icon is displayed in the system tray.

FS350 **Tray-Menu**
Actor: User
Precondition: The application is running.
Action: The user clicks on the tray-icon. The user-interface-managing component displays a menu providing the following options:

* "Start Recording" or "Stop recording" (based on whether a session is already being recorded)
* "Open recordings directory"
* "Quit"

Postcondition: A small menu is displayed above the tray icon, providing above options.

5.4 Per-module specification

This section discusses the default modules which will be available and ship with the application at release in order to fulfill the mandatory requirements.

5.4.1 System modules

System modules are modules which communicate with the operating system. They are invoked as soon as a session-recording is started and only terminate at the end of a recording. All system modules are collecting modules as specified in 10.1.3.

Window-management module

FS400 **Window management events**
Actor: User
Precondition: A session is currently being recorded.
Action: When the user performs one of the following window interactions, a window event (as defined in D310) gets created by this module and sent to the data-managing component.
Postcondition: The data-managing component received a filled-in window event.

FS401 **Focusing a window**
Actor: User
Precondition: A session is currently being recorded.
Action: The user focuses a window. The module creates a focus-window event (as defined in D311) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in focus-window event.

FS402 **Moving a window**
Actor: User
Precondition: A session is currently being recorded.
Action: The user moves a window. The module creates a move-window event (as defined in D312) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in move-window event.

FS403 **Resizing a window**
Actor: User
Precondition: A session is currently being recorded.
Action: The user modifies the size of a window. The module creates a resize-window event (as defined in D313) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in resize-window event.

FS404 **Minimizing/Maximizing/Restoring a window**
Actor: User
Precondition: A session is currently being recorded.
Action: The user minimizes, maximizes or restores a window. The module creates a minimize-window/maximize-window/restore-window event (as defined in D314) and sends it to the data-managing component.

Postcondition: The data-managing component received a filled-in minimize-window/maximize-window/restore-window event.

Mouse module

FS410 **Mouse interaction events**
Actor: User
Precondition: A session is currently being recorded.
Action: When the user performs one of the following mouse inputs, a specialized mouse event gets created by this module and sent to the data-managing component.
Postcondition: The data-managing component received a filled-in mouse event.

FS411 **Mouse Click**
Actor: User
Precondition: A session is currently being recorded.
Action: The user presses/releases the left/right/middle mouse button. The module creates a press/release left/right/middle mouse-button event (as defined in D320) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in press/release left/right/middle mouse-button event.

FS412 **Scroll Wheel**
Actor: User
Precondition: A session is currently being recorded.
Action: The user scrolls the mouse-wheel. The module creates a scroll mouse-wheel event (as defined in D321) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in scroll mouse-wheel event.

FS413 **Mouse Movement**
Actor: User
Precondition: A session is currently being recorded.
Action: The user moves the mouse. The module creates a mouse-movement event (as defined in D322) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in mouse-movement event.

Keyboard module

FS420 **Keyboard interaction events**
Actor: User
Precondition: A session is currently being recorded.
Action: When the user performs one of the following keyboard inputs, a specialized keyboard event gets created by this module and sent to the data-managing component.
Postcondition: The data-managing component received a filled-in keyboard event.

FS421 **Releasing/Pressing a key**
Actor: User
Precondition: A session is currently being recorded.
Action: The user presses/releases a key. The module creates a press/release keyboard-key event (as defined in D330) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in press/release keyboard-key event.

Clipboard module

FS430 **Clipboard events**
Actor: User
Precondition: A session is currently being recorded.
Action: When the user performs one of the following clipboard-interactions, a specialized clipboard event gets created by this module and sent to the data-managing component.
Postcondition: The data-managing component received a filled-in clipboard event.

FS431 **Copying to the clipboard**
Actor: User
Precondition: A session is currently being recorded.
Action: The user copies data to the clipboard. The module creates a clipboard-copy event (as defined in D340) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in clipboard-copy event.

FS432 **Pasting from the clipboard**

Actor: User
Precondition: A session is currently being recorded.
Action: The user pastes data from the clipboard. The module creates a clipboard-paste event (as defined in D340) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in clipboard-paste event.

5.4.2 On-demand modules

On-demand modules are modules which are dynamically run or stopped based on the software they are tracking. These modules need to register with the application core at runtime. All on-demand modules are collecting modules as specified in 10.1.3.

Browser module

FS440 **Browser events**
Actor: User
Precondition: A session is currently being recorded.
Action: When the user performs one of the following browser-interactions, a browser event (as defined in D350) gets created by this module and sent to the data-managing component.
Postcondition: The data-managing component received a filled-in browser event.

FS441 **Opening a new tab**
Actor: User
Precondition: A session is currently being recorded.
Action: The user opens a new tab in the web browser. The module creates an open-tab event (as defined in D351) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in open-tab event.

FS442 **Switching to a tab**
Actor: User
Precondition: A session is currently being recorded.
Action: The user switches to a tab in the web browser. The module creates a switch-tab event (as defined in D352) and sends it to the data-managing component.

Postcondition: The data-managing component received a filled-in switch-tab event.

FS443 **Closing a tab**
Actor: User
Precondition: A session is currently being recorded.
Action: The user closes a tab in the web browser. The module creates a close-tab event (as defined in D353) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in close-tab event.

FS444 **Navigation**
Actor: User
Precondition: A session is currently being recorded.
Action: The user navigates to an URL. The module creates a navigation event (as defined in D354) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in navigation event.

FS445 **Text input**
Actor: User
Precondition: A session is currently being recorded.
Action: The user inputs text into a form or textbox. The module creates a text-input event (as defined in D355) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in text-input event.

FS446 **Button click**
Actor: User
Precondition: A session is currently being recorded.
Action: The user clicks a button. The module creates a button click event (as defined in D356) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in button-click event.

FS447 **Hovering**
Actor: User
Precondition: A session is currently being recorded.
Action: The user hovers the mouse-pointer over a web-element. The module creates a hover event (as defined in D357) and sends it to the data-managing component.

Postcondition: The data-managing component received a filled-in hover event.

FS448 **Text selection**
Actor: User
Precondition: A session is currently being recorded.
Action: The user selects text on a website. The module creates a text-selection event (as defined in D358) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in text-selection event.

FS449 **File download**
Actor: User
Precondition: A session is currently being recorded.
Action: The user downloads a file. The module creates a download event (as defined in D359) and sends it to the data-managing component.
Postcondition: The data-managing component received a filled-in download event.

6 Application Data

The application's main purpose is all about collecting data. This data is supposed to help to find patterns in research activity using a personal computer. Therefore there is a great range of distinctive datasets that are interesting to collect.

6.1 Session recordings

D10 **File format**: All session recordings are stored locally in a file-based format and may be uploaded to remote storage.

D20 **Unique identifier**: A session recording needs to contain a unique identifier

D21 **Timestamp**: A session recording needs to contain a timestamp of beginning and end of the recording

D22 **Event data**: A session recording needs to contain the events in chronological order

D23 **Video stream**: A session recording needs to contain a video stream

D30 **MPEG-Container**: Each session recording is contained in a mpeg-4 video container

D40 **Video stream format**: May vary but must include the following:

Metadata:

- Framerate
- Resolution
- Bitrate

Data:

- Video
- Audio (if enabled)

6.2 Context

D200 **Context**: Depends on the application in use but is structured as follows:

- A unique identifier for the application currently focused by the user
- Application-specific metadata (e.g. the URL of the current tab for the web browser or the title of the current document for Excel)

6.3 Events

Each dataset in a session is described by an event, which has been recognized and issued by a collecting module. Each event includes multiple properties, which define and describe the event.

D300 **Events**: An event must include the timestamp of its occurrence

D301 **Event identifier**: An event must include the identifier of the module it was issued by

D302 **Event type**: An event must include the type which identifies its general action, e.g. "MouseClick"

6.3.1 Window Management Events

A window-management event is issued by the window-management module.

D310 **Generic window management event**: Includes the following data:

- The title of the interacted window

- The name of the process associated with the window

D311 **Window focus event**: Includes only the generic data.

D312 **Window movement event**: Includes the following additional data:

- Old and new location

D313 **Window resizing event**: Includes the following additional data:

- Old and new window size

D314 **Window maximizing/minimizing/restoring event**: Includes the following additional data:

- The new state of the window:
  * Maximized
  * Restored
  * Minimized

6.3.2 Mouse Interaction Events

A mouse-interaction event is issued by the mouse module.

D320 **Mouse click interaction event**: Includes the following data:

- Clicked button:
  * Right-Mouse-Button
  * Scroll-Wheel-Button
  * Left-Mouse-Button
- Type which identifies if the click is a single or double click
- The HWND of the view that was clicked

D321 **Mouse scroll interaction event**: Includes the following data:

- Scroll amount
- The HWND of the view that was scrolled

D322 **Mouse movement interaction event**: Includes the following data:

- Movement vector

6.3.3 Keyboard Interaction Events

A keyboard-interaction event is issued by the keyboard module.

D330 **Keyboard interaction event**: Includes the following data:

- Pressed key
- All of the modifier keys that are pressed:
  * Ctrl
  * Alt
  * Shift
  * Windows

6.3.4 Clipboard Events

A clipboard event is issued by the clipboard module.

D340 **Clipboard interaction event**: Includes the following data:

- Contained clipboard text
- Interaction type:
  * Copy
  * Paste

6.3.5 Web browser Events

A web browser event is issued by the web browser module.

D350 **Generic web browser event**: Includes the following data:

- URL of the interacted website
- The unique identifier of the tab that the event occured in

D351 **Open tab event**: Includes only the generic event data.

D352 **Switch tab event**: Includes the following additional data:

- The unique identifier of the tab that the user switched to

D353 **Close tab event**: Includes only the generic event data.

D354 **Navigation event**: Includes only the generic event data.

D355 **Text input event**: Includes the following additional data:

- Text which has been inputted by the user
- Textbox or form

D356 **Button click event**: Includes the following additional data:

- Button item title
- URL that the button is linked to, if applicable

D357 **Hover event**: Includes the following additional data:

- Element which has been hovered

D358 **Text selection event**: Includes the following additional data:

- Selected Text

D359 **File download event**: Includes the following additional data:

- URL of the file that was downloaded

– MIME type of the file that was downloaded

7 Non-functional Specifications

NF10 **CPU Performance**
The CPU utilization by the application on a user's device with an Intel Core i5 8250U or better and a screen resolution of 1080p or smaller should not exceed 25% while recording a session.

NF20 **Memory Usage**
The memory usage by the application on the user's device should not exceed 400MB while running a session.

NF30 **Start Response Performance**
The delay between the user starting a session via hitting the button and the actual session start should not exceed 1 second.

NF40 **Saving Response Performance**
After the user has stopped a session, the saving process of the session file should take no more longer than 1 minute.

NF50 **Usability**
After a 5-minute introduction the user does not commit more than 1 mistake on an 8-hour working day.

NF60 **Offline Recordings**
The user can record a session and save sessions locally without being connected to the internet.

NF70 **Modularity**
The application should support running at least all five modules, defined in chapter 5, per session.

NF80 **Reliability**
The probability of a failure during a 60-minute session is under 2%.
A failure is an unintended termination of a recording session with a possible loss of recorded data.
This does not apply should the application be run in an unsupported environment or configuration (see chapter 4).

8 Global test cases

8.1 High-level test cases

TC10
(tests FS10, FS54, FS70, FS250, FS90, FS340, FS350)

Identifiable sessions

Precondition: The application is running. The configuration file is valid.

Test steps: The user repeats the following 256 times:

1. The user starts a recording.
2. The user completes the recording process.

Expected result:

* 256 recordings have been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* Each recording contains a different session identifier (as specified in D20).

TC20
(tests FS40, FS54, FS70, FS250, FS90, FS51, FS340, FS350)

Event discarding

Precondition: The application is running. The configuration file is valid. The configuration specifies to discard events of type "mouse:click".

Test steps:

1. The user starts a recording.
2. The user creates an event of type "mouse:left-click" by clicking the left mouse button.
3. The user creates an event of type "keyboard:press-key" by pressing the 'a' key.
4. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The recording contains only the "keyboard:press-key" event in accordance with D300, D301, D302 and D330.

TC30 *(tests FS40, FS51, FS54, FS70, FS90, FS250, FS340, FS350, FS411, FS421)*

Event transforming

Precondition: The application is running. The configuration file is valid. The configuration specifies to active a 10.1.3 that transforms events of type "mouse:left-click" and "keyboard:press-key" into an events of type "other:ctrl-leftclick".

Test steps:

1. The user starts a recording.
2. The user creates an event of type "keyboard:press-key" by pressing the 'CTRL' key without releasing it.
3. The user creates an event of type "mouse:left-click" by clicking the left mouse button.
4. The user creates an event of type "keyboard:release-key" by releasing the 'CTRL' key.
5. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The recording contains the first three events in accordance with D300, D301, D302, D320 and D330. It additionally contains a fourth event of type "other:ctrl-leftclick" that was created by transforming the first two events.

TC40 *(tests FS51, FS54, FS70, FS90, FS250, FS340, FS350, FS411, FS421)*

Configuring active modules

Precondition: The application is running. The configuration file is valid. The configuration specifies to load the mouse module and to not load the keyboard module.

Test steps:

1. The user starts a recording.
2. The user presses the left mouse button which yields an event by the mouse module.
3. The user presses the 'a' key which would yield an event by the keyboard module.
4. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The recording only contains the event "mouse:left-click" event in accordance with D300, D301 and D302. It does not contain the event that would have been generated by the keyboard module.

TC50 *(tests FS51, FS54, FS70, FS90, FS250, FS340, FS350)*

Configuring video recording

Precondition: The application is running. The configuration file is valid. The configuration specifies to record video with a framerate of 60hz, a bitrate of 4096kbps and a resolution of 1920x1080 pixels.

Test steps:

1. The user starts a recording.
2. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The video stream contained in the recording follows D40, where the framerate is 60hz, the bitrate is 4096kpbs (or lower) and the resolution is 1920x1080 pixels.

TC60 *(tests FS51, FS54, FS70, FS90, FS250, FS340, FS350)*

Configuring audio recording

Precondition: The application is running. The configuration file is valid. The configuration specifies to record audio.

Test steps:

1. The user starts a recording.
2. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The video stream contains the optional audiostream.

TC70 *(tests FS54, FS70, FS90, FS250, FS340, FS350)*

Configuring file path

Precondition: The application is running. The configuration file is valid. The configuration specifies to store the recording at path "%user-profile%\documents\recordings".

Test steps:

1. The user starts a recording.
2. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored in the "\documents\recordings" folder inside the user directory of the user currently logged in.

TC80 *(tests FS60, FS310)*

Error detection

Precondition: The default-configuration file got invalidated by appending a bracket ' { ' to the end of the file.

Test steps:

1. The user starts the application.

Expected result:

* The application detects the errors in the configuration and displays an error dialog.

TC90 *(tests FS70, FS80, FS340, FS350)*

User controllability (start)

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user opens the tray menu by clicking on the application icon in the system tray to open the tray menu.
2. The user clicks on the "Start Recording" entry.

Expected result:

* The application displays an information dialog.

TC100 *(tests FS70, FS80, FS300)*

Information dialog

Precondition: The application is currently displaying the information dialog at the start of a recording.

Test steps:

1. The user dismisses the information dialog.

Expected result:

* The application starts a recording and displays a recording indicator.

TC110 *(tests FS70, FS90, FS340, FS350)*

User controllability (stop)

Precondition: The application is running. The configuration file is valid. A session is currently being recorded.

Test steps:

1. The user opens the tray menu by clicking on the application icon in the system tray to open the tray menu.
2. The user clicks on the "Stop Recording" entry.

Expected result:

* The recording is stopped.
* The application displays a save dialog.

TC120 *(tests FS100, FS340, FS350)*

User controllability (quit)

Precondition: The application is running. No session is currently being recorded.

Test steps:

1. The user opens the tray menu by clicking on the application icon in the system tray to open the tray menu.
2. The user clicks on the "Quit" entry.

Expected result:

* The application is no longer running.

TC130 *(tests FS90, FS110, FS340, FS350)*

Quit while recording

Precondition: The application is running. A session is currently being recorded.

Test steps:

1. The user opens the tray menu by clicking on the application icon in the system tray to open the tray menu.
2. The user clicks on the "Quit" entry.
3. The user chooses any option in the save dialog.

Expected result:

* The application terminates immediately if the user decided to discard the recording or terminates after the recording has been saved if the user decided to save the recording.

TC140 *(tests FS120, FS250)*

Save dialog - Save

Precondition: The application is currently displaying the save dialog.

Test steps:

1. The user clicks on the button labeled "Save".

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.

TC150 *(tests FS120, FS250)*

Save dialog - Discard

Precondition: The application is currently displaying the save dialog.

Test steps:

1. The user clicks on the button labeled "Discard".

Expected result:

* The application is no longer running.

TC160 **Recordings folder**

(tests FS130)

Precondition: The application is running.

Test steps:

1. The user opens the tray menu by clicking on the application icon in the system tray to open the tray menu.
2. The user clicks on the "Open recordings folder" entry.

Expected result:

* The folder containing the user's recordings is opened in the Windows File Explorer.

8.2 Core application test cases

TC200 **Video stream recording**

(tests FS70, FS90, FS200, FS250)

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a recording.
2. The user stops a recording.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The recording contains a video stream. The video stream contains the device's primary screen's content.

TC210 **System module management**

(tests FS70, FS90, FS210, FS220, FS250)

Precondition: The application is running. The configuration file is valid. The configuration specifies that the mouse module should be activated.

Test steps:

1. The user starts a recording.
2. The user clicks the left mouse button.
3. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The recording contains an event of type "mouse:left-click" corresponding to the user interaction.

TC220 **On-demand module management**

(tests FS70, FS90, FS230, FS240, FS250)

Precondition: The application is running. The configuration file is valid. The configuration specifies that the web browser module should be activated.

Test steps:

1. The user starts a recording.
2. The user opens the web browser.
3. The user navigates to the civic.genome.wustl.edu webpage.
4. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The recording contains an event of type "browser:navigate" corresponding to the user interaction.

TC230 **Data serialization**

(tests FS70, FS90, FS250)

Precondition: The application is running. The configuration file is valid. The configuration specifies that both the mouse module and the keyboard module should be activated.

Test steps:

1. The user starts a recording.
2. The user clicks the left mouse button.
3. The user presses the "a"-key.
4. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The recording contains both the "mouse:left-click" and the "keyboard:press-key" events.

TC240 *(tests FS70, FS90, FS250, FS260)*

Combinability of modules

Precondition: The application is running. The configuration file is valid. The configuration specifies that both the mouse module and the keyboard module should be activated. It also specifies another module, that merges a "mouse:left-click" and a "keyboard:press-key" event into a "other:ctrl-click" event.

Test steps:

1. The user starts a recording.
2. The user presses and holds the "CTRL"-key.
3. The user then clicks the left mouse button.
4. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The recording contains a "other:ctrl-click" event.

TC250 *(tests FS70, FS90, FS250, FS260)*

Context management

Precondition: The application is running. The configuration file is valid. The configuration specifies that the web browser module should be activated. It also specifies that the web browser module may only be active when the user uses the web browser.

Test steps:

1. The user starts a recording.
2. The user starts navigating to the civic.genome.wustl.edu webpage.
3. The user then immediately switches to another application.
4. The user completes the recording process.

Expected result:

* A recording has been generated (in accordance with 6.1) and stored at the path specified by the configuration.
* The recording contains a "browser:navigation" event but does not contain any event connected to the network requests the web browser made while the user did not have the web browser focused.

8.3 User interface test cases

TC300 **Event-Data extraction**

(tests FS330)

Precondition: One or more recordings have been created and stored at the path "%userprofile%\documents\recordings\sample.mp4".
Test steps:

1. The data scientist starts the command-line tool with the path "%userprofile%\documents\recordings\" of the recordings and another path "%userprofile%\documents\recordings\extracted\" to extract the data to.

Expected result:

* The event data contained in the recordings at path "%userprofile%\documents\recordings\" is extracted in CSV-format to path "%userprofile%\documents\recordings\extracted\".

TC310 **Tray-Icon**

(tests FS340)

Precondition: The application is not running. The configuration file is valid.
Test steps:

1. The user starts the application.

Expected result:

* The application icon is added to the system tray.

TC320 **Tray-Menu**

(tests FS340, FS350)

Precondition: The application is running.
Test steps:

1. The user clicks the tray-icon.

Expected result:

* The tray-menu is displayed.

8.4 Per-module test cases

8.4.1 System module test cases

Window-management module test cases

TC400 *(tests FS20, FS30, FS54, FS70, FS250, FS401)*

Window focus event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user focuses a "Windows Explorer" window.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains a focus-window event, that contains the title of the "Windows Explorer" window, along with the timestamp at which the user focused the window, the identifier of the window-management module and the "window:focus" type of the event.

TC410 *(tests FS20, FS30, FS54, FS70, FS250, FS402)*

Window move event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user moves a "Windows Explorer" window.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains a move-window event, that contains the title of the "Windows Explorer" window, old and new location of the window along with the timestamp at which the user moved the window, the identifier of the window-management module and the "window:move" type of the event.

TC420 *(tests FS20, FS30, FS54, FS70, FS250, FS403)*

Window resize event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user resizes a "Windows Explorer" window.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains a resize-window event, that contains the title of the "Windows Explorer" window, old and new size of the window along with the timestamp at which the user resized the window, the identifier of the window-management module and the "window:resize" type of the event.

TC430 *(tests FS20, FS30, FS54, FS70, FS250, FS404)*

Window maximize/minimize/restore event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user maximizes/minimizes/restores a "Windows Explorer" window.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains a maximize-window/minimize-window/restore-window event, that contains the title of the "Windows Explorer" window, the new state of the window - "maximized"/"minimized"/"restored" along with the timestamp at which the user maximized/minimized/restored the window, the identifier of the window-management module and the "window:maximize"/"window:minimize"/"window:restore" type of the event.

Mouse module test cases

TC440 *(tests FS20, FS30, FS54, FS70, FS250, FS411)*

Mouse click event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user presses the left/right/middle mouse button in "Windows Explorer".
3. The user releases the left/right/middle mouse button in "Windows Explorer".
4. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains a mouse click event, that contains type of the clicked button - left/right/middle, type of the click - single, the HWND associated with the "Windows Explorer" window that was clicked, along with the timestamp at which the user clicked the mouse button, the identifier of the mouse-interaction module and the "mouse:left-click"/"mouse:right-click"/"mouse:middle-click" type of the event.

TC450 *(tests FS20, FS30, FS54, FS70, FS250, FS411)*

Mouse double click event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user performs double-clicks the left mouse button.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains a double-click-mouse event, that contains the type of the clicked button - left, type of the click - double, HWND associated with the "Windows Explorer" window that was clicked, along with the timestamp at which the user double clicked the mouse button, the identifier of the mouse-interaction module and the "mouse:left-double-click" type of the event.

TC460 *(tests FS20, FS30, FS54, FS70, FS250, FS412)*

Mouse scroll event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user scrolls the mouse wheel in "Windows Explorer".
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains a mouse scroll event, that contains the scroll amount, the HWND associated with the "Windows Explorer" window that was scrolled, along with the timestamp at which the user used the scroll wheel, the identifier of the mouse-interaction module and the "mouse:scroll" type of the event.

TC470 *(tests FS20, FS30, FS54, FS70, FS250, FS413)*

Mouse move event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user moves the mouse.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains a mouse move event, that contains the movement vector along with the timestamp at which the user moved the mouse, the identifier of the mouse-interaction module and the "mouse:move" type of the event.

Keyboard module test cases

TC480 *(tests FS20, FS30, FS54, FS70, FS250, FS421)*

Keyboard event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user presses a key while a "Windows Explorer" window is focused.
3. The user releases a key while a "Windows Explorer" window is focused.
4. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the key-press event and release-key event. Each event contains the name of the pressed/released key, the names of all modifier keys that were pressed along with the timestamp at which the user pressed/released a key, the identifier of the keyboard-interaction module and the "keyboard:press-key"/"keyboard:release-key" type of the event.

Clipboard module test cases

TC490 *(tests FS20, FS30, FS54, FS70, FS250, FS431)*

Clipboard copy event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user copies text from a "Microsoft Excel" window.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the clipboard copy event, that contains the clipboard text associated with "Microsoft Excel", the type of the clipboard interaction - copy, along with the timestamp at which the user copied the text to the clipboard, the identifier of the clipboard-interaction module and the "clipboard:copy" type of the event.

TC500 *(tests FS20, FS30, FS54, FS70, FS250, FS432)*

Clipboard paste event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user pastes text in the "Mozilla Firefox/Google Chrome" address bar.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the clipboard paste event, that contains the clipboard text associated with "Mozilla Firefox/Google Chrome", the type of the clipboard interaction - paste, along with the timestamp at which the user pasted the text from the clipboard, the identifier of the clipboard-interaction module and the "clipboard:paste" type of the event.

8.4.2 On-demand module test cases

Browser module test cases

TC510 *(tests FS20, FS30, FS54, FS70, FS250, FS441)*

Browser open tab event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user opens a new tab in "Mozilla Firefox/Google Chrome".
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the open-tab-browser event, that contains the URL of the interacted website and the unique identifier of the new tab along with the timestamp at which the user opened a new tab in "Mozilla Firefox/Google Chrome", the identifier of the web browser module and the "browser:open-tab" type of the event.

TC520 *(tests FS20, FS30, FS54, FS70, FS250, FS442)*

Browser switch to tab event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user switches to a tab in "Mozilla Firefox/Google Chrome".
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the switch-tab-browser event, that contains the URL of the interacted website, the unique identifier of the old tab and the unique identifier of the tab that the user switched to along with the timestamp at which the user switched to a tab "Mozilla Firefox/Google Chrome", the identifier of the web browser module and the "browser:switch-tab" type of the event.

TC530 *(tests FS20, FS30, FS54, FS70, FS250, FS443)*

Browser close tab event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user closes a tab "Mozilla Firefox/Google Chrome".
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the close-tab-browser event, that contains the URL of the interacted website and the unique identifier of the tab along with the timestamp at which the user closed a tab in "Mozilla Firefox/Google Chrome", the identifier of the web browser module and the "browser:close-tab" type of the event.

TC540 *(tests FS20, FS30, FS54, FS70, FS250, FS444)*

Browser navigation event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user navigates to a URL on the google.com search results page.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the navigation-browser event, that contains the URL of the page that the user navigated to and the unique identifier of the tab, which the navigation event occured in, along with the timestamp at which the user navigated to a URL, the identifier of the web browser module and the "browser:navigation" type of the event.

TC550 *(tests FS20, FS30, FS54, FS70, FS250, FS445)*

Browser text input event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user inputs text into a search box on the google.com page.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the text-input-browser event, that contains the URL of the google.com page, the unique identifier of the tab, which the text-input event occured in, the text which has been input by the user, information about search box, where text has been input, along with the timestamp at which the user input the text, the identifier of the web browser module and the "browser:text-input" type of the event.

TC560 *(tests FS20, FS30, FS54, FS70, FS250, FS446)*

Browser button click event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user clicks a button on the google.com page.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the button-click-browser event, that contains the URL of the google.com page, the unique identifier of the tab, which the button-click event occured in, the title of the button item, the URL of the website that the button is linked to (if applicable) along with the timestamp at which the user clicked a button on the page, the identifier of the web browser module and the "browser:button-click" type of the event.

TC570 *(tests FS20, FS30, FS54, FS70, FS250, FS447)*

Browser hovering event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user hovers the mouse-pointer over a web-element on the google.com page.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the hover-browser event, that contains the URL of the google.com page, the unique identifier of the tab, which the hover event occured in, the information about the element which has been hovered along with the timestamp at which the user hovered the web-element, the identifier of the web browser module and the "browser:hover" type of the event.

TC580 *(tests FS20, FS30, FS54, FS70, FS250, FS448)*

Browser text selection event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user selects text on the google.com page.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the text-selection-browser event, that contains the URL of the google.com page, the unique identifier of the tab, which the text-selection event occured in, selected text along with the timestamp at which the user selected the text, the identifier of the web browser module and the "browser:text-selection" type of the event.

TC590 *(tests FS20, FS30, FS54, FS70, FS250, FS449)*

Browser file download event

Precondition: The application is running. The configuration file is valid.

Test steps:

1. The user starts a new session recording.
2. The user downloads a file from the google.com page.
3. The user completes the recording process.

Expected result:

* A recording has been created and stored at the path specified by the configuration.
* The recording contains the download-browser event, that contains the URL of the google.com page, the unique identifier of the tab, which the download event occured in, the URL of the file that was downloaded along with the timestamp at which the user downloaded the file, the identifier of the web browser module and the "browser:download" type of the event.

9 Scenarios and use cases

9.1 User related scenarios

The buttons and the yellow border in scenario 1 to 4 are specified in 10.3.

9.1.1 Scenario 1: User successfully records a session

1. The application displays a button at the bottom right corner of the screen.
2. User clicks the button to start recording.
3. The application starts to record and displays a yellow border on screen during the whole recording time to indicate that the session is being recorded.
4. User researches in different databases during recording.
5. User finishes the research and clicks the same button to stop recording.
6. The application displays a dialog to ask the user, if the recording data should be saved or not.
7. User clicks "save" to save the recorded data.
8. The application saves the recorded data and the yellow border disappears at the same time.

9.1.2 Scenario 2: User discards a session

1. The application displays a button at the bottom right corner of the screen.

2. User clicks the button to start recording.

3. The application starts to record and displays a yellow border on screen during the whole recording time to indicate that the session is being recorded.

4. User is interrupted by multiple received Emails and reads the Emails for a long period of time.

5. User realizes that the research only takes up 10% of the total recording time and gives up on recording by clicking the same button.

6. The application displays a dialog to ask the user, if the recording data should be saved or not.

7. User clicks "discard" to discard the recorded data.

8. The application discards the recorded data and the yellow border disappears at the same time.

9.1.3 Scenario 3: The minimal recording duration is not reached

Notice: This scenario covers features declared optional.

1. The application displays a button at the bottom right corner of the screen.

2. User clicks the button to start recording.

3. The application starts to record and displays a yellow border on screen during the whole recording time to indicate that the session is being recorded.

4. User is immediately interrupted by an Email after the start of recording and decides to stop recording and read the Email first.

5. User clicks the same button to stop recording.

6. The application displays a dialog indicating that the minimal recording duration (see OC10) is not reached and thus the recorded data will be automatically discarded.

7. User clicks "confirm" button on the dialog and closes it.

8. The application discards the recorded data and the yellow border disappears at the same time.

9.1.4 Scenario 4: The maximal recording duration is reached

Notice: This scenario covers features declared optional.

1. The application displays a button at the bottom right corner of the screen.

2. User clicks the button to start recording.

3. The application starts to record and displays a yellow border on screen during the whole recording time to indicate that the session is being recorded.

4. User researches in different databases during recording but leaves their device without terminating the recording.

5. The application displays a dialog, warning that the maximal recording duration of the session (see OC10) is almost reached.

6. The maximal recording duration of the session (see OC10) is reached. The application displays a dialog to ask the user, if the recording data should be saved or not.

7. User is back and clicks "save" to save the recorded data.

8. The application saves the recorded data and the yellow frame disappears at the same time.

9.2 Scientist related scenarios

9.2.1 Scenario 5: Scientist extracts data

1. Data scientist starts the command line tool (see 10.3.1) provided by the application.

2. Data scientist types in commands including the path of the data and an output directory to extract the data from that path to the output directory.

3. Data scientist can now use the extracted data in the output directory.

9.3 Administrator related scenarios

9.3.1 Scenario 6: Administrator changes configuration

1. The administrator opens the configuration file provided by the application.

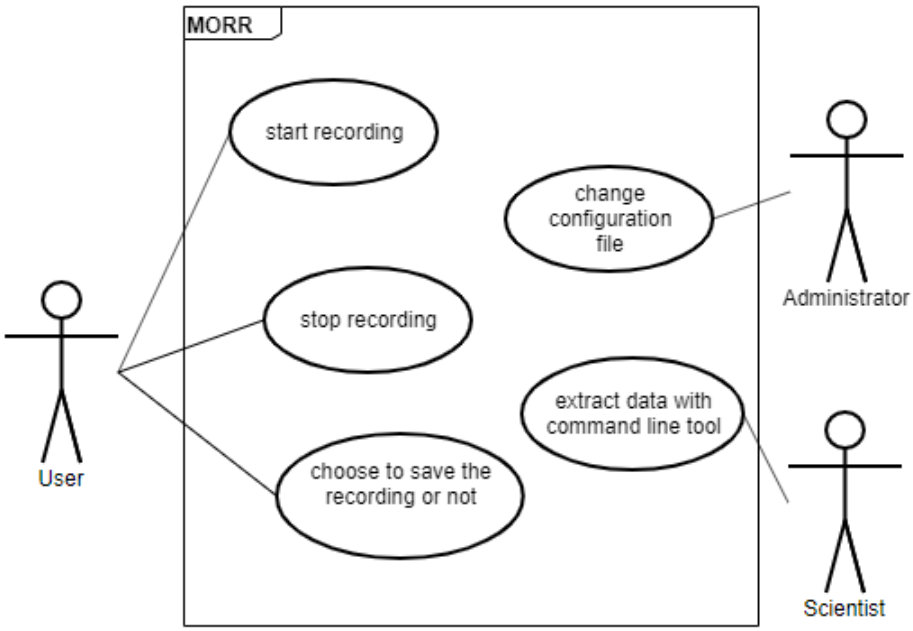
2. The administrator changes the maximal recording duration (see OC10) by editing the configuration file and saving it.

3. The administrator distributes the changed configuration file to a user.

4. The user's application receives the changed configuration file .

5. The maximal recording duration (see OC10) of the application is changed according to the received configuration file.

9.4 Use cases


MORR
start recording
stop recording
choose to save the recording or not
change configuration file
extract data with command line tool
User
Administrator
Scientist

10 System models

10.1 System architecture

The specified criteria demand for a minimalist user interface and little to no bidirectional data flow within the application. As the main focus of this project is collecting, processing and finally storing data, the applied architecture will be a pipeline architecture.

10.1.1 Basic Pipeline

The traditional pipeline architecture refers to a process being split up into several sequential steps, ideally with data buffers in between those, which will be able to independently and possibly asynchronously perform a specific transformation on their data input. The resulting data will then be provided as output for the subsequent step. This works especially well in a scenario where only an unidirectional data flow is required.

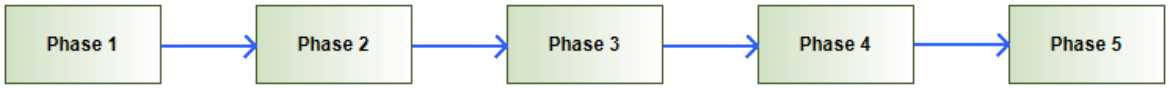


Figure 10.1: A simple pipeline model

10.1.2 The extended pipeline

In this project, the basic pipeline is an insufficient model, as the initial input data will have to be gathered at several, mostly independent places. The web browser module (see FS440 to FS449) will only capture web browser events, the window management module (see FS400 to FS404) cannot also provide the keyboard input etc. In order to solidify all the collected events, the extended pipeline allows for a single processing step to accept input from multiple sources by utilizing the transforming modules as

mentioned in 10.1.3. The result is a tree-structure where the event-data strictly flows from the leafs towards the root, the root being the core application (see 5.2).

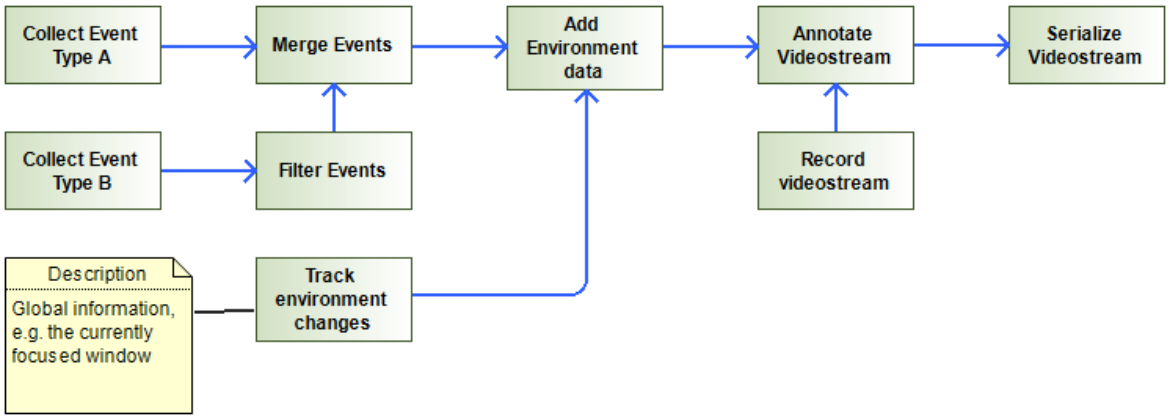


Figure 10.2: An extended pipeline model

10.1.3 Module types

The application needs to provide three different types of modules:

- Collecting modules
  Modules which accept no input from other modules and generate one or more events as output.

- Transforming modules
  Modules which accept one or more events as input from other modules and generate one or more events as output.

- Discarding modules
  Modules which accept one or more events as input from other modules and output no events.

10.2 Dynamic model

The following figure shows an example of how the application could deal with a user clicking on an element in their web browser. The actual flow might differ based on available modules and rule configuration.

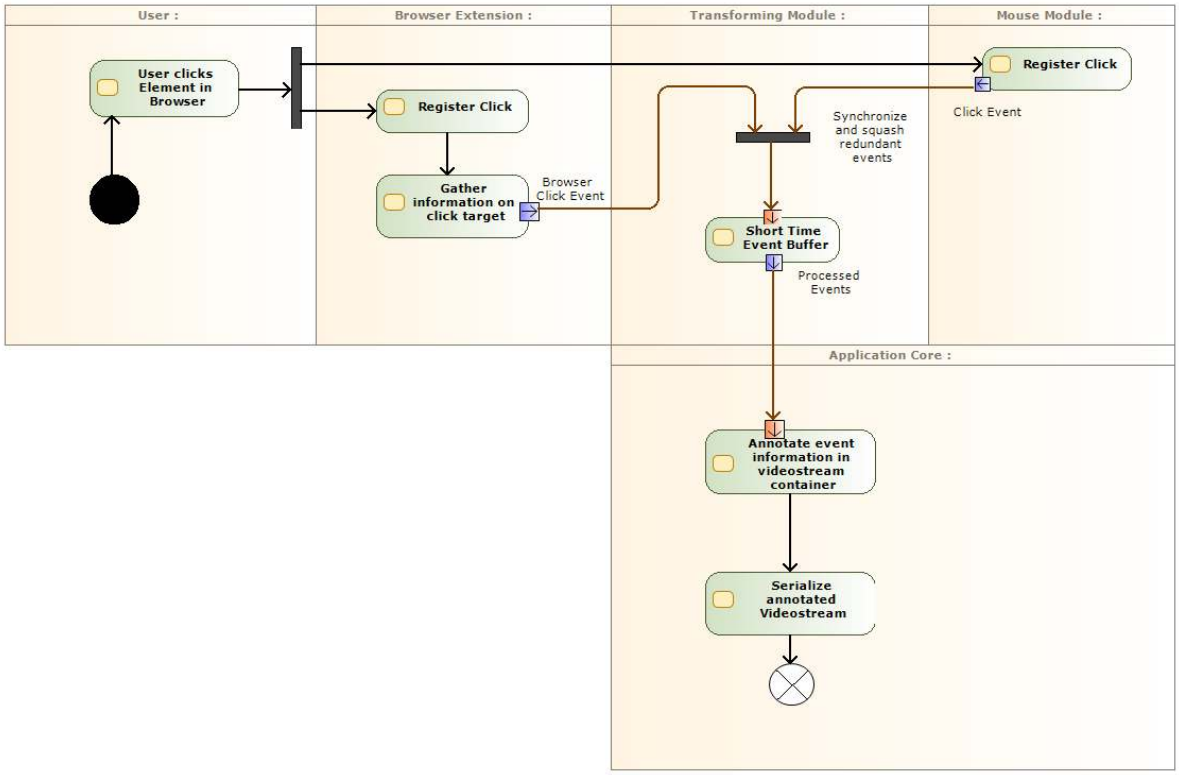


Figure 10.3: User-Click sample activity chart

10.3 User Interface

The user-interface is intentionally minimal as the users should not be disrupted during their usual work-routine. The following figures show drafts and therefore do not represent the final product.

Figure 10.4: A running session recording is indicated by a yellow border.

Figure 10.5: The start/stop recording features may be quickly accessed by clicking on the tray icon (represented by the red dot).

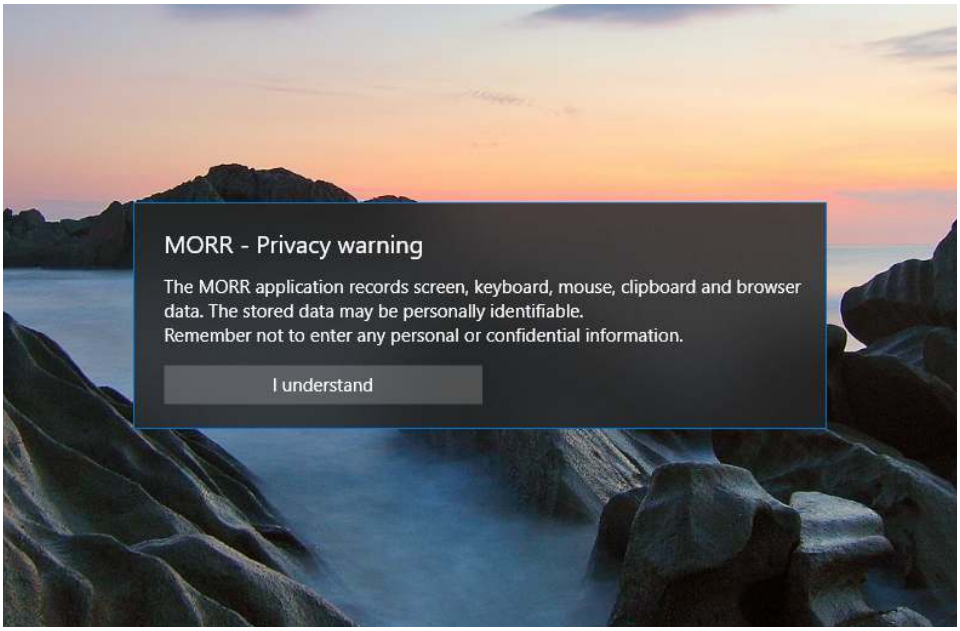


Figure 10.6: The privacy warning appears when a recording is started.

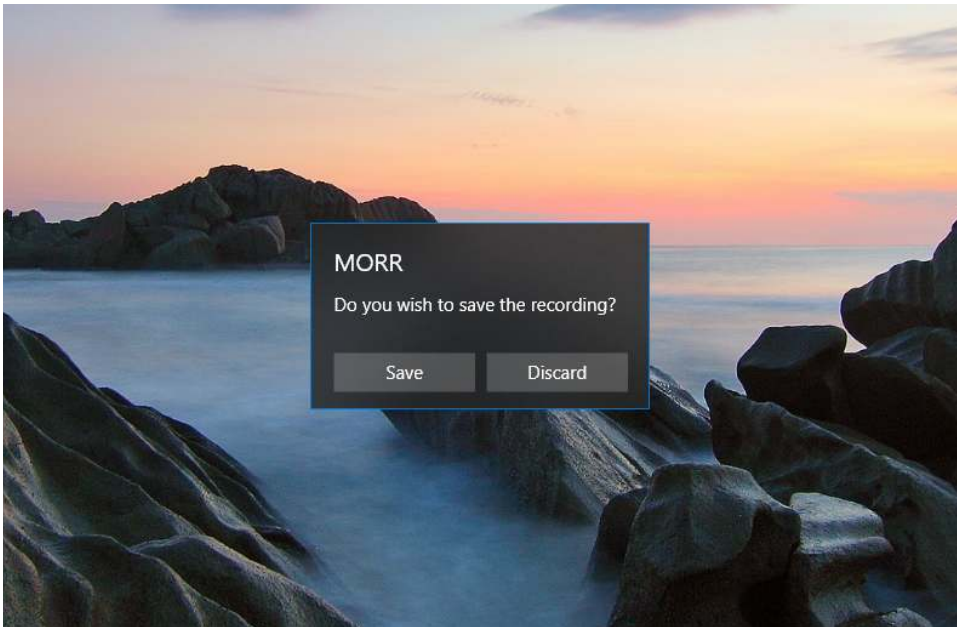


Figure 10.7: The save dialog appears when a recording is stopped.

10.3.1 Data extraction

The data scientists will be able to extract event-information from the video container by utilizing an API (application programming interface) which will be supplied together with the application. Additionally, the application will be bundled with a commandline-tool which allows for extraction of the event-data into the common CSV-format (comma-separated values). See also FS330.

11 Feasibility Study

11.1 Technical Feasibility

The functional specification of this application can be broadly implemented by the Microsoft .NET Core infrastructure. It provides a varied and extensive collection of libraries, which provide us with tools for event logging, video-metadata embedding and inter-module communication.
The user interface framework for this project is WPF. It is feasible enough because our application does not require extensive user interfaces.
Lastly, Google Chrome and Mozilla Firefox provide us with an application interface which can be used to detect events and interactions with the web browser as described in chapter 6.

11.2 Personnel Feasibility

The personnel for this project matches the requirements. The team consists of five computer science students currently pursuing a bachelor of science degree and three supervisors. A requirement for the "Praxis der Softwareentwicklung" module is the successful completion of the "Softwaretechnik 1" module and all orientation examinations. These modules provide fundamental to intermediate knowledge about a software project process and execution.
The time investment for each student is approximately 270 hours. Each team member is equally involved in the implementation of this project.
The project will be executed using the waterfall model.

11.3 Economical Feasibility

As this project is part of a mandatory module for the bachelor's degree in computer science, the students are not receiving compensation.

Therefore there are no concerns about the economic feasibility of this project.

11.4 Legal Feasibility

This project relies heavily on the collection of data from users. With the introduction of the DSGVO by the European Union in 2018 an increased attention must be taken to address these legal regulations.
The application itself always makes sure the user can see that a session is currently running, as described in the functional and non-functional specifications in chapter 5. The user is also directly asked whether he wants to start or stop a session. It must be made sure that the user knows how and why their data is collected on their system and that they are able to delete their data on request.

11.5 Alternatives

An alternative approach to this project is a manual collection of the data. This, however, results in significantly higher costs and time investment. This alternative would also not be able to record data in a matching level of detail, e.g. tracking mouse movement would be hard to realize.

12 Glossary

Administrator A person responsible for setting up and configuring the software. Usually has permissions exceeding those of users.

Data scientist A person working with the recorded data in order to extract knowledge and insights.

Device A single electronic device intended for direct usage by users. In the scope of this project this always refers to a Windows PC as specified in chapter 4.

Event Actions a user performs on a device such as pressing a key or launching a program. The specifics are discussed in chapter 6.

Module In the context of this document, a module refers to an optional component which extends to application's functionality, e.g. by collecting additional data.

Session A session refers to a continuous time interval in which a recording was made on a device.

User A doctor conducting cancer or genetic disorder related research on a device.

Video stream Digital, coherent data encoding the video (and optionally audio-) output of a device.

Web browser An application which allows users to view and interact with websites.