LORENT IT-LÖSUNGEN GMBH
Uellendahler Straße 178
D-42109 Wuppertal

Tel.: +49 (0) 2195 92470-00
Fax: +49 (0) 2195 92470-01
Mail: info@lorent-online.com
www.lorent-online.com

Lailo

Conversational UI Platform

Integrate Lailo on a website

Table of Contents

1 Lailo Smart Character

There are four general strategies on how to integrate the Lailo Smart Character on a website:

1. Use one of our plugins for Content Management Systems
2. Integrate a pre-built widget with only a few lines of code.
3. Use a pre-configured barebone variant, which gives control over the appearance of the Character and its control elements.
4. Build a custom integration of Lailo, using the *Character API* which gives full control over appearance and event handling.

We support the two most common Content Management Systems (CMS) WordPress and TYPO3. For more information on them, see chapter 2. If Lailo should be integrated into another CMS please contact the Lailo support (support@lailo.ai). We are happy to assist you during the development.

The CMS plugins integrate the pre-built widgets, which will be presented in chapter 3. Therefore, if a new widget is released, it will also be available for the CMS plugins. The widgets make it very simple for the frontend developer to integrate Lailo on any website with only a few lines of HTML and JavaScript.

If the Widgets do not fit into the concept of the website, the barebone template should be considered. The template handles basic events that occur during a conversation and can ease a custom integration. The pre-configured barebone template is discussed in chapter 4.

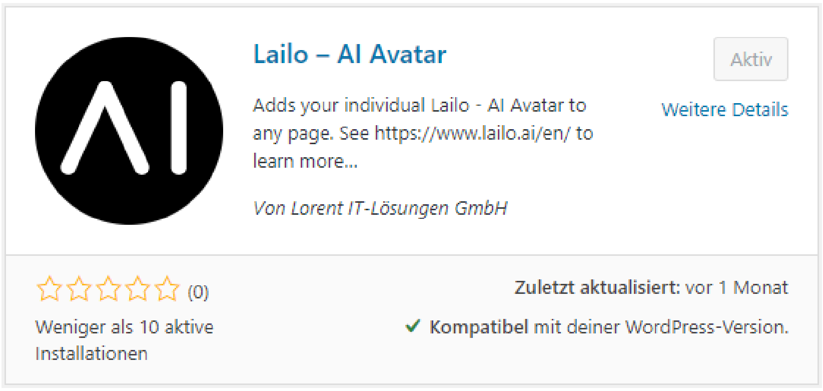
The fourth strategy, which gives the developer the most freedom for the design and functionality, will be introduced in chapter 5. For this solution, the most background knowledge is necessary. However, it gives the customer the opportunity for a highly individual solution.

2 Content Management System Plugins

Content Management Systems help to manage, create, and modify digital content. They typically consist of a backend, where administrators and editors can setup the website. Most CMS can be enhanced by plugins or extensions. Lailo has been integrated in common CMS and therefore make it very easy to integrate.

2.1 Word Press

The Lailo Word Press plugin can be installed from the WordPress plugin store. Within the WordPress backend, navigate to "Plugins" – "Install" and search for "Lailo". The WordPress plugin will appear.



Lailo – AI Avatar

Adds your individual Lailo - AI Avatar to any page. See https://www.lailo.ai/en/ to learn more...

Von Lorent IT-Lösungen GmbH

☆☆☆☆☆ (0)

Weniger als 10 aktive Installationen

Zuletzt aktualisiert: vor 1 Monat

✓ **Kompatibel** mit deiner WordPress-Version.

Figure 1 - Word Press Plugin

Details on the Plugin can also be found on the official WordPress plugin store:

https://de.wordpress.org/plugins/lailo-ai-avatar/

After installing the plugin from the store, a new backend menu "Lailo – AI Avatar" will appear on the left-hand side. Inside this admin page, the user can add new configurations. Each configuration consists of the following parameters (see Figure 2):

- **Smart Character Name**
  A descriptive name of the Character, e.g. "Main Avatar".
- **Bot Secret**
  The bot secret is used to identify which Lailo Smart Character is used during the conversation. Visit https://portal.lailo.ai, go to the settings menu of the Bot and use the Bot Secret which is displayed there.
- **Widget Type**
  Select one of widgets types which are presented in chapter 3.
- **Appearance Configuration**
  Configure primary/secondary colors, primary/secondary text colors and the background color of the input field. An info text will appear when hovering over the question-mark icon.
- **Advanced Settings**
  Below the basic settings, some more advanced settings can be made. Example questions that are displayed within the widgets can be edited here. Also, the widget title, button texts an the input field placeholder can be set here.

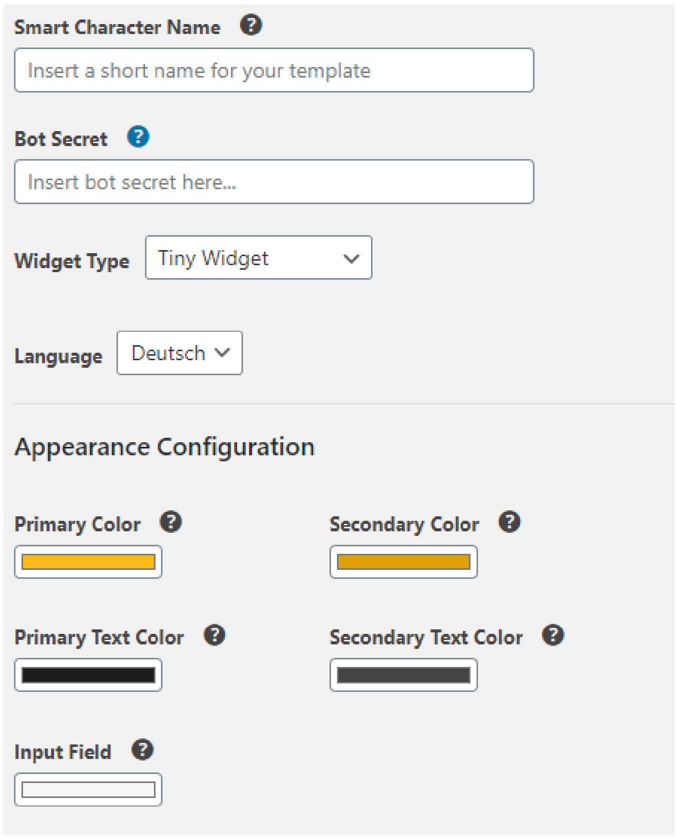
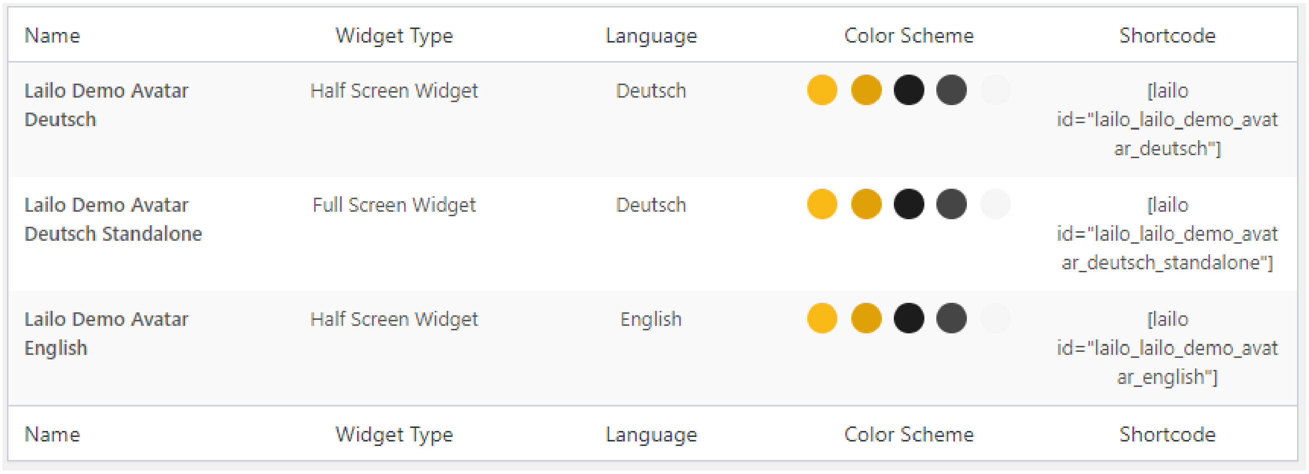

Smart Character Name
Insert a short name for your template
Bot Secret
Insert bot secret here...
Widget Type: Tiny Widget
Language: Deutsch
Appearance Configuration
Primary Color
Secondary Color
Primary Text Color
Secondary Text Color
Input Field

Figure 2 - Word Press Plugin configuration

After saving the above settings a new entry is shown in the overview table (see Figure 3). The plugin automatically generates Shortcodes, based on the “Smart Character Name” which are shown in the last column of the table. Shortcodes are a common way in WordPress to integrate custom functionality. The developer only has to copy this code and insert it on the page, where Lailo should be displayed.

For more information on Shortcodes, see:

https://codex.wordpress.org/Shortcode_API



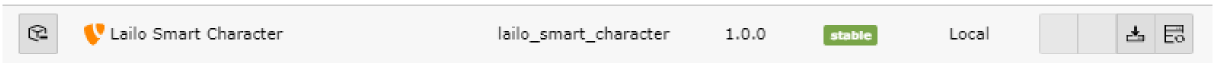
Name	Widget Type	Language	Color Scheme	Shortcode
Lailo Demo Avatar Deutsch	Half Screen Widget	Deutsch	● ● ● ● ○	[lailo id="lailo_lailo_demo_avatar_deutsch"]
Lailo Demo Avatar Deutsch Standalone	Full Screen Widget	Deutsch	● ● ● ● ○	[lailo id="lailo_lailo_demo_avatar_deutsch_standalone"]
Lailo Demo Avatar English	Half Screen Widget	English	● ● ● ● ○	[lailo id="lailo_lailo_demo_avatar_english"]
Name	Widget Type	Language	Color Scheme	Shortcode

Figure 3 - Word Press Plugin configuration list

2.2 TYPO3

The Lailo TYPO3 extension can be installed in the “Extension” section of the TYPO3 backend.

Either search the extension store or contact the Lailo support (support@lailo.ai) to get a ZIP archive of the extension. After installation, make sure that the extension is activated (see Figure 4). A disabled extension shows up in the list but is grayed out.

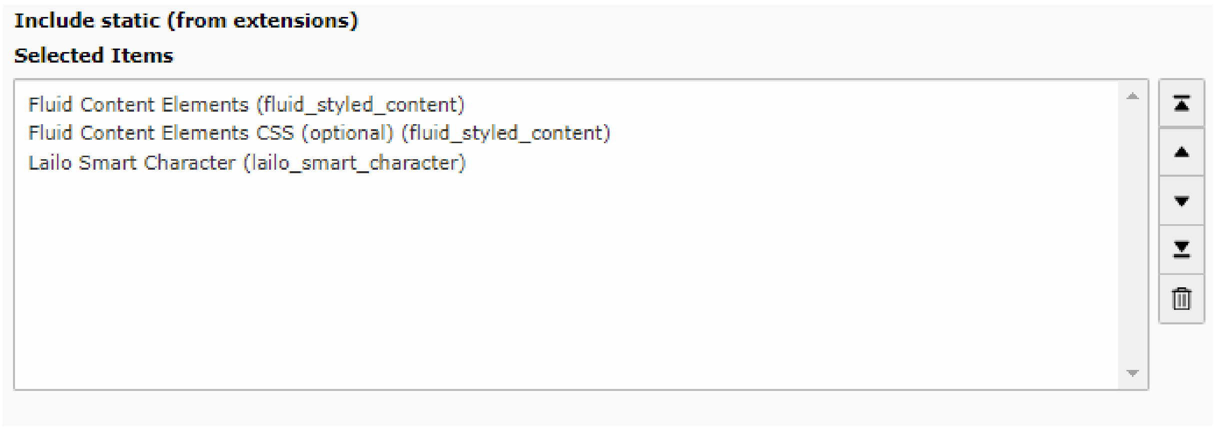


Lailo Smart Character   lailo_smart_character   1.0.0   stable   Local

Figure 4 - Activated TYPO3 extension

After that, go to the Site Template of the page where Lailo should be shown.

Then edit the template and go to the “Include” tab, where the “Lailo Smart Character” extension should be displayed in the “Available Items” within the “Include static (from extension)” section. Click on the extension to move it to the “Selected items” list (see Figure 5).

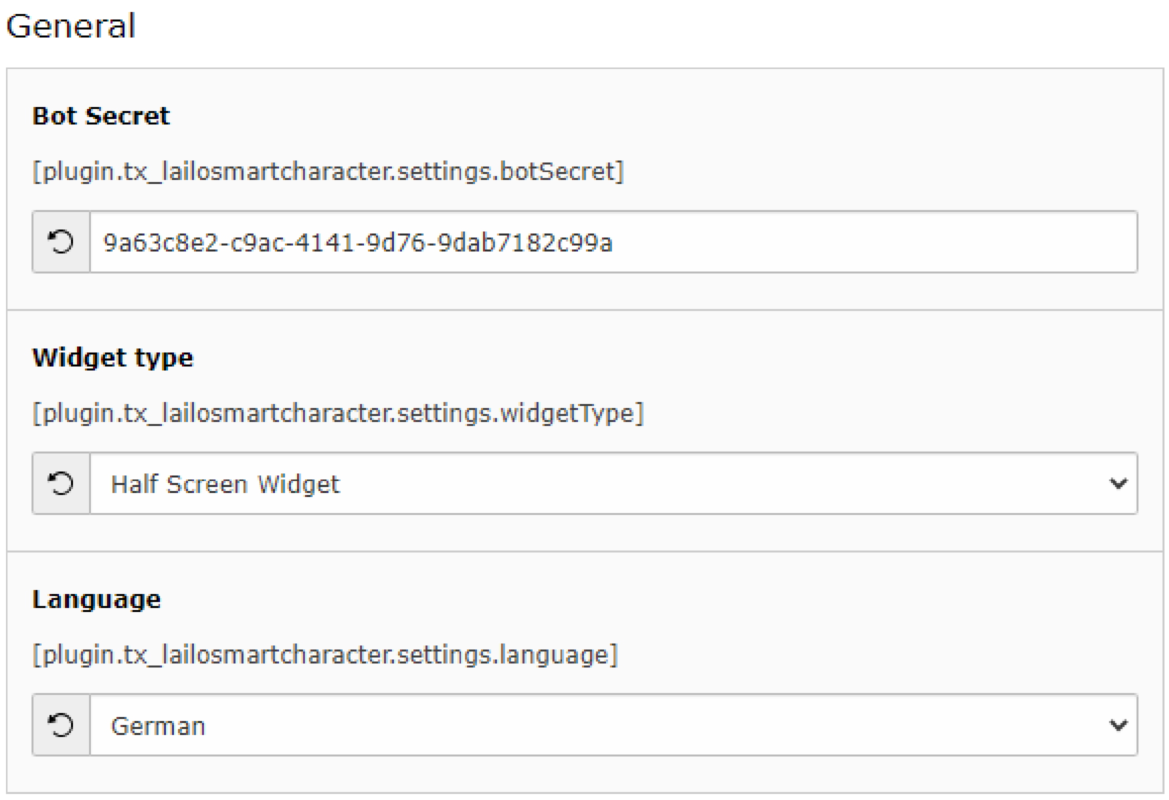

Include static (from extensions)

Selected Items

Fluid Content Elements (fluid_styled_content)
Fluid Content Elements CSS (optional) (fluid_styled_content)
Lailo Smart Character (lailo_smart_character)

Figure 5 - Included Lailo TYPO3 extension

After that, open the "Constant Editor" of the site template and navigate to the "LAILO" category (see Figure 6). Similar to the WordPress Plugin, some crucial settings (Bot Secret, Widget Type, language) can be set here. Appearance configurations, i.e., colors and texts, as well as advanced settings are displayed below. All statements on them in chapter 2.1 also apply for the TYPO3 extension.


General

Bot Secret

[plugin.tx_lailosmartcharacter.settings.botSecret]

9a63c8e2-c9ac-4141-9d76-9dab7182c99a

Widget type

[plugin.tx_lailosmartcharacter.settings.widgetType]

Half Screen Widget

Language

[plugin.tx_lailosmartcharacter.settings.language]

German

Figure 6 - Configure general settings in TYPO3 extension

3 Widgets

The easiest and fastest way to integrate Lailo on a website that does not use a supported CMS, is using one of the pre-built widgets. At the moment, we provide three widgets, which can cover different use-cases.

3.1 Full Screen Widget

In certain cases, the Lailo Smart Character should be displayed as large and eye-catching as possible. Some users may want to have a separate site for the Character or only want to show the character on a single site, which is e.g. displayed on a digital signage screen. Also, on small mobile devices, this can be a neat way to interact with a client.

For this case, we created the *Full Screen Widget*, where the Character is stretched over the complete display. A microphone button (1) on the bottom of the page activates Speech Recognition. Alternatively, a panel can be opened and closed with a button on the right (2). Within this panel (see Figure 8), the client can enter a question via keyboard and click either on the send button or hit *Enter* (3). The answer will be displayed below the text field (4). If an answer also contains links, they will be presented below the answer field (5).

To give the clients a better idea what they can ask the bot, example questions are displayed on below (6). These questions are configurable, so that they match the given scenario.

For more information on how to integrate the *Full Screen Widget* on a website, see chapter 3.5.

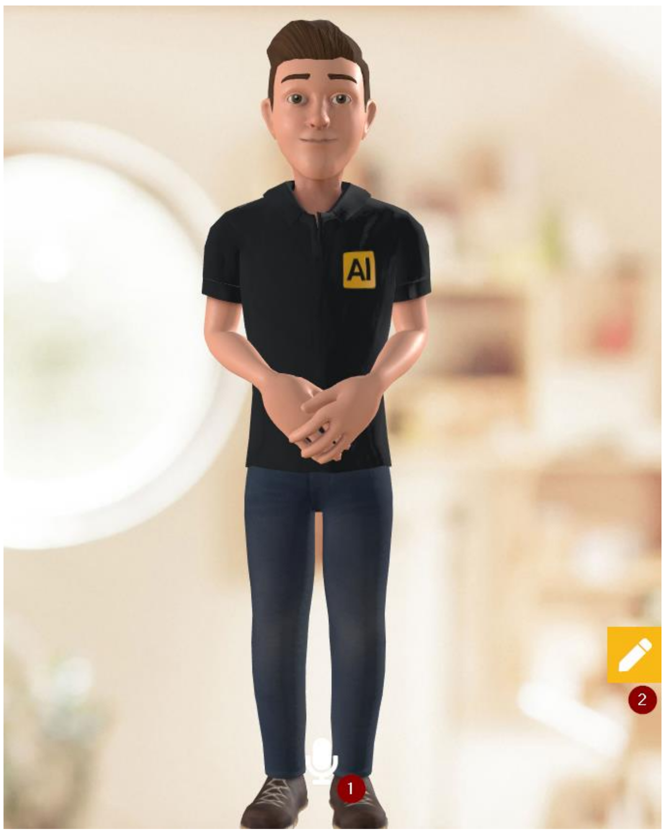
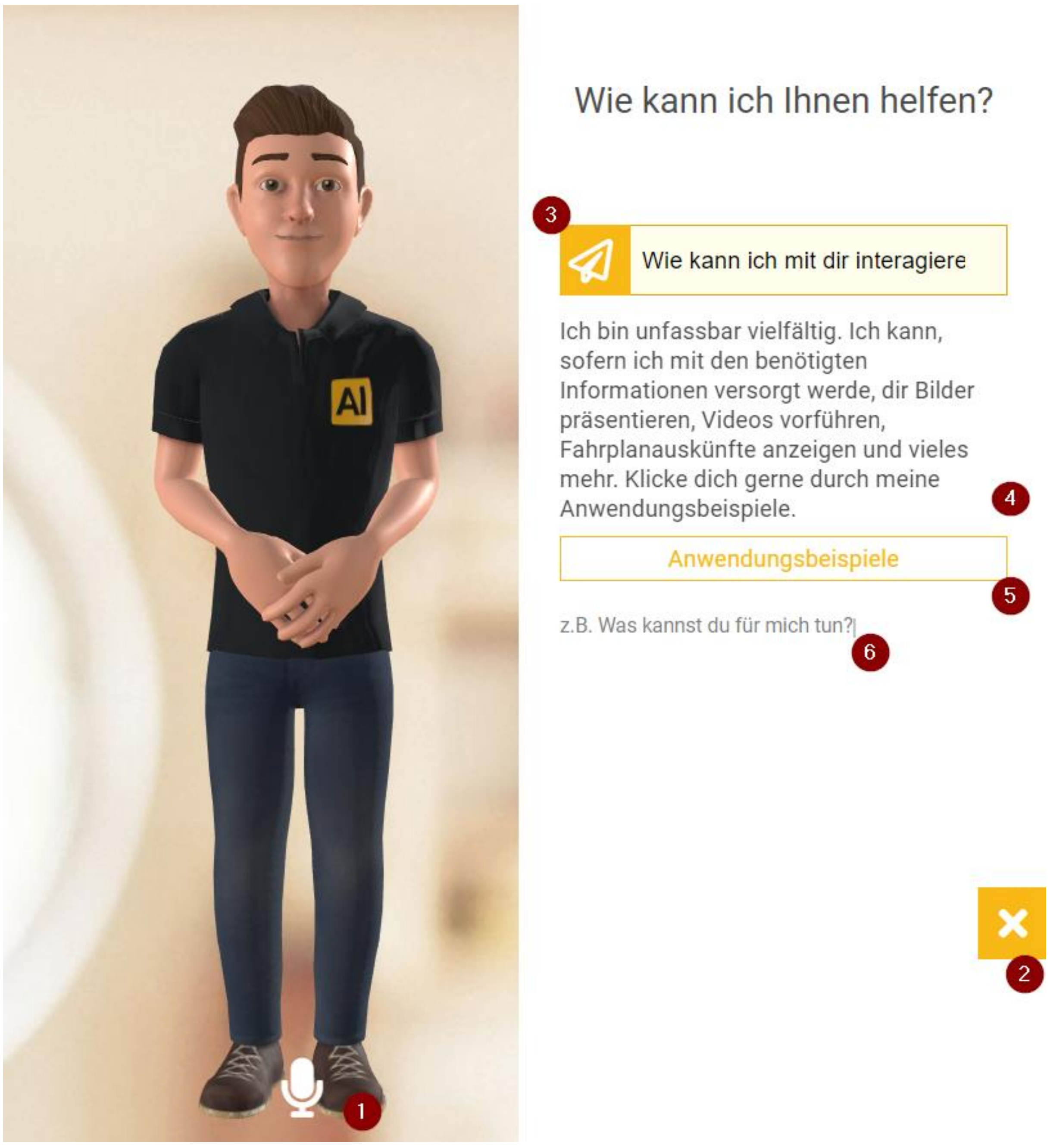


Figure 7 - Full Screen Widget


Wie kann ich Ihnen helfen?
Wie kann ich mit dir interagiere
Anwendungsbeispiele
z.B. Was kannst du für mich tun?
1
2
3
4
5
6

Ich bin unfassbar vielfältig. Ich kann, sofern ich mit den benötigten Informationen versorgt werde, dir Bilder präsentieren, Videos vorführen, Fahrplanauskünfte anzeigen und vieles mehr. Klicke dich gerne durch meine Anwendungsbeispiele.

Figure 8 - Full Screen Widget, with opened panel

3.2 Half Screen Widget

One great feature of the Lailo Smart Character is that it can be integrated on any kind of website. The *Half Screen Widget* sticks a small version of the Avatar and a button on right of the page (see Figure 8). This widget is alongside the regular content and if the client has a question, they can click on the *"Ask me"* button which opens a panel on the right-hand side, which covers almost half of the user's screen (see Figure 10 - Half Screen Widget opened).

Similar to the *Full Screen Widget* the opened panel shows a question field, an answer field, link buttons (if available) and example questions. The client can close the panel again with a click on the button below the panel's content.

Unlike the *Full Screen Widget,* the Speech Recognition button is shown to the left of the question input. If the client starts writing a text, this button changes its behavior and can be used to send the message.

For more information on how to integrate the *Half Screen Widget* on a website, see chapter 3.5.

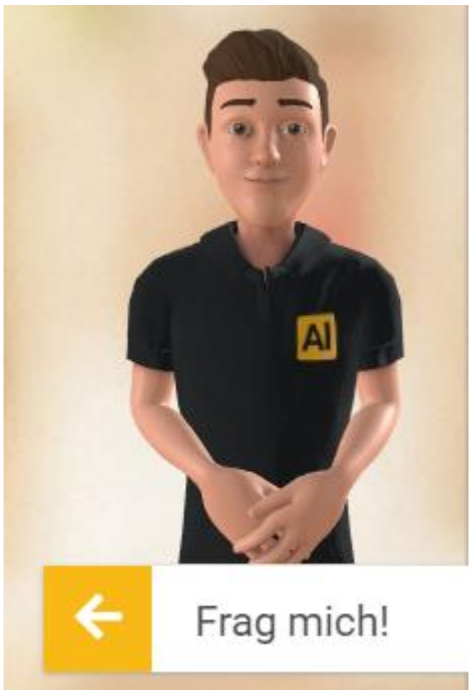
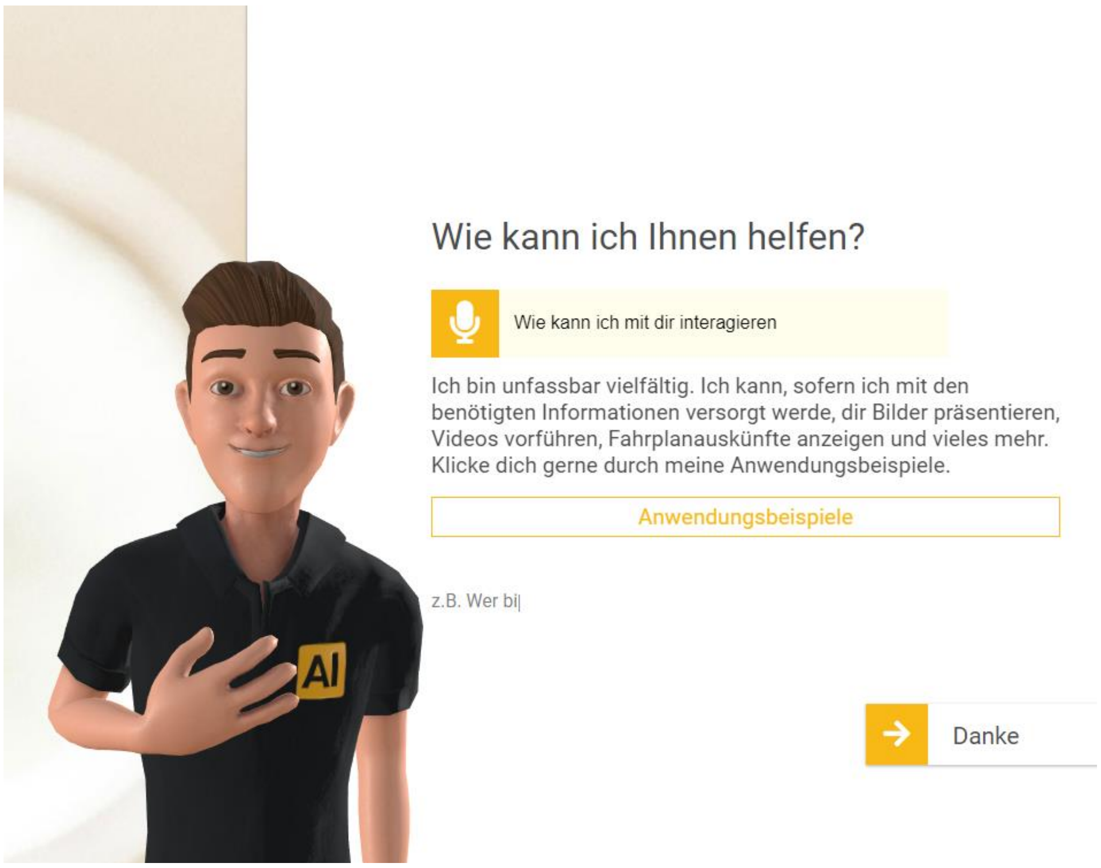


Figure 9 - Half Screen Widget



Wie kann ich Ihnen helfen?

Wie kann ich mit dir interagieren

Ich bin unfassbar vielfältig. Ich kann, sofern ich mit den benötigten Informationen versorgt werde, dir Bilder präsentieren, Videos vorführen, Fahrplanauskünfte anzeigen und vieles mehr. Klicke dich gerne durch meine Anwendungsbeispiele.

Figure 10 - Half Screen Widget opened

3.3 Tiny Widget

The *Tiny Widget* is comparable to the *Half Screen Widget* as both are integrations for an arbitrary website and just add a button and a small presentation of the Character on the right-hand side of the site (see Figure 11).

However, when the client clicks on the *"Ask me"* button, a much smaller panel opens, compared to the *Half Screen Widget (see* Figure 12*)*.

The elements within the panel are again the same: Listen/Send button, question input field, answer text field, links, and example questions.

For more information on how to integrate the *Tiny Widget* on a website, see chapter 3.5.

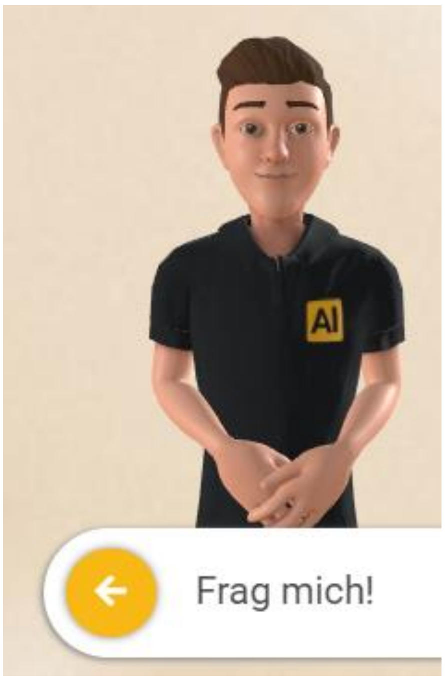
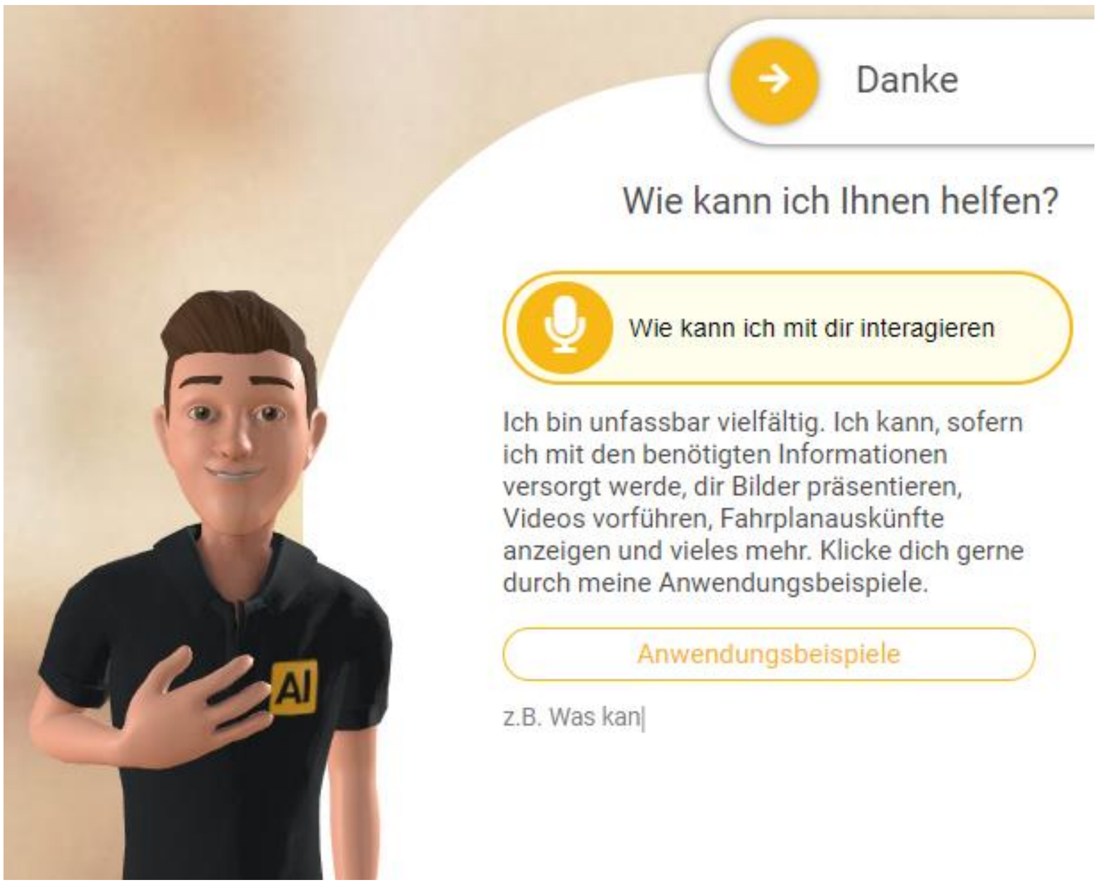


Figure 11 - Tiny Widget



Ich bin unfassbar vielfältig. Ich kann, sofern ich mit den benötigten Informationen versorgt werde, dir Bilder präsentieren, Videos vorführen, Fahrplanauskünfte anzeigen und vieles mehr. Klicke dich gerne durch meine Anwendungsbeispiele.

Figure 12 - Tiny Widget opened

3.4 Mobile view

All Widgets come with a built-in mobile view, which makes them usable on smart phones and small screens. To save space, the Avatar is not displayed, until it is opened via the *"Ask me"* button, which is shown on the bottom of the screen. The control elements are identical to the normal view.

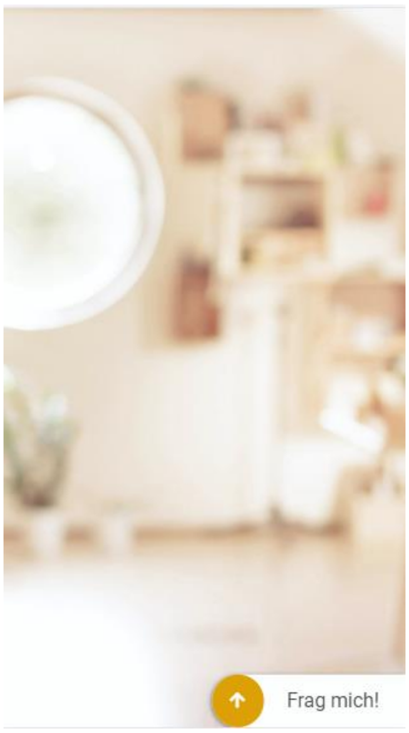
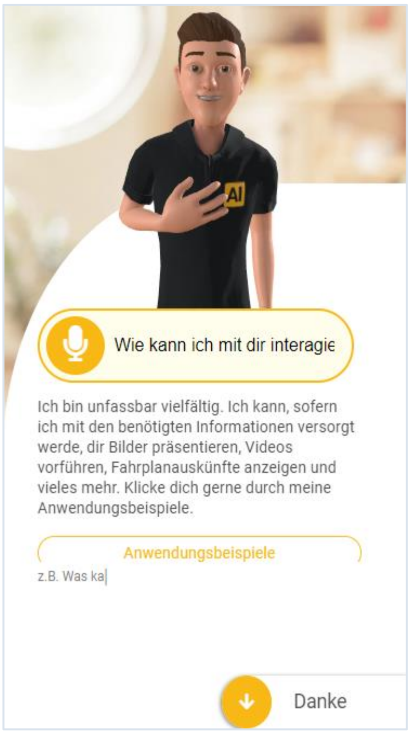
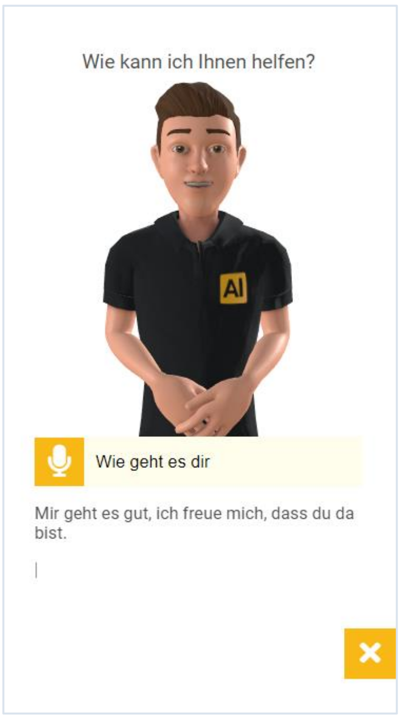


Figure 7 - Closed Tiny Widget mobile view



Ich bin unfassbar vielfältig. Ich kann, sofern ich mit den benötigten Informationen versorgt werde, dir Bilder präsentieren, Videos vorführen, Fahrplanauskünfte anzeigen und vieles mehr. Klicke dich gerne durch meine Anwendungsbeispiele.

Figure 13 - Tiny Widget mobile view



Mir geht es gut, ich freue mich, dass du da bist.

Figure 9 - Full Screen Widget mobile view

3.5 Integration

The major benefit of using the widgets is that it is extremely effortless to integrate Lailo on a website. There are only three simple steps to follow:

(1) Add the Lailo Smart Character Widget CDN in the head section of the HTML[1]. This will load all necessary scripts and manages everything regarding the Avatar and its behavior.

```
<script src="https://cdn.lorent-online.net/lailo/ai-avatar/libs/v1/widgets.js">
</script>
```

The CDN link contains a version tag (v1, which stands for version 1). There may be upcoming versions in the future. However, there will be no breaking changes in an established version. If there is no reason for updating the CDN version, it is fine to stick with that version.

(2) Add an HTML DIV Element in which the whole Widget is placed. It is reasonable to place that element just somewhere on the top level of the document body. The DIV element must have the unique id “lailo-smart-character”:

```
<div id="lailo-smart-character"></div>
```

(3) Initialize the character within the “onload” event of the window. Inside the callback function, one of widget’s initialization-functions must be called

Full Screen Widget:	Lailo.initFullScreenWidget()
Half Screen Widget:	Lailo.initHalfScreenWidget()
Tiny Widget:	Lailo.initTinyWidget()

All init functions expect an object, which passes some import information to the widget:

```
<script type="text/javascript">
      window.onload = function() {
            Lailo.initHalfScreenWidget({
                  botSecret: "12345678-1234-1234-1234-123456789012",
                  language: "de-DE"
            });
      }
</script>
```

The most important property is the “botSecret”. This is a 32-digit unique id which identifies the Character, connects it to the knowledge base and selects the correct 3D model and textures. The bot secret can be found in the Lailo Conversational UI portal[2], on the settings page of a bot.

[1] Make sure that your website structure includes the following DOCTYPE declaration as otherwise the widget might not correctly be rendered: *<!DOCTYPE html>*
[2] https://portal.lailo.ai

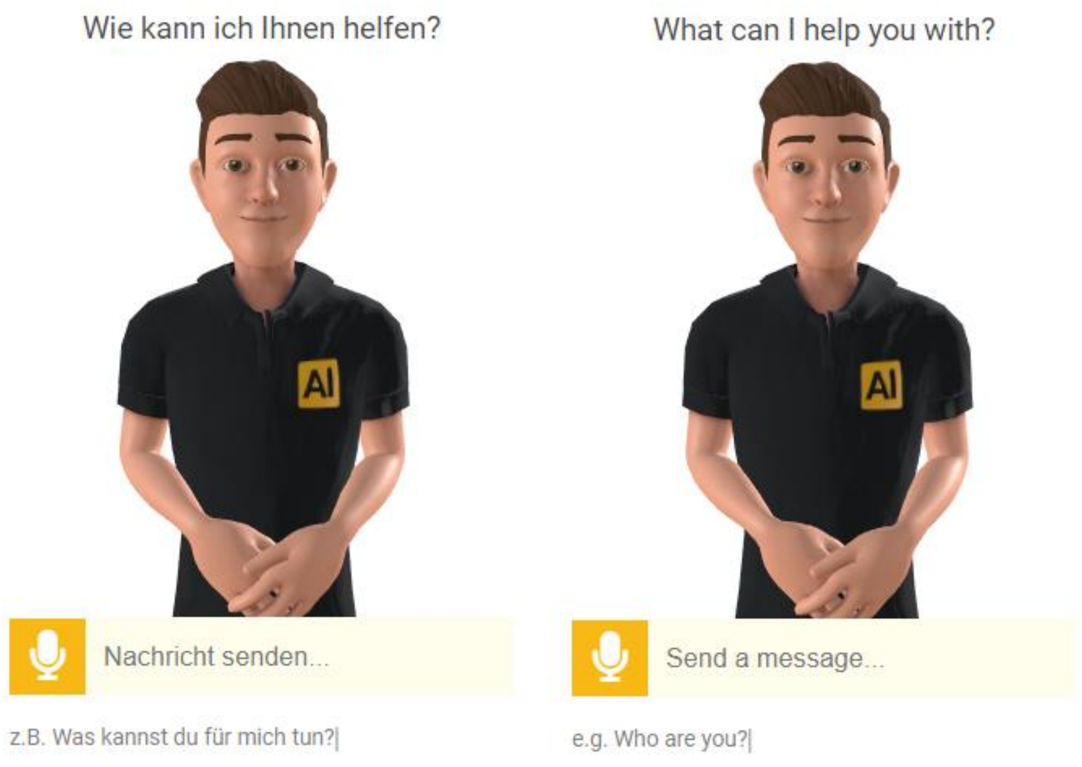
Next, a language should be specified. The initialization function expects a language code of one of the following currently supported languages:

Language	**Language Code**
English	en-US
German	de-DE
French	fr-FR

On the one hand the language influences the standard texts within the widget (see Figure 14) and on the other hand the Speech Recognition uses the language setting internally. Therefore, it is reasonable that the language setting and the used language for the bot do match. The default setting is “en-US”.

The widgets come with pre-configured texts. For example, the example questions, which are shown below the input field, show default values depending on the selected language:

German	Frag mich einfach!	z.B. Wer bist du?	z.B. Was kannst du für mich tun?
English	Just ask me!	e.g. Who are you?	e.g. What can you do for me?
French	Demandes-moi quelque chose!	p.ex. Qui est-tu?	p.ex. Que peux-tu faire pour moi?


Wie kann ich Ihnen helfen?
AI
Nachricht senden...
z.B. Was kannst du für mich tun?|
What can I help you with?
AI
Send a message...
e.g. Who are you?|

However, the initialization object also supports optional properties, that modify all texts. The following example shows all possible text modifications:

```
Lailo.initHalfScreenWidget(
{
	botSecret: "12345678-1234-1234-1234-123456789012",
	language: "de-DE",
	exampleQuestions: ["Wie heißt du?", "Was kannst du?"],
	title: "Wie kann ich Ihnen helfen?",
	inputPlaceholder: "Geben Sie eine Nachricht ein...",
	buttonTexts: {
		opened: "Schließen",
		closed: "Fragen Sie mich",
	}
});
```

The numbers in Figure 15 correspond with the following properties of the initialization object:

(1) Example questions
(2) Title
(3) Input placeholder
(4) Button texts – opened
(5) Button texts – closed

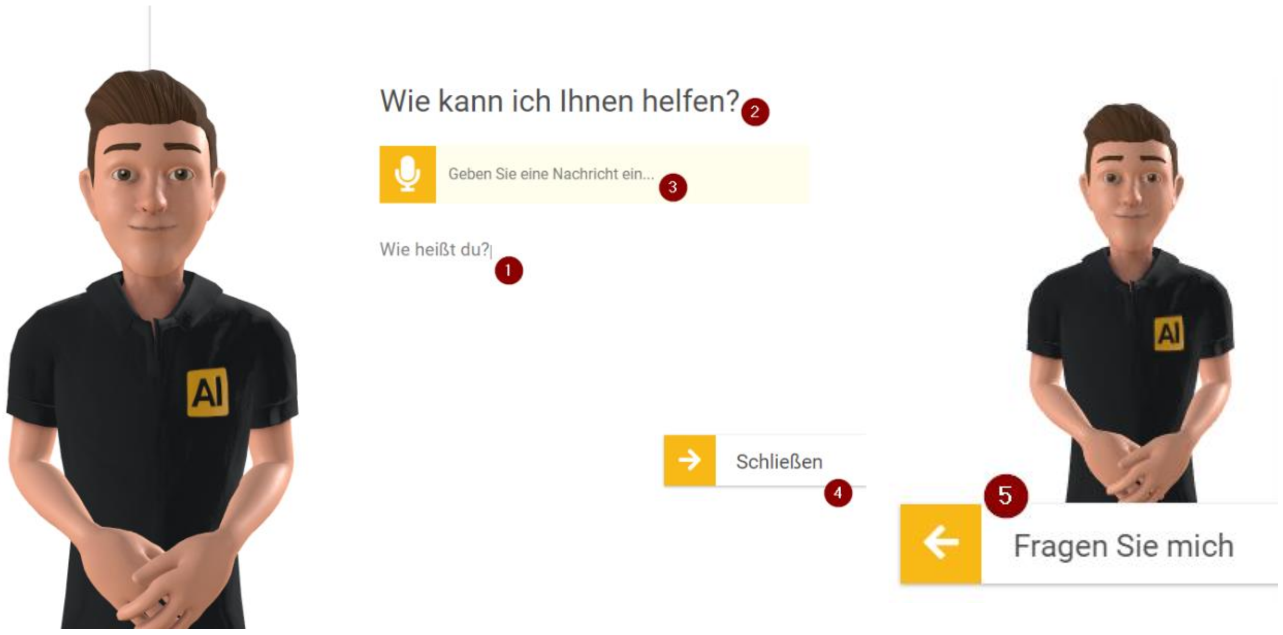


Figure 15 – Opened and closed Half Screen Widget with modified texts

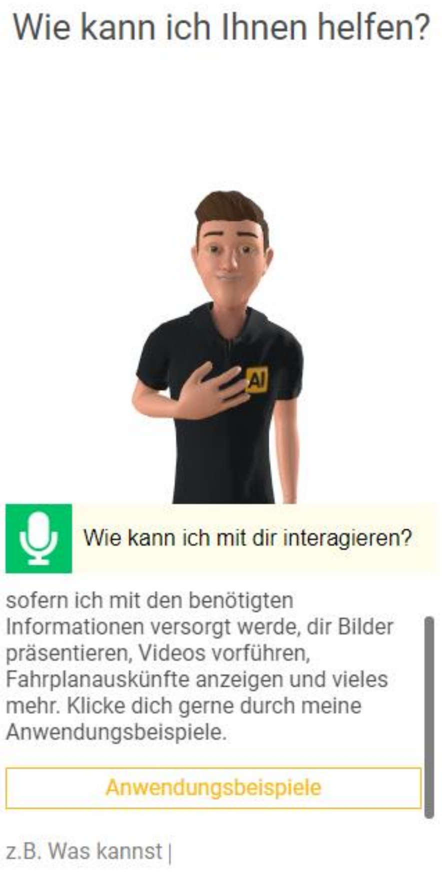
3.6 Customization

The styling of the Widgets is done in a CSS Stylesheet, which is automatically loaded during the initialization. It is therefore also accessible via the CDN. However, it is possible to override those pre-defined settings with a custom CSS file.

In the following example a stylesheet (site.css) with a single value was appended in the head-section of the website, resulting in a redesign of the button in Figure 16. The easiest way to customize the Widget is by opening it with Chrome/Firefox developer tools and tweak the stylings of the elements, until the Widget integrates well into your site.

```
Index.html:
<link rel="stylesheet" href="site.css"/>
```

```
site.css:
#lailo-smart-character  bot-control-element  {
      background-color: #00c267
}
```



Wie kann ich Ihnen helfen?

Wie kann ich mit dir interagieren?

sofern ich mit den benötigten Informationen versorgt werde, dir Bilder präsentieren, Videos vorführen, Fahrplanauskünfte anzeigen und vieles mehr. Klicke dich gerne durch meine Anwendungsbeispiele.

Figure 16 - Widget CSS customization

4 Barebone Template

If the overall design of the provided Widgets does not fit into the intended environment, another option is to use the barebone template. To understand the advantage of the template, it is helpful to understand a bit more about the internal mechanisms of the Lailo Smart Character.

There are certain events, that occur during the usage of the Avatar. For instance, after the initialization of the Character, an **onInitialised** event is fired (see Figure 17). If the user then clicks the microphone button, it takes a few milliseconds until Lailo is listening to the client. When this happens, an **onListening** event is triggered. After the client talked, their question is shown within the **onQuestionAsked** event. Internally, this question is sent to the Lailo Conversational UI platform and an answer is sent back, resulting in an **onAnswer** event. At this point in time, the Avatar starts answering the question, until finally the **onAnswered** event shows that the Character stopped talking. In case of an error the **onError** event is used (not shown in figure).

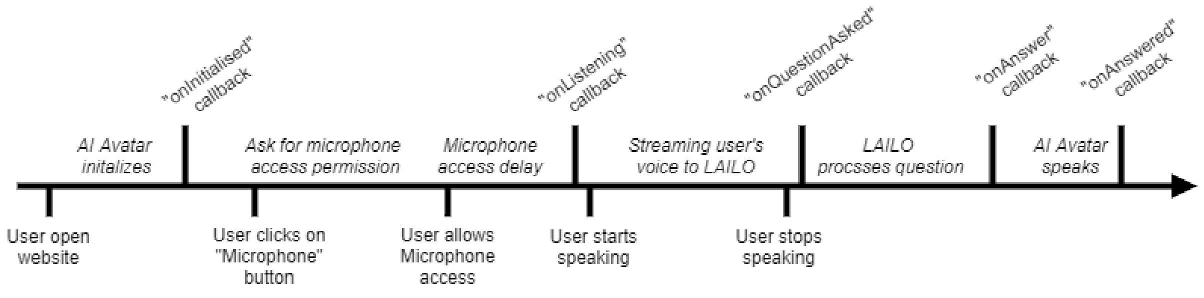


Figure 17 - Character events

Besides the event handling, there are certain elements which are necessary or convenient for using Lailo. For example, a *Listen* button for starting the Speech Recognition and a text box for the answer.

If the frontend developer decides to use the *Character API*, as described as the fourth scenario in chapter 5, they will have to call built-in functions to start listening to the client. The following events can then be used to change the style of the button ("listening mode") and fill in the answer in the mentioned text box. This allows for the frontend developer to design a highly individual solution.

However, in many scenarios it is quite clear what should happen. E.g., if the user clicked the listen button, Lailo should listen and if there is an answer, it should be displayed inside an "answer field". Because of that, the ***Barebone Template*** does a lot of the described event handling. At the same time, the frontend developer is free to code a custom solution, with a custom design.

4.1 Integration

To get started with the Barebone Template, there are only four steps to follow:

(1) Add the Lailo Smart Character Template CDN in the head section of the HTML. This will load all necessary scripts.

```
<script src="https://cdn.lorent-online.net/lailo/ai-avatar/libs/v1/template.js">
</script>
```

The CDN link contains a version tag (v1, which stands for version 1). There may be upcoming versions in the future. However, there will be no breaking changes in an established version. If there is no reason for updating the CDN version, it is fine to stick with that version.

(2) Add all HTML elements that are needed to display and interact with the character. Some of the control elements are optional. E.g., it is possible to use a dedicated "Send question" button, if the user used a question input field. However, the user can also just use the "*Enter*" key to send a message, so a button is not necessary and therefore a design decision.

Element	**Example**	**Mandatory**
Character	<div id="lailo-smart-character"></div>	Yes
Input text field	<input type="text" id="lailo-text-input" />	Yes
Answer field	<div id="lailo-answer-text"></div>	Yes
Microphone button	<button id="lailo-microphone-button">Listen</button>	Yes
Send question button	<button id="lailo-send-text-button">Send</button>	No
Error message field	<div id="lailo-error-field"></div>	No

(3) Initialize the template within the "onload" event of the window. Similar to the Widgets some parameters must be passed to the template.

The most important property is the "botSecret". This is a 32-digit unique id which identifies the Character and connects it to the knowledge base, the 3D model, and the textures. The bot secret can be found in the Lailo Conversational UI portal[3], on the settings page of a bot.

```
<script type="text/javascript">
        window.onload = function() {
                Lailo.initBareBone(
                {
                        botSecret : "12345678-1234-1234-1234-123456789012",
                        language : "de-DE",
                        cameraAdjustment : "longshot",
                        characterCanvasId : "lailo-smart-character",
                        userInputId : "lailo-text-input",
                        botOutputTextFieldId : "lailo-answer-text",
                        microphoneBtnId : "lailo-microphone-button",
                        sendTextBtnId : "lailo-send-text-button",
                        botOutputErrorFieldId : "lailo-error-field"

                });
        }
</script>
```

Next, a language should be specified. The init function expects a language code of one of the supported languages. The language is passed to the Speech Recognition and strongly influences the quality of the language understanding.

[3] https://portal.lailo.ai

Language	**Language Code**
German	de-DE
English	en-US

In contrast to the Widgets, the template allows to set the camera position. With the argument "cameraAdjustment" the camera can be set to one of the following settings:

- closeup
- near
- medium
- longshot
- american

The following parameters, passed to the init function, are used to pass in the IDs of the control elements. As shown in (2), the elements are of different type (div, input, button) and are partly optional.

(4) If everything is hooked up correctly, the basic functionality will work. I.e., the character will be displayed in the given HTML DIV element, a click on the *Listen* button will start Speech Recognition, a text that was written in the question input will be send to Lailo when clicking on the *Send* button and the answer or error will be displayed in the answer or error text field, respectively.

The remaining task for the frontend developer is to style all control elements and position them at the preferred places. As mentioned at the beginning of this chapter, there are several events that occur during operation. To achieve an interactive experience for the user, CSS classes are added and removed from the used HTML elements depending on the current state.

There are three important CSS classes that are used for the *Listen* and *Send question* buttons:

- "lailo-preparing": Indicates that Lailo is currently busy with a process, i.e. preparing to listen to the user or waiting for processing the question.
- "lailo-listening": Shows that Lailo is currently listening to the user.
- "lailo-answering": Is added when the Lailo AI Avatar is currently speaking.

5 Character API

For a highly customized integration of the Lailo AI Avatar, the frontend developer can use the Character Application Programming Interface (API). This enabled the developer to interact and display the Avatar anywhere on the website and gives freedom on the used control elements. Also, it is easily possible to act on the incoming events during the conversation with Lailo.

In practice, a frontend developer who uses the LAILO AI Avatar needs to perform the following steps:

1. Include the Avatar's main JavaScript file on the website
2. Provide a HTML division element (div) in which the Avatar will be displayed
3. Initialize the Avatar
4. Set up control elements that allow a user to interact with Lailo

The control elements (see Figure 18) are independent from the actual Avatar and completely customizable. They only use the Character API which gives the frontend developer the chance to integrate the Avatar in anyway it fits the situation. Please note, that in chapter Best practices suggestions for a suitable integration are made.

One key element of LAILO is the possibility to directly talk to the Avatar. To do that, the user can grant access to a microphone. The audio is then streamed to a Speech Recognition server on a secured line. To guarantee a nice user experience, it's recommended to display the user's question in an appropriate space near the control elements.

Each LAILO – AI Avatar has his own "*BotSecret*" which uniquely identifies the Avatar. During the initialization of the Avatar, which is described in chapter 5.1**Error! Reference source not found.**, this s ecret will become important. It can be seen and changed in the *Settings*-section on the LAILO – Conversational UI Platform, as shown in Figure 19.

Another security mechanism, that is used internally, is the *Communication Contingent*. To prevent LAILO from abusive usage, and to protect our clients from unexpected costs, the AI Avatar can only be used until a certain limit. This limit resets monthly and can also be inspected in the *Settings-section* in Figure 19.

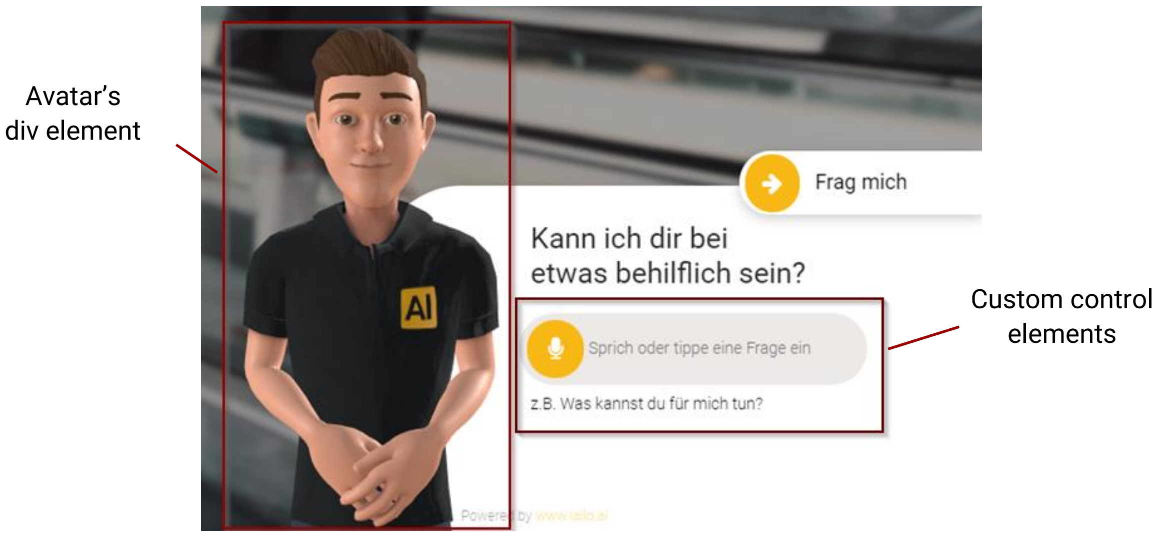

Avatar's div element
Frag mich
Kann ich dir bei etwas behilflich sein?
Sprich oder tippe eine Frage ein
z.B. Was kannst du für mich tun?
Custom control elements
AI
Powered by www.lailo.ai

Figure 18 – LAILO - AI Avatar integration: https://www.lailo.ai

Figure 19 – LAILO | Conversation UI Platform – Settings

5.1 Integration

The Avatar's main JavaScript is can be included via a CDN:

```
<script src="https://cdn.lorent-online.net/lailo/ai-avatar/libs/v1/character.js">
</script>
```

After that, the Character API registers itself on the window object of the browser and is accessible in a normal HTML JavaScript tag (<script>).

```
window.avatar.<api-function>
```

In the following chapter, all API functions will be introduced. For normal usage, it is crucial to initialize the Avatar by calling the "*init*" function. It is recommended to call this function while loading the website, to prevent longer waiting time for user before the first usage.

5.1.1 Initialization

Description

Initializes the AI Avatar.

Function Signature

```
init(callbacks: Callbacks, domElement: HTMLDivElement, botSecret: string,
language: string = “de-DE”): void
```

Example

```
window.avatar.init(
    {
        onInitialised,
        onQuestionAsked,
        onAnswered,
        onAnswer,
        onError,
        onListening
    },
    document.getElementById("corporateAvatar"),
    "12345678-1234-1234-1234-123456789123",
    “en-US”
);
```

5.1.2 Callbacks

The Avatar provides six callback functions, that are called on certain events. It is important that all callbacks are defined, the functions are called exactly how they are shown here, and the functions are provided in the exact order like shown in the example above.

To better understand what each of the callbacks are used for, refer to the diagram in Figure 20 and Figure 21.

`onInitialised(data: object): void`

Is called when the initialization of the Avatar has successfully finished. The data object contains the DirectLine4 object, which is used for the communication with the LAILO | Conversational UI Platform, an "okay"-Flag, and the view-object. These objects are only needed for an advanced usage of the Avatar and can otherwise be ignored. However, the callback itself might be useful to activate user controls.

`onListening(): void`

Is called after the *listen()* function (c.f. chapter 5.1.3.1) is called and the audio stream is actually send to the Speech Recognition server. This callback should be used to give the user a feedback when they should start speaking.

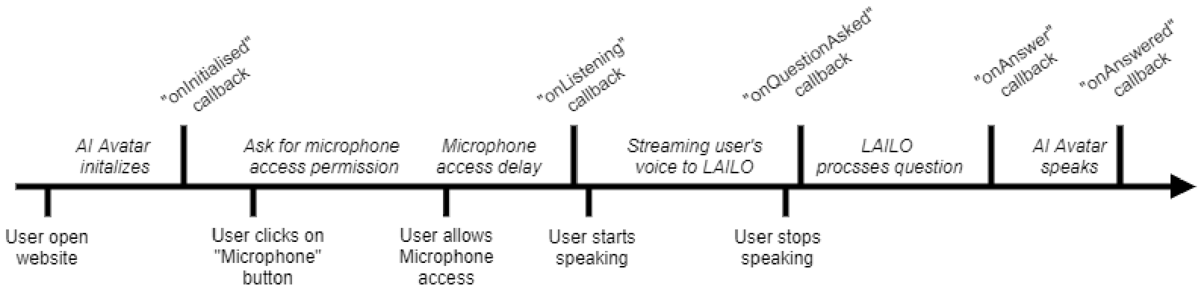


Figure 20 – Session Diagram for Microphone input

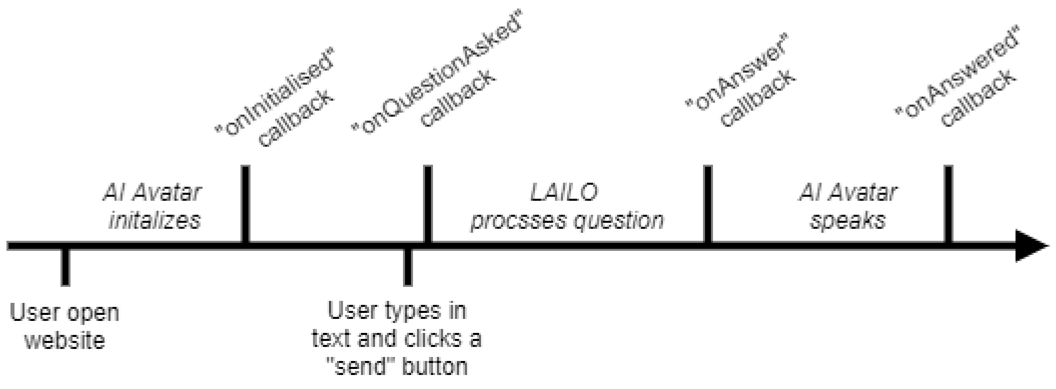


Figure 21 – Session Diagram for Text input

`onQuestionAsked(data: string): void`

Is called after Speech Recognition has returned the user's transcribed sentence. Data contains theactual text which can be displayed in the User Interface. After Speech Recognition the user's message is

4 DirectLine is a communication protocol. For more information refer to the official documentation: https://docs.microsoft.com/de-de/azure/bot-service/rest-api/bot-framework-rest-direct-line-3-0-concepts?view=azure-bot-service-4.0

directly sent to the LAILO – Conversational UI Platform. If the user does not say anything within a few seconds, the Speech Recognition is automatically aborted. In this case, the onQuestionAsked callback is also fired, but with an empty data object.

`onAnswer(data: AvatarAnswer): void`
Is called when the LAILO – Conversational UI Platform sent an answer to a prior "*send*" or "*listen*"-function call. After that, the Avatar automatically starts responding to the user. The AvatarAnswer object contains a field "text" which holds the text that the Avatar says to the user. If an action attachment was set for the triggered answer, the AvatarAnswer also holds an "actions" object. For more information on that, see section 5.2.8.

`onAnswered(): void`
Is called after the Avatar finished responding to the user. This callback can be used to reactivate user controls.

`onError(error: AvatarError): void`
Is called when an error occurs. The error object contains the properties "errorCode", "message" and "interrupt". The interrupt flag indicates if the ongoing process (e.g. streaming audio the Speech Recognition server) is interrupted. If the process is interrupted, it is reasonable to reset frontend elements to their default/idle mode. For more information, the chapter 5.3.

domElement
In order to integrate the Avatar in a website, a div-Element must be available on the website where the Avatar shall be placed. The size of this element determines the initial size of the Avatar. Pass a reference to this div element to the init function, e.g. with the *document.getElementById()* function.

BotSecret
To identify and authenticate the Avatar at the LAILO – Conversational UI Platform, a bot secret is needed. This identifier is automatically generated and has the form of a GUID. The secret will be provided to the customer, after the bot has been created and can be obtained in the *Settings* menu of the Bot on https://portal.lailo.ai.

Language (optional)
Sets the language code (RFC 4646) for speech recognition. Default is German ("de-DE"), currently supported alternative: "en-US".

5.1.3 Functional interface

5.1.3.1 Listen Function

Description
Turns on the user's microphone and streams the audio input to the Speech Recognition server. After the user's input is understood a *onQuestionAsked callback* (cf. chapter 5.1.2) is triggered. Use this function after the user clicked some kind of a microphone button.

Function signature
```
listen(): void
```

5.1.3.2 Send Function

Description
Sends a message to the LAILO - Conversational UI Platform. This function can be used to communicate with the Avatar if no microphone input is desired or available. Of course, a text input field and a "send"-button are necessary. This function could also be used for communication with the LAILO Conversational UI Platform after certain events. For a reasonable user experience, it is not recommended to use the API without any user interaction.

Function signature

```
send(text: string): void
```

5.1.3.3 Interrupt Function

Description

Interrupts either the audio stream to the Speech Recognition server or stops the Avatar from speaking. E.g. this function can be used when the user clicked on the Avatar or a microphone button yet again.

Function signature

```
interrupt(): void
```

5.1.4 Visuals Adjustments

5.1.4.1 IsVisible Property

Description

Indicates if the Avatar is currently shown or hidden. The Avatar is shown by default.

Property information

```
isVisible : boolean
```

5.1.4.2 Hide Function

Description

Hides the Avatar.

Function signature

```
hide(): void
```

5.1.4.3 Show Function

Description

Shows the Avatar.

Function signature

```
show(): void
```

5.1.4.4 Resize Function

Description

Resizes the Avatar canvas. The selected camera shot (cf. 5.1.4.5 for more information) is kept, so that the resulting Avatar size may be less than the given width/height.

Function signature

```
resize(width: number, height: number): void
```

5.1.4.5 AdjustCamera Function

Description

Adjusts the camera settings. Possible values are:

- *closeup*
- *near*
- *medium*
- *longshot*
- *american*

Function signature

```
adjustCamera(camera: string): void
```

5.1.4.6 KnownGestures Function

Description

While responding, the LAILO | AI Avatar automatically analyzes the answer and shows a matching gesture. All known gestures can be retrieved with the *"knownGesture"*-function.

Function signature

```
knownGestures(): Array<string>
```

5.1.4.7 InitiateGesture Function

Description

This function triggers a gesture independent from a normal user conversation. For example, the Avatar might wave or point somewhere on the website, when the user clicks on a button or after a timeout without user interaction has passed. Valid parameter values are all gestures as they are returned by the *knownGestures()*-function.

Function signature

```
initiateGesture(gesture: string): void
```

5.2 Best practices

5.2.1 General

As already mentioned above, the Avatar pack only provides the actual Avatar model and an API to interact with the AI Avatar. However, in this chapter best practice integration approaches will be presented, that should guarantee a good user experience.

In general, the *onError* callback (c.f. chapter 5.1.2) is called whenever an error occurres. Depending on the actual error, the frontend developer should react accordingly. All errors are described in chapter 5.3 in detail.

5.2.2 Speaking vs. Writing

A main feature of LAILO – AI Avatar is the possibility to interact with the Avatar with natural language. The *listen* API function (c.f. 5.1.3.1) starts the language understanding and streams the user's voice to a language understanding system. For that, a built-in browser function (WebRTC) is used. The default setting in every browser is to ask the user for permission:

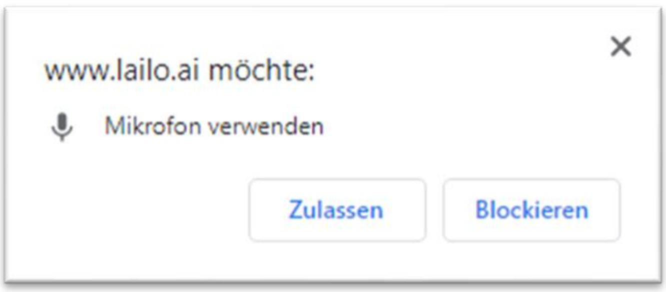


Figure 22 – Permission for microphone usage (Chrome)

If the user denies the access, a *MicrophoneAccessDenied* event will be fired on the *onError*-callback function. In this case it's a good idea to disable the microphone button (see Figure 23). Irrespective of that, show an input field. When the user enters a text there an presses the *Enter* key or clicks a *Send*-

button (not shown in Figure 23), the *send*-function should be used to communication with the LAILO | Conversational UI Platform.

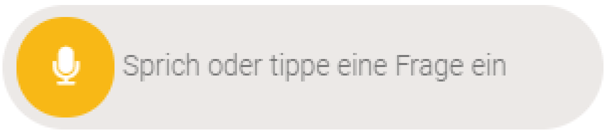


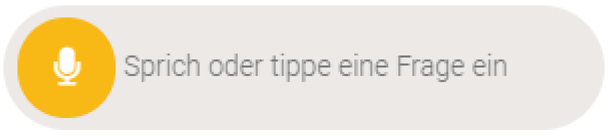
Figure 23 – Microphone Button and Input Field

Note: The *send* function can not only be triggered from a user's text input. A click on a button can also send a message to the Character's API. This leads to lots of interesting new features on a website. For example, LAILO can be triggered by clicking on a navigation link on a Single Page website, or by clicking on a certain HTML div-element on the website. That's actually the part, where the UI designer and frontend developer can get very creative.

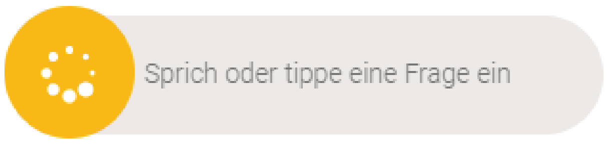
5.2.3 Microphone Access Delay

The user will experience an initial delay, before the microphone's audio stream will be transferred to the language understanding system. On Chrome and Firefox this delay is almost negligible. However, on Safari and older hardware the delay can be in the range of hundreds of milliseconds up to few seconds, which might lead to a bad user experience and speech to text quality, because the user could start to speak too early. Because of that, the *onListening* event will be called, when the audio transmission has started. Therefore, the website should show something like a loading indicator after the user clicks the *listen* button and change this to a symbol which expresses that the service "is listening" once the microphone stream is ready. The following examples shows how this is done on the https://www.lailo.ai website:

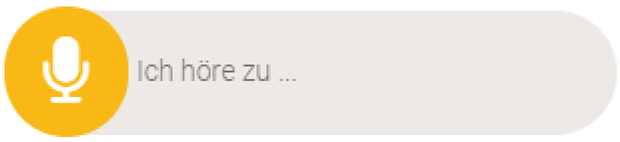
Idle



After click on "start listen" button → Loading indicator

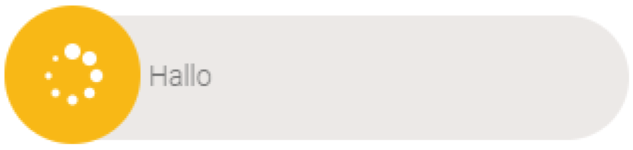


After "onListening" event → bigger Microphone button indicates "listening"



5.2.4 Display user's question

When the Speech Recognition system returns the user's question, the *onQuestionAsked* event will be called. It contains the text which was understood. This text should be displayed to the user. Firstly, this will give a nice user experience. Secondly, if the user is in a crowded area or didn't speak clearly, the Speech Recognition might have problems to understand. If the user then sees that he or she maybe was misunderstood he or she might try to speak louder/more clearly.

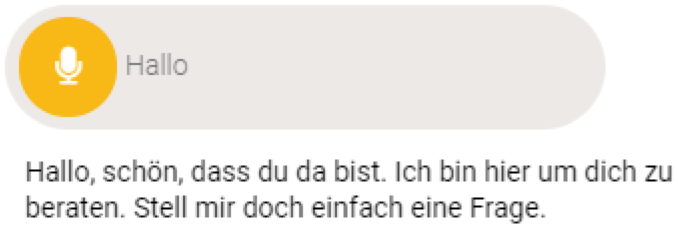

Hallo

5.2.5 Avatar Response Delay

After the user wrote a question or used the Speech Recognition, there will be another short delay, until the answer is a) received by the LAILO – Conversational UI Platform, b) analysed and interpreted, c) answered and d) sent back to the user's browser. It is good practice to show a loading indicator to the user during that delay.

5.2.6 Avatar Response

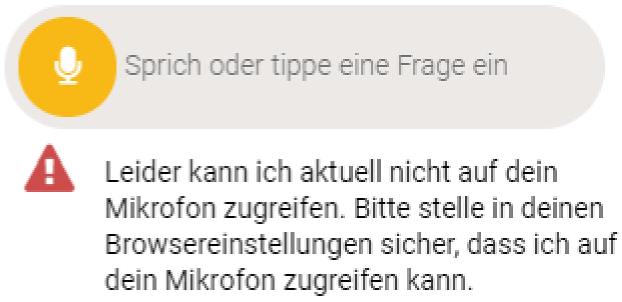
When the Avatar starts to answer the user's question, the *onAnswe* callback is fired. An object is passed with the response text, which can be displayed to the user:


Hallo

Hallo, schön, dass du da bist. Ich bin hier um dich zu beraten. Stell mir doch einfach eine Frage.

5.2.7 Error handling

It might happen that the Avatar is not able to listen or respond to the user. There is a detailed list of possible errors in chapter 5.3. The *ErrorCode* in the *onError* callback can be used to show individual error messages to the user:


Sprich oder tippe eine Frage ein

Leider kann ich aktuell nicht auf dein Mikrofon zugreifen. Bitte stelle in deinen Browsereinstellungen sicher, dass ich auf dein Mikrofon zugreifen kann.

Some problems might be fixable by the user. E.g. the user could check their internet connection or microphone access. Other problems are more severe, e.g. if the browser does not support WebGL, and therefore the Avatar cannot be displayed at all. The frontend developer should handle all errors which are described in the following chapter and should decide which error message is appropriate in the given situation. Never return too technical error messages to the user but try to give easy information about what happened and also try to give the user advice what he or she can do to solve the issue. It is also important to consider the status of the buttons and text boxes to provide a good user experience, even in case of an error.

Generally, errors can happen in a couple of situations. However, Figure 24 shows an example how the “onError” callback will be called after the user denied microphone access.

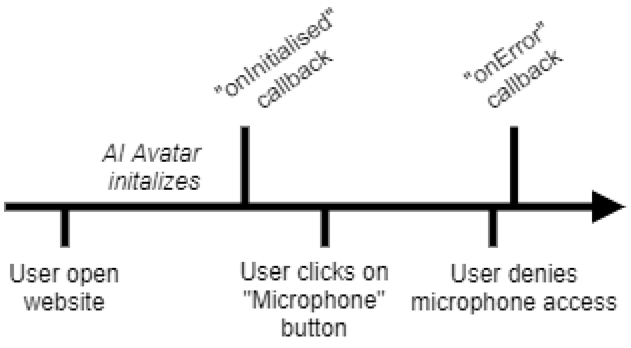
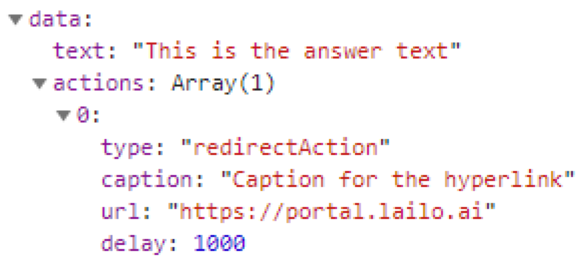


Figure 24 – Session Diagram for MicrophoneAccesDenied error

5.2.8 Action Attachment

Another great possibility to enhance the basic features of LAILO | AI Avatar is the usage of the *Action* attachments. This attachment type can be edited in the LAILO | Conversation UI Platform (see Figure 26). Select a bot and click on the attachment icon (1) and then on the Robot tab (2). The *Action* is a free text meant to hold data in a JSON format. E.g. it could be used to trigger a redirect to another area on the website or to automatically fill in some information in web form. Actually, it could be used for any kind of automatable action on the website. The action is sent via the *“onAnswer”* callback (section 5.1.2), as shown in Figure 25 – Example for data-object in the "onAnswer" callbackFigure 25.



```
▼data:
   text: "This is the answer text"
  ▼actions: Array(1)
    ▼0:
       type: "redirectAction"
       caption: "Caption for the hyperlink"
       url: "https://portal.lailo.ai"
       delay: 1000
```

Figure 25 – Example for data-object in the "onAnswer" callback

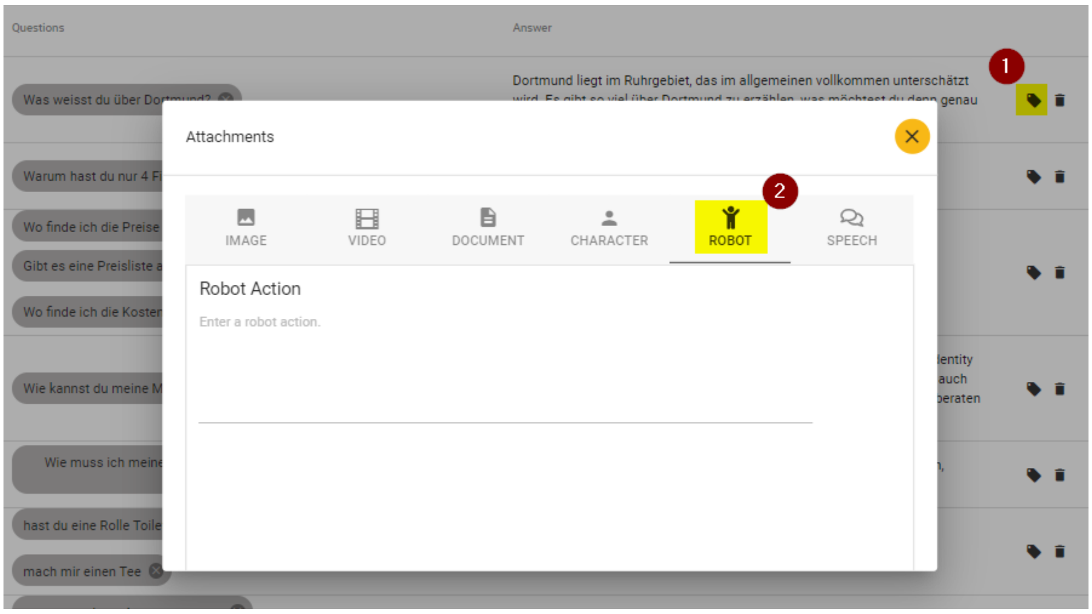


Figure 26 – LAILO | Conversational UI Platform – Action Attachment

5.2.9 No user question recognized

When the user clicks the microphone button, but does not say anything, Speech Recognition will be aborted after a few seconds automatically. It might also be the case, that it was not possible to understand the user's question, because of loud background sounds. In this case the "onQuestionAsked" callback is called with an empty *data* object. To achieve a good user experience, it is highly recommended to visually illustrate that the Speech Recognition is not active anymore.

5.3 Errors

5.3.1 Comunication Errors

Speech Recognition (Speech-To-Text)

ErrorCode	ErrorMessage	Description / Possible Cause / Solution
CommunicationServer-InitFailed	Could not initialize the connection the communication server.	• Internet connection broken just after site load (not very likely) • Communication Server not reachable • Internal error of the LAILO| Conversational UI Platform • Communication Server connection rejected • Communication Contingent exceeded ⇨ Auto. retry after 1 minute
NoCommunication-ServerConnection	Could not connect to communication server.	• Communication Server not reachable • Internal error of Communication Server • Request contingent exceeded ⇨ Auto retry after 1 minute
RefreshCommunication-ServerTokenFailed	Could not refresh communication server connection.	• Internet connection broken • See *CommunicationServerInitFailed* error code • Request contingent exceeded ⇨ Auto. retry after 1 minute
CommunicationServer-ConnectionClosed	The communication server connection has been closed.	• Internet connection broken ⇨ Reconnect after 1 minute

Bot Communication

ErrorCode	ErrorMessage	Description / Possible Cause and Solution
DirectLineTokenFailed	The bot communication failed.	• Internet connection broken • Bot offline • Request contingent exceeded ⇨ Auto retry after 1 minute
RefreshDirectLine-TokenFailed	Could not refresh the communication with the bot.	• Internet connection broken • Bot offline • Request contingent exceeded ⇨ Auto. retry after 1 minute
DirectLineKeep-AliveFailed	Could not refresh the connection to your bot.	• Internet connection broken • Bot offline • Request contingent exceeded ⇨ Next call to bot may take longer than usually (~ 2 seconds)
LowBandwidth	Bandwidth is too low for Speech-to-text.	• The upload time of a single audio chunk was greater than 1 second or the last uploaded audio chunk took 0,5 seconds longer than the first audio chunk. • Both conditions indicate a bad internet connection and it's highly recommended to disable Speech-to-Text. ⇨ Disable Microphone button and display text input field if that's an option.
BotCommunication-Failed	The communication to the bot failed.	• The DirectLine communication to the bot service failed. This might indicate that a used service is not accessible. However, it's more likely

		that the user's internet connection is broken. ⇨ Notify the user or hide Avatar
NoAttachmentsInActivity	Received a message from bot without audio and LipSync attachments.	• Necessary information for the Character's answer is missing in the response.

5.3.2 Media Access and Microphone

ErrorCode	ErrorMessage	Description / Possible Cause and Solution
MicrophoneAccessDenied	Microphone access denied.	Check browser settings and allow microphone access.
MicrophoneNotFound	Microphone not found.	No microphone was detected or microphone is muted in the system settings.
NoGetUserMediaAccess	No get UserMediaAccess.	Couldn't get microphone access from the browser. The reason might be, that the user's device does not have any media devices or the browser's default setting doesn't allow media access at all.

5.3.3 General Avatar Errors

ErrorCode	ErrorMessage	Description / Possible Cause and Solution
InvalidCameraAdjustment	The given camera shot is unknown.	Camera shot is unknown. Only use one of the following camera shots: • *closeup* • *near* • *medium* • *longshot* • *american*

InvalidCanvasSize	Cannot resize canvas with the given values.	Width and height must be given as parameters for the *resize* function call. Values must be greater than 0.
MessageToBotEmpty	Cannot send an empty message to the bot.	There must be provided a message to the *send* function.
InvalidAction	The defined action for an answer is not in a valid format.	The action that was defined in the LAILO portal is not in a correct format (JSON). Thus, it's not forwarded to the Avatar. Read more about Actions in section 5.2.8.
WebGLNotSupported	Avatar cannot be displayed because of missing WebGL support.	Avatar could not be loaded due to missing WebGL support in the used browser. Try to activate WebGL or switch to another supported browser.