User Manual

Interactive App Tutor

Contents of this User Manual

Introduction

- **What is the Interactive App Tutor:** The Interactive App Tutor is a support tool that integrates seamlessly with existing software applications to provide step-by-step guidance, interactive tutorials, in-app documentation, and contextual help to efficiently onboard your users onto your software. It is designed to assist users in learning and navigating software functionalities more efficiently, enhancing the onboarding experience through real-time, in-app assistance.
- **Purpose:** The user manual is designed to guide administrators and users through the setup, configuration, and usage of the Interactive App Tutor system, which provides in-app tutorials and resources for effective onboarding.
- **Overview**: This manual covers the key features that the Interactive App Tutor has to offer. It includes interactive guides, checklists, documentation access, search tools, and help icons. It also details the customization options available to administrators for tailoring the onboarding experience.
- **System Requirements**: The Interactive App Tutor system is compatible with applications using React for the frontend with a Router component, requires HTTPS and WebSocket communication with the backend, and supports integration via the Django-based admin interface.

02

Installation and Setup

Installation and Setup
Frontend

Clone the Code from GitHub

To get started with the frontend of the Interactive App Tutor system, you need to clone the repository from GitHub. Follow the steps below:

1. Open a terminal on your local machine.
2. Navigate to the directory where you want to store the project:
   ```
   cd /path/to/your/directory
   ```
3. Clone the repository using the following command:
   ```
   git clone https://github.com/gibbosphere/interactive-app-tutor-frontend.git
   ```
4. Once the cloning is complete, navigate to the project directory:
   ```
   cd interactive-app-tutor-frontend
   ```

Now you have the frontend code on your local machine, and you're ready for the next step in the setup process.

View the Interactive App Tutor in action

Part of the code you have cloned from GitHub is simply a demo application. It is used to give you a quick preview of the Interactive App Tutor in action as it overlays on top of this demo application. To see the Interactive App Tutor in action, run `npm start` in the command line and navigate to http://localhost:3000 in your browser.

Copy the necessary Code

Copy the entire components folder and paste it into your application root directory. (remember the rest of the code was simply a demo app and is not required in your application)

Installation and Setup
Frontend

Use the InteractiveAppTutor in your code

Import and insert the InteractiveAppTutor React component into your application code. Ensure the demoMode prop is set to false - rather than using the local demo_app_tutorial_data, the frontend will then make API calls to the backend to receive all tutorial and resource data.

```
/** Your other App.js imports **/
import React, { useState } from "react";
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import { usePersistantState } from "./components/hooks";
import InteractiveAppTutor from "./components/InteractiveAppTutor";

const App = () => {
  /** Suggested state variables **/
  const [tutorialActive, setTutorialActive] = React.useState(false);
  const [documentationOpen, setDocumentationOpen] = React.useState(false);
  const [documentationSideToolEnabled, setDocumentationSideToolEnabled] =
React.useState(true);
  const [resourceCircleEnabled, setResourceCircleEnabled] = React.useState(true);

  return (
    /** Be sure to use Router from react-router-dom **/
    <Router>
      /** All your other application content first **/
      /** Your InteractiveAppTutor last **/
      <InteractiveAppTutor
        tutorialActive={tutorialActive}
        exitTutorial={() => setTutorialActive(false)}
        tutorialLogoSrc="/images/iNethiLogoWhite.png"
        documentationOpen={documentationOpen}
        toggleDocumentationOpen={toggleDocumentationOpen}
        documentationSideToolEnabled={documentationSideToolEnabled}
        resourceCircleEnabled={resourceCircleEnabled}
        resourceCircleIconSrc="/images/iNethiLogoWhite.png"
        resourceCirclePos={{ positionY: "bottom", positionX: "right" }}
        resourceCircleSize={60}
        resourceCircleDistFromOuter={20}
        resourceCircleBorder={"2px solid white"}
        openDocumentation={() => setDocumentationOpen(true)}
        demoMode={false}
      />
    </Router>
  );
};
```

Installation and Setup
Frontend

Start your React application

Finally, start your React application using `npm start`. (There will initially be no tutorial data)

Add your own tutorial and resource data using the interactive-app-tutor-backend. Setup of the backend follows.

Installation and Setup
Backend

Clone the Code from GitHub

To get started with the backend of the Interactive App Tutor system, you need to clone the repository from GitHub. Follow the steps below:

1. Open a terminal on your local machine.
2. Navigate to the directory where you want to store the project:
   ```
   cd /path/to/your/directory
   ```
3. Clone the repository using the following command:
   ```
   git clone https://github.com/gibbosphere/interactive-app-tutor-backend.git
   ```
4. Once the cloning is complete, navigate to the project directory:
   ```
   cd interactive-app-tutor-backend
   ```

Now you have the backend code on your local machine, and you're ready for the next step in the setup process.

Start the Server

1. Navigate into the backend directory that stores the manage.py folder:
   ```
   cd backend
   ```
2. Start running the server:
   ```
   python manage.py runserver
   ```

With the backend running on http://localhost:8085, the frontend can now query the backend API for tutorial and resource data that you can manually create and customize via the admin page.

Access the Admin Page

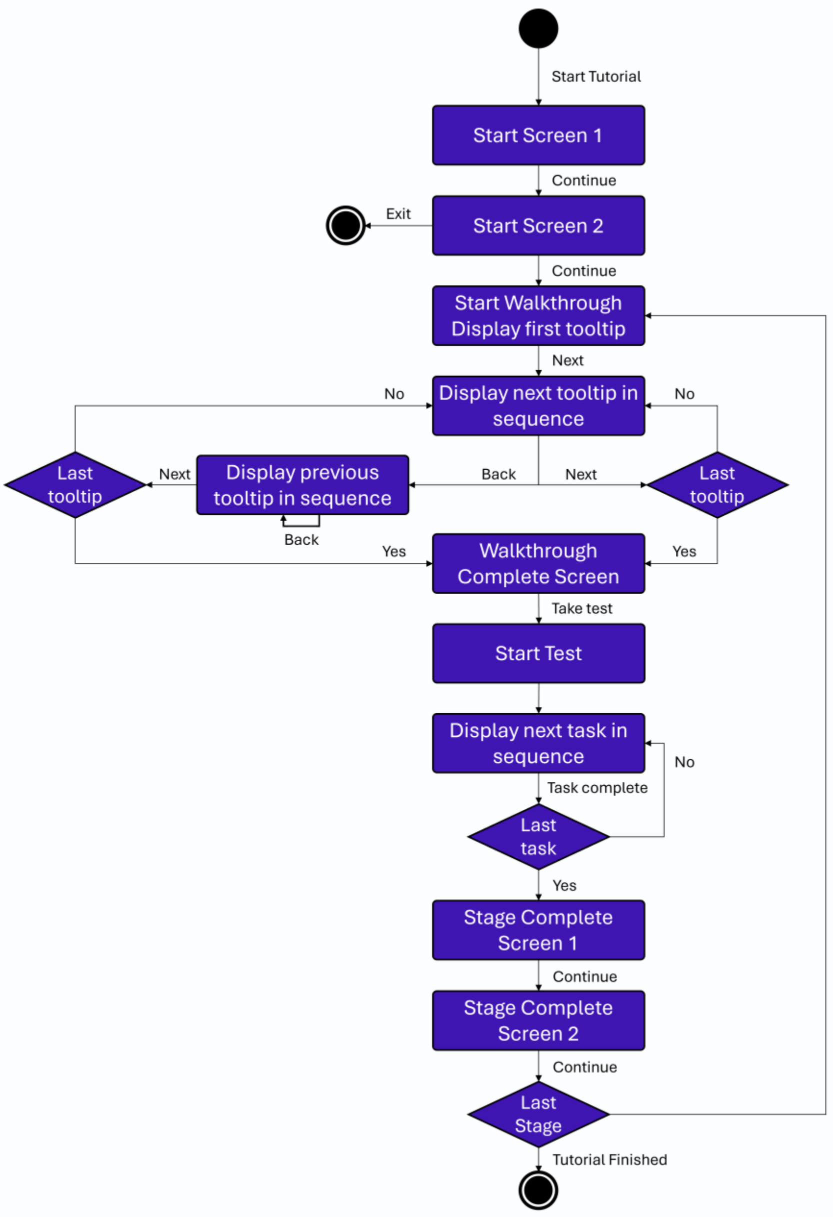
1. Create an admin user to be able to access the admin page
   ```
   python manage.py createsuperuser
   ```
2. Add, edit, or delete data via the admin page at http://localhost:8085/admin, logging in with the credentials you set in the previous step.

03

The Features

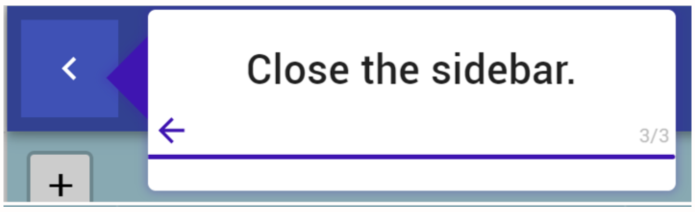
Tutorial

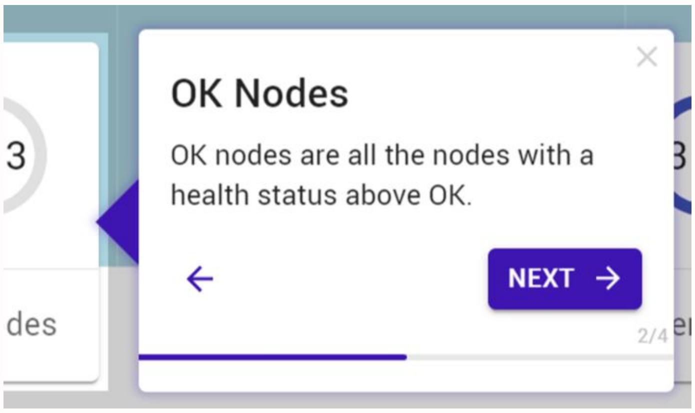
A tutorial is composed of multiple stages, each consisting of a walkthrough and a test. The walkthrough guides the user through various features and functionalities of the underlying NMI using sequential tooltips. After completing the walkthrough, the user takes a test to reinforce the knowledge gained. The test includes a set of tasks that the user must complete, with their progress visualized through a task checklist. The figure below shows the flow of a Tutorial represented as an activity diagram.



Tutorial
Walkthrough

A **walkthrough** is used to walk the user through features of the application. Each walkthrough is made up of **multiple tooltips**. A Tooltip is a small, pop-up element that points to and provides users with contextual information or instructions about a specific feature. It can either display static information (informative) or prompt users to take an action (action). An action and informative tooltip are shown in the figure below




OK Nodes

OK nodes are all the nodes with a health status above OK.

The Tooltip component takes properties (props) that define its appearance, logic, and positioning. They include:

1. type: There are two types of tooltips: informative and action. Informative tooltips provide details about the highlighted application feature(s), advancing with a "next" button. Action tooltips require the user to perform a specific action to move to the next step.

2. title: The title of the tooltip.

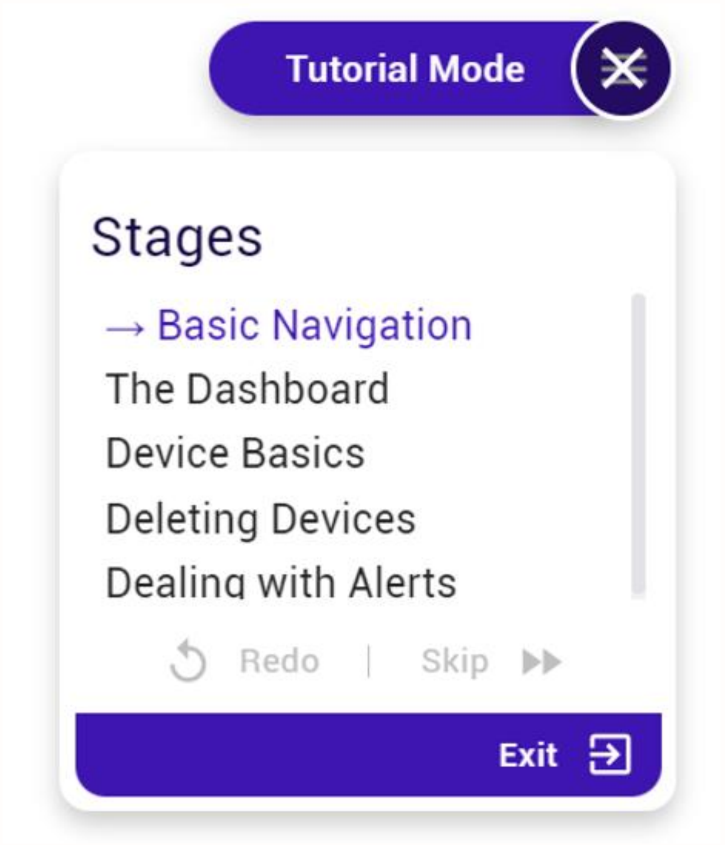
3. content: Descriptive information about the highlighted feature(s). This prop is not used by action-type tooltips.

4. targetElement: The ID of the DOM element that the tooltip references. For action-type tooltips, an event listener is added to detect when the element is clicked, signaling that the action is completed.

5. targetAreaElement: The ID of the DOM element that surrounds the target element, determining the highlighted area. While often the same as the target element, separating them allows greater flexibility. The target area's position dictates the tooltip's location and arrow direction.

Tutorial

Tutorial Menu

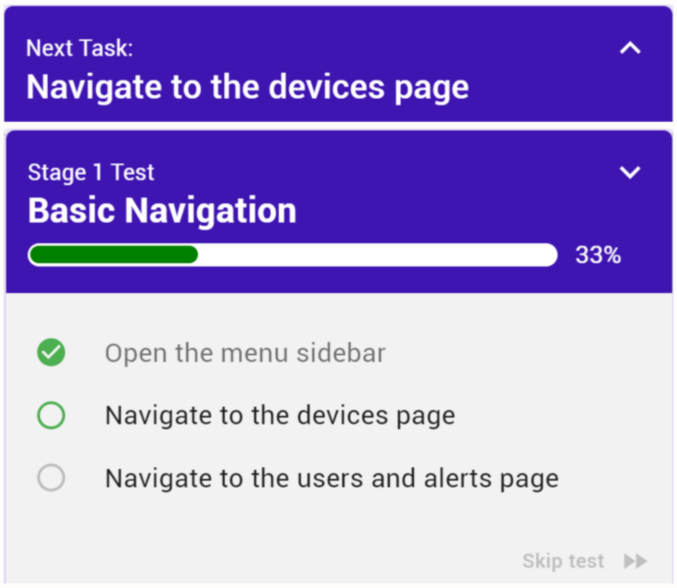


The Tutorial Menu component remains visible throughout the tutorial. Clicking the menu icon opens a panel displaying the tutorial stages and highlighting the current one. It also provides options to restart the current stage, skip it, or exit the tutorial. The collapsed menu is compact enough to avoid distraction but constantly reminds the user they are in tutorial mode. Nine props define the Tutorial Menu's position, appearance, and functionality:

1. positionX: The X position of the menu - "left" or "right".
2. positionY: The Y position of the menu - "top" or "bottom".
3. circleDistFromOuter: The distance of the menu from the edge of the window.
4. tutorialName: The name of the tutorial.
5. stages: A list of all stages in the tutorial.
6. stageNo: The number of the current stage.

Tutorial

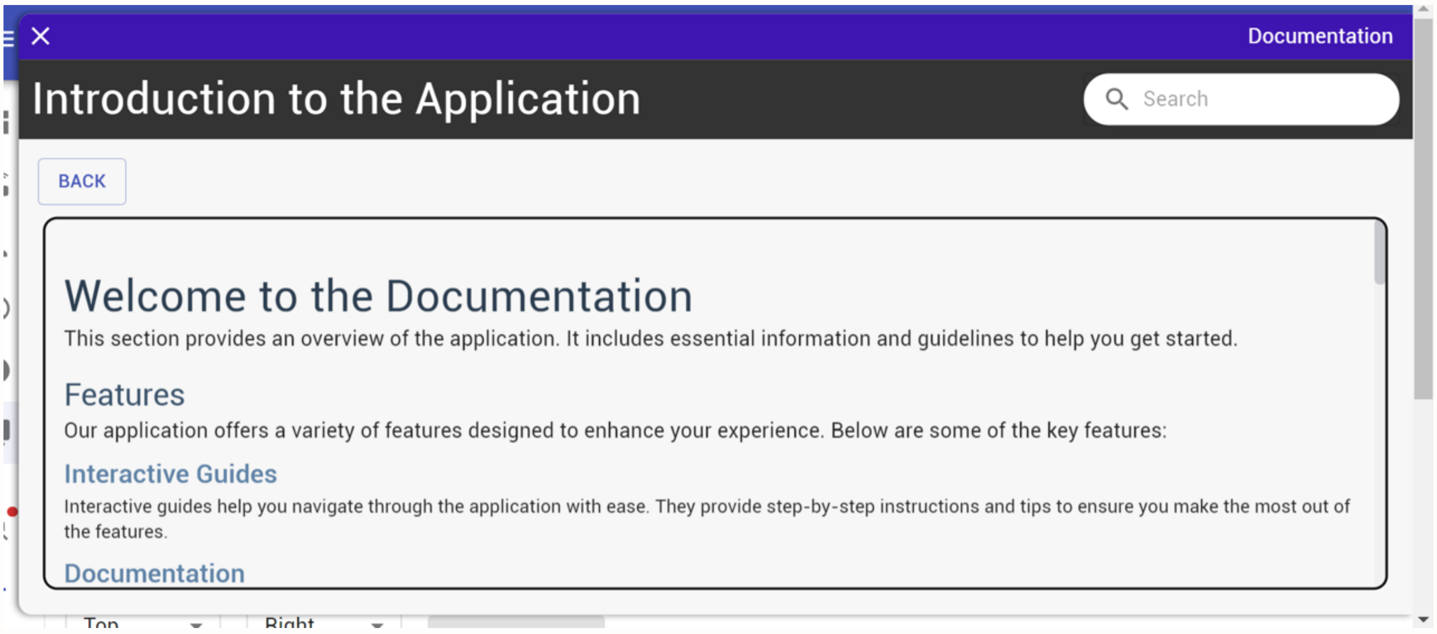
Test Progress Tile


Next Task:
Navigate to the devices page
Stage 1 Test
Basic Navigation
33%
Open the menu sidebar
Navigate to the devices page
Navigate to the users and alerts page
Skip test

The Test Progress Tile component visualizes a user's progress in the tutorial's tests, which consist of a series of tasks. This component acts as a task checklist, helping users track their progress.
The tile can be expanded or collapsed. In its collapsed state, it displays only the current task. When expanded, it shows the entire list of tasks: completed tasks are ticked and greyed out, the current task is highlighted, and upcoming tasks are listed. Additionally, it displays the current stage name and a progress bar indicating overall completion.
On completion of a task, a series of animations play: the completed task is ticked and greyed out, the progress bar grows, and the next task flashes into focus.
The Test Progress Tile's appearance and functionality are defined by seven props:

1. **stageName**: The name of the current stage.
2. **stageNo**: The number of the current stage.
3. **taskNames**: A full list of the tasks in the test.
4. **currentTaskNo**: The number of the current task.
5. **currentTask**: The name of the current task.
6. **onNext**: The function triggered when a task is completed.
7. **onSkipTest**: The function triggered when the user skips the test.

Documentation



Introduction to the Application

Welcome to the Documentation

This section provides an overview of the application. It includes essential information and guidelines to help you get started.

Features

Our application offers a variety of features designed to enhance your experience. Below are some of the key features:

Interactive Guides

Interactive guides help you navigate through the application with ease. They provide step-by-step instructions and tips to ensure you make the most out of the features.

Documentation

The Documentation component provides a flexible and user-friendly way to integrate documentation into applications by utilizing markdown-like syntax rendered by the **DocumentationMarkdownRenderer** component. It offers a slide-out panel accessible via an icon fixed to the right side of the screen. This allows users to quickly access and search documentation without leaving their current context.

Key Features and Functionalities:

•**Slide-Out Panel**: The panel slides out from the right side of the screen when the icon is clicked to display the list of documentation pages.

•**Icon Button**: An optional icon button which toggles the slide-out panel and is fixed to the right side of the screen.

•**Search Functionality**: A search bar within the panel allows users to search through documentation pages and headings. Search results are displayed in a dropdown, where users can click on a result to navigate directly to the relevant section.

Documentation

The DocumentationMarkdownRenderer component works as follows:

• **Text Parsing**: The function uses a regular expression to identify different symbolic notations in the text, such as $heading1{id}{content}, $link{id}{content}, and $video{id}{content} {videoUrl}.

• **UI Element Rendering**: Based on the type of notation, the parser replaces the notation with corresponding Material-UI components. For example, $heading1 {id}{content} is replaced with a Typography component styled as a heading.

• **Rendering Process**: The parsed content is returned as an array of React elements, which are then rendered inside a Box component.

All supported UI elements you can use

$heading1{id}{content}

$body1{id}{content}

$heading2{id}{content}

$heading3{id}{content}

$body2{id}{content}

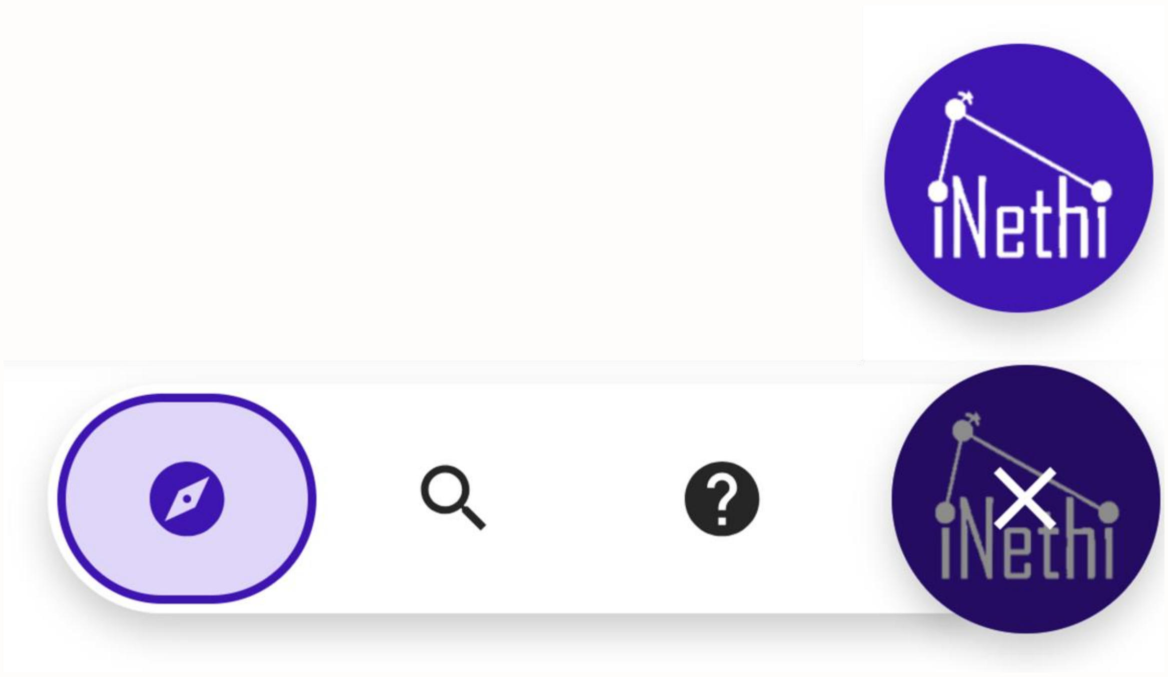
$link{url}{text}

$image{id}{altText}{imageUrl}

$video{id}{altText}{videoUrl}

Resource Circle

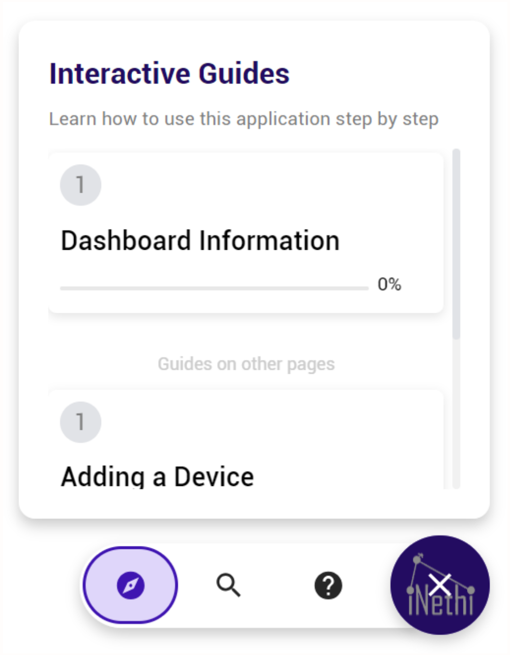
The ResourceCircle component is designed as a lightweight, unobtrusive feature that provides targeted assistance to network managers. Unlike the tutorial, which is more comprehensive and ideal for onboarding completely new users, the Resource Circle is always visible and available without disrupting the underlying application. It offers concise help for specific challenges through three main features: interactive guides, a search tool, and info icons. The Resource Circle and open menu is shown in the figure below.



The core functionality of the Resource Circle centers around seamlessly integrating its sub-components — Interactive Guides, the Search Tool, and Info Icons — managing their visibility and interaction. The circle controls the display of the menu that houses the Interactive Guides, Search Tool, and Info Icons.

Resource Circle

Interactive Guides


Interactive Guides

Learn how to use this application step by step

1

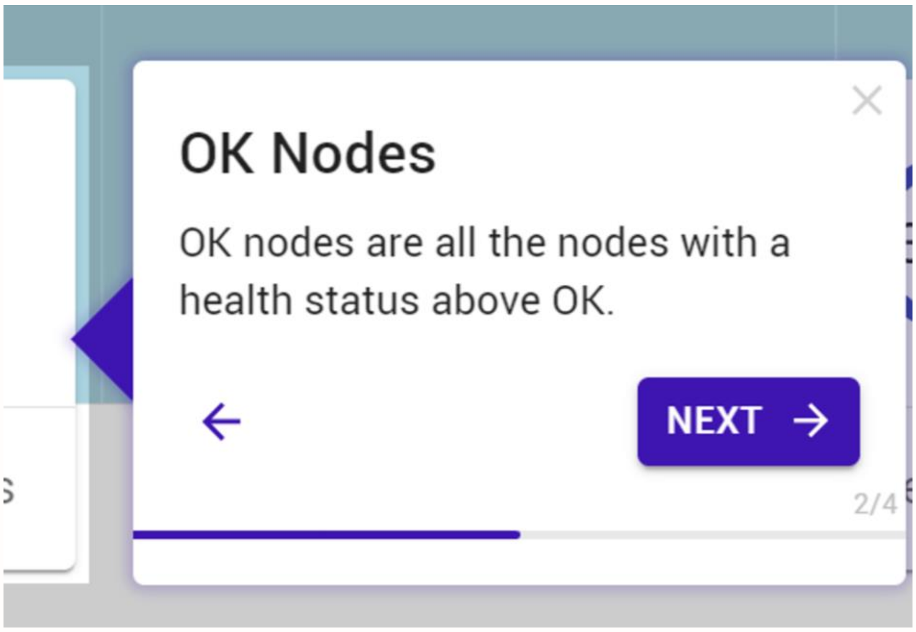
Dashboard Information

0%

Guides on other pages

1

Adding a Device


OK Nodes

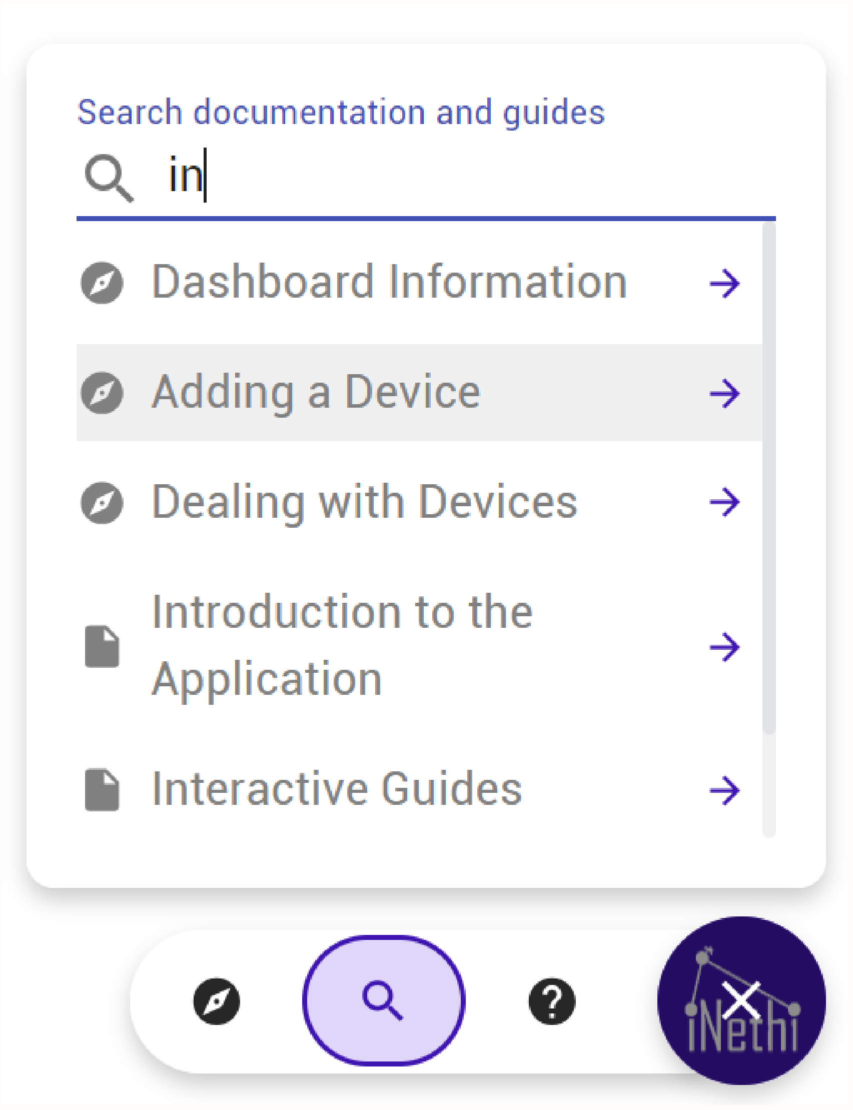
OK nodes are all the nodes with a health status above OK.

Interactive guides consist of sequential tooltips and are therefore essentially tutorial walkthroughs, but are focused and individual. They align with the Resource Circle's purpose by being shorter and more targeted than the tutorial walkthroughs. These guides inform users about specific activities, allowing them to start and exit quickly as needed. Unlike the tutorial walkthroughs, interactive guides can be used on demand and in any order, offering help exactly when it's needed. The logic of the interactive guides mirrors that of the tutorial's walkthroughs, using the same Tooltip component. However, in the interactive guides, the canExit boolean prop is set to true for every tooltip, allowing users to exit whenever they choose.

Interactive guides are accessed through the Resource Circle's menu, where all available guides, whether on the current page or another, are displayed as cards. Clicking on any of these cards starts the corresponding interactive guide.

Resource Circle

Search Tool


Search documentation and guides

in

Dashboard Information →

Adding a Device →

Dealing with Devices →

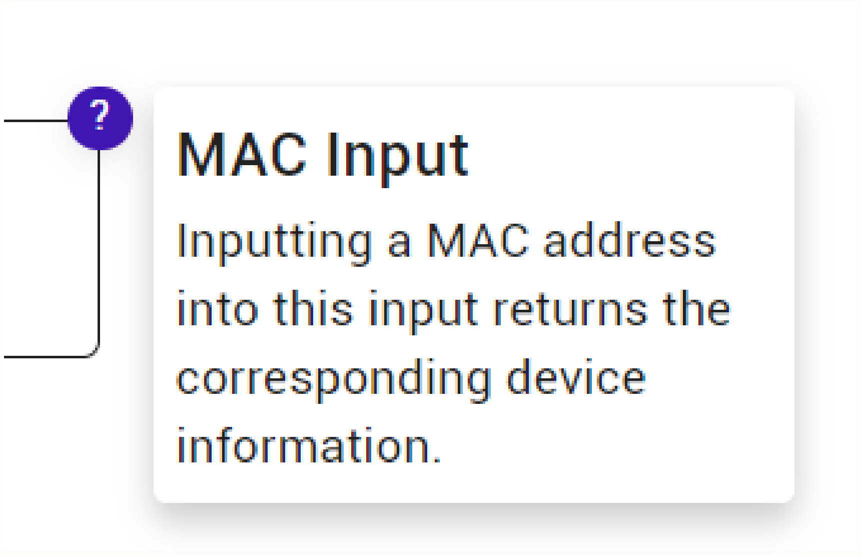
Introduction to the Application →

Interactive Guides →

The search tool helps users quickly find relevant guides or documentation using keywords. It searches through both the interactive guides and documentation content, focusing on headings. The results show guide names and headings with preview content, allowing users to start the guide or access relevant documentation easily. The above figure shows a snapshot of the search tool in use.

Resource Circle

Info Icons



MAC Input

Inputting a MAC address into this input returns the corresponding device information.

Shown in the figure above, info icons are small icons attached to the top-right corner of UI elements. When clicked, they display concise information about the element. These icons follow the positioning and visibility of their target element and can adjust dynamically when the page scrolls, ensuring the information is always accessible. This is done by searching through every ancestor element of the info icon's target element, determining if any are fixed positioned or scrollable, and using this information to add scroll event listeners and dynamically adjust icon positioning and visibility. The three functions that define this logic are *findScrollableParents*, *hiddenByScrollable Container*, and *hasFixedPosition*.

04

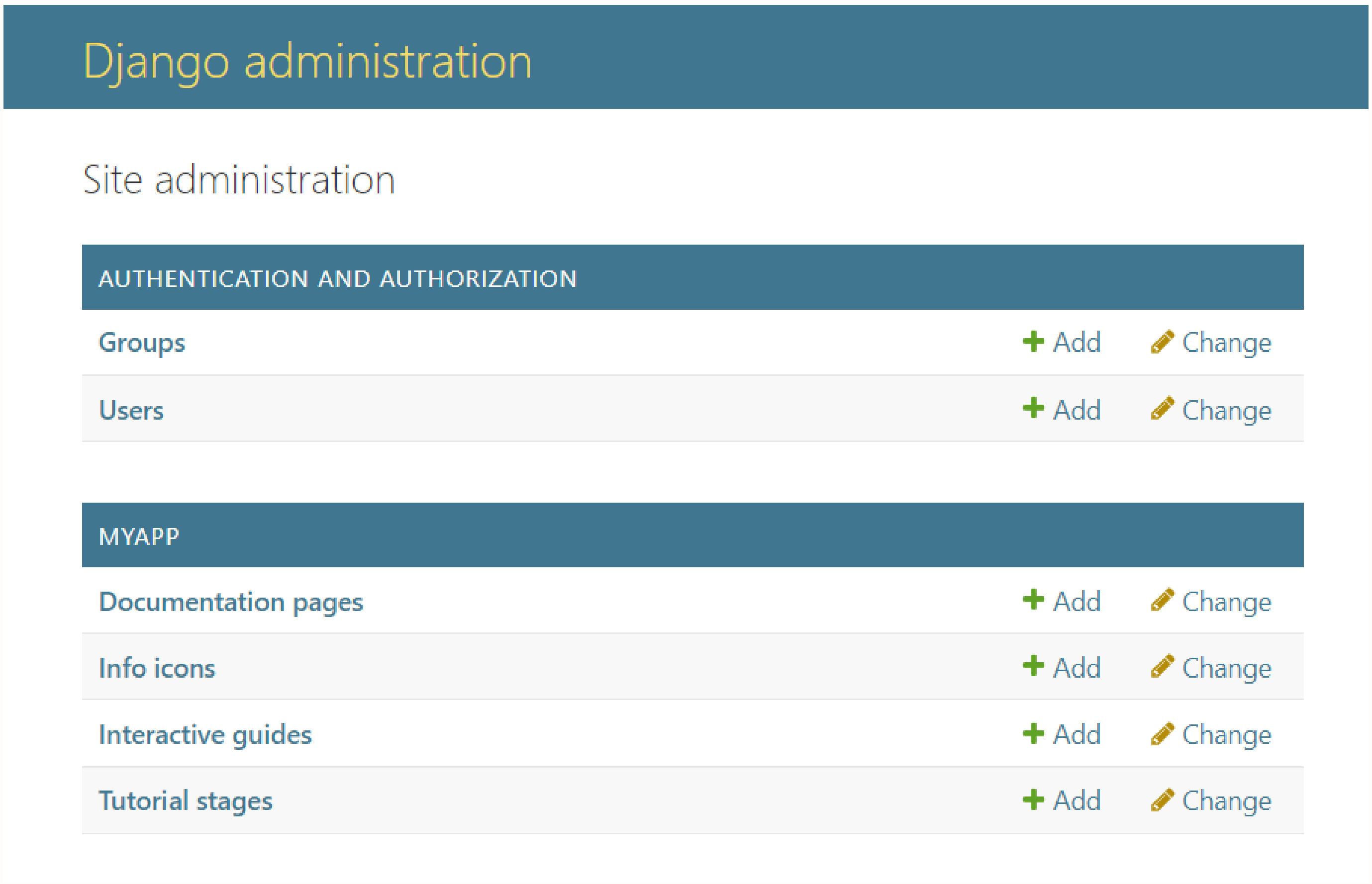
Customizing Onboarding Content

Customizing Onboarding Content

Admin Page

You can add, edit, or delete your own custom onboarding content for your application through the admin page at http://localhost:8085/admin. If you have not yet created an admin user, view this step on the Installation and Setup Backend page of this user manual.

The admin page extends the Django default admin page to access and manipulate your tutorial and resource data stored in an SQL database. It should look something like the below figure.


Django administration

Site administration

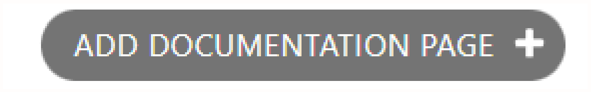
AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

MYAPP		
Documentation pages	+ Add	Change
Info icons	+ Add	Change
Interactive guides	+ Add	Change
Tutorial stages	+ Add	Change

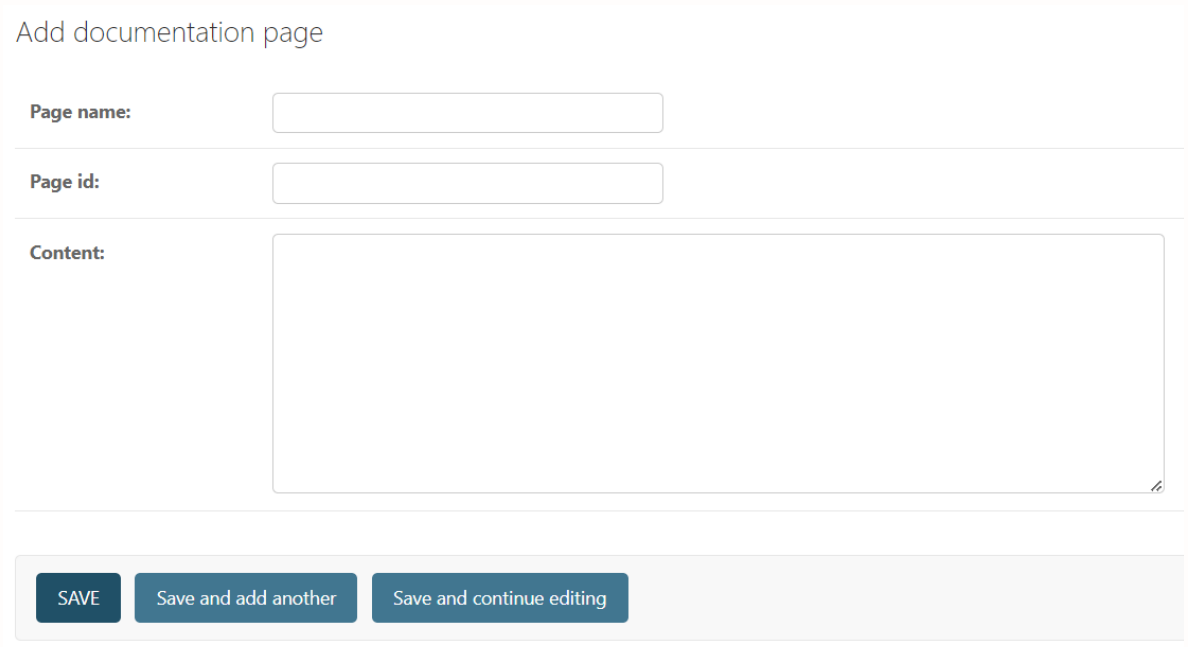
Customizing Onboarding Content

Documentation

Click on Documentation pages on the admin page menu, and then click



You will then be taken to the following page:


Add documentation page

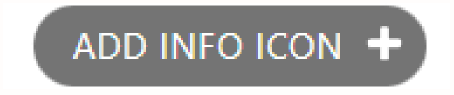
Page name:	
Page id:	
Content:	

Enter a page name, a page id, and then the content of the page. Remember to stick to the correct format when entering content – outlined on the Documentation page of this manual. Press save to add the documentation page

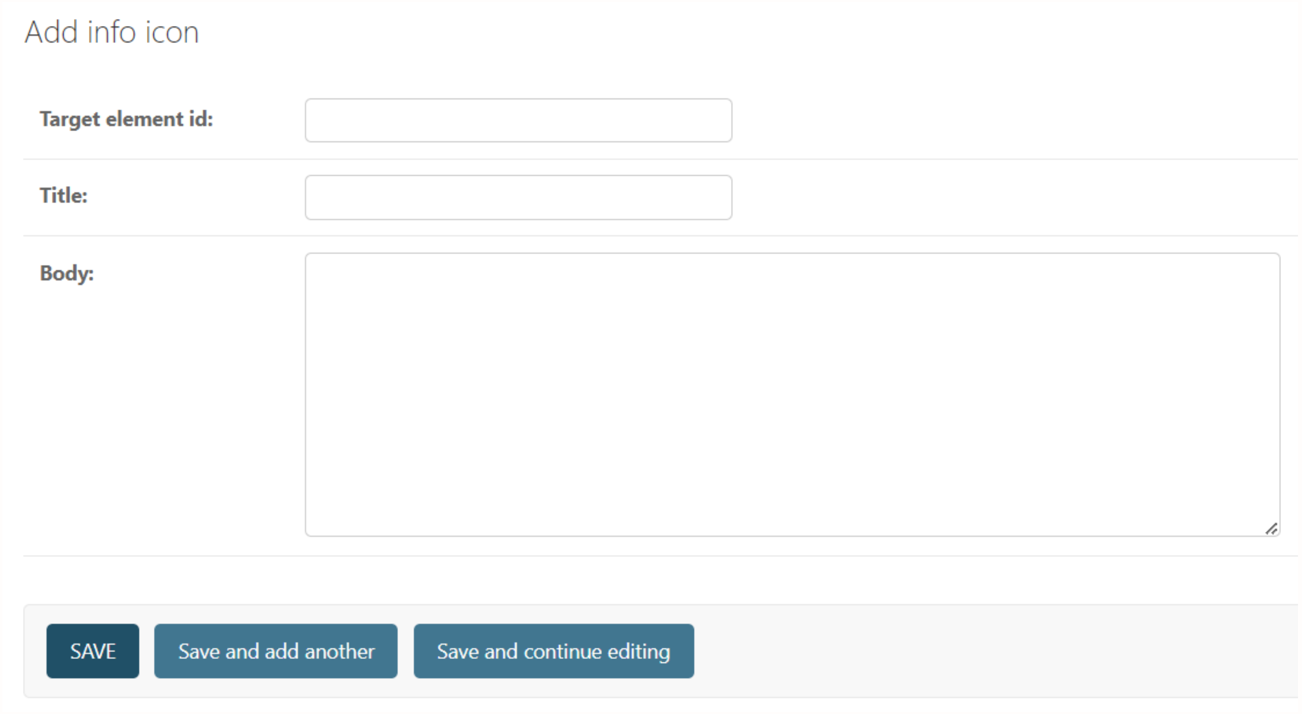
Customizing Onboarding Content

Info Icons

Click on the *Info icons* on the admin page menu, and then click



You will then be taken to the following page:



Add info icon

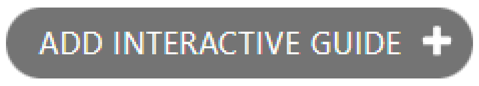
Target element id:	
Title:	
Body:	

Enter the target element id, and the title and body of the info icon. Press save to add the info icon.

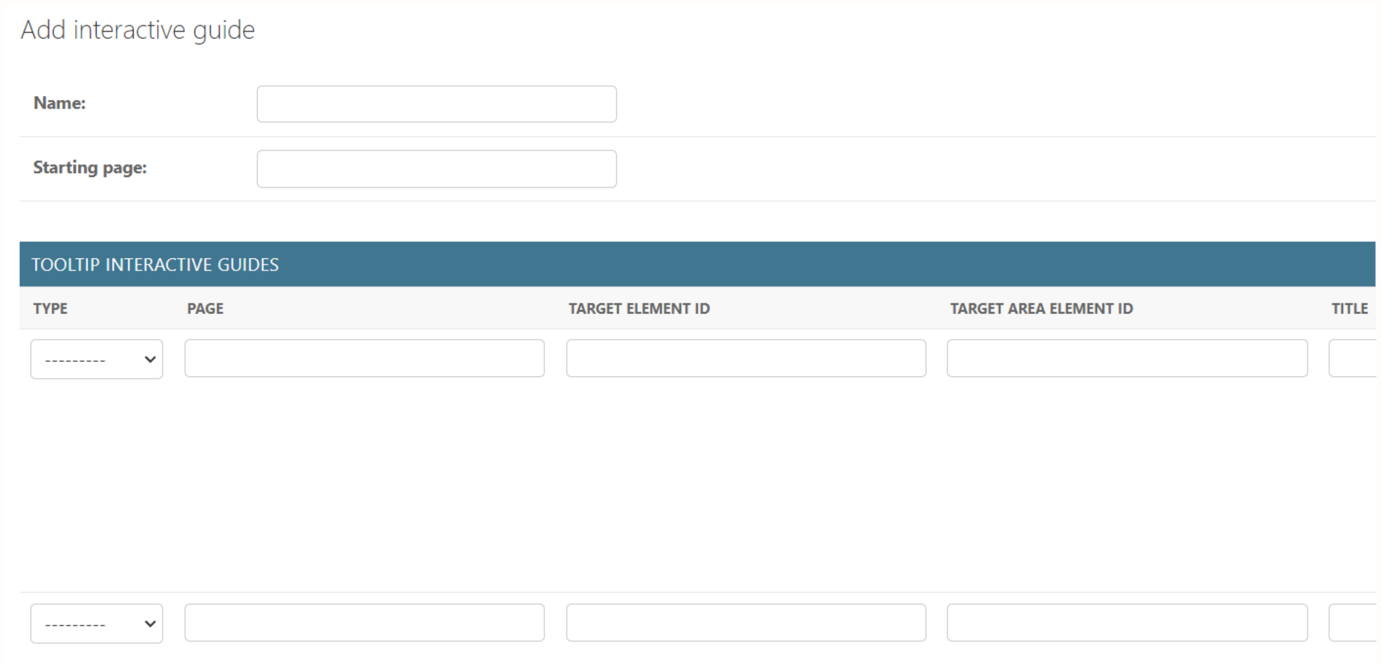
Customizing Onboarding Content

Interactive Guides

Click on the *Interactive Guides* on the admin page menu, and then click



You will then be taken to the following page:



Add interactive guide

Name:	
Starting page:	

TOOLTIP INTERACTIVE GUIDES

TYPE	PAGE	TARGET ELEMENT ID	TARGET AREA ELEMENT ID	TITLE

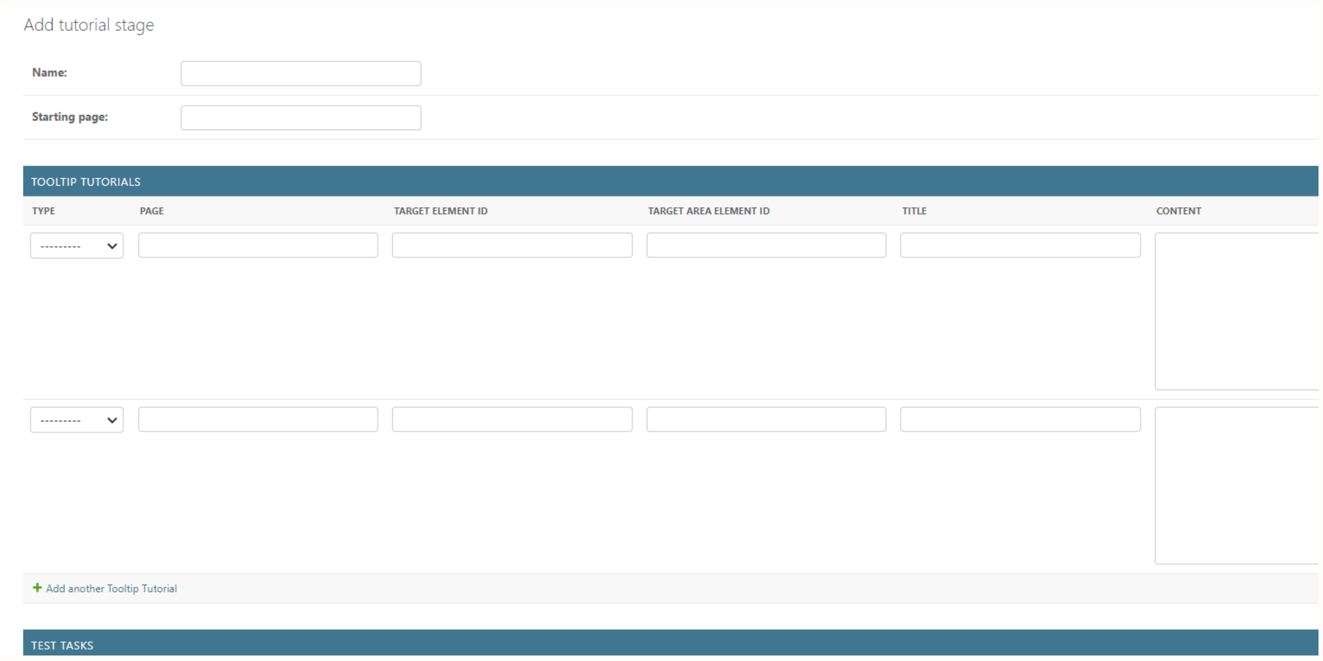
Enter the name of the interactive guide, and the page that the interactive guide starts on. If, for example, you have a page called settings, input **/settings** into the *starting page* field. Then add as many tooltips to the interactive guide as you desire (must have at least one).

Customizing Onboarding Content

Tutorial

Click on the *Tutorial stages* on the admin page menu, and then click ADD TUTORIAL STAGE +

You will then be taken to the following page:



Add tutorial stage

Name:	
Starting page:	

TOOLTIP TUTORIALS

TYPE	PAGE	TARGET ELEMENT ID	TARGET AREA ELEMENT ID	TITLE	CONTENT

+ Add another Tooltip Tutorial

TEST TASKS

A Tutorial is equivalent to an interactive guide. It differs only in the fact that it has a test. So, after filling in to tooltips for the walkthrough part of the tutorial, move on to the test tasks.

The *Test Tasks* are all the tasks that a user needs to complete, in the correct order, to complete the tutorial stage test. A click element is the element that needs to be clicked to complete the task.

For example, the test task description may be "**Navigate to the home and about page**". The click element ids could possibly be "navbar-home-button" and "navbar-about-button".

The text-input-elements (which are optional), are text fields that must be filled in when the click element is clicked to complete that task. For example, the test task description may be "**Fill in "1.16.0.5" in the MAC address field and click submit.**". The click element ***id*** would possibly be "submit-password-button", and the text input element ***id*** could be "mac-address-input", and the ***required text*** would be "1.16.0.5".

Thanks!