ISSN: 2093-5374 / E-ISSN: 2093-5382

WoTs-TDGV: Thing Description Generator and Validator

Dr. Samah Rahamneh[1*], Ala' Hanoun[2], Dr. Fahed Jubair[3], and Khalid A. Darabkh[4]

[1*]Department of Computer Engineering, The University of Jordan, Amman, Jordan.
s.rahamneh@bayt.net, https://orcid.org/0000-0002-4777-9694

[2]Bayt Company, Amman, Jordan. a.hanoun@bayt.net, https://orcid.org/0009-0008-7894-7460

[3]Department of Computer Engineering, The University of Jordan, Amman, Jordan.
f.jubair@ju.edu.jo, https://orcid.org/0000-0002-9311-807X

[4]Department of Computer Engineering, The University of Jordan, Amman, Jordan.
k.darabkeh@ju.edu.jo, https://orcid.org/0000-0002-0362-1975

Received: May 17, 2024; Revised: July 16, 2024; Accepted: August 16, 2024; Published: September 30, 2024

Abstract

The breakneck evolution of the Web of Things (WoT) has led to a burgeoning demand for reliable tools that facilitate the automatic generation and validation of Thing Descriptions (TD). However, emerging tools that generate or validate TD automatically have suffered from abounding limitations and constraints. In this paper, we propose and implement an automatic web-based Thing Description Generator and Validator (TDGV) application. TDGV automates and simplifies TDs generation while ensuring the compliance with the WoT standards. The application generates more accurate thing description results. In order to evaluate the usability of the TDGV, a Concurrent Think-aloud usability study was conducted with a sample of 150 developers. The study revealed positive feedback regarding the user interface and the TDGV usability. The study results highlight areas for enhancement, such as TDGV navigation. TDGV is a convenient and robust Thing Description Generator and Validator web application. It provides a valuable tool for users in the WoT domain, enabling them to automatically and easily generate and validate TDs.

Keywords: Thing Description, Automatic Thing Generation, and Automatic validation, Web of Things, Internet of Things.

1 Introduction

The Internet of Things (IoT) is a network of devices/things that spans over a wide range from hand-held devices to complicated servers connected through the internet (Darabkh & Al-Akhras, 2021; Wang et al., 2021). These Things are controlled by humans or other devices using technologies compatible with conventional internet technologies (Laghari et al., 2021; Eiriemiokhale & Olutola, 2023; Kassab & Darabkh, 2020). When IoT devices were first introduced, there was no widely accepted standard or protocol for connecting and communicating between them. Moreover, many manufacturers from different business areas have been competing for dominance in the market. As a result, a diverse choice of devices has been introduced (Korkan et al., 2020; Darabkh et al., 2023; Majjaru & Senthilkumar, 2023). Figure 1 illustrates the WoT ecosystem comprising a vast range of Things/devices,

Corresponding author: [1]Department of Computer Engineering, The University of Jordan, Amman, Jordan.

access network/gateways, edge services, cloud services, web services, and applications (Brindha Merin et al., 2023).

The diversity resulted in what is referred to as the IoT interoperability problem (Stoyanova et al., 2020; Tzavaras et al., 2023). It made the development of IoT systems and the integration of multiple devices difficult for developers and integrators (Nandini, 2024). It requires them to understand different Application Programming Interfaces (APIs) and communication protocols. As a result, IoT was held back from reaching its full potential (Korkan et al., 2020; Zhang et al., 2022; Lee et al., 2021).

To address this problem the World Wide Web Consortium (W3C) introduced a set of standards that enables interoperability across IoT platforms and application domains, forming the Web of Things (WoT) architecture (Tran et al., 2017; W3C, 2021; Rhayem et al., 2020). While WoT has been gaining momentum in recent years due to the release of various standards by W3C, there is still a perceived lack of resources in certain areas, particularly in terms of training, education, and tools for developers and researchers (Halim et al., 2024; Khalil et al., 2021; Cimmino et al., 2020).

The key component of WoT is the Thing Description (TD), a standardized format for describing the physical device’s capabilities, properties, and interactions with other devices in a way that humans and machines under- stand (Badii et al., 2013). The W3C TD standard defines a common set of concepts and properties that can be used to describe the functionality and behavior of any IoT device, regardless of the underlying communication protocol or application domain (Khoeurt et al., 2023; Faheem et al., 2019; Negash et al., 2019; Kumar et al., 2022). The following list illustrates a naive Thing Description of an electrical door.

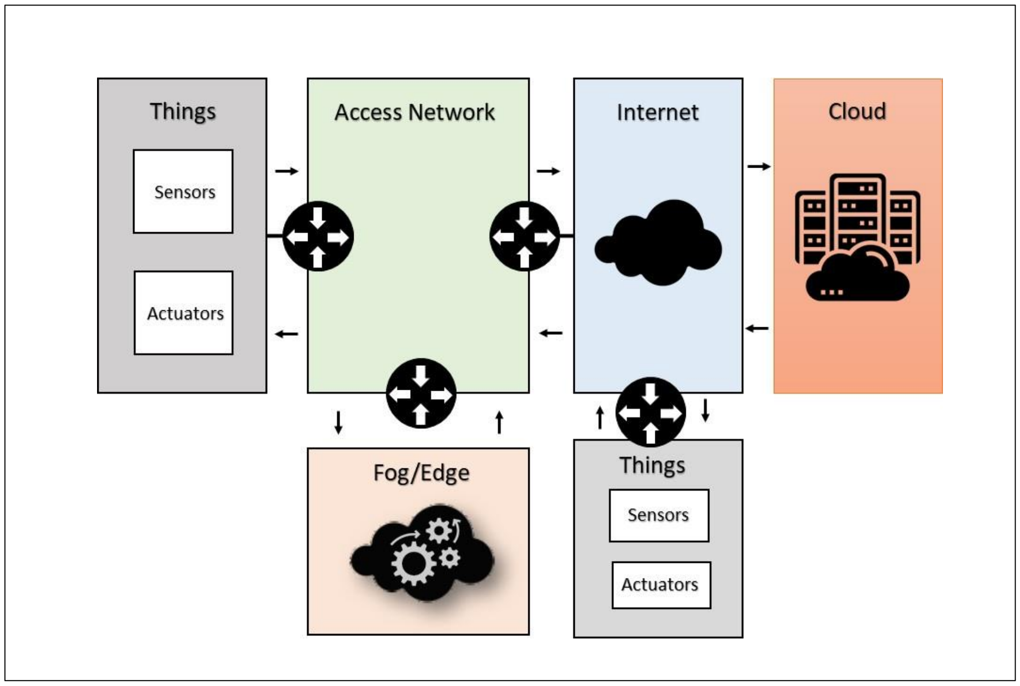


Figure 1: WoT Ecosystem

```

{
    ”@context ”: ” h t t p s : / /www. w3. org /2019
    / wot / t d / v1 ”,
    "id": "urn:dev:wot:electrical-door",
    "title": "Electrical Door",
    "description": "A smart electrical
    door that can be controlled and monitored
    through the Web of Things.",
    "securityDefinitions": {
      "nosec_sc": {
        "scheme": "nosec"
      }
    },
    "security": ["nosec_sc"],
    "properties": {
      "doorState": {
        "type": "string",
        "description": "The current state
        of the door (open/closed).",
        "readOnly": true
      }
    },
    "actions": {
      "open": {
        "description": "Open the door.",
        "output": {
          "type": "string",
          "description": "Status message:
          'Door opened successfully.'"
        }
      },
      "close": {
        "description": "Close the door.",
        "output": {
          "type": "string",
          "description": "Status message:
          'Door closed successfully.'"
        }
```

```
        }
    },
    "events": {
        "stateChanged": {
            "description": "Event fired
            when the door state    changes.",
            "data": {
                "type": "string",
                "description": "The new
                state of the door (open/closed)."
            }
        }
    }
}
```

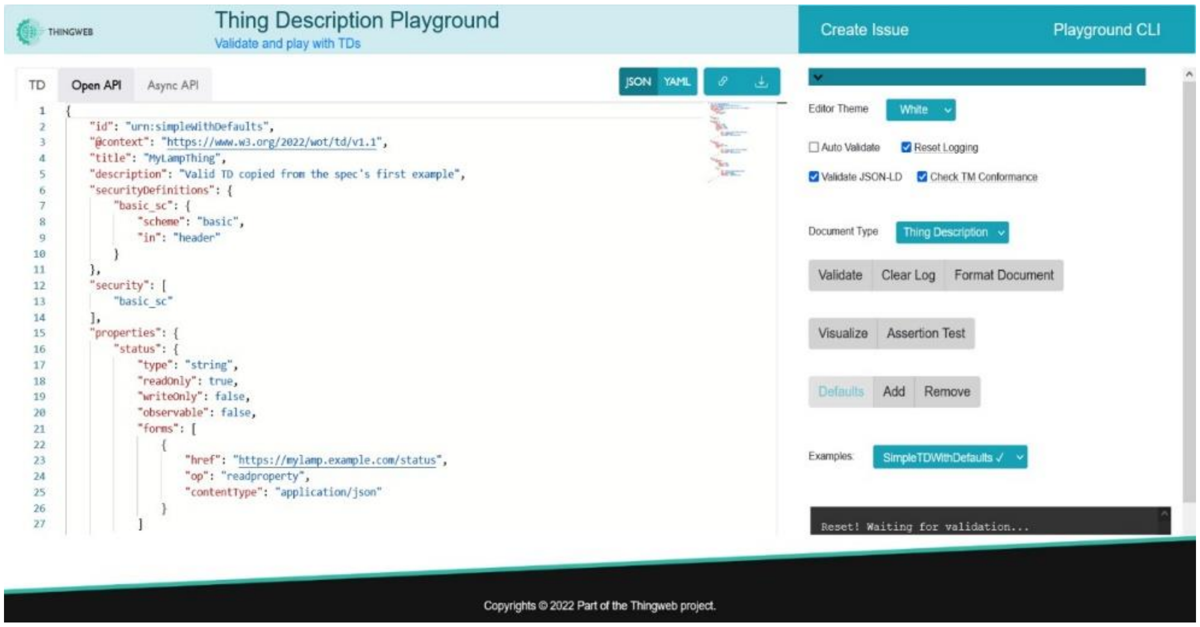
Writing the Thing Description has to follow the set of rules provided by the W3C standard to ensure inter- operability in the WoT systems. That makes writing it manually time-consuming and error-prone. Creating TDs also requires a good understanding of the W3C standard, and the underlying WoT technologies, which may be challenging for non-experts. As a solution, we introduced the WoT Thing Description Generator and Validator, a beginner-friendly and web-based tool for generating and validating TDs.

2 Related Work

The Thing Description Playground (A web of things, 2022) is a part of the Thing web project which is an open-source initiative led by the Distributed Systems Group at the University of Stuttgart in Germany. The initiative aims to promote the development and adoption of the Web of Things through the creation of a set of tools, frameworks and standards.

The Thing Description Playground, whose graphical overview is seen in Figure 2 allows the user to edit a TD in the left-hand side of the screen and automatically see errors and warnings during the TD validation process. The website verifies that the TD is a valid JSON-LD instance besides validating it against the TD JSON schema. The application also provides a visualization feature that allows users to view their TDs in graphical formats such as graphs and trees. However, emerging tools that generate or validate TDs automatically have suffered from abounding limitations and constraints (Iglesias-Urkia et al., 2020; Darabkh et al., 2024). Although the Thing Description Playground is a useful tool for editing and validating TDs, there are some limitations and drawbacks to consider. One such limitation is that the application is primarily intended for expert users, who are expected to write the TD from scratch without any assistance. As a result, users who are new to WoT or the JSON-LD format may find the application challenging to use.

In comparison, our tool is designed to be beginner-friendly and is targeted toward newcomers as well as experts. It simplifies the process of writing a TD as the users can fill out a template instead of having to write it from scratch and the application will ensure that the submitted data is a valid Thing Description that adheres to the W3C standard.



```
{
    "id": "urn:simpleWithDefaults",
    "@context": "https://www.w3.org/2022/wot/td/v1.1",
    "title": "MyLampThing",
    "description": "Valid TD copied from the spec's first example",
    "securityDefinitions": {
        "basic_sc": {
            "scheme": "basic",
            "in": "header"
        }
    },
    "security": [
        "basic_sc"
    ],
    "properties": {
        "status": {
            "type": "string",
            "readOnly": true,
            "writeOnly": false,
            "observable": false,
            "forms": [
                {
                    "href": "https://mylamp.example.com/status",
                    "op": "readproperty",
                    "contentType": "application/json"
                }
            ]
```

Editor Theme: White

☐ Auto Validate ☑ Reset Logging

☑ Validate JSON-LD ☑ Check TM Conformance

Document Type: Thing Description

Examples: SimpleTDWithDefaults ✓

```
Reset! Waiting for validation...
```

Figure 2: Screenshot of the Thing Description Playground Website

Table 1: Main Differences between TDGV and TD Playground

Features	**TDVG**	**TD playground**
Supported TD Versions	1.0, 1.1	1.1
TD Generation	✓ Generates TDs from a template	× Generates TDs from scratch
TD Validation	✓ Validates against chosen version	× Basic validation against 1.1
Ease of Use	✓ Intuitive UI for quick TD generation	✓ UI can be overwhelming for new users
Backward Compatibility	✓ Supports backward compatibility with 1.0 and 1.1 TDs	× No backward compatibility
Live Validation	✓ Real-time validation during TD editing	× No live validation
Open Source	× Closed source	× Closed source

The application also offers helpful tips and hints to guide users through the process of filling out the form, eliminating the need to constantly return to the W3C TD standard for information. This can make the process smoother and easier and improve the overall user experience. In addition to that, the application will be compatible with all previous versions of the TD standard. This guarantees a wider user base for the application. Moreover, TDGV enables developers to produce backward-compatible applications. Table 1 epitomizes the main differences between TDGV and the TD playground.

3 Methodology

1) System Analysis

A "Thing" in TDs refers to a device or service that is realized physically or virtually. The WoT TD describes the metadata and interfaces of the Thing are described by the WoT TD standard by a set of attributes that can describe the metadata, characteristics, and capabilities of things.

The W3C TD provides a description of each attribute and its data type, and declares if each attribute is mandatory or optional in the assignment column. For instance, the “@context” attribute is mandatory and should be included in every Thing Description document. Its data type is a Uniform Resource Identifier (URI) or an array of URIs.

The TD standard also describes the interfaces of a Thing by defining three interaction affordances: Properties, Actions, and Events. Properties are the characteristics or attributes of a Thing that can be read or monitored to expose its current state. For example, a TD of a temperature sensor may have a “temperature” property, allowing users to retrieve the current temperature reading.

Actions are operations or functions that can be performed on a Thing. They allow users to interact with the Thing and perform specific operations. For example, turning a device on or off. Meanwhile, events are notifications or alerts that a Thing can emit to indicate a change or significant occurrence. For instance, a device may emit an event to alert consumers when it detects overheating, enabling timely intervention.

2) Development Environment

TDGV is realized as a web-based application. Web-based applications have cross-platform compatibility. In addition, TDGV can be accessed from anywhere through a web browser. This web-based realization eliminates the need to install specific software, providing convenience and flexibility.

For the implementation, we used the popular web technologies: Hypertext Markup Language (HTML) which is the core language used for structuring and presenting content on the web. We also used Cascading Style Sheets (CSS) and its popular library Bootstrap for the visual presentation and styling of the application. Using these technologies offered us several advantages. For example, wide compatibility as HTML and CSS are universally supported by web browsers, making websites developed with these technologies compatible with a vast range of devices and platforms. Also, Ease of Learning and the available resources are big reasons why we choose these technologies to implement our project. Many tutorials, forums, and libraries exist to assist developers in using these technologies.

After analyzing and designing the system it appeared that there was no need in the system for server-side infrastructure, database connections, or backend logic. So, we decided to make the application client-side only. This enables faster response times and reduces server load, making it suitable for lightweight operations like generating and validating JSON files.

JavaScript was our choice for a client-side programming language. Using JavaScript was very suitable for the nature of our project. As in our project, we will be generating and validating thing descriptions which are JSON files. JavaScript has built-in support for JSON. It provides methods like JSON.parse() and JSON.stringify() that make it easy to convert JavaScript objects to JSON strings and vice versa. This native JSON support simplifies the process of generating and working with JSON data within the JavaScript code.

Finally, we used Visual Studio Code as the integrated development environment (IDE) for the development of our application. The rich set of extensions and plugins available for VS Code provided us with a wide range of tools for code editing, debugging, and version control integration which significantly contributed to the efficiency and effectiveness of our development workflow.

3) System Implementation

Our objective for the system is to create an HTML form that enables users to define their WoT Thing, including its metadata, properties, actions and events. We also wanted to incorporate a TD validation feature for users that have already written their TD and wish to verify if it aligns with the WoT TD standard. Finally, we included an editor section within the application to allow the users to conveniently fix any errors present in their TD.

The first part of the project is the TD Generator, an HTML form that the user will fill to create a TD, instead of having to write the TD manually and from scratch. The form should be self-validating and should not allow the user to submit unless the input data is valid for a TD. When the data is valid the user will be shown a review of the generated TD. If the user is satisfied with it, they can download it as a JSON file. Otherwise, they can go back to the form and modify it.

To implement this functionality we used the JSON Form library. Which is a JavaScript client-side library that takes a structured data model defined using JSON Schema as input and returns an HTML form that matches the schema (Darabkh et al., 2023). We decided to use this library because our project requires a complicated HTML form to be able to implement most attributes of a TD and its properties, actions and events. This library significantly simplified the development process. In addition, the library returns a Bootstrap-friendly HTML form. This means we can simply include Bootstrap CSS files to style and format the form to create a well-formatted form, enhancing the visual appeal and user experience. To visually represent the system, we have included a flow chart in Figure 3, illustrating the steps involved in the TD generation process.

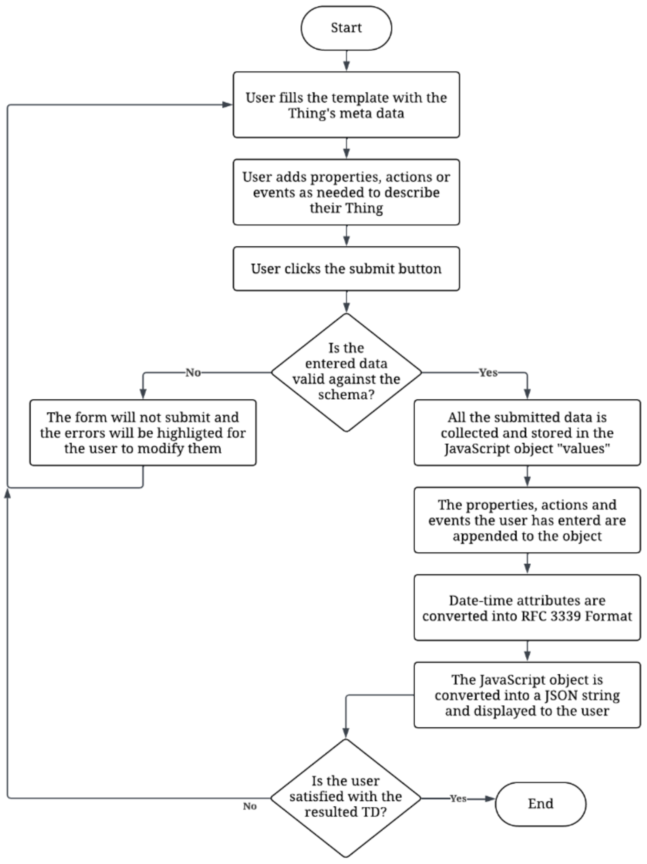

Start
User fills the template with the Thing's meta data
User adds properties, actions or events as needed to describe their Thing
User clicks the submit button
Is the entered data valid against the schema?
No
Yes
The form will not submit and the errors will be highligted for the user to modify them
All the submitted data is collected and stored in the JavaScript object "values"
The properties, actions and events the user has enterd are appended to the object
Date-time attributes are converted into RFC 3339 Format
The JavaScript object is converted into a JSON string and displayed to the user
Is the user satisfied with the resulted TD?
No
Yes
End

Figure 3: Flow Chart of the TD Generator

The TD Validator is the second part of the project. It provides the ability to validate an existing Thing Description. The validator supports different versions of the TD standard the user could choose from. It also allows the user to edit the TD.

Figure 4 presents a flow chart of the TD Validator. It illustrates the validation process which starts with the user selecting the TD version they wish to validate against. It was important for our project to support all versions of the TD schema allowing developers to create backward-compatible applications that can work with older devices or systems.

Currently, there are two versions of the TD standard available: TD version 1.0 and TD version 1.1. TD version 1.0 is the initial version. TD version 1.1 introduces additional metadata properties for describing the context, purpose, and intended usage of a TD. This helps provide more comprehensive information about TD and its associated Thing.

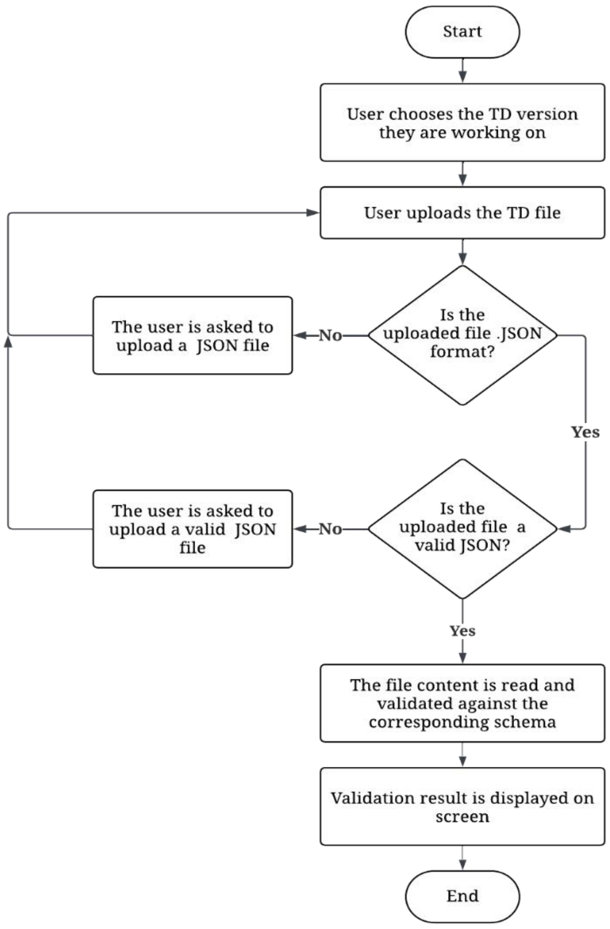

Start
User chooses the TD version they are working on
User uploads the TD file
Is the uploaded file .JSON format?
No
The user is asked to upload a JSON file
Yes
Is the uploaded file a valid JSON?
No
The user is asked to upload a valid JSON file
Yes
The file content is read and validated against the corresponding schema
Validation result is displayed on screen
End

Figure 4: Flow Chart of the TD Validator

The validation process consists of three distinct steps. The first step occurs when the user uploads the TD file. We restrict the uploaded file to only allow JSON format, which aligns with the format of thing descriptions. If the user uploads any other format, they will be alerted to upload a JSON file.

The second step of the validation happens when the user's file is uploaded and read. Our TD Validator requires the TD file to be a valid JSON format. If the uploaded file does not meet this requirement, attempting to read it into a JSON object will result in an error. In such cases, the user will be prompted to upload a valid JSON file to proceed with the validation process.

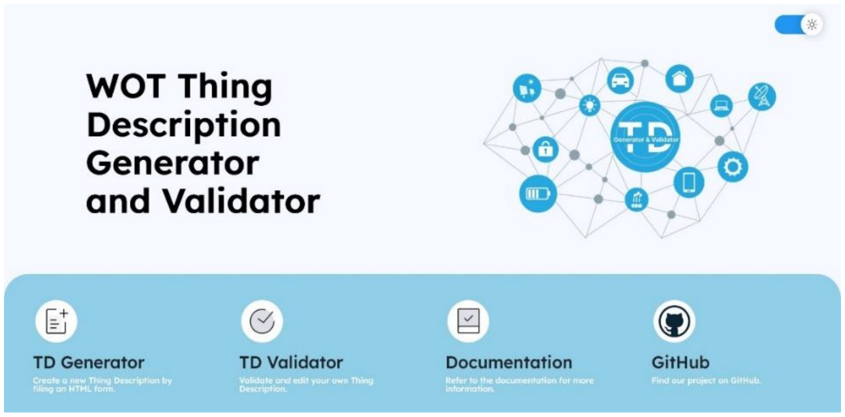
If the uploaded file is confirmed as a valid JSON we proceed to the third step of the validation process. Which is validating the TD against the W3C WoT schema. For this step, we used the library Ajv [26]. Which is a popular and widely used JavaScript library for validating JSON data against JSON Schema definitions which in this case will be the TD schema.

4 Results

In order to comprehensively simulate and document the various usage scenarios of the application, we will present the following run scenarios as practical examples:

1) Scenario 1: Thing Description Generation

In this scenario, we will demonstrate the generation process of a TD for a smart lamp device. The user starts by selecting the TD Generator option from the application's main menu depicted in Figure 5.



Create a new Thing Description by filling an HTML form.

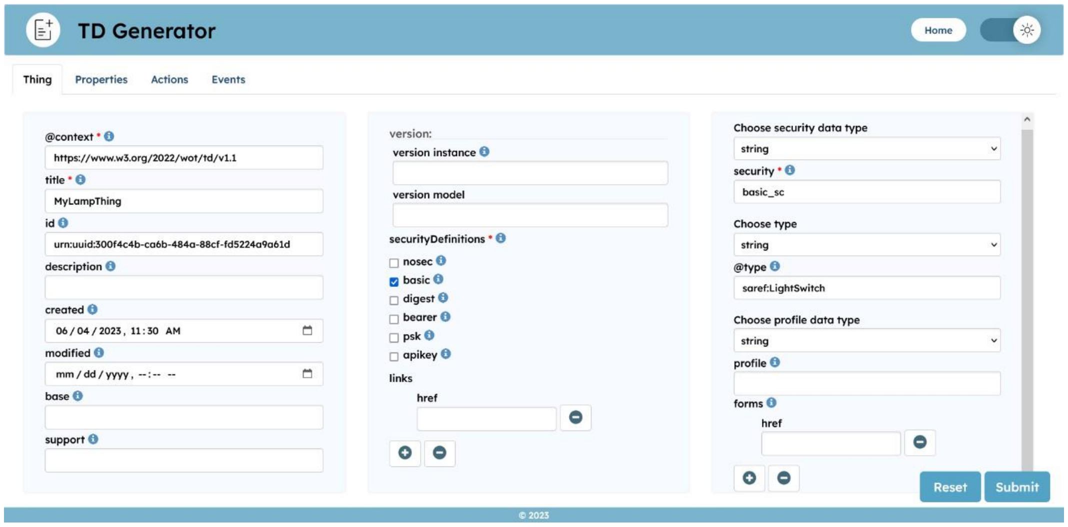
Validate and edit your own Thing Description.

Refer to the documentation for more information.

Find our project on GitHub.

Figure 5: Screenshot of the Home Page for the WoT Thing Description Generator and Validator

After that the user will be taken to the TD Generator interface where they enter the relevant details and characteristics of the device (title, id, security definitions, security and created) as it appears in Figure 6 and 7. An info icon is available for each attribute for enhanced usability. When the user hovers the mouse over the icon, it displays helpful information about the corresponding property.



@context *: https://www.w3.org/2022/wot/td/v1.1

title *: MyLampThing

id: urn:uuid:300f4c4b-ca6b-484a-88cf-fd5224a9a61d

description:

created: 06 / 04 / 2023 , 11:30 AM

modified: mm / dd / yyyy , --:-- --

base:

support:

version:

version instance:

version model:

securityDefinitions *: nosec [✓ basic] digest bearer psk apikey

links

href:

Choose security data type: string

security *: basic_sc

Choose type: string

@type: saref:LightSwitch

Choose profile data type: string

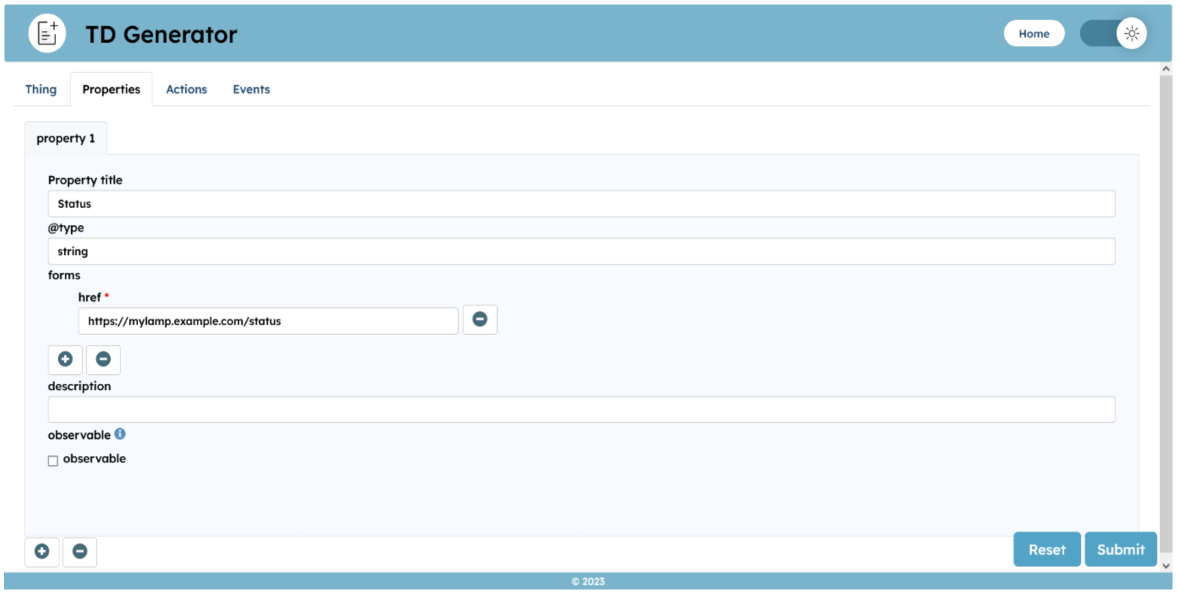
profile:

forms

href:

Figure 6: Screenshot of the TD Generator Interface

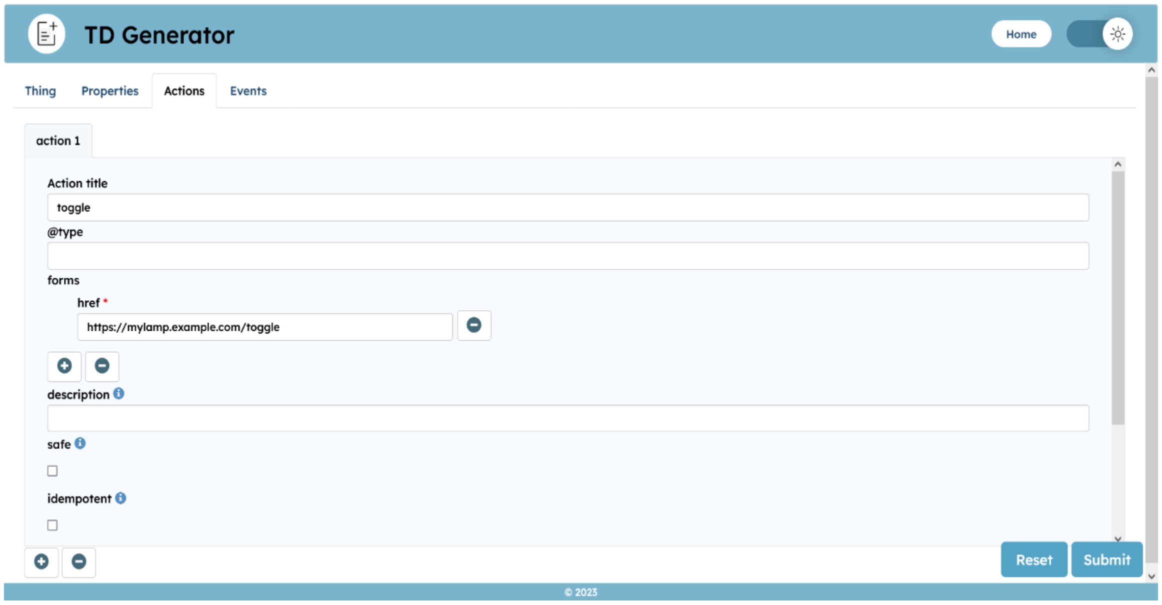
Then the user navigates to the properties tab of the form and adds a “status” property that describes the status of the lamp.



Property title: Status
@type: string
forms
href *: https://mylamp.example.com/status
description:
observable
☐ observable

Figure 7: Screenshot of the Properties Tab of the TD Generator

In the next step the user navigates to the Actions tab showcased in Figure 8 and adds a “toggle” action to the TD. This action is specified to toggle the switch status of the lamp using the POST method on the https://mylamp.example.com/toggle resource specified by the user. The user does not have to specify the method type since POST is the default assumption in TDs for invoking Actions.

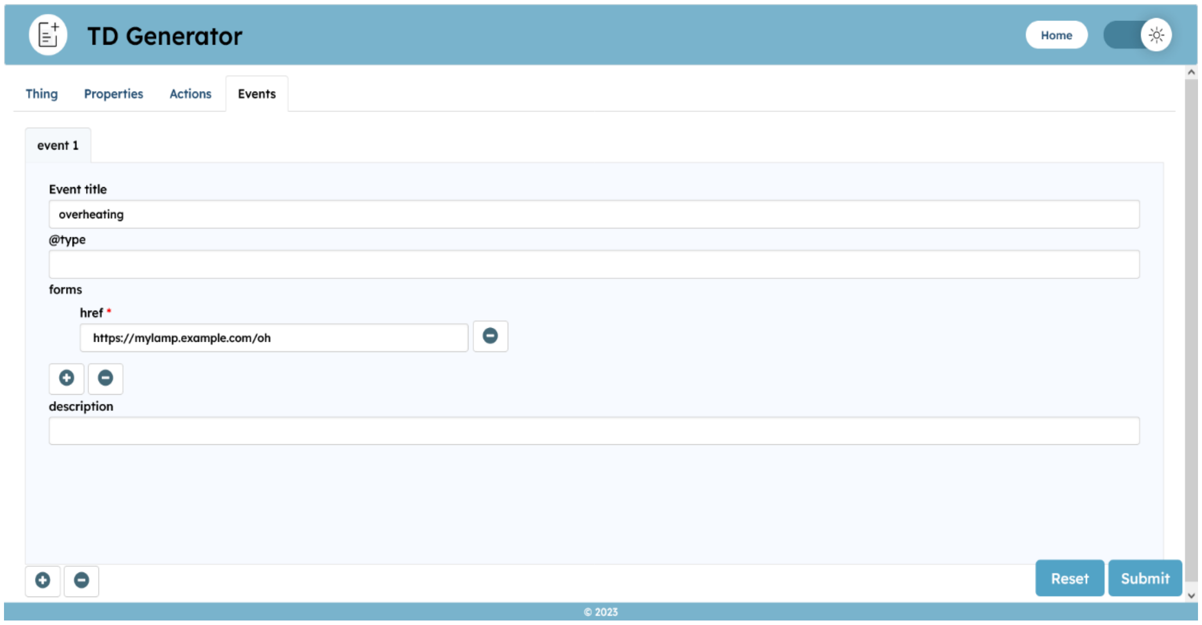


Action title: toggle
@type:
forms
href *: https://mylamp.example.com/toggle
description:
safe ☐
idempotent ☐

Figure 8: Screenshot of the Actions Tab of the TD Generator

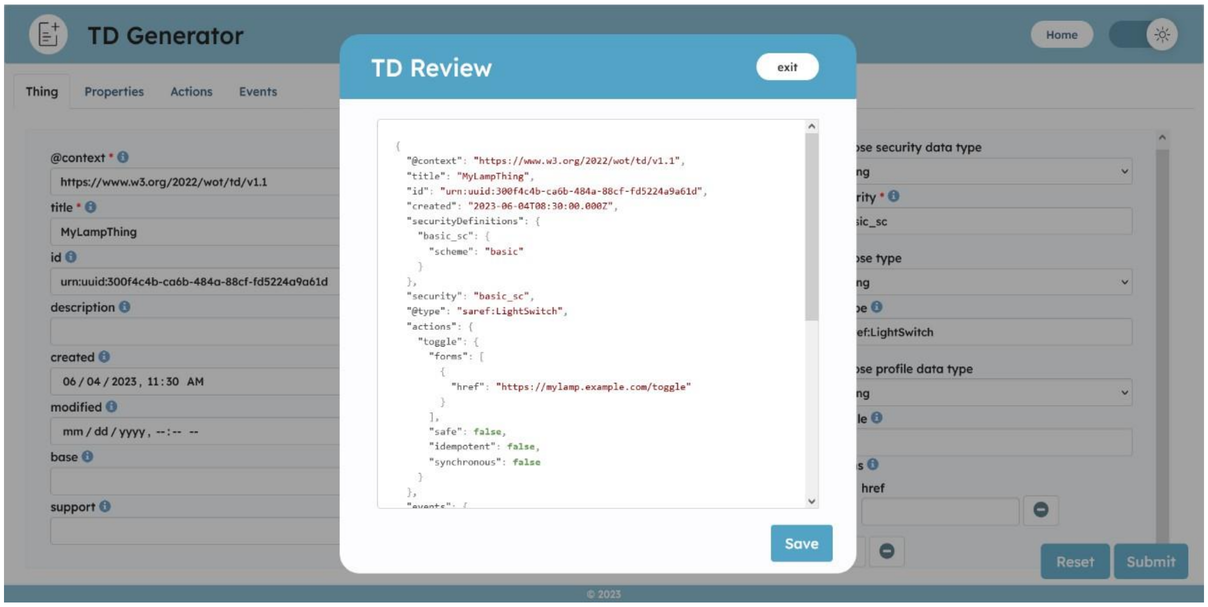
The user then navigates to the Events tab showcased in Figure 9 and adds an “overheating” event. The aim of this event is to be notified of a possible overheating event of the lamp. The notification can be obtained by using HTTP with its long polling subprotocol at the URI https://mylamp.example.com/oh.

Finally, once the user has provided all the necessary information for their Thing, they simply click on the submit button to generate their TD. Because the entered values are accurate and valid, a review of the TD will be displayed as Figure 10 illustrates.



Event title: overheating
@type:
forms
href *: https://mylamp.example.com/oh
description:

Figure 9: Screenshot of the Events Tab of the TD Generator



@context *: https://www.w3.org/2022/wot/td/v1.1
title *: MyLampThing
id: urn:uuid:300f4c4b-ca6b-484a-88cf-fd5224a9a61d
description:
created: 06 / 04 / 2023 , 11:30 AM
modified: mm / dd / yyyy , --:-- --
base:
support:

```
{
  "@context": "https://www.w3.org/2022/wot/td/v1.1",
  "title": "MyLampThing",
  "id": "urn:uuid:300f4c4b-ca6b-484a-88cf-fd5224a9a61d",
  "created": "2023-06-04T08:30:00.000Z",
  "securityDefinitions": {
    "basic_sc": {
      "scheme": "basic"
    }
  },
  "security": "basic_sc",
  "@type": "saref:LightSwitch",
  "actions": {
    "toggle": {
      "forms": [
        {
          "href": "https://mylamp.example.com/toggle"
        }
      ],
      "safe": false,
      "idempotent": false,
      "synchronous": false
    }
  },
  "events": {
```

Figure 10: Screenshot of the TD Review Interface

2) Scenario 2: Thing Description Validation

In the second scenario, we will showcase a validation process of two TD test files.

The user starts by selecting the TD Validator option which redirects them to the interface displayed in Figure 11, then the user selects the version of their TD document which is V1.1. The user clicks the browse button to navigate to the first TD file and upload it.

The uploaded file will be displayed to the user on the left-hand side of the screen as displayed in Figure 12. The system validates the TD against the specified schema and displays the validation result on the right-hand side which showcases that the user-uploaded TD is valid.

The user tests another TD file, this time the validation result in Figure 13 specifies that the TD is invalid because it is messing two required properties "title" and "@context".

The user edits the TD by adding the missing properties. The live validation feature of the application will revalidate the edited TD. The new validation results will be displayed as in Figure 14 to showcase that the TD is now valid against the schema after the user modified it.

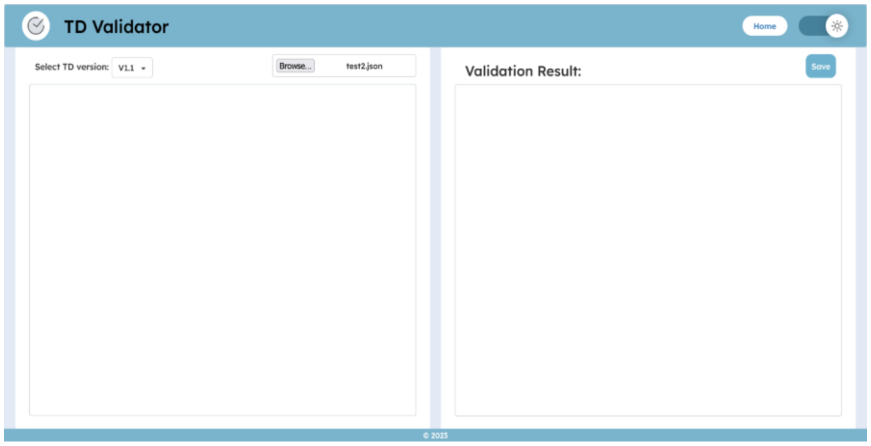
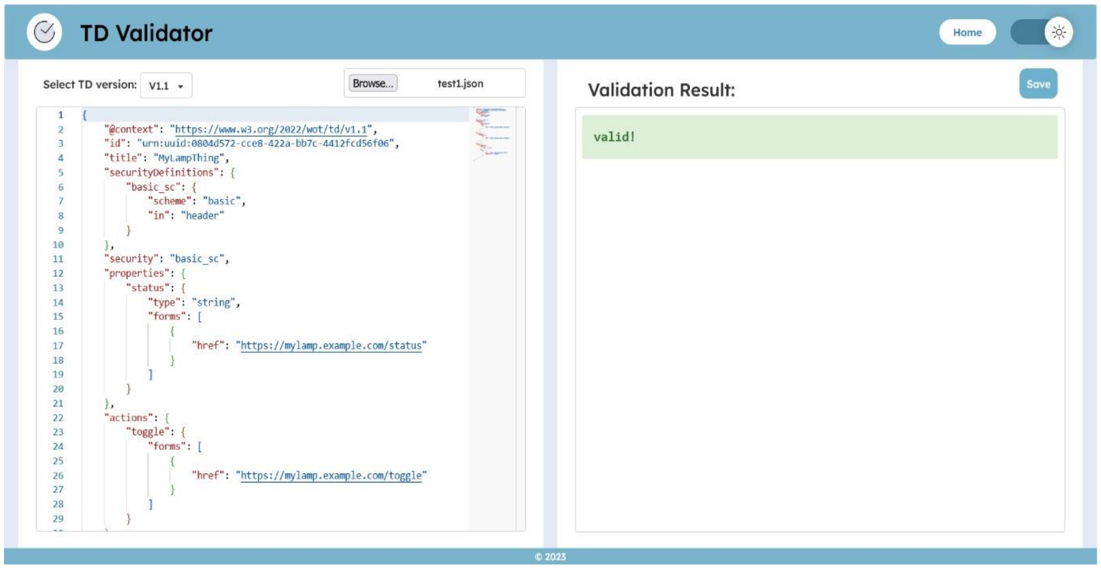


Figure 11: Screenshot of the TD Validator Interface



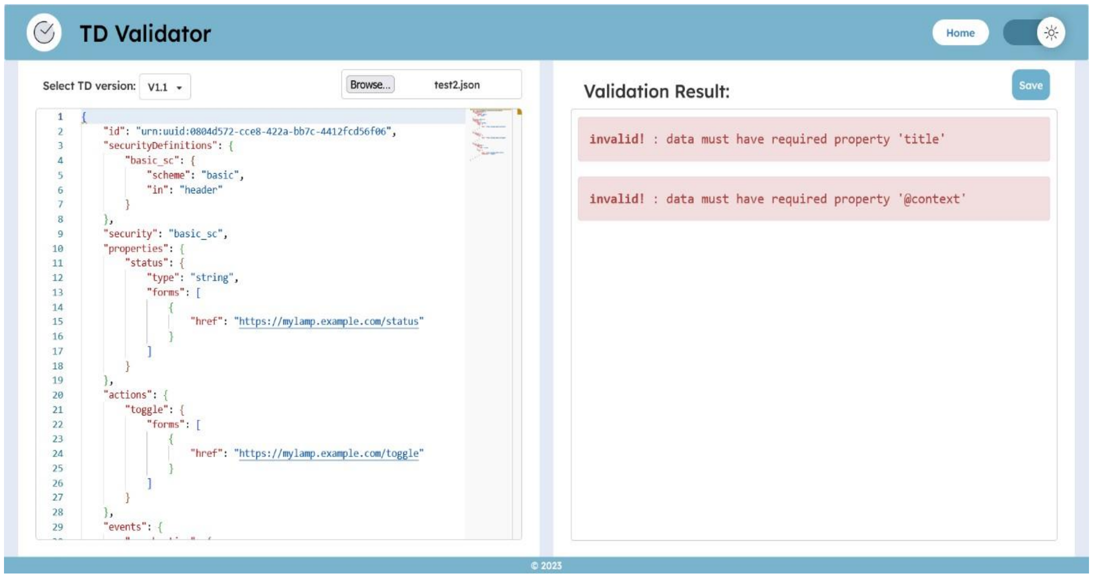
```
{
    "@context": "https://www.w3.org/2022/wot/td/v1.1",
    "id": "urn:uuid:0804d572-cce8-422a-bb7c-4412fcd56f06",
    "title": "MyLampThing",
    "securityDefinitions": {
        "basic_sc": {
            "scheme": "basic",
            "in": "header"
        }
    },
    "security": "basic_sc",
    "properties": {
        "status": {
            "type": "string",
            "forms": [
                {
                    "href": "https://mylamp.example.com/status"
                }
            ]
        }
    },
    "actions": {
        "toggle": {
            "forms": [
                {
                    "href": "https://mylamp.example.com/toggle"
                }
            ]
        }
```

```
valid!
```

Figure 12: An Example of a Valid TD File

5 Evaluation

To evaluate the usability of our application, we employed the Concurrent Think-aloud Method, a well-established research technique utilized to gain valuable insights into an individual's cognitive processes and decision-making during task performance. This method is widely recognized and utilized as an effective means of evaluating the usability and user experience of an application, providing valuable feedback and enhancing the overall quality of our design. We conducted the study with a carefully selected group of 150 participants, considering the scope and complexity of our application. This sample size aligns with the recommended range for this type of study, ensuring a sufficient number of insights and perspectives. Our chosen participants consisted of computer engineering students, representing our target user group and reflecting the intended audience of our application, which caters to individuals interested in WoT. Importantly, all participants were new to the concepts of WoT and TDs, ensuring a fresh perspective and unbiased feedback during the evaluation process. The participants were provided with a comprehensive description of a smart lamp. Additionally, they were given a set of tasks to complete, as outlined in Table 2. These tasks were carefully designed to assess various functionalities and features of our application, ensuring a detailed evaluation of its capabilities and user-friendliness.



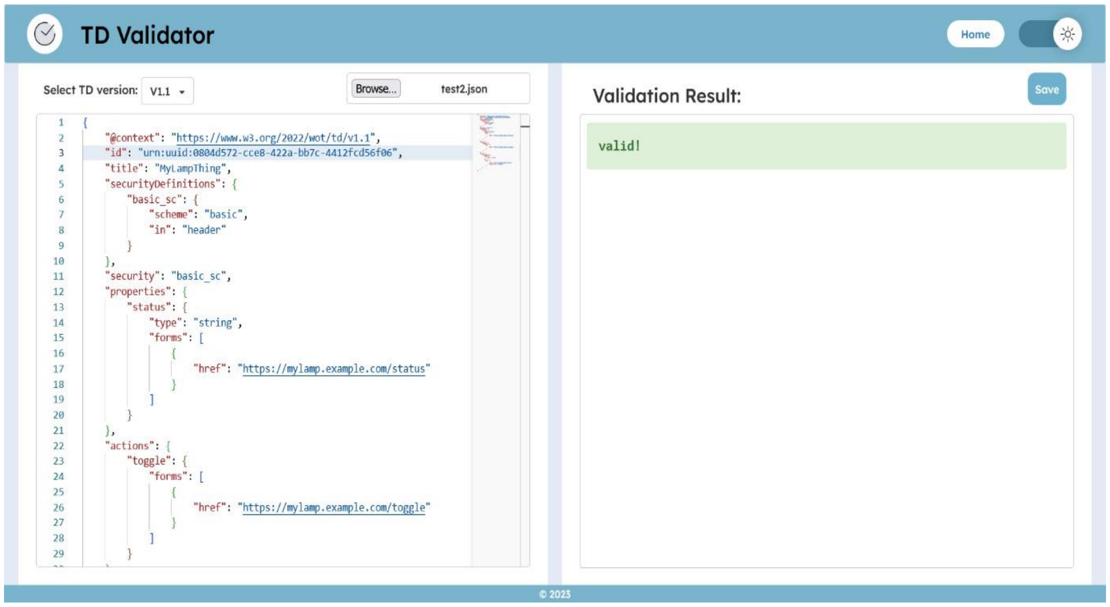
Validation Result:

```
{
    "id": "urn:uuid:0804d572-cce8-422a-bb7c-4412fcd56f06",
    "securityDefinitions": {
        "basic_sc": {
            "scheme": "basic",
            "in": "header"
        }
    },
    "security": "basic_sc",
    "properties": {
        "status": {
            "type": "string",
            "forms": [
                {
                    "href": "https://mylamp.example.com/status"
                }
            ]
        }
    },
    "actions": {
        "toggle": {
            "forms": [
                {
                    "href": "https://mylamp.example.com/toggle"
                }
            ]
        }
    },
    "events": {
```

```
invalid! : data must have required property 'title'

invalid! : data must have required property '@context'
```

Figure 13: An Example of an Invalid TD File



Validation Result:

```
{
    "@context": "https://www.w3.org/2022/wot/td/v1.1",
    "id": "urn:uuid:0804d572-cce8-422a-bb7c-4412fcd56f06",
    "title": "MyLampThing",
    "securityDefinitions": {
        "basic_sc": {
            "scheme": "basic",
            "in": "header"
        }
    },
    "security": "basic_sc",
    "properties": {
        "status": {
            "type": "string",
            "forms": [
                {
                    "href": "https://mylamp.example.com/status"
                }
            ]
        }
    },
    "actions": {
        "toggle": {
            "forms": [
                {
                    "href": "https://mylamp.example.com/toggle"
                }
            ]
        }
```

```
valid!
```

Figure 14: TD Validation Result after Modifications

During the concurrent think-aloud usability study, several noteworthy findings were observed. Participants' consistently provided positive feedback regarding the user interface, expressing appreciation for its well-organized layout and intuitive design. Moreover, all participants successfully completed all four assigned tasks, indicating that our application is user-friendly, particularly for individuals with limited experience in the field. These findings validate the effectiveness of our design approach and highlight the accessibility and ease of use of our application. However, few participants encountered challenges while navigating specific sections of the application.

Table 2: Major Functionalities in TDGV

Task 1: Create a New Thing Description
Imagine you want to create a Thing Description for the smart lamp shown. Use the website's functionality to generate a new Thing Description.
Task 2: Export a Thing Description
Export a Thing Description in JSON-LD format to share it with a development team. Use the website' to generate a JSON-LD file.
Task 3: Validate an Existing Thing Description
Use the website's validator to check if the Thing Description is valid and conforms to the appropriate TD specification. Identify any validation errors or warnings provided by the website.
Task 4: Editing an Existing Thing Description
Fix all validation errors that appeared in the previous step.

6 Conclusions and Future Directions

The rapid growth of the WoT has led to a crucial demand for reliable and automatic TD generation and validation. An automatic and robust TD generator and validator was designed and developed in this paper, namely, TDGV. This web-based application addresses the need for an efficient tool, in the WoT domain, that allows users to create comprehensive Thing Descriptions by determining applicable attributes and generating corresponding JSON files.

The TD Validator ensures the compliance with the W3C TDs standard. The application has the potential to serve as tool to explore and implement TD of the WoT and ensure interoperability in the WoT systems. To ensure the compatibility with the evolving W3C TD standard, the application will be maintained and updated to support the latest versions and specifications. Moreover, artificial intelligence and natural language processing capabilities should be integrated with the application. This integration will allows developers to input Thing descriptions using natural language in an intuitive and user-friendly manner. By leveraging AI-powered language processing algorithms, our application will be able to parse and interpret user input, extracting the relevant information, and automatically generating the corresponding TD attributes. A collaboration feature that enables multiple users to work concurrently on generating and validating Thing Descriptions.

References

[1] A web of things (2022). https://www.thingweb.io/

[2] Badii, A., Carboni, D., Pintus, A., Piras, A., Serra, A., Tiemann, M., & Viswanathan, N. (2013). City Scripts: Unifying Web, IoT and Smart City Services in a Smart Citizen Workspace. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 4*(3), 58-78.

[3] Brindha Merin, J., Aisha Banu, W., Akila, R., & Radhika, A. (2023). Semantic Annotation Based Mechanism for Web Service Discovery and Recommendation. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, *14*(3), 169-185.

[4] Cimmino, A., Poveda-Villalón, M., & García-Castro, R. (2020). ewot: A semantic interoperability approach for heterogeneous iot ecosystems based on the web of things. *Sensors*, *20*(3), 822. https://doi.org/10.3390/s20030822.

[5] Darabkh, K. A., & Al-Akhras, M. (2021). RPL over internet of things: challenges, solutions, and recommendations. *In IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, 1-7.

[6] Darabkh, K. A., AlAdwan, H. H., Al-Akhras, M., Jubair, F., & Rahamneh, S. (2023). A New Routing Protocol for Low-Power and Lossy Networks Utilizing Computational Intelligence over IoT Networks. *In IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*, 97-102.

[7] Darabkh, K. A., AlAdwan, H. H., Al-Akhras, M., Jubair, F., & Rahamneh, S. (2024). A revolutionary RPL-based IoT routing protocol for monitoring building structural health in smart city domain utilizing equilibrium optimizer algorithm. *Soft Computing*, 1-40. https://doi.org/10.1007/s00500-024-09677-0.

[8] Eiriemiokhale, K. A., & Olutola, J. B. (2023). Application of the Internet of Things for quality service delivery in Nigerian university libraries. *Indian Journal of Information Sources and Services*, *13*(1), 17-25.

[9] Faheem, M. R., Anees, T., & Hussain, M. (2019). The web of things: findability taxonomy and challenges. *IEEE Access*, *7*, 185028-185041.

[10] Halim, M., Tahiri, A., El Ghzizal, Y., Adadi, N., & Chenouni, D. (2024). Web Service-Oriented E-learning: Proposition of Semantic Approach to Discover Web Services Related to the E-learning System. *Journal of Internet Services and Information Security, 14*(2), 1-17.

[11] Iglesias-Urkia, M., Gómez, A., Casado-Mansilla, D., & Urbieta, A. (2020). Automatic generation of web of things servients using thing descriptions. *Personal and Ubiquitous Computing*, 1-17.

[12] Kassab, W. A., & Darabkh, K. A. (2020). A–Z survey of Internet of Things: Architectures, protocols, applications, recent advances, future directions and recommendations. *Journal of Network and Computer Applications*, *163*, 102663. https://doi.org/10.1016/j.jnca.2020.102663

[13] Khalil, R. A., Saeed, N., Masood, M., Fard, Y. M., Alouini, M. S., & Al-Naffouri, T. Y. (2021). Deep learning in the industrial internet of things: Potentials, challenges, and emerging applications. *IEEE Internet of Things Journal*, *8*(14), 11016-11040. https://doi.org/10.1109/JIOT.2021.3051414.

[14] Khoeurt, S., So-In, C., Musikawan, P., & Aimtongkham, P. (2023). Multidirectional Trust-Based Security Mechanisms for Sinkhole Attack Detection in the RPL Routing Protocol for Internet of Things. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, *14*(3), 48-76.

[15] Korkan, E., Kaebisch, S., & Steinhorst, S. (2020). Streamlining IoT system development with open standards. *it-Information Technology*, *62*(5-6), 215-226.

[16] Korkan, E., Kaebisch, S., Kovatsch, M., & Steinhorst, S. (2020). Safe interoperability for web of things devices and systems. *Languages, Design Methods, and Tools for Electronic System Design: Selected Contributions from FDL 2018*, 47-69.

[17] Kumar, R., Rani, S., & Awadh, M. A. (2022). Exploring the application sphere of the internet of things in industry 4.0: a review, bibliometric and content analysis. *Sensors*, *22*(11), 4276. https://doi.org/10.3390/s22114276.
[18] Laghari, A. A., Wu, K., Laghari, R. A., Ali, M., & Khan, A. A. (2021). A review and state of art of Internet of Things (IoT). *Archives of Computational Methods in Engineering*, 1-19.
[19] Lee, E., Seo, Y. D., Oh, S. R., & Kim, Y. G. (2021). A Survey on Standards for Interoperability and Security in the Internet of Things. *IEEE Communications Surveys & Tutorials*, *23*(2), 1020-1047.
[20] Majjaru, C., & Senthilkumar, K. (2023). Strengthening IoT Intrusion Detection through the HOPNET Model. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, *14*(3), 89-102.
[21] Nandini, G. M. (2024). IOT Use in a Farming Area to Manage Water Conveyance. *Archives for Technical Sciences*, *2*(31), 16-24.
[22] Negash, B., Westerlund, T., & Tenhunen, H. (2019). Towards an interoperable Internet of Things through a web of virtual things at the Fog layer. *Future Generation Computer Systems*, *91*, 96-107.
[23] Rhayem, A., Mhiri, M. B. A., & Gargouri, F. (2020). Semantic web technologies for the internet of things: Systematic literature review. *Internet of Things*, *11*, 100206. https://doi.org/10.1016/j.iot.2020.100206
[24] Stoyanova, M., Nikoloudakis, Y., Panagiotakis, S., Pallis, E., & Markakis, E. K. (2020). A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues. *IEEE Communications Surveys & Tutorials*, *22*(2), 1191-1221.
[25] Tran, N. K., Sheng, Q. Z., Babar, M. A., & Yao, L. (2017). Searching the web of things: state of the art, challenges, and solutions. *ACM Computing Surveys (CSUR)*, *50*(4), 1-34.
[26] Tzavaras, A., Mainas, N., & Petrakis, E. G. (2023). OpenAPI framework for the Web of Things. *Internet of Things*, *21*, 100675. https://doi.org/10.1016/j.iot.2022.100675
[27] W3C. (2021) Web standards. https://www.w3.org/standards/
[28] Wang, J., Lim, M. K., Wang, C., & Tseng, M. L. (2021). The evolution of the Internet of Things (IoT) over the past 20 years. *Computers & Industrial Engineering*, *155*, 107174. https://doi.org/10.1016/j.cie.2021.107174.
[29] Zhang, T., Gao, L., He, C., Zhang, M., Krishnamachari, B., & Avestimehr, A. S. (2022). Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine*, *5*(1), 24-29.

Authors Biography

Dr. Samah Rahamneh (Member, IEEE) received the M.E. degree from the University of Jordan, Amman, Jordan, in 2014, and the Ph.D. degree in electrical and computer engineering from Western Michigan University, Michigan, USA, in 2020. She is currently an assistant professor with the School of engineering, University of Jordan. She has publications in peer-reviewed journals and conferences. Dr. Rahamneh's research interests include cloud computing resource optimization and security, optimization algorithms, Internet of Things (IoT), and HW/SW partitioning in mobile cloud computing for big data applications.

Ala' Hanoun, is a Computer Engineering graduate from the University of Jordan. Currently, works as a Front-End Software Engineer, specializing in creating intuitive and dynamic user interfaces. Ala' has a strong interest in research related to the Web of Things (WoT) and the Internet of Things (IoT), aiming to integrate these technologies into user-friendly applications.

Dr. Fahed Jubair, graduated from Purdue University in 2014 with a Ph.D. degree in Electrical and Computer Engineering. He received his B.Sc. degree from the University of Jordan in 2007. Dr. Jubair is currently an associate professor of Computer Engineering at the University of Jordan. His main research interests include optimizing compilers, parallel computing, heuristic algorithms, and machine learning.

Prof. Khalid A. Darabkh, received the PhD degree in Computer Engineering from the University of Alabama in Huntsville (UAH), USA, in 2007 with honors. He is the recipient of the Most Cited Researchers Award at the University of Jordan at Scopus during 2017-2021. Prof. Darabkh is among Top 1% JU Researcher's List who published the highest number of quality manuscripts in Scopus. He serves on the Editorial Board of Telecommunication Systems, published by Springer, Computer Applications in Engineering Education, published by John Wiley & Sons, and Journal of High Speed Networks, published by IOS Press. At the end, he is engaged in research mainly on Internet of things, industrial smart cities software-defined networks, vehicular networks, flying ad-hoc networks, Fog networking, full duplex cognitive radio networks, queuing systems.