NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENT OF

ELECTRICAL AND COMPUTER ENGINEERING

MIGUEL CORREIA FRECHES

BSc in Electrical and Computer Engineering

INTELLIGENT RELATIONAL TAGGING SYSTEM FOR SMARTIFICATION BASED DASHBOARD

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon

September, 2022

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENT OF

ELECTRICAL AND COMPUTER ENGINEERING

MIGUEL CORREIA FRECHES

BSc in Electrical and Computer Engineering

INTELLIGENT RELATIONAL TAGGING SYSTEM FOR SMARTIFICATION BASED DASHBOARD

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon

September, 2022

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENT OF
ELECTRICAL AND COMPUTER ENGINEERING

INTELLIGENT RELATIONAL TAGGING SYSTEM FOR SMARTIFICATION BASED DASHBOARD

MIGUEL CORREIA FRECHES

BSc in Electrical and Computer Engineering

Adviser: Doctor João Filipe dos Santos Sarraipa
Invited Assistant Professor, NOVA University Lisbon

Co-advisers: Jorge Miguel da Silva Calado
Researcher, UNINOVA-CTS, NOVA University Lisbon

Examination Committee:

Chair: Doctor Ana Inês da Silva Oliveira,
Assistant Professor, NOVA University Lisbon

Rapporteurs: Doctor Pedro Miguel Ribeiro Pereira,
Invited Assistant Professor, NOVA University Lisbon

Adviser: Doctor João Filipe dos Santos Sarraipa,
Assistant Professor, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2022

Intelligent Relational Tagging System for Smartification Based Dashboard

Dedico aos meus pais, ao meu irmão e a toda a gente que me ajudou durante o percurso académico.

ACKNOWLEDGMENTS

Firstly, I would like to thank my adviser, Doctor João Sarraipa, for giving me advice along this journey and helping me accomplish my objectives.

I also want to thank my co-advisor, Researcher Jorge Calado, who taught me a lot and helped me throughout this process. All your advice will be taken from here on out.

I would also like to mention the important role that NOVA School of Science and Technology had in my academic life. These years were unforgettable in all aspects.

Obviously, I want to thank my mom, dad, brother, and grandparents for their support over the years. This one is for you, avô.

I also want to mention Carlos Caeiro, Pedro Amaral, and Pedro Oliveira, who were writing their dissertations about other smartification systems that created a connection with my work.

Mentioning my dog, who passed away during the thesis development, is also mandatory for me. Much love Jordan.

Last but not least, I want to thank Tatiana, Kiko, Tomás S., Tomás, Manuel, Miguel, Meess, Tiago, Inês, and Casegas. The moments we have been through these last couple of years are life memories. Puxa!

“The advance of technology is based on making it fit in so that you don't really even notice it, so it's part of everyday life.” (Bill Gates).

Abstract

It is mandatory to understand the customer in any process of product creation, and the process of developing new, innovative, and smart furniture is no exception to the rule. As a possible solution to achieve this understanding, an intelligent relational tagging system makes total sense, even because it is a process that involves not only the customer and experts but many interactions between them. These interactions require an intelligent system that can make relevant suggestions based on similar problems of other users, thus saving time and resources. The furniture it-self has already been invented, that said, the goal of this work is not to re-invent the design but to re-invent furniture functionality.

This thesis proposes the development of a web interface and an intelligent classification system to apply tags (taxonomy and folksonomy) to understand the user's needs for smart furniture, namely the application and integration of IoT systems in the furniture creation and manufacture process. The system will be developed to recognize a pre-existent classification associated with smart furniture solutions and allow the user to collaborate by classifying new solutions. After understanding what the customer is looking for, these classifications will be used by the system to make suggestions of possible solutions.

Satisfying the needs of the user and helping in the process of creating furniture utterly adequate to the user is the goal that this web interface and the tagging system aim to achieve.

Keywords: Smartification, Taxonomy, Folksonomy, Tagging, Knowledge Base, IoT

RESUMO

Entender o cliente é obrigatório em qualquer tipo de criação de um produto e o processo de desenvolver mobília nova, inovadora e inteligente não foge à regra. Como possível solução, em relação a perceber o cliente, um sistema relacional inteligente de tags parece fazer todo o sentido, até por ser um processo que envolve não só o cliente e especialistas, mas também muitas interações entre eles. Essas interações requerem assim um sistema inteligente com a capacidade de fazer sugestões relevantes com base em problemas semelhantes de outros clientes, economizando assim tempo e recursos. A mobília em si já foi inventada, ou seja, o objetivo deste trabalho não é reinventar o design, mas sim reinventar a funcionalidade da mobília.

O propósito desta tese é, desenvolver uma interface web e um sistema de classificação inteligente, para aplicação de tags (taxonomia e folksonomia), com vista a perceber as necessidades do utilizador em "smartificar" mobiliário, nomeadamente a aplicação e a integração de sistemas IoT no processo de criação e manufatura da mobília. O sistema será desenvolvido para reconhecer uma classificação pré-existente associado a soluções de mobília inteligente e permitir que os clientes colaborem classificando soluções depois de desenvolvidas.

Satisfazer as necessidades e ajudar no processo de criação de mobiliário, completamente adequado ao utilizador, é o objetivo que a interface web e o sistema de tags a desenvolver espera atingir.

Palavras chave: Smartificação, Taxonomia, Folksonomia, Tagging, Base do conhecimento, IoT

TABLE OF CONTENTS

TABLE OF FIGURES

ACRONYMS

3D	Three Dimensional
API	Application Programming Interface
IoT	Internet of Things
KB	Knowledge Base
KBS	Knowledge-Base Systems
KE	Knowledge Elicitation
NLP	Natural Language Process
POS	Part of Speech
RFID	Radio-Frequency Identification
TF-IDF	Term Frequency-Inverse Document Frequency
NLTK	Natural Language Toolkit
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
REST	Representational State Transfer
API	Application Programming Interface
ML	Machine Learning

1

Introduction

With technology evolving in a way that is becoming part of people's lives, making the daily routine easier must be the primary goal. Although almost everything in our world is changing to have technology embedded, furniture is not yet one of them.

In 1965 the co-founder of Intel said that the number of transistors packed into a given unit of space would double every two years. However, nowadays it is even faster [1]. From this fact, it can be inferred that with more transistors, a device can do more operations and that transistors are becoming smaller. With this statement established, we can acknowledge that with faster, smaller, and cheaper devices, it makes sense that sectors like health care, transportation, and education are including these devices to improve efficiency. This integration of technology in these sectors has already shown the benefits it brings to people's lives, so we can only assume that if we integrate technology in people's houses, in this case, in their furniture, there will be more benefits than disadvantages. In re-flection, if people look around, they see that they are surrounded by technology. However, that disappears in their homes only to be present in their devices or household appliances.

Since this integration between technology and furniture aims to help people in their daily routine, all these devices embedded in furniture must be connected to the Internet, falling into the Internet of Things technologies category. This type of technology seems to be the most evolving technology of our time, allowing users to be constantly connected to devices that surround them. Therefore, it is possible and makes all the sense to assume that giving furniture the possibility to collect, monitor, and analyze data from their surroundings in real time only brings benefits.

Smartification refers to the digital refinement of an existing product by embedding digital technologies and intelligent services [2]. However, the existing offer related to "smart" furniture is exclusive, costly, and does not allow much customization. The European project in which this thesis is involved aims to create a "do it together" ecosystem for the sustainable creation of furniture. Ideas for "smart" furniture can come from customers or inside the company. Sustainability, as it should be in further projects in any area, is also a significant concern, aiming to reduce the amount of CO_2 produced and create and maintain job opportunities.

This thesis plays a role in the project by developing an intelligent relational tagging system to be applied in an app/website that extracts information from customers relative to what they want to "smart".

1.1 Motivation

Aiming to make daily life more accessible has been the focus of the new technologies integrated into the life of human beings, and smart furniture works in the same direction. However, it is challenging to have standardized smart furniture that fits everyone, which can only be achieved if each customer develops their solution. This is where this works by understanding the user's needs and helping to find and develop the right solution.

Consider Bob, a middle-aged man furnishing the house that he just bought. Bob's objective is to have everything smart in his home. Besides the usual smart devices that nowadays are common, like a TV, a fridge, a washer, etc. Bob started to think about what more he could get smart in his home.

A pantry with a barcode reader so that every time he puts groceries there, the pantry keeps track of the expiration date, thus being possible to warn him if a grocery expiration date is running out. A table in his living room has a built-in wireless charger and charging cables for mobile phones. Kitchen cabinets with built-in lights inside and outside only turn on when he opens the cabinets or is nearby the cabinets, saving him much energy. Bob looked on the internet for options to accomplish what he imagined and realized that everything he found was not very customizable and did not go entirely with the idea he had pictured.

Bob's problem has two ways to be approached: what he searched on the internet does not exist or the words written do not match the idea in mind, which is very common when looking

for something new. This thesis and the project involving it respond to these problems by giving the possibility to the customer to create and shape the product in his way.

However, guessing with precision what the user is searching for raises many challenges. Taking the Google search engine, for example [3]. If a user types in "new furniture" the search engine searches the index to find every page that includes those words. This algorithm starts by analyzing the words because understanding their meaning is essential to return good results. Besides, the algorithm tries to understand the category of information it is searching for since this categorization leads to different results with the exact words in a slightly different phrase.

A slightly similar approach can be made to understand and suggest solutions to the customer by developing an intelligent relational tagging system. A good and efficient system that answers all the difficulties of developing a solution for each customer can easily be applied and integrated into other technologies with standardized solutions to have personalized solutions.

1.2 Research Question and Hypothesis

Taking into consideration the development of an intelligent relational tagging system to help users find what they are searching to develop a piece of smart furniture, a research question like the following can occur:

- In a furniture smartification project, how can a UI be employed to formalize and classify the customers' requirements/needs?

If a system can extract information using semantic deconstruction, then it is possible to help customers to find the right smart furniture solution and use that information to enrich a knowledge base (KB) for future use cases.

1.3 Objectives

This thesis aims to develop a web interface capable of identifying user needs related to projects of developing smart furniture. This web interface will use an intelligent classification system based on tags to understand what the user is trying to search.

The website will try to lead the user to the solution that he was looking for without much typing. However, if the user does not find what he is looking for, he can also write it down on

the website, and the intelligent classification system will do the rest and show a possible solution to the user based on what is in the KB.

This KB will be fed with the relation between the solution developed, the user search, and the choices made on the website that guided the user. Hardware companies will be able to insert hardware on the website, which will be inserted in the KB to be available as a possible solution to the customer. The hardware specialist will also feed the KB with suggestions and/or feedback. The system will also show possible configurations to the solution found, leaving the end choice always to the user.

The manufacturing process starts if the user is happy with the solution presented and the functionalities are validated. After the product is delivered, the user will be invited to answer a brief survey giving feedback about the whole process and going to the KB.

1.4 Dissertation Organization

This dissertation proposes an intelligent relational tagging system that helps the customer in the smartification process by providing suggestions. The organization is the following:

- 1. Introduction– This is where the problem is introduced, with the motivation, research question, hypothesis, and organization of the research work also being approached.

- 2. State of the Art – It states the current research work that already exists alongside definitions that are in some form related to the tagging system developed in the dissertation.

- 3. Scenarios and Case Studies – Introduces the scenarios that the tagging system needs to approach. It also exposes real-life case studies that can be used to test the system.

- 4. Architecture for the Tagging Process – Explains the system architecture and compares the proposed system with the existing one. Describes the system's flow and explains the chosen technology for the different parts of the system.

- 5. Implementation – Describes the whole system, the specifications, and features behind the application, created to support the Smartification process and the intelligent relational tagging system. Ultimately, it shows a real-life case using the application with the results achieved alongside each interface's purpose.

- 6. Conclusion - Concludes the work done in this dissertation, approaching what went well and wrong in a critical way. Defines the work that can be done to continue the investigation.

1.5 Original Contributions

The smartification and the scenarios in chapter 3 were inspired by the work already developed by the INEDIT project. The scenarios from chapter 3, the database, and the states diagram from chapter 5 are contributions from all the smartification systems that worked together so that the smartification process can be as fluid as possible. Some systems independently created some database tables, however *tagging_project*, *tagging*, and *tags_relations* tables are original contributions of this dissertation. The Prototyping System influenced the instructions regarding the idea, the detailed solution, and the prototype information since this system is responsible for creating the smart furniture details.

The tagging system implementation was based on implementations already done for other tagging systems. However, the original contribution of this dissertation was adapting those systems to be more specific for tagging information regarding furniture smartification. The smartification service architecture and implementation, including the interfaces, are all original contributions of this dissertation.

2

STATE OF THE ART

In this chapter, the main concepts will be clarified based on definitions of other authors that have relevant work in similar subjects, helping to understand better the technology developed. It will also be reviewed and analyze some of the solutions already available with similar purposes to this thesis.

2.1 Internet of Things

In 1999, Kevin Ashton introduced the Internet of Things to the world in a presentation. Since then, the term's popularity has been skyrocketing and, for most experts, is the most evolving technology of this era. Kevin reflected that computers and, therefore, the internet depend entirely on humans to have information [1]. Connected to this issue, people, in general, are not very good at capturing data about the world. With this reflection came the idea of enabling computers or any other device to gather information and use this data for the benefit of humanity.

IoT can be defined as a network of objects that usually have integrated technology that enables collecting data from their internal or external environment to communicate with a manufacturer, a maker, or even connected objects. Objects that are controlled by mobile applications (watches, fridges, stoves, cookers, etc.) and that are not connected to the internet and use other communications instead can also be included in the category of IoT [2].

This technology needs devices to sense, process, and communicate. With the sensors and actuators becoming more powerful, cheap, and smaller to fit discreetly in almost all devices, it is typical for interest in this type of technology to increase in people and various industries. It is already possible to see this technology in various industries.

In transportation and logistics, tracking the object's route in real-time with more objects containing bar codes, RFID tags, or sensors is possible. Such as tracking vehicles, monitoring the vehicle movement, or even predicting the vehicle location can help avoid crashes [3]. It has already been applied in the healthcare industry by collecting, managing, and sharing user information, reducing bureaucracy. However, in this industry, security and privacy are major concerns.

The concept of having all connected can easily be shifted to this thesis. In smart furniture, this type of technology should be present. This technology allows the user to control the smart functionalities of his furniture through the smartphone, tablet, or even smartwatch.

The integration of this technology is an advantage not only to the furniture owner but also to those who manage data collected through the sensors and actuators in the furniture. With the data collected, it is possible to understand what needs to be improved and even update software without being present. However, integrating this technology is a consumer choice and does not define whether the furniture is smart or not, being only an add-on to the solution.

However, to understand the specific technology that the customer wants to integrate in the furniture, it is necessary to understand the smartification need of the customer. Since this is a recent topic, the customer might not know exactly how to describe the smartification need. That is where it can be handy to use taxonomy and taxonomy algorithms.

2.2 Taxonomy

Taxonomy is a concept that has been studied more and more in recent years by computer science, and it can be considered a list of keywords that obeys in some way a logical condition [4]. Analyzing etymologically, **taxonomy** comes from the Greek *taxis* = ordering and *nomos* = rule, so we can define taxonomy as a classification system that orders information in a class/subclass by a set of rules, representing a simple tree structure [5].

The most general category of this tree is at the top, and it is considered the root node. In this type of tree, a node can be a **parent** and/or a **child**, except the root node, which is always a parent node. A parent node is always a more general category than a child node. At the bottom of the tree, the node is only a child node and cannot be more specific considering the concepts

it is organizing [5]. An excellent and relatable example of a taxonomy construction is presented in Figure 2.1.

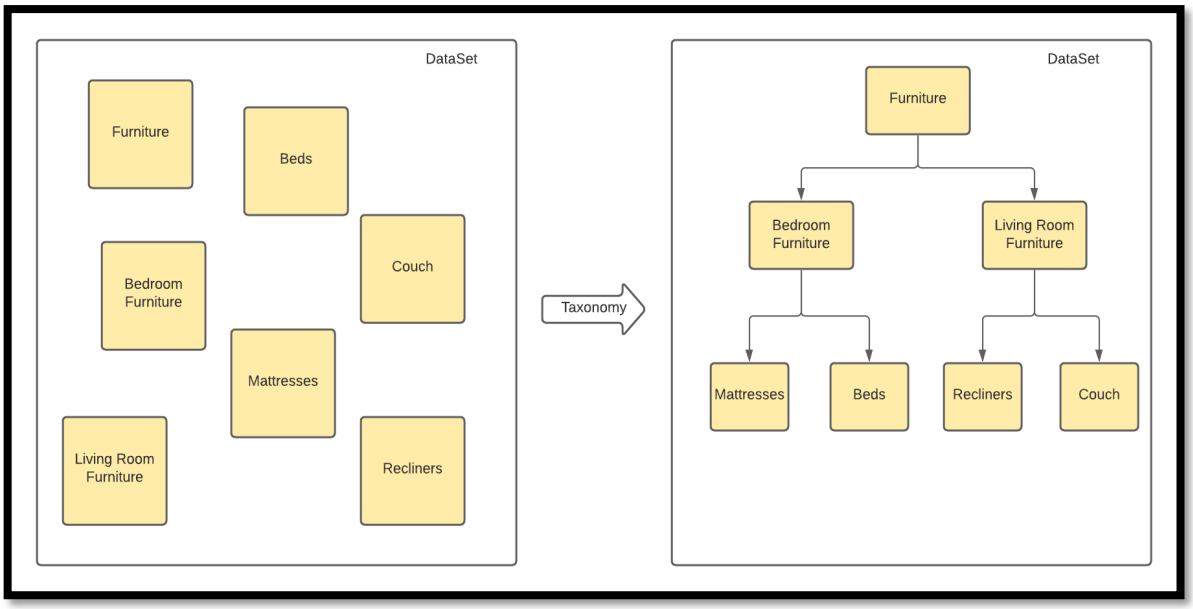


Figure 2.1 - Taxonomy Construction.

There are two ways of taxonomy construction: manual and automatic. The manual approach was the most used a few years ago when the domain experts were the ones that created the taxonomy [6]. However, this approach limits who can create an ontology, and since it is human-dependent, it is exhaustive labor when the dataset is huge. That is, it is not able to scale up. On the other hand, the automatic approach seems to be rising in the last few years, especially with the raising of Natural Language Processing techniques, Machine Learning algorithms, and Artificial Intelligence. Several automatic approaches have the advantage of handling large volumes of data, which is pretty easy to manage. However, some of them lack accuracy [6].

2.2.1 Taxonomy Construction

Regarding taxonomy construction, two techniques are used in almost every algorithm/approach: Top-Down and Bottom-Up [6].

The top-down technique uses the selection of a few numbers of higher categories reaching more specific levels of lower subcategories. This technique is primarily common in taxonomies created manually, which will not be the case in this dissertation. However, the bottom-up

technique is typically used to construct automatic taxonomies. Again, the idea is to select a specific level of categories and try to reach the higher categories.

In order to create taxonomies, another common technique is the approach of clustering. Cluster approaches aim to find the similarity between objects clustered via data segmentation into groups. In [6], there are presented three clustering approaches: Formal Concept Analysis (FCA), Hierarchical Agglomerative Clustering (HCA), and Bi-Section K-Means Clustering (Bi-Section K-Means). Even though these approaches have different ways of creating algorithms, there is a way of comparing them in terms of effectiveness, complexity, and traceability. In terms of effectiveness, it is possible to consider the three algorithms at the same level, good. For the complexity, there are already differences since the FCA has O(2^n) while HAC and Bi-Section K-Means have the same complexity, O(n^2). For traceability, the FCA is good, while the other two have poorer traceability. From this, it is possible to understand that the FCA is the best [7].

In [8], it uses clustering approaches together with adaptive term embedding to create the **Taxogen** algorithm. This algorithm joins all the concept terms into a latent space to extract their semantics and then uses them to build the taxonomy recursively. The result from this algorithm shows that compared to other taxonomy creation algorithms, Taxogen provides better relation accuracy and term coherency, creating a better taxonomy.

Regarding automatic taxonomy construction, the authors of the referenced paper [9] suggested obtaining knowledge and contexts related to the keywords. This is done using a known technique called **Short Text Conceptualization** and a general purpose knowledgebase called Probase[9]. This technique detects and maps terms in short text to instances and attributes in the knowledge base. Then, the new taxonomy is induced with the help of Bayesian inference and through the nearest-neighborhood-based methods to be more efficient and faster. Bayesian inference is a method of statistical inference that uses Bayes's theorem to evaluate the probability of a hypothesis over time, and new information stays available [10]. Therefore, this is a fantastic approach when aiming to develop a taxonomy that can be used to extract valuable information as well as make valuable suggestions. Over time, there will be more and more smartification projects, which will use the full potential of this algorithm, because it will evaluate with higher accuracy the taxonomy of each one of them and then make more accurate suggestions.

Another common approach to constructing taxonomy is using WordNet, a semi-supervised approach that creates from scratch taxonomies through the web hyponym-hypernym pairs.

These approaches learn automatically from the hyponym and hypernym using the root concept, which is a fundamental concept and recursive surface patterns [6].

In [11] it is introduced the **tax2vec algorithm**. This is an algorithm that builds a semantic feature vector that can be used to enrich vectors constructed by text processing methods such as term frequency-inverse document frequency (TF-IDF). TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This algorithm takes input documents and a word taxonomy and outputs a matrix of semantic feature vectors [11]. Furthermore, this algorithm implements multiple feature selection heuristics to show only a number defined by the user of relevant features only to show potentially relevant features. This is an approach that can be turned into something helpful for this thesis by using the algorithm in order to propose to the user relevant content based on the information provided by the user.

Besides algorithms and relevant approaches, some software already exists in the market involving taxonomy, ontologies, and the respective management. For example, one of the solutions of the company TopQuadrant is managing ontologies and taxonomies. This technology allows organizations to use standard industry taxonomies or develop their own, enriching their KB with concepts, terms, relationships, and rules. Furthermore, this solution makes part of the vocabulary management product TopQuadrant offers that focus on improving search and ensuring proper delivery and navigation of information through creating knowledge graphs of vocabularies and associated resources [12].

This categorization of information help organizations organizes their information. When running a website/application, having this type of organization is an important feature to make relevant suggestions to the user based on what he is seeing or looking for.

2.2.2 Taxonomy algorithms comparison

The previous sub-chapters presented various algorithms and software that could be used in this dissertation to achieve one of the goals: extracting the smartification need. It was acknowledged that there are two ways of constructing taxonomies, automatically and manually, each of them having advantages and disadvantages. However, it can be concluded that the automatic way seems to be the more efficient way because it would not be practical to create taxonomies for each smartification project manually.

With this statement established, it was presented the two techniques that are usually used to construct taxonomies: Top-Down and Bottom-Up. Since top-down is usually used for manually constructed taxonomies, the logical technique to adopt in this dissertation is the bottom-up technique. From the clustering techniques presented, it is possible to understand that the FCA is the best technique but is the one that has the most complexity. However, since taxonomies is not a new subject, some algorithms already incorporate several techniques in creating taxonomies. For example, the way suggested in [8] **Taxogen** algorithm produced better results in creating taxonomies than other topic modeling and embedding-based methods. However, this algorithm has the limitation of requiring pre-specified numbers of clusters when splitting a topic. When compared to [9], which uses the **Short Text Conceptualization** technique, the **Taxogen** algorithm is not the best option if the idea is to have a way to construct a taxonomy that gets better over time and utilization.

Tax2vec algorithm shows another approach to the problem by using statistical measurements with semantic feature vectors to evaluate the importance of a word to a document in a collection of documents. This is also an exciting and helpful approach. However, it is more focused on documents that need more effort to implement.

2.3 Folksonomy

A concept that has been growing in the last two decades is folksonomy, mainly because of the appearance of social networks with the possibility of users sharing information, photos, videos, etc. With this possibility came the suggestion of describing or identifying, through words, the content shared to reach more people.

So, **folksonomy**, which comes from folk and taxonomy, can also be called social classification, collaborative tagging, social tagging, and social indexing is the organization and categorization of content that a community of users tagged related to that [13]. In other words, it is an association that a community of users makes between annotations they find pertinent and content or web pages [14].

This type of tagging became very popular with the website del.icio.us, allowing users to save their favorite websites and associate tags with them. Besides that, it also made it possible for users to search for tags, resulting in more efficient results [13].

The idea behind folksonomy is to allow the user to create his tags, which are free of choice and not restricted by any vocabulary. With this, users that use the content can now categorize that same content, helping searches become more efficient and accurate. Furthermore, this type of tagging also makes it possible to suggest more efficient results related to what the user sees. This possibility can easily be shifted to this thesis since one of the objectives of this work is that the user is the one that is giving relevant information in order to use that information to make suggestions.

2.3.1 Folksonomy construction and algorithms

Since this is a topic that cannot be considered new these days, numerous folksonomy algorithms can be used. However, since one of this thesis aims is to guide and help the user find the right smartification solution through the extraction of information and the suggestion of similar projects, it makes total sense to choose the best folksonomy algorithm and tune it in for this thesis purpose.

In [15], it is evaluated state-of-the-art folksonomy algorithms. To do this, a subset was selected and applied to 20 folksonomies created from 5 social tagging system datasets. This experience showed that folksonomy algorithms developed for social tagging systems outperform traditional mechanisms [15]. With this experience, it is possible to start having an idea of the algorithm that this thesis will lean on since it is proven that some are more efficient than others.

Regarding folksonomies algorithms developed for social tagging systems, [16] proposed a method to disambiguate tags and incorporate knowledge from Wikipedia to build folksonomy ontologies for social tagging systems. The method applied showed that the created ontology was more effective in a search personalized task and that similar tags were clustered in high proximity [16]. As future work, which is in line with the objective of this thesis, is planned to apply information extraction/summarization technologies to focus on sentences provided from a dataset with relevant information on a subject.

This is fully aligned with the purpose of this thesis since it will be needed to extract valuable information from the user sentences to categorize them. The value extracted from this categorization can then be used to make relevant suggestions since it will be associated with high proximity to the clustered tags. Higher proximity tags will then be used to search for projects with those same tags, enabling the suggestions made to the user to be relevant and accurate.

Xu, Shengliang et al. (2008) presented a "**personalized search framework in which the users and the web pages are associated by a topic space**". This works by mapping into vectors the interests of the users and the topics of each web page. The topics in a topic space are very similar to the categories of a web page classification. The personalized search ranked the web pages by term matching and topic matching, so when a user searches for something, a web page is ranked by term similarity and topic similarity [14].

This work relates to web pages and users. This thesis focus is slightly different, although the main idea is the same, proposing a personalized search system where folksonomy helps deliver an excellent result to a search/query.

Another work worth to be mentioned is from Chen, Jie, et al. (2014). The main idea of this work is to consider **the user's motivation**. By retrieving the user motivation and adding it to the algorithm, it was possible to achieve higher accuracy and more stability than the traditional algorithms [17]. To achieve that higher accuracy, they compared the resources tagged by the users with the same motivation and recommended the same resources. Their experimental results showed higher accuracy and stability when recommending resources to users.

In [18], the **FolkRank** ranking algorithm is suggested aiming to generate personalized rankings of the items in a specific folksonomy as well as to recommend users, tags, and resources. This is an algorithm that, in his calculations, considers the structure of folksonomies and the results of an evaluation on a large-scale dataset. The results found with this algorithm are that the top folksonomy elements typically fall into a coherent topic area, which can be interpreted as extracting communities of interest from the folksonomy.

In [19], it is proposed that algorithms will be built upon cognitive-inspired tag recommendation to overcome the imbalance between tag recommendation approaches and folksonomy structures with cognitive-inspired algorithms. Although the results presented in this paper are precise, there is an evident difference between these two approaches since the algorithms are designed for dense structures while most social tagging systems are not [19]. The suggested solution here is having cognitive-inspired algorithms capable of modeling the tags vocabulary of a user in a cognitive-plausible way. This solution can be constructive when aiming to develop a tagging system that will partly interfere with folksonomies, which is the case of this thesis.

As it was said before, folksonomy is not a new topic. Much investigation has already been done, and many platforms are running software that answers most of the needs in this area.

There are already some systems in the market, such as Bismart Folksonomy. Bismart is a company that offers solutions in the integration, management, and analysis of data [20]. Folksonomy text analytics is one of the systems they offer and the most relevant technology for this thesis.

This software merges synonyms, separates homonyms, adds a technical or customized dictionary, reduces tags through blocklists, and even considers account errors and duplicate content [21]. In addition, it enables structuring data, analyzing text in real-time and data discovery by highlighting tags from texts, images, videos, and audio. However, this technology focuses more on big data sets, and this thesis focuses more on real-time searches [22].

The Bismart company already installed this type of technology in other companies in different areas, such as tourism, energy, insurance, public administration, etc., all with great success, which indicates that the same success can be achieved in this area.

It is possible to start organizing information from a large dataset or customer-written sentences from concepts such as taxonomy and folksonomy. However, tagging is another method to retrieve and organize information from large datasets or minor sentences.

2.3.2 Folksonomy algorithm comparison

To develop the right system to approach customer smartification, an analysis of the previously presented algorithms must be done to find the right solution.

The algorithm in [16] presented a method that incorporates knowledge from Wikipedia to build folksonomy ontologies for social tagging systems. Complemented with the conclusions retrieved in [15], folksonomy algorithms built for social tagging systems work better than the others, this seems to be the suitable algorithm for folksonomy.

The **FolkRank** ranking algorithm aims to generate personalized rankings of the items in a specific folksonomy and recommend user tags and resources. However, as concluded in [23], this moderate old algorithm cannot use the user-user, item-item relationship very well and lacks results compared to newer ones.

The **personalized search framework in which the users and the web pages are associated by a topic space** presented in [14] seems to be a way that will only start making good recommendations after some use cases. Since there are other ways to make suitable recommendations, the framework presented lacks efficiency initially. The work done by Chen, Jie et al. (2014)

seems to overcome the previous framework. The presented tool showed higher accuracy and stability when recommending resources to users when considering the user's motivation. This tool is relevant to this dissertation since the proposed system considers the user's motivation to do the tagging.

2.4 Tagging

To understand tagging process, it is necessary first to understand what a tag is. A **tag** is a user-generated keyword that has become a way to improve descriptions of online information resources [24]. The growth of this topic is explained by the considerable rise in popularity and use of social networks.

The process of **tagging** is a procedure that annotates various types of content, such as images, videos, books, etc., through web-based services, thus ensuring the effective organization of large amounts of information [25].

Although this is mainly used in social media, tagging can be convenient when searching for something. The primary purpose that users use tags is that they can label or categorize resources in a shared environment, doing that without realizing that with tags, they effectively share and organize a vast amount of data.

From the point of view of a developer, tagging is beneficial because the users classify each piece of resource with a tag becoming easier to organize information. From this organized information that the users voluntarily create, it is possible to retrieve valuable associations that are not visible to the naked eye. These associations can be low-level, such as color or visual element, or more complex, like a specific tag that from the begging has nothing to do with the content but several users have associated [26]. Recognize and use these associations, and facilitate the index of webpages and associated content that search engines like Google use to show the information to the user.

For the context of this thesis, the dataset that will be tagged is the information collected from the surveys that will enter the KB. In addition, the sentences written by the user to explain the problem will also be tagged to search the similar tag solution in the KB.

2.4.1 Tagging Algorithms Comparison

To process the information written by the user, it is possible to use **Part of Speech (POS) tagging**. POS tagging associates each word in a sentence with a POS tag. This type of tagging is mandatory for a standard set of tags to work. The standard set of tags chosen is the Nouns, Verbs, Adjectives, etc., although in this case, it makes sense to choose tags as furniture, solution, problem, etc. [27].

POS tagging has three approaches: rule-based, statistical, and hybrid.

In rule-based tagging, a set of rules is applied along with the contextual information to assign tags to words. In this approach, in order to be efficient, a large number of rules must be used.

In the statistical tagging approach, frequency and probability must be considered. This approach checks the most frequently tag associated with a word in a text that already used this software and uses this information to associate a tag to a word in a text that is running this software for the first time.

In [28] it is presented an evaluation of different methods for unsupervised POS tagging through various variations of **hidden Markov model taggers**. Hidden Markov model generates a sequence of states and then generates each word conditioned on the corresponding state[29]. With these variations, it was discovered that 'the quality of the lexicon made available to unsupervised learner made the most significant difference to tagging accuracy. So, by this, it is concluded that the bigger the context size, the higher performance of these models. The referenced paper also clarified the importance of clean dictionaries/datasets.

However, other papers refer to some algorithms that show better efficiency when compared to Hidden Markov Models. For example, in [30], the results clearly show that the **Viterbi algorithm** is more accurate than the Hidden Markov Models. Viterbi algorithm is a dynamic algorithm in POS tagging that finds the order of the word class – tag.

The hybrid approach first uses the statistical approach and then the hand-coded rules [27]. A hybrid approach was used in [34] to build a tag hierarchy. First, the steps were extracting data from communities, defining computational methods for co-occurrence, and proposing an unsupervised hierarchical building algorithm. Secondly, design a set of optimization algorithms to get a more accurate hierarchy. The results are precise, after the optimization, the hierarchy construction is so much more accurate than compared without the optimization [31]. This type

of optimization can easily be applied to the final tagging system that this thesis uses. Since the tagging system will be used to tag user information, it makes sense to have a better hierarchy in which each node of the hierarchy tree can be a possible suggestion to the user.

MonkeyLearn seems to be the company with the most exciting technology relative to this subject. Their API brings fast and robust text analysis into apps, as they promise, which is the critical technology here. They offer the possibility of building custom topic classification, importing data, defining tags, and instantly making the trained model available for production. Using pre-trained classifiers is also an option instead of creating one [32].

Their technology enables developers to make a custom classifier and implement it in whatever the developer wants, as long as it makes sense. It starts by asking what kind of classifier is desired, topic classification, sentiment analysis, or intent classification [32].

Being a pre-dataset imported, it is time to define tags and train the model by associating the tags chosen to some sentences from the imported text data. After finishing this association, the model is already smart enough to classify text input with the selected tags.

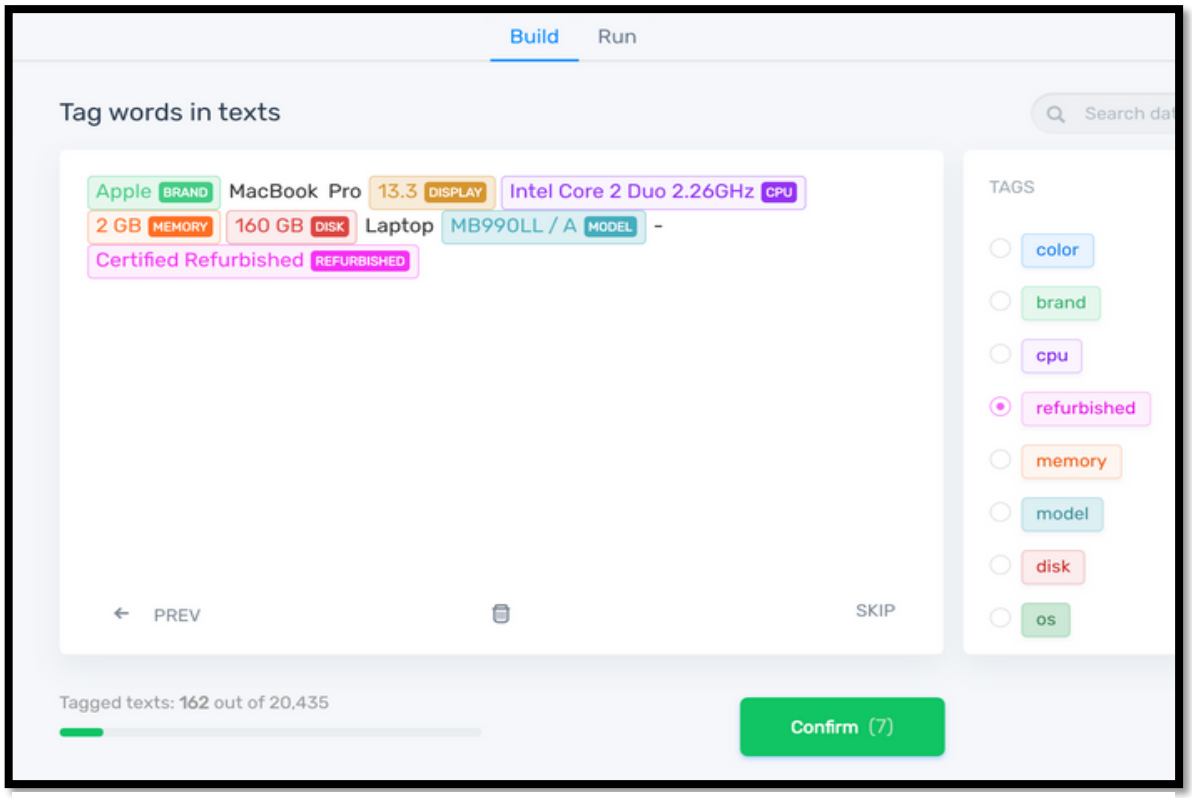


Figure 2.2 - MonkeyLearn Software.

Figure 2.2 is a downloaded photo from the MonkeyLearn website showing the result of their text analysis software. As it is possible to see, the software can categorize information into chosen tags from written sentences.

This is a perfect example of one of the features this thesis aims to develop, making it possible to extract valuable information from customer-written sentences and transform that information into the best solution possible.

Companies like DELL, Drift or Promoter.io are already using MonkeyLearn technology for a purpose similar to this thesis, classifying and analyzing text. To this thesis, it makes all the sense to have this kind of technology in the search engine that the customer will write to expose the problem and possible solutions. With a well-trained classifier with the correct tags, the extracted information from the customer's sentences is the indicator of developing a proper solution.

Regarding the implementation in this dissertation and for the end goal of this work, the major downside of this technology is that it is paid. Nevertheless, the tagging flow that is possible to retrieve from their technology can be a model to follow. With a process flow defined to follow, it is possible to make minor changes appropriate for the objective of the proposed system in this dissertation.

Another solution that can be used to interpret what the customer is writing is some of the existing tagging software, such as the PoolParty technology solutions. PoolParty is a company that allows organizations/projects/companies to organize and manage their knowledge through their "PoolParty Semantic Suite" [33].

PoolParty PowerTagging technology promises, through a combination of semantic search and artificial intelligence, to get documents better classified and relationships created between documents, thus providing more relevant content to the user. The important thing to retain from this technology is the workflow used to tag. They start by enabling the user to select documents to be tagged by the PoolParty extractor. Then return the tags and save those same tags to the content management system (CMS).

The flows presented in MonkeyLearn and PoolParty PowerTagging technologies can serve as a base model to develop this dissertation's tagging system.

However, it also exists some free tools that can be used. Of course, these systems/algorithms are not expected to have as many tools and accuracy as **MonkeyLearn** or **PoolParty PowerTagging** technologies.

In [15] is presented the **TreeTagger**. This probabilistic tagger differs from the others because the transition probabilities are estimated in a decision tree here. The results there show higher accuracy compared to the standard trigram taggers. From that, it was concluded that this tool is a fast and high-quality tool for annotating corpora for POS tagging. Based on this algorithm, the RNNTAgger was presented as an evolution of TreeTagger. Even though compared to TreeTagger, it has higher accuracy, lemmatizes all tokes, is slower, requires an improved GPU speed, and larger parameter files.

Similar to RNNTagger but with more analysis and information exists TextInspector. **TextInspector** is a web-based language analysis tool that gives detailed information regarding readability, complexity, lexical diversity, and critical statistics [15].

Figure 2.3 exemplifies the end work of an algorithm like **TextInspector**, which classifies each word of a sentence into the tags chosen, which are, in this case, Noun, Verb, Pronoun, Determiner, and Adjective.

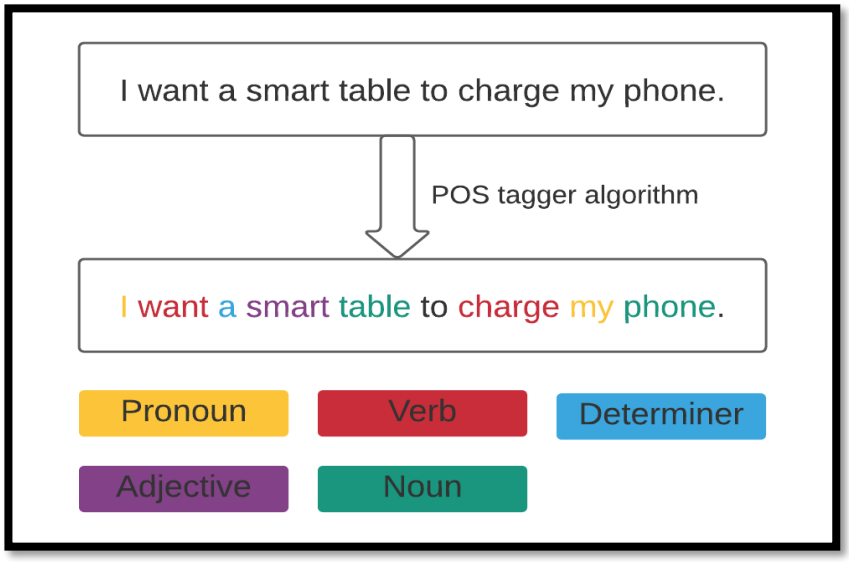


Figure 2.3 - POS Tagger example, created based on Text Inspector software.

Much valuable information is expected to be extracted and created from tagging, folksonomy, and taxonomy, so it makes sense to store that information somewhere so that the whole process does not have to be repeated. So that is where the KB enters.

2.5 Knowledge Base

The **knowledge base (KB)** is a term that has changed and evolved over the years. It started as a term to describe systems that were more complicated than simple relational databases. Nowadays, a **KB** can be considered self-service storage of information that is easy to extract about a product, service, department, or topic. Basically, it is a knowledge hub containing information that can be used to provide good customer service and help the employees find the information they need, quickly and efficiently. The knowledge bases can be internal, external, or both [34].

Internal knowledge bases are created to be used only by a company's employees to help them be more efficient in their tasks or find needed information. Coding practices for developers, company policies, and guidelines for creating content are often resources introduced in internal knowledge bases [35]. Fundamentally, anything that can and should be shared by all the employees is introduced in the KB for easier access.

Unlike the internal ones**, external knowledge bases** focus on the public and serving existing customers or someone looking for information about the company [35]. Usually, the FAQ or Help and Documentation sections are the external bases of the companies.

For the context of this thesis, using a KB makes total sense because since it is part of a project involving other intervenients, having everyone working with the same guidelines and on the same database brings efficiency to the process.

Several companies owe their success to top-notch knowledge bases. Spotify is a model in this case. Their help center is helpful, organized, and extremely simple. It also offers a great way to value the user's feedback by asking questions that can quickly be answered with Yes or No [35]. This is also an essential feature of this thesis, where besides helping the user find the right solution, it will also value the user's after-sale feedback.

Knowledge-based systems (KBS), or knowledge base software, can be defined as computer program that captures and uses knowledge to solve complex problems and extract relevant information for users [36]. Learning is a fundamental element of a KBS because it improves the

system over time, thus delivering better responses to the user. To improve the accuracy of this responses, artificial intelligence is used [37]. These systems are generally designed with the same architecture: **User Interface, Inference Engine, and Knowledge Base** [38].

The user interface is the component that the users interact with to retrieve information stored in the KB. The logic rules and the coding applied to the KBS to retrieve answers from the knowledge base are made in **the inference engine**. This engine is the key to providing good answers to the user because all the information and decisions are based on these rules. Finally, as explained in this topic, the KB is where the actual knowledge is stored [36].

However, this knowledge must be structured and modeled to be stored in the KB**. Data modeling** is the process of creating a data model so that it is possible to store data in a database by defining the conceptual representation of data objects, associations between them, and rules. The main focus of the data models is on what data is needed and how it is needed to organize data.

There are three main types of data models: **Conceptual data model, logical data model, and physical data model**. Typically, these types follow a sequential path with the objective that it is possible to implement the data model.

The conceptual data model maps out the types of data needed regarding the business application process. It is in this model that the rules and concepts are defined [39]. When finished, it can be used to create a logical data model. This model shows how the data entities are related, and the data is described from a technical perspective, which is used to understand the required application and database designs. Logical and conceptual data models are not connected to any specific technology platform. Once these two are defined, creating the physical data model is possible. This model defines the structure of the database or the file system in order to store and manage data. This includes defining tables, columns, constraints, etc. [40].

Several data modeling techniques define these types of data models: **Hierarchical Technique, Object-Oriented model, Network Technique, Entity-relationship model, and Relational Technique**[39].

The hierarchical Technique aims to organize data in a tree-like structure. In this technique, a child node can only have one parent, which makes this method a one-to-many method. This technique is not very used nowadays however, it served as a base model for the very used method today, XML, Extensible Markup Language.

Network Technique was also a very used data modeling technique that nowadays is not very used. This evolution of hierarchical modeling allowed child nodes to be connected to multiple parent records.

Aiming to have a more flexible technique than the two above, it came up **Relational Data Technique**. This technique maps the relationship between data elements stored in different tables. As an evolution of this technique, it came Entity-Relationship data modeling. This allows the mapping of entities, attributes, and relations between them visually. It is considered one of the most efficient ways to capture data and update processes by providing a better view of the data. This was the technique used to create the data model for the implementation of this system, which allowed the several systems involved in the implementation to get a better overview of the data process.

Another benefit of data modeling, when aiming to create a knowledge base that will be extended to multiple systems, is that it is possible to agree on data definitions and standards, thus facilitating the implementation of each of these systems. Another benefit of doing this, focusing more on the objective of this thesis, is the better use of available data assets. This is a fundamental goal regarding this dissertation because it is mandatory to use the data in the knowledge base provided by the other users to make good and relevant suggestions to the customer need. However, since this is a new project with an innovative goal, some requirements that can directly interfere with and ultimately change the current knowledge base data model might be missing.

2.5.1 Knowledge Base Software

Besides this being a technology that is evolving, already exists some paid software that is worth mentioning.

Document360 is a company faced with helping companies build their KB. They offer a KB site for users and employees in their technology, a KB assistant, a KB portal, an Application Programming Interface (API), and integrations and extensions. Their solution is already implemented in several companies, such as Spryker, Mambu, or Customer.io, covering different market areas [41].

Stonly is another company that offers a technology solution worth mentioning. They offer the solution of building interactive guides to lead the users to the solution and the possibility of launching a knowledge platform quickly. To make the guides as effective as they can be to

each user, their KB can learn and show solutions to the users instead of articles. This guide aims to lead the user to a solution by narrowing it down hypothesis [42].

The **GuruScan** company also has a technology that must be taken into consideration. They guide themselves by the slogan "Find knowledge in your organization" which says a lot about the technology they offer. Helping to find hidden knowledge in an organization is one of the features they offer with their technology, which is a pretty relevant topic for this thesis, knowing that it will involve many people. GuruScan's technology also allows the integration of other departments or companies into one KB without much effort. This is also an exciting feature compared to this thesis because it allows the project to scale up, which is one of the goals of this thesis project.

HelpSite is also a simple knowledge base software that can help publish a Knowledge Base with FAQs and articles by allowing the user to create a knowledge base fully customized and hosted on a specific domain name. The benefits of using this software are: It is effortless to use; the user can select the content to be public, partial public, or entirely internal; the layout of the KB is also fully customizable and also provides a smart contract form that automatically suggests information as the user types the message.

Helpjuice is a software-as-a-service platform that provides a beginner-friendly knowledge management system with excellent customer support. It also provides free customization to set up the knowledge base quickly and easily. In addition, this is a cloud-based knowledge base system that enables the possibility of having a self-host Helpjuice on the server.

Comparing them, some of them may have more advantages than others. For example, Document360 and HelpSite offer a very easy-to-use interface and the option of having the knowledge base public, semi-public or private. However, Document 360 offers analytics, while the HelpSite does not. On the other hand, Helpjuice is very similar to Document360 by offering a simple-to-use interface alongside the analytics tool. However, it is much more expensive than the offers of the Document360 software.

Like the software described above, extensions or features have many similarities and differences. However, since they all are paid software, the companies never genuinely share the technology that made their software work. Therefore, one of this thesis focuses on achieving similar software to be applied in the project, aiming to have all the knowledge in one place, making it easier to develop the product, help the customer and find the right solution.

Besides introducing valuable information from customers through processes like tagging into the KB, valuable information can also be extracted from experts or documents written by experts, which is the theme of the next sub-chapter.

2.6 Knowledge Elicitation

With the idea in mind to enrich the KB, it is necessary to introduce valuable knowledge to have better solutions for the customers. Therefore, acquiring that knowledge from the experts in the areas that matter to this thesis is necessary.

Getting information from experts can be called **knowledge elicitation** (KE). KE can be defined as a group of techniques and methods to extract knowledge, by direct interaction, from an expert in a specific domain [43]. Although this is not a trivial task, there is no defined methodology to do this it will be presented as the most common technique. A little exercise can be done to understand the difficulty of doing this, it is needed to put ourselves in the position of a specialist in a subject and formalize all the knowledge we have on the subject [44]. Not an easy task.

The efficiency of the methods and techniques used to elicit knowledge varies depending on the knowledge domain's complexity and the problem's size. Some of the methods are appropriate for certain domains but not for others. However, if the goal is to obtain knowledge asking directly to an expert or a group of experts is an obvious method, and it can be called an **interview** [45]. The success of this type of interview depends on the questions asked and on the ability of the expert to explicit their knowledge [46].

Another frequently used technique is **protocol analysis**, where the expert is asked to perform a task while thinking aloud. It is known that experts find it easier to talk about specific examples than abstract terms, thus becoming a way to reduce the inherent problems of interviews [46]. In addition, by concentrating on real cases and talking about them with the experts, the conversation is generally more centralized in only one subject, and the expert opinion is more coherent and structured [45].

Induction is another method, however, a controversial one. Induction is the opposite of deduction since it is possible to deduce facts about specific cases in a deduction from a general rule. In induction from specific cases, rules can be induced [45]. This method is objective, repeatable, and consistent, making conclusions from the results more accessible. Several algorithms that

induce rules from data sets are inherent to this method, usually used in machine learning and natural language processes [29].

Like these methods described, many more can be applied to extract knowledge from the experts. Retrieving this knowledge is an essential part of this thesis because smartification is not an easy subject in which anyone can give a valid opinion. Even though the customer is the person that is going to suggest the smartification solution, the expert is the one that decides if it is possible to perform or not. Therefore, even though they are not experts, retrieving the proper knowledge from the customer is also an important task related to this topic.

However, this knowledge extraction is something that nowadays machines can already do by us humans, and typically faster and with higher accuracy. In the following subchapter, a way of extracting information will be presented as well as several techniques.

2.7 Natural Language Processing (NLP)

NLP started to be approached in the early '50s as an intersection between artificial intelligence and linguistics [47]. **NLP** is a branch of computer science concerned with understanding text and spoken words in the same way and as quickly as humans do.

This technology combines computational linguistics with statistical, machine learning, and deep learning models to enable machines to process human language [47]. Nowadays, this process can already be done in the form of text or voice data and even make suggestions based on the speaker/writer's sentiment.

Four key factors enabled the quick incorporation of this technology in the most used devices: the quick increase of computer power, availability of a large amount of linguistics data, high accuracy ML methods, and a better understanding of the structure of human language [48].

The first approaches to computational linguistics were to write down the rules and vocabularies of human language for computers. However, it has proven to be a very time expensive and difficult task, mainly due to the variability, ambiguity, and context dependency.

Nowadays, the best systems use advanced Machine Learning (ML) algorithms as the primary approach to identify syntactic and semantic information about the context of each sentence.

The application of this technology can be extensive. However, the most common applications are Information Retrieval, Information Extraction, Question-Answering, Summarization, Machine Translation, and Dialogue Systems [49].

By the described applications, it is possible to conclude that this is an exciting technology regarding the aim of this dissertation. Ultimately, NLP must extract meaning ('semantics') from text or voice data, extracting relations between sentence elements like nouns, verbs, adjectives, etc., which goes in the same direction as this dissertation.

2.7.1 NLP Tools

The Natural Language Toolkit (**NLTK**) is one of the leading platforms regarding NLP tools. It uses python language in order to handle human language data. It has more than 50 trained models that can be used alongside a large panoply of libraries that can be used for classification, tokenization, tagging, etc.

NLTK is also a free, open-source, and community-driven project that can be considered the most robust tool referenced in this work and an excellent candidate to be applied in this thesis. Furthermore, this tool can easily be used in several cases for this thesis, enabling the POS tagging system to work together with a similarity algorithm, thus creating a folksonomy [50].

The major drawback of NLTK is that it requires significant resources if the dataset is a massive amount of data, which can be a problem for future work in this dissertation.

For this level of implementation complexity and considering all of the benefits mentioned about NLTK, this seems to be the platform that best fits this thesis, checking many of the requirements necessary to achieve the final objective of this thesis.

A similar NLP tool is the **CoreNLP**. This tool runs in Java programming language and enables users to derive linguistic annotations for texts, including token and sentence boundaries [50].

CoreNLP is also a free and open-source tool with the significant advantage of high scalability, unlike NLTK. As a result, this is an excellent tool to scrap information from open sources, analyze the sentiment in sentences, and an excellent option for text processing and generation.

A more straightforward tool that has fewer additions and fewer features when compared to CoreNLP and NLTK is **OpenNLP**. This tool is a machine learning-based toolkit with the same goal as the others, to process natural language text [50]. It supports most of the everyday NLP

tasks and is the right solution for tokenization, POS tagging, and sentence detection when aiming for a more simple but valuable and accurate tool for NLP.

2.7.2 GPT-3

The company OpenAI has its mission to guarantee that Artificial Intelligence benefits all humanity. From multiple software developed by this company, **Generative Pre-trained Transformer 3 (GPT-3)** is a deep learning model capable of producing human-like text. GPT-3 is the third generation of GPT language models that have as main functions the capabilities of classifying text, generating text, summarizing text, or even question answering [51]. The main difference of this third model from the previous is the contained mode size, being 100 times larger than the previous model in terms of parameters. This is considered by many the cutting edge of NLP.

The code is unavailable to the public, as the model is only available through a provided API. Besides that, the model is still in beta, and it is necessary to fill out a form to join the waitlist.

Through the use of semantic analytics, the model is capable of having a conversation by creating entire sentences. The model does this by considering the words and their significance and analyzing each word's context in the sentence [52].

With particular adaptations, this model could be advantageous in this dissertation. By having a model that can perform reading comprehension with a vast dataset of 175 billion parameters, this comprehension would be perfect for interpreting user sentences. This would then be used to provide functionalities suggestions and filter the suggested projects.

However, this model, like all other machine learning models, has certain limitations that must be considered. This model's slow inference time is one of them [51]. The model takes quite a long time to produce predictions which can be a negative point when using it alongside responsive interfaces where the user expects to have a dynamic and fast interaction. Besides that, GPT-3 has a lack of interpretability, which is a common problem of models that have massive datasets.

2.8 Accessibility and Usability

When designing a website or a web tool that everyone with access to the internet will use is essential to have in mind people with disabilities, and that is where accessibility and usability

are essential. **Web Accessibility** is the practice of making the website more usable by as many people as possible. This covers not only people with disabilities but also people with any hardware, software, language, location, or even ability to browse the web [53].

That is why there are **four principles of accessibility**, which, if followed, serve as guidelines to guarantee the possibility of using websites and web tools to everyone: **Perceivable, Operable, Understandable, and Robust** [54].

Perceivable is the capability of having the information as easy as possible to be processed. In other words, having the information available not only in one format provides for those who cannot read the option of listening and for those who cannot listen to the option of reading the information. So if a person cannot comprehend the content, the website or web tool cannot be considered perceivable [55].

Operable is the necessity of providing different ways of using the website or the web tool. The most common tool to browse through a website is the mouse. However, alternatives such as the keyboard can be used to navigate. This principle also covers the idea of providing a generous amount of time to perform tasks, be possible to control animations and media, and provide instructions [56].

Understandable is having the information crystal clear because the website/web tool can be perceivable and operable but not understandable. To achieve this, the information must be readable through clear, objective, and easy-to-comprehend language. To consider a website/web tool understandable, it is also necessary to behave as it is supposed to be predictable and help people not make mistakes through input assistance [57].

Robust is the need to have a technology that can be used by anyone, by any browser, any device, and that works well across platforms, of course, within reasonable limits. Although errors are inevitable and are part of the creation process, following development standards and conventions when writing code is a way of reducing the possibility of these occurring [58].

Allied to accessibility and the need for an interface that everyone can use comes the term usability. **Usability** is one relevant factor of a Web application quality that aims to provide a solution to the problem that is the interaction between technology and some users.[59] The standard defines usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use” [60]. This standard was defined in ISO-9241-1, however the definition that the community

adopted is the one introduced by Nielsen. Nielsen referred to usability using five concepts necessary to have in consideration when developing a user interface: **Learnability, Efficiency, Memorability, Errors, and Satisfaction** [59].

Learnability can be verified by evaluating how easy it is for users to do basic tasks and learning how the system behaves.

Efficiency is how fast and productive the user can perform the tasks once he/she has learned how the system works.

Memorability refers to how fast a user remembers how the system works after a long time of not using it.

Errors are the capability of having a system with few errors to support the errors that users usually make and recover quickly.

The satisfaction measures how much the user found the system easy and pleasant to use.

Regarding these accessibility and usability subjects, one organization is mandatory to talk about. World Wide Web Consortium (W3C) is an international community that works to develop web standards to lead the Web to its full potential [30].

In order to create these standards, the main activity of this organization is to develop protocols and guidelines to ensure the growth of the Web. When designing these principles, one principal goal is to make the Web available to all people, 'whatever their hardware, software, network infrastructure, native language, culture, geographical solution, or physical or mental ability.

For these purposes, a list of accessibility requirements for websites, web applications, browsers, etc. must be followed when developing one of these technologies. Therefore, IEEE, which also has one of the missions to provide an accessible web presence to all people, decided to follow the W3C 2.0 Level A accessibility encouraging the web publishers to follow even higher accessibility levels.

As is expected, the biggest technology companies in the world have been adapting and creating their accessibility standards and best practices, giving the developers a structure to follow when developing interfaces.

In this article [61], Microsoft gave an overview of concepts and technologies related to accessibility that must be taken into consideration when developing Windows apps. It is demanded

that support for keyboard interactions and screen readers, support for user customization, such as font, zoom, color, and high contrast settings, as well as alternatives or supplements for parts of the UI. This article also gives examples of assistive technologies, screen reader supports, and keyboard support controls, followed by what are the main aspects of accessible texts and how to support high-contrast themes. Even though these are all principles and standards that must be considered when developing Windows apps, some concerns are transversal to all development of UI.

Apple had the same idea as Microsoft and presented its access best practices to developers. In [39], Apple defines the main priorities, in app design, as **simplicity** and **perceivability**. **Simplicity** is defined as enabling familiar with consistent interactions that make complex tasks simple, and **perceivability** as making sure that all content is achievable by everyone. Their best practices also mention that personalization must be supported to personalize their user experience. An inspector tool is also available that helps the developer audit, test, and fix the application regarding accessibility features. The conclusion made from the presented Microsoft principles can also be applied here. Even though these guidelines are for apple apps, they can be extended to other use environments.

In order to explain why the use of accessibility guidelines, Google referenced 'The World Bank' research that states that 15% of the world population has some disability and that accessibility concerns can improve the app's versatility [47]. The suggested guidelines from Google when building applications were **to Increase Text Visibility, Use Large and Simple Controls, and Describe each UI element**. In order to facilitate the developer's work of testing accessibility, Google also presents an Accessibility Scanner app and a way to integrate an Accessibility Test Framework in the code.

Multiple frameworks began to be developed to address all the challenges of developing an accessible web. All these frameworks address a way to solve the most common difficulty when delivering web accessibility: **clarity** [62].

The Accessibility Roles and Responsibilities Mapping (ARRM) framework is worth mentioning. This framework was created by W3C, aiming to facilitate the process of web accessibility in the context of organizations/companies. ARRM proposes a decision tree method that helps the teams sort out the tasks for each team member based on their roles and skills [62]. The defined framework is mainly composed of three steps. It starts by identifying team roles based on ARRM role definitions, and then accessibility tasks are distributed based on these roles

concluding with making a checklist of actions to implement [62]. Adapting this framework usually leads to a great collaboration of the team and knowledge between it. In addition, this leads to clear documentation of each team member's responsibilities and tasks, resulting in greater team efficiency.

Another framework worth mentioning is the **USEFul** framework. USEFul leans more towards the automation part of testing the web accessibility development. This framework aims to achieve a fully automated usability evaluation process to facilitate a usually forgiven task when web development [63]. This proposed autonomous process is done by analyzing the frontend code of the website and crossing that information with accessibility principles. Besides that, it also inspects the website to check for any violation of the usability guidelines [63].

These are all concepts and information that must be considered when developing a GUI that can be used by anyone who wants to design smart furniture. Not being possible to adopt all this information and frameworks in the development phase of this dissertation, they are all essential to address and consider. The relevance of these topics becomes even more critical when it comes to such a difficult and abstract topic that it is smart furniture. For that, it is necessary to have all of these concepts well defined and try to apply them in developing this technology.

3

Scenario and Case Studies

This chapter presents the scenarios that support the development of this smartification system and the case studies that will serve to test it. There is also going to be presented the context of the scenario and the direct intervenients, as well as the principal scenarios.

3.1 Scenarios

This dissertation is part of a more significant European project called INEDIT. This project targets the creation of an Open Innovation European "Do It Together" ecosystem to co-create sustainable furniture [30]. The project also wants to explore the innovation around Social Manufacturing within the Circular Economy with global design but local production. The idea is to enable the consumers' creativity, go through the designers' professional skills to shape the product and then deliver a sustainable, smart, personalized new product quickly.

INEDIT project will demonstrate the "Do It Together" approach through four use cases: Sustainable wood panels manufacturing, 3D printing of wood, 3D printing of recycled plastic, and Smartification. The Intelligent Relational Tagging System for Smartification Based Dashboard dissertation is directly involved with the Smartification use case, where the objective is to study and create smart furniture pieces to improve the user's quality of life. This smart furniture can be achieved by integrating a panoply of devices that can sense and act accordingly.

Before starting the smartification process, the consumer must choose between "smartify" an existing piece of furniture or requesting a new piece of furniture. The whole smartification process comprises several sub-processes that can sometimes run in parallel. This will request many interactions between the customer and the "smartification team". Defining each intervenient is a fundamental step to understanding the work that will be developed in this dissertation.

The **consumer** is the one that has a problem that requires smartification. It involves several steps, making most of the interactions with the web interface. First, it explains the problem, specifies the furniture that requires smartification and the solutions in mind, and is suggested

possible solutions and relevant functionalities from the web interface. User confirmation is always necessary, and it also can reconfigure the solution. After receiving the solution user, through the web interface, can give some feedback about the smartification process.

A **furniture designer** is responsible for designing the furniture if the user does not want to choose every detail of the solution. Depending on the type of customer, this might be the entity that directly interacts with the user interface system.

The **user interface system** is responsible for developing the web interface that will be used by the user and for developing the tagging system to be integrated into the interface to help the user find the right solution. It is directly involved with the user and the system engineer.

The **prototyping system** is responsible for creating a hardware list to be introduced in the KB, which the web interface will use to suggest smartification ideas to the user. It is also responsible for validating or creating the solution and the respective development instructions. Therefore, it is involved more directly with the development and quality engineers.

The **quality assurance system** is responsible for validating the hardware list. It is also this system's job to validate the 3D dimensions of the solution to fit the solution in the desired furniture. In the process, the quality engineer is the most direct connection to the furniture manufacturer and is responsible for reporting the solution and the technical specifications. It is also responsible for reporting the geometrical solution to the system engineer.

The **runtime system** has two primary responsibilities. First, along with the system, the engineer is responsible for creating the technical specifications of the solution. However, the main work of this system is to deal with the runtime process, in other words, process the data acquired, maintain, and update the smart furniture developed.

The **furniture manufacturers** are responsible for reporting the dimensions to the quality assurance system and developing and modifying the solution to meet the needs defined by the customer.

As seen in Figure 3.1, the proposed system presented in this dissertation has five main tasks the system must accomplish.

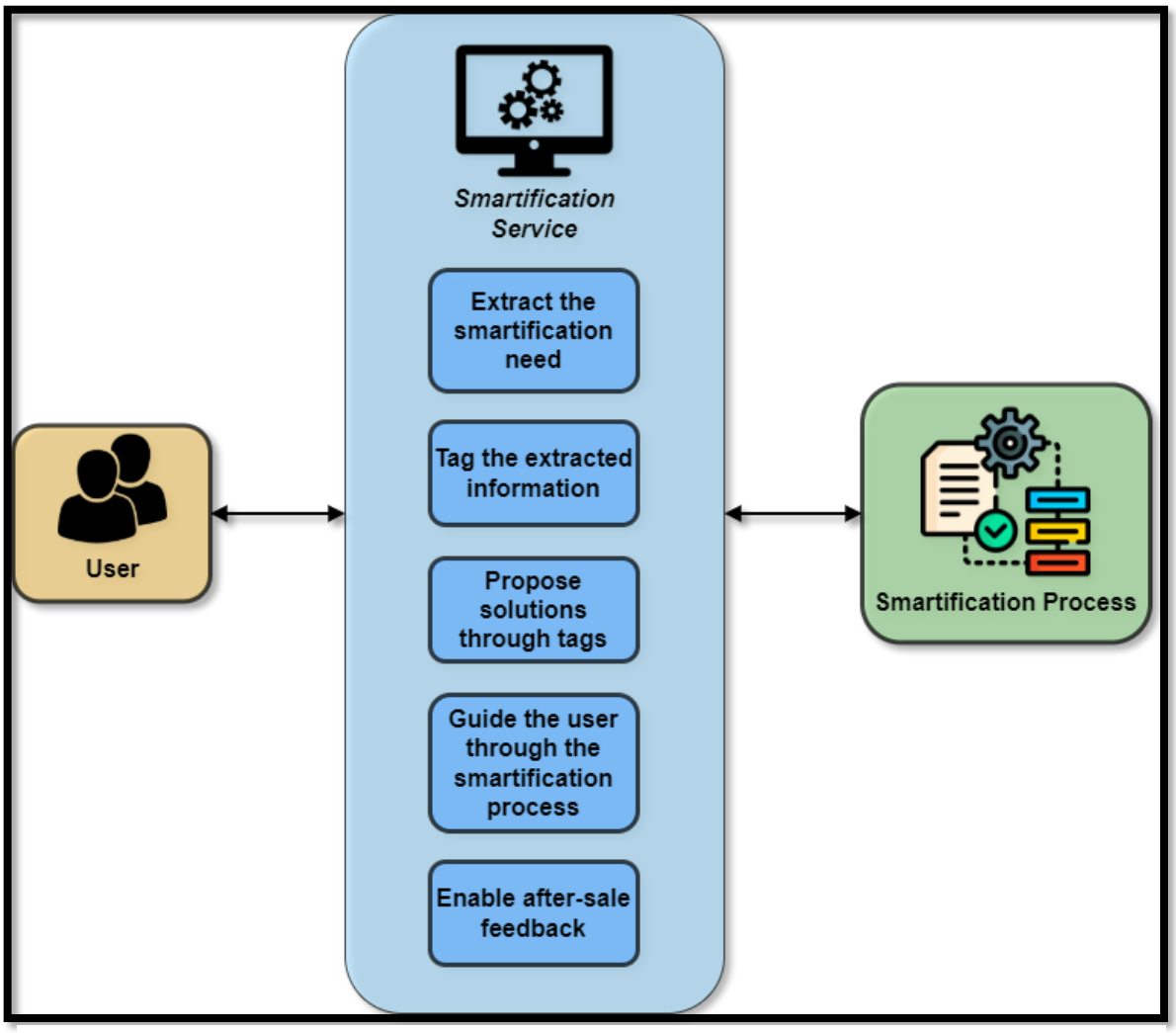


Figure 3.1 - User Interface System Tasks.

As said, the user interface system will interact with the customer and the other intervenients in the smartification process. The main tasks that this system purposes to achieve are:

- The extraction of the customer smartification needs;
- Tag that same extracted information and insert it into the KB;
- Propose solutions and functionalities with the help of tags associated;
- Guide the customer through the smartification process since it is not an easy topic;
- Enable after-sale feedback.

3.2 Main Scenario

The main scenario is where it is represented as an overview of the User Interface System in the smartification project. The interactions between the different intervenients entities are identified, and the most important tasks that each one of them has regarding the User Interface System.

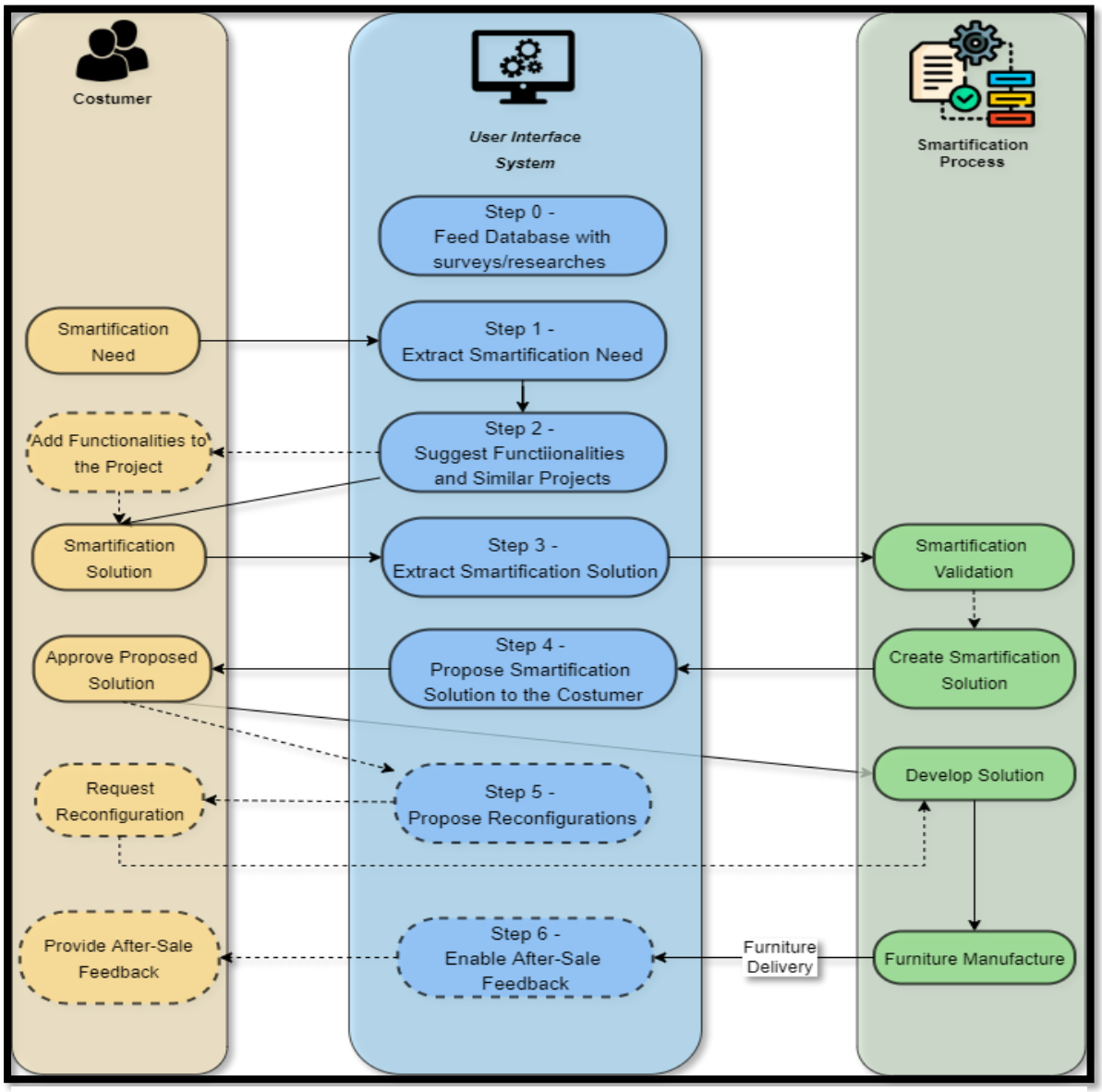


Figure 3.2 - Main Scenario Flow.

In Figure 3.2, it is possible to see all the stages the User Interface System will get through in the blue vertical rectangle. The yellow vertical rectangle shows all the user actions, and the green vertical rectangle shows the actions of all the other intervenients in the smartification

process. Since the smartification project is a continuous process involving many intervenients, some steps depend on others. Not all of the steps are mandatory in every smartification project. The mandatory steps are represented with continuous traces, while the optional ones are represented in dashed lines. The order of steps in a smartification project are:

- Step 0 is a fundamental step for the excellent function of the whole User Interface System. In this step, the database is populated with associated projects and associated tags, allowing the tagging system to make suggestions.
- Step 1 comprises the introduction of the customer's smartification needs in the GUI. After that, the User Interface System will extract the information regarding the customer's need.
- In step 2, the User Interface System will suggest functionalities and similar projects regarding the smartification need extracted in the previous step. Then, the suggestions will appear as a possibility to the customer, allowing the customer to choose the addition of the suggested functionalities to the project.
- During step 3, the customer is invited to describe the smartification solution that the User Interface System will extract and send to the smartification process.
- Step 4 involves the validation of the smartification by the Prototyping System and creating a good solution presented to the customer through the User Interface System.
- Step 5 is an optional stage that is dependent on the approval or reconfiguration request of the solution by the customer.
- Once the customer approves the solution, the manufacturing of the prototype starts in step 6. At the same time, the customer is redirected to an optional survey about the smartification process.

This is an overview of the steps meaning that the complexity of each step is much greater than described above. In order to have a better understanding of this process and the interactions between systems, secondary scenarios are going to be described in the next sub-chapter.

3.3 User Story

As said before, an accessible and easy-to-understand GUI is also a significant target in this thesis. By reaching this target, the process becomes user-friendly. Furthermore, since the system proposed in this thesis is integrated into a more extensive process, it is crucial to consider the user experience by avoiding significant differences when jumping between platforms/services.

Retaking the Bob example.

Before interacting with the service presented in this thesis, Bob will interact with the INEDIT platform. In the INEDIT platform, Bob will create and configure an account, defining the furniture details. Afterward, the smartification project begins, and Bob is redirected to the system this thesis proposes.

Bob will be invited to answer some questions during the first interactions with this system. From Bob's answers, the system retrieved “bed”, "led", "lights", “warm” and “massages” tags. These tags are essential to success because the system uses them to search for other projects with equal/similar tags, which are shown to the user. These tags will also be used to suggest functionalities that Bob can add to his project. From Bob's perspective, he will start by answering a question and be helped from that point forward at every step of the process.

Besides the concerns with the tagging system and with the GUI, it is also necessary to consider the data models and the design of the database. These are also essential for the proper functioning of all the systems combined. Moreover, since the system proposed in this thesis will have multiple interactions, the agreement between all data models and database designs is mandatory for all systems.

4

ARCHITECTURE FOR THE TAGGING PROCESS

This chapter proposes the architecture for the tagging process, explaining the structure and technologies that should be used to implement the hypothesis that this dissertation presents.

4.1 Architecture Overview

In order to develop an intelligent relational tagging system that enriches a knowledge base to guide users to the right smartification solution, this dissertation proposes an architecture to use in the smartification process. This architecture comprises multiple technologies and components that create the Intelligent Relational Tagging System for Smartification Based Dashboard proposed in this dissertation.

The smartification process uses a panoply of available devices to provide awareness of objects and their environment. This process is extendable to several different utilizations, and, in this dissertation, context is applied to furniture. To achieve smartification, the customer must have a smartification need, even if it is not well defined in the customer's mind. In this context, the system suggested by this dissertation can be applied. This system does not aim to solve the furniture smartification process but to help the user through that process by providing functionality suggestions and a similar projects catalog.

This architecture was developed considering the scenario presented in the previous chapter. However, the utilization of this system can have further implementations beyond furniture smartification. The proposed architecture comprises multiple technologies/components: Survey/Research, Knowledge Base, Tagging System, and GUI. The function of each of them will be explained further in the document. However, the GUI is the only one the user will interact with.

Typically, this User will be a customer however the architecture was created so that the smartification expert can also use the system.

Figure 4.1 illustrates an overview of the proposed architecture.

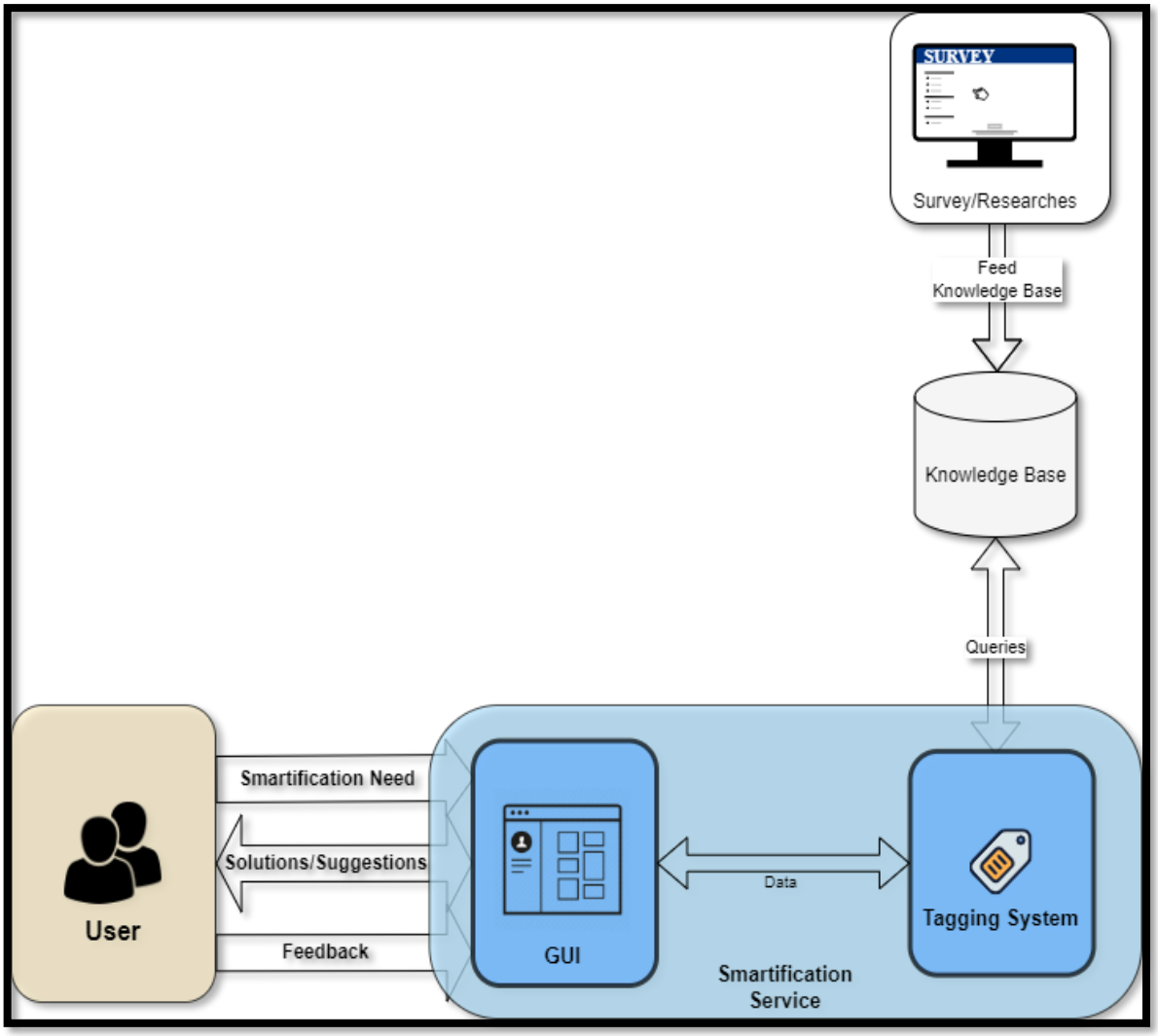


Figure 4.1 - Architecture Overview.

One of the keys to the success of this framework, and the first entry point of data, is the **surveys/researches** that will feed the knowledge base. These surveys will provide an initial dataset in the knowledge base, which projects created by customers will further enrich.

The **knowledge base** contains all the information and data. It has the function of receiving information from the surveys/research and the tagging system, as well as providing data to the GUI and the tagging system.

The **tagging system** is responsible for tagging the information provided by the user through the GUI. This system interacts directly with the GUI and knowledge base and is part of the smartification service. From the GUI, the tagging system receives smartification needs, specifications, possible solutions, and feedback regarding smart furniture and provides solutions, functionalities, and possible reconfigurations to the GUI. The interaction with the knowledge base is to store the tagged information and, through **Queries,** retrieve data to be handled and analyzed so that it can be presented in the GUI.

The **GUI** is what the user can see and interact with. Here, the user will have the possibility to detail the smartification needs and specifications, give feedback, and access the product catalog and suggested functionalities. For the design of this GUI, usability and accessibility standards were considered.

The following subchapter gives a more detailed explanation of the proposed architecture and the interaction between the different components.

4.2 Scenarios Instantiation

Chapter 4.1 presented an overview of this dissertation's architecture for the tagging process, developed by contemplating the scenarios presented in chapter 3. In this section, the architecture followed to implement the solution explained in the chapter 5 will be detailed. This architecture is specifically designed to match the customer needs in the furniture smartification process, however it can be easily shifted to be applied to another end objective. It aims to achieve the main tasks presented in chapter 3. Figure 4.2 shows an overview of the architecture instances organized accordingly to the flow of interactions, which can be divided into smaller steps explored in this section.

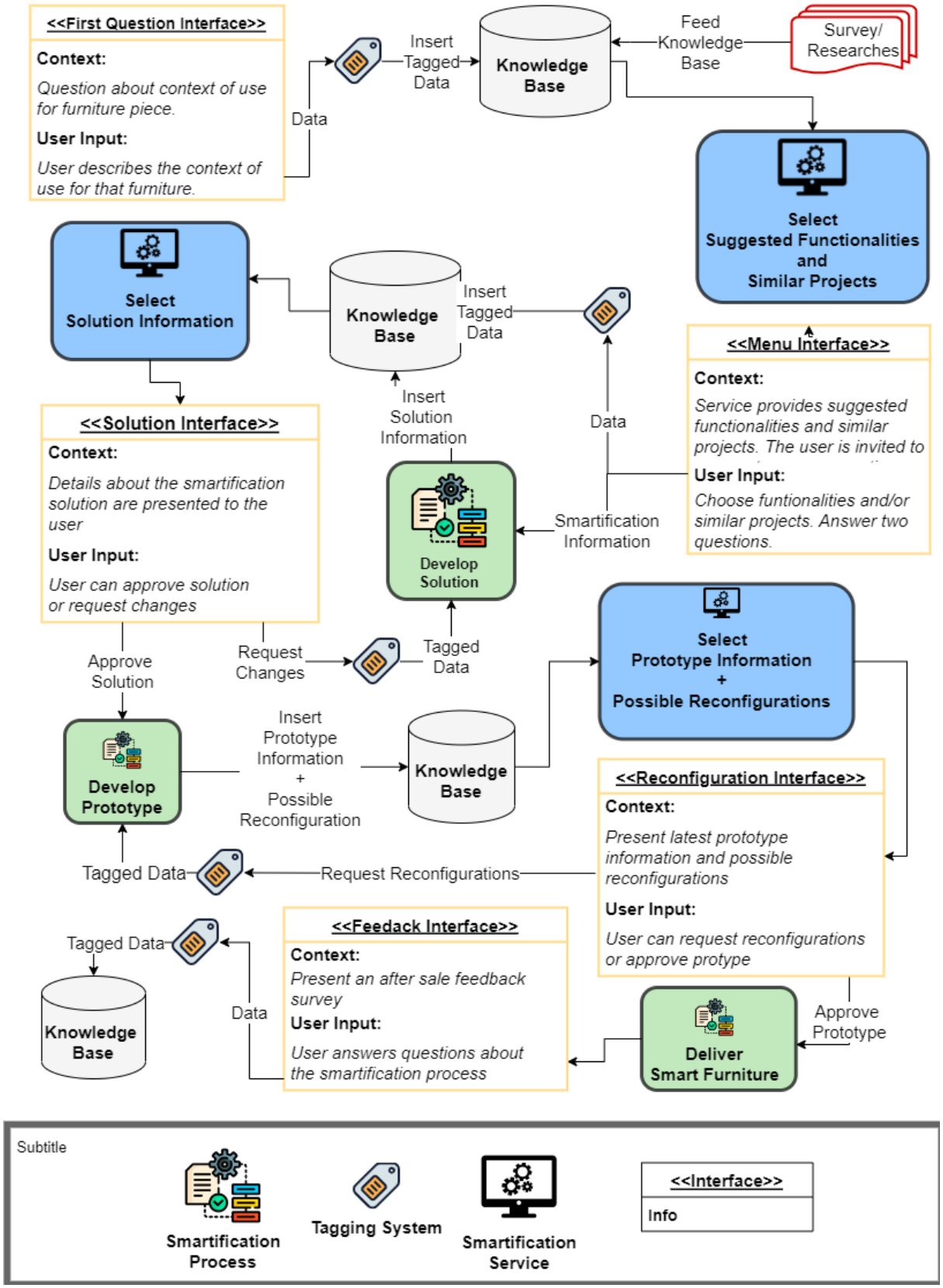


Figure 4.2 - Overview illustration of architecture instances organized accordingly to the flow of interactions.

As explained in the Scenarios sub-chapter, the architecture needs to fulfill some requirements. First, there is the User Interface System, which is the system presented in this dissertation, but there is also a Prototyping System, a Customer, a Quality Assurance System, a Runtime System, and the Furniture Manufacturers. This illustration mainly shows the flow of the User Interface System. Figure 4.2 exhibits the interactions between this system and others, represented by the smartification process icon. The illustration explanation will detail each interaction between the different systems.

From an implementation perspective, the starting point is the feed of the knowledge base with surveys. These surveys are google forms specifically created aiming to extract knowledge from the user. These surveys were developed for a better understating of the questions and examples to be given in the GUI. Creating these surveys was an iterative process, allowing expert and non-expert users to understand and answer the questions, which provides feedback from a larger group of people.

Since the user is redirected from another platform where some of the furniture information is already defined, the first interaction with this system is answering about the context of use for the furniture piece. The tagging system will analyze this description and then insert it into the knowledge base as tagged and non-tagged information.

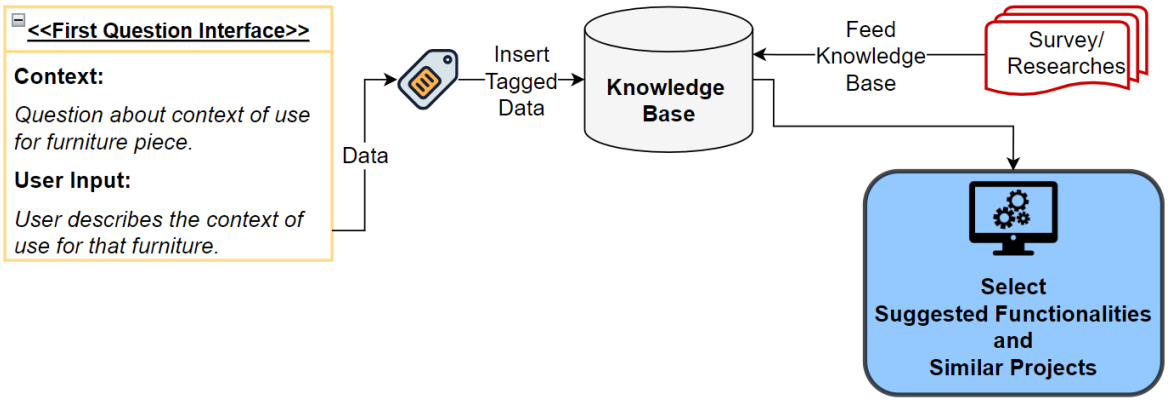


Figure 4.3 - Architecture instance applied. Part 1/5.

From the first interaction, the smartification service can retrieve functionalities and projects related to the current project due to the tags. This can be done by analyzing the tags associated with the current project, which will be used to select similar projects and functionalities. Therefore, in this system's first uses, the surveys previously inserted in the knowledge base have such importance. These retrieved projects will be presented to the user through the GUI. In this second interaction, the user can select a project to add functionalities to the current project and answer two more questions to specify the desired smartification solution better.

From the interface, the data will go through the tagging system to be stored in the knowledge base. Figure 4.4 represents the smartification information sent to the smartification process to develop the solution. However, this information is also stored in the knowledge base, and the systems in charge of the solution development step can retrieve it from there. Once the solution is developed, all the details are inserted into the knowledge base.

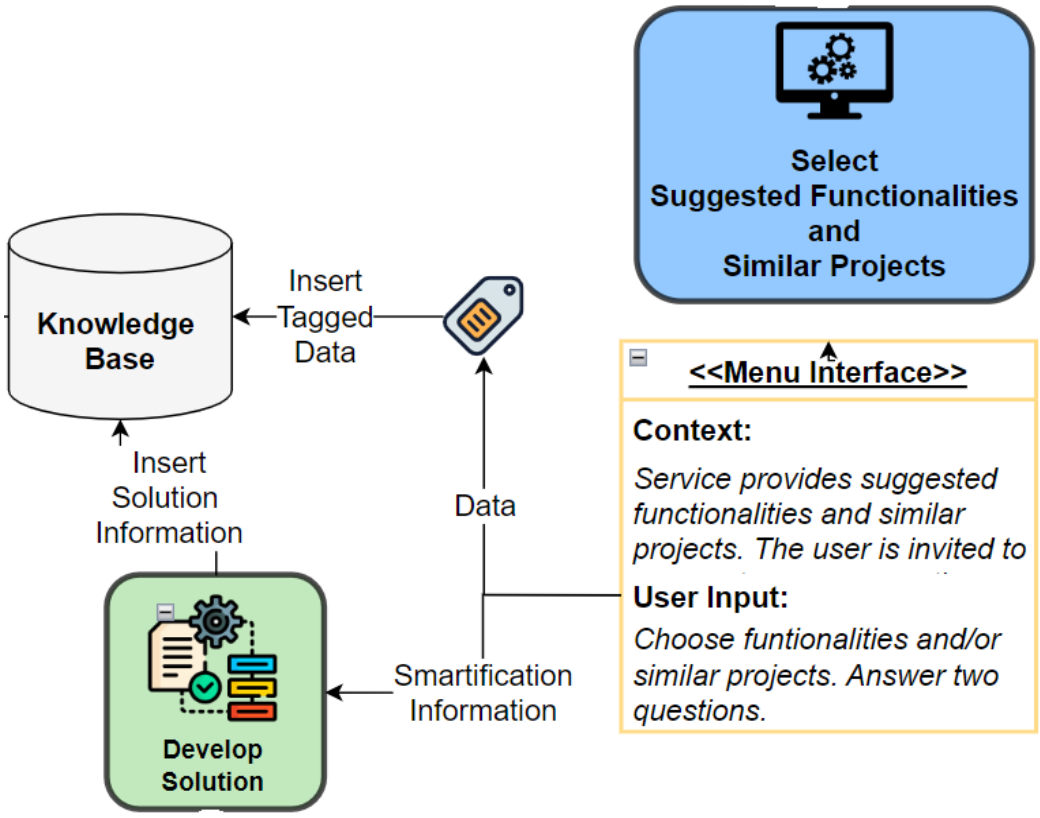


Figure 4.4 - Architecture instance applied. Part 2/5.

After solution development by the Prototyping System and validation by the Quality Assurance System, some details are shown to the user as it can be seen in Figure 4.5. Then, the recurring interaction between the user and the smartification project can start. First, the smartification service selects the suitable solution information available in the knowledge base created in the solution development step. Afterward, this solution information is presented to the user through the GUI. Then, the user can either click to approve the solution or request changes in this interface.

By requesting changes, the user is redirected to another interface where the possibility to write the desired changes is enabled. This information will be processed by the tagging system and inserted into the knowledge base. At this point, the smartification process will develop another solution, and this step will be repeated until the user approves the proposed solution.

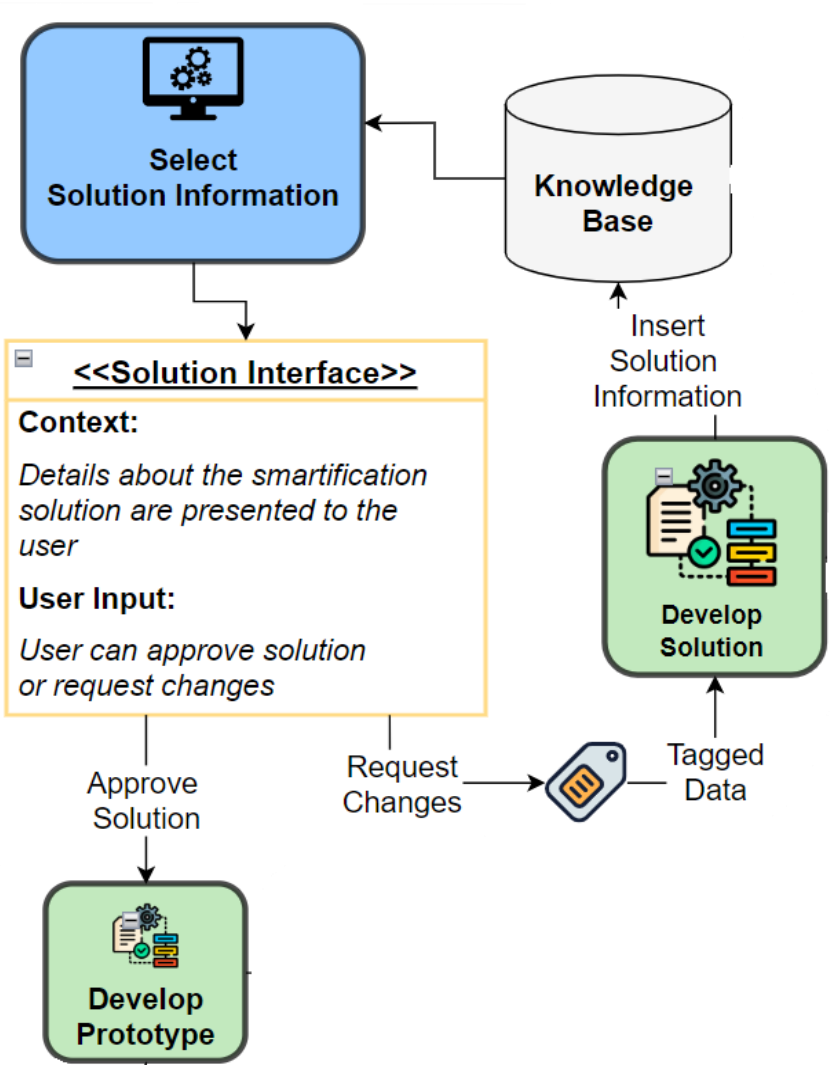


Figure 4.5 - Architecture instance applied. Part 3/5.

This step represented in Figure 4.6 is expected to be iterative since user approval will only happen after meeting all user smartification needs. Once approved, the smartification process continues by the initiation of prototype development.

The prototype development can be long and repetitive since it depends on user approval. Once the smartification process finishes developing the prototype, the related information will be inserted into the knowledge base alongside possible reconfigurations to the developed solution. The proposed Smartification Service will extract the data, which will be presented to the user in the GUI. In this GUI, the user can see all the detailed information alongside a video of the developed prototype. At this point, the user can either accept the prototype or request reconfigurations.

If the user request reconfigurations, the information is tagged by the tagging system and inserted into the knowledge base associated with the current project. This is a helpful feature for future users because these reconfigurations can be presented in future projects. Furthermore, requesting reconfigurations restarts this process because the prototype developed will be adjusted to customer desires.

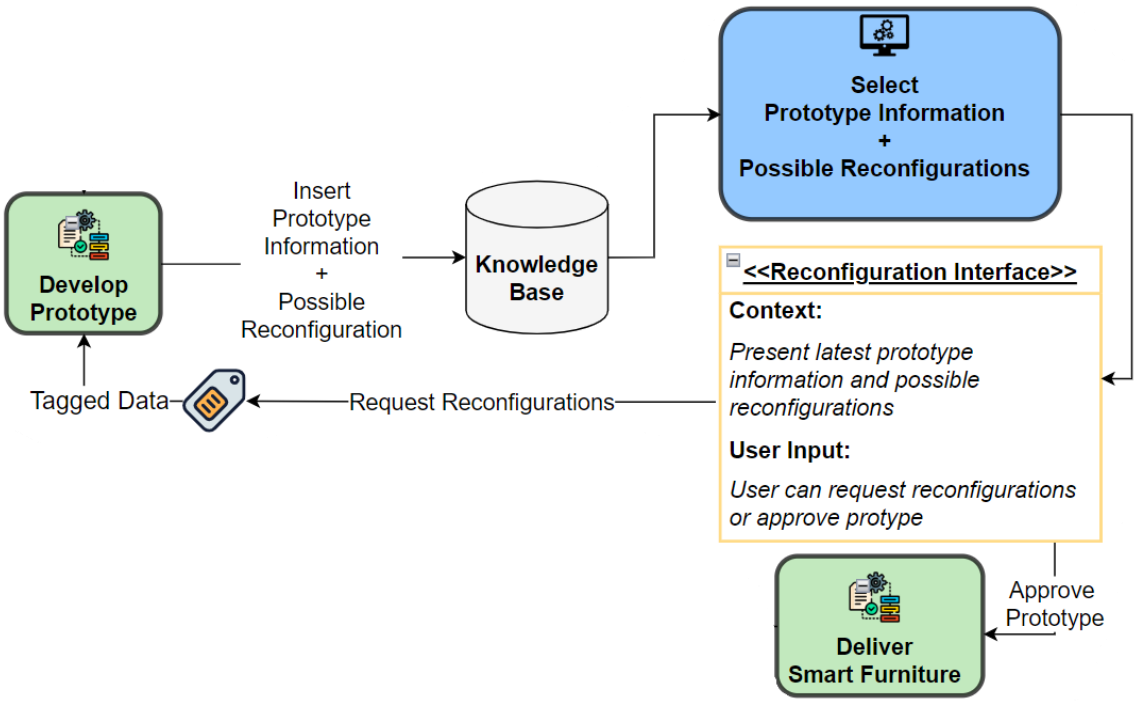


Figure 4.6 - Architecture instance applied. Part 4/5.

However, if the user agrees with the prototype, it will be approved trough the GUI, and the smartification process will begin the Deliver Smart Furniture process.

The delivery of the smart furniture piece closes the smartification process cycle, as it can be seen in Figure 4.7. However, the interaction between the user and the smartification process continues. Once the Deliver Smart Furniture process starts, the user is redirected to a feedback survey. Here the user can give input about the smartification service, and the obtained data will be processed by the tagging system and inserted into the knowledge base.

As it was possible to observe in the figures above, the service that this dissertation proposes is composed of several components, like the database, the tagging system and the GUI. Next, each of them will be reviewed in detail and explain why the specific software was chosen.

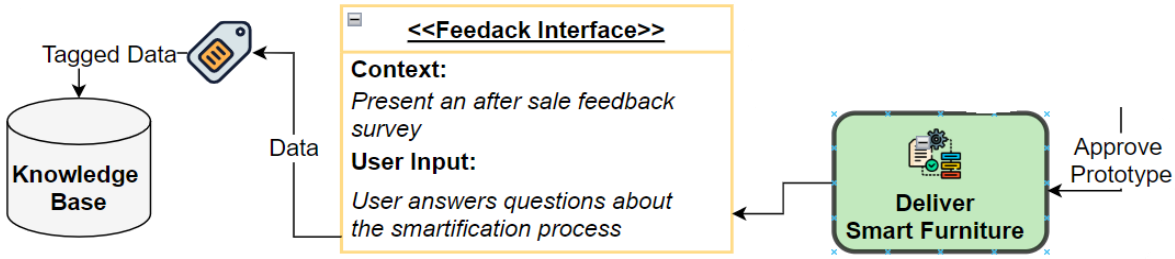


Figure 4.7 - Architecture instance applied. Part 5/5.

During the implementation stage of the project, some changes might happen. However, this architecture will always be true and can be implemented if the objective is to help a customer in a smartification project.

4.2.1 GUI

A GUI, a graphical user interface, is computer software that allows a person to interact with a computer through symbols, images, pointing devices, or visual metaphors [64]. Alongside technology in general, the GUIs also evolved over time.

It started with a few buttons to allow the user to navigate and have minimal interactions and evolved to modern interfaces that can be navigated through voice commands. The GUI is considered a key feature to the success of any software program that requires user interactions.

This thesis is not an exception because most interactions between the customer and the smartification project are through the GUI.

The frontend part of this dissertation proposes using the **flutter framework** [65] for interface development.

Flutter is a free and open-source UI framework created by Google for building applications from a single codebase. One programming language and one code base can be used to create different apps. Originally it was meant for mobile application development, but later on, it was extended to support platforms like the web, Windows, and Linux [65].

The choice of the flutter framework is influenced by the fact that this is faster than many other application development frameworks, especially with the "hot reload" feature, where it is faster to build the project, test, and fix bugs [10]. Furthermore, with the concerns of accessibility and usability for interface development, flutter allows the developer to control every aspect and every pixel on the screen, thus enabling customization in every aspect.

The dart language is another benefit of the flutter framework. Dart programming language, developed by Google, is a language targeting fast-performing apps on multiple platforms. Even though flutter is a current framework, there is already some excellent documentation with easy examples for different use cases and a growing and robust community. The cherry on top of the cake is that this is an open and free source [66]. This framework is already very concerned and well prepared for the accessibility principles, which is a bonus to this thesis.

Chapter 5 will explain all created interfaces and the reasons behind their nature. In order to store and retrieve information, it is necessary to have a database.

4.2.2 Database

A database is an organized collection of structured information, or data, typically stored electronically in a computer system [67]. The information is stored in an organized way through tables, rows, columns, and indexes. Due to this type of organization, the data is easy to access, manage, modify, and update.

Almost every database uses a structured query language (SQL) to write and query data. SQL is considered a programming language used by almost all relational databases to manipulate data. Via this programming language, it is possible to query, update and reorganize data, as well as to create and change the structure of a database system.

Typically, a database is controlled by a database management system (DBMS) that serves as an intermediary interface between the user, or other programs, and the database itself. In this interface, it is possible to use SQL to manage the data in the database.

Considering the wide range of existing database types, this thesis requires utilizing a relational database. This type can be defined as a database that stores and provides access to data points related to one another. It is a straightforward way of representing information in a table, in which each row is a record with a unique ID and each column holds the data's attribute [68]. The benefits of having relational databases are the possibility to manage data in a secure and rule-based way, presenting a consistent method of information that specific points relate to other information [68].

In this thesis, the PostgreSQL[69] was adopted due to all the inherently attached benefits. This is an open-source relational database that supports both relational and non-relational querying. Furthermore, it is a reliable, stable, and resilient database management system, offering more than two decades of community development, and is used from web applications to analytic applications [70].

PostgreSQL is a project created by Michael Stonebreaker in 1986, aiming to add to the previous project, POSTGRES, the features needed to support various data types in its entirety. This evolution brought several benefits, such as the rich features and extensions supported, thus being highly scalable in the quantity of data and the number of concurrent users it can accommodate. PostgreSQL also has suitable reliability, following good database management practices. Moreover, the fact that PostgreSQL has an open-source license presents one of the most important benefits. Having the source code available under an open source license makes it possible to use at no charge, which is a great asset when associated with high scalability [70].

4.2.3 Tagging System

The definition of a tagging system depends on understanding the tag concept. As explained in Tagging sub-chapter, a tag is a user-generated keyword that has become a way to improve descriptions of online information resources [24]. In this context, these tags must be generated from the customer's answers and associated with the project. Once this association is done, each project will contain valuable information that can be used to suggest smartification functionalities and to give a catalog of similar projects to the customer. These suggestions will simplify the smartification process by providing meaningful help to the customer.

The relationship between the tagging system and the previously defined taxonomy and folksonomy concepts becomes evident, with the system leaning more towards the folksonomy approach. This system enables collaborative tagging in a different approach to the common folksonomy. This is done by allowing the users to remove tags that they might consider useless to the project but blocking them from adding tags. The tags are created/generated and associated with the project by the tagging system, allowing users to check them before the final association. Even though this system leans more towards folksonomy, taxonomy is also present by categorizing the projects initially by type of furniture and then by smartification functionalities. Therefore, it becomes easier to use finished projects to make functionalities suggestions and to populate the suggested project's catalog.

The proposed tagging system will run on an API used by the frontend part of the project to generate the tags and get the similarity between tags. This system will be a web application that the GUI will consume. The system will be built as a web application using Flask [71], a python module that facilitates the development of web applications [71].

This tagging system will have a main file that handles the requests done by the Interface/Frontend part of the project. This main file then calls the appropriate functions accordingly to the requests done. This is the **tagging system flow**:

- The Information sent by the Interface System is the input parameter of **TagsFunction** function.

- **TagsFunction** handles the input and returns a response.

- This response is then the input parameter of **checkTags** function.

- **checkTags** function iterates through the input parameter vector and acts accordingly by calling or not calling the **similarity** function.

- If **similarity** function is called, the return of this function is then used to the return of **checkTags** function.

What each function does and returns will be explained in more detail in the following subsections. Following the flow of tagging system, the first function explained is **TagsFunction**.

4.2.3.1 TagsFunction

TagsFunction is a function that receives sentences as an input parameter and returns a list of the tags generated in those sentences. This starts by tokenizing the received sentence by using NLTK's recommended tokenizer. Then the generated tokens are iterated. Each token will go through an NLTK POS algorithm that returns the part of speech of each word in the given sentence. From the study, the most valuable information in the customers' sentences for a smartification process are nouns, adjectives, verbs, and numerals. Therefore, every word that matches one of these POS fields selected is added to the response vector of the function. Before sending the response, this list is iterated to check if each entry is one of the most common words that have no value to the tagging system. Therefore, this function returns in a list all the words from a given sentence that are part of the grammatical field and have valuable information to the project.

As an example, let us use the sentences: "*I want my bed to have a charger that can charge two devices at the same time. I want to lay down my phone on the bed and it starts to charge but also to have a cable to charge overnight.*" as the input of the **TagsFunction**.

```
Words:
['I', 'want', 'bed', 'charger', 'charge', 'two', 'devices', 'time', '.']
POS:
['PRON', 'VERB', 'ADJ', 'ADJ', 'NOUN', 'NUM', 'NOUN', 'NOUN', '.']
Added:bed
Added:charger
Added:charge
Added:two
Added:devices
Added:time
Words:
['I', 'want', 'lay', 'phone', 'bed', 'starts', 'charge', 'also', 'cable', 'charge', 'overnight', '.']
POS:
['PRON', 'VERB', 'VERB', 'NOUN', 'NOUN', 'VERB', 'NOUN', 'ADV', 'NOUN', 'NOUN', 'NOUN', '.']
Added:lay
Added:phone
Added:starts
Added:cable
Added:overnight
Sending:
['bed', 'charger', 'charge', 'two', 'devices', 'time', 'lay', 'phone', 'starts', 'cable', 'overnight']
```

Figure 4.8 - TagsFunction working overview.

Figure 4.8 details the function working. Firstly, it recognizes that the input has two sentences because of the dot and separates them into two sentences that the algorithm has to go

through. Then, it starts by making a clean-up in the sentence by only saving words that fit in POS.

Sentence 1:

Words:

['I', 'want', 'bed', 'charger', 'charge', 'two', 'devices', 'time', '.']

POS:

['PRON', 'VERB', 'NOUN', 'VERB', 'NOUN', 'NUM', 'NOUN', 'NOUN', '.']

Sentence 2:

Words:

['I', 'want', 'lay', 'phone', 'bed', 'starts', 'charge', 'also', 'cable', 'charge', 'overnight', '.']

POS:

['PRON', 'VERB', 'VERB', 'NOUN', 'NOUN', 'VERB', 'NOUN', 'ADV', 'NOUN', 'NOUN', 'NOUN', '.']

Then the algorithm goes through this first list of words and adds to the response list of the function the ones that match the POS identified as valuable to the smartification process. Before returning the response, the algorithm goes through this response list and removes the words that match the POS selected but are considered useless to be added as project tags. This is done by having a pre-defined list of useless and meaningless words that the algorithm will go through and compare to the words in the response list. The algorithm is also prepared to deal with equal tags in different sentences by only allowing unique words in the response list.

Response vector:

['bed', 'charger', 'charge', 'two', 'devices', 'time', 'lay', 'phone', 'starts', 'cable', 'overnight']

Still following the **tagging system flow**, this response will be the input parameter of the **checkTags** function. However, since the **checkTags** function calls the **similarity** function, it is easier to understand first the **similarity** function and then the **checkTags** function.

4.2.3.2 Similarity

This function has one word as the input parameter and returns a list with only one word. Typically, the input parameter will be the iteratively each of the tags present in the input parameter list of the **checkTags** function, previously generated from the **TagsFunction**. The idea here is to select all the tags already in the database and compare them with the input word. The one with the highest similarity is the one that is returned by the function.

The function starts by retrieving from the database all the tags generated by the other projects and putting them separately in a list. Then it goes through this list of tags and compares the similarity with the function's input parameter. This similarity is created by encoding through a SenteceTransformer model the tags that will be compared and using the cosine function to return a level of similarity. The function then compares each tag's similarity level in the database with the input word and finds the tag more similar to the input word. However, this similarity must be higher than 85% to retrieve the tag from the database. If it is lower than 85%, which means that the similarity between the input word and all of the generated tags until now in the database is not enough, this new tag must be inserted into the database. After several attempts to check the algorithm's working, the 85% threshold was selected.

This allows the creation of a word cloud, where a tag can be associated with multiple words. This reduces the number of tags in the database but increases the accuracy of suggested projects and suggested functionalities. Different expressions and explanations of the same action or thing will lead to the same suggested functionalities and projects.

A database simulation was created to check the similarity function in action to understand the similarity function better. For the example, consider the following list as the database the function will look at.

Database Example:

- bed

- couch

- chair

- door

- cabinet

- desk

- closet

- recliner

```
['bed', 'couch', 'chair', 'door', 'cabinet', 'desk', 'closet', 'recliner']
Input: bedroom
Similiraty level:
0.8946835
Best Match:bed
```

Figure 4.9 - Similarity function returning a tag from the database.

Figure 4.9 presents the result of the function working alongside the dataset used. For example, the function's input parameter is the word “bedroom”. The algorithm compares that word with all the tags in the example database and returns the one with the higher similarity level, in this case, "bed". However, sometimes the database will not contain any tag that matches the desired similarity level regarding the input word. In these cases, the function's return will be the same input word, as it is possible to see in Figure 4.10.

```
['bed', 'couch', 'chair', 'door', 'cabinet', 'desk', 'closet', 'recliner']
Input: commands
Similiraty level:
0.69237083
Best Match:commands
```

Figure 4.10 - Similarity function returning the inpt word as tag.

When this happens, a new tag is inserted into the database as a new entry. This new entry will increase the types of associations that can be done in future cases, creating a new cloud word associated with "commands" in this particular case. As expected, this algorithm does not work correctly 100% of the time, causing some associations that are not correct.

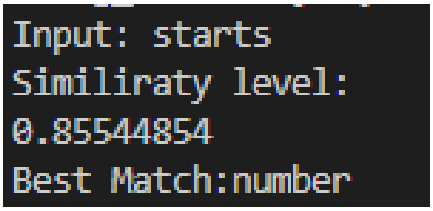


Figure 4.11 - Similarity function creating wrong associations.

Figure 4.11 is an example retrieved from the tagging system already running with real smartification projects. In this case, the database used was from the production environment.

As it is possible to see, with the input "starts" the similarity level reached with the tag "number" is almost 86%. Even though the similarity level was above the defined threshold, the association does not make sense. Of course, this can be solved by setting a higher level of similarity to match input words with tags from the database. However, increasing the threshold would result in the loss of valuable matches. The 85% was not chosen randomly, but it came from an iterative process of checking matches between input words and tags from the database. This process started with 70% as the accepted similarity level, and from errors like Figure 4.11, this level was redefined to 85%, which was found to be the perfect combination. At this level, the similarity errors are shallow, and the value matches are very high, which was not occurring with similarities above 85%.

As explained, this function does not have to be executed with every tag generated from the **TagsFunction** because the system is prepared to save relationships between tags and words: saving execution time and machine resources. The function checks if this **similarity** function needs to be executed in the **checkTags** function.

4.2.3.3 checkTags

The **checktags** function receives as an input parameter a list, the result of the **TagsFunction**, and returns a list of tags. The purpose of this function is to choose if it is needed to call the **similarity** function or not. This is done by going through the input list and comparing those words/tags with those already stored in the database.

Going iteratively through the input list, the flow of this function is:

- Check if the word retrieved from the input list is already present in the database with a tag related.

  - If yes, the tag related is retrieved and added to the output list of tags.

  - If not, the **similarity** function is called.

The tag and the similarity level come from the result of the **similarity** function. If this similarity level is higher than 85%, the relation between the input word and the tag from the database is saved so that it can be used next time.

Saving these relations between tags and words saves a lot of function execution time and computer resources. In the first instances of this tagging system, these relations were not saved, and with this improvement, the application's execution time was reduced to almost half of the time.

The presented functions will work alongside the frontend part of the project to suggest functionalities and projects to the user. The presented functions will work as a web application that the frontend with HTTP [72] requests will consume. HTTP or Hypertext Transfer Protocol (HTTP) is the foundation of what the Internet is. It is an application layer protocol that was conceived to transfer data between devices connected to a network. An HTTP request is what is done between applications to ask for information to load interfaces [71].

JSON [66], which stands for JavaScript Object Notation, was the chosen data model format adopted to interchange data between platforms. It is a lightweight data-interchange format usually used to send data between computers and is language-independent. This format is syntactically similar to the code for creating JavaScript objects, which makes it easy to convert JSON data into JavaScript objects and vice-versa [66].

Via FLASK, a micro framework that uses Python to build Web applications, the backend/tagging system is always running, which can be accessed by the frontend via predefined URL addresses. This way, it is easier to debug in the creation process of this system, but it also enables other projects to use this system.

5

IMPLEMENTATION

The architecture presented in the previous section serves as a model for the application's implementation. This application will guide the customer through the smartification process, as well as aid in extracting the customer's smartification need. The application comprises three backend components: the Knowledge base, the RESTful API [73], the Python application, and one frontend component, the GUI. In this chapter, all of these components will be explained in detail, as well as the connections between them.

5.1 Application Structure

The application contains four major components, the database, the RESTful API, the tagging system, and the GUI. Each has different objectives, although they all work together to bring this application alive.

The database is running in the *PostgreSQL* cloud service provided by *ElephantSQL*. It is used to store all the information about the smartification projects, the different states of the project, and all the information that changes along the smartification process. *ElephantSQL* was the chosen cloud service mainly because it has free storage until 20MB, which is enough for testing several smartification projects. It also offers good quality servers, such as Amazon Web Services [69]. As mentioned before, the data management selected was *PostgreSQL*, which supports *ElephantSQL*, and JSON, being free and open-source. In order to retrieve and change the data in the database, it is necessary to use an intermediary service between the frontend, backend, and the database. Glitch was the tool chosen to create a free web app that can run the RESTful API with Node.js, which runs as the intermediary service. This service is explained in detail in the RESTful API sub-section. The tagging system, or backend, was made with Python using Flask framework to create a Web application that the interface can consume. The interface, or frontend, was developed with the Flutter framework, which uses the Dart programming language. It is possible to see the relationship between all the components of this application in Figure 5.1.

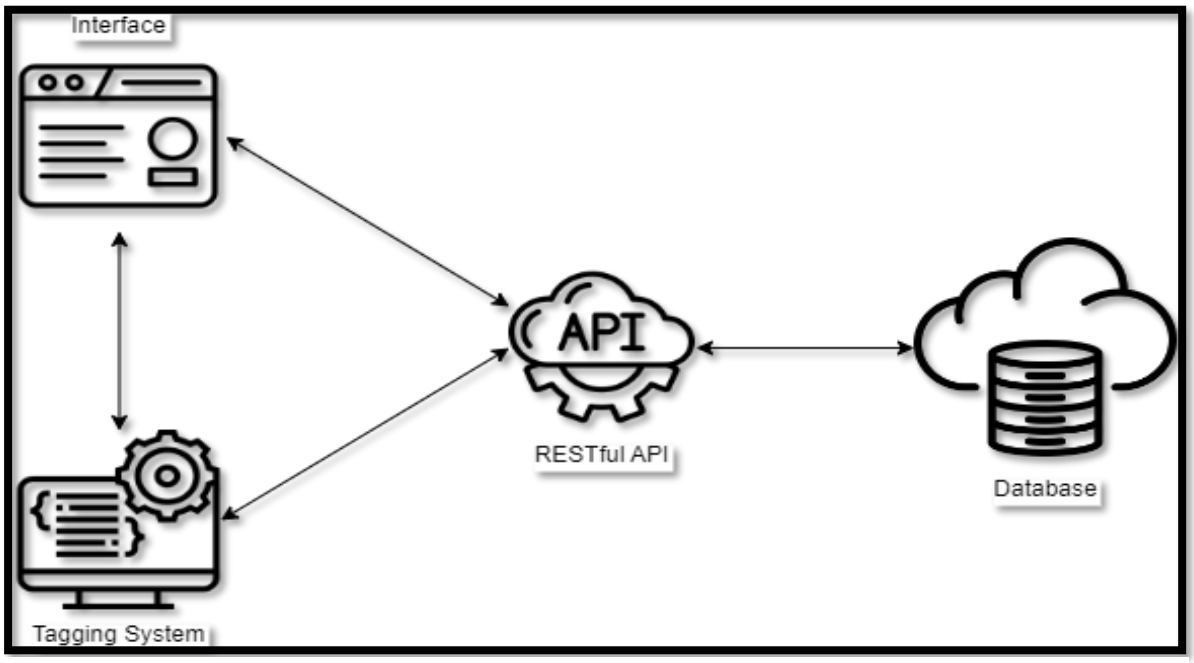


Figure 5.1 - Application Structure.

As it is possible to observe, to have any type of interaction with the database, it is necessary to go through the RESTful API by an HTTP request. Then the API deals with the request and communicates with the database via SQL commands. Finally, information returned from the database goes by the API, redirecting the answer for the requested component. Besides these interactions, the interface and the tagging system can also interact via HTTP requests.

5.2 Database and State Diagrams

For developing a software application project, one of the most critical structures to define is the data. All the data needed to develop a smartification project is stored in the database, which follows a structure shared by the different systems involved in the smartification development process. It happens so that every system knows the purpose of each field in the database and that every system can have an overview of the process of a specific smartification project.

5.2.1 Project State

As mentioned in the Database sub-chapter, the table Project has a field *project_state* representing the project's current state. The following diagram represents the different states of the project and their meanings.

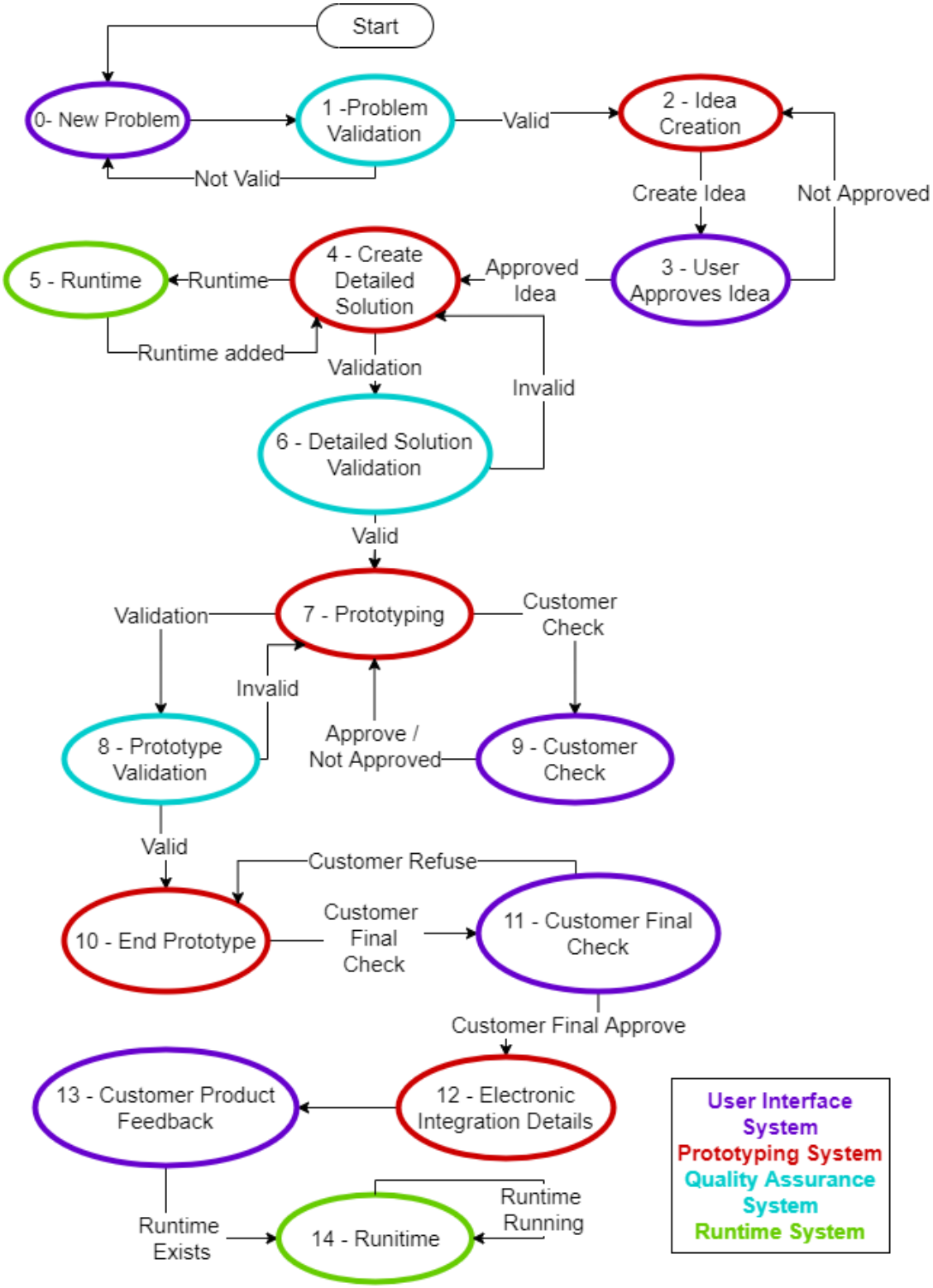


Figure 5.2 - Values of *project_state* field.

This is a significant field because, as mentioned, each smartification project has the intervention of several independent systems. With this value, every system can keep track of the evolution of the project.

The "start" state sets the beginning of this diagram. However, since this diagram represents the field *project_state* from the database, the first effective state of this field is New Problem (0). This state represents the moment of furniture smartification need and context description by the customer through the User Interface System. Afterward, the Quality Assurance System will validate the smartification need and the context of the furniture, reaching the Problem Validation state (1). If everything is valid, the project state goes to Idea Creation (2). If invalid, returns to state 0. This can be easily seen in Figure 5.3.

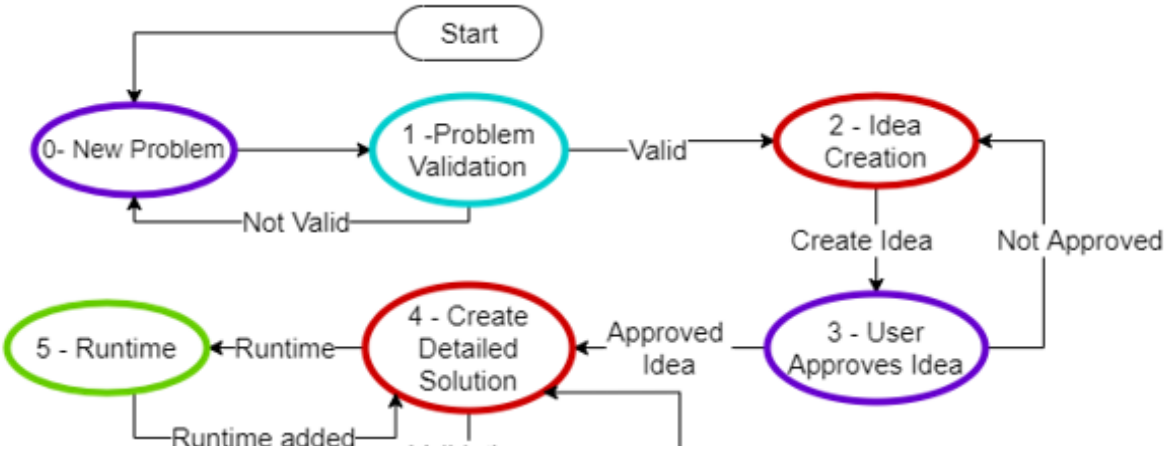


Figure 5.3 - Values of *project_state* (0-5).

In state 2, the Prototyping System will create a new idea to propose to the customer based on the information provided in state 0. Once the idea is created, the *project_state* goes to the User Approves Idea (3) state, where the customer will review the suggested idea via the User Interface System. If the customer approves the idea, the state goes to Created Detailed Solution (4). However, if not approved, the state returns to 2, and due to the User Interface System, the customer can also provide reasons for reproval.

In state 4, the Prototyping System creates a detailed solution to propose to the customer. Then, if needed, the project state goes to Runtime (5). In this state, the Runtime System can edit the detailed solution to add the runtime specifications leading to state 4. If it is not needed to go to state 5, the state will be Detailed Solution Validations (6), which can be seen in Figure 5.4.

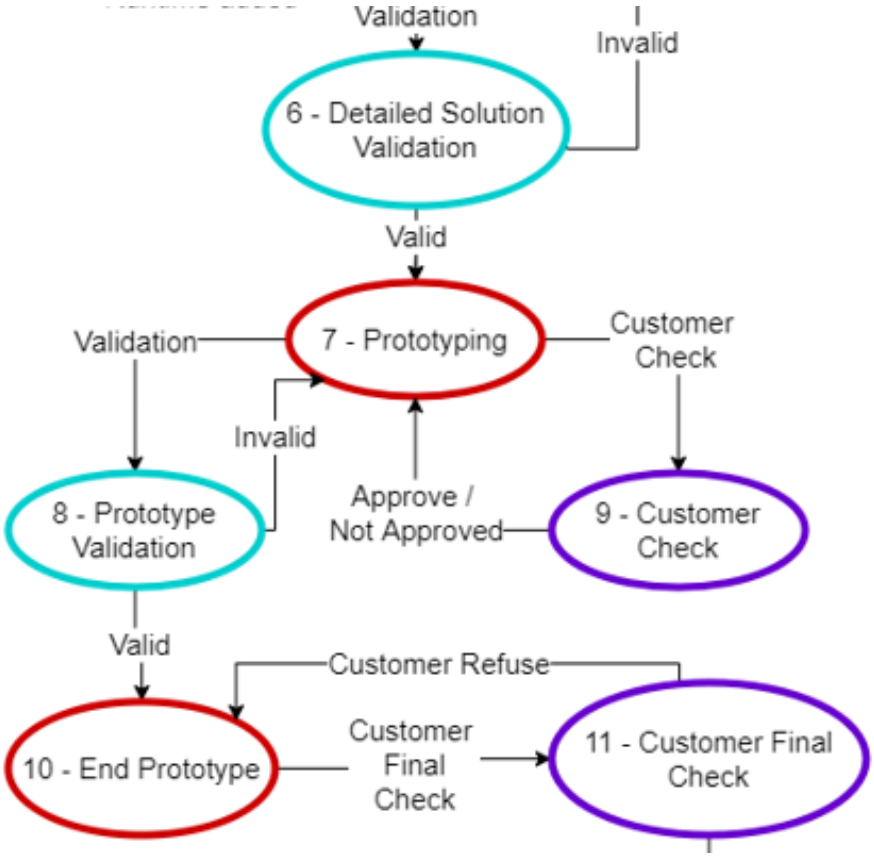


Figure 5.4 - Values of *project_state* (6-11).

In state 6, the Quality Assurance System will validate the proposed detailed solution, leading the *project_state* to different values. If valid, it will go to state Prototyping (7). On the other hand, if something is invalid in the detailed solution, it will return to state 4.

In state 7, the Prototyping System will start creating the prototype, and once it finishes, the state will go to Customer Check (9). Here, the User Interface System will show the customer a video of the prototype and a brief description. In this state, the customer can either not approve and provide feedback or approve, leading to state 7 again. This loop can happen several times until the customer and the Prototyping System find an agreement about the prototype. Once this happens, the state goes to Prototype Validation (8), where the Quality Assurance System validates the prototype. The outcome of this validation can lead to End Prototype (10) or, once again, back to state 6.

State 10 is where the Prototyping System finishes the development of the prototype and adds possible reconfigurations to the solution, leading the state to Customer Final Check (11).

In state 11, the User Interface System shows the customer the final details of the prototype and enables the customer to see the associated reconfigurations. In this state, there are two outcomes. First, if the customer refuses the prototype, the state returns to 10. Customers approving the prototype or requesting one of the reconfigurations lead the state to Electronic Integration Details (12), which can be seen in more detail in Figure 5.5.

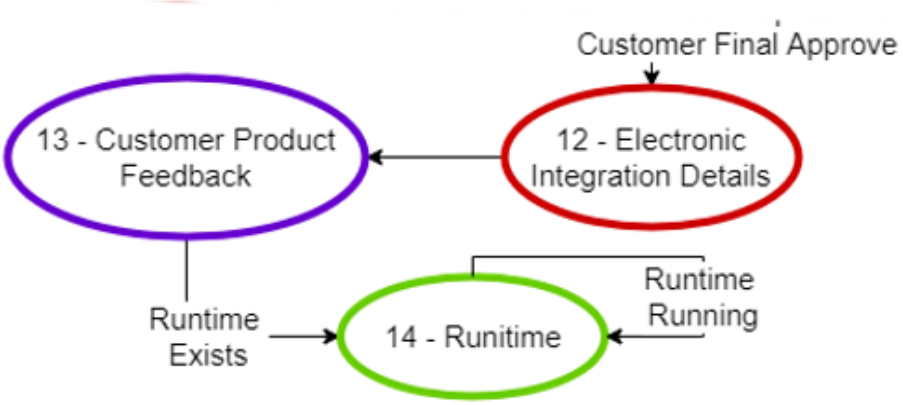


Figure 5.5 - Values of *project_state* (12-14).

In state 12, the Prototyping System creates a document to guide the electronic integration done in the prototype. When finished, the state goes to Customer Product Feedback (13). At this state, the user is enabled by the User Interface System to give feedback about the smartification process.

To finish the diagram, if runtime exists on the smartification piece that the customer is using, the state will change from 13 to Runtime (14), which will be the end state of the *project_state* field. However, if runtime does not exist, state 13 is the final state.

5.2.2 Database

The database is where all the smartification projects and all the project-related data are stored. The design in this database enables the customer to have more than one project associated and allows multiple projects running simultaneously. Another concern considered in the database's design phase is allowing all the systems involved in the smartification process to know the stage of each smartification project. The database is divided into ten main tables representing the users, project, ideas, tagging, furniture specifications, detailed solutions, ideas, prototypes, reconfigurations, and hardware list. Three additional tables represent the connection between some of the main tables. It is possible to see the entire database in Figure 5.6.

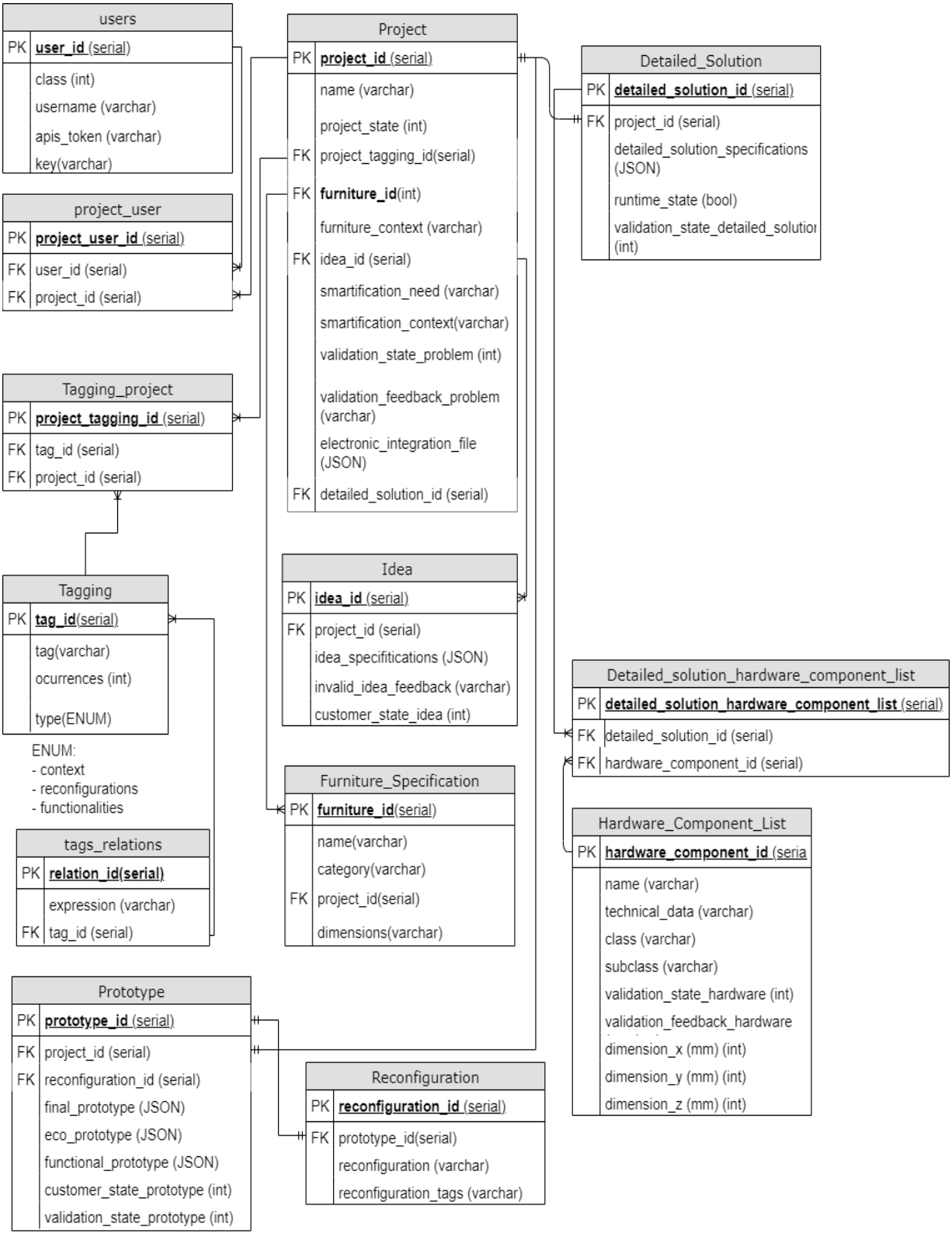


Figure 5.6 - Database.

The **user** table is on the left upper side of the image, representing the user in the smartification process, who will have a *user_id*, *class*, *username*, *apis_token*, and *key*. These last two are only for authentication purposes to connect to other systems. The *user_id* defines the unique ID that each user has. The *username* is self-explanatory, representing the name that the user has associated. Finally, the *class* field identifies the type of user. For example, class 1 represents Quality Assurance System, class 2 Prototyping and Runtime Systems, and class 3 Customers who interact through the User Interface System.

The design allows one user to have multiple smartification projects, and that is what represents the **Project_user** table. It contains the *user_id* that comes from the **User** table, the *project_id* that comes from the **Project** table, and the *project_user_id*.

Project	
PK	**project_id** (serial)
	name (varchar)
	project_state (int)
FK	project_tagging_id(serial)
FK	**furniture_id**(int)
	furniture_context (varchar)
FK	idea_id (serial)
	smartification_need (varchar)
	smartification_context(varchar)
	validation_state_problem (int)
	validation_feedback_problem (varchar)
	electronic_integration_file (JSON)
FK	detailed_solution_id (serial)

Figure 5.7 - Project table.

The **Project** table is the main table in the smartification project since it is associated with almost all of the other tables and can be seen in more detail in Figure 5.7. The *project_id* is a primary

key that identifies each project. In the **Project** table, the customer will directly impact *name* (changeable at the beginning of the process), *furniture_context*, *smartification_need,* and *smartification_context*, by answering the questions in the interface. The *project_state* and *validation_state_problem* are both fields used to control the project's progress. The *validation_state_problem* field can have three different values, 0 if the project is pending validation, 1 if it is validated, and 2 if it is invalid. The Quality Assurance System gives this value. The *project_state* values are going to be explained in detail after presenting this overview of the database.

As shown in Figure 5.6, the **Project** table is related to multiple tables.

The **Idea** table is where the first idea will be drawn to present to the user. Each idea has an *idea_id* associated with a *project_id* and contains a JSON file (*idea_specifications)* detailed in the Idea Specifications sub-chapter. Additionally, each idea has an *invalid_idea_feedback,* and a *customer_state_idea*, to control the idea status. *Customer_state_idea* can have values from 1 to 6: 1- Idea suggested by the Prototyping System; 2- New idea requested by the customer; 3- Idea invalidated by the prototyping system; 4- Idea validated by the prototyping system accordingly to idea specifications; 5- Idea validated by the customer; 6- Idea invalidated by the customer.

The **Detailed_Solution** table is also associated with the **Project** table. This table gives the user a more specific view of the solution before it goes to prototyping. Together with the tables **Detailed_solution_hardware_component_list** and **Hardware_Component_List**, it is here that the pieces of hardware will start to be integrated into the furniture. Regarding this dissertation, the Customer will only impact some fields of the *detailed_solution_specifications* JSON file, which will be explained in more detail in the Detailed Solution Specifications sub-chapter.

Following the smartification process detailed in the Architecture Overview sub-chapter, the **Prototype** table is where all the information about the prototype being developed is saved. This table has the *prototype_id* as the Primary Key and retrieves *project_id* and *reconfiguration_id* from **Project** and **Reconfiguration** tables, respectively. Both *customer_state_prototype* and *validation_state_prototoye* control the progress of the project. *Customer_sate_prototype* can go from 0 to 6: 0- the prototype needs customer feedback; 1- the customer approves the prototype; 2- the customer reproves the prototype; 3- prototype in final stage pending final feedback; 4- customer accepts the final stage of the prototype; 5- customer rejects the final stage of the prototype; 6- customer requests reconfiguration.

The *validation_state_prototoye* depends on the Quality Assurance System and does not impact this dissertation. Connected to the **Prototype** table is the **Reconfiguration** table, which enables the interface to show reconfigurations to the customer. The *reconfiguration* field is where the reconfiguration is explained and has the *prototype_id* as the foreign key and *reconfiguration_id* as the primary key.

The **Furniture_Specification** table is where the information about the furniture is stored. This table has *furniture_id* as the Primary Key, which identifies each piece of furniture, and the Foreign Key is *project_id*, which comes from the **Project** table. In addition, this table has the *name* field, which is the name attributed to the furniture; the *category*, the field that defines if it is a bed, sofa, chair, etc.; and the *dimensions* field, which gives information about the height, width, and depth of the furniture.

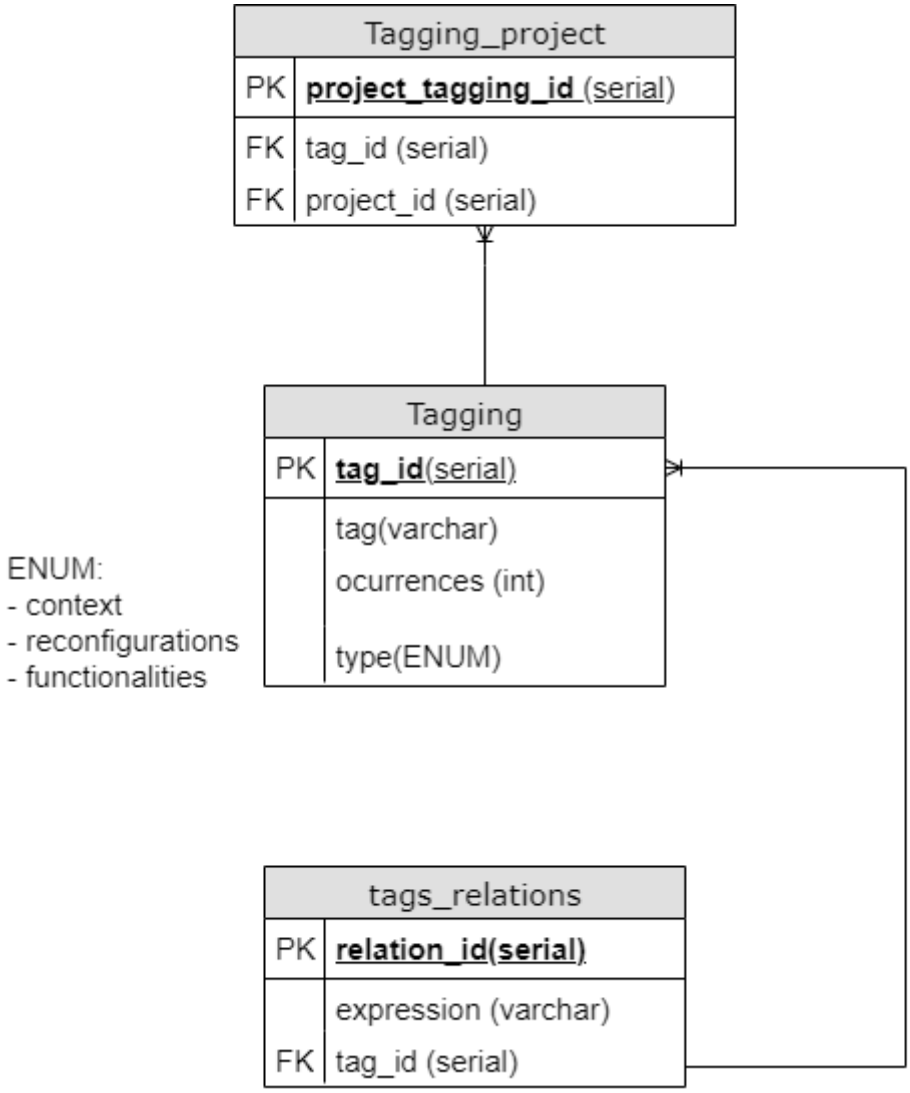


Figure 5.8 - Tagging_project, tagging, and tags_relations tables.

The last three tables, **Tagging_project, Tagging,** and **tags_relations,** are considered the most important to this dissertation since they enable the ability to suggest projects and functionalities to the customer. **Tagging** table contains the *tag_id*, an id unique for each tag(Primary Key); the *tag*; the *occurrences* that count how many times each tag has occurred; and the *type* of tag, which can be context, reconfigurations, and functionalities. This table connects to the **Project** table with the **Tagging_project** table, which receives the *tag_id* from the **Tagging** table and *project_id* from the **tagging_project,** and has Primary Key *project_tagging_id*.

Regarding **tags_realations** table, it has: *relation_id*, an unique id for each relation(Primary Key); the *tag_id* that comes from tagging table, which identifies the tag associated with the word; and the *expression,* which is the word associated with the tag. By associating expressions

(words from the customer sentence) with tags, it was possible to reduce the execution time of tags generation. This table is relevant to this dissertation because it saves the words written by the customer through the GUI and associates those words with a tag, generating a word cloud around some tags. This table can have multiple expressions associated with just one tag. What defines if this association between a word and a tag is inserted in this table is the tagging system explained in the Tagging System sub-chapter, precisely the **Similarity** function.

5.3 Specifications Documents

As mentioned in this thesis, multiple systems work together to develop smart furniture for the customer. With this goal in mind, it is mandatory to have specifications that every system follows so that the work between systems flows in the best way possible. Therefore, in this section, the specifications impacting the User Interface System are detailed, starting with the Idea Specifications, then the Detailed Solution Specifications, and finally Eco Prototype.

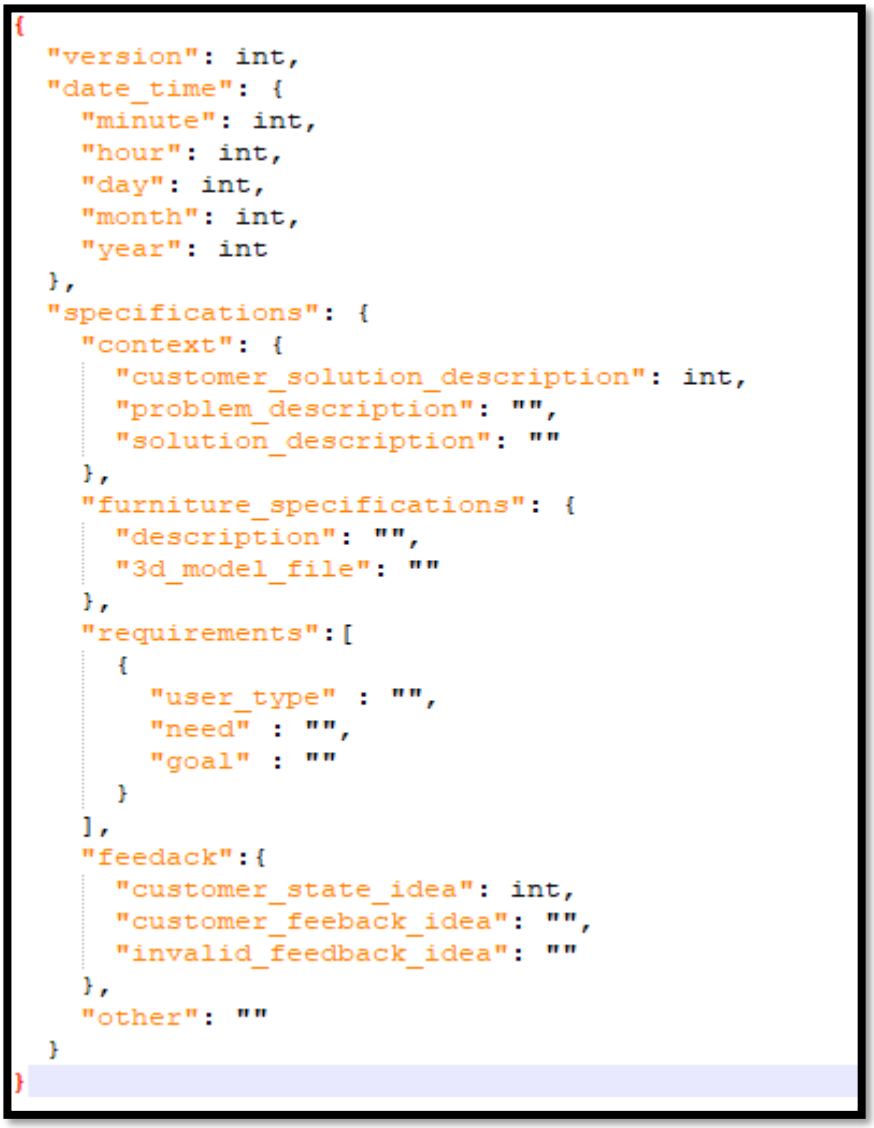
All of these documents are going to follow the JSON format. JSON stands for JavaScript Object Notation. It is a lightweight data-interchange format usually used to send data between computers and is language-independent. This format is syntactically similar to the code for creating JavaScript objects, which makes it easy to convert JSON data into JavaScript objects and vice-versa.

5.3.1 Idea Specifications

Besides aiming for the harmony of all systems when developing smart furniture, these documents also aim to reduce database size and complexity. As illustrated in Figure 5.6, the *idea_specifications* field is present in the idea table. This document is where the customer's ideas, requirements, and functionalities will be inserted. This document will also explain the project overview to the customer before the detailed solution is developed.

As explained before, the customer can refuse the idea specifications and propose changes to the Prototyping System, so this document requires different versions. Therefore, the document has three main fields: the version, the date_time, and specifications. The version indicates how many iterations the document has. The date_time field represents the date and the time of the

version creation. Finally, the specifications contain an object with all the requirements related to the idea. In Figure 5.9, it is possible to see the JSON structure adopted in more detail.



```
{
  "version": int,
  "date_time": {
    "minute": int,
    "hour": int,
    "day": int,
    "month": int,
    "year": int
  },
  "specifications": {
    "context": {
      "customer_solution_description": int,
      "problem_description": "",
      "solution_description": ""
    },
    "furniture_specifications": {
      "description": "",
      "3d_model_file": ""
    },
    "requirements":[
      {
        "user_type" : "",
        "need" : "",
        "goal" : ""
      }
    ],
    "feedack":{
      "customer_state_idea": int,
      "customer_feeback_idea": "",
      "invalid_feedback_idea": ""
    },
    "other": ""
  }
}
```

Figure 5.9 - JSON format of *idea_specifications*.

Concerning this dissertation, the most important fields are inside the specifications object. Inside this object, there are five other objects: context, furniture_specifications, requirements, feedback, and other.

The "context" object contains the description of the problem (problem_description), the solution idea described by the customer (customer_solution_description), and the description of the solution provided by the Prototyping System (solution_description). This last field will be

shown to the customer via the User Interface System developed in this dissertation, allowing the customer decision about idea approval.

The "furniture_specifications" object contains the furniture description and a link to the 3D furniture model (3d_model_file). However, the "furniture_specifications" object does not directly impact this dissertation.

"Requirements" is a different object because it is an array. There can be multiple requirements in the "requirements" object, each with a user_type, need, and goal. The user_type indicates the type of user; the need field identifies the necessity to apply the requirement; the goal explains the expected outcome.

As explained in the Project State sub-chapter, the customer can approve or refuse the idea. If the idea is refused, the customer is required to provide feedback for idea improvement. The "feedback" object allows this to happen. This object contains the customer_state_idea, a field that represents the state of the idea specifications, and the customer_feedback_idea, the field that saves customer feedback provided via the User Interface System. The invalid_idea_feedback saves feedback from the Prototyping System when something in the idea is not valid.

The "other" field saves the information regarding the selected functionalities by the customer amongst the suggestions from the tagging system. Additionally, this field saves any other information that does not fit the structure.

5.3.2 Detailed Solution Specifications

Following the Idea Specifications structure, the Detailed Solution Specifications contain the version, date_time, and specifications objects. To the specifications object from the Idea Specifications structure, new fields/objects were added. Therefore, in addition to context, furniture_specifications, feedback, and other fields/objects, were added the non_functional_requirements, detailed_solution_descriptions, technical_specifications, runtime_specifications, and instructions fields/objects.

Following the Idea Specifications document, all information was grouped in one document to have all systems working harmoniously, reducing complexity and saving space in the database. This document is stored in the *detailed_solution* table, and the document's structure can be seen in detail in Figure 5.10.

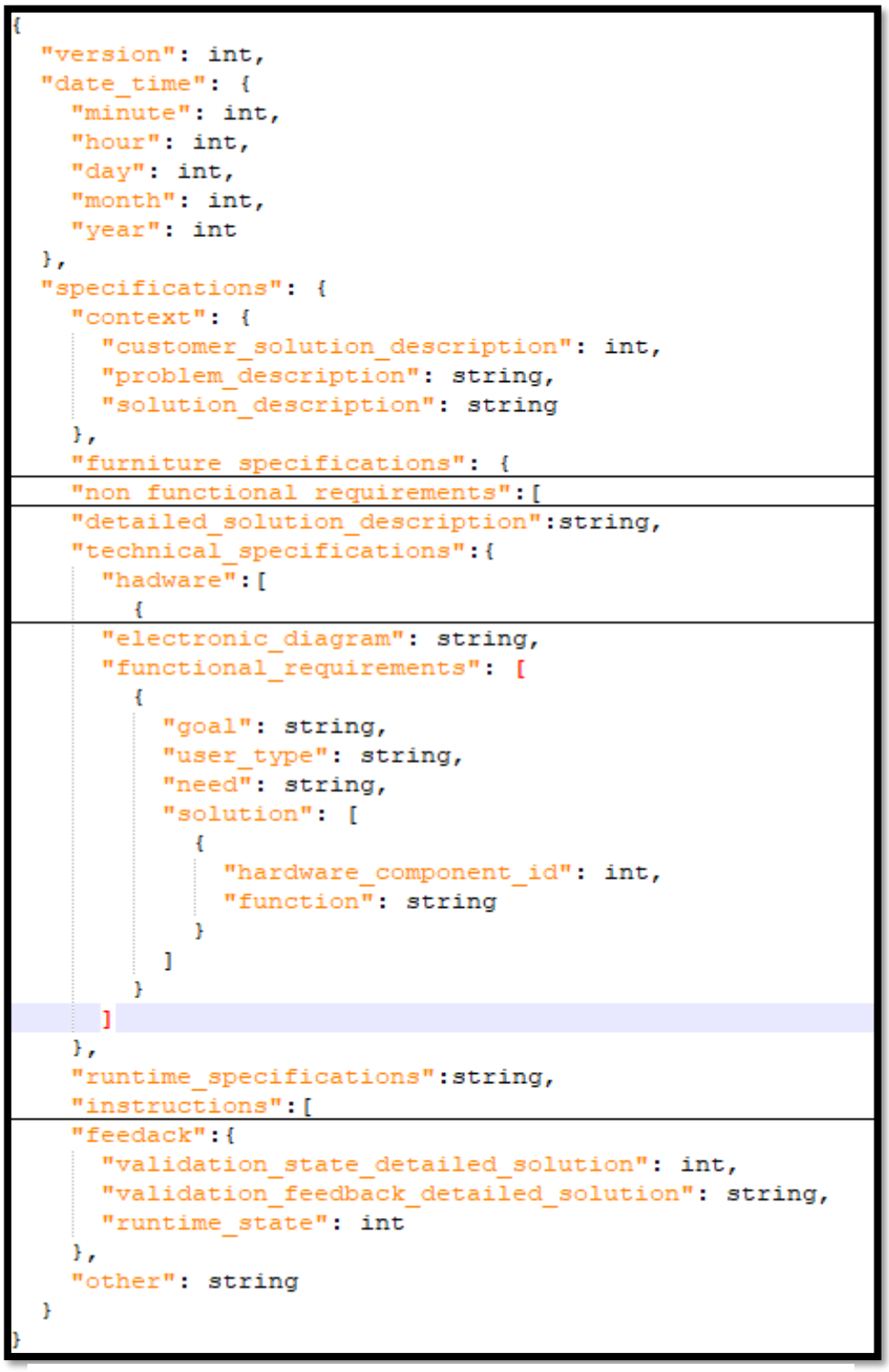


Figure 5.10 - JSON format of *detailed_solution_specifications*.

Figure 5.10 presents part of the structure of the detailed_solution_specification JSON to highlight the fields critical to the smartification service developed in this dissertation. The version and date_time objects follow the same structure as the other documents. Following that, it is

possible to see one big object, specifications, cut into multiple small objects. However, the objects not referenced here are also essential to develop smart furniture but are referent to the other systems involved.

The "context" object is the first to appear in the specifications object that contains customer_solution_description, problem_description, and solution_description. These three fields refer to customer answers during the smartification process's initial steps.

The "technical_specifications" object contains three other small objects, hardware, electronic_diagram, and functional_requirements. Among these, the functional_requirements object is relevant within this dissertation scope. The proposed service retrieves the information from the goal field to make suggestions and retrieve functionalities from other existent projects. These functionalities are suggested to the customer in the GUI.

Following the smartification process, after the creation of the detailed solution, the systems start to develop the prototype. For that, a document is needed to keep track of the changes occurring.

5.3.3 Prototype Specifications

The prototype specifications also use the JSON data format for all other specifications. This specification is stored in the prototype table, as shown in Figure 5.6. The prototype specifications have two versions, the *eco_prototype*, and the *final_prototype*. Even though, *eco_prototype* and *final_prototype* have the same structure, *eco_prototype* is developed to present to the customer a more environmentally friendly prototype, while *final_prototype* presents the standard prototype without the same environment concerns. This enables the customer to choose between multiple solutions that have different building materials with different environmental sustainability concerns.

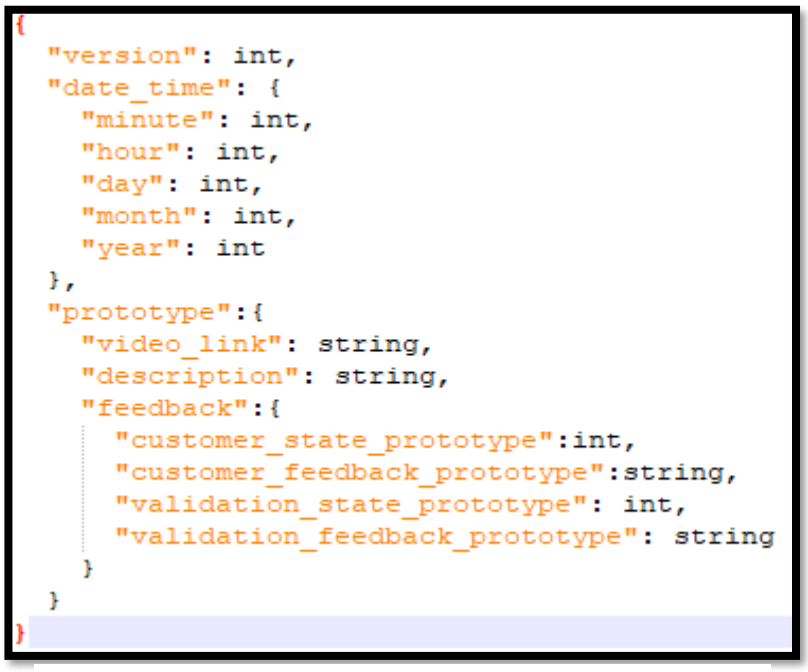


Figure 5.11 - JSON format of *prototype_specifications*.

The version and the date_time fields inside the *final_prototype* structure follow the same structure as the previously explained specifications. Then it has the prototype object. This is where the video link of the prototype is stored, so it can be easily changed over the iterations that the prototype will suffer. The description field of the prototype object will be used to describe the changes to the customer. When the user approves the last version of the prototype, this description will be updated and available for future projects.

Inside the prototype object, it also exists the feedback object that contains customer_state_prototype, customer_feedback_prototype, validations_state_prototype and validation_feedback_prototype. The customer_state_prototype is an integer that will vary between 0 and 6, corresponding to the change in states 7 and 9 in project_state explained in the Project State sub-chapter. The validation_state_prototype will vary between 0 and 2 to keep track of the changes in the Quality Assurance System, meaning a minor impact on this dissertation.

The customer_feedback_prototype allows customer feedback about the prototype during the development stages, enabling constant interaction between the furniture development systems and the customer. This field can be changed multiple times, depending on the iterations, until the smartification process finishes the prototype development.

5.4 RESTful API

Following the application structure presented in Figure 5.1, it is necessary to connect with the database to save and load information. This is done via an API, in this case, a RESTful API. This is based on representational state transfer, REST, an approach commonly used in developing web services for communication. There are several options to have this running in the cloud. However, the Glitch website, which allocates in its cloud, was used to have the RESTful API running. Glitch is the platform the provides the infrastructure to build web applications by supporting frameworks like React and Node.js that ease the process of having web app's up and running [69].

This was written in Node.js, open-source software that allows the execution of JavaScript code outside of the browser. One of the many reasons to choose this framework is because Node.js is well known for handling multiple requests simultaneously with high performance. This needs to be considered when one of the goals of the INEDIT project is to have this service up and running with multiple smartification projects running simultaneously. For example, to connect to the PostgreSQL database, node-Postgres was used, a collection of modules allowing interfacing with PostgreSQL.

The implemented methods in this RESTful API followed the CRUD system, which allows making the four basic operations: Create resources (POST), Reading resources (GET), Update resources (PUT), and Deleting resources (DELETE).

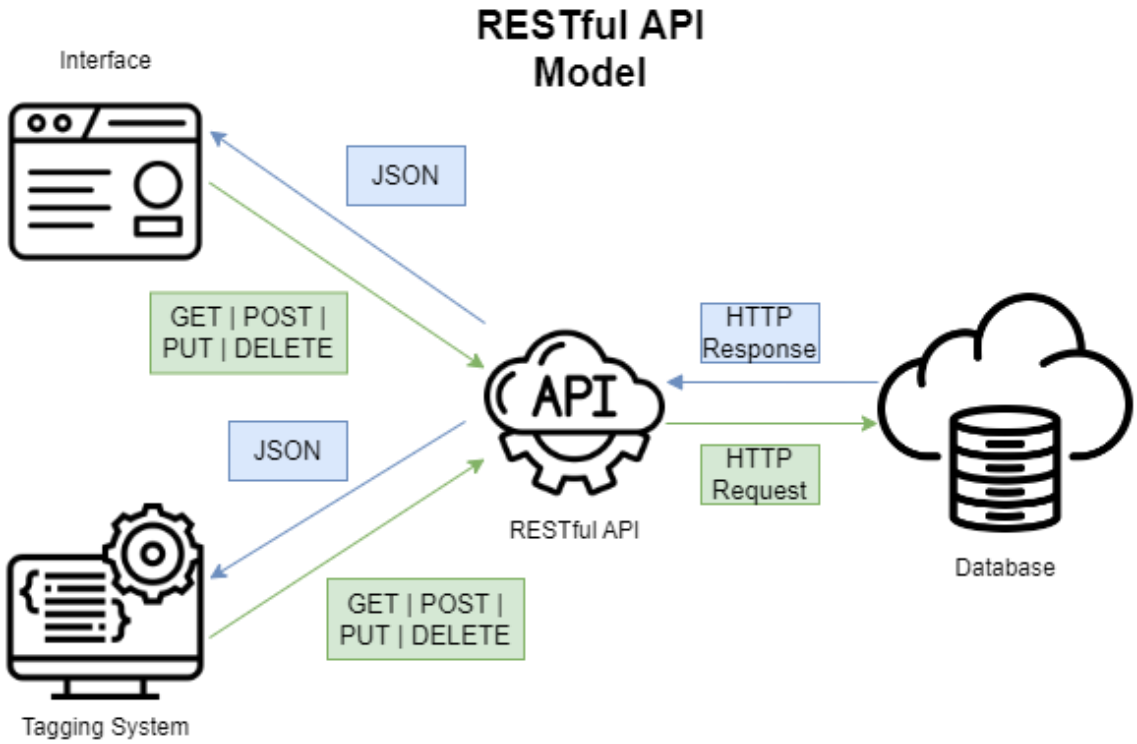


Figure 5.12 - RESTful API model.

Figure 5.12 presents the RESTful API model in the smartification service developed in this dissertation. The RESTful API communicates with all the systems and handles all requests. However, to save energy, the API is asleep whenever it does not receive HTTP requests. The typically flow is that the CRUD methods come from both Interface and the Tagging System as HTTP requests to the API. Then, the API process these requests into queries for the database engine as an HTPP request, answering the result of the query as an HTTP response. The API deal with this response, returning a file in JSON format as an HTTP response to the system that made the request, either the Tagging System or the GUI.

5.5 Interface

The application is composed of several systems described in the previous sections. However, since this is a more consumer-oriented service, the interface is essential to work as desired.

As the previous chapters explained, the interface interacts directly with the tagging system, or **backend**, and with the RESTful API. The customer uses the interface to develop the smartification piece, often called the application's **frontend**. While developing one, the developer must get into the user's skin and provide controls that make this experience as smooth, easy, and accessible as possible. This is done to allow every user with whatever device and whatever capability to use the interface. With these concerns and goals in mind, these interfaces were developed to maximize the development of the smartification project by the customer. However, a connection to the Internet is always needed since the interface runs on a web application, which is a good choice considering that it reaches around 60% of the global population [74]. This interface was mainly developed to run as a web application, however the choice of the Flutter framework was considered due to the possibility of scaling the interface into a mobile app.

The Flutter framework uses the programming language Dart to allow the developer to build mobile, web, and desktop apps. This allows the creation of unique looks with the User Interface that have a normal behavior across platforms with the same code. Adding to these benefits, it has high-speed testing, which every developer desires, a growing community, and rich libraries. The built-in HTTP library is one of them, which in this case, helps communicate with the backend service and the RESTful API.

Designing User Interfaces is focused on two objectives. First, it predicts the user's possible needs. Secondly, it ensures that the interface elements are accessible, understandable, and usable. Nowadays, Interface design is more straightforward due to the abundance of interfaces users have been subjected to in recent years, which makes them expect certain elements to have a specific behavior. Elements can be considered buttons, checkboxes, text boxes, hyperlinks, etc. In a general way, we can define three major categories of elements: Input Elements, Output Elements, and Helper Elements. In addition to designing an appealing interface, it is more important to give the user the expected behavior for each element in terms of functionality.

5.5.1 User Journey

User Journey is the steps users take through the GUI to reach their goal. This journey involves multiple interfaces, interactions, and decisions by the user, each leading to different paths. The user journey can be seen in detail in Figure 5.13.

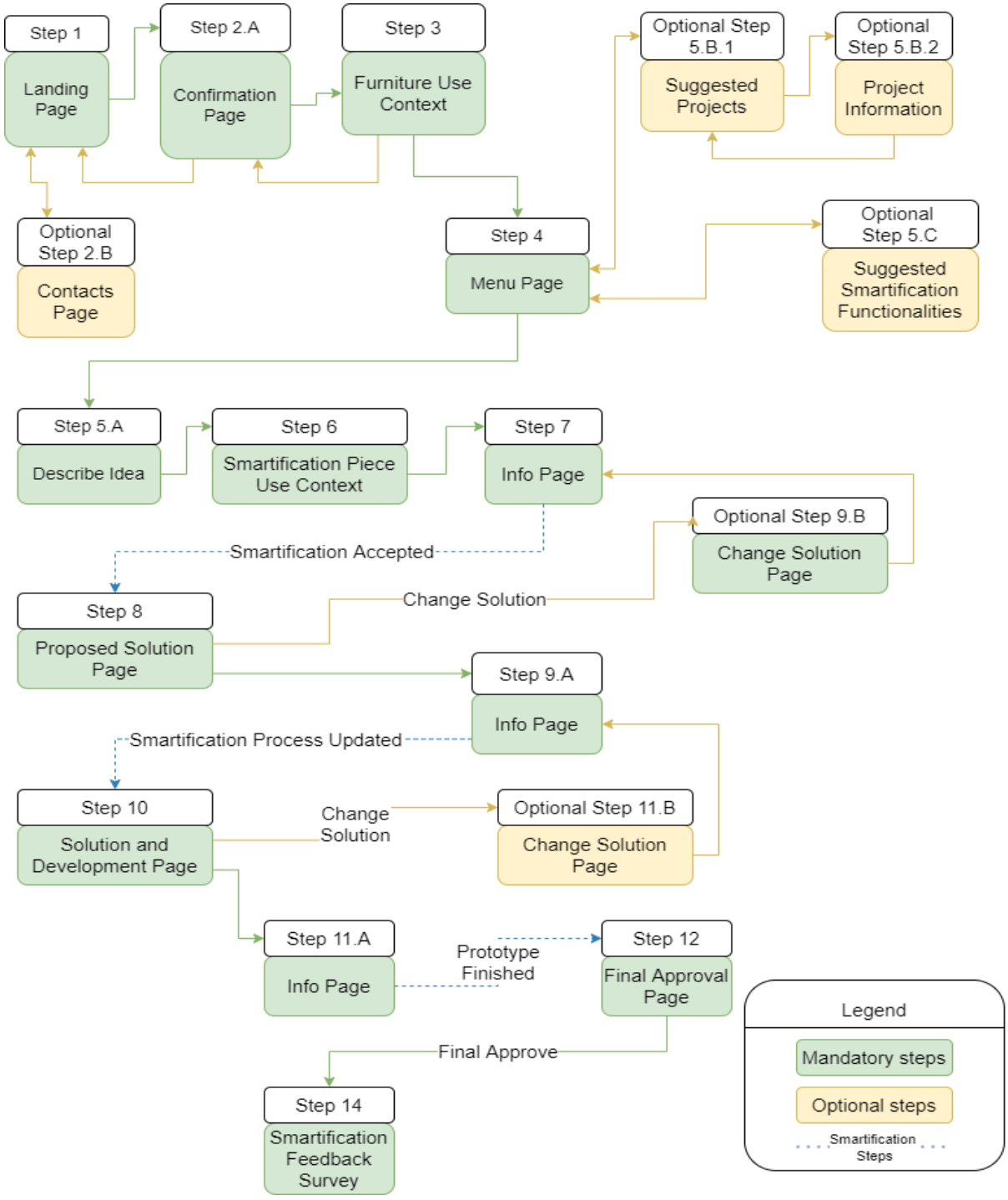


Figure 5.13 - User Journey.

The user journey gives an overview of all the interfaces in the smartification service and the available options. Figure 5.13 does not have step 13, jumping from 12 to 14. This happens on purpose because step 13 was regarding the reconfigurations page which was not implemented in this dissertation. As expected, building a good interface from scratch is challenging, and this dissertation was no exception.

5.5.2 Interface Creation Process

To obtain the final interfaces, several processes and countless changes were made. It started with drawings in notebooks, and from constant feedback to improve little details, it was possible to end with a user-friendly interface. This evolution is shown in Figure 5.14.

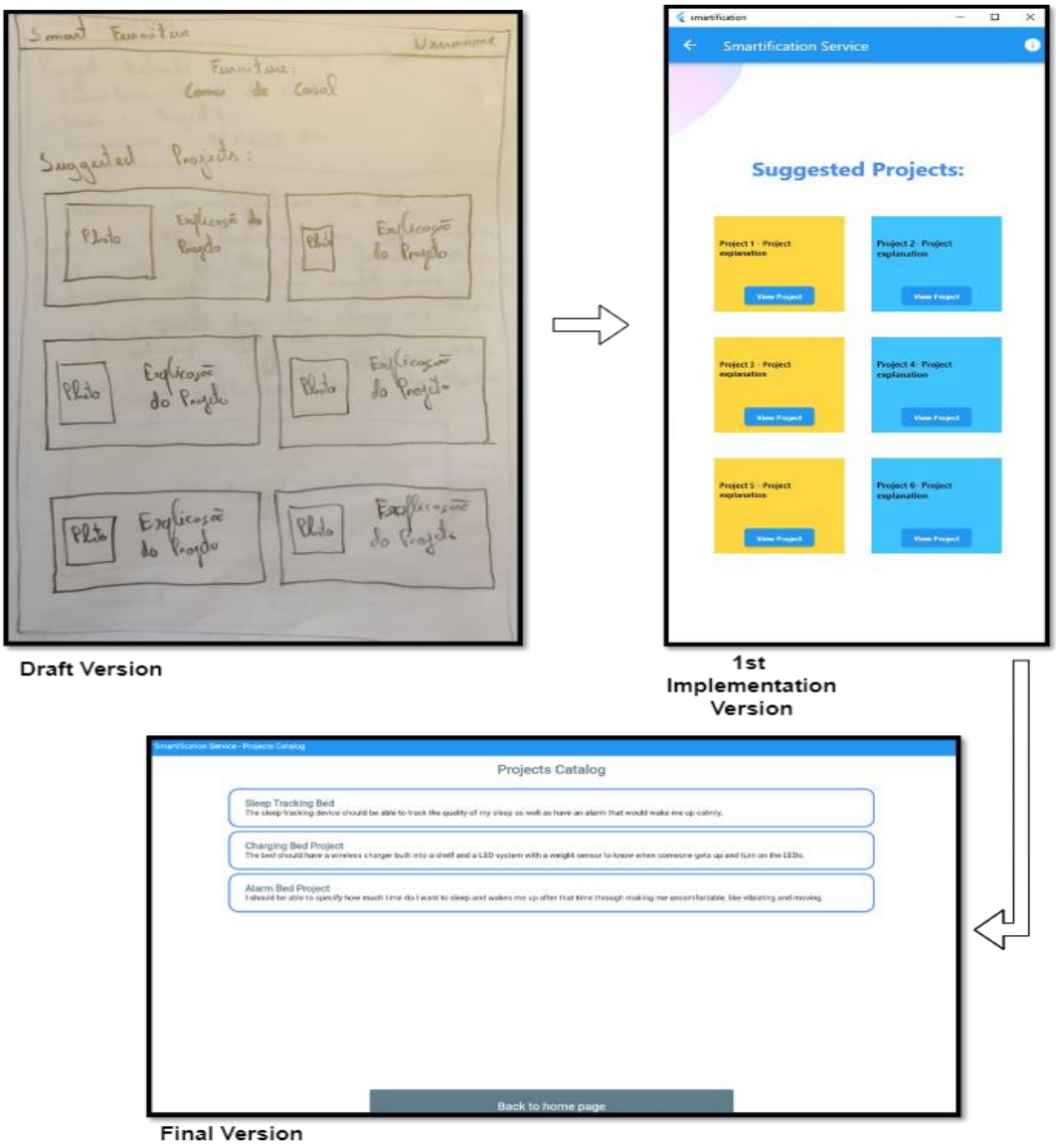


Figure 5.14 - Interface creation process for Projects Catalog interface.

From Figure 5.13, Step 5.B.1 interface was selected because it shows the process done in all interfaces and the backend (tagging system) working with the frontend.

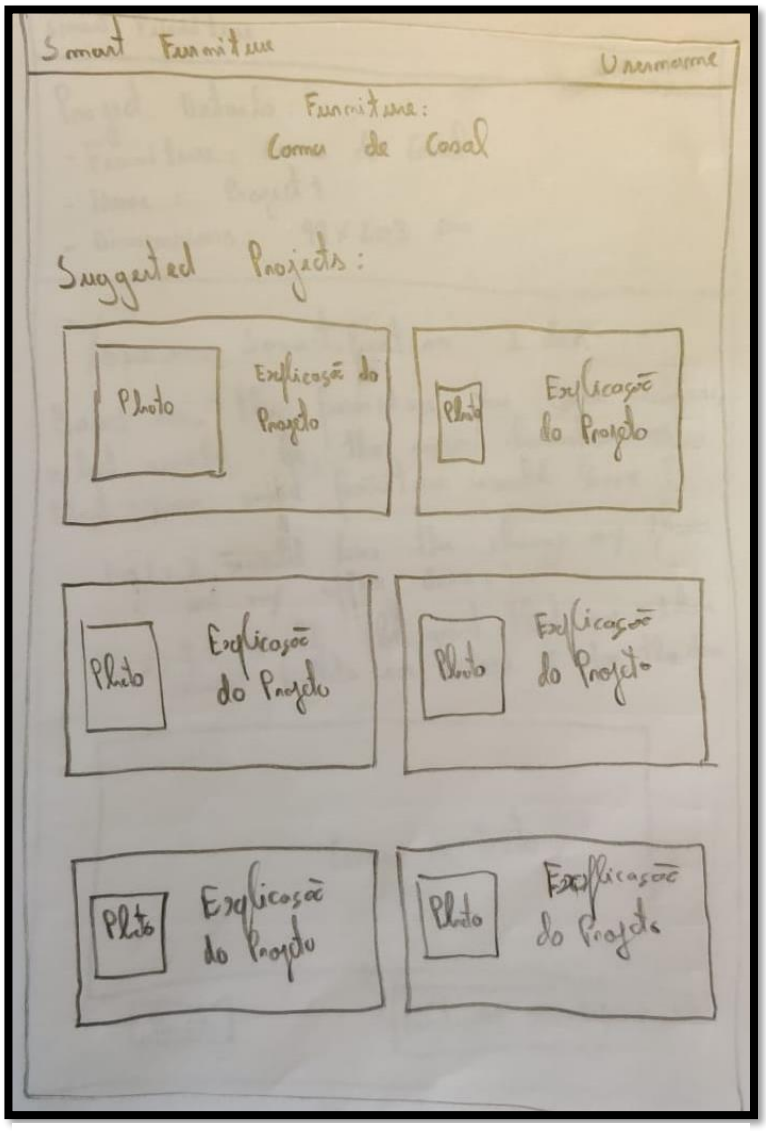


Figure 5.15 - First draft version of Projects Catalog interface.

In Figure 5.15, it is possible to see the Projects Catalog interface's first version. In this version, the top of the screen contained the title, and the username, followed by the furniture. The rest of the page would contain clickable boxes with an explanation of the project and a photo. These clickable boxes would lead the user to a page with detailed information about the selected smartification project. From surveys requesting interface opinions, the tittle was changed alongside the username field at the top of the page. Specifying the users' project furniture was found pointless and removed. However, the idea of the clickable boxes was

maintained with minor adjustments. From this first draft, removing what was not desired was possible, and moving from drawing the interface to coding it.

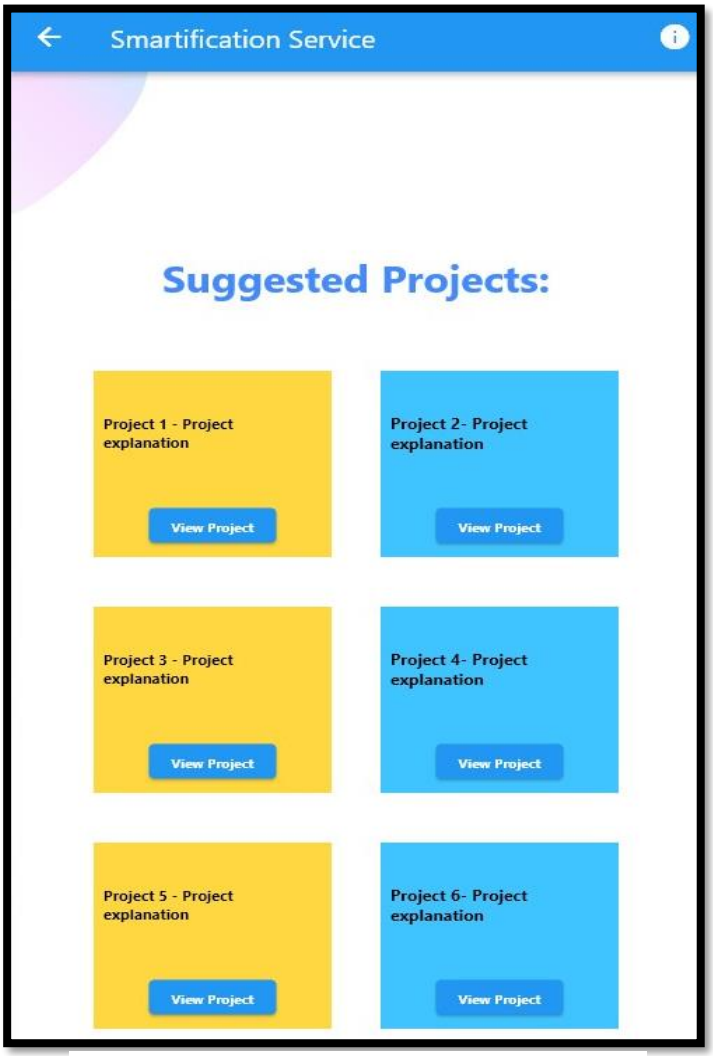


Figure 5.16 - First implementation of the Projects Catalog interface.

Figure 5.16 is considered the second iteration of the Products Catalog Interface. However, the first code implementation for this interface shows the differences between the first draft and the implementation. In this stage, the project was focused on developing for mobile. However, it was decided to develop a web-oriented interface. Since Flutter supports one codebase for all devices, the change of direction of the project did not impact the interface development.

From the first draft, the username was replaced by the "i" icon (upper right side of the interface), which opens the project details when clicked. The center of the page followed the draft idea by implementing the clickable boxes. Once again, when the first implementation of the

interface finished, that same interface was shown and discussed, leading to multiple changes. These iterations happened multiple times, with constant feedback leading to several changes in the interface. The following Figure 5.17 shows the last version of the interface.

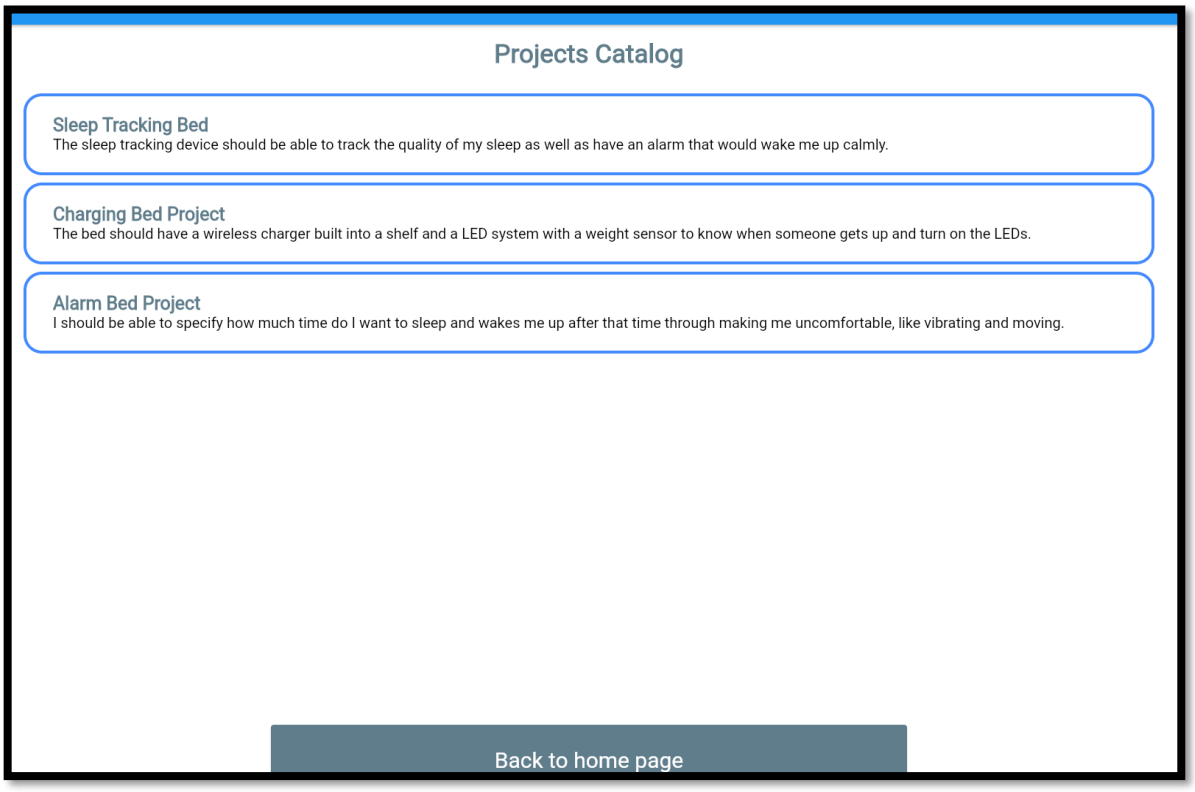


Figure 5.17 - Projects catalog interface.

In this last version, the "i" icon was removed because it was found useless, and the tittle of the page was changed to ease the user's understanding. In addition, the clickable boxes were changed to a different style to make all the box areas clickable and user-friendly. The arrow on the left upside of the interface present in the first implementation was also removed and replaced by a button at the end of the page to return to the previous page, in this case, "Home Page". All of these changes focused on user usability and accessibility, which are once again fundamental concepts when developing interfaces. This process was done in all the twenty-one interfaces developed for the frontend part of the smartification service.

As previously mentioned, this smartification service comprises several systems that communicate. The outcome of these interactions between the systems can be seen in several interfaces. To show everything in motion and working, a use case scenario will be presented along with an explanation of each interface. The steps described here are shown in Figure 5.13.

The Bob example from the User Story sub-chapter will be adapted to better understand this process's details. In addition, some interfaces presented were cropped to improve the readability of this dissertation and to enhance some essential details to understand the smartification service working.

5.5.3 Step 1 - Landing Page

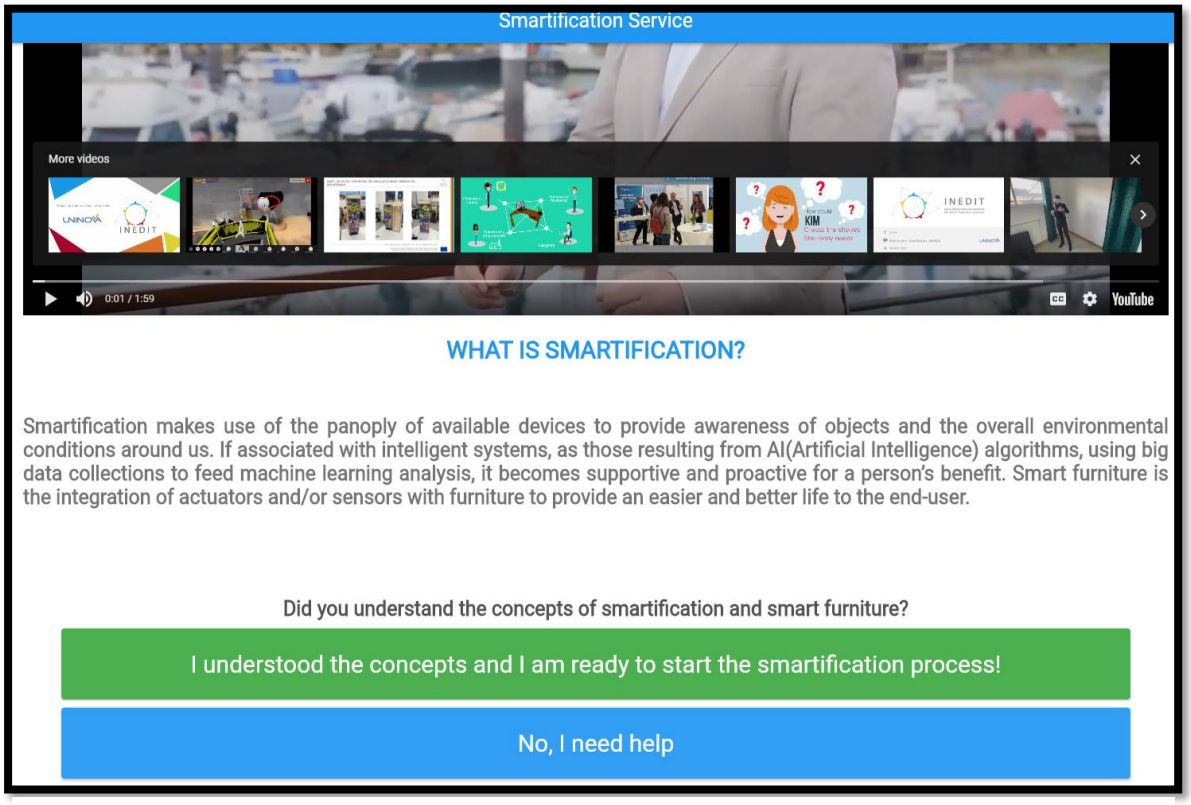


Figure 5.18 - Landing page interface.

This interface shows a video explaining smartification and a text explanation. This video starts by itself when the customer arrives at this interface. The text is necessary due to accessibility/usability concerns. This interface has two buttons that lead the user to different pages. The "No, I need help" button leads the user to Optional Step 2.B - Contacts Page, while the "I understood the concepts and I am ready to start the smartification process!" button leads the user to Step 2.A - Confirmation Page.

Optional Step 2.B interface is not shown in this dissertation. However, this is a simple interface where the user can click a button and gets redirected to the INEDIT Contacts page.

Bob arrives at this page after being redirected from the INEDIT platform. Bob checks the video and reads the explanation, understanding the smartification process. Then, Bob presses, "I understood the concepts, and I am ready to start the smartification process!" and proceeds to Step2.A – Confirmation Page.

5.5.4 Step 2.A - Confirmation Page

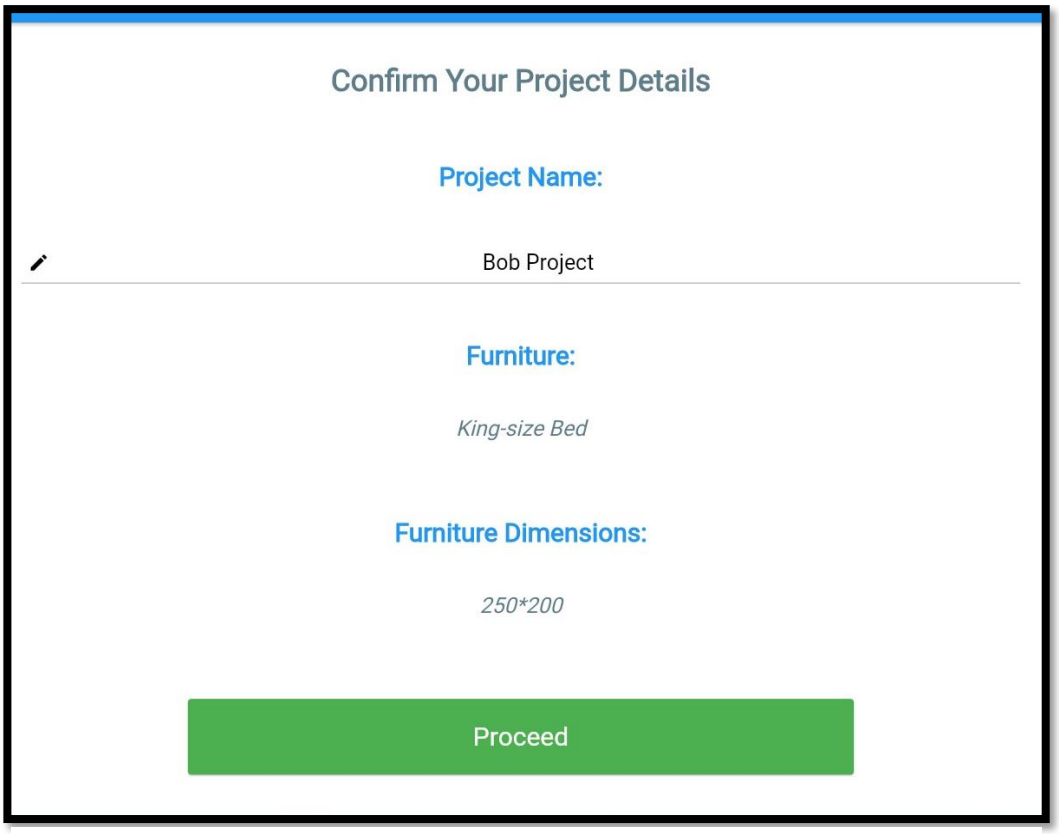


Figure 5.19 - Confirmation page interface.

Before the customer starts the smartification process, the possibility to confirm project information previously inserted at the INEDIT platform is provided. Here the user can change the project name and proceed, leading him to Step 3 - First Question Page.

Bob had already specified the furniture, dimensions, and project name, as it is possible to see in Figure 5.19. He is comfortable with the project name and clicks "Proceed" leading him to Step 3.

5.5.5 Step 3 - First Question Page

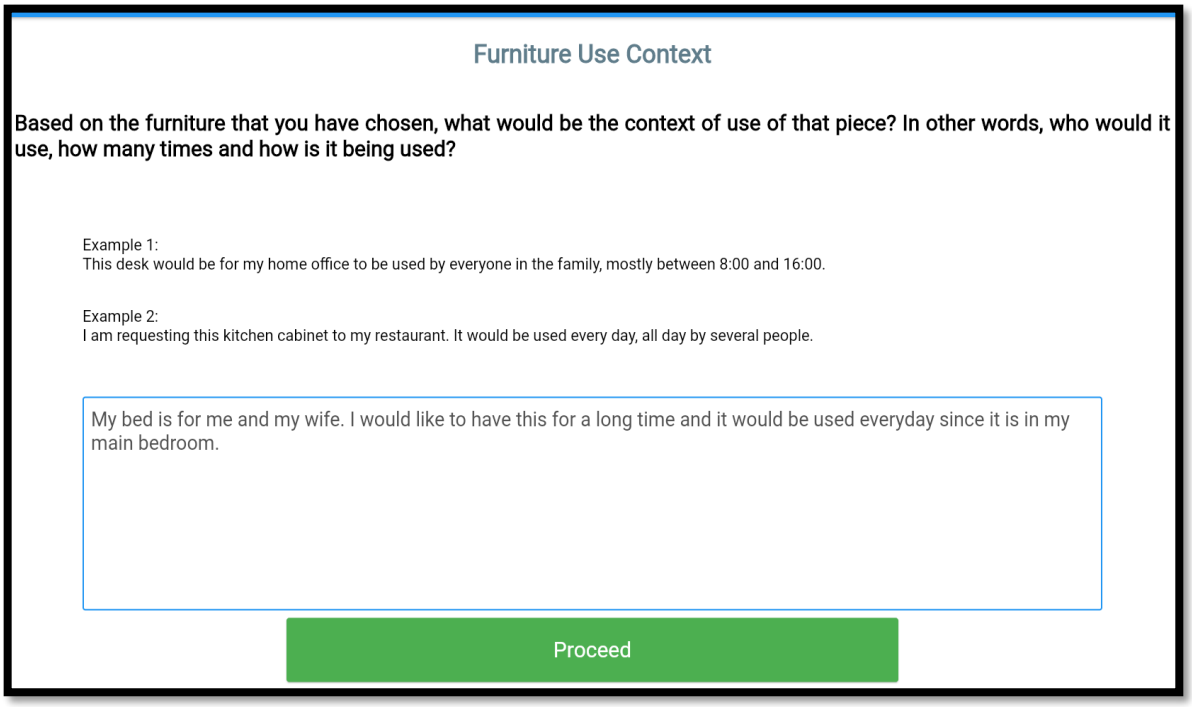


Figure 5.20 - First question page interface 1/2.

From this interface forward, it is possible to see the tagging system working. Here the customer is questioned about the furniture using context followed by two answer examples. These answer examples were studied and improved over time to influence the customer to give similar answers. With this, preparing the backend to receive answers with a specific structure was also possible. This step comprises two interfaces, the first that requires an answer from the user to proceed and the following interface, which can be seen in Figure 5.21.

In this first interface of this step, Bob inserts the answer, "My bed is for my wife and me. I would like to have this for a long time, and it would be used every day since it is in my main bedroom." and clicks "Proceed". By clicking "Proceed", the backend part is called, and the tag generation begins, appearing a loading message while this happens. Once it is finished, the interface changes to the following.

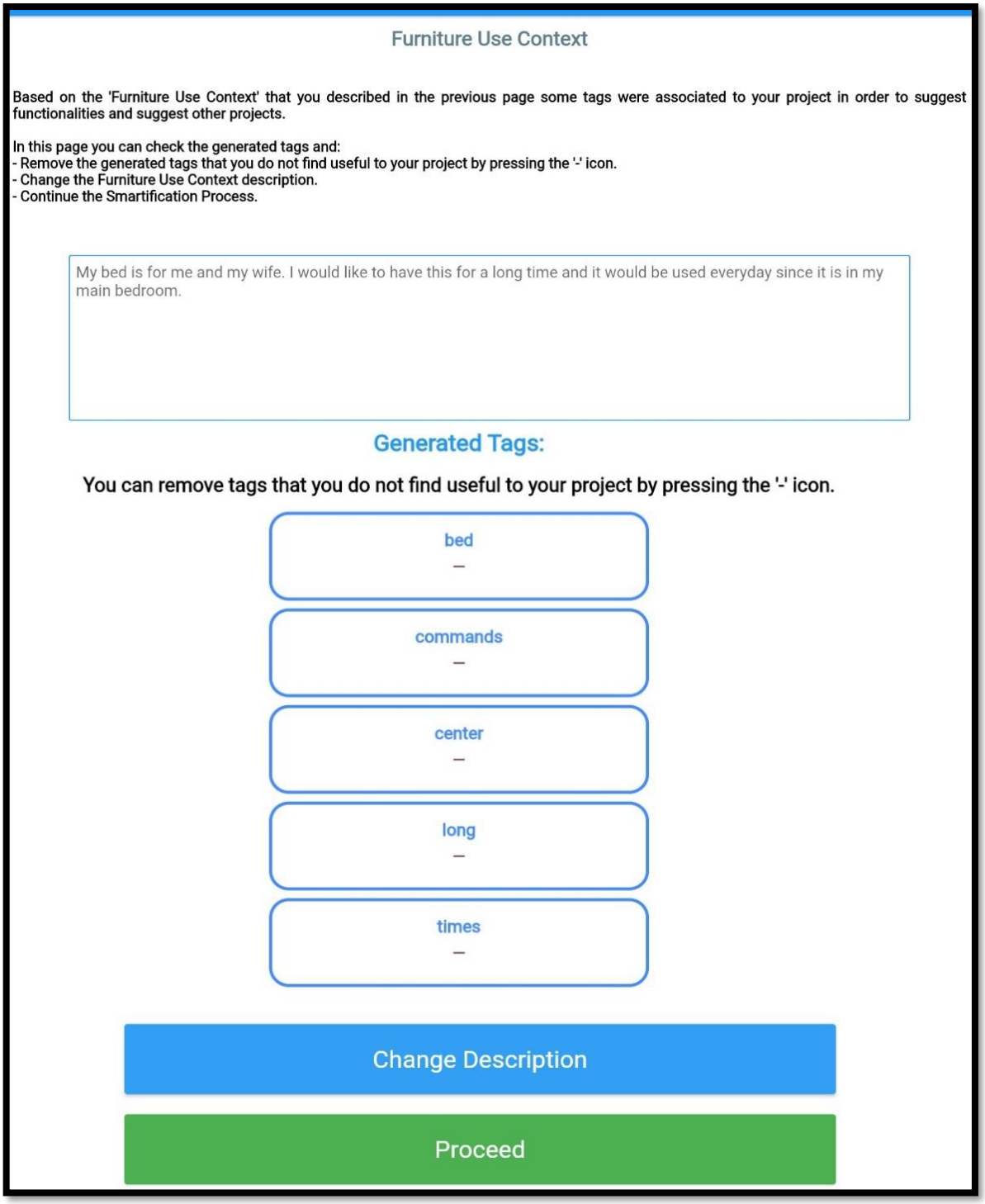


Figure 5.21 - First question page Interface 2/2.

The interface changes, but the customer stays on the same page. A new message appears at the top of the interface explaining the available options. The answer given by the customer remains visible, not editable, and can only be changed if the customer press the "Change

Description" button. The tags generated by the backend also appear in this interface associated with a button enabling their removal from the project. The options in this interface are: "Proceed", leading the user to Step 4 - Menu Page, or "Change Description", which deletes the tags generated and allows the customer to change his answer, leading the customer to the previous interface.

Bob removed the "long" and "times" tags, maintained the "bed", "commands" and "center" tags associated with the project, and clicked the "Proceed" button, leading him to Step 4.

5.5.6 Step 4 - Menu Page

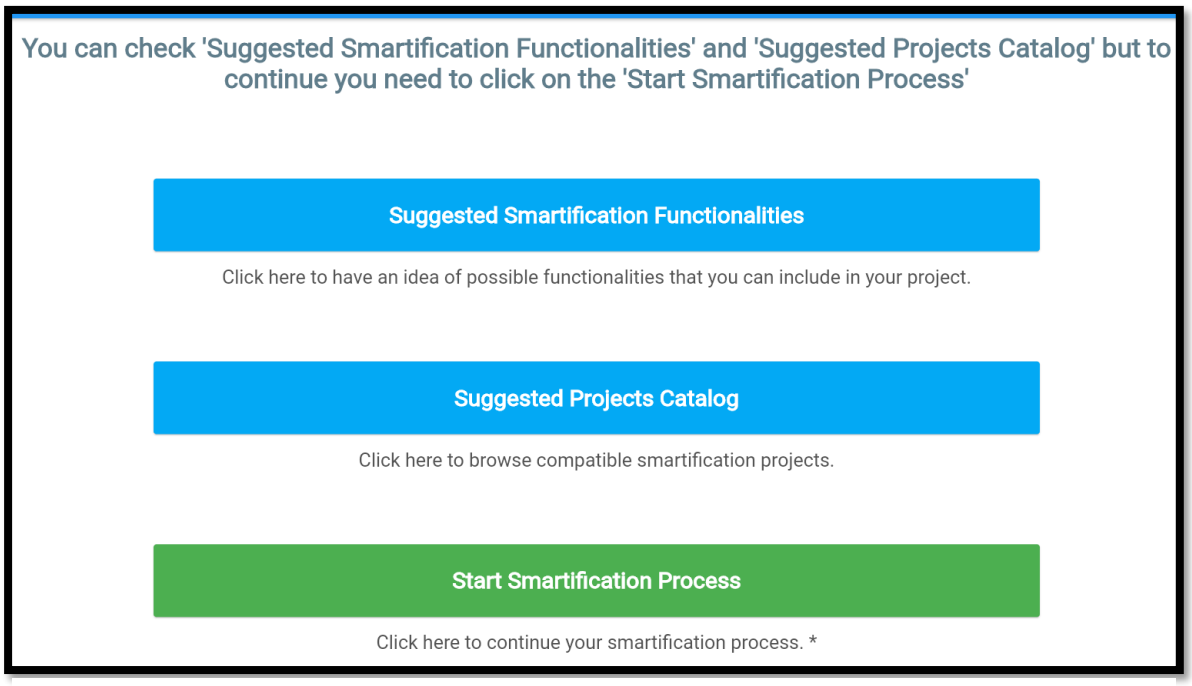


Figure 5.22 - Menu page interface.

This interface presents information about customer options at the top of the screen. Here the customer can choose between three buttons, redirecting to different pages. However, to start the Smartification process, it is always necessary to click the green button. Bellow each button, there is a brief explanation of what the customer will find. For example, the first button leads the user to the Optional Step 5.C- Suggested Smartification Functionalities, the second one to the Optional Step 5.B.1 - Suggested Projects, and the third button leads the user to Step 5.A - Second Question Page.

Bob started by clicking "Suggested Smartification Functionalities".

5.5.7 Optional Step 5.C- Suggested Smartification Functionalities

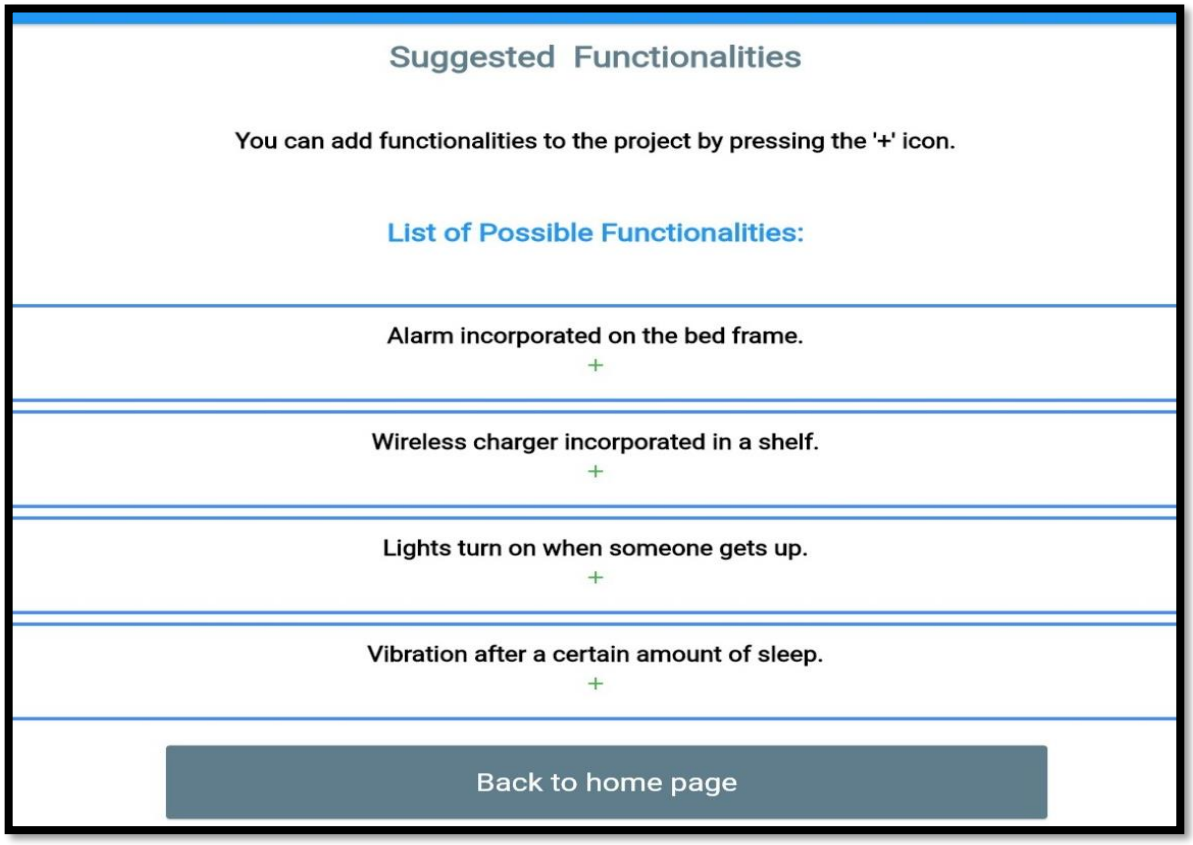


Figure 5.23 - Optional step 5.C interface.

This interface further illustrates the backend system working. Here, it is possible to see a list of functionalities retrieved from finished smartification projects. All of these are functionalities that were implemented due to other customers' requests. These functionalities are not randomly presented but retrieved from projects containing the same tags as this project. The tags used to search for functionalities were generated in Step 3 - First Question Page. Customers can add functionalities to their projects in this interface by pressing the "+" icon below each proposed functionality. The button "Back to home page" leads the user to Step 4 - Menu Page again.

From this list, Bob added the functionality "Wireless charger incorporated in a shelf" and "Lights turn on when someone gets up". Every time the user adds functionality, the message in Figure 5.24 pops up. The functionality is added to Bob projects and removed from the list in Figure 5.23.

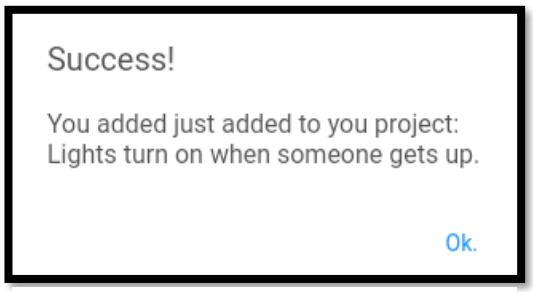


Figure 5.24 - Optional step 5.C pop-up.

Then, Bob presses the button "Back to home page" which leads him again to Step 4 - Menu Page. Finally, in the interface from Figure 5.22, Bob decided to explore other smartification projects and pressed the button "Suggested Projects Catalog" leading him to Step 5.B.1 – Suggested Projects.

5.5.8 Optional Step 5.B.1 - Suggested Projects

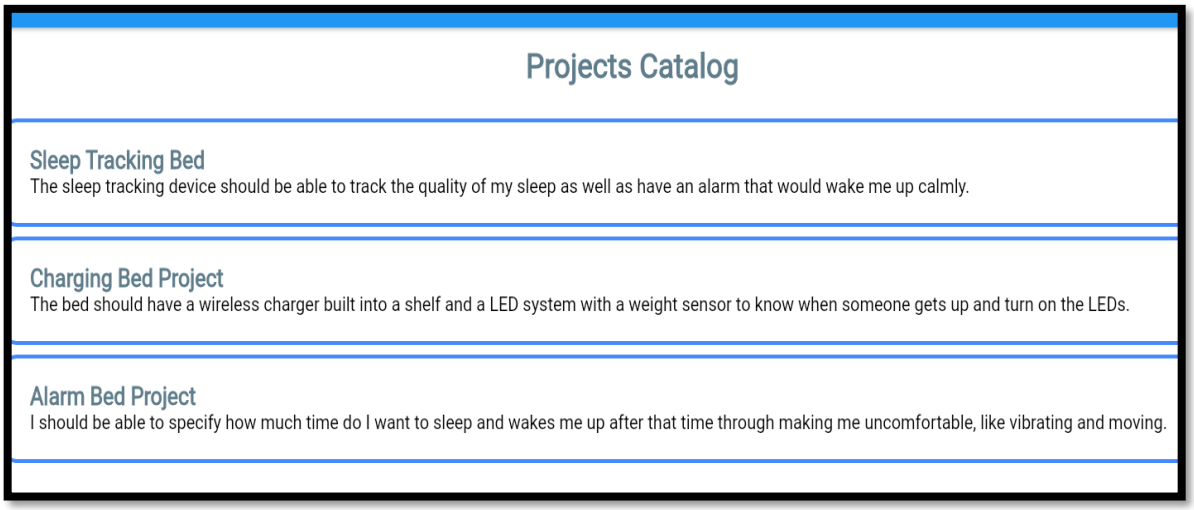


Figure 5.25 - Optional step 5.B.1 interface.

Here it is possible to check other smartification projects with similar furniture. This list of suggested projects is not static and depends on the projects present in the database. Over time, the Projects Catalog will increase in size, enabling customers to have more projects to check and retrieve functionalities. In this interface, the smartification need was added below the project's name. The user has two options, check one or multiple projects by clicking on the clickable area of each project, or return to Step 4 - Menu Page by clicking on the "Back to Home Page" button.

Bob clicks on the Sleep Tracking Bed project to get ideas for his bed project, leading to Optional Step 5.B.2 - Project Information.

5.5.9 Optional Step 5.B.2 - Project Information

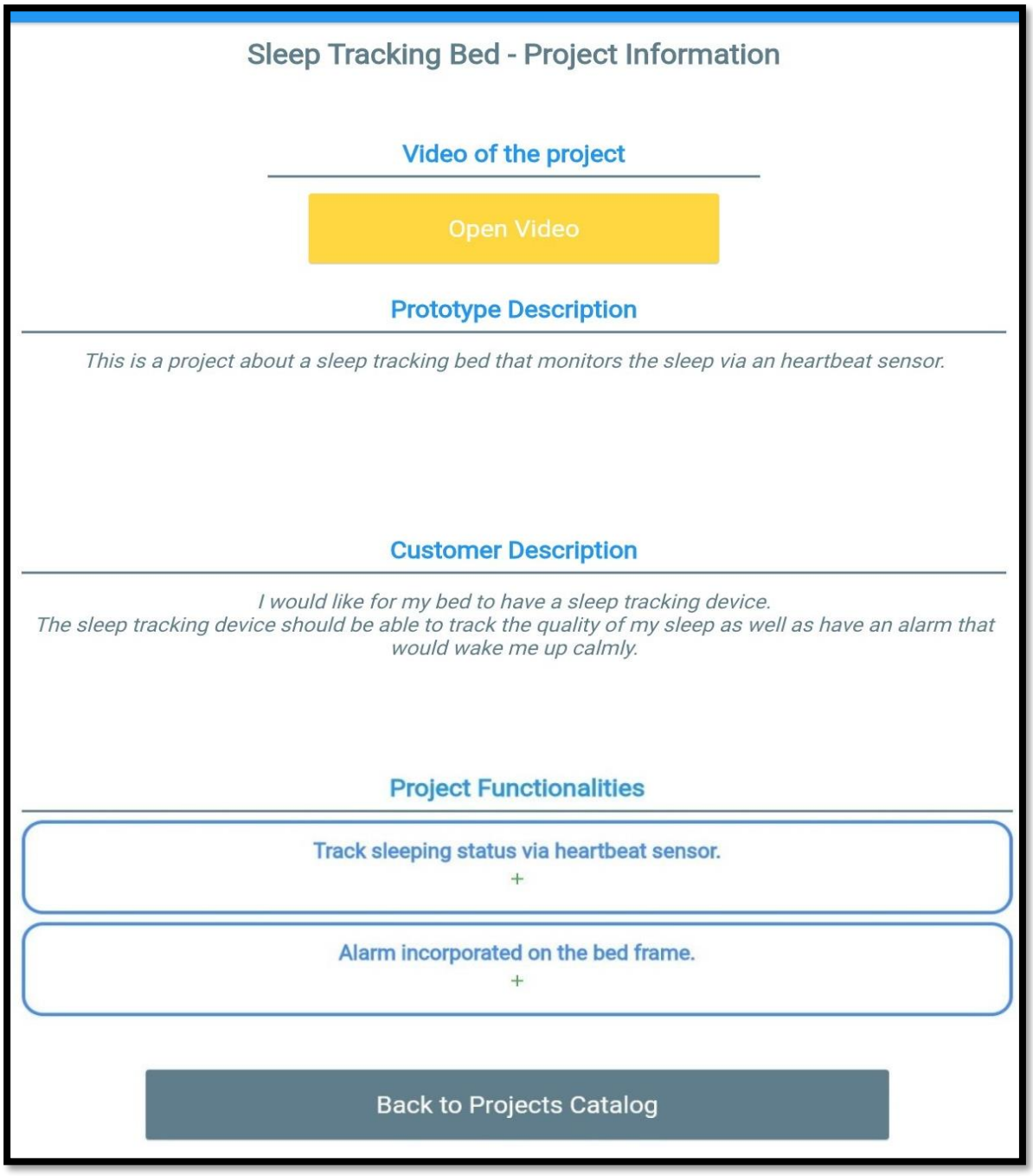


Figure 5.26 - Optional step 5.B.2 interface.

This page enables the customer to check several details about the selected project. This page structure was designed and imagined to be as simple as possible and to highlight the essentials: the possibility to check a video of the finished smart furniture, the associated functionalities, and the description used by the customer for smartification needs. These functionalities

also follow the idea of not showing already added functionalities. In this case, if Bob already added "Track sleeping status via heartbeat sensor." in Optional Step 5.C- Suggested Smartification Functionalities, this functionality would not appear even if it was part of the selected project. A message appears as a pop-up when the user presses the '+' button, like in Figure 5.24 .

Bob checked the project video by pressing the "Open Video" button. Then, Bob checked the prototype description and the project functionalities. He added "Alarm incorporated on the bed frame" to his project and returned to Step 4 - Menu Page, where he started the smartification process by pressing the "Start Smartification Process" button. Leading Bob to step 5.A – Second Question Page.

5.5.10 Step 5.A - Second Question Page

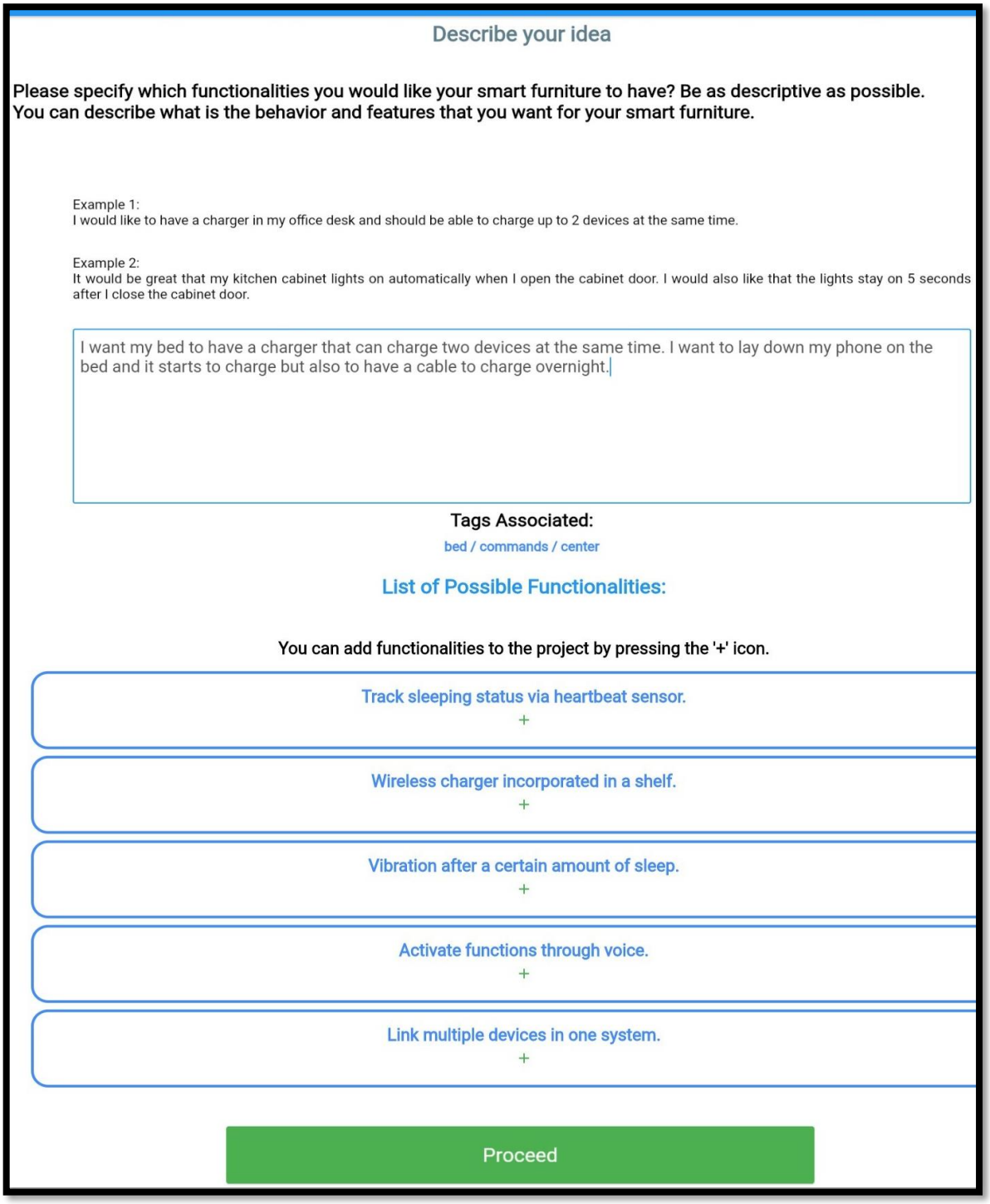


Figure 5.27 - Step 5.A interface 1/2.

Almost all the working pieces of this smartification service are in action on this page, which can be considered the most important page for this dissertation. Consequently, this page is the

one that helps the customer the most by providing several functionalities before and after the smartification need description.

Before the customer writes the smartification need, the functionalities provided are the same as shown in Optional Step 5.C- Suggested Smartification Functionalities, without the ones already added to the project. This page follows the structure used in Step 3 - First Question Page, which is also used in Step 6 - Third Question Page. It starts with a question followed by two answer examples and a text box enabling the customer to describe the smartification need. The customer is asked to specify the functionalities desired for the furniture piece. Then, the associated tags are displayed just for the user information, followed by the suggested functionalities. These suggested functionalities follow the same idea of Optional Step 5.C- Suggested Smartification Functionalities, meaning that the displayed functionalities are retrieved from projects with at least a similar tag to the user's project. Again, to have an intuitive interface, a message pops up on the screen when the user adds a functionality, as shown in Figure 5.24. The "Proceed" button triggers the tagging system and generates the tags from the customer's answer. These new tags will instantly benefit the customer because they will contribute to suggesting fitting functionalities in the following interface.

Bob checked the examples, wrote his smartification need, and before clicking "Proceed" added the functionality "Activate functions through voice.". Once the "Proceed" button is clicked, a waiting message appears while the tagging system generates the tags. When this is over, the interface is updated, but the user remains in the same step. The following interface appears.

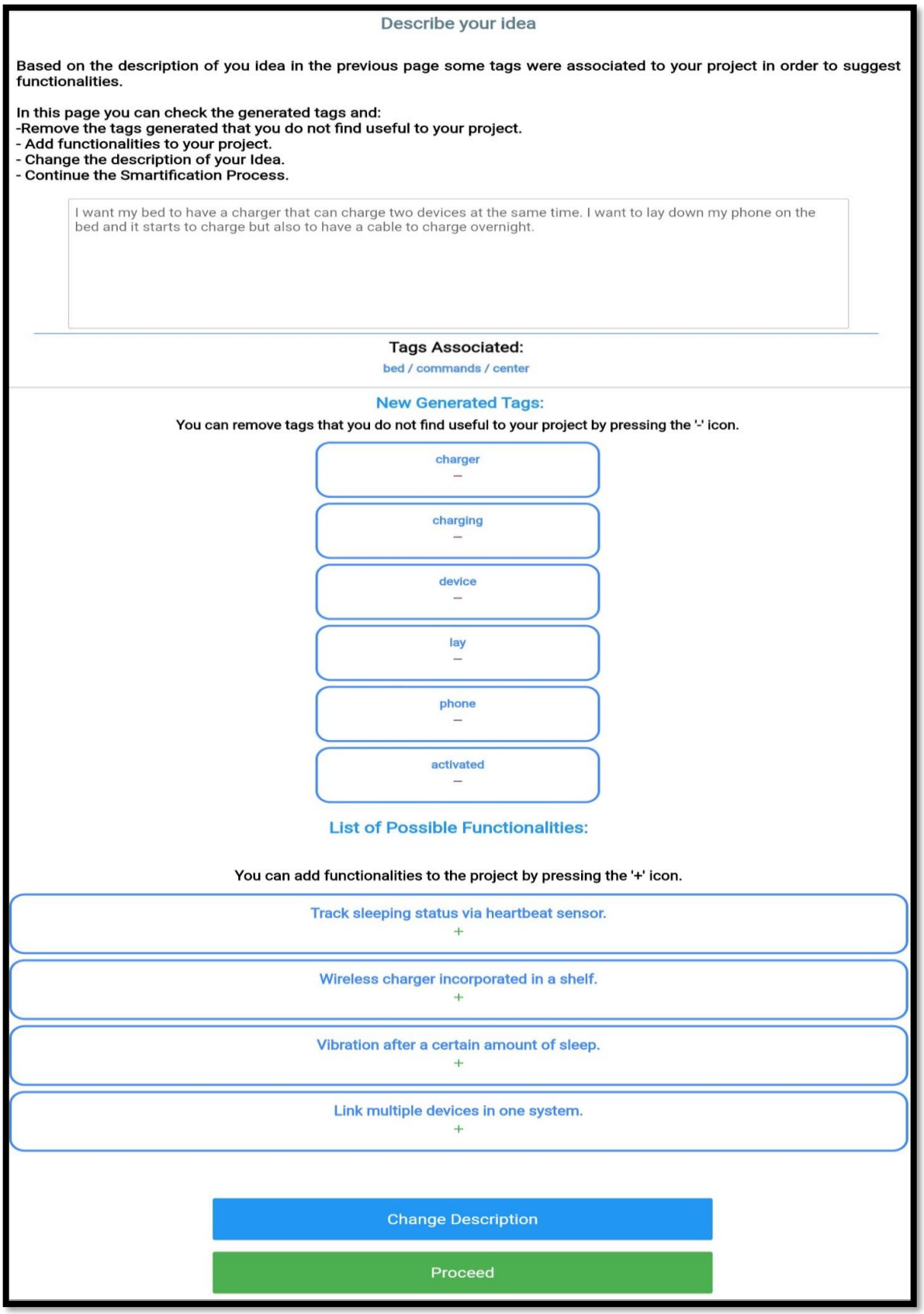


Figure 5.28 - Step 5.A interface 2/2.

The user has multiple options in this interface, as described at the top. It is possible to:

- Remove the newly generated tags.
- Add more functionalities to the project
- Change the idea description, which consequently changes the new tags generated and the suggested functionalities, by clicking "Change Description". This leads the user to the previous interface.
- Continue with the smartification process by clicking "Proceed", which leads the user to Step 6 – Third Question Page.

When Bob arrives at this interface, he cleans up the newly generated tags, leaving only the ones present in Figure 5.28 and then clicks "Proceed", advancing to Step 6.

5.5.11 Step 6 - Third Question Page

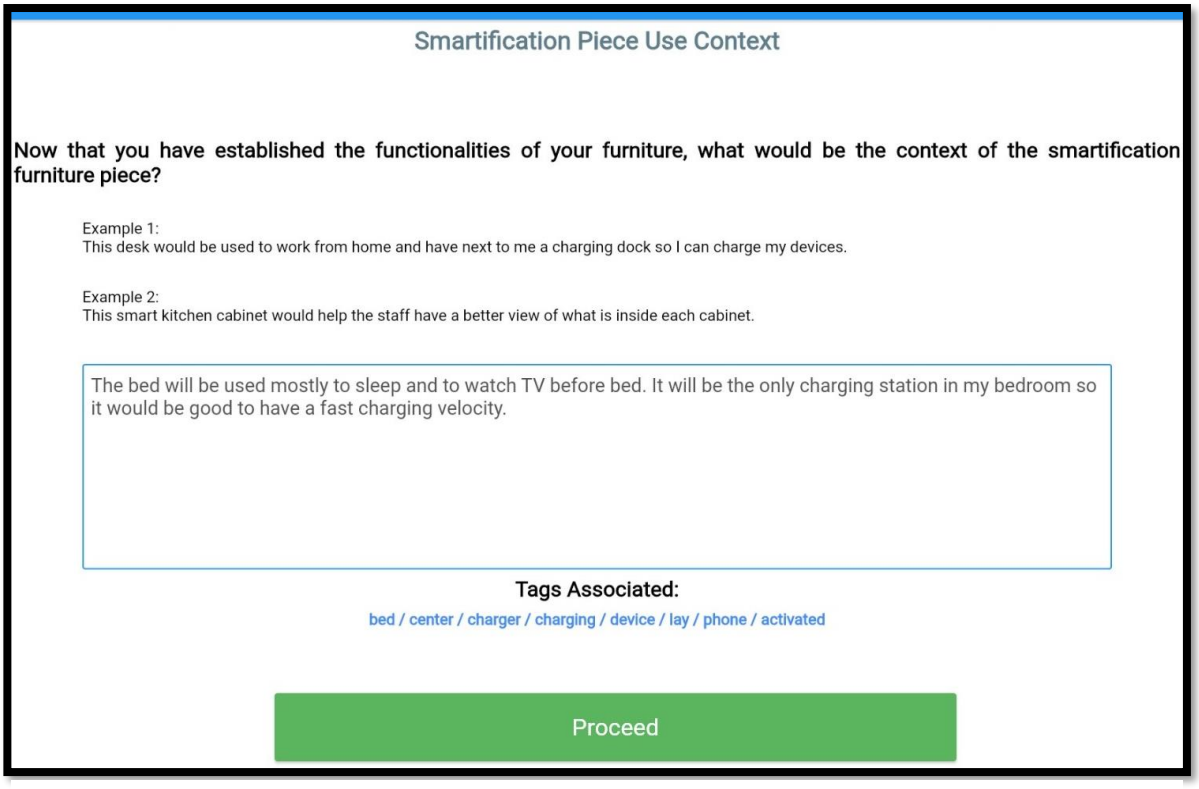


Figure 5.29 - Step 6 interface 1/2.

Following the same structure as Step 3 - First Question Page and Step 5.A - Second Question Page, the interface at the top has a question regarding the smartification piece utilization context, two answer examples, a text box, the tags associated with the project, and the "Proceed" button. This interface is the third and last question to the user before sending all the

information about the smartification project to the other system directly involved in the smartification process. As with all the answer examples, the sentence structure was studied to subconsciously influence the user in their answer. This was done to prepare the tagging system to receive the sentences a certain way and quickly identify the correct tags, as previously mentioned.

Bob checked the examples, wrote his smartification piece utilization context, and clicked "Proceed". This triggered the tagging system and instantly showed a loading message on the screen. Once the tagging system process finishes tagging all the information, the interface is updated, showing a different interface but maintaining the user on the same page. The interface presented is shown below.

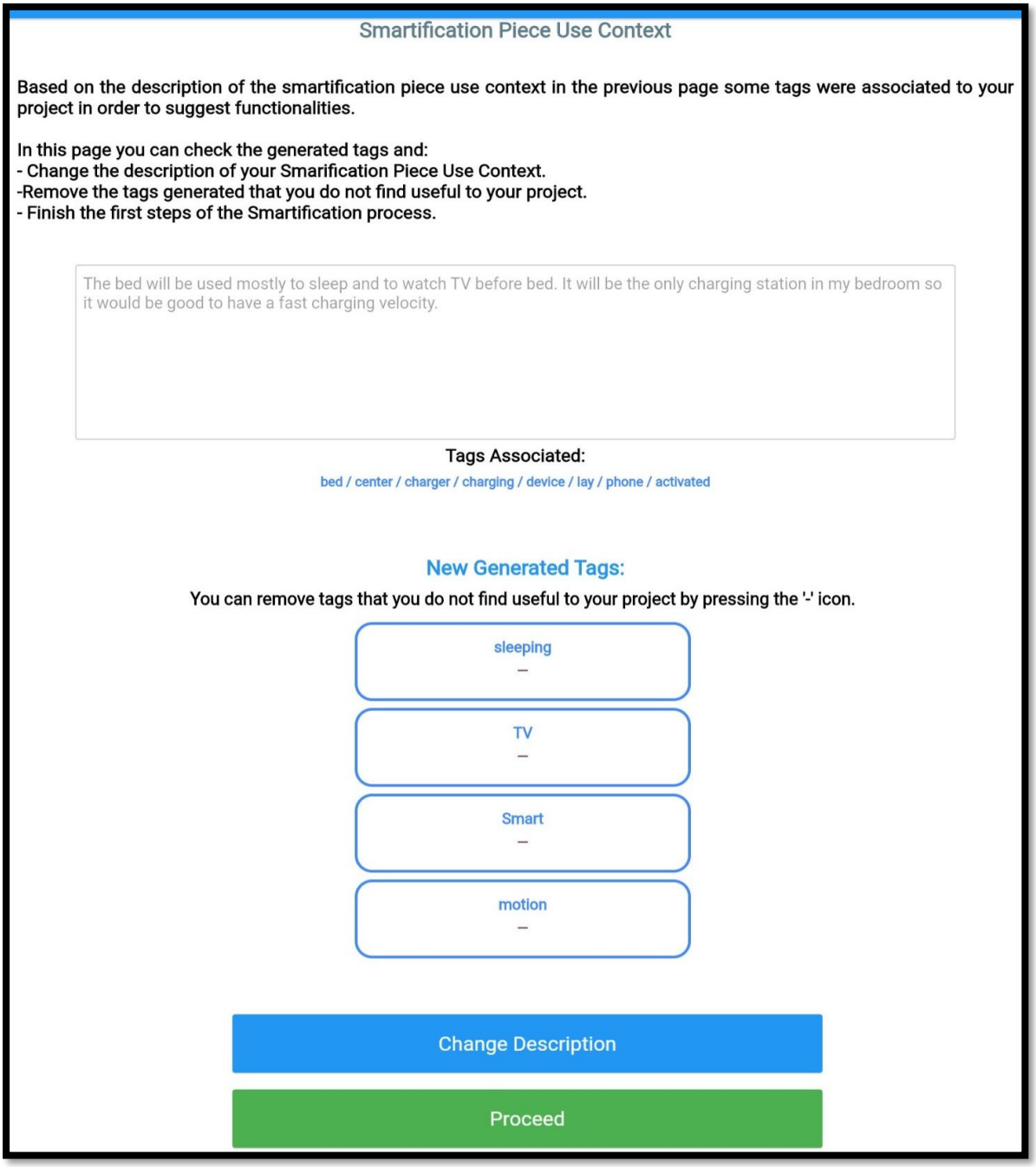


Figure 5.30 - Step 6 interface 2/2.

This is the last interface where the work of the tagging system can be seen. The tags are generated from the user's answers, and the possibility to choose which ones are adequate for the smartification project is enabled by the user. Finally, the available options are presented at the top of the interface. As in Step 3 - First Question Page and Step 5.A - Second Question Page, the user can either change the answer by clicking "Change Description" which leads to the

previous interfaces, or continue the smartification process by clicking the "Proceed" button, which leads to Step 7 – Info Page.

Bob removed the unwanted tags for his project, maintaining the ones shown in Figure 5.30*, and then clicked on the "Proceed" button.*

5.5.12 Step 7 - Info Page

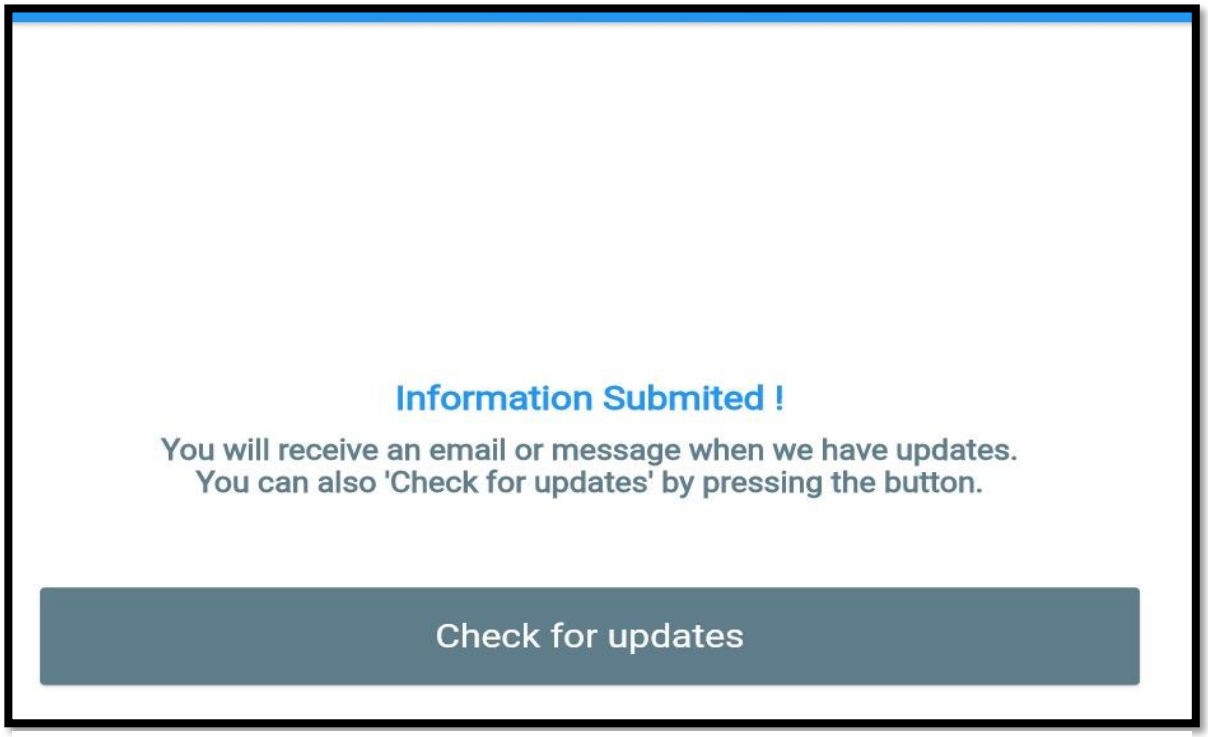


Figure 5.31 - Step 7, 9.A and 11.A and interface.

This interface is shared for steps 7, 9.A and 11.A. It was created as the middle step between the different stages of a smartification project. As explained before, in some parts of the process, the user will have to wait for information from the other systems, being at this point that this interface enters. A message is displayed in the middle of the interface, followed by the "Check for Updates" button. This button can be clicked to check the project status. This interface will show up every time the project state, previously referenced in the sub-chapter Project State, has the values 1, 2, 4, 5, 6, 7, 8, 10, or 12.

Following the Bob example, once the other systems come up with a solution and this solution is inserted in the database, the project state will be updated, and Bob will go to Step 8 – Proposed Solution Page.

5.5.13 Step 8 - Proposed Solution Page

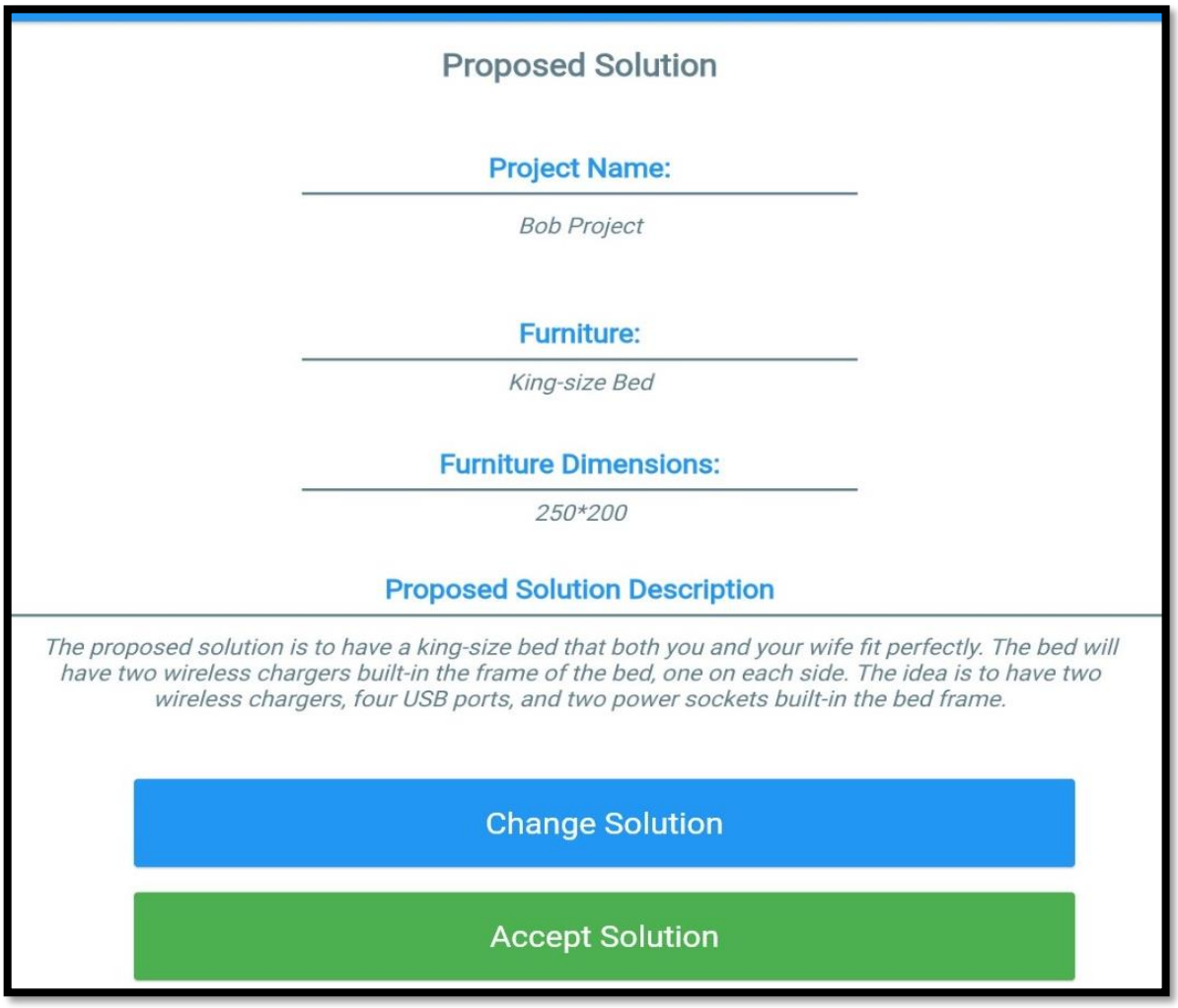


Figure 5.32 - Step 8 interface.

Based on the user's answers and the functionalities added to the project by the user, the Prototyping System, the Quality Assurance System, and other Smartification experts work together to propose a solution. In this interface, the proposed solution is shown. Here, the user can check several project details, like the furniture and the furniture dimensions. These details can be changed if all smartification experts consider them necessary. This interface also displays the description made about the proposed solution. This interface has two options, request changes about the proposed solution by clicking the "Change Solution" button or accept the proposed solutions and continue the smartification process by clicking "Accept Solution". The "Change Solution" button leads the user to Optional Step 9.B - Change Solution Page, while

the "Accept Solution" leads the user to Step 9.A – Info Page. Since Step 9.A – Info Page has the same interface as Step 7 - Info Page, this page will not be presented in this dissertation.

Bob checked all the project details and read the proposed solution description, founding some changes he would like to inform. Then, since he wanted to change the solution, Bob clicked on "Change Solution" button leading him to Optional Step 9.B – Change Solution Page.

5.5.14 Optional Step 9.B - Change Solution Page

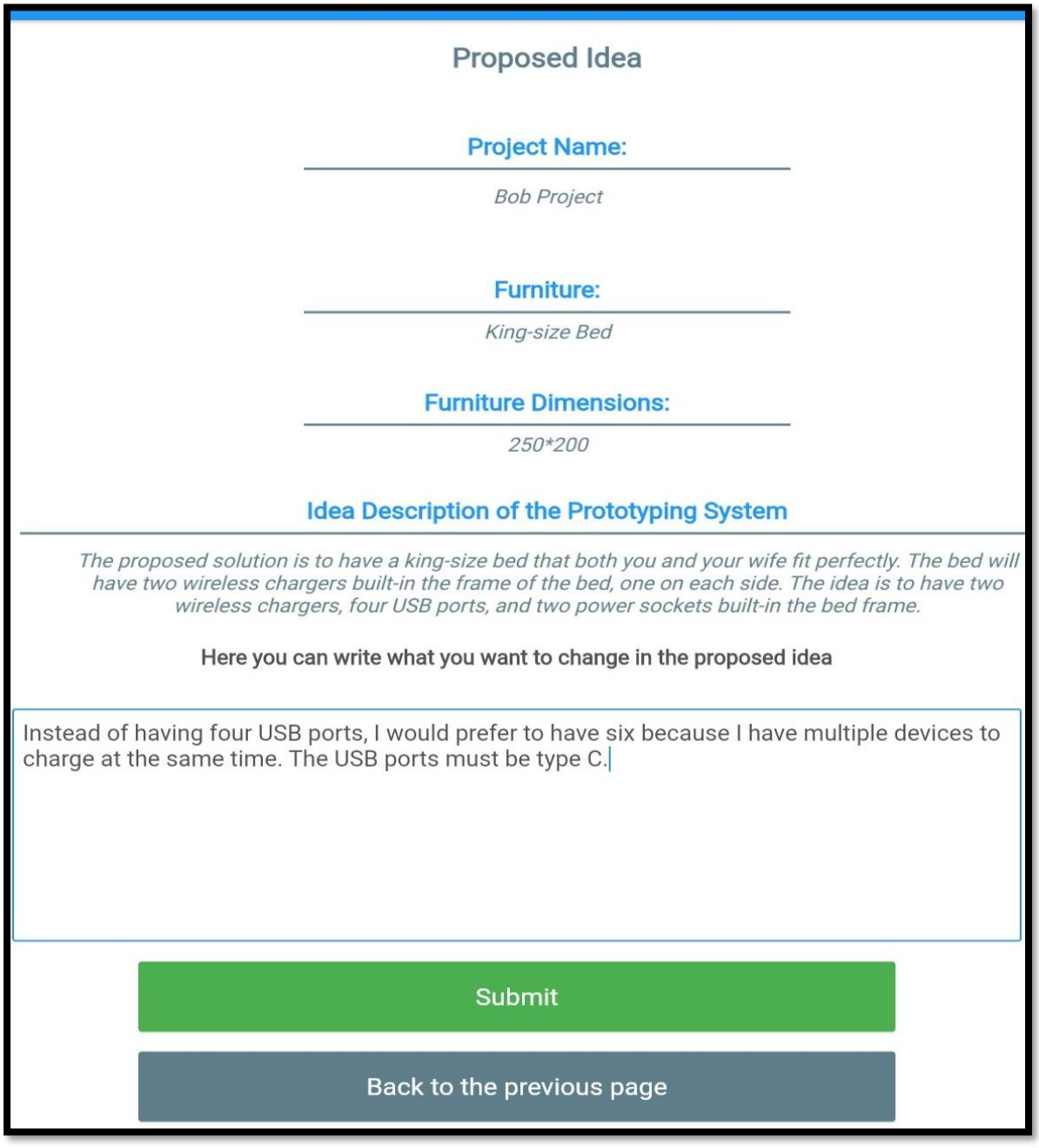


Figure 5.33 - Optional Step 9.B interface.

This interface follows the same structure as Step 8 - Proposed Solution Page, which allows a similar user experience throughout the smartification process. Project details and the proposed solution description are visible as in the interface in Step 8, with the addition of a text box where the user can write the requested changes regarding the proposed solution. The only way to get to this interface is through the "Change Solution" button present in the interface of

Step 8. To return to Step 8, the customer can click on the "Back to the previous page" button. However, if requesting changes is really what the user wants, those changes must be written in the textbox and then press the "Submit" button. This button will lead the customer to Step 7 – Info Page, where the smartification experts will validate the requested changes and propose a new solution to the user, which can be seen again in Step 8. The process will stay in this loop until the user is happy with the solution description and presses the "Accept Solution" button in Step 8.

Bob had some ideas that, for him, were mandatory for his smartification project. He wrote his ideas, which can be seen in Figure 5.33, and clicked the "Submit" button. The interface reacted, and he was redirected to Step 7, where the smartification experts evaluated the requested changes. Those changes were considered, and a new solution was proposed, taking Bob to Step 8 again. With the proposed solution updated with Bob's requested changes, he clicked "Accept Solution," and the smartification process entered a new stage, taking Bob to Step 9- Info Page. Once again, the interface of Step 9 is the same as that of Step 7. At this stage, the smartification experts started to create a prototype. When that first iteration of the prototype was finished, the project state changed, leading to a new step in the frontend, Step 10 – Solution in Development Page.

5.5.15 Step 10 - Solution in Development Page

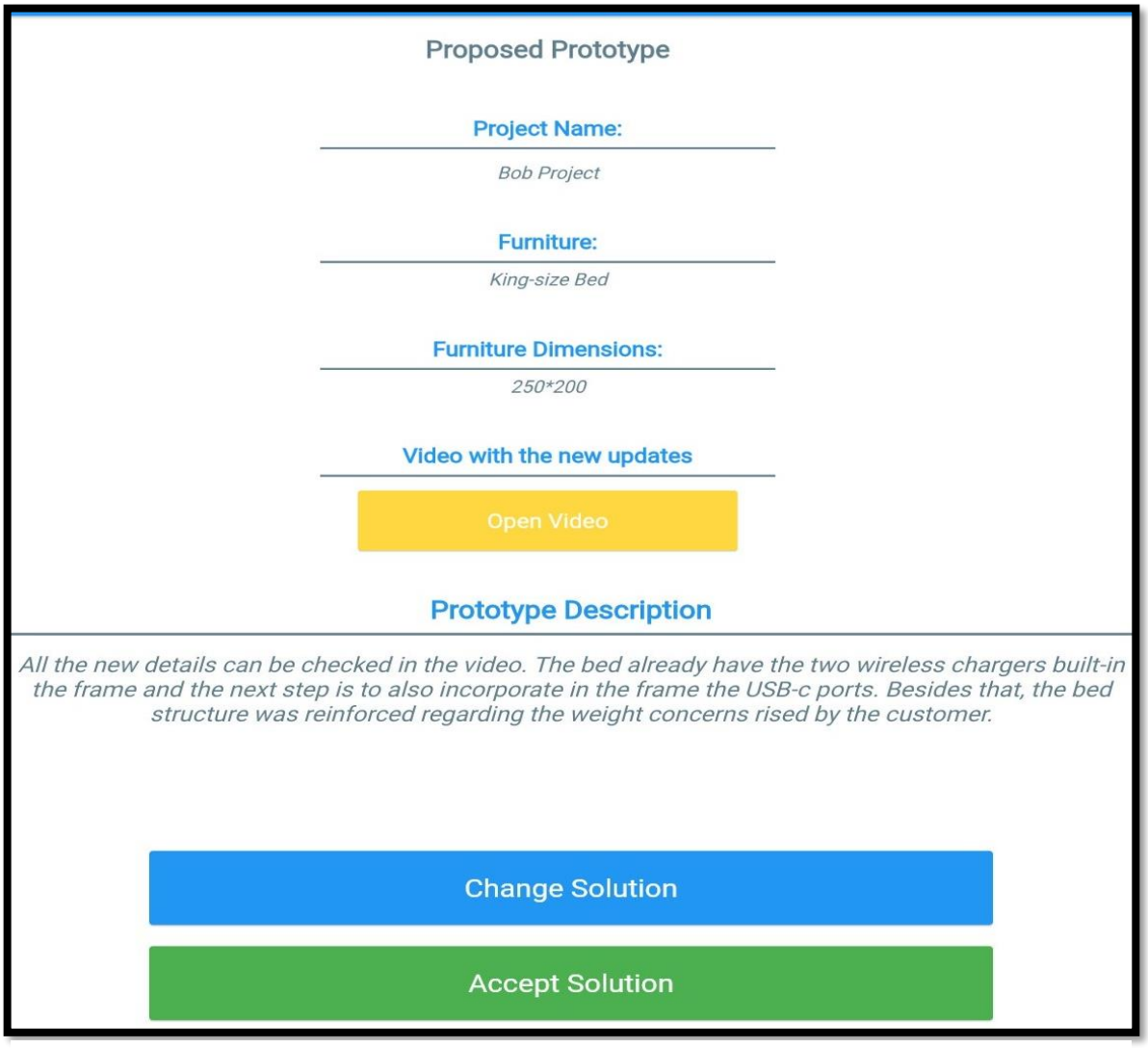


Figure 5.34 - Step 10 interface.

This interface will be shown several times to the customer because of the necessary iterations to involve the customer in the development process. This step aims to show the latest updates of the smartification process to the customer via text and video, enabling the customer to accept or request changes in the prototype. As aforementioned, some interfaces follow the same structure to ease and facilitate the interaction between the user and the GUI. The interface in this step is no exception and follows the structure of Step 8 - Proposed Solution Page with only one change besides the titles. The main difference between this interface and the Step 8 interface is the possibility of pressing a new button that opens a video showing the current state of the prototype. In this interface, it is also possible to see the project details, such as the project name, furniture, and dimensions, as well as the prototype description. The user

has two options: request changes in the prototype by pressing "Change Solution" button or accept the prototype's current state by pressing the "Accept Solution" button. The "Change Solution" button leads the user to Optional Step 11.B – Change Solution Page, which follows the same structure as Optional Step 9.B - Change Solution Page. Optional Step 11.B will not be presented because it has the same objective as Optional Step 9. The only difference is that the smartification project is in the prototype stage instead of the proposed solution stage. The "Accept Solution" button leads the user to Step 11.A – Info Page, that has the same content as Step 7 - Info Page.

Since this step will happen multiple times during prototype development, not all of the iterations will be shown. Bob started by checking the video and the prototype and then requested a change in the solution by pressing the "Change Solution" button, leading him to Optional Step 11.B – Change Solution Page. Then, Bob raised some concerns about the strength of the bed because he and his wife have around 200kg of combined weight, submitting that information in Optional Step 11.B, which led him to step 9.A. Next, the smartification experts updated the prototype description and the video, which can be seen in Figure 5.34. Here, Bob saw the video, read the description, accepted the solution to his concerns, and clicked "Accept Solution" to proceed with the smartification process, leading him to Step 11.A – Info Page. This interface will show up until the prototype is considered finished by both the Prototyping System and the user. Once that happens, one last confirmation is needed, which can be done in Step 12 – Final Approve Page.

5.5.16 Step 12 - Final Approval Page

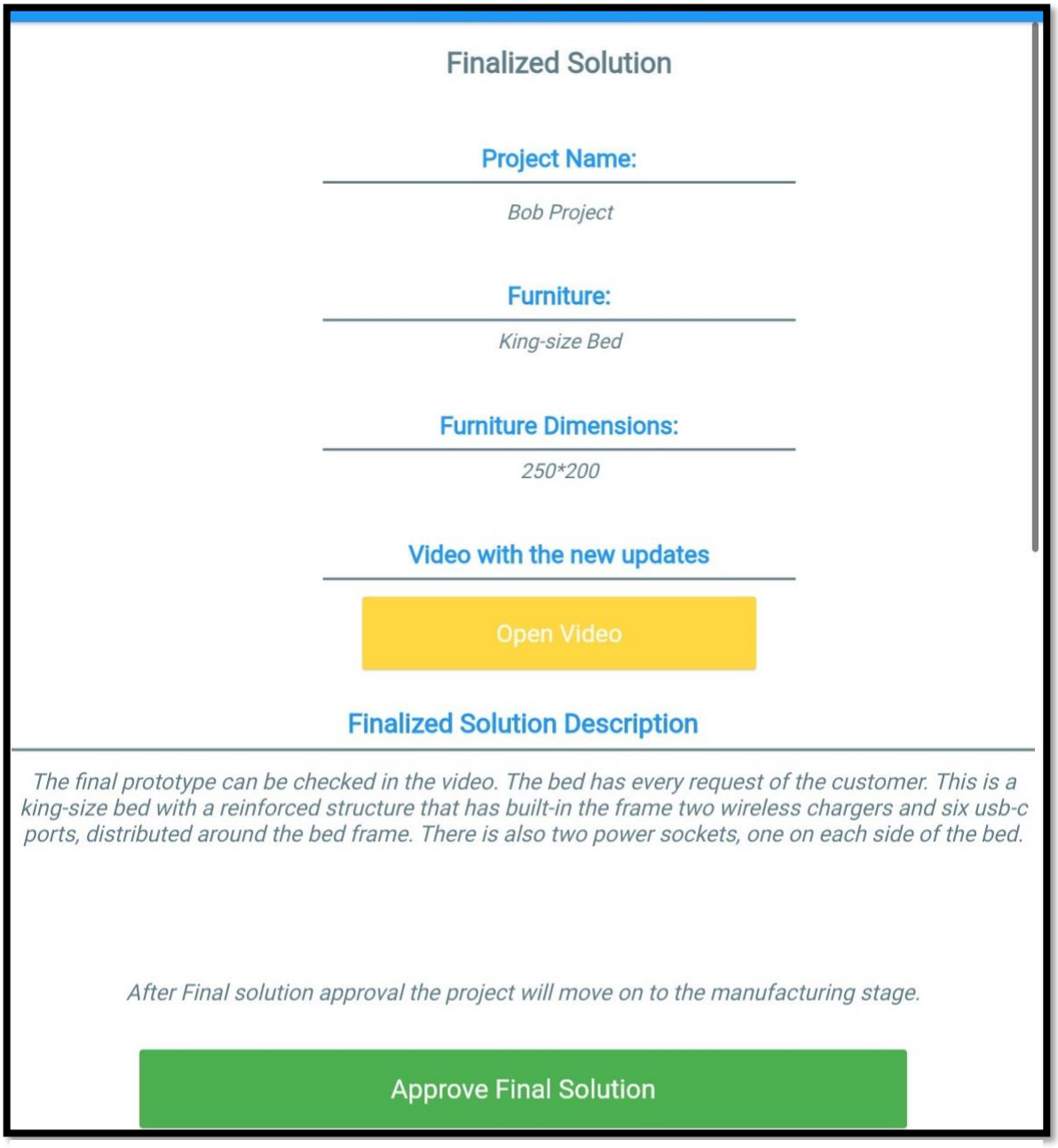


Figure 5.35 - Step 12 interface.

This confirmation page lets the customer check all final prototype details and watch a video about the prototype itself. In this step, the interface has the same structure as in Step 8 - Proposed Solution Page and Step 10 - Solution in Development Page, aiming to facilitate the customer utilization of the GUI. The interface will show the project information, the latest solution description, the button that opens up the video, and information before the "Approve Final Solution" about the next steps of the smartification process. This is the last developed

interface regarding the smartification service presented in this dissertation, however, clicking "Approve Final Solution" will lead the user to a survey created in Google Forms about the smartification project process. Even though it is not a developed interface, it is considered the last step of the smartification service. As explained in the survey, these answers will be used to improve the smartification project through the adjustment and improvement of aspects that users considered that can be improved.

At this stage, Bob already knew what to expect from the video and the description, so in this interface, he confirmed that information and clicked "Approve Final Solution". Bob was redirected to the survey, where he answered all the questions, providing feedback to improve the smartification process.

5.6 Results

The application provides helpful suggestions by extracting valuable information from the user's sentences. The Interface sub-chapter provides a detailed explanation of almost all of the developed interfaces. It also presents an example of a customer using the interface. Through this example, it is possible to assess the application performance and the practical results obtained by the customer.

Bob started by answering the first question, helped by the answer examples provided by the application. From this point forward, the application will constantly extract information from the user sentences, as shown in Figure 5.21. The tags are the result of the extracted information from Bob's sentences. Even though the smartification need is asked in the second question, the tags from the first question are used to start providing suggestions. These suggestions can be checked in two different steps, Optional Step 5.B.1 - Suggested Projects (providing similar projects) and Optional Step 5.C- Suggested Smartification Functionalities (displaying relevant smartification functionalities). However, these functionalities are also suggested in Step 5.A - Second Question Page, before and after Bob answers about the smartification need. When Bob answers the second question, new tags will be generated, and consequently, the suggested functionalities are updated, typically increasing the list size of suggested functionalities.

The suggestions provided by the smartification service greatly help the user and dramatically increase the ease of the smartification process. This is proven in Optional Step 5.C- Suggested Smartification Functionalities, where Bob adds functionality from the list of suggested functionalities even before providing his smartification need to the system. Moreover, the GUI was

proven effective during the different stages of the smartification process. Through the GUI, constant interaction between the customer and the prototyping system is allowed, thus facilitating the user's active voice in the furniture idealization and prototyping process.

The obtained results from the development of the application were expected. It was possible to extract information from the user sentences and provide meaningful help through the suggestion of smartification functionalities and the suggestion of similar projects. These results elevate the furniture smartification process to another level by transforming this process, from the customer's view, into a more autonomous and easier one.

By categorizing information, it was possible to differentiate functionalities and use that difference to adjust the suggested functionalities for each project. Besides that, the development of a GUI considering the guidelines for developing accessible and usable interfaces resulted in the user's guidance through the smartification process. To analyze the ease the interface introduces to the user in describing smartification needs, ten people were asked to answer the questionnaire and then use the interface to express their smartification needs simulating the smartification process. This group of people had the ages between 26 and 40. 66% of this group graduated in technology, while 34% had no related studies. The actions of the other systems present during the smartification process were simulated to enable the test group to achieve the final steps of the smartification process without really creating smart furniture solutions. When asked if they liked the smartification process with an evaluation between 1-5, 10% gave a 3, 60% considered a 4, and 30% a 5, with all respondents considering the smartification process not a hard process, as can be seen in Annex A. From that survey results, it was also possible to conclude that the tags generated, the suggestion of functionalities and projects, and the capability of achieving a suitable smart furniture solution were among the most liked features of this application. However, from several suggestions to improve this application, the most common were miss-generated tags and a lack of projects in the project's catalog. An approach to improve the miss-generated tags will be discussed in the Future Work subchapter. The lack of projects could be resolved by increasing the number of projects stored in the database, which will eventually happen if this service starts being used for the INEDIT project production environment.

All users who tested this application found it quite interesting that it is possible to actively participate in the various stages of furniture smartification. Although this is not the main focus of the work developed, this was considered to be present during the application development.

This possibility follows the same line of thought as the concept of co-creation referenced by the INEDIT project.

6

Conclusion

The dissertation aimed to create an intelligent relation tagging system for a smartification-based dashboard. The hypothesis proposed, "If a system can extract information using semantic deconstruction, then it is possible to help customers to find the right smart furniture solution and use that information to enrich a knowledge base (KB) for future use cases.", was supported with the creation of a Smartification Service. This intelligent relational tagging system, composed of two main parts, frontend, and backend, proved to help achieve a suitable smart furniture solution. The developed system extracted valuable information from the customer sentences through semantic deconstruction to help the customer through the smartification process. Additionally, the extracted information was treated to populate a knowledge base to be used for future utilization cases. Therefore, the developed tagging system that handles the information extraction can be used in further applications, not being restricted to smart furniture projects.

The work developed in this dissertation is usable by everyone that wants to change a piece of existing furniture into a piece of smart furniture. Furniture smartification is a relatively recent topic, and not everyone has the creativity and imagination to create functionalities for furniture. Therefore, the significant advantage of using the developed work is that this service helps the user go through a process that can be difficult for most people.

Taking advantage of others' ideas often leads to faster innovation and more creative ideas. The work developed enhances this theory, with the users reaching more detailed and creative ideas based on suggested functionalities and similar projects. As exemplified in the Interface subchapter, the user can start the smartification process with preconceived ideas; throughout the process, these ideas might change, and new ones emerge. This is mainly due to the user seeing suggested functionalities and interacting with other previous smartification projects completed with this service.

Changing existing furniture into a smart piece can significantly impact most people's lives. However, it can be a life changer for the elderly and the disabled since it allows the different types of users to have furniture adapted to their needs with specific functionalities. Once again, these adaptations can be a solution hard to achieve when thinking alone, and that is what the developed work in this dissertation achieves, providing the customer with helpful suggestions of functionalities.

The research work done in this dissertation is the first step toward an area that can and should be further investigated. The co-creation of smart furniture is still a recent area, but this technology field can significantly impact the world by providing furniture adjusted to all types of people's needs.

In summary, the scientific knowledge achieved during this dissertation can be considered relevant because of the effective development of an intelligent relational tagging system for a smartification-based dashboard. Therefore, this tool may achieve a sociological relevant status due to its life-enhancing potential by helping adapt existing furniture to any user requirements. Furthermore, this tool can be helpful in an increasingly technological world due to the ability to potentially transform any furniture into smart furniture independently of the customer's knowledge.

6.1 Future Work

Several paths can be taken to improve the current state of the work developed in this dissertation. The application presented is composed of multiple systems that can be improved. Therefore, various scenarios can be considered for each system to improve the work.

Regarding the database, some aspects were considered in the design phase but not explored in the application's development phase. As it is possible to see in Figure 5.6, the Tagging table has a field regarding the number of *occurrences* of a tag. Despite the application incrementing this number with each generated tag, almost nothing is done with this information. *Occurrences* can be used to improve the project's catalog accuracy by using that number to rank the most critical projects regarding the user project. Together with *occurrences*, the *type* field in the tagging table can be used to improve the accuracy of the suggested functionalities. Considering both these fields (*occurrences* and *type*), better functionalities can be suggested in the different interfaces. With these changes, the functionalities suggested, and the project catalog will be even more appropriate to the users' needs. Still, regarding the database, improving

the amount of smartification projects in the database might contribute to better suggestions. This improvement would have a considerable impact in the early days of the application, at a stage where there would be few completed projects to serve as an example. This could be done by having more people answer the surveys and creating more smartification projects.

Regarding the tagging system or backend, some aspects can be addressed to improve the quality of this system. For example, improving the list of words by restricting unnecessary words regarding furniture smartification projects would result in a more efficient tagging system. The process in the developed work was analyzing the tags with the most occurrences and checking if they are helpful regarding smartification projects. One way to improve this process is by considering the context of the sentences. Thus, despite being tags that fall within the criterion of proper tags, they would be ignored for being useless. This enhanced feature could be implemented by adding a more efficient and intelligent NLP tag-generating system, like GPT-3 mentioned in the State-of-the-Art chapter. This system would improve the semantic deconstruction efficiency of the user sentences, achieving more accurate tags and consequently resulting in better suggestions. However, GPT-3 only gained more relevance during the mid-stages of the development done in this dissertation, with the required software presenting several issues related to uncertain availability due to access authorizations during the effective timeline of this work. This dissertation can be considered a proof of concept of the work that can be done and improved regarding extracting the user needs with the semantic deconstruction of sentences.

Some adjustments can be made to have more accessibility and usability features for the GUI part of the application. This will ease the user experience through the GUI, consequently improving the effectiveness of all the other systems and reaching more people. Another useful feature that could be added to the GUI is the dynamic examples for the interfaces with questions. This could be done by checking the user project, retrieving similar projects, and consequently withdrawing the users' answers to these questions, using them as examples.

The intention of writing and publishing a paper with the results achieved in this dissertation exists.

The results of this dissertation will be presented to the INEDIT project, which will deliberate about implementing this service in their smartification system. If accepted, the intelligent relational tagging system developed in this dissertation will go live around November 2022, helping customers in their furniture smartification process in real-life scenarios. This can lead to

more feedback to improve the current system. However, some initial approaches on how to improve were already discussed in this section.

This work is a foundation regarding furniture smartification guidance that can be improved by implementing the extra features suggested in this section and others.

REFERENCES

[1] K. Ashton, "RELA TED C ONTENT RFID-Powered Handhelds Guide Visitors at Shanghai Expo Despite Sluggish Growth, Taiwan's RFID Industry Remains Committed Mobile RTLS Tracks Health-care Efficiency RFID Journal LIVE! 2010 Report, Part 2 That 'Internet of Things' Thing," 2010. Accessed: Feb. 12, 2021. [Online]. Available: http://www.rfidjournal.com/article/print/4986.

[2] D. Paret and P. Crégo, "Definitions and Position," in *Wearables, Smart Textiles and Smart Apparel*, Elsevier, 2019, pp. 5–8.

[3] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4. IEEE Computer Society, pp. 2233–2243, Nov. 01, 2014, doi: 10.1109/TII.2014.2300753.

[4] M. Pérez-Montoro and L. Codina, "The Basis of Navigation," in *Navigation Design and SEO for Content-Intensive Websites*, Elsevier, 2017, pp. 11–63.

[5] "RUN: Semantic adaptability for the systems interoperability." https://run.unl.pt/handle/10362/17157 (accessed Jan. 22, 2021).

[6] "(PDF) Taxonomy construction techniques - issues and challenges." https://www.researchgate.net/publication/260318181_Taxonomy_construction_techniques_-_issues_and_challenges (accessed Feb. 28, 2022).

[7] "[PDF] TAXONOMY CONSTRUCTION TECHNIQUES – ISSUES AND CHALLENGES | Semantic Scholar." https://www.semanticscholar.org/paper/TAXONOMY-CONSTRUCTION-TECHNIQUES-–-ISSUES-AND-Rao/ff89917911da2a1f7a314bb1b2f033e8ec2bad0b (accessed Mar. 05, 2022).

[8] C. Zhang *et al.*, "TaxoGen: Unsupervised Topic Taxonomy Construction by Adaptive Term Embedding and Clustering," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 2701–2709, Dec. 2018, doi: 10.48550/arxiv.1812.09551.

[9] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen, "Short text conceptualization using a probabilistic knowledgebase," *IJCAI Int. Jt. Conf. Artif. Intell.*, pp. 2330–2336, 2011, doi: 10.5591/978-1-57735-516-8/IJCAI11-388.

[10] "Flutter vs. React Native: 5 Reasons Why Flutter Framework is Better." https://www.mantralabsglobal.com/blog/flutter-framework/ (accessed Feb. 27, 2022).

[11] E, M. Martinc, J. Kralj, N. Lavrač, and S. Pollak, "tax2vec: Constructing Interpretable Features from Taxonomies for Short Text Classification," *Comput. Speech Lang.*, vol. 65, Jan. 2021, doi: 10.1016/j.csl.2020.101104.

[12] "TopQuadrant - Enterprise Data Governance — TopQuadrant, Inc." https://www.topquadrant.com/ (accessed Feb. 24, 2021).

[13] "Trends in Enterprise E-discovery from the Corporate Perspective," in *E-discovery: Creating and Managing an Enterprisewide Program*, Elsevier, 2009, pp. 1–18.

[14] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu, "Exploring folksonomy for personalized search," 2008, doi: 10.1145/1390334.1390363.

[15] M. Strohmaier, D. Helic, and R. Kern, "Evaluation of folksonomy induction algorithms," *ACM Trans. Intell. Syst. Technol*, vol. 3, 2012, doi: 10.1145/2337542.2337559.

[16] "[PDF] Construction of Disambiguated Folksonomy Ontologies Using Wikipedia | Semantic Scholar." https://www.semanticscholar.org/paper/Construction-of-Disambiguated-Folksonomy-Ontologies-Tomuro-Shepitsen/c07e0e69928ac275f06ea0ce74d767e1db9f0c5b (accessed Mar. 01, 2022).

[17] J. Chen, B. Qiang, Y. Wang, P. Wang, J. Huang, and J. H. An, "An Optimized Tag Recommender Algorithm in Folksonomy," pp. 47–56, 2014, doi: 10.1007/978-3-662-44980-6_6ï.

[18] A. Hotho, R. Jäschke, C. Schmilz, and G. Stumme, "Information retrieval in folksonomies: Search and ranking," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4011 LNCS, pp. 411–426, 2006, doi: 10.1007/11762256_31.

[19] D. Kowald and E. Lex, "Overcoming the Imbalance Between Tag Recommendation Approaches and Real-World Folksonomy Structures with Cognitive-Inspired Algorithms," 2018, doi: 10.1145/nnnnnnn.nnnnnnn.

[20] "Bismart Folksonomy Text Analytics." https://landing.bismart.com/en/bismart-folksonomy (accessed Feb. 17, 2021).

[21] "Folksonomy - Bismart: Big Data & Business Intelligence Company." https://bismart.com/en/business-intelligence-solutions/folksonomy/ (accessed Feb. 17, 2021).

[22] "What's the Difference Between Folksonomy and Taxonomy?" https://blog.bismart.com/en/difference-between-folksonomy-and-taxonomy (accessed Feb. 17, 2021).

[23] J. Zhao, Q. Zhang, Q. Sun, H. Huo, Y. Xiao, and M. Gong, "FolkRank++: An Optimization of FolkRank Tag Recommendation Algorithm Integrating User and Item Information," *KSII Trans. Internet Inf. Syst.*, vol. 15, no. 1, pp. 1–19, Jan. 2021, doi: 10.3837/TIIS.2021.01.001.

[24] J. Trant, "Studying social tagging and folksonomy: A review and framework," *J. Digit. Inf.*, 2009.

[25] O. Nov, M. Naaman, and C. Ye, "What drives content tagging: The case of photos on flickr," 2008, doi: 10.1145/1357054.1357225.

[26] M. Marinhas da Silva and D. João Miguel da Costa Magalhães, "Automated Image

Tagging through Tag Propagation," 2011.

[27] D. Kumawat and V. Jain, "POS Tagging Approaches: A Comparison," 2015.

[28] M. Banko and R. C. Moore, "Part of speech tagging in context," *COLING 2004 - Proc. 20th Int. Conf. Comput. Linguist.*, 2004, doi: 10.3115/1220355.1220435.

[29] J. Gao and M. Johnson, "A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers," *EMNLP 2008 - 2008 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf. A Meet. SIGDAT, a Spec. Interes. Gr. ACL*, pp. 344–352, 2008, doi: 10.3115/1613715.1613761.

[30] "Part of Speech Tagging Using Statistical Approach for Nepali Text." https://publications.waset.org/10006246/part-of-speech-tagging-using-statistical-approach-for-nepali-text (accessed Mar. 01, 2022).

[31] S. Wang, T. Wang, X. Mao, G. Yin, and Y. Yu, "A Hybrid Approach for Tag Hierarchy Construction," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10826 LNCS, pp. 59–75, 2018, doi: 10.1007/978-3-319-90421-4_4.

[32] "MonkeyLearn - Text Analysis." https://monkeylearn.com/ (accessed Feb. 24, 2021).

[33] "PoolParty Semantic Suite - Your Complete Semantic Platform." https://www.poolparty.biz/ (accessed Feb. 23, 2021).

[34] "What is a Knowledge Base and Why Do You Need One?" https://www.knowledge-management-tools.net/what-is-a-knowledge-base.php (accessed Feb. 18, 2021).

[35] "What Is a Knowledge Base and Why Do I Need One." https://document360.com/blog/what-is-a-knowledge-base/ (accessed Feb. 18, 2021).

[36] "A Guide to Knowledge Based Systems | Smartsheet." https://www.smartsheet.com/knowledge-base-systems-and-templates (accessed Feb. 20, 2021).

[37] "What is a knowledge base | Atlassian | Atlassian." https://www.atlassian.com/itsm/knowledge-management/what-is-a-knowledge-base (accessed Feb. 18, 2021).

[38] "Knowledge-Based System | KMS Lighthouse." https://www.kmslh.com/glossary/knowledge-based-system/ (accessed Feb. 20, 2021).

[39] "Data Modelling - Understanding Tools and Techniques Involved." https://www.xenonstack.com/insights/data-modelling (accessed Mar. 06, 2022).

[40] "What Is Data Modeling? - Definition from SearchDataManagement." https://www.techtarget.com/searchdatamanagement/definition/data-modeling (accessed Mar. 06, 2022).

[41] "Self Service Knowledge Base Software Features - Document360."

https://document360.com/features/ (accessed Feb. 20, 2021).

[42] "Knowledge base software that's interactive | Stonly." https://stonly.com/product/knowledge-base-software (accessed Feb. 20, 2021).

[43] N. R. Shadbolt and P. R. Smart, "Knowledge Elicitation: Methods, Tools and Techniques." Accessed: Dec. 16, 2020. [Online]. Available: http://www.amazon.co.uk/Evaluation-Human-Work-Fourth-.

[44] "Knowledge Elicitation and Its Techniques | Artificial Intelligence." https://www.engineeringenotes.com/artificial-intelligence-2/expert-systems/knowledge-elicitation-and-its-techniques-artificial-intelligence/35594 (accessed Dec. 16, 2020).

[45] "Knowledge Elicitation Methods." https://web.cs.wpi.edu/~jburge/thesis/kematrix.html (accessed Feb. 21, 2021).

[46] A. Hart, "Knowledge elicitation: issues and methods," *Comput. Des.*, vol. 17, no. 9, pp. 455–462, Nov. 1985, doi: 10.1016/0010-4485(85)90293-3.

[47] "Build more accessible apps | Android Developers." https://developer.android.com/guide/topics/ui/accessibility (accessed Mar. 07, 2022).

[48] A. Przepiórkowski and M. Ogrodniczuk, Eds., "Advances in Natural Language Processing," vol. 8686, 2014, doi: 10.1007/978-3-319-10888-9.

[49] E. Liddy, "Natural Language Processing," *Sch. Inf. Stud. - Fac. Scholarsh.*, Jan. 2001, Accessed: Mar. 07, 2022. [Online]. Available: https://surface.syr.edu/istpub/63.

[50] "NLTK :: Natural Language Toolkit." https://www.nltk.org/ (accessed Mar. 07, 2022).

[51] "What is GPT-3 and why is it so powerful? | Towards Data Science." https://towardsdatascience.com/what-is-gpt-3-and-why-is-it-so-powerful-21ea1ba59811 (accessed Sep. 17, 2022).

[52] "What Is GPT-3 And Why Is It Revolutionizing Artificial Intelligence?" https://www.forbes.com/sites/bernardmarr/2020/10/05/what-is-gpt-3-and-why-is-it-revolutionizing-artificial-intelligence/?sh=15de8235481a (accessed Sep. 18, 2022).

[53] "Introduction to Web Accessibility | Web Accessibility Initiative (WAI) | W3C." https://www.w3.org/WAI/fundamentals/accessibility-intro/#what (accessed Nov. 04, 2021).

[54] "The Four Principles of Accessibility." https://www.willowtreeapps.com/craft/the-four-principles-of-accessibility (accessed Nov. 04, 2021).

[55] "Principle 1: Perceivable | Understanding Web Accessibility Course." https://courses.idrc.ocadu.ca/understandinga11y/1_perceivable.html (accessed Nov. 09, 2021).

[56] "Principle 2: Operable | Understanding Web Accessibility Course."

https://courses.idrc.ocadu.ca/understandinga11y/2_operable.html (accessed Nov. 09, 2021).

[57] "Principle 3: Understandable | Understanding Web Accessibility Course." https://courses.idrc.ocadu.ca/understandinga11y/3_understandable.html (accessed Nov. 09, 2021).

[58] "Principle 4: Robust | Understanding Web Accessibility Course." https://courses.idrc.ocadu.ca/understandinga11y/4_robust.html (accessed Nov. 09, 2021).

[59] S. Abrahão and C. Cachero, "Web usability and accessibility MULTIPLE: Multimodeling Approach for Quality-Aware Software Product Lines View project," *Artic. J. Web Eng.*, 2008, Accessed: Nov. 10, 2021. [Online]. Available: https://www.researchgate.net/publication/262234352.

[60] "ISO/TS 20282-2:2013(en), Usability of consumer products and products for public use — Part 2: Summative test method." https://www.iso.org/obp/ui/#iso:std:iso:ts:20282:-2:ed-2:v1:en (accessed Nov. 10, 2021).

[61] "Accessibility overview - Windows apps | Microsoft Docs." https://docs.microsoft.com/en-us/windows/apps/design/accessibility/accessibility-overview (accessed Mar. 07, 2022).

[62] "How to use a framework to plan web accessibility better." https://www.siteimprove.com/blog/how-to-use-a-framework-to-plan-web-accessibility-better/ (accessed Sep. 18, 2022).

[63] "USEFul - A Framework To Automate Website Usability Evaluation (Part 1) - Usability Geek." https://usabilitygeek.com/useful-a-framework-to-automate-website-usability-evaluation-part-1/ (accessed Sep. 18, 2022).

[64] Y. M. Altman, "Graphics and GUI," in *Accelerating MATLAB Performance*, 2020.

[65] "Flutter - Build apps for any screen." https://flutter.dev/ (accessed Feb. 27, 2022).

[66] "JSON." https://www.json.org/json-en.html (accessed Sep. 21, 2022).

[67] "What Is a Database | Oracle Portugal." https://www.oracle.com/pt/database/what-is-database/ (accessed Sep. 30, 2021).

[68] "Machine Learning (ML) for Natural Language Processing (NLP) - Lexalytics." https://www.lexalytics.com/lexablog/machine-learning-natural-language-processing (accessed Sep. 30, 2021).

[69] "ElephantSQL - PostgreSQL as a Service." https://www.elephantsql.com/ (accessed Sep. 23, 2022).

[70] "What is PostgreSQL? – Amazon Web Services." https://aws.amazon.com/rds/postgresql/what-is-postgresql/ (accessed Oct. 13, 2021).

[71] “Welcome to Flask — Flask Documentation (2.2.x).” https://flask.palletsprojects.com/en/2.2.x/ (accessed Sep. 22, 2022).

[72] “What is HTTP? | Cloudflare.” https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/ (accessed Sep. 22, 2022).

[73] “Glitch: The friendly community where everyone builds the web.” https://glitch.com/ (accessed Sep. 23, 2022).

[74] “• Internet users in the world 2022 | Statista.” https://www.statista.com/statistics/617136/digital-population-worldwide/ (accessed Aug. 29, 2022).

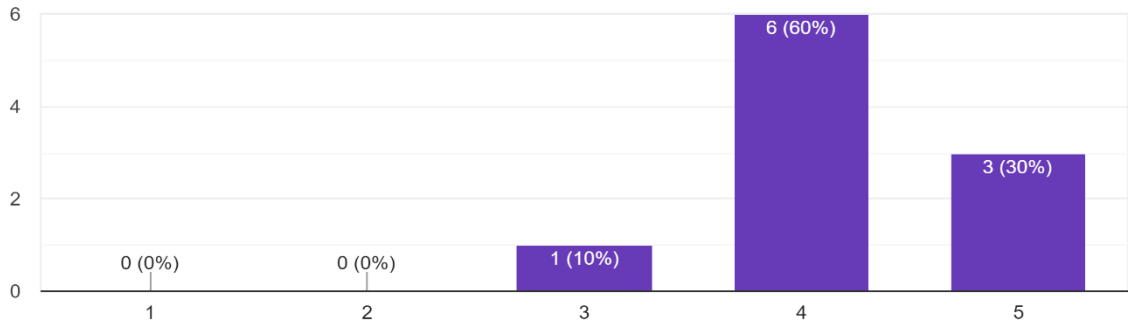
A Annex

Survey Results

Survey Results

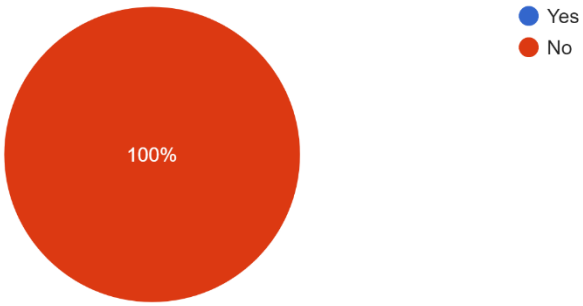
Did you like the smartification the smartification process?

10 respostas



Did you find the smartification process an hard process?

10 respostas



What did you like the least?

10 respostas

- Some generated tags did not make sense
- Lights according to mood: I don't think I would use that functionality. Regardless how I feel, I personally prefer that my home is cosy and with matching colors, so If I were happy I would not be happy anymore if my kitchen turned all green!
- Could have better suggestions
- The tags generated could be better
- Not enough projects
- The interfaces could be better
- The ammount of tags that my project generated
- Projects catalog did not show me any projects

What did you like most about our service?

10 respostas

- The suggestions during the process
- The tags generated
- The projects catalog have similar projects
- The possibility to participate and change the functionalities during the development stages
- The amount of functionalities suggested during the process
- It enabled me to detail my needs
- Functionalities suggestions and the projects catalog
- The generated tags
- It enabled me to achieve my smart furniture

How can we make the smartification process better?

10 respostas

add more and more possible customization

Have more suggested projects

The GUI could be better

Allow to add tags to the project

Increase the number of suggested projects

Do not suggest functionalities that are impossible to achieve for my furniture.

Increase the number of suggestions and improve the interfaces design

Enabling to check the functionalities associated with each tag

Increase the projects catalog