

AGH

AGH University of Science and Technology

Faculty of Computer Science, Electronics and Telecommunications

Engineering Thesis

Design and Implementation of a Website for a Zoological Garden

Author:

Wiktoria Martyńska

Degree programme:

Information And Communication Technologies

Supervisor:

Katarzyna Kosek-Szott, PhD. Eng.

Cracow, 2023

For Japko, the cat, who was no help at all

Contents

1. Introduction	5
1.1. Background Information	6
1.2. Objective	6
1.3. Methodology	6
1.4. Limitations	6
2. Methodology	7
2.1. Interview with Employees	7
2.2. Research Approach	8
2.3. Existing Website Comparison	8
3. Web Development Technologies	10
3.1. React Library	10
3.1.1. Styled Components	11
3.1.2. I18next Framework	11
3.1.3. React-paypal-js Library	12
3.1.4. Axios Library	12
3.2. Django Framework	12
3.3. API Usage	14
3.3.1. OpenWeatherMap API	14
3.3.2. PayPal API	15
3.3.3. Google Maps API	16
3.3.4. Instagram API	17
4. Comparison of the Old and New Websites	19
4.1. Home Page	19
4.2. Ticket Information Page	22
4.3. Parking Information Page	25
4.4. Sponsorship Page	27
5. UX Tests	28
6. Conclusions	30
7. Limitations	31
A. Appendix	32

1. Introduction

The Internet has become an essential tool for accessing information and connecting with others. As such, having a strong online presence has become increasingly important for businesses, organizations, and institutions. This is especially true for zoological gardens, which rely on visitors and support from the community to thrive.

The purpose of this engineering thesis is to design and implement a website for a zoological garden that is intuitive, informative, and user-friendly. This website should provide a platform for the zoological garden to share information about its animals, exhibits, events, and conservation efforts with the public. By creating a website that meets the needs and expectations of visitors, the zoological garden can improve its outreach and engagement with the community, as well as increase its revenue through online ticket sales and donations. This project will involve a comprehensive analysis of the current state of the zoological gardens' online presence, as well as a thorough evaluation of user needs and preferences.

The resulting website should be a valuable resource for both the zoological garden and its visitors. Essentially, it must be accessible to a diverse audience, comprising individuals with varying levels of age and computer literacy, as this is a common requirement for many public utility services.

This thesis is structured into seven chapters. The initial chapter offers a brief overview of the project and its background. Chapter 2 describes the most crucial project phases and delineates their contents. Chapter 3 explicates the choices of specific technologies that were employed in order to attain the intended outcomes. Chapter 4 familiarizes the reader with the appearance of the final project implementation. It compares an exemplary existing website project with the new one, pointing out the most important differences between the two approaches. Chapter 5 describes the importance of User Experience tests and how they were conducted. Chapter 6 serves as a concluding summary and synthesizes all elements of the thesis in order to evaluate and explicate the project's overall significance. Finally, Chapter 7 discusses the challenges and obstacles encountered during the project and how they were overcome.

1.1. Background Information

The Krakow Zoological Garden approached AGH University with a request for a new website project. Given that the request did not specify any details of the new website design, it was necessary to conduct detailed interviews in order to proceed with the work.

1.2. Objective

The defined project objective was to create a new website that would be responsive to the employees needs and conform to contemporary site standards. Due to frequent changes in the zoological gardens structure, ticket prices, etc., the website also had to be highly configurable and adjustable. The principal assumptions of the project were:

- Transparency and readability of information,
- Content reorganization and condensation, i.e., reducing the amount content on the website without losing important information to improve its availability and clarity,
- Accessible site administration,
- Responsive, extensible design.

The desired features for the main website included an interactive map, a donation button, a gallery of recent Instagram posts, and a live temperature display.

1.3. Methodology

In order to start the project, a brief interview with the employees of the zoological garden was mandatory. Due to the organisation size and complexity, each department had their own needs and requirements. This fact mostly influenced the back-end part of the project.

The consultations were followed by an extensive research concerning zoological garden websites from around the world. The analysis not only provided a lot of inspiration for possible website features, but it also increased awareness of common mistakes. Final results were also consulted with the zoological garden representatives. Upon receiving positive feedback, the project advanced to the testing phase (Described in Chapter 5). The details on the implemented methodology are described in Chapter 2.

1.4. Limitations

Several obstacles have been met during the project realisation (e.g., inability to implement an internal ticket service, lack of free website solutions). These issues presented significant challenges and required careful consideration to overcome. The limitations are further discussed in Chapter 7.

2. Methodology

The project was divided into four phases (i.e., interviews, research and websites comparison) to ensure a clear and organized approach to completing the necessary tasks. By breaking down the management of the project, it was easier to determine the proper sequence of operations and bring the project to a successful conclusion. The following chapter details these phases and explains how they were implemented.

2.1. Interview with Employees

The Cracow zoological garden showed interest in a new website project. However, no specific requirements were given, so it was necessary to conduct detailed interviews to gather the necessary information. After a meeting was arranged, all key assumptions were clarified.

Due to the complex structure of the zoological garden, each department required specific needs to be fulfilled. For example, employees responsible for the Wolski Forest needed the ability to manage content related only to their department. The Django user management system (described in Section 3.2) allowed for the creation of different user groups with different permissions, enabling this differentiation.

It was also requested, that website management should be simple and straightforward. The Django administration interface (mentioned in Section 3.2) provides intuitive tools for managing website content, though it is necessary to learn the Django basics and become familiar with the new dashboard.

During the interviews, it was also determined that some of the features proposed in this thesis, such as an online tour reservation calendar, turned out to be less important for the zoological garden staff than the currently implemented features. Through these agreements, additional time was prevented from being spent on implementing unnecessary features.

The interviews also helped to clarify the details of condensing and reorganizing the website content. The employees accepted the idea and confirmed which sections could be shortened, moved, or deleted, which significantly reduced the amount of information and made it more accessible and easy to find.

2.2. Research Approach

The second phase of the project involved extensive research into available tools and existing zoological garden websites from around the world. The advantages, disadvantages, and technologies used on these sites were taken into consideration, and the results are presented in Table 2.1 in Section 2.3.

This phase provided numerous inspirations and ideas for additional website features and showed what could be achieved with particular technological solutions. It also highlighted common mistakes to avoid. It was found that many of the available zoological garden websites were outdated and poorly responsive, as they were built using outdated frameworks and technologies.

2.3. Existing Website Comparison

When conducting research, several zoological garden websites from around the world were analyzed. Only the most noteworthy ones are presented in this thesis. Table 2.1 shows the technologies used by each website, as well as their main advantages and disadvantages.

Table 2.1: Comparison of zoological garden websites.

Website URL	Used technologies	Advantages	Disadvantages
https://zoo.poznan.pl/	Apache, Reveal.js, jQuery, Platform JS, Google Plus/Twitter widgets	Clean navigation bar, integrated ticketing system	Chaotic content presentation, overwhelming design
https://zoo.wroclaw.pl/	Bootstrap, jQuery, Lightbox	Animations, useful information, interesting field games	Non proportional disturbing pop-ups, stripped down English version, external and out-dated ticketing system

Continued on next page

Table 2.1: Comparison of zoological garden websites. (Continued)

https://zoo.waw.pl/	Bootstrap, jQuery, RespondJS, Apache 2	Interesting background animations, smooth scrolling, minimalistic and clean home page	Some fonts are too large, no animals photos, confusing news layout
https://www.zoomiami.org/	Bootstrap, jQuery, Lightbox, Modernizr	Animated statistics counters, current temperature and opening hours presented on the navigation bar, great responsiveness	None
https://www.omahazoo.com/	Bootstrap, jQuery, Lightbox, Modernizr	Video background, minimalistic and intuitive navigation bar	Some questionable font choices, responsivity problems

Almost every website analyzed in the conducted research implemented *Bootstrap* and *jQuery*. *Bootstrap* is a front-end development framework [1], and *jQuery* is a lightweight JavaScript library that provides pre-built code solutions [2]. These technologies would not be necessary to implement if the project uses *React*, as it already includes everything that can be achieved with these solutions. Alternatively, the *styled-components* library offers a more flexible and customizable approach than *Bootstrap* [3].

While the research did not identify any interesting tools, it served as a source of inspiration for numerous website features.

3. Web Development Technologies

The choice for particular technical solutions was based on the project requirements and a thorough research of actual frameworks and libraries. After evaluating the available options and taking the outlined assumptions into account, React library [4] and Django framework [5] were selected.

This chapter provides a detailed explanation of the technological solutions selected for this project.

3.1. React Library

React is a JavaScript library for building user interfaces. It was developed by Facebook and is now maintained by a community of developers. React allows developers to create reusable components that can be rendered on the server or the client, making it suitable for building single-page applications and mobile applications. Currently, React is the most popular front-end JavaScript library for building Web applications (used by Facebook, Instagram, Netflix and Twitter) [6].

One of the key features of React is the Virtual DOM, which is a lightweight in-memory representation of the actual DOM (document object model) [7]. When a component's state changes, the Virtual DOM is updated to reflect the changes. React then compares the updated virtual DOM with its previous version and determines the minimum number of DOM operations needed to update the actual DOM. This process, called reconciliation, improves the performance of applications by reducing the number of unnecessary DOM updates [8].

In addition to the virtual DOM, React also provides a declarative API for building user interfaces [9]. This means that developers can describe the desired UI (user interface) state and React will handle the rendering and updating of the UI. This makes it easier to build complex UIs and manage their state, as the declarative approach allows developers to focus on the "what" rather than the "how" [4].

React is also known for its use of components, which are self-contained units of code that represent a piece of the UI. Components can be combined to create larger, more complex UIs, and they can also be reused throughout an application. This makes it easier to develop and maintain large applications, as developers can create and manage smaller pieces of code rather than one monolithic code base [4].

React Hooks are a crucial feature that allows developers to use state and life cycle methods inside functional components. One of the most commonly used Hooks is the `useEffect`, which adds the ability to perform side effects from a function component [7]. This Hook is particularly useful when working with API interfaces, as it allows to perform asynchronous tasks and manage the component's life cycle. Section 3.3 provides an example of how the `useEffect` Hook was called in order to perform side effects within a function component.

Due to its large community, resolving any potential problems or learning React concepts is legitimately simple. There are numerous sources to choose from, although the official comprehensible and straightforward documentation was often sufficient itself to meet the requirements of this thesis.

3.1.1. Styled Components

Styled-components is a popular library that allows developers to write actual CSS code to style React components [10]. It works by creating a function that returns a styled component, which can be used like any other React component.

This library combines JavaScript and CSS, using a technique known as CSS-in-JS [11]. This approach offers a highly flexible and reusable method for styling React components. The library has proven to be extremely helpful (to meet the goals of this thesis) in organizing and optimizing the development process, particularly when it comes to managing CSS classes. Overall, *styled-components* has greatly improved the modularity and maintainability of the code, as well as the overall website design [10].

3.1.2. I18next Framework

i18next is a popular internationalization JavaScript framework which integrates with a variety of front-end technologies, including React [12]. The *i18next* tool offers a range of components and functionalities responsible for the smooth translation feature implemented in the project. With *i18next*, translated elements are automatically re-rendered upon a language change, without requiring a page reload. This is possible by the interpolation feature, which allows dynamic values to be inserted into translations. The *useTranslation* function provided by *i18next* allows to easily retrieve translated text when the user switches the language of the website. Listing 1 presents an example *useTranslation* function call, as well as the function responsible for the language change.

Listing 1. Example of *useTranslation* function call and the language change function definition

```
import { useTranslation } from 'react-i18next';  
const onLanguageChange = (langCode) => {  
  i18n.changeLanguage(langCode);  
}
```

```
};
```

The `onLanguageChange` function is called on clicking the flag button placed in the navigation bar, which leaves the user with a clear and intuitive solution.



Figure 3.1. Navigation bar translation buttons: English version (left), Polish version (right).

3.1.3. React-paypal-js Library

By default, PayPal's implementation requires the inclusion of the JS SDK `<script>` tag in an HTML file. This results in components such as PayPal buttons being rendered only after the script has finished loading. This approach goes against the component-oriented design of React, as it requires external scripts to be loaded from the `index.html` file. To address this issue, the `react-paypal-js` library offers a solution that allows developers to abstract away the complexities of loading the JS SDK. It enforces best practices by default, ensuring optimal user experience for website visitors.

3.1.4. Axios Library

Axios is a popular JavaScript library that allows users to make HTTP requests from a web browser. It can be used within React for a convenient way to make HTTP requests to a given endpoint [13]. Implementing such client was mandatory in order to use the API interfaces. *Axios* has good defaults to work with JavaScript Object Notation (JSON) data. Unlike alternatives, such as the *Fetch API*, it rarely requires setting any particular headers or performing tedious conversion procedures.

3.2. Django Framework

Django is a free and open-source web framework written in Python [5]. It follows the model-template-view (MTV) architectural pattern, which separates an application into three main components: the model, the view, and the template.

The model component refers to the data and logic of the application. It is responsible for storing, validating, and manipulating data. The model is often implemented using a database and an object-relational mapper (ORM) that allows developers to interact with the database using high-level Python code rather than SQL.

The view component refers to the presentation and logic of the application. It is responsible for handling user requests, rendering templates, and returning responses to the user. In Django, views are implemented as Python functions or methods that take a request and return a response.

The template component refers to the structure and layout of the application. It is responsible for rendering dynamic content within a fixed structure. In Django, templates are implemented as HTML files with placeholders for dynamic content, which are filled in by the view component at run-time.

One of the key benefits of Django is its emphasis on rapid development and pragmatic design [14]. It provides many built-in features and tools that enable developers to build web applications quickly and efficiently, including a built-in admin interface, user authentication, form validation, caching, etc. Django is also highly scalable and can handle high traffic websites, thanks to its robust database-driven backend and efficient request-response cycle [15]. The integration of React and Django has become a prevalent method for constructing web software worldwide [16], [14].

In this work, Django played a key role in managing the website’s news section, including creating, deleting, and editing posts. In addition to handling form validation, this framework is responsible for storing and organizing acquired data efficiently. Figure 3.2 presents the table responsible for storing validated data from the sponsorship form.


sponsors	
 id	linestring
name	linestring
address	linestring
postal_code	int
city	linestring
phone_number	bigint
animal	linestring
period	int
since	datetime
email	linestring
rodo_agreement	boolean
data_agreement	boolean

Figure 3.2. Data base table example.

3.3. API Usage

An API is a set of rules and protocols that define how different software programs or systems can interact with each other. It serves as an intermediary layer between two systems, allowing them to communicate and exchange data with each other through a documented interface. APIs are widely used in software development to enable integration between different systems. The implementation of all four project features mentioned in Section 1.2 required the use of APIs. Figure 3.3 presents a graphic concept of an API in which a web application makes a request to the API service by sending an HTTP request. The API service receives the request and processes it, checking for any required authentication. The API service retrieves the requested data using a web server, that fetches it from a database. The API service formats the data and sends it back to the web application. The application receives the response and processes it by parsing.

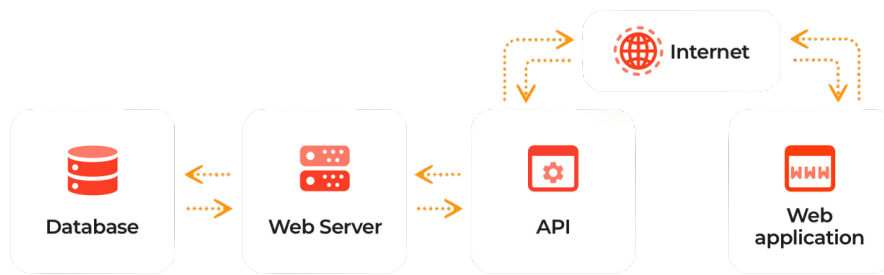


Figure 3.3. The concept of an API. Source: [17].

The API functionalities in the project are realised by React [13] and Django.

3.3.1. OpenWeatherMap API

The first API interface utilized in this project was provided by *OpenWeatherMap* [18]. The available free plan was enough to implement a feature that displays the current temperature on the website's navigation bar, as shown in Figure 3.4.



Figure 3.4. Temperature API usage. Trademark source: [19].

Listing 2 shows an API request made using the Axios HTTP client. The request is sent to the following URL: <https://api.openweathermap.org/data/2.5/weather?lat=50.054810&lon=>

`19.927839&appid=$APIKEY&units=metrics`. Provided URL contains the coordinates for Cracow and an API key generated from the OpenWeatherMap website.

Listing 2. *OpenWeatherMap* API request.

```
useEffect(() => {  
  axios.get(url).then((response) => {  
    setWeather(response.data);  
    temperature = response.data.main.temp;  
  });  
}, [url]);
```

The *useEffect* Hook is used to execute the API request on every page render, allowing users to see the current temperature every time the page is loaded. As the API response contains a lot of detailed weather information, only the current temperature is extracted from the acquired data and displayed on the navigation bar.

3.3.2. PayPal API

Currently, the website offers only one way to support the zoological garden, which is through animal sponsorship [20]. This method is more complicated and complex than a simple, one-time donation feature. The suggested new support sub-page was approved by the employees and created in order to provide additional, simple way for users to further support the garden.

The donation button feature was implemented using the *react-paypal-js* component, serving as a key element on the support sub-page. PayPal provides a *developer dashboard* that allows developers to obtain the necessary credentials and create sandbox accounts to set up the API. This dashboard is a valuable resource for developers who want to use PayPal's API in their projects.



Figure 3.5. Donate button implementation.

The *DonateButton* function presented in Listing 3 utilizes data generated by the PayPal configuration form to initiate the donation process. Further part of the donation process is handled entirely by PayPal.

Listing 3. DonateButton function.

```
import React, { useEffect, useMemo, useRef } from 'react';

let counter = 0;
const generateId = () => {
  return `ID-${++counter}`;
};

const DonateButton = () => {
  const buttonRef = useRef(null);
  const buttonId = useMemo(() => `ID-${generateId()}`, []);
  useEffect(() => {
    const button = window.PayPal.Donation.Button({
      env: 'sandbox',
      hosted_button_id: '9H4ALX94HBDVJ',
      image: {
        src: 'https://www.paypalobjects.com/pl_PL/...',
        alt: 'Donate with PayPal button',
        title: 'PayPal - The safer, easier way to pay
        ↪ online!',
        width: '20',
        height: '20'
      }
    });
    button.render(`#${buttonRef.current.id}`);
  }, []);
  return (
    <div ref={buttonRef} id={buttonId} />
  );
};
```

3.3.3. Google Maps API

The Google Maps Embed API is a service that provides an easy way to display maps on the website. It was implemented in order to allow the visitors to use an interactive map showing zoological garden location and check suggested directions. The map itself is defined as a simple HTML request and does not require any JavaScript, therefore its implementation was straightforward.

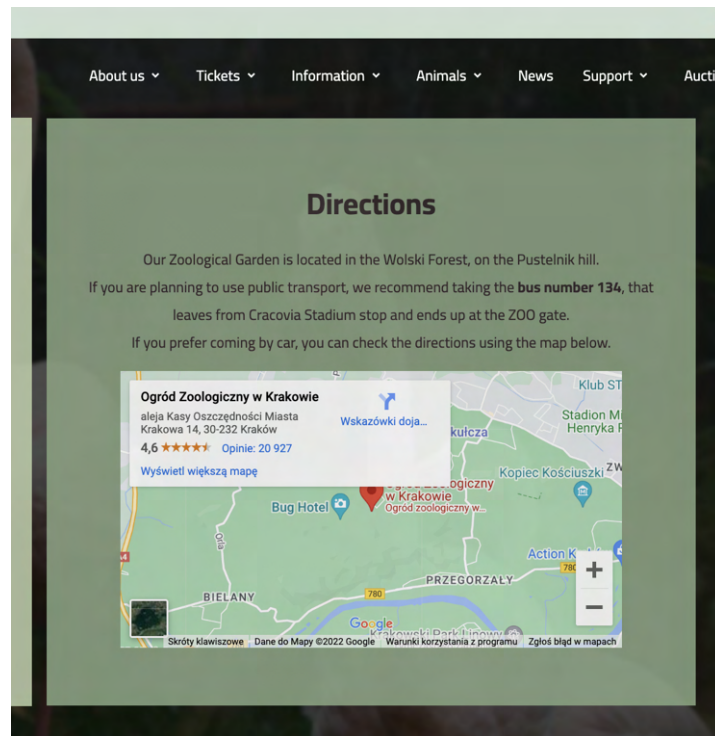


Figure 3.6. Google Maps API implementation.

The API can be set as *src* attribute of an *iframe* HTML tag, as shown in Listing 4.

Listing 4. Google Maps API definition.

```
<MapWrapper>
<iframe width="110%" height="300px" loading="lazy"
  ↪ frameborder="0" scrolling="no" marginheight="0"
  ↪ marginwidth="0" id="gmap_canvas"
  ↪ src="https://www.google.com/maps/embed?pb=[...]">
</iframe>
</MapWrapper>
```

3.3.4. Instagram API

The *Instagram basic display API* provides read-only access to account's basic data, including photos and their captions.

The *following* API was implemented using the *react-ig-feed* component. After creating an application using the Meta dashboard, *App ID* and *App Secret* variables were generated. In order to use particular account credentials, the owners have to authenticate it with the developer's Meta app. Successful authentication resulted in obtaining a special token, that was later passed to *react-ig-feed* component.

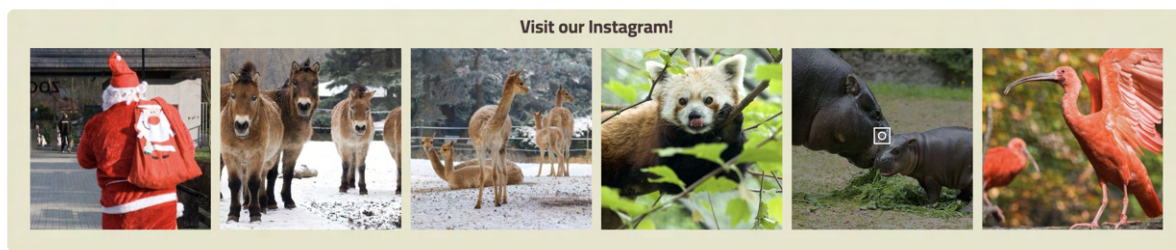


Figure 3.7. Instagram API showing recent photos from the Instagram page.

This feature allows to display the six recent photos from the zoological garden Instagram page. Upon clicking on a specific photo, the user will be redirected to the corresponding post on the Instagram.

4. Comparison of the Old and New Websites

The following chapter presents an overview of the achieved results. The current website pages are presented in a comparison with the new versions created during the project, highlighting the most significant differences and the issues addressed in the new website design. For the sake of brevity, only the most important information will be included in this report. Both the desktop and mobile views will be taken into consideration.

4.1. Home Page

According to the user interviews described in Chapter 5, current home page of the website is full of information, making it sometimes difficult to determine which sections are the most important. The layout is cluttered with images (Figure 4.1), excessive animations and widgets overlapping the navigation bar. This can be confusing for users, potentially leading them to give up on searching for the desired information.

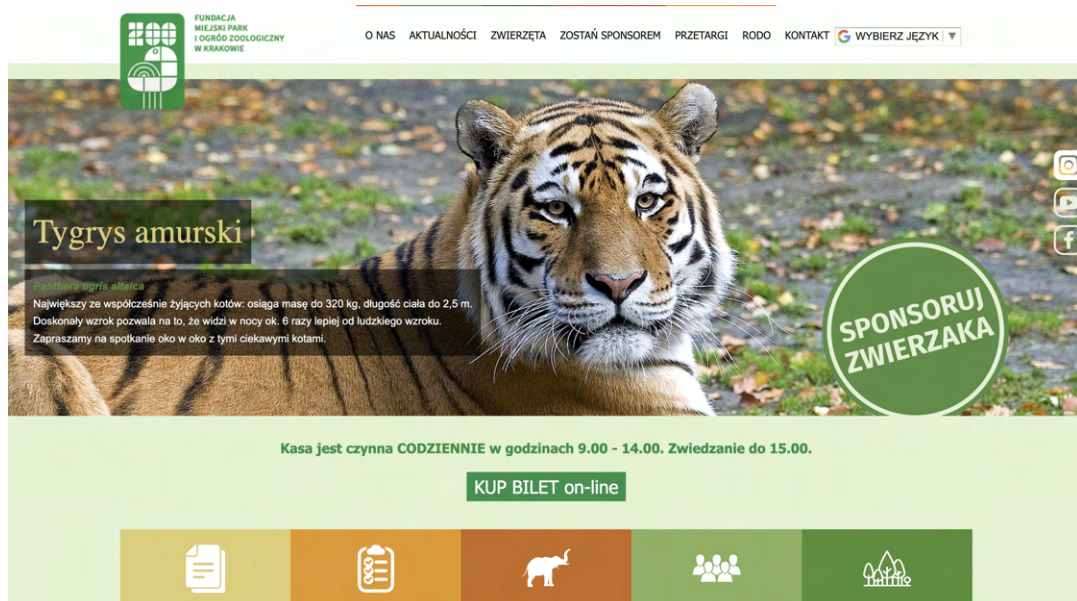


Figure 4.1. Current website home page [19].

The new home page design (Figure 4.2) is minimalist, featuring a video hero background that is giving users a glimpse into the zoological garden alleys. The video showcases the facilities and animals at the zoological garden, helping visitors become familiar with the location and encouraging them to visit. The clean, streamlined design of the page makes it easier for users to find the information they are looking for.

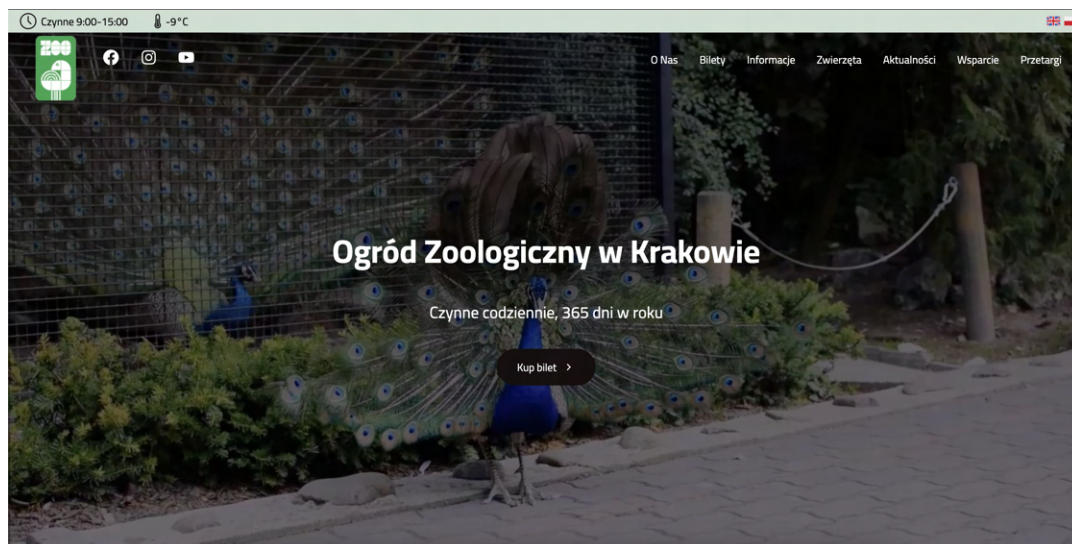


Figure 4.2. New website home page.

The current website has five separate information sections (Figure 4.3), with a total of 27 subsections. It causes visitors to struggle finding the specific information they are looking for.

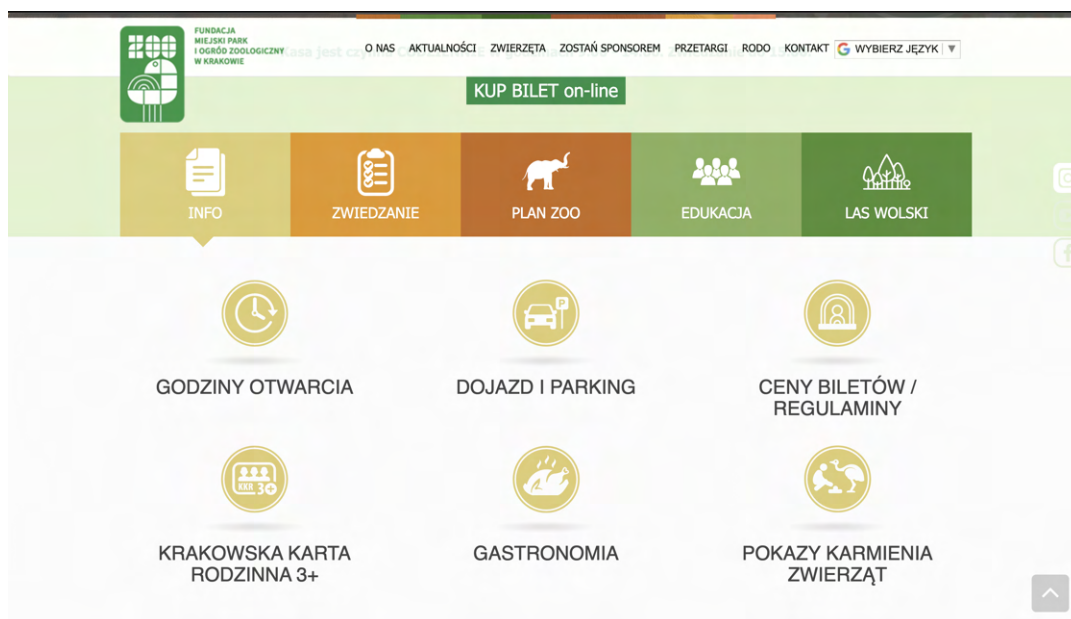


Figure 4.3. Current website information section [19]

By reducing the content amount, the number of subsections has been reduced to eight (Figure 4.4), resulting in a organised and transparent layout. All of the details that were previously displayed on the home page have been moved to appropriate subsections, allowing the information to be easily accessed while also freeing up space on the home page.

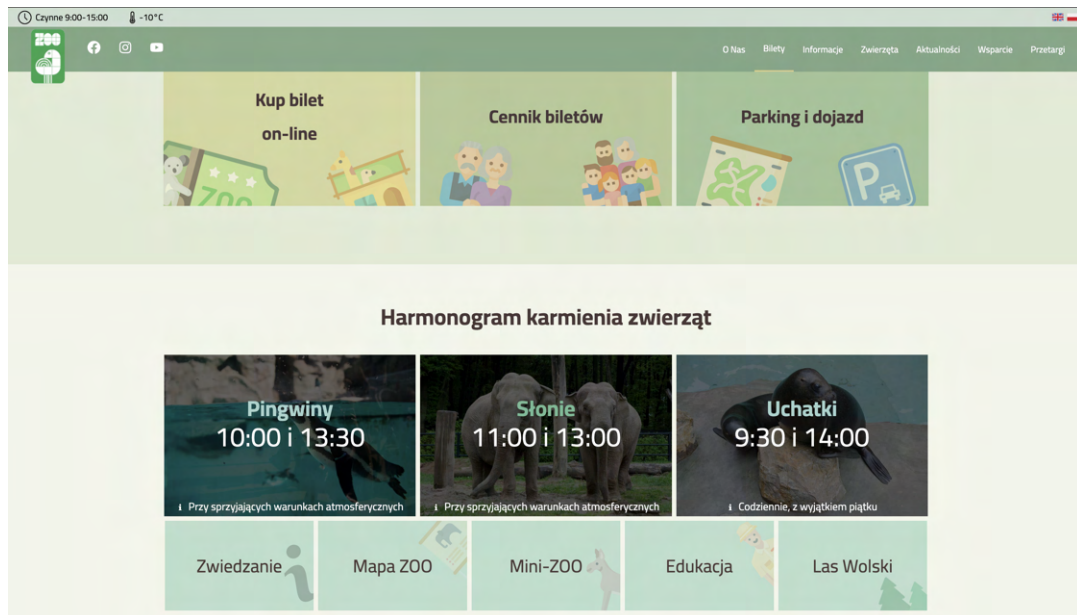


Figure 4.4. New website information section.

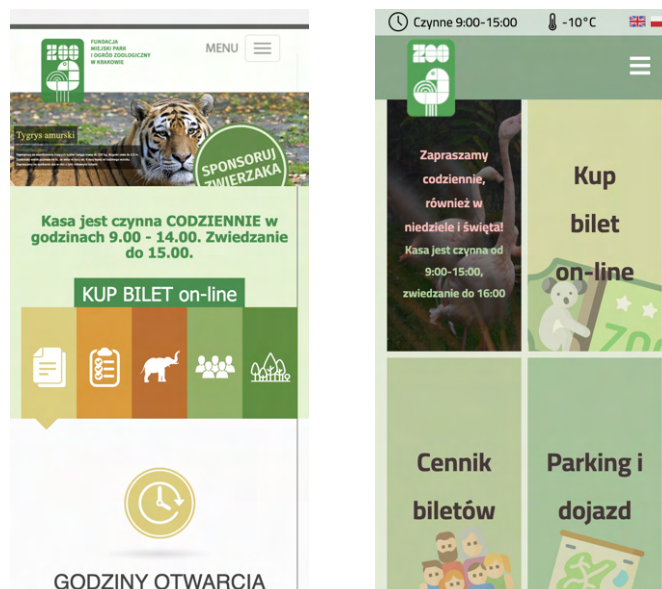


Figure 4.5. Mobile website layout comparison of the home screen: current website [19] (left), new website (right).

The mobile version of the website is particularly unclear, as shown in Figure 4.5, on the left. The new website addresses these issues and significantly improves the mobile view

(Figure 4.5, on the right), ensuring that users have a positive experience regardless of the device they are using.

4.2. Ticket Information Page

The current ticket information page is cluttered with buttons unrelated to ticket purchasing (Figure 4.6), requiring users to scroll down to see the actual prices (Figure 4.7), which are presented in plain text.

The prices are divided into two sections for different seasons, adding an additional layer of complexity for users. User interviews (described in Chapter 5) revealed that it is difficult to quickly and easily find the desired ticket price and purchase a ticket.



Figure 4.6. Current pricing page initial view [19].

The pricing information overflows the navigation bar (Figure 4.7), making the text difficult to read. This issue has already been mentioned in Section 4.1 and it occurs in most of the website sub-pages.

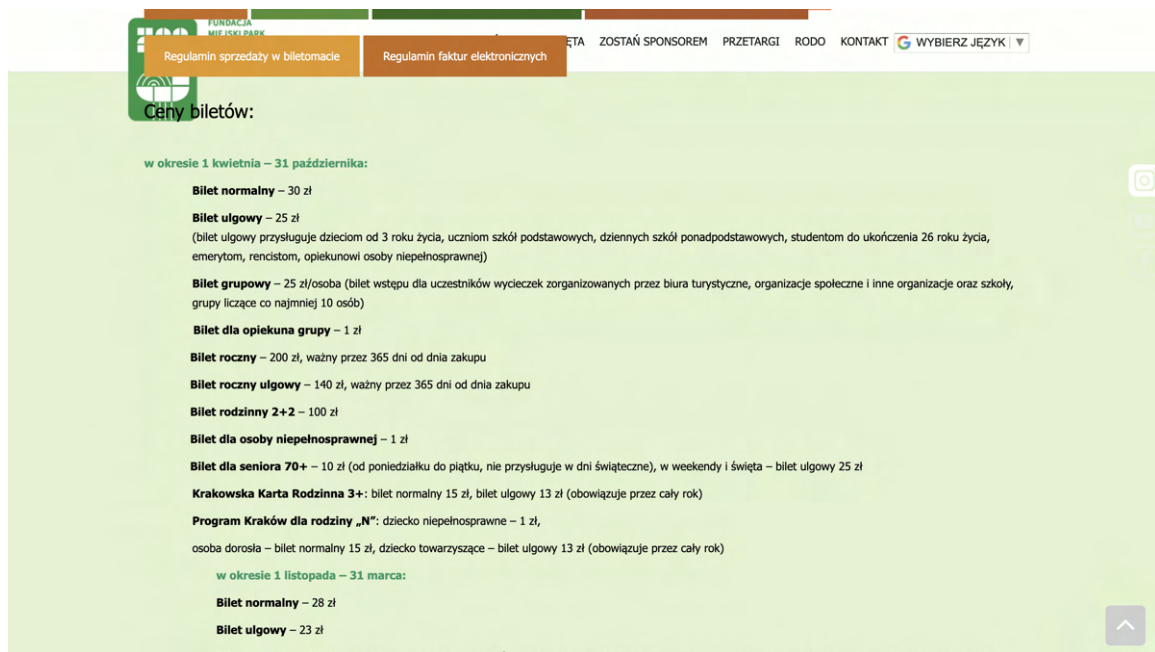


Figure 4.7. Current pricing page scrolled down [19].

The new website displays ticket prices in a visually appealing and easy-to-understand format. The most popular tickets are prominently displayed on the page, as shown in the Figure 4.8. If users want to see a detailed pricing list (Figure 4.9), they can easily access it with a dedicated button.

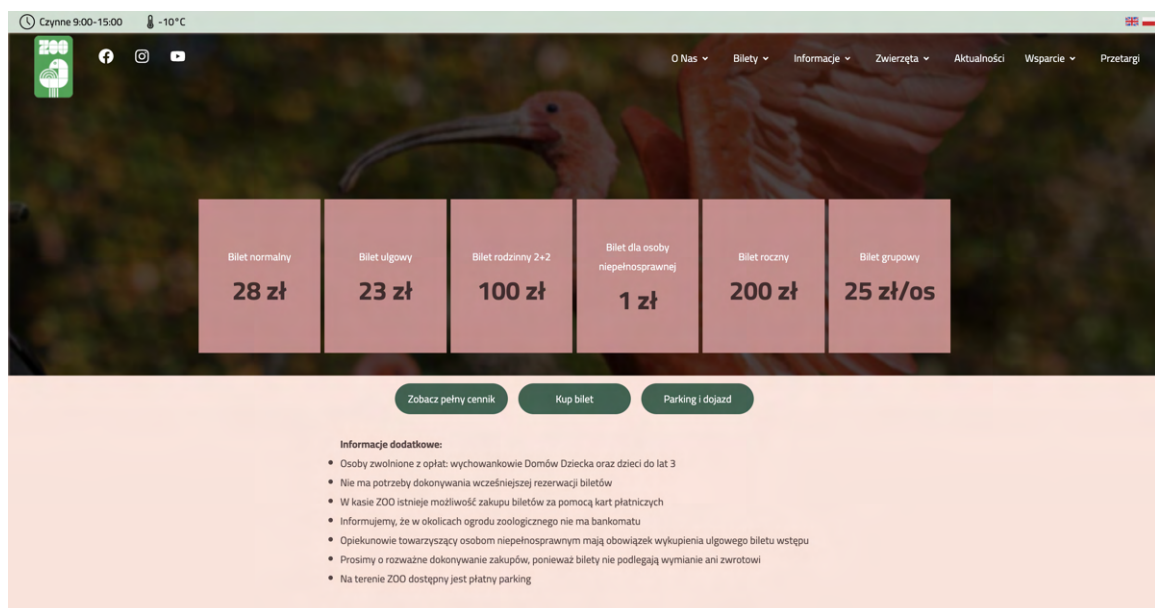


Figure 4.8. New pricing page initial view.

Bilet normalny 28 zł

Bilet ulgowy 23 zł

Bilet rodzinny 2+2 100 zł

Bilet dla osoby niepełnosprawnej 1 zł

Bilet roczny 200 zł

Bilet grupowy 25 zł/os

Rodzaj biletu	Cena
Normalny	28 zł
Ulgowy*	23 zł
Grupowy*	23 zł/os
Dla opiekuna grupy*	1 zł
Rodzinny 2+2	90 zł
Dla osoby niepełnosprawnej	5 zł
Dla seniora 70+*	5 zł

Rodzaj biletu **Cena**

Normalny	200 zł
Ulgowy	140 zł

Krakowska Karta Rodzinna 3+ **Cena**

Bilet normalny	14 zł
Bilet ulgowy	11 zł

Kraków dla rodziny "N" **Cena**

Dziecko niepełnosprawne	5 zł
Bilet normalny	13 zł
Bilet ulgowy	11 zł

Informacje dodatkowe:

- Osoby zwolnione z opłat: wychowankowie Domów Dziecka oraz dzieci do lat 3
- Nie ma potrzeby dokonywania wcześniejszej rezerwacji biletów
- W kasie ZOO istnieje możliwość zakupu biletów za pomocą kart płatniczych
- Informujemy, że w okolicach ogrodu zoologicznego nie ma bankomatu
- Opiekunowie towarzyszący osobom niepełnosprawnym mają obowiązek wykupienia ulgowego biletu wstępu
- Prosimy o rozważne dokonywanie zakupów, ponieważ bilety nie podlegają wymianie ani zwrotowi
- Na terenie ZOO dostępny jest płatny parking

Figure 4.9. New pricing page displaying a complete pricing table.

The new site only presents the valid prices using a simple if-else statement based on the current month. Additional information is displayed in an easy-to-read list format. From this page, users can quickly access parking information or proceed directly to ticket purchasing. Figure 4.10 compares mobile website layouts for the current (figure on the left) and the new (figure on the right) implementations.

roku życia, uczniom szkół podstawowych, dziennych szkół ponadpodstawowych, studentom do ukończenia 26 roku życia, emerytom, rencistom, opiekunowi osoby niepełnosprawnej)

Bilet grupowy – 25 zł/osoba (bilet wstępu dla uczestników wycieczek zorganizowanych przez biura turystyczne, organizacje społeczne i inne organizacje oraz szkoły, grupy liczące co najmniej 10 osób)

Bilet dla opiekuna grupy – 1 zł

Bilet roczny – 200 zł, ważny przez 365 dni od dnia zakupu

Bilet roczny ulgowy – 140 zł, ważny przez 365 dni od dnia zakupu

Bilet rodzinny 2+2 – 100 zł

Bilet dla osoby niepełnosprawnej – 1 zł

Bilet dla seniora 70+ – 10 zł (od poniedziałku do piątku, nie przysługuje w dni świąteczne), w weekendy i święta – bilet ulgowy 25 zł

Krakowska Karta Rodzinna 3+:
bilet normalny 15 zł, bilet ulgowy 13 zł (obowiązuje przez cały rok)

Bilet normalny 28 zł

Bilet ulgowy 23 zł

Bilet grupowy 25 zł/os

Bilet dla osoby niepełnosprawnej 1 zł

Bilet roczny 200 zł

Bilet rodzinny 2+2 100 zł

Figure 4.10. Mobile website layout comparison of ticket information: current implementation [19] (left), new implementation (right).

4.3. Parking Information Page

The current website has a disorganized and cluttered display, with information presented in plain text that overlaps the navigation bar. The provided images, which likely depict a low-quality map, are unreadable and of little use to most visitors. The conducted user interviews confirm that.

The website is overloaded with unnecessary details that make it difficult for users to find the crucial information they are seeking. Figure 4.11 presents the current website parking subsection. The upper image shows the initial view of the page, while the lower image presents the page after scrolling down.



Figure 4.11. Current parking information page: the upper image shows the initial view, the lower image presents the page after scrolling it down [19].

The new website (Figure 4.12) includes the same information as the previous version, but presents it in a clear and intuitive structure. Prices are displayed in concise, condensed tables. Low-quality image maps have been replaced with an interactive Google Map widget.

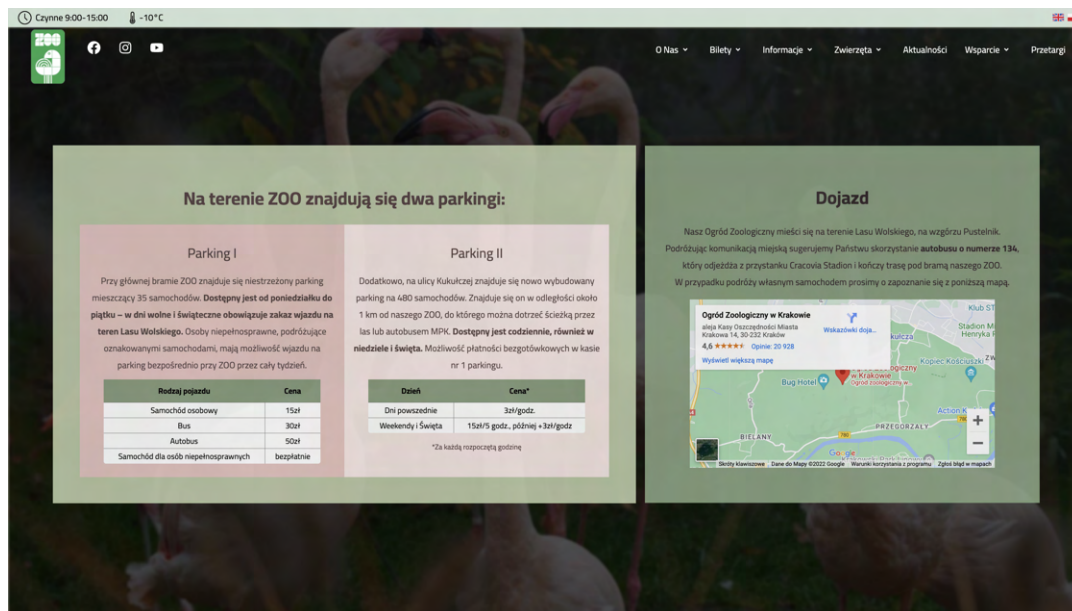


Figure 4.12. New parking information page.

Current parking information page suffered from the same issues as the previous sub-pages. The new website addressed them, making the layout more user-friendly and approachable. Figure 4.13 compares the mobile website layouts.



Figure 4.13. Mobile website layout comparison of the parking page: current implementation [19] (left), new implementation (right).

4.4. Sponsorship Page

The current *sponsorship* page (Figure 4.14) requires users to scroll through the entire page to find the sponsorship form, which is located in a section titled “How to become a sponsor?” rather than the more intuitive “Become a sponsor” section. The form is presented in unappealing, plain text.



Figure 4.14. Current sponsorship page [19].

The new website (Figure 4.15) has eliminated the confusing, similarly-named sections, moving the sponsorship form to a section titled “Become a sponsor” and creating a separate section for sponsorship prices. The information previously located in “How to become a sponsor?” has been incorporated into the “Become a sponsor” section, resulting in a more intuitive and user-friendly layout.



Figure 4.15. New sponsorship page.

5. UX Tests

User experience (UX) tests were conducted to analyze the usability of the developed website and ensure that the project assumptions were met. These tests allow for the objective evaluation of how easily users can complete a given task.

The current Krakow zoological garden website has several issues with page navigation and usability. These issues have been identified and resolved according to the design principles [21]. The conducted tests allowed to verify the implemented solutions.

To mimic the diverse range of potential zoological garden website visitors, fifteen testers were selected from a range of ages (18 to 60) and had varying levels of technological skills. To ensure the accuracy of results, the two versions of the web page were randomly displayed during the test, following best practices for controlling for potential biases as outlined by W. Craig Tomlin in his book on behavioral and usability testing [22]. This method helps to eliminate any external factors that might influence the performance of the pages being tested, allowing to attribute any observed differences to the specific design and content of the pages. The time taken to complete each task was measured, and the subjects were briefly interviewed about the differences, advantages, and disadvantages of the two websites.

The subjects were asked to complete five tasks: buy a ticket, check ticket pricing, check parking information, check animal feeding hours, and find a sponsorship form. The average test results for the current website are presented in Table 5.1, while the results for the new website are presented in Table 5.2.

Table 5.1. UX test results comparison - current website.

Task	Current website	Standard deviation
Buy a ticket	3 s	0.98 s
Check tickets prices	21 s	10 s
Check parking information	46 s	17 s
Check animal feeding hours	25 s	13 s
Find a sponsorship form	27 s	12 s

Table 5.2. UX test results comparison - new website.

Task	New website	Standard deviation
Buy a ticket	2 s	1 s
Check tickets prices	12 s	4 s
Check parking information	36 s	17 s
Check animal feeding hours	9 s	6 s
Find a sponsorship form	6 s	2 s

The test results confirm that the goals were achieved. The readability of the website has improved, and most tasks were completed almost 50% faster on the new website compared to the old one.

One task that did not show a significant improvement on the new website was checking parking information. This is because the parking information is located in the "Ticket" section, not the "Information" section. This suggests that this content should be moved to a better location. Before the content compression process, the "Information" section had more than 25 sub-pages, and two of the subsections were moved to the "Ticket" section. Moving the third section containing the parking information to a more appropriate location would likely make it easier to find.

After the employee interview, the proposed content compression idea was well-received and resulted in reducing the "Information" section to five sub-pages. In this case, it would be more reasonable to define the parking information in a more accurate location.

After the tests, subjects were briefly interviewed about the disadvantages and advantages of both the current and new websites. All participants mentioned the chaotic and cluttered structure of the current website, making it difficult to find important information. In contrast, the new website was praised for its organized and intuitive structure.

The interview with test participants also revealed that the new website parking sub-page location is non-intuitive, which was one of the two disadvantages noted by the testers. The second flaw was mostly reported by the majority of older testers and concerned the lack of a "Contact" tab on the navigation bar that is present on the current website and will be added in the new one. Participants appreciated the reduced amount of content and found the new website more transparent and less overwhelming.

Conducting UX tests with a larger group of people would provide more opportunities to identify other potential areas for improvement.

6. Conclusions

The research outcomes offered valuable insight into web development processes and technologies, highlighting the most popular solutions and the reasons behind their popularity. It also identified the issues that many websites struggle with and presented a challenge to create a page that breaks the mold. The analysis revealed that even the largest zoological garden websites are outdated. By identifying these flaws, it was possible to create a website that can easily compete with the world's most known zoological gardens.

The conducted research and the completed work demonstrate the importance of effective website design and implementation for promoting a zoological garden and attracting visitors. Effective website design and implementation are crucial for creating a positive user experience and ensuring the site's effectiveness. Attention to user needs and preferences, as well as the use of minimalist design and condensed content, can significantly improve the site's usability.

Multimedia elements such as videos and images can enhance the website's appeal. Regular testing and maintenance are necessary to ensure the website is functioning properly and meeting the needs of its users.

The work completed in this thesis has resulted in a more effective and user-friendly website for the zoological garden. The UX tests have proven that the new project successfully addresses all of the current website issues and have also identified additional areas for improvement.

Future research could include further testing and analysis to continually improve the website and optimize its performance. It is likely that with time, zoo websites will be updated and set new standards that will facilitate further development.

7. Limitations

During the course of project realisation several obstacles have been met, most of which turned out to be manageable. A fundamental and non-manageable issue concerned the inability to obtain the application programming interface (API) from the current ticket sales company, that would allow visitors to buy tickets directly from the zoological garden's website. This led to following the old solution of an external URL reference, which may raise trust issues or mislead potential visitors. Unfortunately, there was no other option available due to the lack of the API.

Another of the encountered difficulties was the limited availability of free solutions for implementing different website features (e.g., donation button, background video player). In fact, most of the available options required payment. In order to implement a voluntary donations feature, a research of major online payment system companies' offers was conducted. No free options were available, although PayPal offered a free *Payments REST APIs* and manageable sandbox accounts to set up the donation feature. This factor determined the choice for PayPal solution implemented in the project.

Another obstacle concerned the lack of free, working video players feature. The project home page consists of the zoological garden facilities video footage that serves as a hero background. Playing the video from an external server would greatly enhance the website's performance. Regrettably, all the available online video-playing services, such as *Vimeo*, required additional payments. The sole option left was to play the video using an HTML tag and a local file. All the other website desired features were covered with free alternatives.

One of the project assumptions was that the site administration should be easy and accessible, but this also presented some difficulties. The current zoological garden website was developed using Wordpress, which is an online, browser-based content management system (CMS) that allows users to create an entire website without manual code editing [23]. There are numerous templates to choose from, with the content being transferred automatically between configurations. It is not possible to attain the level of simplicity, provided e.g., by Wordpress, in site administration without employing a content management system. However, it is also worth noting that CMS systems come with additional costs, and more advanced website customization requires purchasing an expensive plan. For example, setting up a custom domain or removing the company's branding is only possible with a premium plan.

A. Appendix

The source code of the implemented website has been compressed and attached to the thesis in a separate file.

Bibliography

- [1] „Bootstrap”. URL: <https://getbootstrap.com/docs/4.1/getting-started/introduction/> (visited on 2022-12-26).
- [2] „jQuery”. URL: <https://api.jquery.com/> (visited on 2022-12-26).
- [3] Ryan Lanciaux. „Modern Front-end Architecture”. Apress, 2021.
- [4] Chinnathambi K. „Learning React”. Addison-Wesley, 2017.
- [5] „Django framework”. URL: <https://docs.djangoproject.com/en/4.1/> (visited on 2022-12-15).
- [6] „React – a JavaScript library for building user interfaces”. URL: <https://reactjs.org/>.
- [7] Anthony Accomazzo, Nate Murray, and Ari Lerner. „Fullstack React: The Complete Guide to ReactJS and Friends”. Fullstack.io, 2017.
- [8] „React – Virtual DOM”. URL: <https://reactjs.org/docs/faq-internals.html>.
- [9] „React – Declarative API”. URL: <https://reactjs.org/docs/reconciliation.html#gatsby-focus-wrapper> (visited on 2022-12-13).
- [10] „React-specific CSS-in-JS styling solution”. URL: <https://styled-components.com/docs> (visited on 2022-12-13).
- [11] „CSS-in-JS technique”. URL: <https://css-tricks.com/a-thorough-analysis-of-css-in-js/> (visited on 2022-12-20).
- [12] „react-i18next”. URL: <https://react.i18next.com/> (visited on 2022-12-18).
- [13] Ujjawal Shrestha. „Implementing API in ReactJS”. In: (2020).
- [14] Arun Ravindran. „Django Design Patterns and Best Practices - Second Edition: Industry-standard web development techniques and solutions using Python”. Packt Publishing, 2018.
- [15] William Vincent. „Django for Beginners: Build websites with Python and Django”. Paperback, 2018.
- [16] „Django and React as a canonical combination”. URL: <https://www.creolestudios.com/benefits-of-reactjs-with-django/> (visited on 2022-12-16).

- [17] „Cleveroad”. URL: <https://www.cleveroad.com/blog/what-is-an-api/> (visited on 2022-12-16).
- [18] „Open Weather Map API”. URL: <https://openweathermap.org/api> (visited on 2022-12-16).
- [19] „Cracow Zoological Garden website”. URL: <https://zoo-krakow.pl/> (visited on 2022-12-16).
- [20] „Cracow Zoological Garden sponsorship page”. URL: <https://zoo-krakow.pl/jak-zostac-sponsorem/> (visited on 2022-12-16).
- [21] Joel Marsh. *„UX for Beginners: A Crash Course in 100 Short Lessons”*. O'Reilly Media, 2016.
- [22] W. Craig Tomlin. *„UX Optimization: Combining Behavioral UX and Usability Testing Data to Optimize Websites”*. Apress, 2018.
- [23] McKeever S. *„Understanding Web content management systems: evolution, lifecycle and market”*. In: (2003).