POLITECNICO DI TORINO

Master degree course in Digital Skills for Sustainable Societal Transitions

Master Degree Thesis

Integration and exploitation of AI tools for designing and operating modern website

Relatori
Urgese Gianvito
Fanuli Giuseppe

Candidato
Tannaz Jamshidi

July 2025

Abstract

This thesis explores how AI-assisted tools can support the design and development of modern websites, especially for individuals with little formal experience in web development. The main objective is to understand how these tools can simplify the creation process, making it faster and more accessible, while still allowing for high-quality, professional outcomes.

The focus is particularly on front-end development, where AI tools proved most helpful in the early stages. By providing immediate visual feedback and enabling quick prototyping, these tools encouraged a hands-on, trial-and-error approach that made the learning process more intuitive. Instead of building everything from scratch, users could experiment, observe changes in real time, and iterate quickly—an approach that helped them grasp front-end concepts more effectively.

In addition to visual AI design tools, the use of conversational agents like ChatGPT added another layer of support. Acting as an on-demand assistant, ChatGPT provided explanations, suggested solutions, and helped troubleshoot problems throughout the development process, making it easier for beginners to stay engaged and solve challenges independently.

While AI tools offered clear advantages in terms of speed, creativity, and accessibility, the thesis also highlights their limitations. Tasks that required more complex logic, custom interactions, or polished refinements still needed human intervention and critical thinking. This underlines the importance of combining AI tools with human oversight and problem-solving skills.

Alongside the technical work, the thesis also emphasizes the value of thoughtful user interface (UI) and user experience (UX) design. Careful attention was paid to color schemes, typography, layout, and visual elements, all treated as part of a coherent design system. Creating a visually consistent, accessible, and user-friendly experience was essential to meet both aesthetic and functional goals.

This methodology was applied in a real-world case, the design and development

of the official website for the inNuCE Lab (Neuromorphic Computing and Engineering Lab) at Politecnico di Torino. The project demonstrates how AI-powered tools, when used alongside design best practices and human creativity, can lower the barrier to entry in web development while delivering professional results.

Contents

List of Figures

Chapter 1

Introduction

Having a website in this day and age has a massive impact on success. One of the main reasons is that a website can increase an organization's credibility, helping to communicate trust and professionalism to consumers. It also enhances the first impression of your business and showcases your brand effectively [1].

Prebuilt website solutions offer notable convenience and cost-effectiveness, making them especially appealing to small businesses or individuals with limited budgets and technical expertise. These platforms are quick to deploy and typically easy to use. However, they come with significant limitations, particularly in terms of flexibility and customization. Prebuilt solutions may not meet the needs of more complex or unique business requirements and can present challenges related to scalability and integration with other systems [2].

On the other hand, professional web development provides a high degree of customization, allowing businesses to create websites tailored to their specific needs and branding. This approach is ideal for organizations with distinct requirements or those looking to stand out in a competitive market [4]. Nonetheless, this method requires a greater investment of time and money. The success of such projects also depends heavily on clear and consistent communication between stakeholders and developers to ensure the final product meets expectations.

AI has significantly advanced the development of web interfaces through the generation of AI-powered automated code. It has the potential to transform the entire design process, from AI-driven design solutions to sophisticated image-to-code translation methods. Modern AI models are capable of converting various design inputs into coherent, usable code, thereby shortening development cycles and enhancing the quality, consistency, and efficiency of online interfaces.

In this thesis, I aim to demonstrate that with the help of AI-based tools, it is

possible to learn both the design and development processes involved in creating websites, while also supporting the automation of various development tasks. These tools not only facilitate the understanding of professional web design workflows but also streamline and accelerate the development process by automating repetitive or complex steps, ultimately making web creation more accessible and efficient.

Chapter 2

Background

This project builds on the integration of effective web design principles, AI-powered tools, and a structured development pipeline to create a modern and user-friendly website for the **inNuCE - Nuromorphic Computing and Engineering Lab**. At the core of the design process were fundamental principles, which guided the visual structure and improved the overall user experience. The inNuCE Lab, known for its focus on neuromorphic computing and interdisciplinary research, needed a platform that reflected its identity and mission.
To support this, various AI tools were explored and utilized for different purposes. Alongside these, a set of design and development tools including *Figma*, *Adobe Illustrator*, *Visual Studio Code*, and the *React framework* played a key role in translating ideas into a functional and polished website. This combination of thoughtful design, intelligent tools, and a collaborative workflow helped deliver a visually coherent and technically robust web presence for the laboratory.

In this chapter, I will introduce the technologies and concepts adopted for the development of the platform, which will be further discussed in the next chapter in terms of how they were integrated to achieve the objectives of this thesis project.

2.1 Web Design

The use of design principles in website design is highly effective, as it enhances user navigation and overall experience. These principles help users find information efficiently, navigate between pages smoothly, and feel more satisfied when reaching their goals. Additionally, consistent use of visual elements like layout, color, and typography strengthens brand recognition and shapes user perception. For designers, these principles also provide a clear structure, making it easier to organize content in a way that is both visually appealing and user-friendly [3].

2.1.1 Design Principals

. Design principles form the foundation of effective visual communication. Without them, a design may lack clarity, structure, and usability. These principles guide how content is organized and perceived, ensuring that the intended message is communicated clearly and efficiently to the user [4].

Line

The line is one of the most fundamental elements in design. It connects two or more points and can take various forms—straight, curved, or wavy. Lines are commonly used to organize content, create separation, emphasize elements, or guide the viewer's attention throughout a layout.

Shape and Form

Shapes are defined as two-dimensional areas with clear boundaries and can be either geometric or organic. They help structure visual content, build illustrations, and attract attention. When shapes take on a three-dimensional quality—either physically or through visual techniques such as shading and perspective—they become forms. Understanding form enables designers to create depth and realism in flat compositions.

Texture

Texture refers to the surface quality of a design element, either actual or implied. It adds depth and interest to a design by suggesting materials such as smoothness, roughness, or softness. Texture should be applied thoughtfully to enhance rather than distract from the overall composition.

Balance

Balance is the equal distribution of visual weight across a design. It contributes to stability and harmony, making layouts more pleasing and readable. Designers often use tools like the rule of thirds and grid systems to achieve balance through the careful placement of elements, color, and space.

Typography

Typography refers to the style, arrangement, and appearance of text. It plays a critical role in readability and user engagement. Effective typography uses consistent font styles and sizes, with an emphasis on simplicity and hierarchy, to create a coherent and accessible visual language.

Hierarchy

Hierarchy helps establish a visual order, guiding users to the most important content first. This can be achieved through size, contrast, font weight, or spatial placement. A well-defined hierarchy improves comprehension and ensures that the user's attention is directed effectively.

Images

Images enhance visual appeal and can communicate information quickly. High-quality, relevant visuals help reinforce the message and create emotional connections. Designers should select images that are contextually appropriate, visually sharp, and stylistically consistent with the overall design.

Color

Color significantly influences mood and user perception. Understanding hue, saturation, and value—as well as color theory principles such as complementary and analogous schemes—allows designers to create visually engaging and accessible compositions. Color is also vital for contrast, emphasis, and brand identity.

Layout and Composition

Layout and composition provide structure and flow to a design. Effective layouts utilize key principles such as proximity, alignment, contrast, repetition, and whitespace. These principles ensure that elements are logically organized, easy to read, and aesthetically balanced, enhancing both usability and visual harmony.

2.1.2 UI/UX Design

User Interface (UI) and **User Experience (UX)** design are fundamental components in the development of digital products, particularly websites and applications. **UI design** focuses on the visual aspects of a product—such as layout, typography, color schemes, and interactive elements—ensuring the interface is aesthetically pleasing and intuitive. In contrast, **UX design** addresses the overall experience a user has while interacting with a system, including usability, accessibility, and satisfaction. A well-crafted UI contributes to a positive UX, but the two disciplines, while closely related, involve distinct processes and goals. Effective UI/UX design plays a crucial role in reducing user frustration, enhancing engagement, and improving the overall functionality of web platforms [5].

2.2 Software Engineering

2.2.1 Introduction to the Software Engineering Process

The **software engineering** process is a structured approach to designing, developing, testing, and maintaining software systems. It provides a disciplined framework that ensures software is built systematically, meeting both user needs and technical standards. The process typically includes stages such as requirements analysis, system design, implementation, testing, deployment, and maintenance. A crucial early step in this process is the definition of software requirements, which sets the foundation for all subsequent phases [6].

2.2.2 Functional and Non-Functional Requirements

Functional requirements describe what the system must do. They define the specific functions, features, and interactions that the software should support, such as login functionality, data entry, search operations, or report generation. These requirements are essential for shaping the system's behavior and ensuring it delivers the expected services to users [7].
Non-functional requirements define how the system should perform its functions. They address attributes such as performance, usability, security, scalability, and maintainability. Although they don't describe specific behaviors, they are critical for delivering a quality system that meets user expectations and operational constraints [7].

2.3 Introducing inNuCE Lab

2.3.1 The inNuCE Lab

The **inNuCE Laboratory**, part of the EDA Group at Politecnico di Torino, is dedicated to advancing *neuromorphic computing*—a revolutionary paradigm inspired by the structure and functioning of the biological brain. By emulating neural processes, neuromorphic systems enable AI applications that are more adaptive, energy-efficient, and scalable.

inNuCE focuses on developing transformative applications in areas such as autonomous systems, health monitoring, edge computing, and smart environments. Their mission is to bridge the gap between neuroscience and artificial intelligence by creating innovative tools, computational models, and neuromorphic architectures. These solutions are designed to enhance the performance and energy efficiency of intelligent technologies, particularly in robotics and intelligent automation. An overview of the lab's core research themes and real-world use cases is illustrated in **Figure 2.1**.

A key outcome of their work is the *Cloud-Based Heterogeneous Prototyping Platform*, which provides an integrated environment for designing, building, and testing brain-inspired systems. The laboratory is funded by **EBRAINS-Italy** [8], a national research infrastructure that supports cutting-edge neuroscience and neuro-inspired technologies as part of the European EBRAINS initiative. This collaboration empowers inNuCE to contribute to shaping the future of intelligent, autonomous systems grounded in biological inspiration.

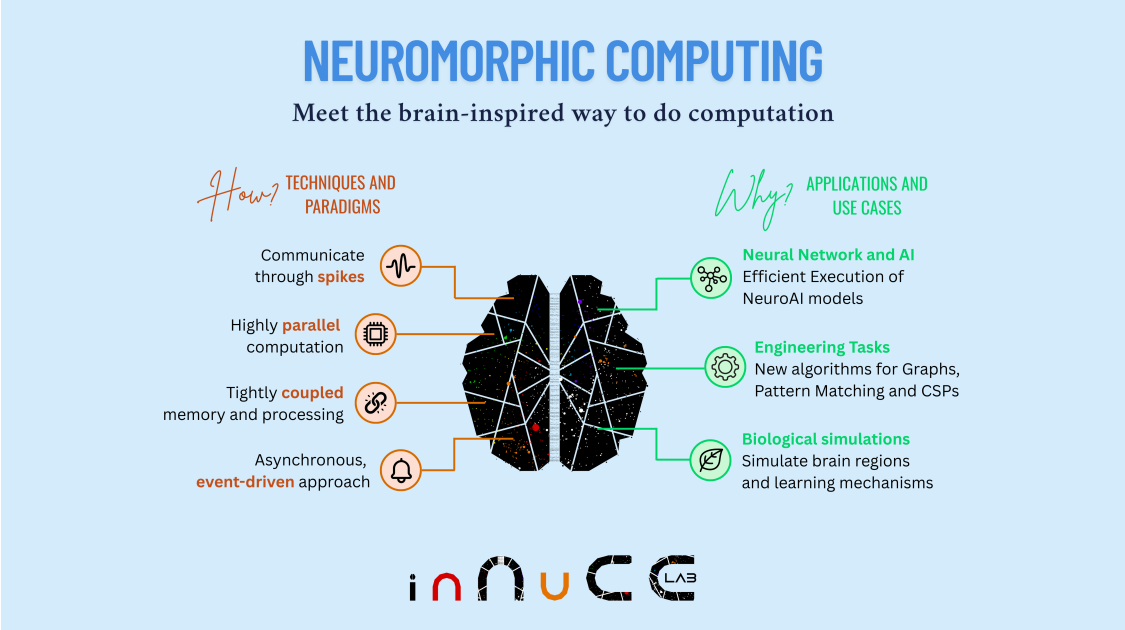

Figure 2.1: Core concepts and use cases of neuromorphic computing at inNuCE Lab

2.4 Artificial Intelligence Tools

The integration of **Artificial Intelligence** (AI) into web development presents significant opportunities for businesses and developers by enabling the creation web applications, AI helps companies better respond to user needs and behaviors—ultimately enhancing their competitiveness in the digital marketplace. AI technologies automate many routine development tasks such as code generation, layout design, and content creation. This not only accelerates the development process but also frees developers to focus on more complex and creative challenges, leading to increased productivity and innovation in the workflow [9].

In the following section, I will describe some of the most commonly used AI tools in modern web development.

2.4.1 ChatGPT

ChatGPT is an **Artificial Intelligence (AI)** conversational agent developed by OpenAI, based on **Generative Pretrained Transformer (GPT)** models. Leveraging deep learning techniques, it processes natural language prompts and generates contextually relevant, coherent responses. In the domain of web development, ChatGPT supports a wide range of tasks, including content generation, code writing and debugging, ideation, and technical documentation.

As of 2025, ChatGPT integrates multiple GPT models, each with varying levels of accessibility and performance [10]:

- **GPT-3.5**: This model is freely available to all users. It is capable of handling general conversational tasks, basic writing, and simple programming assistance.
- **GPT-4.5 (also known internally as o4 or o4-mini)**: Accessible through a paid **ChatGPT Plus** or **Pro** subscription, this model demonstrates significantly improved capabilities in reasoning, structured writing, and understanding complex prompts.
- **GPT-4.1**: Available exclusively via the OpenAI API, this variant is optimized for advanced programming tasks and offers robust logical processing, making it well-suited for backend development and algorithmic problem-solving.

.

2.4.2 Uizard

Uizard represents a significant advancement in user interface design tools by offering an intuitive, AI-assisted platform that streamlines the transition from concept to interactive prototype. Particularly notable is its **Wireframe Scanner** feature, which allows users to upload hand-drawn interface sketches and automatically convert them into digital wireframes as shown in **Figure 2.2**. This functionality greatly accelerates the early stages of the design process, enabling rapid iteration and reducing the need for manual reproduction of initial ideas.

Uizard's user-friendly editor further supports high-fidelity mockup creation without requiring prior design or coding experience, making it an accessible tool for both technical and non-technical stakeholders involved in the development of digital products. Although Uizard was not used in this project, its existence demonstrates how AI enables more accessible design practices in web development [11].

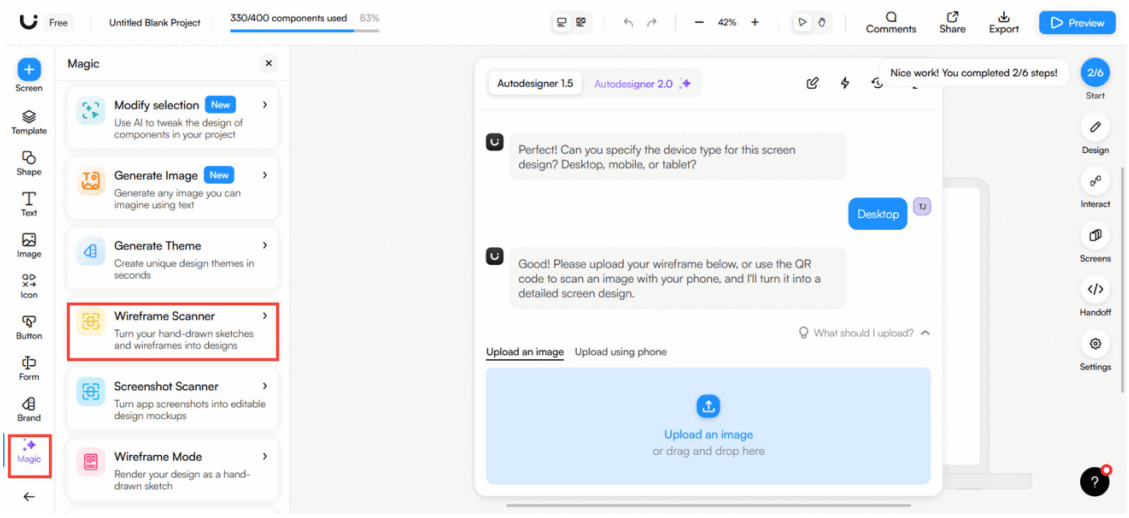

Figure 2.2: Turn Hand-Drawn Wireframes into Editable Mockups with Uizard

2.4.3 CodeParrot

CodeParrot is an AI-powered developer tool developed by the startup CodeParrot AI, founded in 2022 by Vedant Agarwala and Royal Jain [12]. Unlike traditional code generation models trained on GitHub repositories, CodeParrot focuses on transforming design elements—such as Figma components or even screenshots—into clean, production-ready front-end code. It acts as a **design-to-code copilot**, preserving component structure and aligning with the developer's existing UI libraries and codebase.

CodeParrot is fully integrated with **Visual Studio Code (VSCode)**, allowing developers to use it seamlessly within their existing workflow. This tight integration makes it easy to invoke the tool directly while coding, reducing context-switching and increasing efficiency.

Available as a VSCode plugin, CodeParrot enables teams to accelerate the front-end development process by automating UI implementation and reducing manual handoff work between designers and developers. As illustrated in **Figure 2.3**, the tool supports rapid prototyping by intelligently parsing design components and generating semantic, editable code blocks.

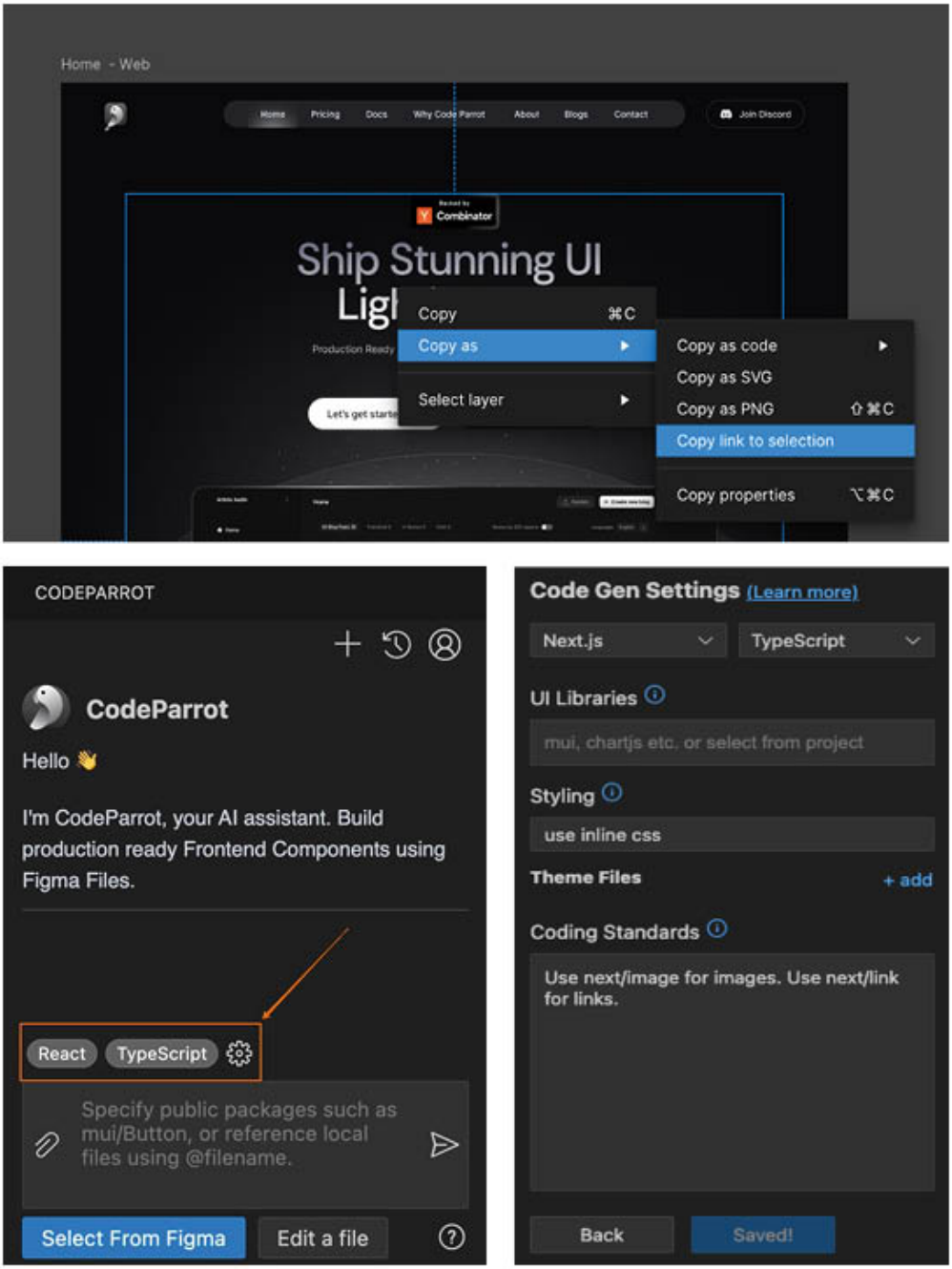

Figure 2.3: From selecting Figma component to code generation — CodeParrot

2.4.4 Vercel

Vercel was used in this thesis as the primary platform for hosting the frontend of the website. It provided a streamlined and efficient deployment pipeline, enabling automatic previews and updates directly from the connected **GitHub repository**. This continuous integration and delivery setup allowed for rapid iteration and real-time collaboration throughout the development process. Although Vercel itself was not the subject of experimentation, its use contributed significantly to the workflow by simplifying deployment and ensuring optimal performance. As a developer-oriented platform, Vercel exemplifies current trends in web development infrastructure that emphasize speed, scalability, and ease of use [13].

Figures 2.4, 2.5, and 2.6 demonstrate the process of creating a laboratory website on Vercel, illustrating respectively the stages of prompting, the resulting output, and the generated code.

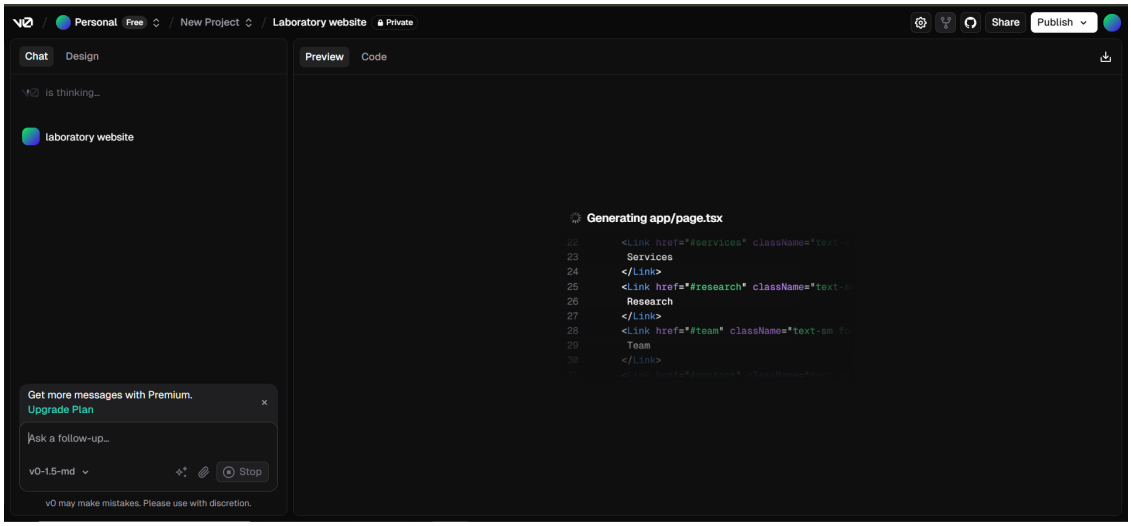

Figure 2.4: creating laboratory website in prompt-Vercel's

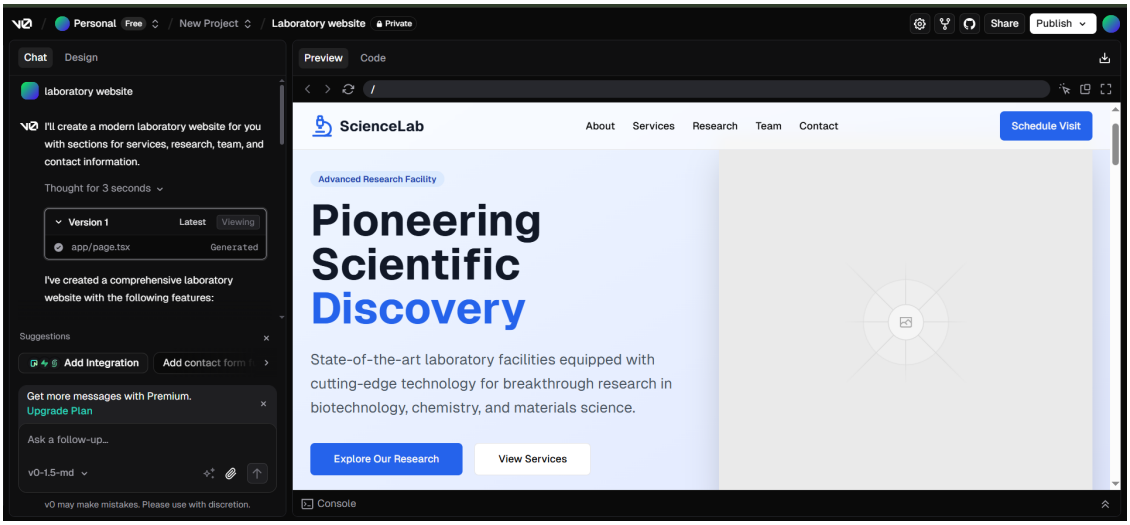

Figure 2.5: result-Vercel's

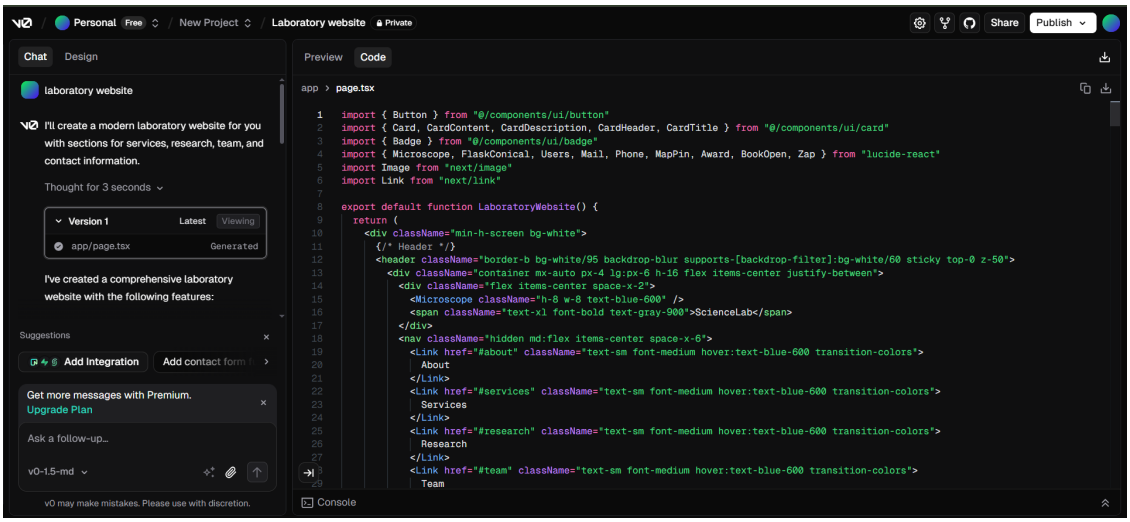

Figure 2.6: code generation in Vercel

2.5 Pipeline Tools

This project followed a structured development pipeline using a combination of design, development, and deployment tools. Each tool played a specific role in transforming initial ideas into a functional and interactive website.

2.5.1 Figma

Figma is a **web-based UX design** tool that enables teams to collaborate in real time by accessing a single, live **URL**. As a key part of UX design, focuses on creating

meaningful and user-friendly experiences by understanding user needs, researching case studies, and testing designs—Figma allows designers to build wireframes and create interactive prototypes without coding. Additionally, Figma's prototyping tool helps developers understand how the user experience should look and feel [14].

2.5.2 Adobe illustrator, Adobe photoshop

Adobe Illustrator was primarily used for creating **scalable vector graphics** (SVGs), such as icons, diagrams, and illustrations. These assets were particularly important in sections of the website that needed visual explanations of research topics or technical concepts. Because vector graphics are resolution-independent, they ensured visual clarity on both desktop and mobile displays [15].

Adobe Photoshop was used for image editing and optimization. This included tasks such as cropping, resizing, background removal, color correction, and compressing image files for web performance. Photoshop helped prepare high-quality JPEG or PNG images while keeping their size optimized for faster loading times—an important aspect of user experience and SEO [16].

Together, these tools enabled the integration of custom, polished visual elements into the web interface. While Figma was central for layout and UI structure, Illustrator and Photoshop added a layer of visual identity and branding to the final product, especially in image-heavy or illustrative sections of the website.

2.5.3 Online Editing Tools (Supporting Resources)

In addition to professional design and development tools, several online editing platforms were used throughout the process for specific quick tasks. For example, background removal tools like **remove.bg** [17] were employed to isolate subjects from images without manual masking, which helped prepare cleaner visuals for the site. Tools such as **Convertio** [18] was used for image compression and format conversion, ensuring that assets were web-optimized and loaded quickly. These tools, while simple, significantly improved workflow efficiency and reduced manual effort in the content preparation phase.

2.5.4 Visual Studio Code

Visual Studio Code (VSCode) is a lightweight, open-source source code editor developed by Microsoft, widely adopted by developers for its flexibility, performance, and extensive extension ecosystem. It supports multiple programming languages, including JavaScript, TypeScript, Python, and C++, and provides features such as syntax highlighting, IntelliSense (code completion and suggestions), integrated

terminal, Git version control, and debugging tools. Its broad compatibility and user-friendly interface make it a popular choice for both front-end and back-end development workflows. In the context of this project, VS Code was used as the primary development environment due to its seamless integration with React and other modern web technologies [19].

2.5.5 Builder.io

Builder.io is a visual **Content Management System (CMS)** that enables developers and designers to collaboratively build websites through a visual interface. More specifically, it is a headless **CMS**, meaning the content and the presentation layer are decoupled, offering greater flexibility for modern front-end frameworks such as React.

One of Builder.io's key features is its ability to convert Figma designs into clean React components as shown in **Figure 2.7**, streamlining the handoff between design and development. Additionally, its built-in AI integrations assist in generating content and layout suggestions, enhancing both productivity and creativity during the development process.

Serving as a bridge between UI/UX design and front-end implementation, Builder.io empowers non-developers—such as marketers and designers—to update and manage front-end content dynamically without writing code. This functionality is especially valuable for teams aiming to accelerate development cycles and reduce dependency on engineering resources. Brands like **Everlane** [20] and **Gymshark** [21] utilize Builder.io's visual CMS and AI-powered tools to maintain agility in their digital content workflows [22].

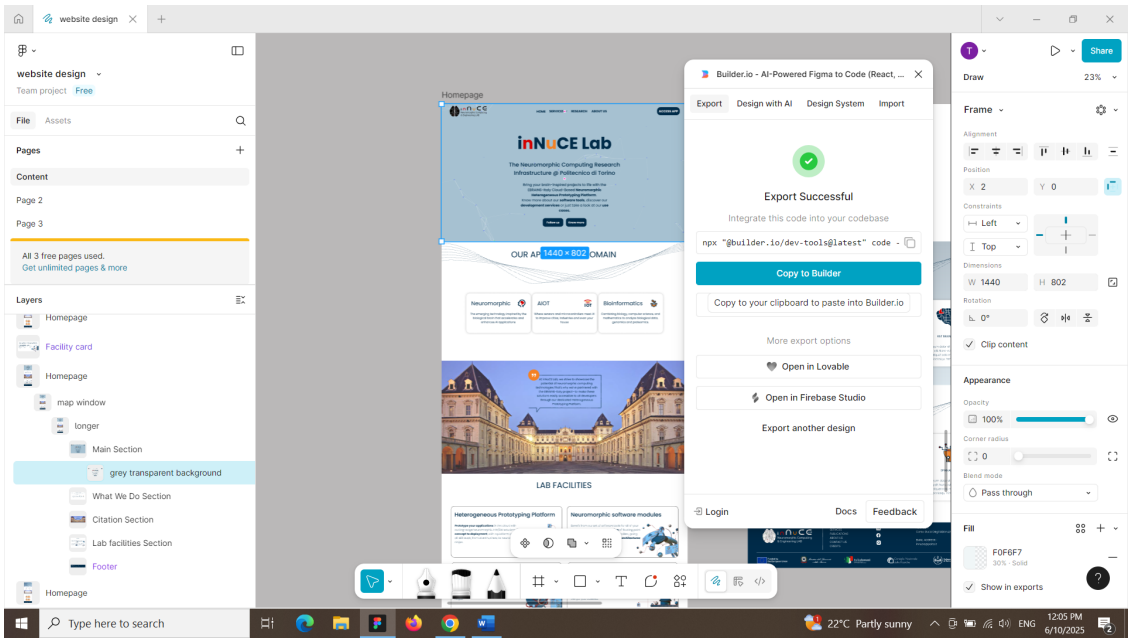

Figure 2.7: builderio plugin

2.5.6 React framework

React is a JavaScript library for building user interfaces that simplifies the creation of interactive UIs through a declarative approach, where developers design views for each state in the application, and React efficiently updates and renders the right components when the data changes. Its component-based architecture allows for building encapsulated components that manage their own state, enabling the composition of complex UIs. Since component logic is written in JavaScript instead of templates, passing rich data and keeping the state out of the DOM [23] becomes easier. Additionally, React's *Learn Once, Write Anywhere* philosophy allows developers to create new features without rewriting existing code, and it can be used to render on the server with **Node** or build mobile apps through **React Native** [24].

To initiate the development of the React-based web application, the **Vite** build tool was used as an alternative to the traditional Create React App setup. Vite is a modern frontend build tool that offers significantly faster startup and hot module replacement by leveraging native ES modules and a highly optimized development server. It is particularly well-suited for modern JavaScript frameworks like React. The React project was initialized using the command `npm create vite@latest`, followed by selecting *React* and *TypeScript* as the desired template. This command scaffolds a minimal and efficient project structure, enabling faster build times and

improved developer experience. Once the setup was complete, the project dependencies were installed using `npm install`, and the development server was launched with `npm run dev`. Vite's integration with modern tools like TypeScript, JSX, and environment variables made it an ideal choice for building a high-performance, scalable front-end application.

2.5.7 tsParticles

To enhance the visual appeal and interactivity of the website, **tsParticles** was used—a lightweight, customizable JavaScript library that integrates seamlessly with React. tsParticles enables the creation of dynamic particle effects such as animated backgrounds, bubbles, floating shapes, and more. These effects were implemented to improve the user experience on landing sections and key visual areas of the site.

The library supports a wide range of configurations through simple **JSON**, allowing for full control over properties like particle speed, size, shape, color, and interaction (e.g., particles responding to mouse movement). By using the react-tsparticles wrapper, the animations could be embedded directly into React components without performance issues [25].

Chapter 3

Materials and Methods

This section presents the practical steps followed during the development of the website for the Neuromorphic Computing Laboratory at Politecnico di Torino. It explains the learning process to gain the necessary skills and knowledge, and the detailed pipeline that describes each phase of the website's design and implementation.

3.1 Design and Learning Process

3.1.1 Planning and Goal Definition

The development process began with a clear definition of the website's purpose and its intended audience. The objective was to create a platform for the Neuromorphic Computing Laboratory at Politecnico di Torino that could serve both experts in the field and individuals with no technical background. To guide this process, a comparative analysis was carried out on websites of similar research laboratories, particularly those affiliated with other universities. This analysis informed the foundational structure of the site, helping us identify which sections were necessary and how users would navigate between them. Based on this, a sitemap was created that outlined the required pages and ensured a logical, coherent flow across the entire website. The aim was to provide an accessible yet informative platform with an intuitive user journey.

Apart from the general pages—such as the *Home Page*, the *About Us* section, and those that provide information about the facilities and services—we decided to make the *Research* section of the website more interactive and accessible to a wider audience beyond just researchers. In the *Research* section, instead of simply providing a link to each paper, users will also have access to a clear and concise explanation of each paper. Additionally, a short vocal summary, similar to a podcast, will be available to enhance accessibility and engagement.

3.1.2 Visual Design

Color Palette

After the planning phase, the next step was to establish the website's visual identity. The design was developed using **Figma** 2.5.1, a collaborative interface design tool that allowed for precise layout control and efficient prototyping. The color palette was carefully chosen based on *Politecnico di Torino*'s graphic guidelines [26], ensuring consistency with the institution's branding. The primary colors used were `#002B49`, a dark blue representing PoliTo's official website, and `#F48124` , the university's signature orange. These colors were maintained throughout the design to create coherence while incorporating a dynamic visual language typical of modern technological platforms [26].

Typography

Typography was unified using the *Poppins font*, aligning with the university's official branding guidelines. Titles for each section were set at *48px* with a weight of 500, while other text sizes varied between *16px* and *32px*, adjusting in weight based on aesthetic and readability considerations [26].

Icons

Icons and illustrations were either sourced from design platforms or custom-edited using **Adobe Illustrator** to align with the desired aesthetic, maintaining consistency with the color scheme and overall design approach. The sources of these graphical elements are acknowledged in the credits section of the page, listing icons provided by *Flaticon* [27], *WordArt* [28], and *Freepik* [29]. This ensures proper attribution and transparency regarding the visual assets used in the website's design.

3.2 Functional Requirements

The first type of requirements considered are the Functional Requirements, which are typically the easiest to understand because of their clear and practical nature. They outline the essential features and operations that the application must deliver to support the organization's needs. While they form the foundation of the system, they can still involve complex technical implementation. These requirements are

usually prioritized by the organization and initially provided as a basic list, which the engineer then organizes and refines [30].
Functional requirements have been further categorized into three classes described below.

High Priority Functional Requirements

The high priority functional requirements represent the essential features that ensure the core functionality and structure of the website. These are necessary for the system to fulfill its primary purpose and enable users to navigate and interact with key content effectively.

FRN Name	**Description**
Navigation Bar	Display a fixed, consistent navigation bar across all pages for seamless user navigation.
Homepage Overview	Provide a summary of the lab's mission, application domains, and facilities.
Use Case Cards	Present each use case as a clickable card that opens a dedicated details page.
Use Case Filtering	Enable filtering of use cases based on category or technology to enhance discoverability.
Research Page Listing	List publications by year with summaries and links to external resources.
Common Footer	Display a consistent footer across all pages to support navigation and accessibility.

Medium Priority Functional Requirements

The medium priority requirements enhance usability and provide additional value to users. While not critical for basic operation, they improve the overall experience and help users access more detailed or organized information.

FRN Name	**Description**
Infrastructure Infographic	Display an infographic explaining technologies and workflows used in the prototyping platform.
Tool Information Cards	Display software/hardware tools as clickable cards linking to external documentation.
Team Member Profiles	Show team members with profile pictures in the "Meet Our Team" section.

Low Priority Functional Requirements

The low priority requirements include non-essential enhancements that contribute to the visual appeal and interactivity of the website. These features are optional and can be implemented if time and resources allow, without impacting the system's fundamental performance.

FRN Name	Description
Hover-Linked Team Profiles	On the About Us page, hovering over a team member's photo reveals a LinkedIn profile link.
Dynamic Visual Effects	Use of animations and transitions (like particle animation) for visual appeal, not affecting core usability.
Informational Design Elements	Use of infographics and visual design elements (e.g., in Infrastructure page) to support content presentation.

3.3 Non-Functional Requirements

The Non-Functional requirements define the overall quality, performance, and usability of the website rather than the specific features it offers. These requirements ensure that the website not only functions correctly but also delivers a professional, smooth, and user-friendly experience aligned with the lab's identity and communication goals.

Usability

The website is designed to be intuitive and accessible. Features such as a *fixed navigation bar*, a *consistent footer* across pages, and organized content sections (e.g., *Homepage, Services, Research, About Us*) aim to ensure users can easily understand and explore the site. The visual hierarchy and layout support clear information delivery, especially for new visitors and stakeholders.

Performance

The presence of interactive elements like particle animations, filtering systems, and media content (e.g., images, links, cards) implies the need for fast load times and smooth transitions. These elements should perform efficiently across modern browsers without affecting responsiveness or navigation.

Responsiveness and Compatibility

The website must adapt well to various screen sizes and devices (desktops, tablets, smartphones), ensuring a consistent experience regardless of how users access it.

The layout, images, cards, and hover effects should scale appropriately to maintain visual clarity and usability.

Accessibility

While not directly mentioned, accessibility can be inferred as a requirement for a public research lab website. This includes clear navigation, readable text, alt text for images, and keyboard navigability to support diverse user needs.

Maintainability

The site should be easy to update as research projects evolve, team members change, or new infrastructure is added. Using modular components like cards and filter systems supports future scalability and content management.

Reliability

The website should be consistently available and stable, without broken links or errors—especially important when directing users to external documentation or showcasing research credibility.

3.4 Logo Design

3.4.1 Alternative Logos

To design the logo, we began by considering the core concept of neuromorphic computing, which draws inspiration from the structure and function of the human brain. To reflect this, we decided to incorporate a brain-like element as a central visual component, representing both the biological roots and the intelligent processing capabilities of the field. Since our project is closely aligned with **Ebrains-Italy** [8], a major platform in the neuromorphic research ecosystem, we chose to maintain a similar visual format to the Ebrains-Italy logo ,shown in **Figure 3.1**. This alignment supports visual continuity and conveys relevance within the broader research community. Additionally, we integrated the *Fibonacci pattern*, symbolizing the natural growth of complex systems from simple beginnings. This not only highlights the organic inspiration behind neuromorphic architectures but also reflects the gradual and structured evolution of intelligence within such systems.

Figure 3.1: Ebrain-Italy Logo

In the following figures, various alternative logo designs for Innuce are presented in Figures 3.2 ,3.3 ,3.4 ,3.5 .

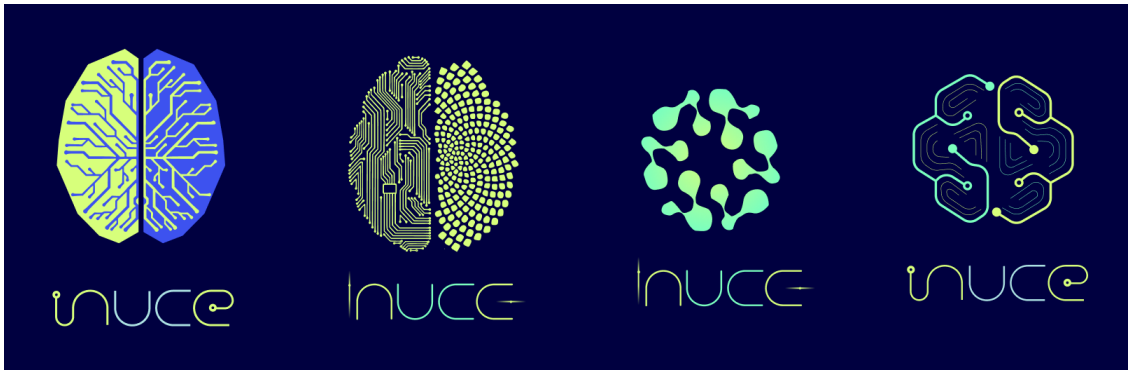

Figure 3.2: Alternative Logo Designs for inNuCE, two-color version

Figure 3.3: Alternative Logo Designs for inNuCE,multicolor version

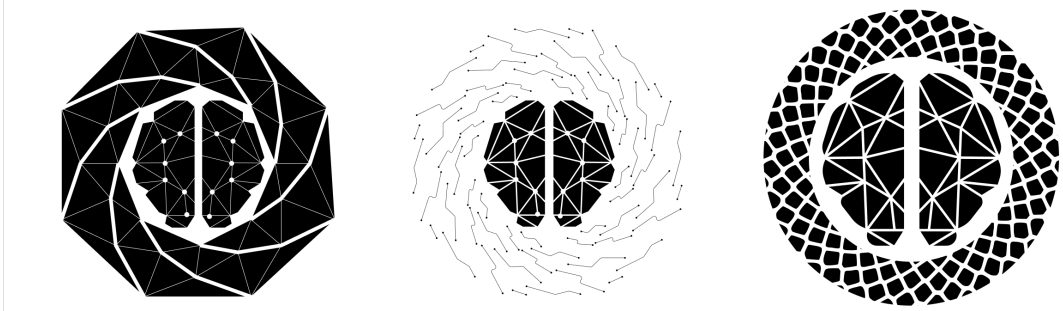

Figure 3.4: Alternative Logo Designs for inNuCE, Monochrome version-1

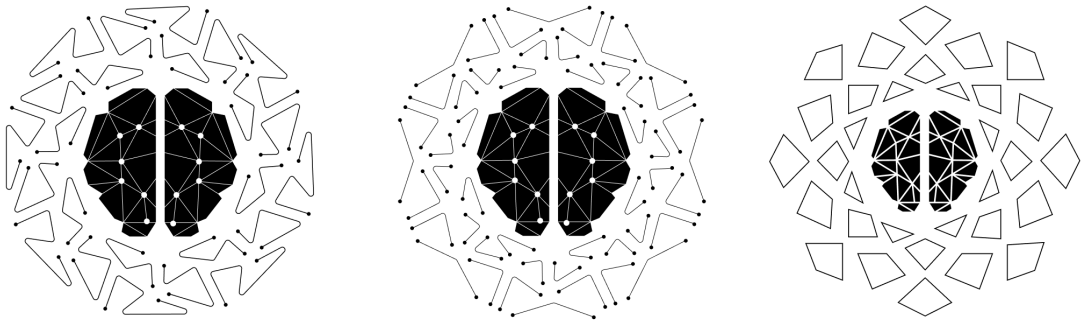

Figure 3.5: Alternative Logo Designs for inNuCE, Monochrome version-2

3.4.2 Final Logos

In the final stage of the logo design, we chose to preserve the brain-like structure reminiscent of the **Ebrains-Italy** logo to maintain visual consistency within the neuromorphic research landscape. While we initially explored incorporating the **Fibonacci** pattern directly, we eventually opted for a background element that subtly evokes the form of a **galaxy**—a natural spiral structure that reflects the spirit of Fibonacci-inspired growth without replicating the exact mathematical pattern. This design choice supports the idea of gradual, organic complexity emerging from a central core, much like neuromorphic systems evolve. To ensure adaptability across various platforms and contexts, we developed three different logo variations, each optimized for different sizes and usage scenarios.As illustrated in **Figure 3.6**, the black version of the logo is shown, while **Figure 3.7** presents the white version. The choice between the two depends on the background color to ensure optimal visibility and contrast.

Figure 3.6: Final inNuCE Logo Design-black

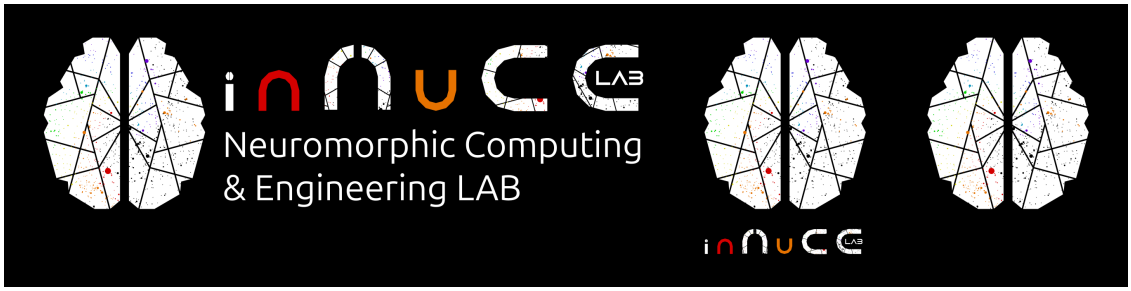

Figure 3.7: Final inNuCE Logo Design-white

3.5 Development Workflow

3.5.1 Front-End Development

With the visual framework in place, the implementation phase began by setting up the front end using the **React framework**. React was selected for its modular component-based architecture and flexibility in handling dynamic content. The initial focus was on translating the Figma design into working code.

To support this, **Builder.io** was integrated into the workflow. The Builder.io Figma plugin was installed by opening Figma, navigating to the Plugins section, and searching for Builder.io under the Plugins and Widgets tab. Once found, it was added via the Save button. Alternatively, the plugin could be installed directly from the Builder.io Figma Plugin page.

To import a design, selected Figma frames were exported using the Builder.io plugin. After opening the plugin from the Figma toolbar and clicking the Export Design button, the plugin processed and analyzed the frames. Once completed, different AI-generated code versions were presented. The TypeScript with CSS option was selected, enabling direct conversion of static components into React-compatible code.

This integration significantly accelerated the development of static parts of the

website and provided insight into how UI elements are structured within React. While Builder.io was especially helpful for a beginner working with React, the generated output often required refinement. Interactive components or those requiring complex state management—such as the navigation bar or dynamic content blocks—were manually coded, with support from online learning resources and AI assistance from **ChatGPT(GPT-4o model)**.

To further enhance the user experience with dynamic and engaging visuals, the tsParticles library was also integrated into the React environment. The installation was done via the terminal using the command `npm install @tsparticles/react`. Since tsParticles needs to be initialized only once per application lifecycle, the engine was loaded using the loadFull function in the main entry point (typically **App.tsx**). Additionally, the `loadPolygonMaskPlugin` was imported to enable polygon-shaped particle constraints defined by SVG files.

This polygon masking technique was applied on the *Use Cases page*, where particles were restricted to animate within a user-defined polygon shape. This created a visually striking animation that aligned perfectly with the SVG layout, adding interactivity and enhancing aesthetic appeal.

In contrast, more traditional background particle effects were used on the *Home page* and *Coming Soon* sections (e.g., the *Access App page*). These simpler animations did not involve polygon masks and were rendered using the `<Particles />` component with tailored configurations defined in modular .ts files (e.g., *homeParticlesOptions.ts*).
These configuration files allowed customization of properties such as particle color, speed, interactivity, and background opacity, and were imported into the corresponding React components as shown in **Figures 3.1 3.8** This modular structure ensured that particle configurations remained reusable and easy to maintain across different sections of the site. While polygon-masked animations emphasized creativity on content-heavy pages, the subtler background effects contributed a modern visual layer without overwhelming the user interface.

This combination of automated tools and manual coding made development faster and helped build a better understanding of front-end technologies.

Listing 3.1: UsecasesHeader.tsx - Implementation of tsparticles in Usecase Header

```
import "../UseCases/UsecaseHeader.css";
import React, { useMemo } from "react";
import Particles from "@tsparticles/react";
import { type Container, type ISourceOptions } from "@tsparticles/
    engine";
```

```
import { BACKGROUND } from "./constantUc";

const UsecasesHeader = () => {
  const options: ISourceOptions = useMemo(() => BACKGROUND, []);

  return (
    <section className="UsecasesHeader">
      <div className="UcHeaderContainer">
        <h1>Discover our Use Cases</h1>
        <h3>
          InNuce labs offer a complete , easy to use Prototyping
             Platform for
          all Neuromorphic needs.
          <br />
          from the development ,with the most updated and
             innovative tools and
          libraries, to the development, done on the most popular
             and
          cutting-edge hardware.
          <br />
          <b>
            Just bring your own idea : the computer power, the
               tools and the
            hardware is on us.
          </b>
          <br />
        </h3>
        <button onClick={() => {}} tabIndex={0} aria-label="
           Discover Use Cases">
          Discover Use Cases
        </button>
      </div>

      <Particles className="particleUc" options={options} />
    </section>
  );
};

export default UsecasesHeader;
```

inNuCe-website > src > components > UseCases > TS constantUc.ts > BACKGROUND > backgroundMask

```ts
import { type ISourceOptions } from "@tsparticles/engine";
import SD from "../../assets/logo/Vector.svg";

export const BACKGROUND: ISourceOptions = {
  autoPlay: true,
  background: {
    image: "",
    position: "50% 50%",
    repeat: "no-repeat",
    size: "cover",
    opacity: 1,
  },
  backgroundMask: {
    composite: "destination-out",
    cover: {
      opacity: 1,
      color: {
        value: "",
      },
    },
    enable: false,
  },
  clear: true,
  defaultThemes: {},
  delay: 0,
  fullScreen: {
    enable: false,
    zIndex: 0,
  },
  detectRetina: true,
  duration: 0,
  fpsLimit: 120,
  interactivity: {
    detectsOn: "window",
    events: {
      onClick: {
        enable: false,
```

Figure 3.8: Particles Configuration (TypeScript) for Visual Effects

3.5.2 Back-End Development

Simultaneously with front-end development, the back-end was designed to handle the dynamic aspects of the site and manage content efficiently. **Express.js** [31] was chosen as the runtime environment for server-side development due to its scalability, performance, and seamless integration with JavaScript-based front ends. A suitable database management system was selected based on project needs — such as storing publication metadata, research summaries, and team member profiles — ensuring that the site could serve dynamic content reliably. Integration between the database and front-end ensured that any updates to data would be reflected in

real-time on the user interface, enabling efficient content management.

3.6 Website Structure and Design

The website was designed in Figma to create a modern, accessible, and visually consistent experience that reflects the identity and research focus of the inNuCE Laboratory. Each page was designed with clarity, modularity, and user engagement in mind. This section breaks down each page and its components.

3.6.1 Homepage

The homepage, **Figure 3.9**, serves as the user's first point of contact with Innuce Lab's digital identity. In Figma, the homepage was crafted to immediately convey innovation and interactivity.

Figure 3.9: Home Page

Landing Section

Intent: Create a strong first impression through a visually engaging **hero section**.
Design Elements:

- A lightweight particle animation background (`tsParticles` in React) symbolizes neural activity. Initial versions used a large GIF file and later an MP4 video as backgrounds, but both caused slow loading times and high memory usage due to their size.
- Title and subtitle are center-aligned with ample spacing to draw focus.
- A call-to-action button (*Discover More*) guides users down the page.
- The **Politos** font was selected to balance modernity with academic professionalism.

Application Domains Section

Design Features:

- This section is designed using horizontal cards or blocks, each representing a research or application domain (e.g., Neuromorphic, AIOT, Bioinformatics).
- Each block uses iconography and short text, making the section scannable and intuitive.

Quote Section

Design Features:

- Designed as a quote section from the inNuCE group, paired with a representative image of Politecnico di Torino's Valentino campus, where the lab is based. .

Facilities Overview

Design Features:

- Arranged using a layout grid of four cards, each featuring relevant graphical visuals representing the lab's software modules and prototyping platform.
- This layout was designed in Figma to ensure scalability, visual balance, and clarity , enhancing both readability and responsiveness.

Navigation Bar

Design Features:

- A sticky top navigation ensures persistent access to key pages.
- The bar includes hover effects and a logo placed in the top-left corner.
- Emphasis was placed on a clean design with minimal items to avoid overwhelming the user.
- On specific pages—such as the *Homepage* and *Services* subpages—the navigation bar incorporates a scroll-triggered visual effect, it starts with a transparent background and transitions into a solid background once the user begins scrolling.
- On pages where this design element is unnecessary, a standard solid background is used for consistency and simplicity.
- A button labeled *Access App*,**Figure 3.10**, currently under development, directs users to a dedicated page displaying a "Coming Soon" message over a dynamic particles background.

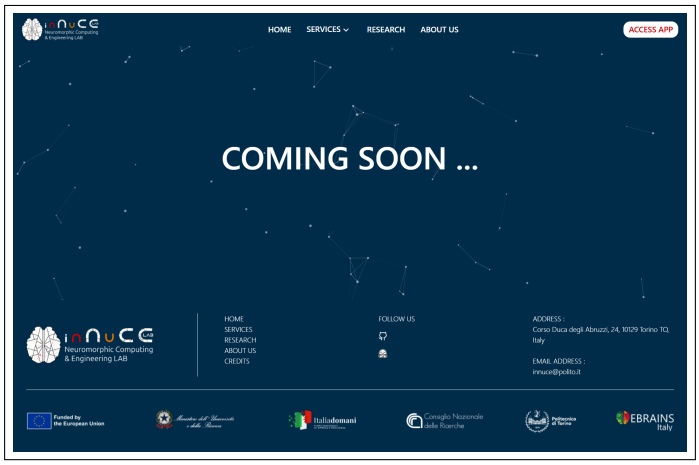

Figure 3.10: Preview of the Access App page (feature under development)

Footer

Design Features:

- Designed to mirror the navigation bar in simplicity.

- Includes contact information, logos of affiliations (e.g., Ebrains-Italy), and external links.
- Maintains the brand color palette and contributes to a cohesive end-of-page experience.

3.6.2 Services – Infrastructure

This page, **Figure 3.11**, introduces the lab's prototyping infrastructure and available technical resources. The design focuses on clarity, interactivity, and accessibility for both researchers and visitors.

Infographic Section

Design: A custom-designed linear infographic is positioned at the top of the page.

- **Purpose:** To visually summarize the lab's research pipeline — from initial idea conception to testing and eventual deployment.
- Created in Figma using vector shapes and smooth gradient flows, the infographic symbolically represents each process stage in a continuous, visually engaging format.

Tool Cards Section

Design: Positioned below the infographic, a responsive card layout displays the lab's key hardware and software tools.

- Each card includes:
  - A clear title,
  - A concise description,
  - clicking the title of each card directly navigates the user to the external documentation or detailed resource page.
- The cards were designed in Figma with careful attention to spacing and depth. Subtle shadow effects were applied to ensure visual separation between elements.

Design Intent

- Allow researchers and external visitors to quickly identify and access detailed information about the lab's technological resources.

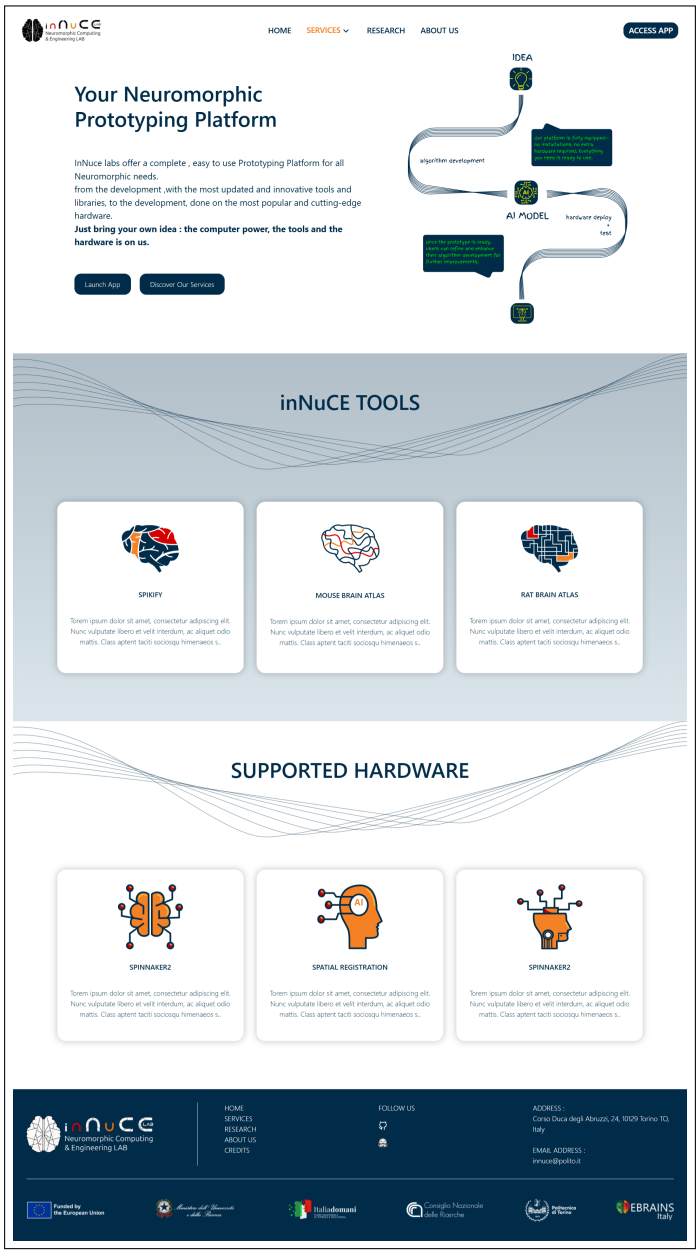

Figure 3.11: Services/Infrastructure Page

3.6.3 Services – Use Cases

This page,**Figure 3.12**, was designed as an interactive showcase of the lab's past and ongoing projects. It highlights a dynamic interface that supports exploration through filtering and modular layouts.

Figure 3.12: Services/Use Cases Page

Filtering System

A vertical filter bar is positioned on the left side of the *use case grid*, allowing users to refine their exploration based on technology used, application categories, and associated hashtags. These filters are implemented as *checkable options*, enabling multi-selection and easy toggling. Designed in Figma, the filter interface ensures high accessibility through strong contrast, generous spacing, and clear labeling. This layout also improves usability by keeping the filter controls visible

while scrolling through projects on larger screens.

Use Case Cards

Rather than using a repetitive uniform grid, the layout is inspired by the golden ratio. This creates visual rhythm by placing two smaller cards side-by-side with the combined width equivalent to a larger card placed beside or beneath them. The resulting variation provides a more engaging and aesthetic structure compared to equal-sized cards in a traditional grid.

The background behind the cards features a unified linear vector graphic designed in Illustrator. This vector design subtly echoes the lab's logo and the linear layout seen in the Infrastructure page's infographic and other sections. Transparency is applied selectively within the vector pattern, adding visual depth and playful layering without distracting from the content.

Each project is presented as a clickable card component. The cards include:

- A clear project title
- A concise description
- A button navigating to the use case details page

The overall arrangement and visual style of these cards can be seen in **Figure 3.13**, which illustrates the golden ratio–inspired layout and background vector design.

Figure 3.13: Use Cases Layout

Use Case Detail Page

Upon selecting a card, users are directed to a dedicated detail page for that specific project. This page, **Figure 3.14**, includes the following elements.

Figure 3.14: Services/Use Cases/Use Case Detail Page

- A prominent hero image related to the project (e.g., a photograph, chart, or rendered visualization)
- A clear project title and subtitle
- Paragraphs of descriptive content, often organized using a two-column layout for improved readability
- A list of hashtags related to the project, such as the technologies used, application domains, and research themes,etc; these serve as interactive filters on the previous page

3.6.4 Research

This page ,**Figure 3.15**, serves as a dynamic hub for the lab's publications and AI-enhanced summaries. The design focuses on clarity, interactivity, and intelligent content presentation.

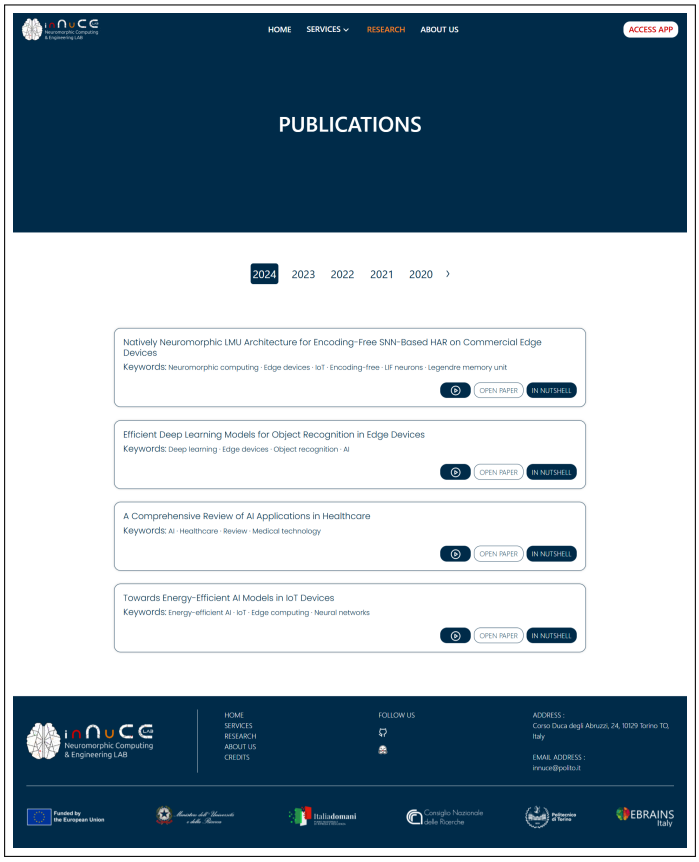

Figure 3.15: Research Page

Year-Based Grouping

Publications are grouped by year using collapsible accordion sections, allowing users to efficiently navigate a large number of entries.

Publication Cards

Each publication is presented as a structured card, containing:

- **Title**
- **Authors**
- **DOI or external link** (typically pointing to IRIS)
- **Two interactive features:**
  - *In a Nutshell* button: Opens a modal or expands a collapsible section with an AI-generated summary presented in four concise parts. (introduction, methods, results, conclusion)

– *Play* icon: Triggers a podcast version of the summary, enabling users to listen instead of read.

3.6.5 About Us

The *About Us* page, **Figure 3.16**, provides background information about the lab and introduces the team. The content is presented in a clear and accessible way.

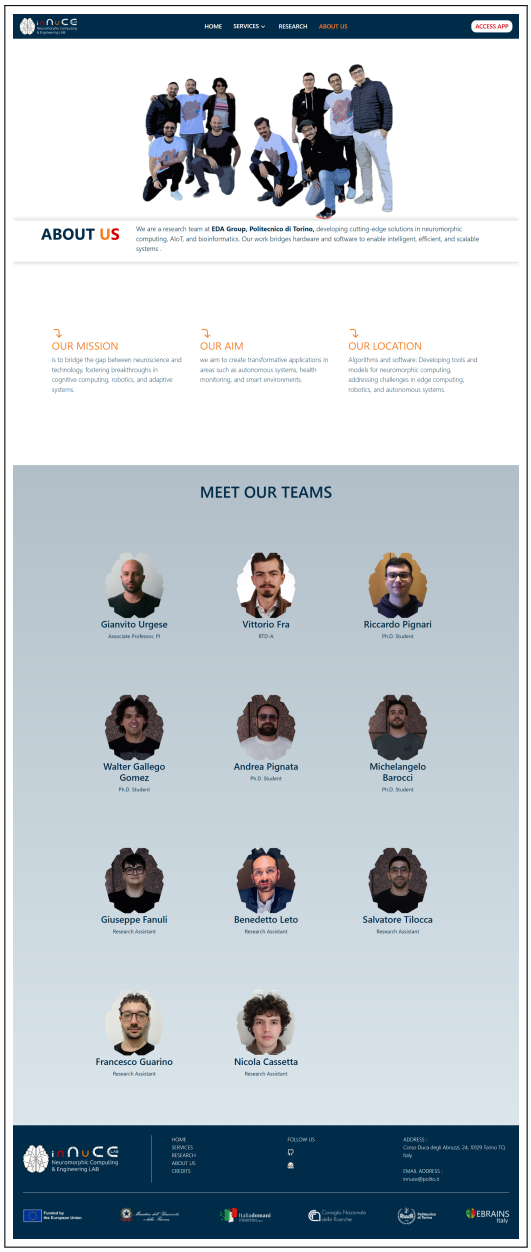

Figure 3.16: About Us Page

Mission and History

The first section shows a group picture of the Innuce team, edited to match the website's graphic style. It then explains the lab's mission, goals, and where it is located.

Meet Our Team Section

This section presents individual team members through profile cards, each including:

- A professional photo displayed in a layout inspired by the Innuce logo
- Full name and title for easy identification
- A LinkedIn icon that appears on hover, allowing users to quickly access the person's professional profile if interested

In Figma, the hover effect was simulated using overlay states, making it possible to prototype and preview the interactive behavior of the final implementation.

Design Focus

- The visual style was designed to create a personal connection with visitors, highlighting the people behind the research.
- A friendly yet professional tone was used — in line with the rest of the site, but slightly warmer to emphasize human presence and collaboration.
- The profile card layout helps users recognize team members at a glance, while the hover interaction adds a simple and intuitive way to connect.

3.7 Builder.io Output vs Final Output

As previously mentioned, Builder.io proved to be a useful starting point, particularly for someone with limited coding experience. It facilitated the creation of basic components and offered a quick way to begin the implementation process. However, the tool shows clear limitations when dealing with more complex components, especially those designed in Figma. In such cases, the auto-generated code often becomes either overly complex or structurally incoherent, requiring significant manual intervention.

This is clearly illustrated in **Figure 3.17**, which shows the React code automatically generated by Builder.io for the use case cards section. While the layout

roughly matches the original design, the code lacks semantic structure and introduces unnecessary nesting and inline styles. In contrast, the manually refined version—shown in **Figure 3.18**—demonstrates a cleaner component hierarchy, improved readability, and adherence to best practices, resulting in a final implementation that more accurately reflects the Figma design.

Additionally, **Figure 3.19** presents the folder structure generated by Builder.io for the InNuCE tools section located in the *Services/Infrastructure* area. This structure served as a foundation but was later reorganized to support maintainability, modularity, and scalability as the project evolved.

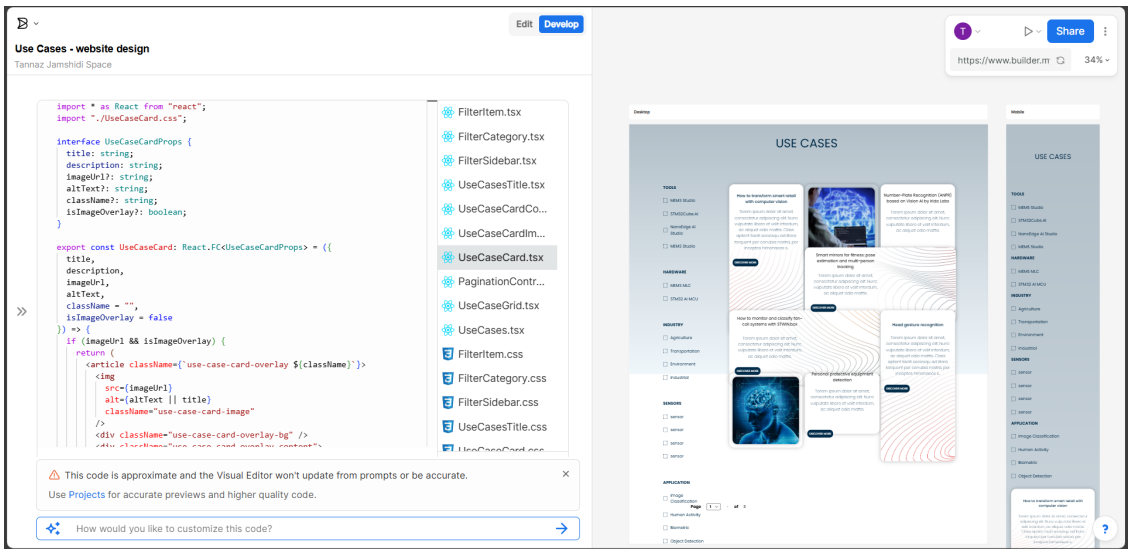

Figure 3.17: codes generated in builder.io - use cases

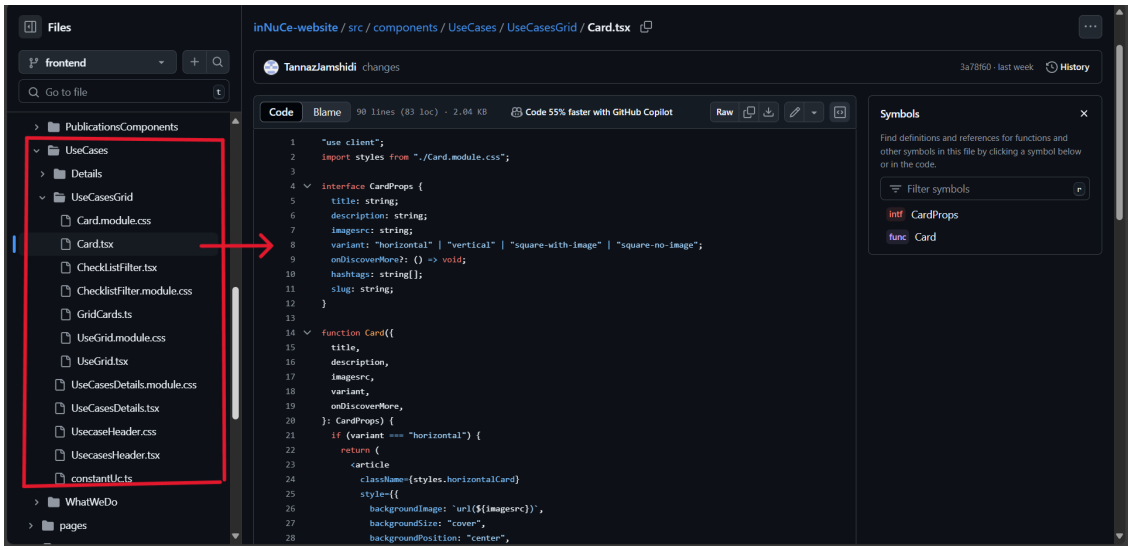

Figure 3.18: corrected codes for matching the Figma Design - usecases

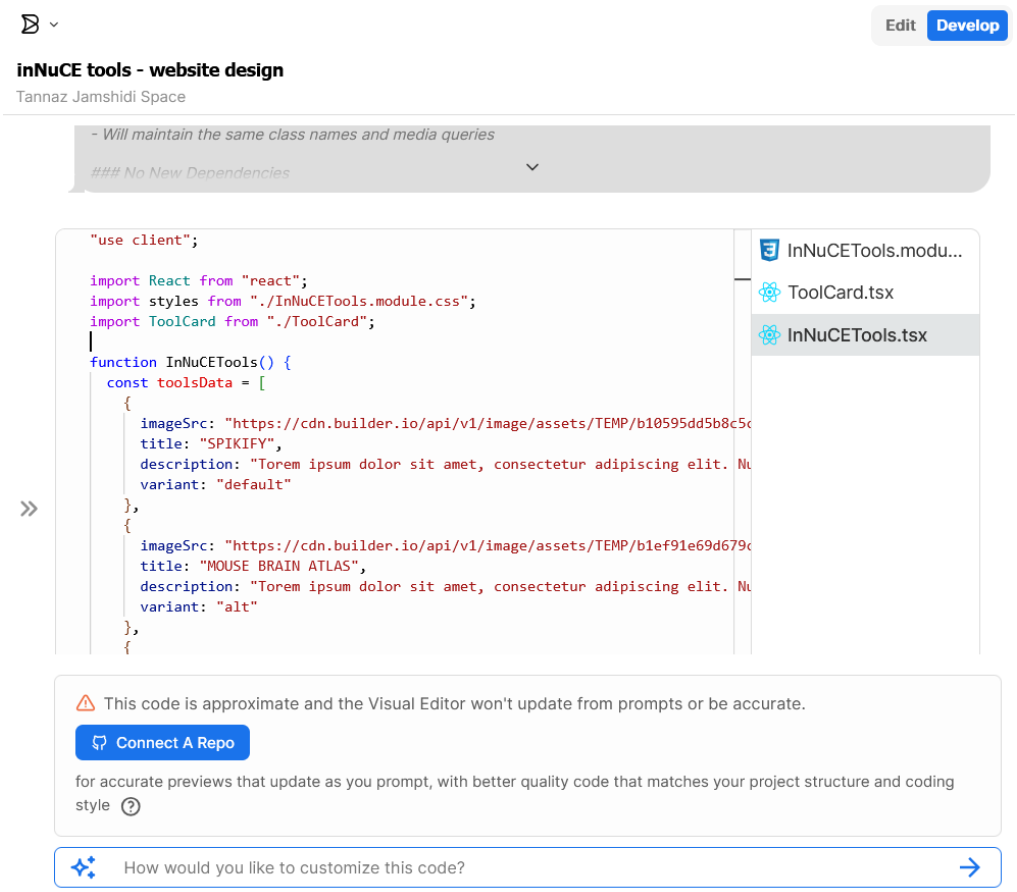

Figure 3.19: Example of folder structure generated by Builder.io for InNuCE tools section in service/infrastructure section

Chapter 4

Results and discussion

4.1 Overview of the Final Website

The final version of the website is a modern, visually cohesive platform designed to present the research, publications, and ongoing activities of the neuromorphic computing lab. It includes several key pages: the *Home page* introduces the lab with animated elements and a clear visual hierarchy; the *About page* outlines the mission and presents the team; and the *Research page*, which is the only dynamic section, lists publications by year and integrates advanced features such as a direct link to each paper on the IRIS [32] database, an AI-generated summary titled *In a Nutshell* and a podcast generation button to convert the summary into audio. Hosted at **https://innuce.polito.it**, the website is intended for both academic and general audiences, with a focus on usability, clarity, and scalability for future developments.

4.2 Comparison Between Design and Implementation

While the initial designs created in Figma provided a clear visual and structural blueprint for the website, the transition from static design to functional implementation revealed several challenges and necessary adaptations. Using Builder.io allowed for a rapid transformation of the Figma components into React code, preserving much of the layout and styling. However, the generated code often lacked clean structure and semantic HTML. As a result, manual adjustments were required to refine component hierarchy, correct spacing inconsistencies, and ensure accessibility standards. For example, components such as the Navigation Bar and the Hero Section were carefully reviewed and rewritten in React (see **Listing4.2** and **4.1**), improving the overall maintainability and responsiveness of the layout.

Furthermore, certain interactive features—such as the dynamic rendering of publications and AI-generated content—could not be fully designed in Figma and had to be implemented programmatically. These are illustrated in the dynamic Research page, where advanced features like summary generation and podcast playback are integrated . Despite these limitations, the overall visual fidelity between the design and the final implementation remained high, demonstrating the usefulness of AI-assisted tools in bridging the gap between design and development while highlighting the continued importance of manual intervention for quality and functionality.

Listing 4.1: Navigation.tsx - React Navigation Bar

```
import "./Navbar.css";
import { NavItem } from "./NavItem";
import { useLocation } from "react-router-dom";
import { useNavigate } from "react-router-dom";
import logoW from "../../assets/inNuceWhite.svg";
import logoB from "../../assets/inNuceBW.svg";

import { useState, useEffect } from "react";

const navigationItems = [
  { label: "HOME", to: "/" },
  {
    label: "SERVICES",
    hasDropdown: true,
    parentTo: "/services",
    dropdownLinks: [
      { label: "INFRASTRUCTURES", to: "/infrastructures" },
      { label: "USE CASES", to: "/usecases" },
    ],
  },
  { label: "RESEARCH", to: "/research" },
  { label: "ABOUT US", to: "/aboutus" },
];

export const Navigation: React.FC = () => {
  const location = useLocation();
  const navigate = useNavigate();
  const [menuOpen, setMenuOpen] = useState(false);
  const [scrolled, setScrolled] = useState(false);

  const toggleMenu = () => {
    setMenuOpen(!menuOpen);
  };

  const handleScroll = () => {
    if (
```

```
      location.pathname !== "/aboutus" &&
      location.pathname !== "/accessapp" &&
      location.pathname !== "/research" &&
      window.scrollY > 50
    ) {
      setScrolled(true);
    } else if (
      location.pathname !== "/aboutus" &&
      location.pathname !== "/accessapp" &&
      location.pathname !== "/research"
    ) {
      setScrolled(false);
    }
  };

  useEffect(() => {
    if (
      location.pathname === "/aboutus" ||
      location.pathname === "/accessapp" ||
      location.pathname === "/research"
    ) {
      setScrolled(true);
    } else {
      setScrolled(false);
    }

    window.addEventListener("scroll", handleScroll);
    return () => {
      window.removeEventListener("scroll", handleScroll);
    };
  }, [location.pathname]);

  const handleAccessButtonClick = () => {
    navigate("/accessapp");
  };

  return (
    <nav className={'navbar ' + (scrolled ? 'scrolled' : '')}>
      <img
        loading="lazy"
        src={scrolled ? logoW : logoB}
        className="logo"
        alt="InNUCE Logo"
      />
      <div className={'LinksContainer ' + (menuOpen ? 'show' : '')}
         >
        {navigationItems.map((item, index) => {
          const isActive =
            location.pathname === item.to ||
            item.dropdownLinks?.some((link) =>
```

```
              location.pathname.startsWith(link.to)
            ) ||
            (item.parentTo && location.pathname.startsWith(item.
               parentTo));

          return <NavItem key={index} {...item} isActive={!!
             isActive} />;
        })}
      </div>

      <div className="hamburger" onClick={toggleMenu}>
        <div className={menuOpen ? "bar open" : "bar"}></div>
        <div className={menuOpen ? "bar open" : "bar"}></div>
        <div className={menuOpen ? "bar open" : "bar"}></div>
      </div>

      <button className="accessButton" onClick={
         handleAccessButtonClick}>
        ACCESS APP
      </button>
    </nav>
  );
};
```

Listing 4.2: HeroSection.tsx - React Component

```
import React, { useMemo } from "react";
import "./HeroSection.css";
import { BACKGROUND } from "./constants";
import Particles from "@tsparticles/react";
import { type Container, type ISourceOptions } from "@tsparticles/
   engine";

const HeroSection: React.FC = () => {
  const particlesLoaded = async (container?: Container): Promise<
     void> => {
    console.log(container);
  };

  const options: ISourceOptions = useMemo(() => BACKGROUND, []);

  return (
    <section className="hero">
      <Particles
        id="tsparticles"
        particlesLoaded={particlesLoaded}
        options={options}
      />
      <div className="herocontainer">
        <h5>
          i<span className="special-n">n</span>N
```

```
          <span className="special-u">u</span>CE Lab
        </h5>
        <h4>
          The Neuromorphic Computing Research Infrastructure @
             Politecnico di
          Torino
        </h4>
        <h3>
          Bring your brain-inspired projects to life with our own
             Cloud-Based
          <b> Neuromorphic Prototyping Platform</b>.<br></br> Know
             more about
          our
          <b> software tools</b>, discover our <b> development
             services</b> or
          just take a look at our <b>use cases</b>.
        </h3>
        <div className="buttons">
          <button onClick={() => {}} tabIndex={0}>
            KNOW MORE
          </button>
          <button onClick={() => {}} tabIndex={0}>
            FOLLOW US
          </button>
        </div>
      </div>
    </section>
  );
};

export default HeroSection;
```

4.3 Evaluation of AI-Enhanced Features

The AI-enhanced features integrated into the website, namely the automatic generation of publication summaries *In a Nutshell* and the conversion of these summaries into audio podcasts, were evaluated based on their accuracy, usability, and contribution to user engagement. The summarization tool provided concise and relevant overviews of complex research papers, making technical content more accessible to a broader audience. However, occasional inaccuracies and omissions in the generated summaries highlighted the necessity for human oversight and refinement. Similarly, the podcast feature improved accessibility for users who prefer audio formats or have visual impairments, though the synthetic voice quality and pacing sometimes affected listener experience. Despite these limitations, the AI-driven functionalities successfully enriched the website by offering diverse content consumption options and demonstrated the practical potential of AI in enhancing

academic dissemination. Future improvements could focus on increasing summary precision and enhancing naturalness in speech synthesis to further elevate the user experience.

4.4 Future Works

The current project focused primarily on the design and development of the front end of the inNuCE website, with the goal of creating a visually engaging, user-friendly interface that effectively communicates the lab's research and values. While the front end is fully implemented, the back end remains to be developed.

The intended back-end architecture was conceptualized in parallel with the front-end design. It includes dynamic content handling and efficient data management. Express.js was identified as the preferred runtime environment for server-side development, chosen for its scalability, high performance, and compatibility with JavaScript-based front ends. A suitable database management system will be selected to store and manage publication metadata, research summaries, team profiles, and other relevant content.

Once implemented, the back end will enable seamless integration between the database and the user interface, allowing updates to be reflected in real time and supporting a more dynamic and interactive user experience. This future development will complete the system's functionality and ensure maintainability and scalability.

The project also demonstrated how the development barriers of modern websites can be reduced thanks to the availability of AI-based tools. These tools proved valuable in supporting developers during the design and implementation process, accelerating prototyping, and providing learning opportunities throughout. By integrating AI-assisted platforms into the workflow, the project highlighted how such technologies can streamline development and make modern web creation more accessible and efficient.

Bibliography

[1] Sagar Gholve Sonali Suryakant. «The Impact of AI on Web Development». In: *International Journal of Scientific Research in Modern Science and Technology* (2024).

[2] Venkata Krishna Reddy Tummeti. «Website developer: Web application». In: (2003).

[3] R. Carina Tobias Warbung N. Soedarso. «Website design using design principles to increase user satisfaction». In: (2023).

[4] Oluwatoyosi F. *The importance of Design Principles and how they impact good designs*. 2022. URL: https://uxdesign.cc/the-importance-of-design-principles-and-how-they-impact-good-designs-93b58b723918 (visited on June 11, 2025).

[5] Jesse James Garrett. *The Elements of User Experience: User-Centered Design for the Web and Beyond*. New Riders, 2010. ISBN: 9780321683687.

[6] Ian Sommerville. *Software Engineering*. Pearson Education, 2015.

[7] Roger S. Pressman and Bruce R. Maxim. *Software Engineering: A Practitioner's Approach*. 8th. McGraw-Hill Education, 2014.

[8] EBRAINS-Italy. *EBRAINS-Italy Official Website*. Accessed 2025-06-19. 2025. URL: https://www.ebrains-italy.eu/.

[9] Aibek Mamadalievich Toktorbaev. «The Role of Artificial Intelligence in Website Creation». In: *Publishing Center Science and Practice* (2025).

[10] OpenAI. *GPT models in ChatGPT*. 2024. URL: https://help.openai.com/en/articles/6825453-chatgpt-release-notes (visited on June 19, 2025).

[11] uizard. *AI-powered UI Design*. 2025. URL: https://uizard.io (visited on May 7, 2025).

[12] CodeParrot AI. *Why CodeParrot?* Accessed 2025-06-19. 2025. URL: https://codeparrot.ai/blogs/why-codeparrot.

[13] vercel. *The AI code generator for web UI*. 2025. URL: https://v0.dev (visited on May 27, 2025).

[14] Figma. *What is Figma*. 2025. URL: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma (visited on Apr. 2, 2025).

[15] adobe. *adobe$_i$llustrator*. 2020. URL: https://www.adobe.com/products/illustrator.html.

[16] adobe. *adobe$_p$hotoshop*. 2018. URL: https://www.adobe.com/products/photoshop.html.

[17] remove.bg. *remove background*. 2025. URL: https://www.remove.bg/.

[18] convertio. *converio*. 2024. URL: https://convertio.co/it/.

[19] microsoft. *visual studio code*. 2023. URL: https://code.visualstudio.com/ (visited on Apr. 2, 2025).

[20] Everlane. *Everlane Official Website*. Accessed: 2025-06-19. 2025. URL: https://www.everlane.com/.

[21] Gymshark. *Gymshark Official Website*. Accessed: 2025-06-19. 2025. URL: https://www.gymshark.com/.

[22] builder.io. *Builder's Visual Development*. 2025. URL: https://www.builder.io/m/explainers/visual-development-platform#main (visited on Apr. 2, 2025).

[23] MDN Web Docs. *Document Object Model (DOM)*. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model. Accessed: 2025-06-23. n.d.

[24] React. *React*. 2025. URL: https://github.com/facebook/react (visited on Apr. 3, 2025).

[25] tsparticles. *tsparticles*. 2025. URL: https://particles.js.org/ (visited on May 1, 2025).

[26] Polito. «LINEE GUIDA PER LE AZIONI DI INFORMAZIONE E COMUNICAZIONE A CURA DEI SOGGETTI ATTUATORI». In: (2022).

[27] Flaticon. *Free vector icons and stickers*. https://www.flaticon.com. Accessed: 2025-06-21.

[28] WordArt.com. *Create Word Art Online*. https://wordart.com. Accessed: 2025-06-21.

[29] Freepik. *Graphic Resources for Everyone*. https://www.freepik.com. Accessed: 2025-06-21.

[30] Elia Ferraro. «Studying and Redesigning a Web Application: User-Centered Analysis and Process Optimization». MA thesis. polito, 2024.

[31] Express.js contributors. *Express - Node.js web application framework*. https://expressjs.com/. Accessed: June 2025. 2024.

[32] Politecnico di Torino. *IRIS Institutional Repository*. Accessed: 2025-06-19. 2025. URL: https://iris.polito.it/.