Enabling General Applicability for Brain–Computer Interface Software Through Cloud Computing

Daniel Burger

SAE Institute in collaboration with
Middlesex University London

This thesis is submitted for the degree of
Bachelor of Science, Web Development

SAE Institute Zürich August 2022

Abstract

Brain–computer interfaces are already improving the circumstances of a small subset of the human population with neurological conditions (e.g. prosthetic control for paralysed patients). On the other hand, brain–computer interfaces for the general population could increase the potential for a better understanding of the brain due to generating more brain data. As a result, such mass-market brain–computer interfaces could improve the lives of healthy people through more natural or efficient interactions with technology or people around them or by directly altering human brains for certain benefits.

In this bachelor's thesis, the author describes the first steps toward a mass-market and generally applicable brain–computer interface software system that enables unidirectional neural communication with computers via the cloud. The author introduces the term neural/cloud interface in the current state of neuroscience research and the context of key industry players.

Unlike in the natural sciences, a system such as a neural/cloud interface does not inherently occur, but is only achievable through engineering; thus, it must be created before it can be studied. Accordingly, through a case study project at the neurotechnology start-up IDUN Technologies, the reader is educated on the exact definition, motivation, project context and, most importantly, the results of building a neural/cloud interface system in practice. The central supposition is that a neural/cloud interface system may already be feasible with today's software technologies and is not just speculation as in similar research on brain/cloud interfaces.

In addition to the case study's findings, an example of a cloud software architecture is developed and discussed in detail to securely stream, process and store brain data over the internet while respecting end-user privacy.

Ultimately, this work aims to be a stepping stone into this new and interdisciplinary field between brain–computer interface software and cloud computing by providing a condensed overview of all critical aspects, insights and findings for future neural/cloud interface engineers. This would be achieved by building on the knowledge gathered during the implementation of this bachelor's project.

*This work is dedicated to the **IDUN Technologies team,** who have been extremely helpful in my journey into the neurotechnology industry. They taught me a lot about working in a scientific setting and allowed me to investigate various aspects of non-invasive brain–computer interfaces in combination with cloud computing.*

*Additionally, I would like to express my sincere gratitude to **Dr. Cao Tri Do,** who advised me and patiently taught me a great deal about the theoretical aspects of neuroscience, which was essential for completing this work.*

Table of contents

List of figures

List of tables

Chapter 1
Introduction

This chapter introduces the reader to the primary focus, key topics, and broad explanations of this thesis. It also presents the supposition, goals, and objectives of its primary content and structure.

1.1 Background

There has been a long-standing interest in developing neural interfaces—systems that sense and interact with the nervous system's electrical activity. Successful research into the development of technologies that enable neural interfacing has been underway for decades, with the first experiments being conducted by Jacques J. Vidal in the late 1970s (**vidal_real-time_1977**). In particular, a related discipline focusing on the direct interaction between brains and computers via a brain–computer interface (BCI) has gained momentum with the emergence of companies such as Neuralink (**mor_brain-computer_2021**).

One aspect of BCIs is the development of imaging technologies that enable the measurement of brain activity. A distinction can be made between different methods of measuring brain activity signals at different locations. On the one hand, there are invasive sensors used with electrocorticography (ECoG), a sensor-based[1] imaging method that places electrodes on the surface of the brain; on the other hand, non-invasive sensors are placed on the body, such as with electroencephalography (EEG). Both methods measure the electrical field elicited by the firing of neuronal populations. However, the further away the electrode is from the brain and the more tissue (e.g. scalp, skull, cerebrospinal fluid, and cortex) lies between firing neurons and the measurement sensor, the more the spatial resolution decreases.

A second aspect of BCIs is the development of software that reads and interprets data from sensors. Both aspects present their own set of challenges and complexities. Nonetheless, complete and applicable BCIs work in practice and have been used for many years in patients with neurological disorders (**braingate_publications_nodate**). There are also consumer and non-clinical BCIs available, such as the OpenBCI and Neurosity products, which aim to democratise the use of EEG systems by offering low-cost hardware and open-source software.

[1] Other options next to sensor-based imaging methods are, for example, scan-based methods such as computed tomography (CAT) that utilises special X-rays to produce axial images of the brain.

1.2 Relevance

The possibilities for sufficiently and directly connecting the human brain to the outside world via computers are seemingly endless, given the purely physical[2] assertion that all our feelings, memories, dreams, and thoughts are most likely the sum of electrical activities in our brain. There are several use cases for utilising insights from our brain to interface with computers, such as controlling prosthetic limbs for amputees (**campbell_amputee_2014**) as shown in **??**, enabling communication for people with locked-in syndrome[3] (**chaudhary_spelling_2022**), or diagnosing neurological problems and improving the mental capacities of elderly patients (**belkacem_brain_2020**). These are just a few of the promising examples that could be cited.

Fig. 1.1: An amputee is using his mind to control two robotic arms to perform several tasks that require fine motor control (**campbell_amputee_2014**).

[2]There is research on the quantum mind that describes how classical mechanics cannot explain consciousness; however, this is not considered in this thesis as this research is still in its early stages.

[3]Locked-in syndrome describes the paralysis of all voluntary muscles in its entirety, thereby making it impossible for people to communicate with the outside world.

These examples make it evident that BCIs can significantly impact the field of therapeutics and accessibility for a small subset of the human population. However, one can envision not only alleviating deficient living conditions but also improving the lives of healthy people through more natural or efficient ways of interacting with technology or by directly altering human brains for certain benefits such as the possibility to enhance or delete bad memories (**spiers_enhance_2014**) or record and guide dreams (**haar_horowitz_dormio_2020**).

Many use cases still seem a long way from being applicable today, yet many experts and even entire companies are developing BCI hardware and software, such as Neuralink, that is aimed at the general population (**urban_neuralink_2017**). The general applicability of a BCI system to the mass market will depend on several factors, of which the form factor and invasiveness of the hardware are likely to be essential aspects. Nevertheless, the totality of the ecosystem in which the software resides is a valuable aspect that should not be overlooked.

1.3 Opportunity

Whether it is a bidirectional and invasive BCI or a unidirectional and non-invasive BCI, the data collected from the brain would always need to be processed, contextualised, and classified to produce an intelligible output to interface with, as shown in **??**.

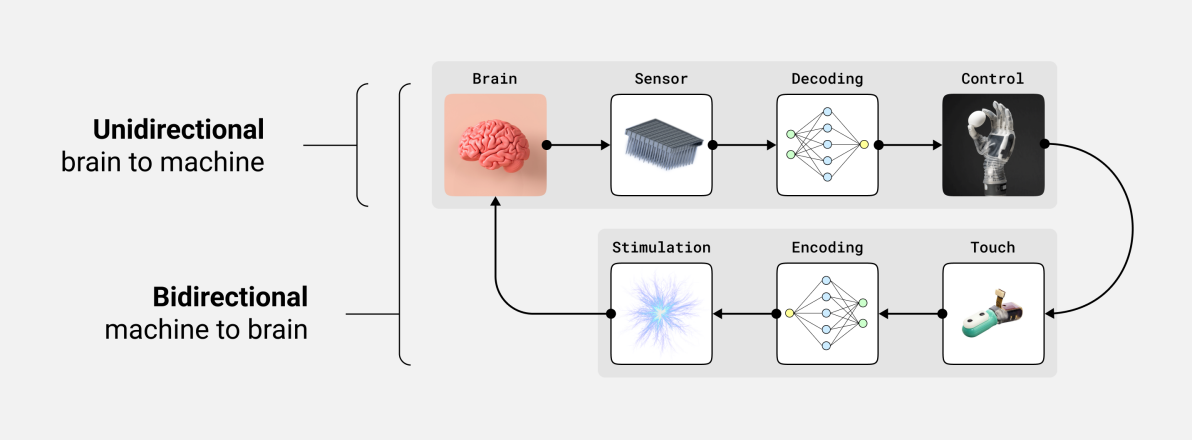

Fig. 1.2: Conceptual difference between a unidirectional and a bidirectional BCI and a simplified overview.

Most current BCI software systems being developed—for example, for a BCI implanted in a living patient—are typically deployed in a local environment; that is the software system and its components are located on a physically nearby computer.

The author sees an opportunity to move BCI software from local environments to the cloud to enable a variety of benefits for the general population and mass market. The present thesis determines what components would be required by a cloud-based software system that is ready for the mass market. The emphasis is on a holistic view of such a system, which means that the entire technology stack and context are taken into account.

1.4 Supposition

There is already promising research on the implications of brain/cloud interfaces (B/CI) by **martins_human_2019** (**martins_human_2019**) and **angelica_cognitive_2021** (**angelica_cognitive_2021**), which analyses bringing hypothetical large-scale BCI software systems into the cloud. Nonetheless, their research focuses on hypothetical scenarios in the future, usually premised on the development of other technologies such as neural nanorobotics, vital advances in 5G, or the presence of supercomputers in the cloud (e.g. for the augmentation of the human brain); thus, they are somewhat removed from today's realities. To distinguish the research presented in this thesis, the author coins the term 'neural/cloud interface' (N/CI), which refers to a holistic software system that connects a neural interface device such as a BCI to the cloud and then to other neural interfaces, software systems, or physical devices.

The primary supposition is that an N/CI is feasible with contemporary software technologies, requiring only theoretical groundwork based on empirical software engineering. To shed more light on this, this thesis looks at the process and lessons learned from the author's perspective in developing a real-life N/CI in the industry.

1.5 Goals and objectives

The overarching goals of this thesis are to provide an overview of the context and definition of an N/CI and, most importantly, the software components of which it is composed. In order to achieve these goals, the author must achieve the following objectives:

1. Describe the context and motivation behind creating an N/CI.
2. Establish a clear definition as well as the distinctions and advantages of an N/CI.
3. Identify and define the most relevant aspects required to realise an N/CI in practice.
4. Illustrate an example architecture of an N/CI to implement its components in practice.

Chapter 2
Project context

This chapter describes the project's context and the current literature findings. The limitations of neuroscience are discussed as well as the state of current non-invasive and sensor-based BCIs, the motivation for developing cloud-based BCI software for the general population, and the broad definition of an N/CI.

2.1 Limitations of BCIs

The possibilities of BCIs are not without limitations. In addition to hardware limitations, the author addresses a broader issue related to neuropsychology that directly correlates with the software aspects, in addition to the challenges of computability.

2.1.1 Decoding neural data

It is important to emphasise that the task of decoding neural data is different from decoding thoughts, which is a critical factor for BCI software to enable the control and interaction via thoughts. Moreover, decoding neural data and extracting the thoughts behind it so that the software can understand them are disciplines in their own right. This is similar in machine learning use cases: for example, getting computers to recognise letters written on a photograph is a very different problem from interpreting the written words in sentences (i.e. computer vision and natural language processing). Another part is understanding the sentences and their meaning, as in natural language understanding (NLU).

NLU is considered an artificial intelligence (AI) hard problem, which means that the difficulty of these computational problems is assumed to be equivalent to solving the central problem of artificial general intelligence[1] (**demasi_theoretical_2010**). Understanding less structured data, such as neural data, is more complex than understanding structured and human-invented syntax such as written language because it contains more hidden features and semantics than a paragraph of text. As a result, the author assumes that completely understanding neural data can also be considered an AI-hard problem. This constraint illustrates well how far current research is from being able to interpret a person's thoughts based on measured neural data.

[1] Based on the assumption that general human-level intelligence could be computable.

2.1.2 Abstract thoughts

To further emphasise the complexity of interpreting neural data, a practical example will be presented: *Imagine a red house in the middle of a forest.* Depending on the individual thought process, one might imagine the house with temporary visual imagery in mind, as in visual thinking, or one might imagine it more verbally, such as conceptually comprehending each word sequentially of what a red house is and that it is located in a forest (**amit_asymmetrical_2017**). **??** illustrates the range of options.

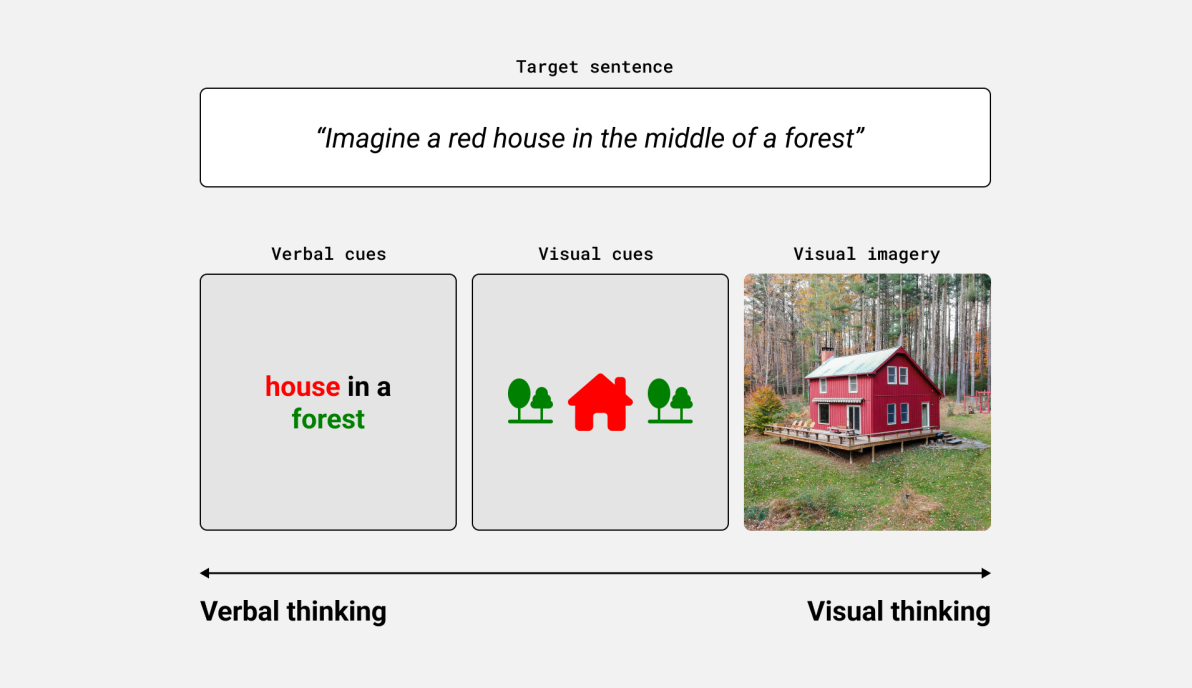

Fig. 2.1: Difference between verbal and visual thinking using the target sentence of a red house in the middle of a forest.

Additionally, it should be considered that different types of thoughts exist at different levels of abstraction and complexity. One can assume that the visual image of a red house in the forest is more abstract and far-fetched than, say, the movement of one's own thumbs, which has a clear physical counterpart. It becomes even more complicated when one imagines abstract concepts that cannot be visualised, such as the idea of a company. A company is an abstract, collectively agreed-upon concept that lacks a clear physical counterpart[2]; it is, therefore, even more complex to decode the meaning of measured brain activity in this case than with the red house.

[2]Some people might think of a company building when imagining a company, others might imagine their website, their logo, or physical products.

2.1.3 Technological limitations

Even though mind reading or decoding abstract thoughts seems difficult, some functional tasks of the brain are still extractable due to their localisation. Localisation means that these signals are generated by local brain areas that can be identified, such as the motor cortex, which has been shown to be responsible for muscle movement (see **??**).

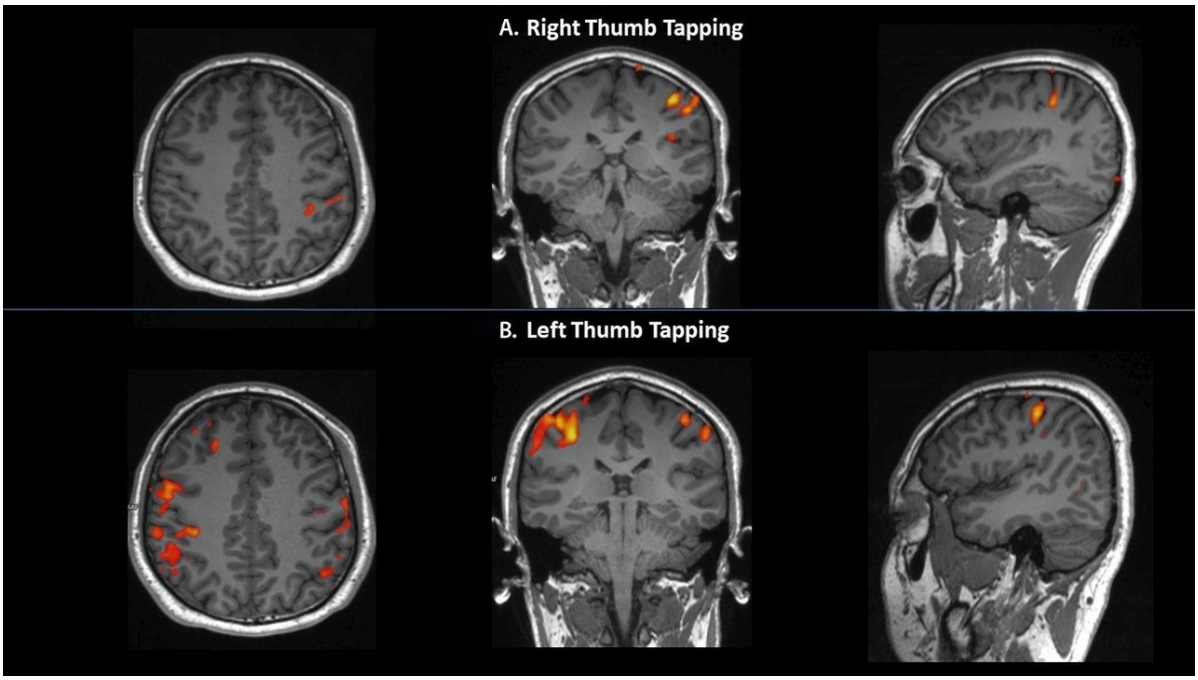

Fig. 2.2: Localised neurons during right- and left-thumb movement using functional magnetic resonance imaging (fMRI; **rashid_bilateral_2018**).

Examining the areas of the brain responsible for activating individual muscle strands can yield a comparable response of muscle stimulation in the brain and thus be measured as input for BCI software, for example, to move a prosthesis. However, the more specific, behavioural, and abstract the thoughts are, the less spatially visible are the responsible brain areas. By reference to the intention of identifying, for example, the thought of a red house in a forest, the author has identified three technological limitations:

- To understand single thoughts, it is essential to have sufficiently clear data with a certain level of detail (e.g., the level of detail that elicits the action potentials of individual neurons[3]) and temporal precision (an action potential has a short duration of about

[3]Action potentials are the fundamental neurobiological and neurochemical processes through which neurons transfer information to each other.

one millisecond [**byrne_resting_2021**]) to perform studies to extract possible localisation of individual thoughts. Current neuroimaging technologies cannot capture in sufficient detail every process of the entire brain at once to extract the activity of individual neurons while also having high temporal precision.

- Even if we could measure every single neuron in the brain with high temporal precision, we would have an extreme amount of data generated concisely. Suppose we would collect a float[4] per neuron that represents the rate of change in voltage with respect to time with a frequency of 1 millisecond and then record each neuron in the brain a million times a second (taking into account that the average human brain has around 86 billion neurons); under these conditions, we would generate 305.53337637684 petabytes of data per second. This is currently not feasible for commercially available storage and processing resources.

- Even if we had the technology, it would still be challenging because of the difficulty of reproducing experiments in neuroscientific studies—usually referenced as the 'replication crisis' (**maxwell_is_2015**). It is probably impossible to generate clean-slate brain data that is comparable to previously recorded data since our neurophysiological brain tissue changes over time due to neuroplasticity (**nierhaus_immediate_2021**). Moreover, we are in different states of mind every millisecond of our existence, which can produce different effects, such as insufficient sleep, being disturbed by something, mental distraction due to an important event that may have occurred since the last measurement, or a salient thought that randomly occurs while recording neural data.

2.1.4 Lack of data

As mentioned in the previous section, the last two points depend on advances in storage systems or the possibility that we do not actually need such precise brain data to understand single thoughts. However, to address the first point, some promising solutions already exist for measuring large parts of the brain with high temporal and spatial precision, such as time-domain functional near-infrared spectroscopy (TD-fNIRS), which the company Kernel employs in its Flow device (**ban_kernel_2021**). TD-fNIRS sensors detect changes in concentrations of oxygenated and deoxygenated brain cell activity by using near-infrared light in response to neuronal activity. According to Kernel, the precision of TD-fNIRS is sufficient for a clearer understanding of the brain and using it for BCI applications. The company, however, claim that collecting and organising longitudinal brain data

[4]The size of a float on a Windows 64-bit application is 4 bytes which was used for the calculation.

from a variety of subjects is the key to solving the most difficult challenges in neuroscience (**kernel_hello-humanitypdf_nodate**).

Building on Kernel's claim, a recent publication also claims that even datasets with several hundred people are too tiny to offer consistent insights into the brain; as a result, most published neuroscience studies with dozens or even hundreds of people could all be incorrect in their conclusions (**marek_reproducible_2022**). In neuroscientific studies, brain tissue and activity variations have been linked to variances in cognitive capacity, mental health, and other behavioural features which set an important foundation for our current understanding of the brain. Neuromarkers[5] of behavioural features are frequently sought in such studies to further understand the brain. **marek_reproducible_2022** (**marek_reproducible_2022**) claim that most of the neuromarkers would not work when the collected dataset is more extensive, which would pose a general problem for the field of neuroscience. UK Biobank's collection of brain scans is one of the first efforts to solve this problem (**noauthor_imaging_nodate**), but it is still far from what we might need since, as **marek_reproducible_2022** (**marek_reproducible_2022**) claim, we might even need millions of datasets to start understanding the brain (**callaway_can_2022**). This is both fascinating and a possible significant constraint for BCIs, because understanding the brain is essential to making sense of the measured data they interface with.

A counterargument, however, is offered by Andrew Ng, artificial intelligence (AI) pioneer and founder of the Google Brain research lab. Ng believes that machine learning, which underpins all BCI software, should be developed in a data-centric manner, which means that quality should be preferred over quantity. That is, the quality of the data on which models are trained should be as high as possible to answer specific research questions rather than focusing on merely collecting a huge amount of data (**brown_why_2022**). However, this brings one back to the replication crisis, which is the difficulty in generating clean or universally valid brain data comparable to previously collected data due to the nature of our ever-changing brain.

Ultimately, there will almost certainly always be a mix of both approaches. As a result, for generally applicable and mass-market-ready BCIs, a relatively large amount of qualitative brain data collected in specific and reproducible experiments or environments is required. This is where high-end, customer-focused BCIs could come into play because the adoption rate of a device suitable for everyday use is higher than the number of subjects in research labs, resulting in larger and more longitudinal datasets. This, combined with more targeted

[5] A neuromarker is a biomarker that is based on neuroscientific data to detect biological properties such as a disease or illness. For more information, the interested reader can refer to **jollans_neuromarkers_2018** (**jollans_neuromarkers_2018**).

experiments, improved neuroimaging technologies, and advances in machine learning, could unlock enormous potential in brain research.

2.2 BCI landscape

This section will discuss the current landscape of customer-focused and non-invasive BCIs, their applications, and the distinctions within their software offerings.

2.2.1 Real-world BCI applications

As mentioned in **??**, consumer-focused BCI products are already commercially available. OpenBCI, a non-medical BCI company, does not provide a specific use case, but they provide hardware (as depicted on **??**) as well as software that is universally applicable. These can be used in research wherever EEG is employed and in developing BCI applications. Several neurofeedback or research apps have been created using OpenBCI's products (**openbci_openbci_nodate**). Taking this information into consideration, one can see that the OpenBCI customer is responsible for developing their own BCI applications or incorporating it into their research, rather than having a sophisticated end-user application directly from OpenBCI.

Fig. 2.3: OpenBCI's EEG device (**be_superhvman_conde_2017**).

Fig. 2.4: NextMind's BCI headset (**louise_neurotechnology_2019**).

Fig. 2.5: Muse's meditation headband (**muse_muse_nodate**).

Another example of a commercial BCI is NextMind's product, as shown in **??**. The company does not focus on having an end-user application for its BCI, but focuses instead on offering a software development kit (SDK) so that the Unity real-time engine can use NextMind's technology for brain-controlled actions in video games. One significant difference between NextMind and OpenBCI is that NextMind includes a built-in classification of

neural data captured by hardware—in this case, classification of active visual focus on virtual objects based on steady-state visual evoked potentials (SSVEP). Because its business model is presumably based on the unique selling proposition of its active visual focus classifier, NextMind does not provide access to the raw EEG data collected by the sensors. Nonetheless, NextMind's product is less focused on a specific use case, as it is applicable to all kind of games inside the Unity engine. A relatively closed and specific BCI is illustrated by the EEG headband by Muse, as shown in **??**. Its purpose is to measure meditation and sleep. The company also offers an end-user app to help people better understand their meditation and sleep and how to improve them. Compared to the previously mentioned products, the Muse headband is not a unidirectional BCI per se as there is also biofeedback based on neural data. However, the key difference between Muse and OpenBCI is that the neurotechnology has been abstracted. Users do not need to know anything about neuroscience, neurotechnology, or the interpretation and classification of neural data to achieve useful functionality for their use case. They also do not need to understand the software system's underlying architecture. They only need to know how to pair the device with their smartphone via Bluetooth.

Aside from full-stack BCI solutions, in which a company provides a complete BCI solution, including hardware and software, some companies focus solely on the software aspect. One such example is Neuromore, a company that provides a neural signal processing software platform. The company is hardware agnostic, which means one can plug nearly any BCI hardware or sensor into their computer and connect it to their Neuromore Studio software.

Neuromore Studio, as shown in **??**, is free and open-source software that runs locally on various platforms. It provides a variety of drag-and-drop interfaces for creating and managing signal processing pipelines. For example, one can transform EEG data to extract band power, create triggers based on band-power selection, and generate conditional outputs to perform tasks such as moving a character in a video game. The author aims to differentiate the offerings of these consumer-oriented BCIs along a spectrum. On one side of the spectrum are BCI companies that provide the hardware (with software that at least connects to the device) and are then more generally applicable to use cases not defined by the company behind the BCI (such as OpenBCI). On the other side are BCI companies that are application-specific in terms of both the software and the hardware, such as the Muse headband.

Although this thesis focuses on consumer-oriented BCIs, the applications of various BCI offerings can still be distinguished based on whether they are more consumer-oriented or research-oriented—such as the distinction, for example, between NeuroSky (a company creating EEG-based BCIs for hobbyists) and Emotiv (a company creating professional and

Fig. 2.6: Screenshot of the Neuromore Studio software (**neuromore_neuromore_nodate**).

expensive EEG systems), which are more research-oriented. However, both NeuroSky and Emotiv provide a research version and a consumer or enterprise version of their software and hardware, aiming for general-purpose applicability across customer segments and use cases. Other considerations include whether the applications are steady-state evoked, such as those based on a frequency of noise laid on top of virtual objects to detect which object the person is looking at (e.g. NextMind), or whether they track the totality of mental states without evoking neural signals with external stimuli, such as in tracking sleep or concentration levels, both of which arise primarily inside the brain. This distinction can be labelled as passive, active, or reactive BCI, as **alimardani_passive_2020** coined in their work on passive BCIs (**alimardani_passive_2020**). However, the author does not want to include this dimension because it would introduce additional complexities related to the BCI software application layer.

The application layer, as shown in **??**, is the part of a BCI that acquires the interpreted data from a classification model and turns it into applicable functionality to interface with a physical or digital counterpart to perform functions such as moving a player in a game or initiating sound on the computer via its speakers. There is also the physical part, the brain, and a possible physical interaction counterpart in the form of, for example, a robot arm. The totality of the software stack is responsible for processing the data, that is, extracting

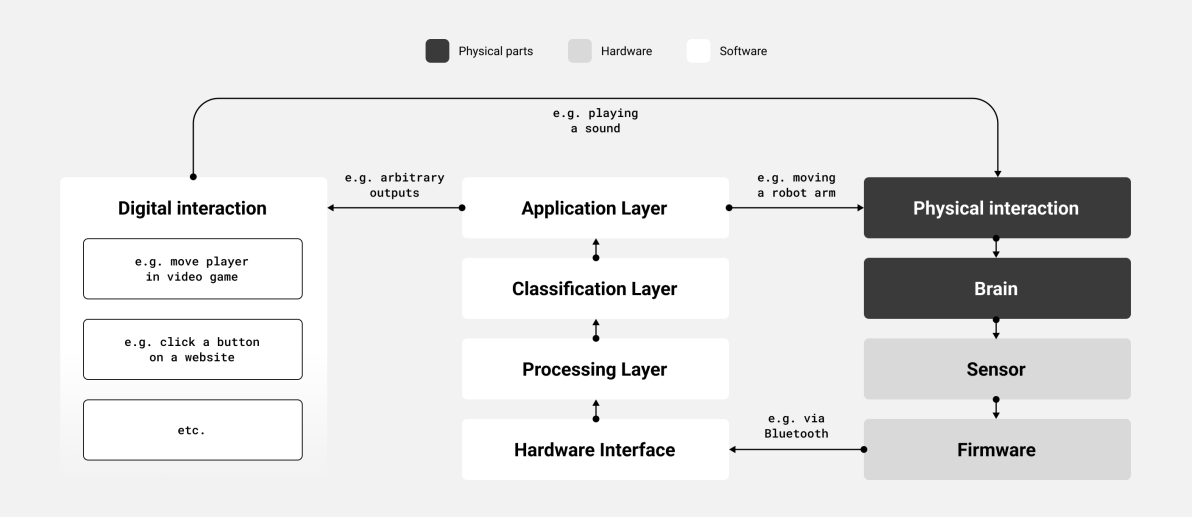

Fig. 2.7: Architectural overview of BCI components.

the relevant information from the raw data and turning it into any desired and meaningful output for the application layer.

2.2.2 Unobtrusive hardware and software

The unobtrusiveness of hardware and software is another aspect to consider when discussing BCIs. Unobtrusiveness in hardware means that it is either not visible at all[6], as when sensors are implanted beneath the skull, or that it is in a form factor that is already socially established. The hardware prototype of IDUN Technologies, a Swiss startup, as shown in **??**, measures brain activity in the ear canal that aims to resemble the form factor of established in-earbuds. **??** shows the Notion product from Neurosity, which measures EEG on the head and is not comparable to a socially established form factor such as earbuds. What is considered socially established and accepted truly depends on the society and context, as one could argue that wearing a Neurosity device under a hat while talking to a friend is more acceptable than wearing in-earbuds. Still, the implications of different form factors must also be considered, such as the possibility of moving the device and thus creating motion artefacts in the signals or the position of the sensors. The ear canal is ideally located close to the brain's auditory cortex but not so much to the visual cortex, which is located at the back of the head. However, further hardware implications for BCIs are not a topic covered in this thesis.

[6]Other things related to the overall user experience (UX) are sometimes included in the notion of unobtrusiveness, such as comfort, reusability, and convenience; however, the author is implying only physical characteristics such as shape and size.

Fig. 2.8: IDUN Guardian hardware, rather unobtrusive BCI.

Fig. 2.9: Neurosity Notion hardware, rather obtrusive BCI.

Nonetheless, it is perhaps not as simple to discuss the unobtrusiveness of software as it is with hardware. Unobtrusive[7] software, as defined by the author, is the abstraction of the underlying software or system that executes the logic to fulfil a task without the user knowing what the technical requirements are. For example, to use an HP ENVY Photo 6200 printer with one's Android phone, one must first download the HP Smart app and the HP Print Service Plugin app that acts as a driver for the printer to get it set up and running (**hp_hp_nodate**). In the case of the HP printer, the user must understand some of the underlying technical requirements in order for it to work, rather than simply concentrating on the task of printing something. An example of unobtrusive software is a computer mouse that one simply plugs in and starts using immediately[8].

Unobtrusive software in BCI refers to the ability to connect one's hardware to the computer or smartphone and use it without the need for additional software such as drivers or command-line interface (CLI) software. For example, to use an OpenBCI device, one needs to open the graphical user interface (GUI) app, connect the hardware, presumably test its quality, begin a data stream session, and output the stream via a network system such as a Lab Streaming Layer (LSL), connect to the signal from software such as Neuromore Studio (**openbci_neuromore_nodate**), run the data through a classification pipeline, and then connect the output from Neuromore to a video game via the engine itself to have

[7] Other words for unobtrusive could be discreet, fully-integrated, invisible or simply 'in the background'.

[8] Unobtrusiveness usually correlates with usability, but it is not always the case; more advanced users would not consider locked-in abstraction as more usable.

controls for the video game. Clearly, this software is not unobtrusive. There are examples of software included as an executable file, and which is thus relatively unobtrusive; but this software is closely linked with the hardware and the brand behind the hardware, or it is in the proof of concept (PoC) stage rather than a production-grade application. Buying a new pair of headphones and plugging them into one's computer to enjoy neuro-enhanced[9] experiences interacting with the brain's outputs or measuring brain data across all apps and the operating system would be examples of truly unobtrusive BCI software.

2.2.3 Production-grade software

Another aspect of BCIs is the state they are in, such as the production maturity of the software. There is no clear definition of what production-grade software is; but in most cases, software developers agree on the following characteristics:

- Software that works at any time when access is required. It is therefore capable of frequent and intensive use in commercial or industrial environments.
- Software whose behaviour is deterministic and predictable and is, therefore, well-tested, well-documented, and optimised in terms of speed, efficiency, and security for the given context (e.g. the size of the user base). Usually, developers agree on a Definition of Done (DoD) inside their team to what is considered production-ready; some examples include test coverage of 80+%, peer-reviewed and commented code, and a common code style guide.
- Software that runs in a production environment—that is, on a cloud computing cluster for actual users rather than in a test environment for test users or on hardware delivered to real customers—and that can adapt itself to the context, such as to a higher access rate or possibly insecure user-generated input. In most cases, especially in cloud computing, production-grade also means larger datasets (such as in databases), the possibility of a greater number of edge cases due to a larger user base, and most importantly, more available computing power on production instances.

As stated earlier, most BCIs, such as the OpenBCI, are usually not intended for production. They are intended for PoCs such as controlling objects in games or conducting research. End-to-end and full-stack BCIs for production are rare, as most are highly specific and not intended for general applicability (such as the Muse headband) or the software aspect is intended for PoCs or research (such as with Emotiv). Pure software products, such as the one

[9] Neuro-enhanced software can be described as adding additional features to input methods via the brain.

from Neuromore, lack the hardware component and lack an SDK that can be integrated into existing software for a variety of platforms. Neurosity aims to provide a universally usable and unobtrusive software stack that is even open source. However, because the hardware is not unobtrusive enough, it does not meet the author's definition as mass-market-ready and production-grade—apart from the fact that it is not known whether their software stack is aimed to be used in production (**neurosity_neurosity_2022**) due to third-party developers providing disclaimers that it is a work in progress (**turney_notion_2022**)). Companies such as Neuralink are presumably working on a general-purpose, unobtrusive, and production-grade software system that enables developers to build production apps and even platforms on top of it for a variety of use cases without being limited (**musk_integrated_2019**). Since the intended hardware is also unobtrusive in its form factor (being implanted), Neuralink has a high potential to become one of the first general applicable and mass-market-ready BCIs if one ignores that surgery is required to acquire the device (**neuralink_approach_nodate**).

2.3 Definition of an N/CI

All the previously mentioned aspects feed into the definition and motivation of an N/CI, which the author introduced as a new term in **??**. **??** goes into more detail about the motivation and clear definition of an N/CI, and the aspects of cloud computing in combination with BCI software that can be displayed in a three-dimensional axis, as shown in **??**. The invention of the term 'N/CI' was a byproduct of this project's implementation and will be discussed in more detail in **??**.

Fig. 2.10: Visualisation of the term neural/cloud interface with its three axes and differentiation of six terms.

Chapter 3
Methodologies

This chapter describes the project-related academic methodologies that fall within the author's situation and experience as well as the planned approach to achieve the goals and objectives of the thesis. The reader is introduced to the rationale for the intended workflows and software tools.

3.1 Derivation of the case study

In October 2021, the author began working as a cloud software engineer at IDUN Technologies, an ETH spin-off startup in Zürich, to further develop their existing software products. IDUN had already created a PoC software system that included a web-based single-page application (SPA) hosted on AWS Amplify, a backend-as-a-service product aiming to simplify the deployment of backends for mobile and web apps. IDUN's in-ear sensor sent EEG data to a physical network bridge via Bluetooth and then to the cloud via the internet. The raw EEG data[1] was saved and made available for download in various file formats.

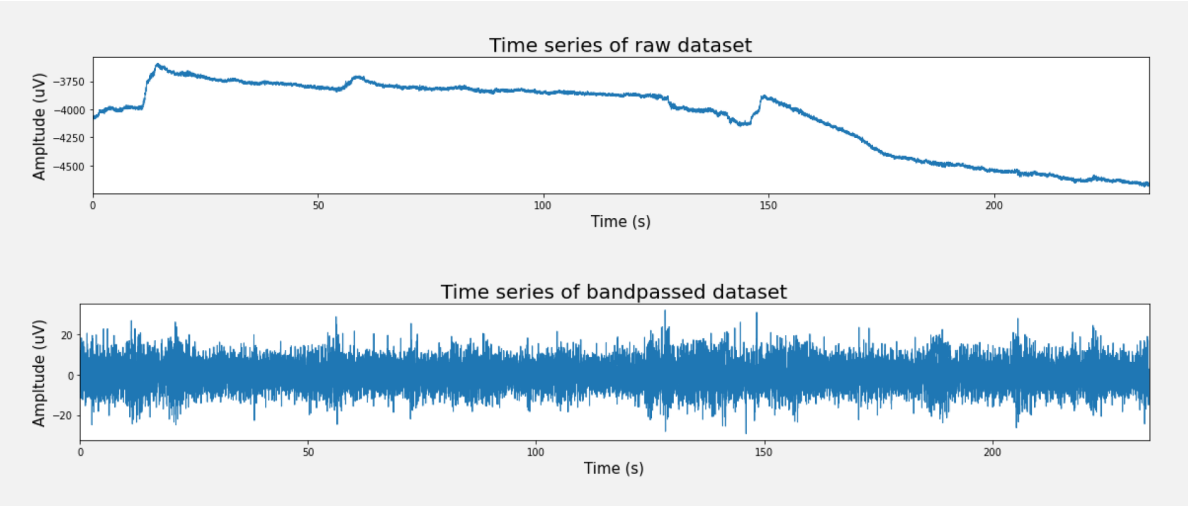

Fig. 3.1: Difference between raw and filtered data from IDUN's product.

In addition to raw data, the IDUN SPA provided processed data, such as filtered data, which included low-pass and high-pass filtering of EEG data as shown in **??**. This processed EEG data was then saved alongside the raw version on the cloud.

[1] Raw EEG data refers to data measured by the in-ear sensor without any additional processing.

The EEG data could also be visualised in near-real-time on the SPA as a time-series X- and Y-axis plot. Additionally, users could control the device by sending start and stop commands to the hardware components. The PoC system, whose architecture is shown in **??**, was in a relatively unstable state and had various error sources that made it impossible to reliably record EEG data for more than a few minutes, making the product unusable for existing customers or a mass-market launch. While working on the system, the author encountered various technological bugs and flaws as described in **??**.

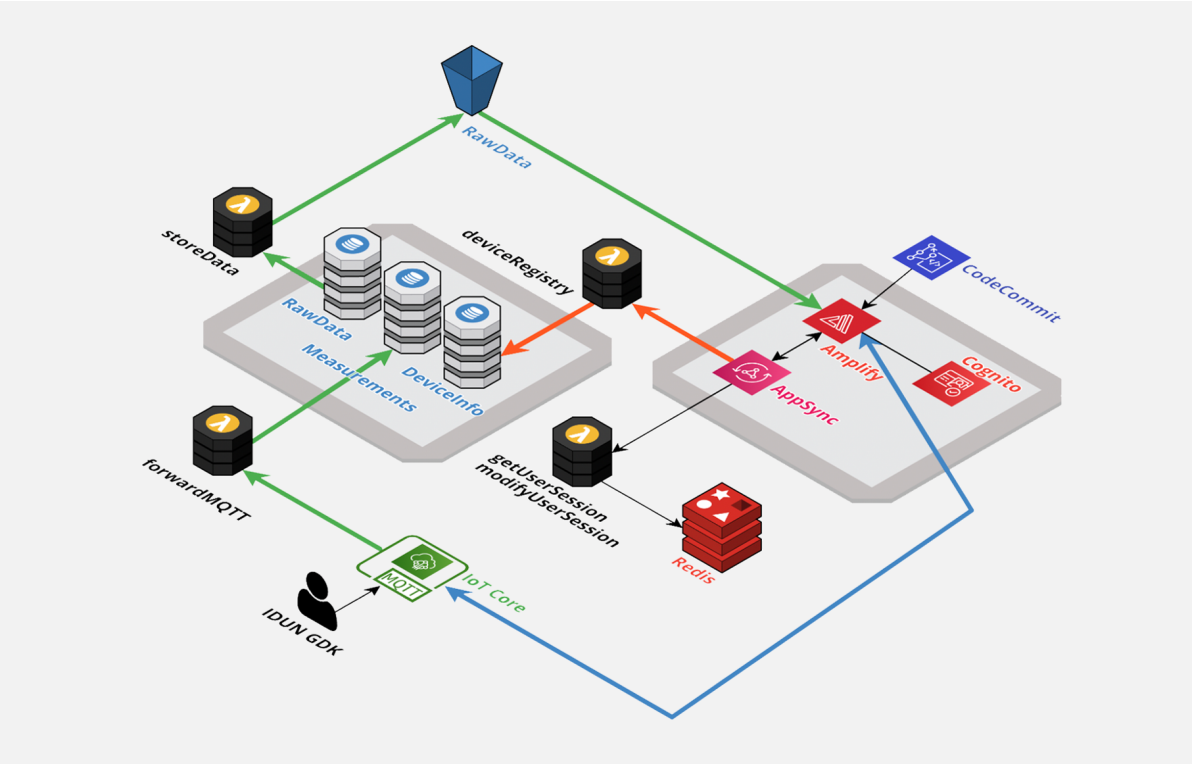

Fig. 3.2: IDUN's software architecture at the end of 2021.

Due to growing challenges with the existing software system and an ever-increasing technical debt from the software not being test-driven or developed with clean code quality standards (resulting in bugs and quirks that are difficult to track down), the author proposed to halt the implementation of new features and restructure the system from the ground up using a software engineering-oriented approach. The company's management approved the request for this redevelopment in early December 2021. At the time, the author was already working on his original bachelor's project, which focused on an EEG-controlled multiplayer game, assuming that IDUN's software system would be stable by the time the bachelor's project began. The original focus of the project was officially changed at the end of 2021 to create a thesis on the redevelopment of IDUN's cloud software.

3.2 Case study

As previously mentioned, IDUN Technologies is a full-stack company that produces in-ear EEG sensors in the form of earphones. Their vision is to sell their hardware and license a software product coupled with the hardware called Neuro-Intelligence Platform (NIP), as shown in **??**.

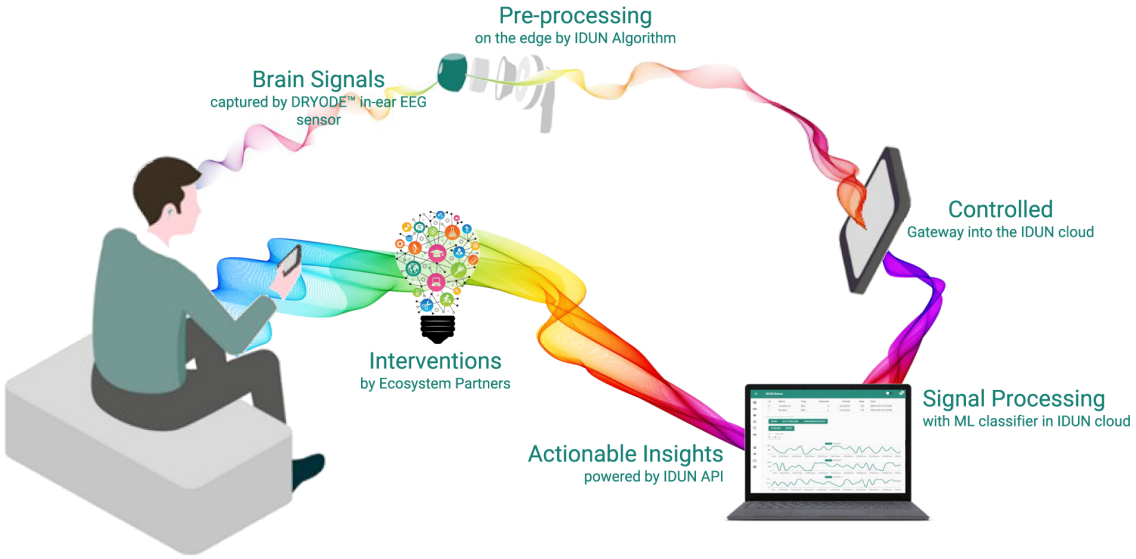

Fig. 3.3: IDUN's vision of a closed neurofeedback loop (**idun_guardian_nodate**).

The essential mission is that an unobtrusive BCI device, such as an in-ear headset, can be worn for the majority of the day while measuring neural data that is sent in real-time to the cloud to process and classify actionable insights that other developers can use to build their interventions (e.g. apps, websites, and games) on top of it to promote mental health and well-being. Furthermore, the aspect of longitudinal data is essential, implying that it could be advantageous to store neural data over a long period and run classifiers on it from time to time to understand one's brain better—one of the possible solutions to the challenge of a lack of data described in **??**.

Moreover, it is essential to select an appropriate research method to develop a system that would fulfil the company's mission and vision. The author chose a case study because it is an effective method for dealing with unusual and atypical cases while providing new and unexpected perspectives in certain situations such as in a deep-tech startup like IDUN.

3.3 Procedure

This section describes the planned procedure for conducting a case study-based research methodology to develop the proposed N/CI at IDUN Technologies as part of their NIP.

3.3.1 Project stages

The goal was to conduct qualitative research and examine the use case from various perspectives. In addition, two other methodologies were used in the case study: (a) expert interviews, which entail locating experts on topics such as cloud or BCI and asking them questions that will assist in answering implementation questions; and (b) group discussions, which aim to learn about people's attitudes and opinions about related topics.

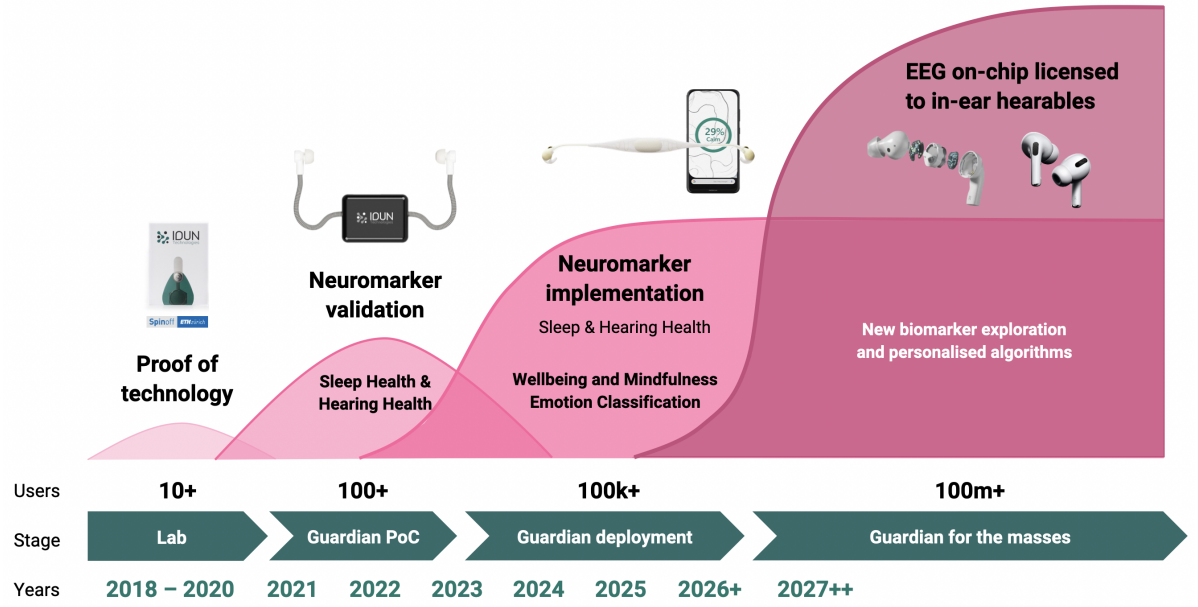

Fig. 3.4: IDUN's plan to achieve their goals (**idun_guardian_nodate**).

Before defining the project stages, it is essential to look at the timing of IDUN's roadmap, as shown in **??**. There are three essential pieces of information: the stage in which the technology resides, the estimated user base size, and the time frames that should be applied to these stages. The author was working towards a version scheduled to be released in 2023—that is, working on the N/CI placed in the Guardian deployment phase. As the company is still implementing the first neuromarkers into the product and is focusing on two use cases (sleep and hearing health) due to the size of the team and the mass production of hardware and sensors, the sales team is targeting the sale of devices to about 100,000

people. Another constraint is the deadline of the bachelor's thesis of the author, which is set for the 5th of August 2022. User size, company stages, and the deadline for the thesis are important context information for defining the project stages for this use case. The proposed project stages to execute the case study are presented in **??**.

Project stage	Description
1.1 Define key technical requirements and constraints	Based on the internal hardware, the neuroscience and data science departments and the IDUN mission and roadmap, the author needs to define the purely technical requirements and constraints. Some constraints are, for example, hiring of advisors, time limits and risks due to investment rounds. Some purely technical requirements come from, for example, the firmware team or the material scientists, such as maximum latency or the data structure of the digitalised and amplified EEG.
1.2 Extensive literature research	Already since the beginning of the original bachelor thesis, the author started to do literature research in the field of BCI, which is very helpful for the new focus. Further literature review now includes books on data-intensive applications and cloud, articles and research papers.
2.1 External user interviews	Defining user personas with IDUN's product manager, application engineer and sales team, finding real people to represent these personas, preparing an external interview framework and questions, gathering as many insights as possible.
2.2 Creative workshop and prototyping	Based on the insights from the user interviews, conduct a creative workshop with IDUN's product manager and application engineer. The results are design artefacts such as wireframes, user flows, interactive prototypes and architecture diagrams which are essential for internal group discussions.
3.1 Internal group discussions	The design artefacts are used for an internal validation with the department heads. Several group discussions are held with each department, ranging from materials science to business development. Based on the group discussions, the artefacts are adapted and improved.
3.2 Expert interviews	Define experts in different areas of BCI, cloud and EEG, and neuroethics. Create a framework for expert interviews and questions that are still unclear or open based on the design artefacts. Conduct expert interviews and gather as many insights as possible to adapt and improve the design artefacts.
4. Start bootstrapping	While the design process is still ongoing, the author can already start implementing the most important technical requirements with anything that is a flexible bootstrapping of internal development processes to ensure quality assurance, for example, not being tied to the other phases.
5. Iterative and agile implementation via Scrum	IDUN works with Scrum, for this the author has introduced Scrum in the research and development department. Scrum at IDUN has three-week sprints and there are ten sprints from January 2022 to the end of July 2022 that can be used to go through the design process and implement the designs. In the process, there are several iterative processes to validate and test the increments of the system based on insights from the implementation, ongoing literature research, insights from other departments or the design process (i.e. user interviews) that is still ongoing.

Table 3.1: Project stages of the bachelor's project.

Stage 1.1 is the crucial stage before starting anything else. Stage 1.2 is continuously in progress. Stages 2.1 to 3.2 take place simultaneously while Stage 4 is underway[2]. As soon as Stages 3.2 and 4 are completed, Stage 5 can be started.

3.3.2 Group discussions

The author believes it is critical to conduct user interviews with external individuals. Consequently, this informs the development of a deep-tech-focused product in such a way that extensive company-specific knowledge is not required. Based on the user interviews, the goal is to create design artefacts in the design tool Figma for GUI-related topics as well as in the whiteboard collaboration tool Miro for architecture-related topics. These design artefacts are utilised to communicate the outputs of the design process to IDUN's internal

[2]Number 4 refers to bootstrapping, which usually refers to the practice of setting up development environments that include initial Git repositories, folder structures, and creating accounts on cloud providers.

staff based on the findings from the external user interviews. The ideas and findings should be presented and validated relative to IDUN staff's understanding and product perceptions via internal group discussions. Because the development of a complete neurotechnology product involves many disciplines ranging from physics to machine learning, the company must have a shared understanding of what the software component of the NIP—that is, the N/CI—will look like and what functionalities it will have.

3.3.3 Expert interviews

It is critical to consult experts through group discussions after the external and internal validation and design process and its iteration. In the author's experience, it is difficult to engage consultants at an early stage of the process without knowing what one wants to build because advisors, say, from a consultancy firm, tend to push for commercially oriented ideas to generate their revenue rather than thinking about the project's long-term success. Experts are chosen based on the author's experience and the experience of other IDUN Technologies employees, such as data scientists and neuroscientists. According to a previous blog post by the author, the importance of ethics in the field of BCI is critical (**burger_influence_2021**). As a result, it is crucial to include neuroethics experts in the plans and ideas for IDUN's N/CI as well.

3.4 Software and tools

There are many software and tools that can be used to construct a comprehensive software system such as IDUN's N/CI. The author had some leeway in deciding which software and tools (i.e. libraries and frameworks) to use. Nonetheless, some software and tool limitations should be considered when beginning the implementation:

- IDUN has received a large number of credits from Amazon for its AWS account, which should therefore be used to avoid creating redundant costs when using another cloud provider.
- The Python ecosystem has strong roots among engineers and researchers at IDUN, which should be taken into account. Examples of this are the Python packages SciPy or MNE for processing and visualising EEG data, which do not exist in any other language.
- The author needs to familiarise himself with the software and tools required to build IDUN's N/CI. For example, the AWS cloud would be an example of the author needing

to transfer his knowledge from the Google Cloud Platform (GCP) experience and understand different ways of implementing certain cloud features in AWS.

- Any new tool that is not open source and free to use should be carefully considered in terms of future maintainability, security, and learning effort.
- Wherever possible, redundancy in the workload should be avoided, as a venture capital-funded startup like IDUN has a particular runway time that describes the company's remaining time before it runs out of cash. Spending an entire sprint working on something that is then thrown away can be business-critical, so every software and tool decision should be carefully weighed.

3.5 Checkpoints for progress

Progress is reviewed at the end of each three-week sprint. IDUN's entire research and development team and stakeholders, such as C-level management, attend the review events. Each department asks questions about the author's progress and findings. The planning for the upcoming sprint will take place based on the topics discussed in the reviews. Asana, a project management tool, mirrors the overall product backlog and the personal backlog of the author's bachelor's project and tracks progress in terms of story points and remaining epics in its dashboard. In addition to Asana, documentation of all project stages will be documented in Notion. Screenshots of the mentioned documentations are attached in **??**.

Before moving on to the next chapter, it should be noted that the quality of the proposed project stages is an essential part, made even more crucial when one considers the extensive list of flaws in the current IDUN software system, as summarised in **??**. These flaws can only be avoided by implementing extensive quality assurance measures (i.e. unit tests and automated continuous integration pipelines) during the bootstrapping project stage. To quote Robert C. Martin from his book *Clean Architecture*: 'The only way to go fast, is to go well' (**martin_clean_2018**).

Chapter 4
Implementation

The implementation details of creating the software components for the NIP of IDUN are covered in this chapter, along with key events in the empirical software engineering process such as the recognition of novelty and the requirement for an N/CI definition.

4.1 Timeline

Procedures listed in the project stages in **??** are a good guide for project implementation; however, in the end, such plans run in unexpected ways. As a result, researching and implementing a non-trivial system such as an N/CI requires a high level of agility.

The effective timeline at the time of writing is shown in **??**; it includes the previously mentioned project stages but is structured differently from what was initially described. **??** explains why some project stages were completed differently than initially assumed.

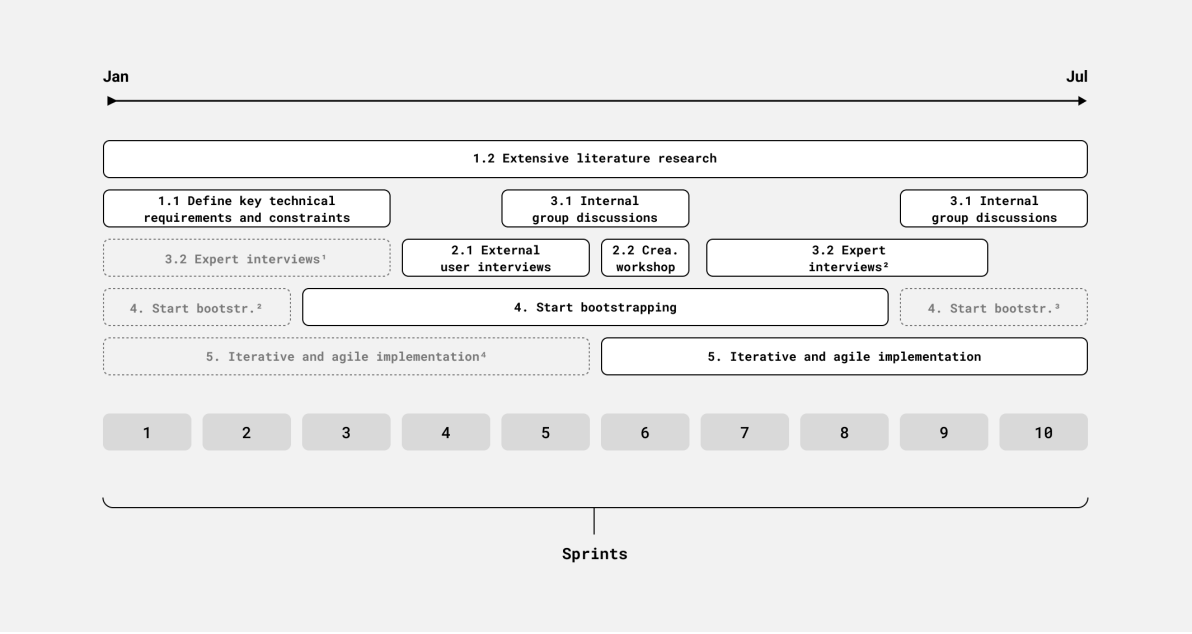

Fig. 4.1: Effective timeline of the last 10 sprints with project stages in white and specifically planned stages outlined in grey.

Special stage	Description
[1] 3.2 Expert interviews	The author was able to use the first expert discussions in advance thanks to the help of one of the sales staff's networks. The experts were Nuvibit's experienced enterprise cloud and solution architects. The first topics were strictly technical in nature, focusing on medium-term technological decisions in the context of the company and the timetable.
[2] 4. Start bootstrapping	This special stage describes the phase in which the author established more organisational structures, such as a professional Scrum or GitHub setup. Furthermore, the time was used to create a more professional AWS organisational setup with various organisational units, as described in the AWS Best Practice Guide (**blackham_best_2020**).
[3] 4. Start bootstrapping	Since the creation of two different Python SDKs (as discussed later in this chapter), more bootstrapping tasks were due at the very end of the given time frame, mostly including the setup of a privately installable Python package and SDK-specific quality assurance pipelines and automations.
[4] 5. Iterative and agile implementation	Iterative and agile implementation began as soon as needed, without waiting for the design process to be completed (i.e. avoiding a waterfall process). Prior to gaining insights from the design process, time was spent on everything else, such as evaluating technologies, further bootstrapping, first example codebases, or maintaining the old codebase a bit.

Table 4.1: Special project stages in the effective schedule as shown in **??** and an explanation of the sequencing.

Several key events occurred during the implementation that shaped the future course of the project and research. This was primarily due to initially unplanned early expert discussions or uncertainties in the requirements as the user-centred design process was still being prepared. These key events are overlaid with the effective project plan as shown in **??**. The three green key events are the most influential key events.

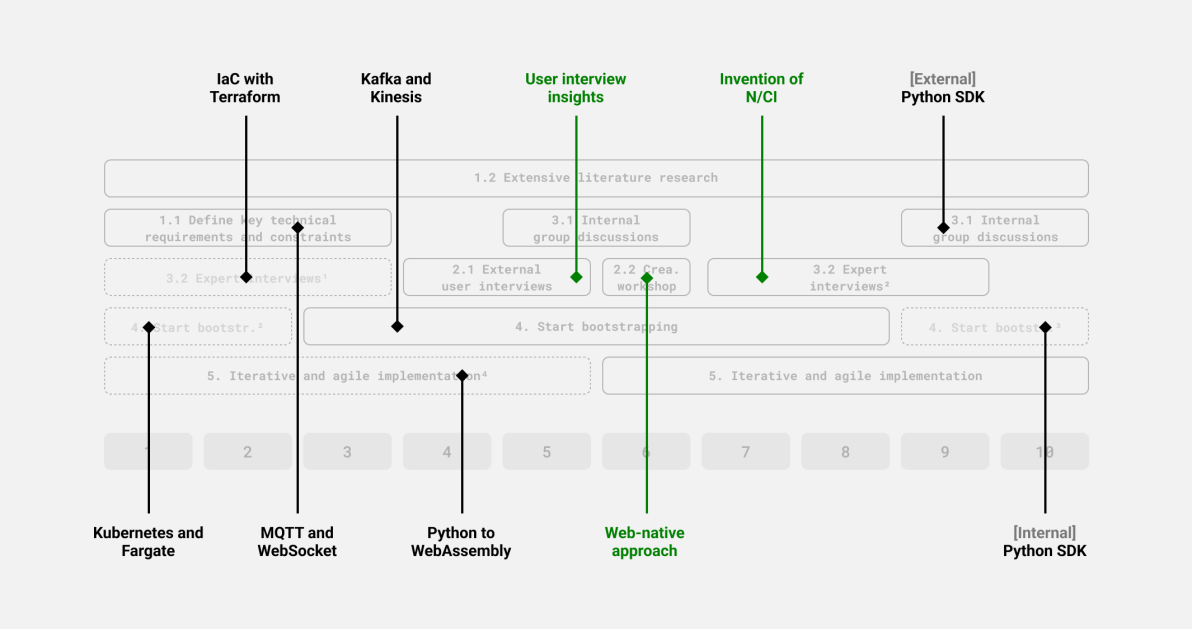

Fig. 4.2: The key events overlaid on the effective project schedule as illustrated in **??**, with the most influential events coloured green.

4.2 Key events while building an N/CI

This section discusses the most influential key events as presented in **??** and explains why they occurred as they did and why they were critical to the success of the project. The following outline is not chronologically presented but begins with the most influential key events.

4.2.1 User interview insights

The conduct of user interviews was one of the most significant key events. This process began with developing customer personas based on the sales team's previous experiences with real customers and the planned customer segments targeted by C-level management. In summary, the author does not want to go into too much detail about how the personas were created and how the process went because the focus is on the results based on the user interviews—not on the persona creation process itself. The personas are illustrated in **??**, and a descriptive overview of the personas can be found in **??**.

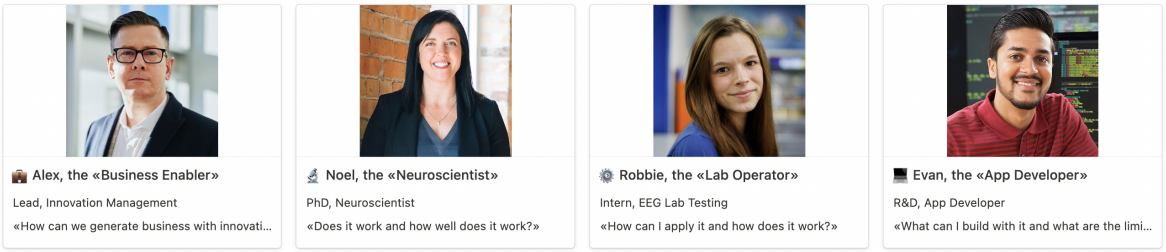

Fig. 4.3: IDUN's customer personas (**idun_guardian_nodate**).

People were chosen to represent the personas as accurately as possible. Finally, the author compiled a list of twenty individuals, of which eight people were chosen and invited for separate interviews. The interview outline was planned in collaboration with IDUN's product manager and an external industry UX expert, Laura Bendixen, who works as a UX designer at Blick.ch. The author developed the outline based on Laura's expertise with previously conducted user interview sessions, which can be found in **??**. All user interviews were conducted remotely and lasted no more than 1.5 hours. During the interviews, insights were transcribed on Post-Its. People working with BCIs or EEG were interviewed, as were doctoral students working in research labs, developers who had never worked with BCIs or had extensive experience building their own BCI software, and business enablers from companies responsible for BCI-based accessibility.

Persona	Occupation	Description
Alex	Lead, Innovation Management	Alex works in a large/small company to bring new innovations and technologies into business and product development roadmaps. Alex informs the company's senior management about new innovations and how they may impact future business and strategic initiatives. Alex has a budget to spend but is supported in decision-making by colleagues in R&D, product development and innovation scouting.
Noel	PhD, Neuroscientist	Noel works within large and small organisations to build innovations based on neuroscience and technologies into future products with an outlook of 5+ years. Noel leads a team or group and informs innovation managers such as Alex about new neurotech products and how they may provide new value to customers. Noel has to get approval to start new projects and obtain budgets for larger initiatives. Noel informs key stakeholders in the organisation and is supported in decision-making by colleagues in R&D, engineering teams, and innovation scouting.
Robbie	Intern, EEG Lab Testing	Robbie works within a lab and research group, possibly in large or small organisations, either commercial or academic. Robbie often starts from a study protocol developed with or given by Noel, where a test procedure needs to be followed to have consistent data collection methods with different test subjects. The work of Robbie supports the assumptions and study endpoints or goals defined by Noel and neuroscience colleagues. The results may be used in product development or to prepare a research report, white paper, publication, etc. Robbie works hands-on in the lab, knows how to put on a wet-electrode EEG system, looks at raw signals and understands if the data is being collected correctly. Robbie debugs test setups when things are not running correctly, may process results with scripts and is familiar with lab setups like sleep labs. Robbie can also process the study results, put them together for interpretation together with Noel and support technical details that may (but are not often) communicated to Alex.
Evan	R&D, App Developer	Evan works within large and small organisations to build engineered solutions based on a toolkit of code and hardware technologies that form the foundations of future products with an outlook of 1–5 years. Evan works in a team or as part of a technical group and works with smart people like Noel. Evan does not have a background in neuroscience but knows about electrical signals and how to build prototypes and demos. Evan needs the approval to spend more than 500 CHF on anything. Evan helps Noel to give great presentations to stakeholders in the organisation but is generally not in strategy discussion. Evan is supported in Scrum and development tasks by colleagues in the R&D and engineering teams.

Table 4.2: Descriptive overview of IDUN's personas.

The questions were non-leading and open-ended, with the primary goal of allowing interviewees to express themselves, capture their perceptions of the BCI industry, and introduce IDUN's vision and mission upfront. The goal was to determine what software offerings a mass-market BCI product would need to provide, such as convincing developers who have never worked with BCI to include neuro-enhanced features or convincing researchers to use IDUN's NIP in their research. The questions and answers covered different topics. More technically oriented people inquired about speed, performance, and privacy concerns (i.e. more production-readiness thinking), whereas researchers inquired about signal quality, the ability to synchronise with other data streams, and access to raw data (i.e. general applicability thinking). After the last user interview, all Post-Its were compiled, similar insights were grouped, and the product manager, as well as the author, categorised the most important insights into a list:

- There are two prominent use cases: (a) using the NIP for research and (b) developing an end-user-facing app. These are two critical distinctions. Researchers would use the

NIP to learn about the brain, such as through simple-setup remote experiments or real-life long-term experiments outside of laboratories. The NIP would, in such a use case, still be used in production, but not for potentially thousands or millions of users in an app intended for end users. The company or developer using the NIP would have physical access to the hardware in the research use case because they would most likely buy one or more devices and hold them in their repertoire. In contrast, companies targeting production use cases would ship the devices as white-labelled hardware through their own or third-party channels. Perhaps one could even say that because IDUN distributes the device so well, companies and developers do not need to sell the device themselves but can assume that users already own such headphones. Companies and developers can then access the brain API such as they can assume that every end-user has a geolocation API in their smartphone.

- Visual demos that are easily accessible for someone owning the hardware are essential for all personas, most notably an Alex. A visual and simple demo should demonstrate what one can do with the NIP from IDUN and inspire others to conduct research, build apps, or enable business and opportunities. These demos need to explain to the different personas the various benefits; in addition, the end-user should be able to try out the demos to see the benefits of BCI-enhanced technologies. Individuals such as Evan should be able to adapt demos with their codes to create their own versions of them.

- The user of the NIP needs to have the lowest level of control possible over the data flow and classification but also with an option for less technical users to use it and build things, similar to AWS. AWS itself is only an API to use cloud IT resources, but they also have the AWS Console app to build everything via a GUI instead only via the API. It is important that people such as Noel need to know precisely what algorithms were used or the methodologies for specific classifiers to justify them and integrate them into their research. They do not, in most instances, actually need raw data access, as plotting functionality would also be enough; therefore, IDUN's NIP should have some functionalities to plot and visualise data.

- In order to ease the use of the GUI and the API of the NIP, IDUN needs to provide—in addition to sufficient documentation—comprehensive libraries for specific environments that make the unobtrusive implementation of the NIP easier. One such example would be providing a client-side JavaScript package that is lean and easy to use and can be integrated into existing apps. Examples for such code integration would also

need to be provided by IDUN so that people like Evan can easily adjust them and more quickly start integrating them into their apps or research codebases.

- End-users need control over their data so that other companies accessing the NIP API cannot collect their data on their servers and do everything they want. Imagine if the operating system layer of one's smartphone would not offer an opt-in mechanism for the cameras; in such a case, third-party developers could install spyware and record without the user's permission. When users start to use multiple apps with an NIP integration, there needs to be some way to control the opt-in and data access without the app developers having any say in the decisions. There needs to be a technical limitation from the side of the hardware or the cloud to ensure user privacy and data security for end-users.

- Researchers are interested in the raw EEG data and the synchronisation possibilities with specific protocols such as LSL to combine, for example, a heart rate sensor with the EEG collected from IDUN's hardware. The NIP software offering needs to give enough freedom to combine the data without letting all other collected data flow into the NIP from IDUN. This means that the NIP should be able to collect, transform, and most importantly, label data in correlation with other data streams with a local option to satisfy researchers such as Noel.

These insights were critical in determining that there must be sufficient freedom for raw data research without releasing raw data into mass-market production environments to protect end-user privacy. As a result, an opt-in mechanism outside third-party ecosystems is required. A library is also required that can be implemented in end-user apps and that connects directly to the hardware without the need to download a companion app from IDUN itself. This library should allow researchers to access it locally and use their preferred protocols to synchronise tags and labels with other data streams. Nonetheless, IDUN's NIP should once again provide some control over how much raw data can be collected and should restrict access if the proposed research project or experiment time frame is exceeded. Furthermore, additional NIP functions should be added to visualise easy-to-use demos for all personas, as well as a way to use and control the NIP without having to write code—but with enough flexibility and transparency that there is trust in the system and even advanced users can work with a visual tool instead of code.

4.2.2 Invention of N/CI

The insights from the user interviews, combined with the ongoing research on B/CI and remote BCI presented in the introduction, were key to realising the novelty of this new interdisciplinary approach to developing a production-grade and mass-market BCI software system running in the cloud and targeting the general population via general applicability. It should be noted that it was realised only in the seventh sprint that such a system had not yet been defined in research. The findings from the user interviews lay primarily in the direction of the N/CI definition, while the literature review was primarily focused on distinguishing between existing research and definitions.

Once a first draft of the definition was written down, the research and separation of work packages could be more easily tackled and explored due to a better understanding of what was new and unexplored. By the end of the seventh sprint, the author's goal was to further develop the definition and motivation in the form of a written appendix as a byproduct of the main work of this thesis. **??** presents the output of this key event.

4.2.3 Web-native approach

Prior to Sprint 6, the author had already developed some ideas about future architecture roadmaps, as shown in **??**.

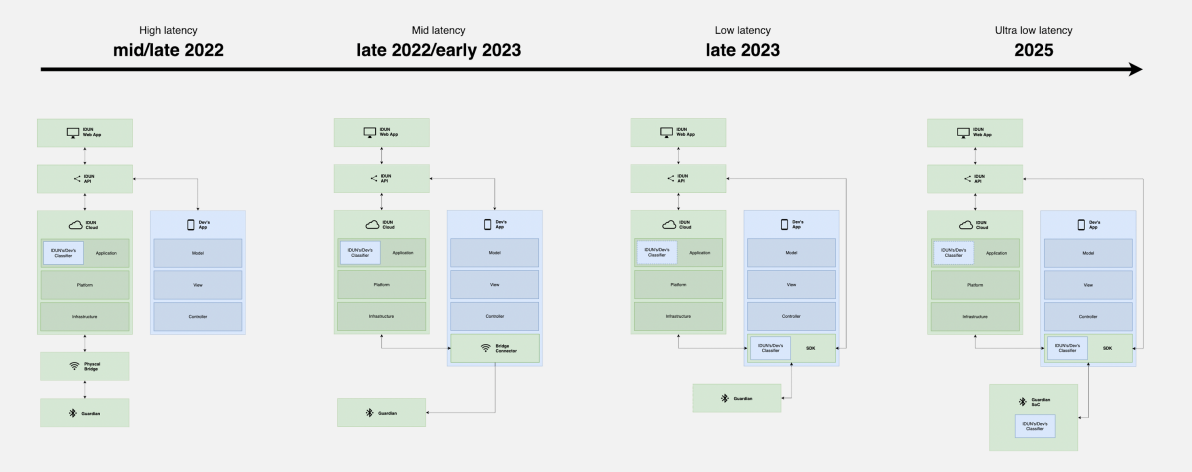

Fig. 4.4: Original architecture roadmap of IDUN's NIP going from a high latency system in mid/late 2022 to an ultra-low latency system in 2025.

The initial architecture roadmap was part of the project's initial phase to define the key technical requirements and constraints and show how the NIP software stack would move from a high latency system to a low latency system. Nonetheless, one aspect of this old archi-

tecture roadmap was that IDUN would continue to maintain and develop the physical hardware network bridge and eventually port it to end-user devices as a companion app. However, just before Sprint 6, the author came upon the work of **flynn_brainsplay_nodate** of the BCI platform called Brains at Play and its use of Web Bluetooth (**brainsplay_add_nodate**), which is a new and experimental API for browsers that allows websites to connect directly to a Bluetooth Low Energy (BLE) device. The Brains at Play platform is a hardware-independent BCI software platform with similar goals to Neuromore Studio. The main difference is that they offer their app as a web app; therefore, everything runs in the browser without downloading additional software. The author contacted Garrett Flynn, one of the founding partners, and discussed the implications and findings of using Web Bluetooth EEG data, such as that from IDUN's sensor. (Later, Garrett Flynn was also invited to the user interviews as one of the persona representatives for Evan.) Garrett Flynn's findings were entirely positive, and he strongly recommended working with the API as it makes deploying a BCI platform much more effortless than deploying applications for any operating system, especially any BLE interface. However, one of the main disadvantages of Web Bluetooth was browser compatibility, as shown in **??**.

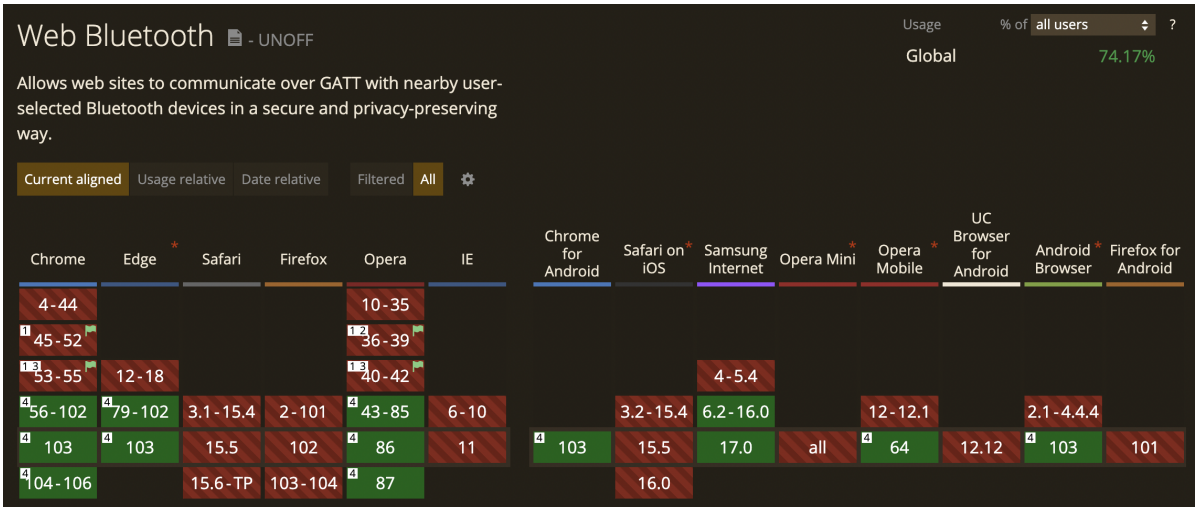

Fig. 4.5: Browser compatibility overview of Web Bluetooth (**caniuse_web_nodate**).

Nevertheless, IDUN is building a product that aims for an extensive user base in 2–3 years, as can be seen in their roadmap in **??**. This means that browser compatibility could come and go over the coming years once browser manufacturers start implementing the API. One problem, however, is that the Web Bluetooth API cannot be run in a service worker, making it impossible to run in the background when the device has its screen locked or another app is currently opened on a smartphone (**webbluetoothcg_service_2018**). This

is an intermediate problem, even for the upcoming smaller and specific user groups with the current personas. Among a variety of technologies under investigation, the author identified Capacitor, a JavaScript library, as a suitable candidate for the current use case. It essentially can compile and run web apps as native apps with the ability to access native APIs (**ionic_capacitor_nodate**) such as the camera or—especially important for IDUN—the BLE API (**capacitor-community_capacitor-communitybluetooth-_2022**). Developers can use Capacitor to write BLE code with the same interface as Web Bluetooth, compile the code, and run the applications on native devices with background functionality.

After further research, as shown in more detail in **??**, almost the entire Sprint 6 was spent evaluating the possibility of using Web Bluetooth for IDUN's NIP. At the end of the sprint, a working PoC SPA was created with an iOS and Android build that fulfilled all needs in terms of developer experience, the latency of the EEG signal from the device to a visualisation plot, and the power consumption of lower-end smartphones. Following this, the author proposed to change the original architecture roadmap as depicted in **??**, skipping the first two steps and going directly to step 3 in order to accelerate the development pace and enable greater market maturity for IDUN's NIP as soon as the end of 2022.

The use of a web app, combined with the use of modern APIs and the compilation of the app for mobile applications when compatibility is not guaranteed, is called a web-native approach (**ionic_web_nodate**). In addition to the time saved by not having to develop platform-specific apps but using a single code base instead, this approach also has the advantage that a library that abstracts the logic of connecting a BLE device such as the IDUN device can already be created into a single installable unit in the form of an NPM package. This library can be implemented in end-user apps and connects directly to the hardware without the need to download a companion app from IDUN itself. This is one of the solutions to the findings from the user interviews. The time saved with a web-native approach allowed IDUN to already focus on creating such a library and using it in their web app, which would be the Console or GUI as in the AWS example, allowing 'dogfooding' for IDUN's own software offerings and eliminating preliminary issues before releasing the library to the public (**techopedia_what_2016**).

4.2.4 Further key events

Further key events of the case study depicted in **??** are described in more detail in **??**.

Chapter 5
Results and summary

This chapter presents the concrete results, a concise summary of the project in the form of key aspects of an N/CI, and an example architecture—all considered in light of the initial goals and objectives.

5.1 Results

This section discusses the final results of the implementation chapter. The author begins by discussing the most important key aspects and insights of an N/CI for the software components of the NIP for IDUN before delving deeper into architectural aspects of the technical implementation.

5.1.1 Key aspects of an N/CI

Previously, the author coined the term N/CI defined by the three-dimensional axes spanned by production-readiness, general applicability, and unobtrusiveness (see **??**). However, **??** does not address what is required to develop a holistic technological definition. Subsequently, the following sections discuss some architectural, ethical, and privacy aspects of the model N/CI built for IDUN Technologies.

Stream-based events

One of the most critical technical aspects of an N/CI is the stream-oriented and event-driven architecture. An event-driven architecture is typical in modern applications, built with microservices and using events to trigger and communicate between decoupled services (**amazon_web_services_inc_event-driven_nodate**). The stream-oriented aspect describes the main events occurring in such a system. Neural data recorded from IDUN's sensors is being streamed from the end-user's devices and processed on the cloud to transform and classify the raw data to generate applicable and intelligible output. There are two main differences between the stream types when building an N/CI:

1. **Synchronous streams** for active and real-time BCI.
2. **Asynchronous streams** for passive, reactive BCI.

Either a user streams neural data from the device to the cloud in order to generate actionable insights—such as controlling an object in a game—or the user tries to improve audio amplification based on where they look. Such use cases necessitate real-time classification within an acceptable latency. (Latency will be discussed more in the following section).

The other component occurs when an asynchronous stream is used in place of a synchronous stream. For example, while asleep, the user does not require real-time insights from their sleep. When they wake up, the system must stop the stream, classify it, and display the sleep stages. These various aspects are crucial because the data stream is required to undergo some sort of data transformation, such as the calculation of frequency band power (typically done in EEG signal processing). Therefore, there needs to be a specific time window (epoch) to calculate a plausible output. To determine the proportion of, say, alpha waves (which typically have frequencies between 8 and 13 Hz) in a user's neural data, one must decompose the raw EEG signal into frequency components via a fast Fourier transform (FFT). An example of FFT transformed data is visualised in **??**.

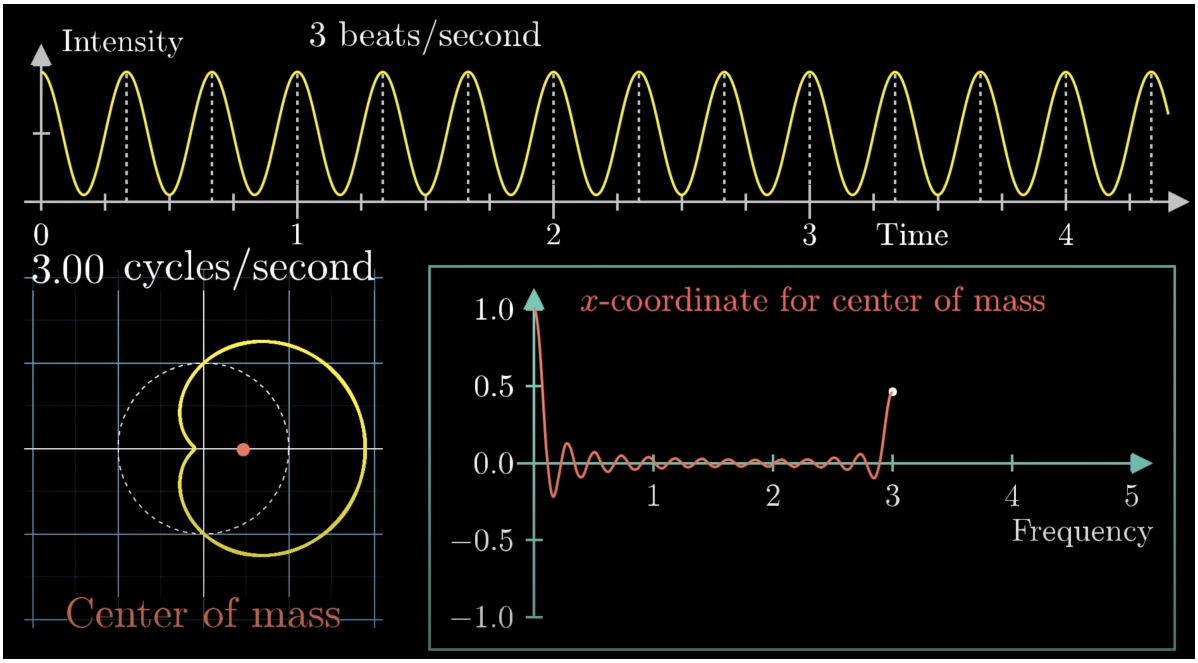

Fig. 5.1: Illustration of how FFT works (**3blue1brown_but_2018**)

A given frequency in the time series is used to calculate the cycles to extract the originally occurring frequency signal based on the epoch length. As a result, there is an epoch-based classification of neural data, on the one hand, in which the classifier makes predictions based on an epoch-by-epoch sequence. On the other hand, there is a per-sample shift classification in which the prediction occurs within a buffer that stores a fixed length of historical samples and moves sample by sample following a first-in-first-out (FIFO) principle. The latency of

the per-sample classification is 0.004 seconds (due to the 250 samples of the IDUN sensor), whereas a per-epoch classification initially has a latency of 1 second due to filling up the buffer for FFT. **??** visualises the differences between these stream transformation types.

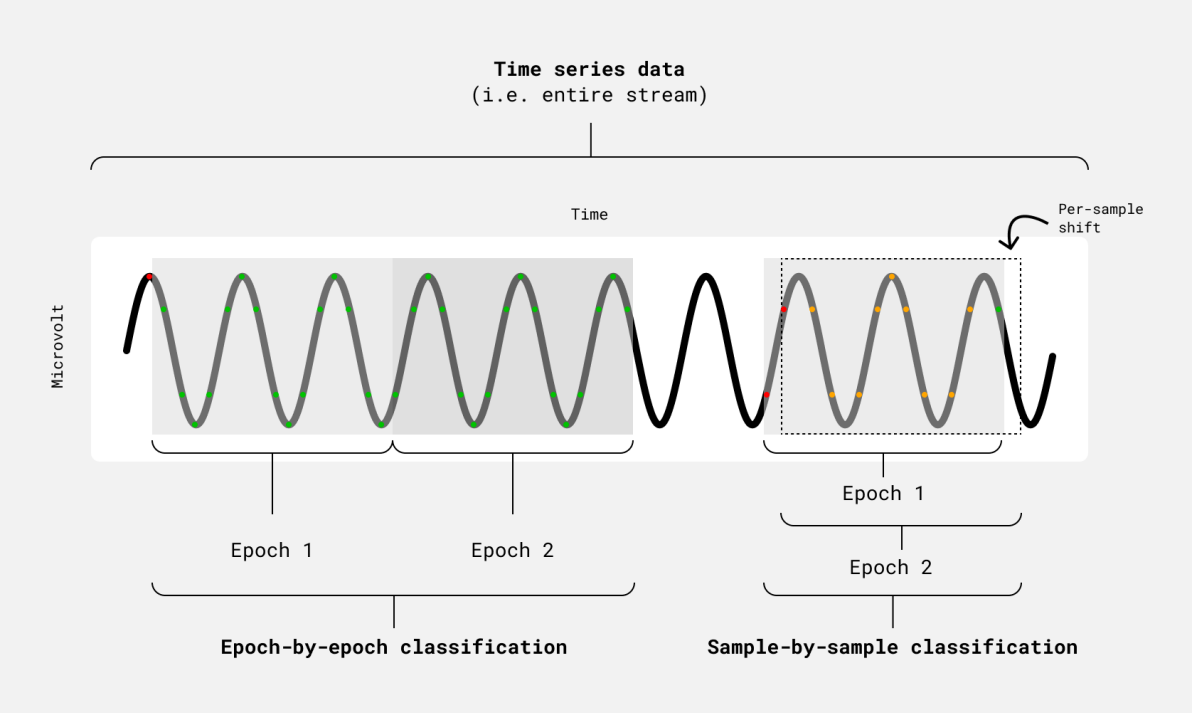

Fig. 5.2: Illustration of two real-time classifications for a BCI application.

Critical and non-critical applications

The discussion of stream-based events leads us to the aspect of critical and non-critical use cases. In a critical use case such as microsleep detection while driving, one needs a synchronous neural stream with a low-latency classification of whether the driver is currently awake, asleep, or about to fall asleep. In such use cases, the classification needs to be as fast as possible, such as in a sample-by-sample classification; what is critical, therefore, is how fast the connection between the measuring sensor and the classifier is. In highly critical use cases where a user's life may depend on it, the classification would need to be done without an active internet connection so that the classification model runs on the hardware itself and not in the cloud. The reason for this is to allow interventions within milliseconds (e.g. in the form of a loud sound to warn or wake up the driver) as a few moments can make the difference between life and death.

Less critical applications can fall back on online modes and run classifiers over an active internet connection in the cloud. To reduce latency, edge cloud computing could be

introduced, whereby business logic executed in the cloud is brought geographically closer to the end user via edge locations (**nomios_what_nodate**). The aspect of edge-cloud or edge-computing in general is a topic not covered under the aspects of an N/CI in this thesis since it was not yet a requirement to build a critical system at IDUN.

However, the distinction between critical and non-critical streams can be made not only for synchronous streams (as described in the preceding examples), but also for asynchronous streams, particularly in the context of research. In general, research represented by, say, IDUN's persona Noel does not necessitate synchronous streams or real-time classifications but rather the most reliable possible acquisition of neural data from test subjects. The critical aspect in such a use case can also be much higher than when recording neural data during sleep in consumer-oriented applications. For example, if the device goes offline or the Bluetooth connection is lost, the device must cache the data locally and be able to restore and send it once the device is back in range or a sufficiently stable internet connection is restored. The author suggests having a fallback system in the form of buffering for each data stream if neural data becomes corrupted or misordered due to timestamp mismatches in sending the data over the internet (meaning that a particular EEG sample could arrive earlier than another sample that would have occurred earlier in the time series). Therefore, a sorting algorithm would need to run over the arriving samples as shown in **??**.

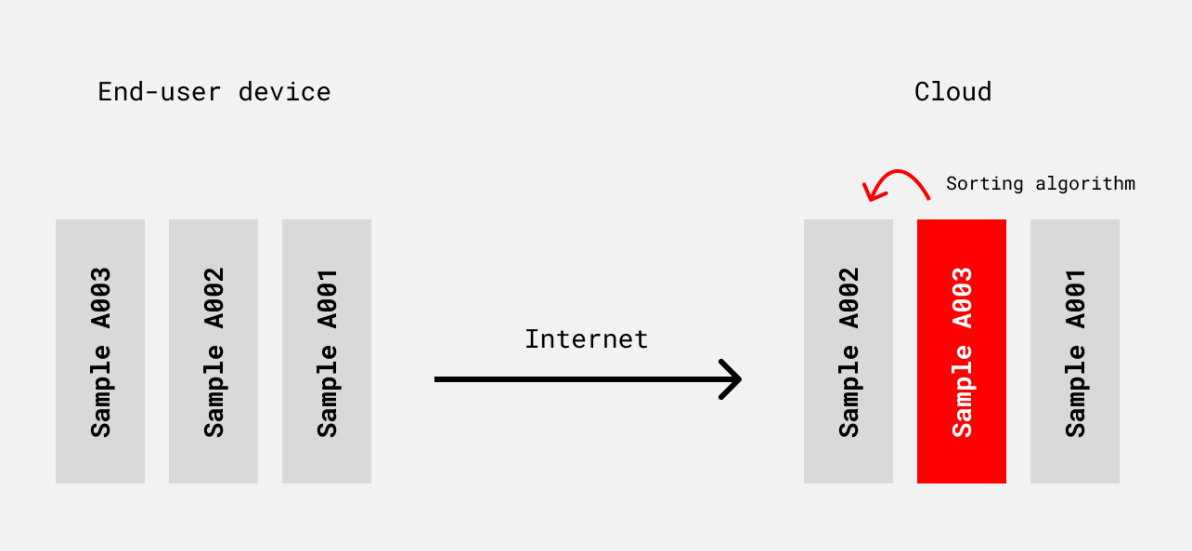

Fig. 5.3: Mechanism of sorting samples as they might arrive in the wrong order on the cloud via the internet.

Fortunately, AWS Kinesis, a streaming service used for IDUN's cloud (more information in **??**), already comes built-in with a guaranteed order mechanism via sequence numbers to solve this problem (**amazon_web_services_inc_amazon_nodate**). The aspect of locally

caching is also important when the device is offline. The software periodically sends a health check request to the cloud via WebSocket in a given interval and buffers all data on the interval. If no promise returns after a certain threshold, the device (or SDK on the end-user device) starts to buffer, as shown in **??**.

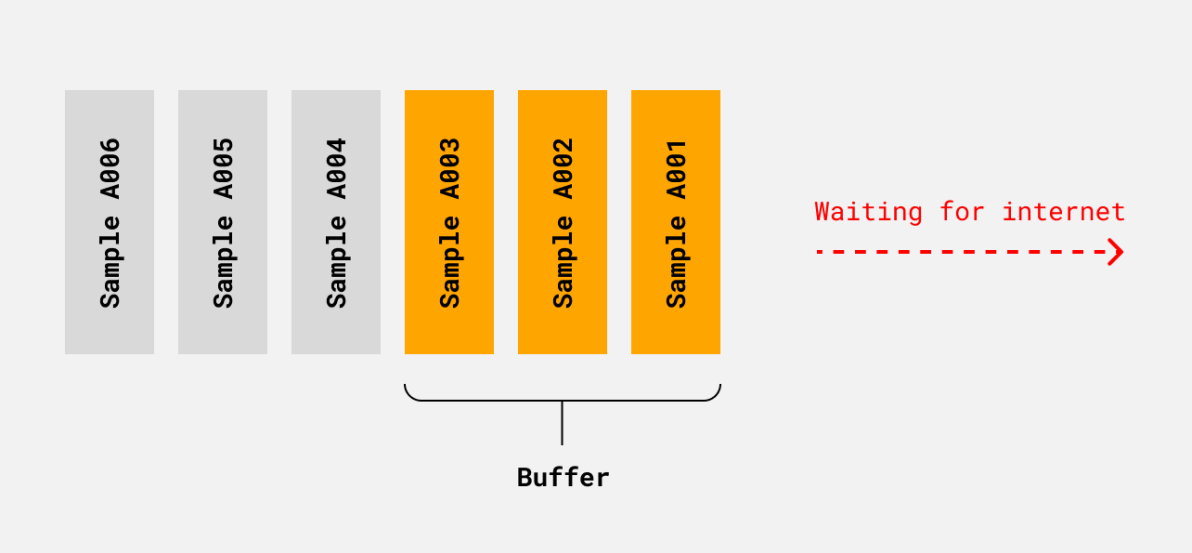

Fig. 5.4: Buffering mechanism on the device (i.e. end-user device such as a smartphone) when there is no internet connection.

Opt-in and encryption

Because IDUN develops an unobtrusive library (in the form of an SDK, as detailed in **??**) that can be easily implemented in existing software such as web apps or mobile apps, the developers of these apps have actual access to the hardware and initiate the Bluetooth connection between the sensor and the end-user device. IDUN must encrypt the recorded data on the hardware before it reaches the third-party app to protect user privacy and the security of end-user neural data.

This is a critical step in preventing third-party providers from collecting extensive data and classifying insights from users they may not want. IDUN securely decrypts the data in its cloud using the private key in the asynchronous encryption example. Third-party developers can thus only access data that the user has authorised. The duration of data storage, the type and level of detail of certain classifications of their neural data, and sharing and usage analysis are all opt-in options. **??** depicts how such an opt-in mechanism might appear as a GUI in a third-party mobile app. A more detailed version can be found in **??**.

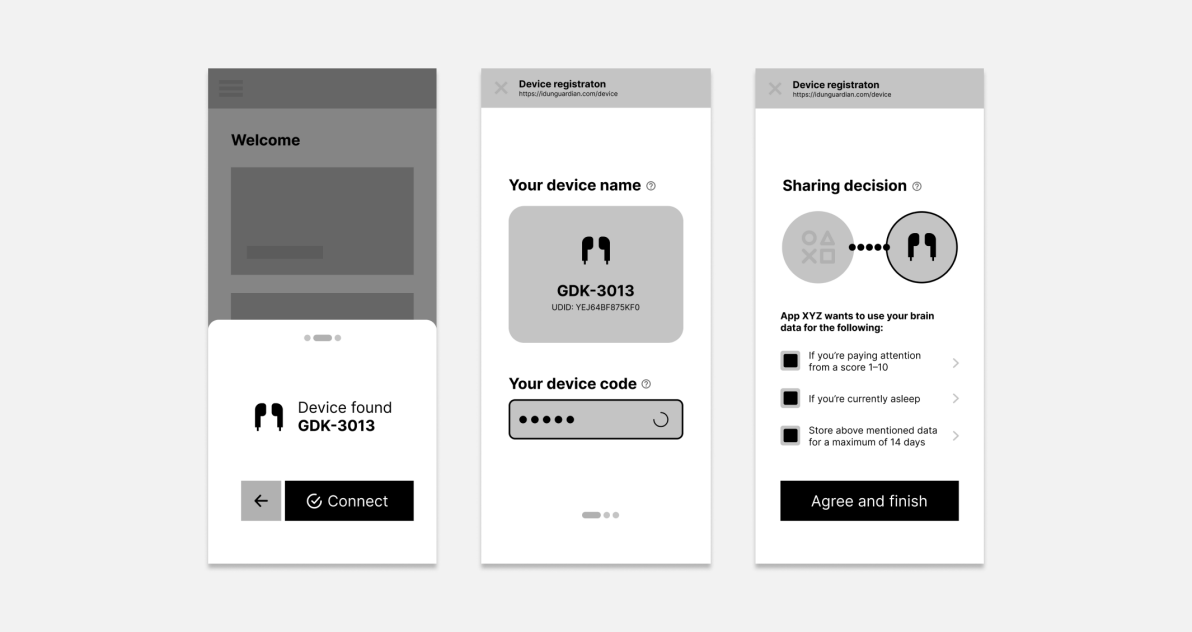

Fig. 5.5: Wireframes for the opt-in mechanism. Starting with the device connection via BLE, the device registration (or login) and then opt-in for app-specific options.

Unencrypted raw data should never be accessible to third parties; moreover, it poses some technical challenges. To visualise and display the raw time series of IDUN's EEG sensors, the individual display images would have to be rendered on the server and streamed to the client as a video stream. Another difficulty is synchronising and organising the rotation of public and private keys, which must be done without interference from third parties. Asynchronous encryption, together with envelope encryption, is a production-ready and battle-tested method to solve this issue. In envelope encryption, the data key (public key) is itself also encrypted with the private key (**google_cloud_envelope_nodate**).

Graph data access

Aside from the stream-oriented, event-driven nature of an N/CI, the aspect of data storage is critical. It is important to note that, due to the high frequency of the EEG data, it became apparent during the implementation phase that storing the neural data in a database is not an efficient method. Storing it as a static file on an object storage system such as AWS S3 in a text-based file format such as comma-separated values (CSV) files seemed more appropriate and scalable since it is too much granular data for a conventional database. This is similar to how other static data such as images get stored on object storage, with only

the URL to the object being stored in a database rather than storing the entirety of the bytes from the image inside the database (**datanamic_store_nodate**).

In addition to storing the neural data in the given data structure, it is essential to note that other sensor data, such as inertial measurement unit (IMU) data, which is also collected by the IDUN hardware to track the subject's three-dimensional location, rotation, and movement, is also stored alongside and synchronised with the neural data. In addition to the data based on the two time series, EEG and IMU, IDUN collects meta-information such as classified versions of the data (e.g. eye movement data), which must also be synchronised with the raw data streams. The more classification and post-processing logic applied to raw data streams, the more metadata and relationships must be linked to the original file, as **??** shows.

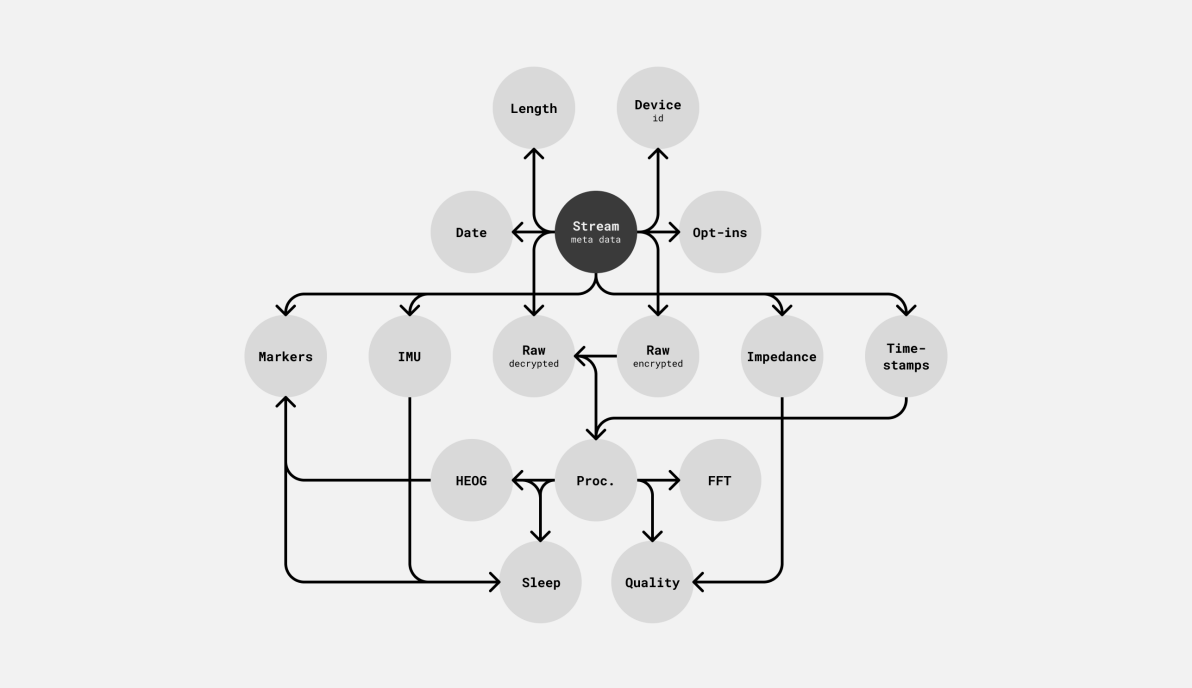

Fig. 5.6: Data model in the form of a graph with the current setup and scope.

A graph database is the most natural way to store this large number of relationships between various stored files (entities). Relationships are treated as first-class citizens in graph databases, and most of their value is derived from these relationships. In graph databases, nodes are used to store data entities, while edges are used to store relationships between entities (**amazon_web_services_inc_what_nodate**). Due to time limitations, the author did not implement a graph database but chose a relational database such as AWS Aurora since the current amount of data relationships is still manageable. To quickly swap

the underlying database technology without rewriting a significant portion of the business logic, the author intends to use an object-relational mapping (ORM) tool such as Prisma to treat relational data models via a graph-like schema (**prisma_data_nodate**).

5.1.2 Example architecture of an N/CI

Based on the key aspects of the N/CI for IDUN's NIP mentioned in the previous section, the author presents as one of the main results the cloud architecture depicted in **??**.

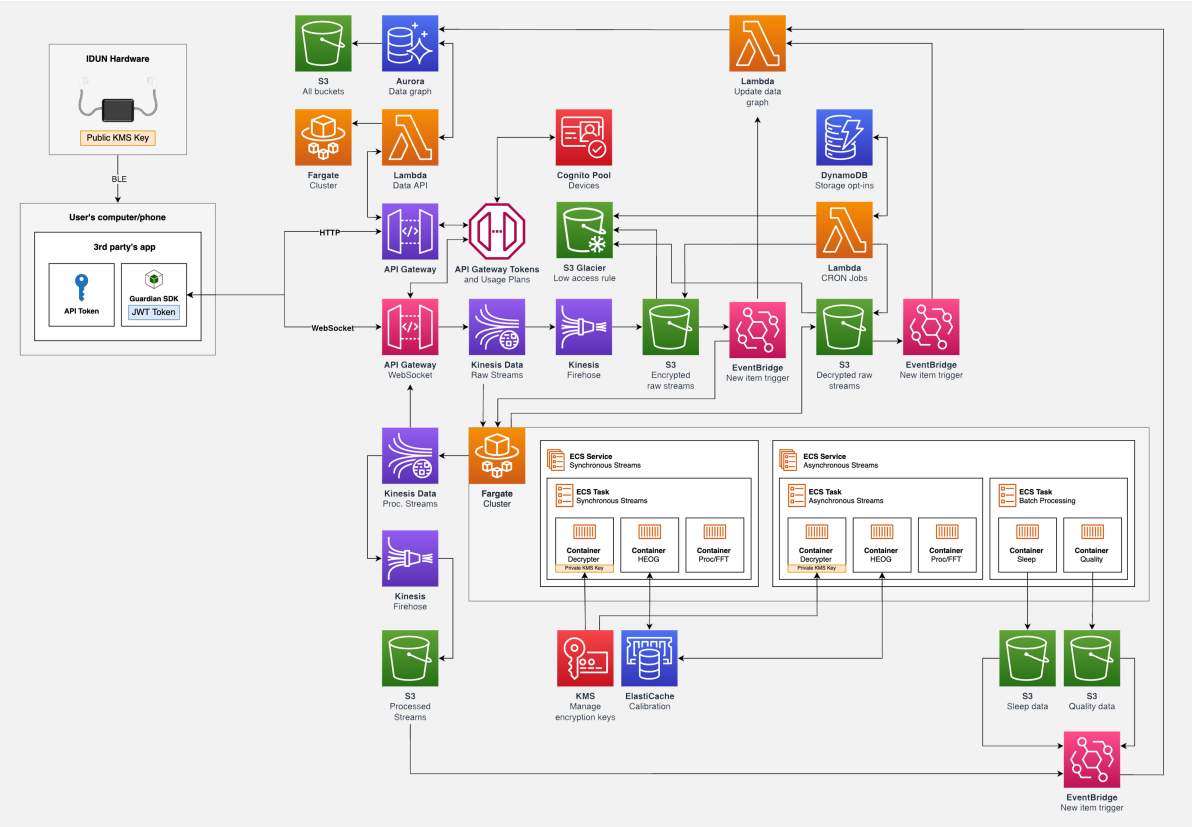

Fig. 5.7: Implementation architecture of IDUN's N/CI.

The architecture does not detail the SDK and demo parts described in the user interview insights in **??**; rather, the goal is only to explore the cloud aspects of an N/CI and its core architectural components. The diagram contains some patterns that have a specific rationale. The following list describes the different patterns and sections of the mapped architecture and the associated technical descriptions. It is important to note that some parts are not discussed as they are not part of this work, such as the networking aspect of the cloud setup—for example, virtual private clouds (VPCs), network address translation services (NATs), and security groups and their traffic rules.

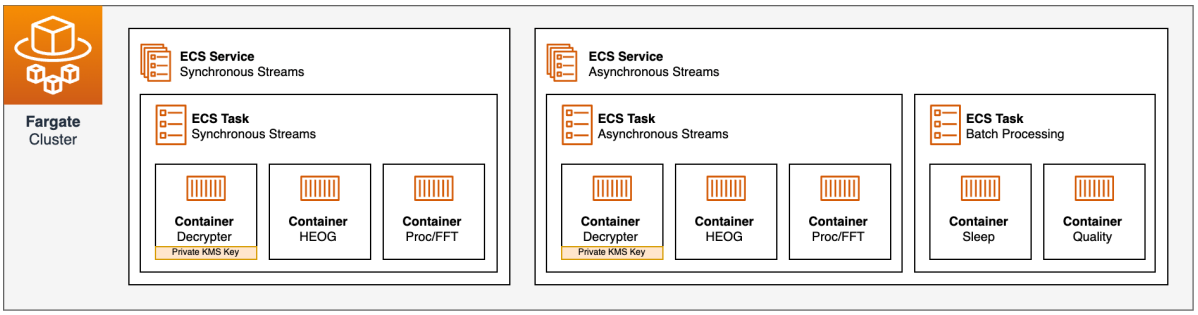

Fig. 5.8: The Fargate cluster of the example architecture for IDUN's N/CI.

- **??** shows the Fargate cluster of the IDUN cloud setup. There are two primary services: one for business logic applied to synchronous streams and one for asynchronous streams. The distinction can usually be described as the left service being a system of record and the right service being a derived data system (**kleppmann_designing_2017**). This separation describes that the system of record processes the raw data directly and, in this case, in a real-time stream, whereas the derived data system generates other data from the raw data without a real-time aspect. There is a certain redundancy, as can be seen, for example, with the HEOG[1] container (a classifier that recognises the direction a user is looking). This is due to the aspect of critical and non-critical mentioned earlier. The HEOG container for synchronous tasks runs on a more powerful machine than, for example, the asynchronous container; consequently, their hardware task descriptions differ in available computing power, as the asynchronous neural data processing is not as critical in terms of latency as the synchronous containers.

  Another aspect is that the asynchronous service also contains batch processing containers for sleep or general quality classifiers that do not need a stream-based aspect. These classifiers would run on demand or as batch processing events to produce other transformed data not represented as a time series, such as neural data or the other stream-based classifiers. For more information on the decision to use a Fargate cluster, see **??**.

  Another important aspect is the decrypted container, which uses the private keys of the key pair to encrypt the data on the hardware and then decrypt it within the services to process and handle the actual and encrypted raw data. When the data is stored in S3 buckets, it is encrypted both in flight and at rest.

[1] HEOG in this context, which is an acronym for horizontal electrooculogram, and describes the electrical signal recorded to detect the horizontal eye movement.

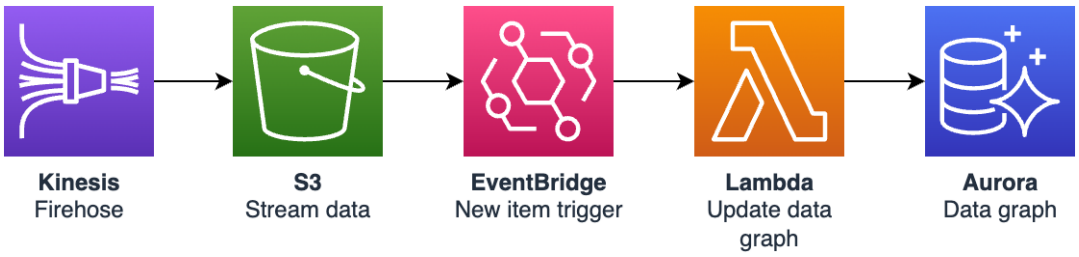

Fig. 5.9: The initial data flow in IDUN's N/CI.

- **??** describes the initial data flow of streamed neural data. The data is sent via the WebSocket protocol to Kinesis, a data ingestion system for data streams. A complimentary service for Kinesis, called Kinesis Firehose, is then used to aggregate the data and store the streamed data as a CSV in an object store—in this case, S3. As more data is streamed through Kinesis, Firehose appends the newly captured data to the CSV file already created on S3. Once the stream is stopped, the CSV file on S3 is marked as complete, which triggers an event in the EventBridge service, which then triggers a Lambda function to add the newly captured file and its meta-information (such as size and identifier) to the data graph database—in this case, the Aurora database service. This event-driven aspect allows the Fargate cluster not to be disturbed and the focus to remain on the streaming aspect with the creation of the data graph being event-driven. In contrast to creating another service in the Fargate cluster, a serverless function in Lambda is not always used to run this business logic because they are event-driven and per-request; thus, they can be cold rather than hot functions[2], which is what classification logic on the Fargate cluster should always be as people can stream EEG data continuously for hours.

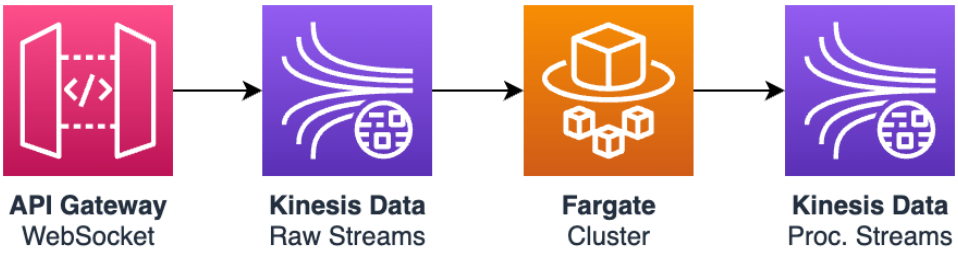

Fig. 5.10: The real-time data flow in IDUN's N/CI.

- **??**, compared to the previous bullet point, shows the real-time data flow. It describes that neural data is sent via the WebSocket protocol and then ends up in the ingestion

[2]Hot and cold functions describe the underlying state of the infrastructure on which business logic is executed. The serverless aspect of cloud computing functions, such as Lambda, can be 'turned off' as soon as the business logic is not needed at the moment and is therefore described as cold.

service of Kinesis. Kinesis stores the data stream and the new incoming data stateful and leaves it to the Fargate cluster to copy and transform the data according to the desired classification, such as HEOG or FFT. Once the Fargate cluster returns the processed or transformed data, it is sent to another Kinesis stream, which then returns the data via the same WebSocket API gateway. As a client, one would only subscribe to one API WebSocket API endpoint in this architecture, although the streams are processed separately with two ingestion services (the Kinesis instances Raw Streams and Proc. Streams).

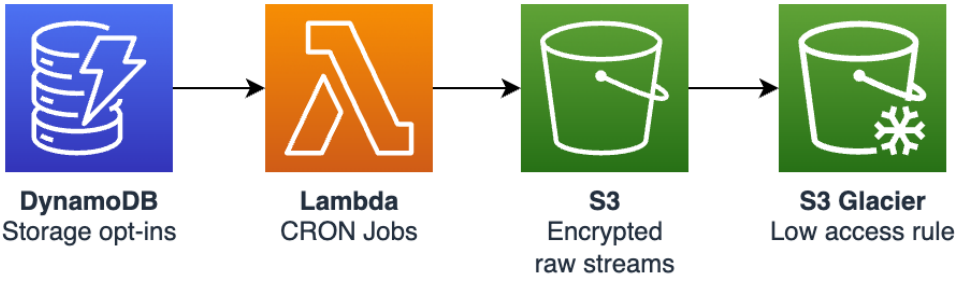

Fig. 5.11: CRON jobs example architecture in IDUN's N/CI.

- **??** describes the CRON[3] job processes to process and dictate specific opt-ins from users regarding long-term data storage. For example, if the user has declared that IDUN should not store sleep data in the cloud for longer than four weeks, business logic must check which files are older than four weeks and then delete them. This task of repeatedly checking opt-ins stored in a DynamoDB database can also be handled by a Lambda. DynamoDB is a NoSQL database perfect for data whose schema is challenging to track due to the complexity of opt-in variations. This business logic can happen four times a day, for example, and has no real-time impact on end users, so this is a good use case for using business logic that can be cold.

  Another aspect is that the CRON job Lambda defines old files, for example, and places them in cold storage on S3 Glacier. One can transfer old and infrequently used data from warm to cold storage in seconds while the data remains discoverable and searchable (**sayed_introducing_2021**). This is a process of removing less frequently used or requested data from the N/CI's computational space, which primarily saves cloud costs.

[3]CRON comes from the command line utility that is a job scheduler on Unix-like operating systems to execute tasks repeatedly without human intervention.

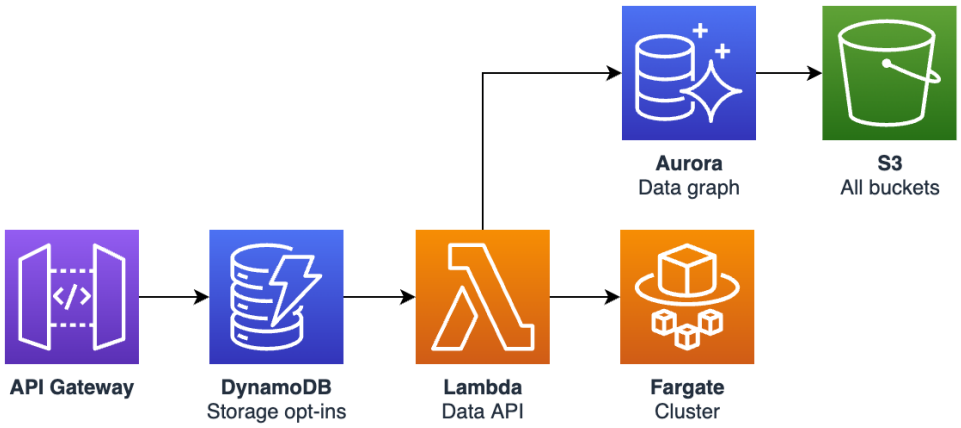

Fig. 5.12: API access part of the architecture of IDUN's N/CI.

- **??** shows the API access part of the example architecture for IDUN's N/CI. The author uses the API Gateway service here—not the WebSocket version, but the standard Hypertext Transfer Protocol (HTTP) version—as there is no real-time and stream-based aspect to this API. Based on the opt-in of the users, which is accessible via the API for third-party developers, a Lambda accesses the Aurora database, which contains the references to all data stored in S3 buckets within the cloud. For example, if certain data is not available (e.g. if the device is only streaming raw data for sleep classification, but the developers want to get the FFT version of the file), then the data API Lambda sends a request to the Fargate cluster to process the data via a batch processing service, as shown in **??**. The updated data is then stored in an S3 bucket directly by Fargate; this triggers another event in EventBridge, as shown in **??**, which then updates the data graph. Once the data graph of the specific transformed version of the file has changed, it can be returned as an HTTP response via the API Gateway service.

5.2 Summary

Within this thesis, the author has provided the foundational work on the development and definition of an N/CI. In addition, the author has addressed the challenges and the limitations surrounding the implementation of a model N/CI at IDUN. Working in a fast-moving startup such as IDUN made it challenging to work on the project and write the written part; nevertheless, the author is satisfied with the result as described in this thesis. The author learned many things, ranging from fundamental theoretical neuroscience to cloud computing.

This thesis first introduced the topics and some of the essential terms and their backgrounds in **??**, drawing conclusions from the current state of research that have framed the aims and objectives of the author. The assumption that the implementation of an N/CI is feasible with contemporary technologies has been demonstrated, but there are still unknown shortcomings that need to be identified. Nevertheless, the aim was to provide the reader with an overview and context for the newly introduced definition of N/CI and the lessons learned during its practical implementation at IDUN. The context and motivation for creating an N/CI, as described as the first item in the list of objectives, were derived in Chapter 2 and explained in more detail in **??**. Compared to existing systems and research, the clear definition, distinction, and advantages of an N/CI were also described in more detail in **??**. The identification and definition of key aspects to realise an N/CI to demonstrate general applicability for BCI software were shown using key events in **??** and **??**. The implementation details of an example architecture of an N/CI, which was the fourth and final objective of this thesis, were presented in **??**, leading to a summary of all the goals and objectives achieved in this project.

5.2.1 Reflection

Creating an example N/CI in the industry as part of IDUN Technologies' NIP was a complex undertaking with many moving parts and imponderables. Moreover, since no other person is publicly building an N/CI or anything that would fall under the definition of an N/CI, the author could not count on experienced engineers who have built such a system for a mass-market BCI before. The most significant problem in the context of this work was the sheer scale of the project, as reflected in the length of the thesis. In hindsight, it might have been better to focus more precisely on one deliverable rather than having four objectives and primary goals. The author should also have defined more clear criteria for what the work packages should have been and what defines them as 'finished'. The work was more theoretical than initially anticipated and focused on the definition and key aspects of an N/CI, ultimately a byproduct of the original work.

5.2.2 Outlook

The author intends to publish certain parts of this work in the form of a research paper. An exciting aspect of this is the definition and motivation behind creating the new discipline in which neural/cloud interfacing resides. In addition, the author will continue to work at IDUN Technologies to develop their N/CI further and document the upcoming technological challenges in the form of blog posts.

One important change that will occur once IDUN sells its device to the general public is that the currently proposed architecture, as shown in **??**, will become a localised system—that is, the cloud data centre is in one geographical location (in the case of IDUN, this is EU-Central-1 in Frankfurt). This aspect could be worked on as part of a master's thesis. For future research, it could be fascinating to explore a distributed cloud system as scaling around the world would require a distributed system to reduce latency and fault tolerance.

5.2.3 Conclusion

An N/CI is an essential and new definition of a discipline that combines cloud computing and BCI software. This new discipline brings many new challenges, such as making a neural interface software system ready for production, which invites numerous privacy and ethical challenges. Building an N/CI along the lines of the definition explored by the author was an exciting and highly challenging endeavour, but one that led to many valuable insights and findings. There is a need for further research into performance and latency aspects, as well as the totality of machine learning models running in production, which was not mentioned and was not the focus of this thesis. Nevertheless, many findings from user interviews, group discussions, and expert interviews were presented and visualised along with the holistic model architecture, which can be used further to develop better and more robust N/CI architectures through future case studies. Following the publication of a shorter and more scientifically sound version of this thesis, as mentioned in **??**, the author hopes to promote and establish the term and discipline of N/CI and the future behind it from the perspective of interacting directly with brains and better understanding our brains—for the benefit of humanity.

Appendix A
Definition and motivation of an N/CI

This appendix discusses the definition, need, and differentiation of an N/CI and the paradigm shifts associated with it when discussing BCI software.

BCI software on the cloud

As can be seen in **??**, the three software layers of a BCI-component illustration, as pictured in Figure 2.7, are highlighted. This is due to the fact that these components are not bound to a physical interface such as a cable or Bluetooth.

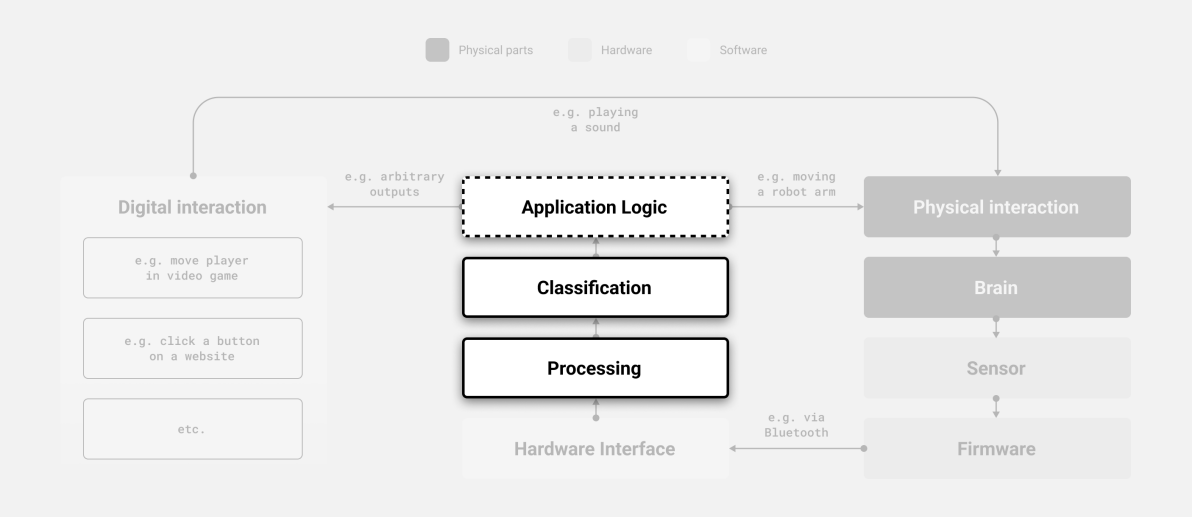

Fig. A.1: Highlighted software components as previously shown in **??** of a BCI that could be moved to the cloud.

Running software on the cloud means that developers or companies can access provisioned information technology (IT) infrastructure through the internet, usually with a pay-as-you-go pricing model (**amazon_web_services_inc_what_nodate**). The development speed of software applications can drastically improve since software developers can only focus on software rather than incorporating the hardware and network aspect of setting up their server farms, therefore abstracting away the hardware. What started with simple computers that can be rented in a remote server farm such as with Amazon Web Services (AWS) and Elastic Compute Cloud (EC2; **barr_amazon_2006**) ended up being a diverse offering from cloud providers with various abstraction levels, as shown in **??**.

Type	Description
Infrastructure as a Service (IaaS)	IaaS gives access to data storage space, virtual or dedicated computers, and network services. The greatest degree of flexibility and administrative control over your IT resources are provided by utilising IaaS.
Platform as a Service (PaaS)	PaaS lets developers concentrate on developing and managing their code rather than worrying about the underlying infrastructure (often hardware and operating systems). An example is Kubernetes.
Software as a Service (SaaS)	SaaS provides a whole product that is run and controlled by the service provider. The phrase SaaS often refers to end-user apps (e.g. web-based email). Developers don't have to be concerned about how the service is handled or whether the underlying infrastructure is maintained.

Table A.1: The three abstraction levels and types of cloud computing (**amazon_web_services_inc_what_nodate**).

Most businesses are anticipated to embrace a cloud-first strategy by 2025, according to Milind Govekar, vice president of IT research and consultancy company Gartner, and will not be able to fully implement their digital plans without the usage of cloud-native architectures and technologies (**gartner_gartner_nodate**). The impact and importance of cloud computing cannot be underestimated, and its success is also reflected in the annual spending on cloud computing resources, estimated at €474 billion in 2022 (**gartner_gartner_nodate**). Cloud computing is such an extensive and complex topic that it could quickly fill entire books. The author categorises three broad but essential points that certainly play a vital role in BCI software: (a) dedicated and deterministic environments, which means that an environment of a software programme always stays the same independent of the end-users hardware; (b) elastic and high-performance availability, which refers to cloud computers that have an on-demand and adjustable high-performance; and (c) provided services for speed, which refers to the concept of pre-made and dedicated software written primarily for the cloud and specific use cases. The following list goes into more detail about these topics in the context of BCI software:

1. **Dedicated and deterministic environments:** Running code for BCIs on different end-user platforms, such as Windows or Android, can have drawbacks because each device has its processor, graphics card, operating system version, and drivers. This can make developing software that requires stable and good performance—such as neural data

processing pipelines—time-consuming and difficult to maintain, as developers must keep track of every factor of the end-user devices. This is fine for BCIs that are not intended for the general public, such as specially designed BCIs for people with, say, locked-in syndrome; but for the general public, a vast amount of different end-user devices could come into play. When code runs on a dedicated machine, such as a cloud computer with clearly defined hardware and operating system specifications, it becomes less error-prone and more deterministic.

2. **Elastic and high-performance availability:** Because the cloud usually runs as an as-needed model, the initial purchase cost of computer hardware is split and shared across usage. Developers have access to tremendous computing power that would not be easily afforded if purchased independently. As a result, when developing a computationally intensive algorithm, developers can use high-performance central processing units (CPUs) and graphics processing units (GPUs) to complete tasks much faster than consumer hardware on end-user devices. Performance can also be increased as needed, for example, to handle computationally heavier tasks that are not used as frequently or to handle more requests when the demand for the software increases due to an increase in the number of users—a process known as elasticity (**gartner_definition_nodate**). Furthermore, the cloud provides far more storage capacity than end-user devices.

3. **Provided services for speed:** The vast majority of cloud providers are providing more specialised services as we move closer to PaaS. Provisioned database servers, for example, exist solely to serve as a database; the underlying hardware is therefore optimised for the database software running on it. Hundreds of cloud computing services are available, including 200 from AWS alone (**amazon_web_services_inc_what_nodate**)—all of which address specific use cases. This is extremely useful when it comes to cloud-based BCI software. One example is the live streaming of brain data, which is discussed in greater detail in Chapter 4. The services accelerate development speed by eliminating the need for teams to reinvent the wheel repeatedly, in addition to also providing out-of-the-box scalability due to built-in elasticity and the concept of fully abstracted hardware via the serverless model (**redhat_what_2022**).

An N/CI utilises cloud computing by running certain software components of a BCI system in the cloud with the benefits previously mentioned. Multiple BCIs can communicate with each other or with other software or hardware over the internet, enabling remote human-computer interaction. **??** depicts how two or more BCIs can interface with one another using cloud software. The red arrows represent the typical local BCI. The blue arrow

depicts how BCI B can execute business logic via the cloud over the internet, enabling digital and physical interactions with high performance. The green arrow depicts how BCI A can communicate via an N/CI with BCI B even if they are not in the same geographical location.

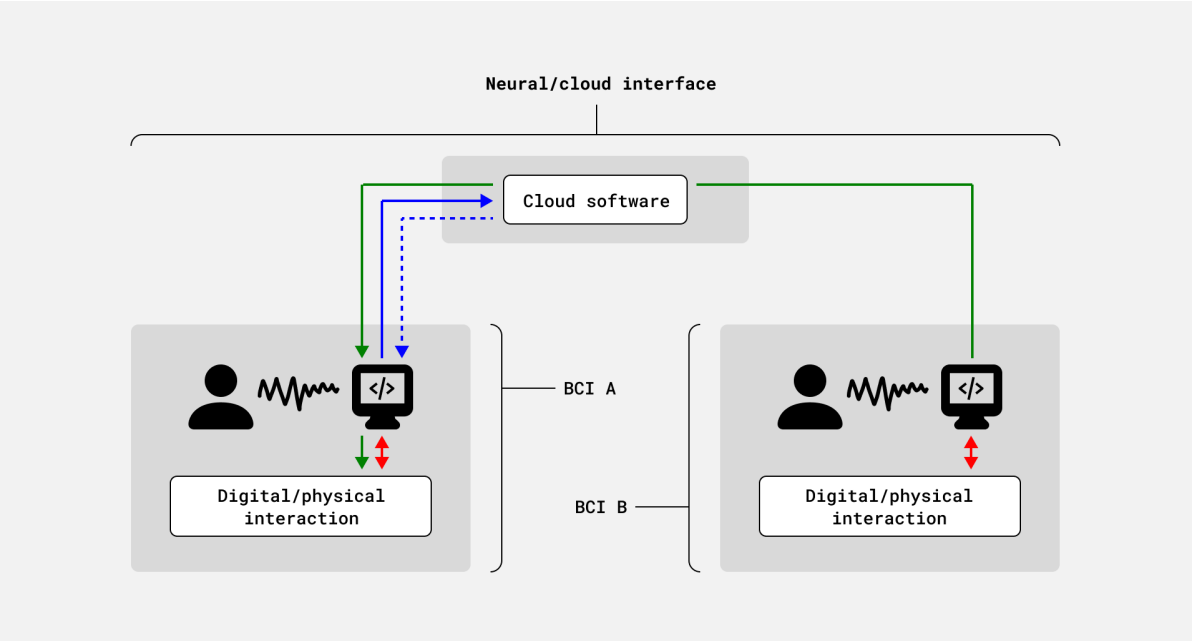

Fig. A.2: An N/CI is the connection between multiple BCIs.

Distinction between existing research

The concept of running BCI software components remotely and on the cloud is not novel, as research into the concept known as asynchronous BCI (**an_design_2016**) or internet-based BCI (**lampe_brain-computer_2014**) has been ongoing for some time. However, when one looks at the more recent research from **zhang_internet_2018** (**zhang_internet_2018**) on their cloud-based deep learning framework to enable what they describe as Human-Thing Cognitive Interactivity, one sees a strong emphasis on algorithms and machine learning but less on cloud architecture and mass-market-readiness (**zhang_internet_2018**). They address the latency and size of EEG samples sent in near-real-time to a server, as well as the corresponding calculation, but there are no more details in regard to the proposed and very simplified architecture chart's components or effective cloud architecture, all of which factor into the author's aim to develop an N/CI. Another related and even more recent paper by **ahamad_system_2022** (**ahamad_system_2022**) looks at the system architecture of a BCI for the Internet of Things (IoT), but this time from the perspective of algorithms optimised

for time series data such as EEG, but still with no mention of the effective cloud architecture of such a system (**ahamad_system_2022**).

The author of this thesis introduces the concept of three-dimensionality for the definition of an N/CI based on the previously mentioned topics and research that touch on the issues of this thesis, which are essential to achieve general applicability for BCIs from the perspective of the software system for the actual implementation of such a system.

Requirements of an N/CI

The term N/CI positions itself as a software system in the intersection that can undoubtedly be defined as production-grade rather than in the PoC stage, unobtrusive implementation rather than obtrusive software, and general-purpose applicability rather than made just for a specific use case. **??** illustrates the position of an N/CI within the mentioned properties. Subsequently, **??** summarises the definitions as described in this thesis.

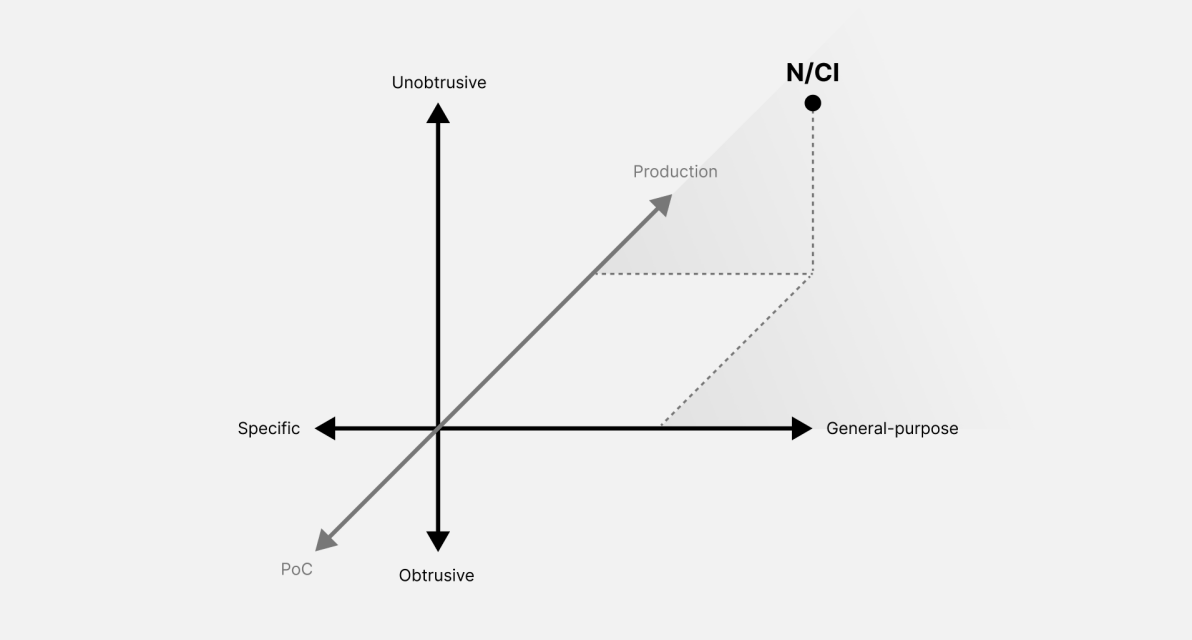

Fig. A.3: Visualisation of the three-dimensionality of the term neural/cloud interface with its three axes and differentiation of six terms.

Axis labelling	Description
Production	As previously stated, this is the range in which a software system is deemed ready for production. Because the definition is vague, it is difficult to identify specific requirements that must be met for a system to be considered production-ready. However, for an N/CI, this means running in a production environment, for example, in a cloud, on a real-world end-user server rather than, for example, in a proof-of-concept environment such as in a lab.
Unobtrusive	As previously stated, an unobtrusive software system is one in which the end-user does not need to understand the underlying architecture and requirements in order to use the software or even know certain parts of the software system. In the reverse case, users need to install special packages or download additional companion apps to make an N/CI work on their computer or smartphone is not the aim of the author's definition of an N/CI.
General-purpose	A general-purpose software system is one that can be used for a variety of functions. As an example, consider AWS. It is general-purpose, which means that developers can create cloud software on AWS that can be either a financial application or a backend for a mobile game; there are no specific use cases. This means that an N/CI, unlike the NextMind BCI or the Muse headband, should provide general-purpose functionality rather than specific use cases, i.e. it is not limited to a specific set of functions.
PoC	A BCI software system that serves as a proof of concept cannot be considered N/CI because it is not intended for production and thus all the effort required to create a production system, such as quality assurance with unit or end-to-end testing, is unnecessary. A PoC system does not usually run in a production environment, as the goal is, for example, to test a specific functionality of a use case rather than to deliver the software to end-users.
Obtrusive	An obtrusive BCI system, for example, may still be production-ready if it targets specific use cases while remaining unobtrusive because neuroscientists or developers, for example, expect to have access to the underlying software architecture or technical requirements and thus do not want to abstract from it. If it is obtrusive software, such as OpenBCI, this usually means building the production-ready part on top of it is necessary, which does not fit the author's proposed definition of an N/CI.
Specific	If a BCI system is only used for one use case, such as Muse for sleep and meditation, companies or developers who want to offer a different use case, such as a mind-controlled keyboard with a P300 system will have to reverse engineer Muse's EEG output or use a different BCI hardware that is less specific and closed, and then build their own production-ready and unobtrusive software on top of it.

Table A.2: Axes label descriptions of the three-dimensionality for the definition of an N/CI as shown in **??**.

With an unobtrusive form factor like the one developed by IDUN Technologies, a significant hardware barrier to becoming a mass-market BCI has already been tackled. Next to the hardware, IDUN intends to provide a business-to-business (B2B) software platform, allowing third-party developers to create software on top of IDUN's offerings through a universal brain application programming interface (API). Because they allow others to consume this API in end-user-facing apps, it must be production-ready and able to be integrated unobtrusively. IDUN's hardware and software aim to be general-purpose rather than specific, allowing others to build any BCI-enhanced app (**idun_guardian_nodate**).

All IDUN's requirements systematise one of the first BCIs aimed at the general public, which satisfies the broad definition of an N/CI. The fundamental motivation of the author is to standardise collaboration and research on this novel and interdisciplinary field of BCI software and cloud computing.

Appendix B
Flaws and bugs previous system

This appendix lists the flaws and bugs discovered during the author's work on the original IDUN Technologies PoC software system.

- AWS Amplify is great for frontend developers who want to build simple backends with CRUD[1] operations, but it is not intended for anything custom-made, such as the streaming-focused aspect of EEG data. Therefore, AWS Amplify must be abandoned as soon as possible, or the project will be built with the wrong tools and foundations.
- The network bridge was a Raspberry Pi 4 Model B running Python code which, after some analysis, turned out to be the primary source of most of the bugs due to limits in computational power and ARM-based CPU architecture.
- The cloud's heartbeat functionality was missing, which meant that the cloud knew nothing about the hardware devices, and simply assumed that data would flow in as soon as the start command was sent to the device, creating a 'happy path' scenario.
- The cloud infrastructure was automatically provisioned by AWS Amplify, which uses AWS CloudFormation in its core. CloudFormation is an infrastructure as code (IaC) tool, that runs inside AWS CodePipeline, AWS's continuous integration service. This tool stack made everything coupled to specific AWS services where the technical decision to use them was not made based on reasoning but was based on Amplify's creators to utilise them.
- All software in the cloud was built using the AWS Console (the AWS GUI). Therefore, no current state in the form of IaC reflected the current infrastructure, which made it difficult to reproduce the cloud in different environments, for example, such as preventing a blast radius if something went wrong.
- The data was streamed via the messaging and queuing middleware telemetry transport (MQTT) protocol developed by IBM in the late 1990s (**yuan_getting_2017**). MQTT is a publish-and-subscribe protocol commonly used for IoT devices that regularly send telemetry data. The purpose of MQTT was not to send high-frequency EEG data in real-time but rather to minimise network bandwidth (**mqtt_use_nodate**). Therefore,

[1] CRUD is an acronym describing general operations of a backend system: create, read, update and delete.

there was also a need to rethink this technological decision based on the nature of IDUN's EEG sensor, which records EEG data at 250 Hz (250 samples per second).

- The SPA was a thick client, meaning that it ran a lot of business logic, such as filtering raw data for real-time visualisation. This was another technological misstep, as the client-side JavaScript ecosystem is far inferior to the Python ecosystem that could run in the backend to handle such tasks. If possible, shipping business logic that could hold intellectual property to clients should be avoided, if possible, especially with a commercial product.

- The SPA was not connected to a single endpoint on the backend and used the MQTT stream and the non-real-time aspects of the app (e.g. login or list of recorded EEG data) via different sources. For example, the MQTT stream was subscribed directly from the device itself and did not run through AWS Amplify's API, which made it cumbersome to couple the systems into a coherent and robust API.

- The state of the entire system was difficult to handle due to the decoupled logic from the MQTT stream and Amplify's API. Combined with the lack of a hardware heartbeat, it was tedious to figure out what the user was doing and what was being sent. As an interim solution, AWS ElastiCache—a provisioned service for in-memory databases such as Redis—was set up. Unfortunately, the application state was now both handled in the SPA and simultaneously on ElastiCache, which led to new problems such as sending EEG data to the void if the user closed the browser during a data stream.

Appendix C
User interviews and outline

This appendix chapter lists the outline and rough framework used to conduct the user interviews, as well as the notes taken by the author and IDUN's product manager, Mark Melnykowycz, during the interview sessions. The notes are represented in the state they were in at the time of recording. The focus was on conducting user interviews especially in the Robbie and Evan personas, as they were the first actual users of the software platform targeted by the C-level management.

User interview outline for the Robbie persona

Notion PDF export to be found in the file *user_interview_with_a_robbie.pdf* in the appendices directory in the user interviews subdirectory.

User interview outline for the Evan persona

Notion PDF export to be found in the file *user_interview_with_an_evan.pdf* in the appendices directory in the user interviews subdirectory.

User interview with Paul Doyle

Notion PDF export to be found in the file *user_interview_with_paul.pdf* in the appendices directory in the user interviews subdirectory.

User interview with Ghena Hammour

Notion PDF export to be found in the file *user_interview_with_ghena.pdf* in the appendices directory in the user interviews subdirectory.

User interview with Nicole Zahnd

Notion PDF export to be found in the file *user_interview_with_nicole.pdf* in the appendices directory in the user interviews subdirectory.

User interview with Melanie Baumgartner

Notion PDF export to be found in the file *user_interview_with_melanie.pdf* in the appendices directory in the user interviews subdirectory.

User interview with Mayank Jain

Notion PDF export to be found in the file *user_interview_with_mayank.pdf* in the appendices directory in the user interviews subdirectory.

User interview with Martin Hutchings

Notion PDF export to be found in the file *user_interview_with_martin.pdf* in the appendices directory in the user interviews subdirectory.

User interview with Jacopo de Araujo

Notion PDF export to be found in the file *user_interview_with_jacopo.pdf* in the appendices directory in the user interviews subdirectory.

User interview with Garrett Flynn

Notion PDF export to be found in the file *user_interview_with_garrett.pdf* in the appendices directory in the user interviews subdirectory.

Appendix D
Further implementation key events

This appendix lists other key events that took place during the 10 sprints of the project to create the next version of IDUN's NIP.

Python to WebAssembly

Another key event during the 10 sprints took place about two sprints before evaluating the web-native approach. The idea was that, as mentioned in **??**, when it comes to how important and necessary the Python ecosystem is for IDUN and external people as presented in the personas, it might be possible to create Python code for specific neural signal data processing, compile it to WebAssembly (a low-byte compilation target for high-level languages that run mainly in the browser) and then use it in web applications such as the GUI application of IDUN's NIP. Sprint 5 evaluated compiling Python into WebAssembly, which seemed promising at first but turned out to be too far removed from current capabilities. There is an interesting open-source project called RustPython—a Python interpreter written in Rust that allows the interpreted code to be compiled in WebAssembly. However, the project's maturity is currently unsuitable for production (**noauthor_rustpython_2022**).

Another option would be to use Pyodide, which is the CPython interpreter ported to WebAssembly (**noauthor_pyodide_2022**). Apart from the fact that the creation of a web application would be drastically more significant if a complete Python interpreter were sent to the browser, this interpreter also does not currently have the functionality to add additional Python packages apart from the standard library. This makes it easy to build vanilla Python on top of WebAssembly codebases, but not with custom packages installed, such as machine-learning packages needed for EEG classifiers. The only options left were to develop one's own Python-to-WebAssembly compiler, contribute to the two open source projects, or give up the task of embedding Python code in a web application such as IDUN's GUI application to interact with NIP's API. The latter was chosen because it was still too early, immature, and time-consuming, as the results of Sprint 5 showed.

Kubernetes and Fargate

One of the first technical decisions the author made involved which technology to use for creating backends for an n-tier[1] system such as IDUN's NIP. The use of a microservice approach was already established by the fact that a polyglot backend was needed, such as Python for data-heavy tasks and TypeScript for real-time and API-specific tasks due to the maturity in the previously mentioned examples of each language. If one decides to take a microservice approach, it can still be debated whether to proceed with serverless functions instead of hosting container images. Given the requirement to use AWS mentioned in **??**, the only way to host serverless functions was to use AWS Lambda. AWS Lambda was already used in the previous PoC version of the software system at IDUN and did not work as well as in cases where, for example, batch processing of previously collected data would exceed the maximum computation time or CPU limit of AWS Lambda. This limitation, coupled with the challenge of handling dozens if not hundreds of serverless functions and managing the entire collection of multiple functions and their versions and endpoints, led to the decision not to use Lambda for the most essential and critical parts of the backend in the author's experience so far. Building microservices without serverless functions on AWS provides multiple solutions: the most prominent ones are utilising Kubernetes via AWS Elastic Kubernetes Service (EKS) or AWS Fargate, which is a serverless version of Kubernetes (**amazon_web_services_inc_serverless_nodate**) that makes some parts of Kubernetes easier to handle, as shown in **??**.

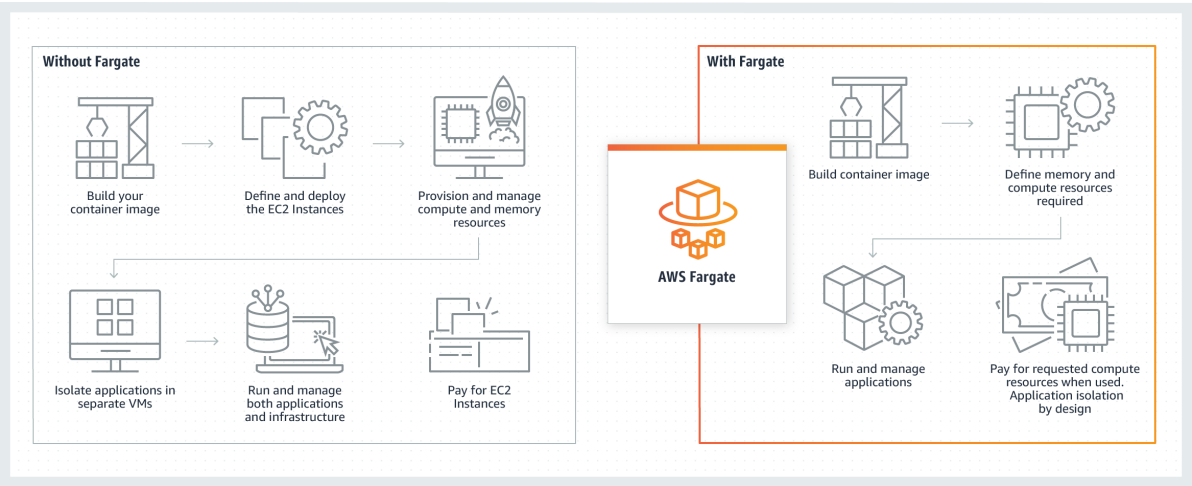

Fig. D.1: Comparison of Fargate versus other container management tools (**amazon_web_services_inc_serverless_nodate**).

[1]The term n-tier refers to an architecture design pattern where the logic of presentation, application processing, and data management are decoupled from each other, as in the case of multiple microservices.

The author consulted cloud experts in the first sprints from the company Nuvibit to guide the decision. The author himself is not experienced in Kubernetes and would need to learn many ways to implement a Kubernetes cluster, which was one of the items to avoid mentioned in **??**. Nuvibit recommended going with AWS Fargate since it is similar to Kubernetes but abstracts most things away to concentrate on adding business logic rather than handling the overheads that come with introducing a Kubernetes cluster. Fargate is also essentially serverless, making it cheaper for a product with hard-to-estimate usage such as IDUN's NIP in the beginning; but it still gives enough flexibility in specifying the underlying hardware for computationally heavier tasks that would exceed AWS Lambda. As soon as IDUN moves toward large-scale deep learning models that utilise specific GPU units, Fargate will come to its limits since specifying GPU tasks is impossible (**amazon_web_services_inc_aws_2019**). One way or another, this decision will have to be reconsidered in the future to avoid a long-term commitment to one provider such as AWS and to the increasing investments of Kubernetes from several cloud providers.

Kafka and Kinesis

Another early technical decision, made with the help of expert interviews, concerned which streaming technology to use. As with many things in software, there are hundreds, if not thousands, of ways to solve a problem, such as streaming text-based data such as the EEG data from IDUN's device over the internet to the cloud. Initially developed by LinkedIn, Apache Kafka is a prominent technology for such a use case. It is battle-tested, used in production by large technology companies (**apache_apache_nodate**) and well maintained by an active community (**apache_apache_2022**). However, similar to Kubernetes, it comes with much overhead; moreover, the author himself has no experience with Kafka, so he would have to learn everything from scratch. Another option is to use AWS Kinesis, a service that AWS provides to process data streams over the cloud, as Kafka does, but without the overhead of managing the Kafka instances themselves. AWS Kinesis allows streams to be replayed, restored, and sent to an AWS Fargate cluster (e.g. real-time processing of the data) or data to be stored securely in storage systems such as AWS Simple Storage Service (S3), as shown in **??**. One important aspect is that the data source can also come from a WebSocket stream, which will be addressed in the following section. However, the decision for Kinesis was relatively easy as the reasoning was similar to that between Kubernetes and AWS Fargate. The decision was also supported by the experience and opinion of the cloud experts at Nuvibit.

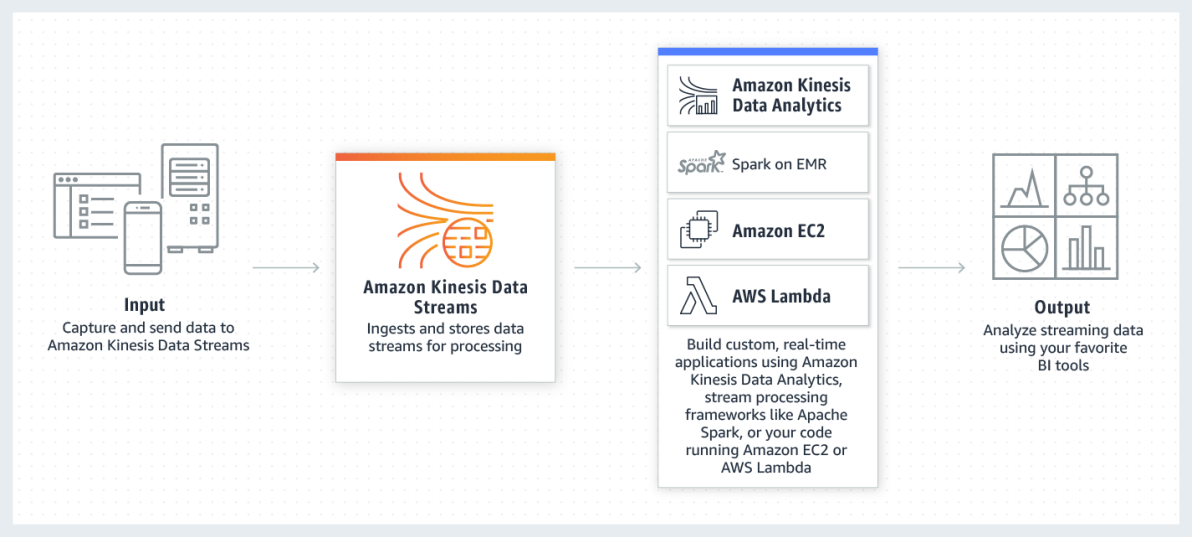

Fig. D.2: Example use-case architecture of AWS Kinesis (**amazon_web_services_inc_amazon_nodate**).

MQTT and WebSocket

As previously stated, a decision had to be made regarding the technology to transfer data from a physical device, such as IDUN's EEG sensor hardware, to the cloud. Previous engineers at IDUN used MQTT to stream data from the hardware directly to the web app, as mentioned in **??**. MQTT is not well suited to sending high-frequency data in real-time, such as EEG, and it is geared toward lower-power IoT devices rather than, say, a computer or smartphone to which the IDUN device would be connected. As a result, the decision to use something else had to be considered. Another option is to use WebSocket, which is designed for high-frequency updates that can be updated in real-time in both directions. IDUN would want to use a high-frequency protocol because HTTP can only handle about ten requests per second, whereas WebSocket can handle nearly 4000 requests in the same amount of time. The main reason for this significant difference is that the browser limits the number of concurrent HTTP connections, whereas a WebSocket connection has no limit on the number of messages it can send or receive (**luecke_http_2018**). Sending EEG data at a frequency rate of 250 samples per second from the IDUN hardware and receiving multiple classification responses from the cloud based on the chosen classification necessitates more than ten requests per second.

WebSocket also integrates easily with the earlier technology decision for AWS Kinesis, as Kinesis can subscribe to an API gateway service from AWS that can be used to build Representational State Transfer (REST) APIs. (This is discussed in greater depth in **??**). An

internal group discussion with firmware engineers for IDUN's hardware was organised to validate this decision, as it was too BCI-specific to ask Nuvibit's cloud engineers. The alignment with the firmware roadmap and its interface, which would be necessary for the BLE library that would consume it was key to the success of building an example N/CI at IDUN.

IaC with Terraform

As mentioned in the list of bugs and flaws in **??**, AWS Amplify took over handling IaC via AWS CloudFormation. When creating the new version of the system without Amplify, the author was now free to choose which IaC technology to use. The author could have again used AWS CloudFormation (but without AWS Amplify), or he could have used the more modern AWS Cloud Development Kit (CDK), which provides specific syntax in specific languages for building IaC (**amazon_web_services_inc_aws_nodate**). However, there is a well-known technology in the industry called Terraform—an open source and YAML-based tool from the company HashiCorp—which is similar to AWS CloudFormation's JSON syntax in declaratively describing IT resources in the cloud. The significant difference, however, is that it is cloud agnostic, meaning that Terraform can be used to build and deploy IT resources not only on AWS but also on any cloud provider currently supported by HashiCorp (**hashicorp_browse_nodate**). Nevertheless, Terraform is currently still in beta, which is usually not a very good sign for production readiness; nonetheless, Terraform is used in production by various large tech companies (**stackshare_why_nodate**). Again, Nuvibit's cloud experts were consulted to decide which IaC tool to use, as the author had no experience with AWS' CDK or AWS CloudFormation. As a result of the expert interview with Nuvibit, it was clear that Terraform was the right way forward for IDUN as future multi-cloud setups or other IT resources can also be handled via Terraform (e.g. Auth0 authentication tenant setups, GitHub organisation setups via IaC, and others). In addition, the author already had experience with Terraform, which was another argument for using this.

The decision to use Terraform was also made relatively early in the process and laid the foundation for every infrastructure built after that to be based on Terraform code. Tools such as Infracost and *tfsec* were set up as part of the bootstrapping project stages in order to assure quality in the form of linting over IaC code via *tfsec* in continuous integration and continuous delivery (CI/CD) pipelines or calculating price estimations based on IaC code as soon as writing it in the editor via the Infracost add-on.

Python SDK

The key event labelled 'Python SDK' in **??**occurred quite late in the process but was one of the important events. It describes the realisation of creating a public and private Python library mainly for the Noel persona. The realisation followed internal group discussions with the neuroscience team at IDUN. Before these internal group discussions, the author assumed that a GUI application such as the Console application of IDUN's NIP in combination with an API and a client-side library would suffice for applications created by Evans. However, as mentioned in the last point of the user interviews listed in **??**, there would need to be a way to control the EEG data collected by IDUN's device and synchronise it locally with other data sources, such as heart rate data, without having to send the data to the cloud[2].

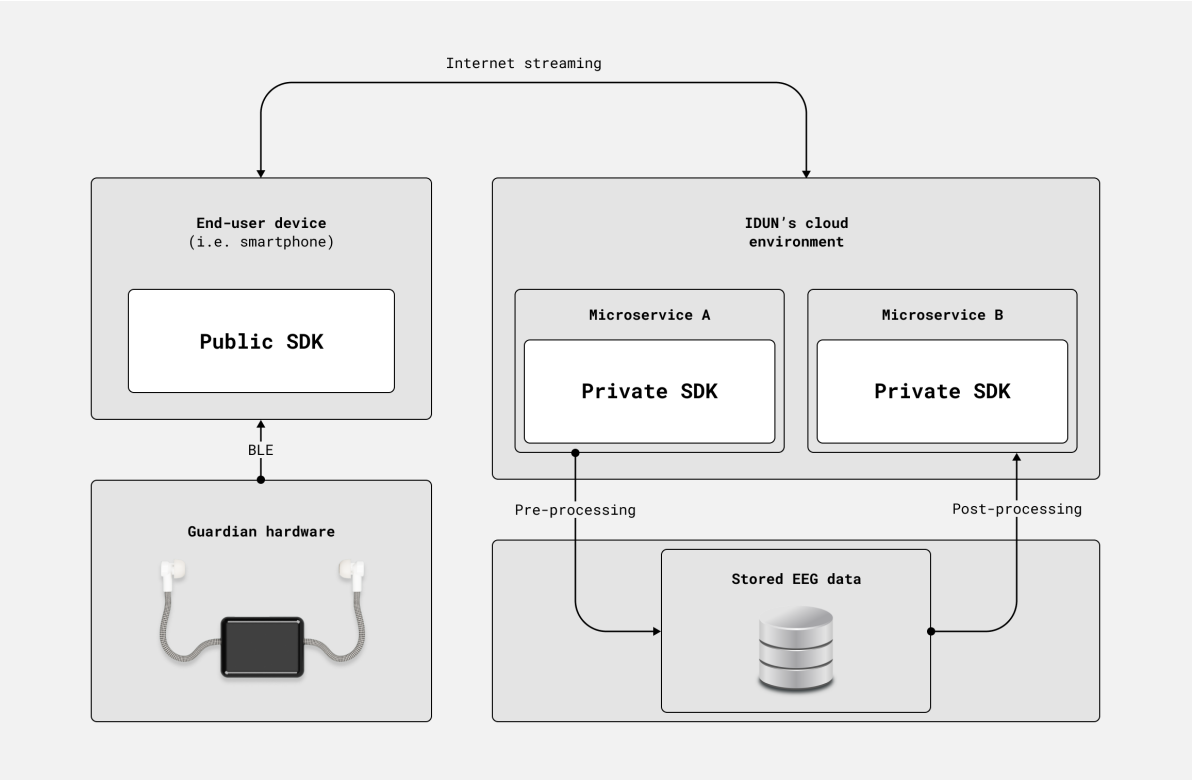

Fig. D.3: Overview of where the public and private SDK can be utilised in multiple microservices on the IDUN cloud.

Noel would use a GUI application such as the idea for the console application of IDUN's NIP. However, they would nonetheless still like to have low-level control of the data to include it in PsychoPy experiment scripts or to connect the data via an LSL stream to a reference

[2]The reason for this is that one cannot synchronise data sufficiently over an internet network due to latency problems or time drifts.

EEG device to compare EEG signals (which is particularly important for IDUN's internal researchers).

For this, they would need a library that allows them to connect to the device without an app, if possible, only for research purposes. After the internal group discussions with the neuroscience department, the author proposed building an SDK that does precisely this. Parts of this SDK can also be made public as part of the software offerings from IDUN, such as connecting to a device for limited research purposes or to visualise plots specifically made for IDUN's EEG data, which was also mentioned as one of the user interview insights in **??**. An internal SDK version of this can be used to include raw data from pre- and post-processing pipelines that are all part of IDUN's provided intellectual property and should not be made public. Nonetheless, this internal SDK can be used in the neuroscience team and in production—for example, Python microservices that process data on the cloud, as depicted in **??**.

The reason why one would want to have that is that if a data engineer creates a new pipeline for processing EEG data that can be used in research or even in production for a microservice, then they would all need to have the exact source of the code. An internal Python package for the SDK is the solution for this. This realisation came fairly late and is, as of this writing, still ongoing.

Appendix E

Design artefacts

In this appendix, design artefacts are displayed that were created during the 10 sprints of the project to create the next version of IDUN's NIP.

Console web app wireframes

The following wireframes are the result of user interviews and internal group discussions. The following figures show the future version of the Console web application for IDUN, which is scheduled for release in early 2023.

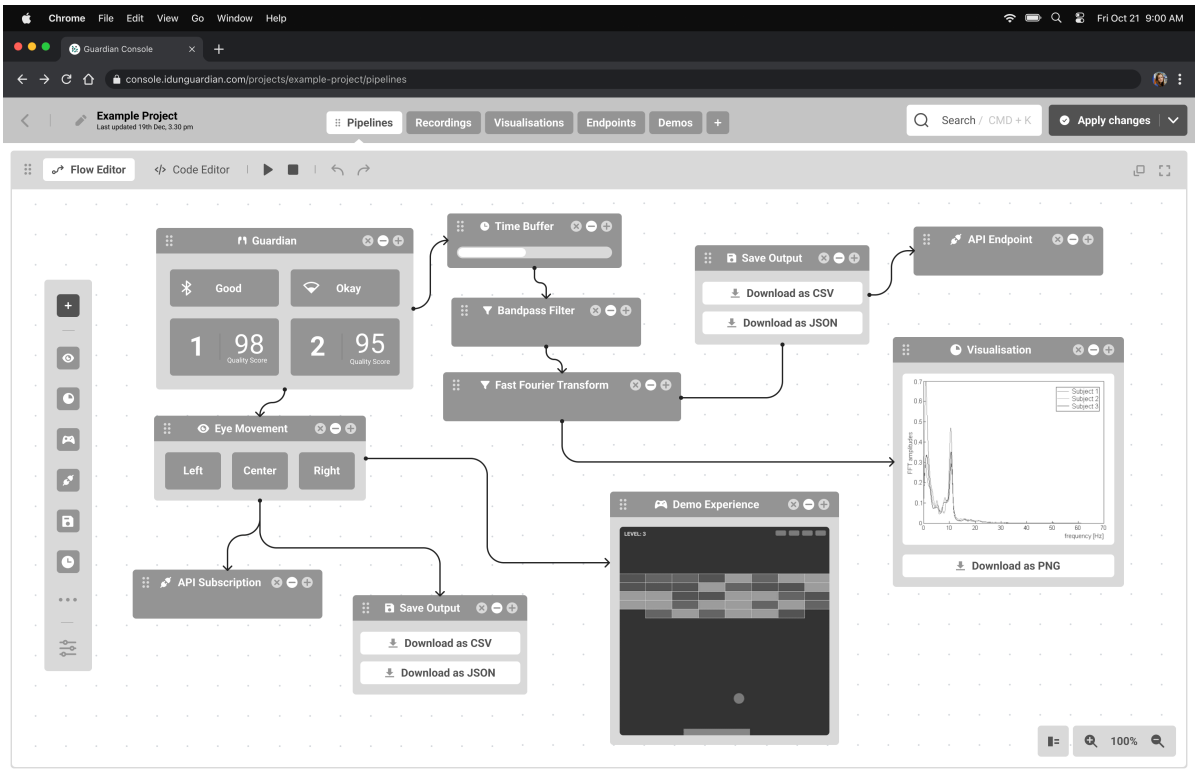

Fig. E.1: Screenshot No. 1 of an exploratory draft for the web version of the Console to be released in early 2023.

?? demonstrates the ability to create deployable neural signal processing pipelines via a drag-and-drop no-code editor.

?? shows how, in addition to the visual no-code editor, advanced users, such as the Evan persona, can also work directly on code generated via the IDUN SDK.

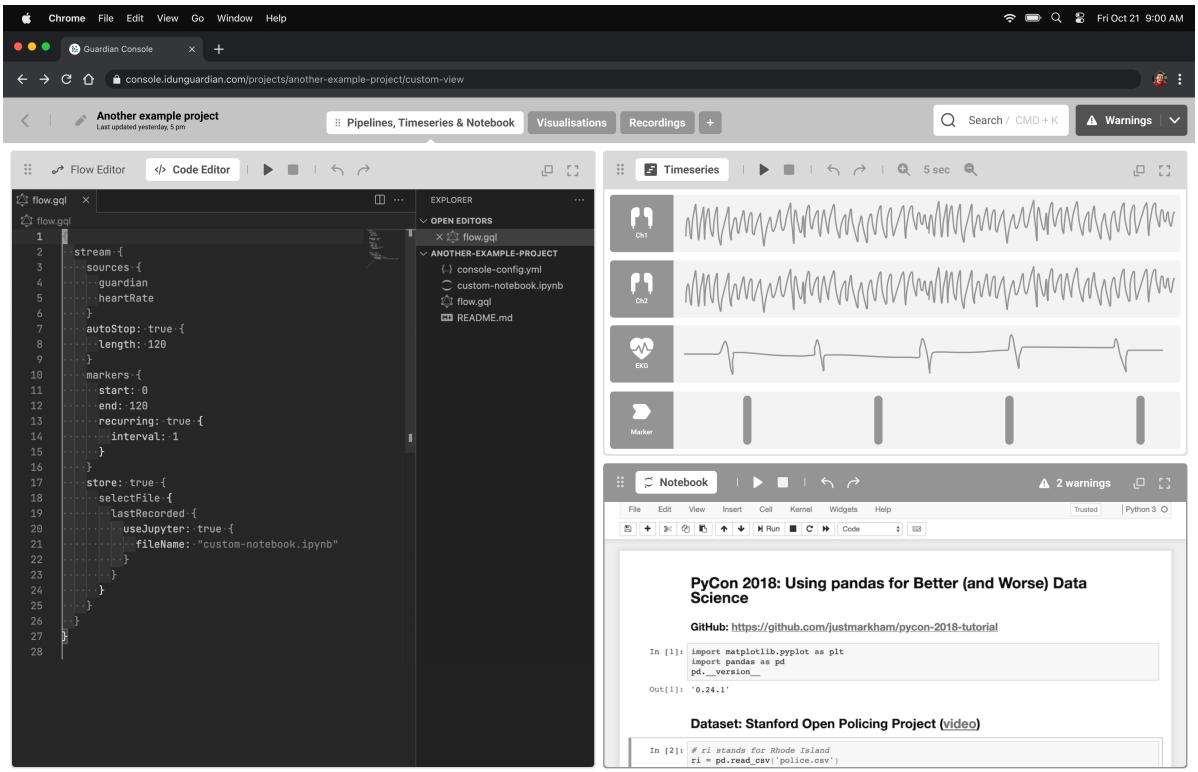

Fig. E.2: Screenshot No. 2 of an exploratory draft for the web version of the Console to be released in early 2023.

Authentication web app wireframes

The following wireframes are the result of internal group discussions, expert interviews and, in particular, the findings of IDUN Technologies' neuroethics advisory board in order to protect user privacy and restrict access to the generated insights and raw EEG data. The following figures show the future version of an authentication web app for IDUN, scheduled for release in early 2023.

?? shows how a slide-over would look in the third-party app if no device from IDUN is connected to, for example, a smartphone. After searching for BLE devices with certain characteristics, the nearest device is displayed with the option to connect to it. After this process, the SDK integrated in the third-party app checks whether this device is registered in the IDUN cloud with this identifier.

?? shows how the process would look if IDUN's device is not yet registered in the cloud, for example, the opt-in for the specific third-party app in this example is not yet granted. Therefore, the user must authenticate with a device password (which they must reset the first time they use it) and then select specific options to which they agree.

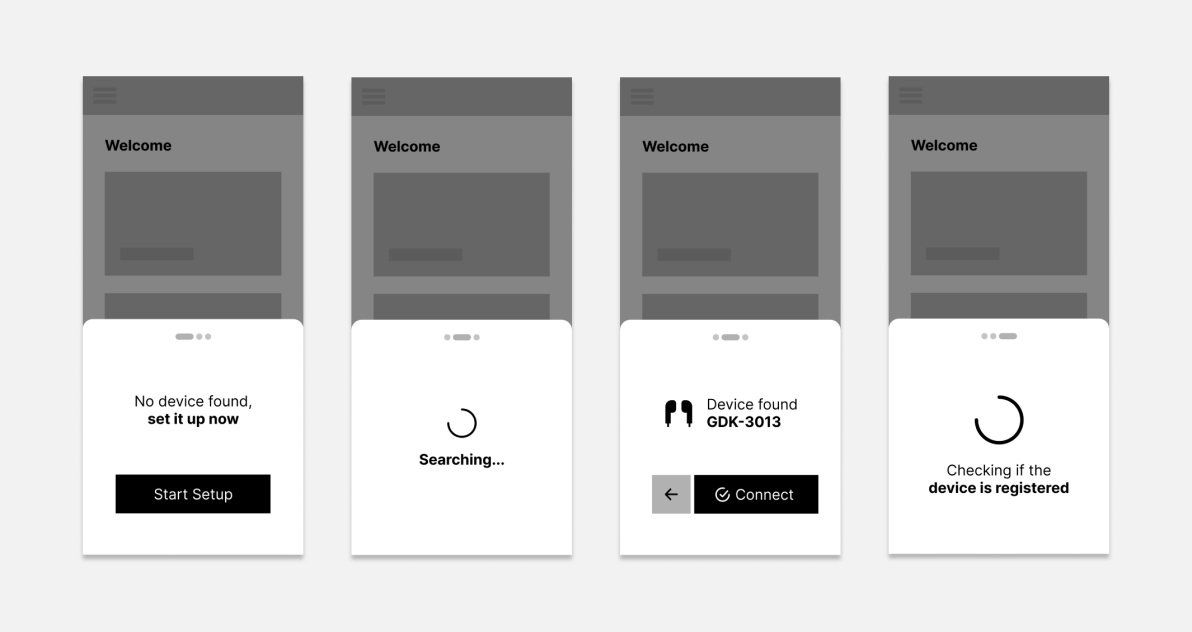

Fig. E.3: Screenshot No. 1 of an exploratory wireframe draft for the version of the authentication web app to be released in early 2023.

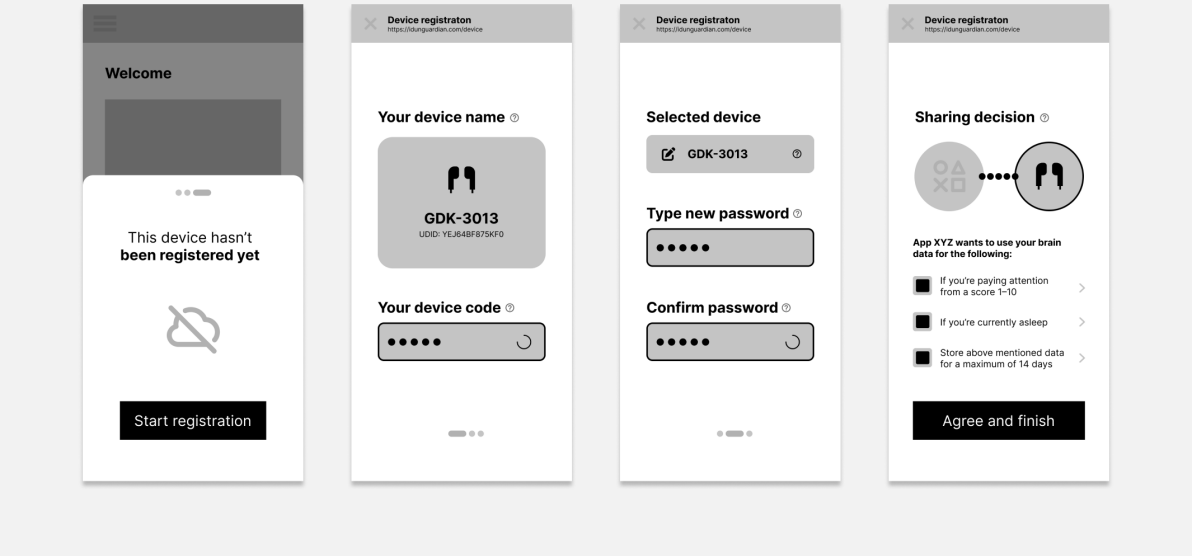

Fig. E.4: Screenshot No. 2 of an exploratory wireframe draft for the version of the authentication web app to be released in early 2023.

Appendix F
Attached repositories

This appendix lists and describes the attached repositories, that is, the code produced during the 10 sprints to create the new version of IDUN's NIP and the source code of the written part of this project.

idn-guardian-cloud

This repository is a monorepo that contains the infrastructure for the cloud for IDUNs N/CI at the time of writing.

idn-guardian-console

This repository contains the source code for the Console web app for IDUNs N/CI at the time of creation. It also contains an example of exporting an iOS and Android app.

idn-guardian-sdk

This repository is a monorepo containing the source code of the Guardian SDK, IDUN's public SDK. It contains all files at the time of writing and is not complete as most of the business logic, for example, to connect to the BLE device, is handled in the Console web app repository.

idn-internal-sdk

This is a shallow Python repository that will host the internal SDK for IDUN's backend microservices.

spatial-place

This is the actual repository where the work was written in LaTeX. It contains all source files, READMEs and other information.

Appendix G
Other documents

This appendix lists further documents and artefacts attached to this thesis, such as more detailed documentation of the topics mentioned in the thesis.

Customer personas documentation

Notion PDF exports of all persona pages can be found in the files inside the appendices directory in the personas subdirectory.

Web-first approach proposal

Notion PDF export to be found in the file *web-first_approach_proposal.pdf* in the appendices directory in the web-first approach proposal subdirectory.

Other documentation

Other documentation can be found as Notion PDF exports in the appendices directory in the other docs subdirectory. An example is the documentation on setting up the AWS organisation or the documentation on the data model.