



**Isaac Tomé  
dos Anjos**

**VIRHUS: uma plataforma computacional para a  
simulação de sinais fisiológicos de humanos  
virtuais**

**VIRHUS: a computational platform for the  
simulation of physiological signals for VIRtual  
HUmanS.**





**Isaac Tomé  
dos Anjos**

**VIRHUS: uma plataforma computacional para a  
simulação de sinais fisiológicos de humanos  
virtuais**

**VIRHUS: a computational platform for the  
simulation of physiological signals for VIRtual  
HUmanS.**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor Ilídio Castro Oliveira, Professor auxiliar do da Universidade de Aveiro, e da Doutora Susana Manuela Martinho dos Santos Baía, investigadora do Instituto de Engenharia Eletrónica e Informática de Aveiro, da Universidade de Aveiro.



**o júri / the jury**

presidente / president

Professor Doutor António Joaquim da Silva Teixeira  
Professor Associado C/ Agregação, Universidade de Aveiro

vogais / examiners committee

Doutor Francisco Xavier Fonseca  
Investigador Doutorado, Instituto Politécnico do Porto

Professor Doutor Ilídio Fernando de Castro Oliveira  
Professor Auxiliar, Universidade de Aveiro



**agradecimientos /  
acknowledgements**

I want to acknowledge the support provided by my family, friends, and coworkers. I would also like to show my deep appreciation to my supervisors who arranged the necessary dataset and helped me finalize my project.





## Palavras Chave

Réplicas digitais; aprendizagem automática; dispositivos médicos em software; biossinais; computação afetiva; simulação.

## Resumo

A capacidade de aceder aos biossinais de participantes para actividades de investigação é limitada, especialmente em contextos informais, tais como projectos académicos. Estas limitações podem ser parcialmente ultrapassadas através da utilização de software para simular dados fisiológicos suficientemente fidedignos. Neste trabalho, propomos e desenvolvemos uma plataforma computacional para simular (sinais biológicos de ) seres humanos virtuais como um serviço. O sistema adopta o conceito de réplica digital ("digital twin") para estruturar os processos de simulação. Neste caso, o sistema não está a monitorar participantes reais, mas utiliza sinais pré-gravados como entradas para autocodificadores que geram um sinal sintético realista para um ser humano virtual, ou seja, uma réplica digital. Os sinais pré-gravados utilizados foram o electrocardiograma, a actividade electrodérmica e os sinais electromiográficos que foram marcados com a emoção em progresso. O sistema, VIRHUS, fornece um ambiente web interactivo para criar os seres humanos virtuais necessários e gerir os processos de simulação. Um backend escalável cuida da geração assíncrona de sinais, que podem ser transmitidos para pontos de acesso programático (e consumidos por aplicações externas) ou exportados como ficheiros por conveniência. Como prova de conceito, os dados "humanos virtuais" podem ser parametrizados para incluir traços emocionais nos biossinais (feliz, triste,...), gerando variações significativas nos dados para os programadores de aplicações.



**Keywords**

Digital twins; machine learning; software as a medical device; biosignals; affective computing; simulation.

**Abstract**

The ability to access bio-signals of participants for research activity is limited specially for informal settings like academic projects. These limitations can be in part overcome it by using software to simulated good enough physiological data. In this work we propose and develop a computational platform to simulate (biological signals of ) virtual humans as a service. The system adopts the concept of digital twin to structure the simulation processes. In this case, the system is not sensing real participants, rather uses pre-recorded signals as inputs to auto-encoders that generate realistic synthetic signal for a virtual human, i.e., a digital twin. The pre-recorded signals used were the electrocardiogram, electrodermal activity and electromyography signals which were labeled with the ongoing emotion. The system, VIRHUS, offers an interactive web environment to create the required virtual humans and manage the simulation processes. A scalable backend takes care of the asynchronous generation of signals, that can be streamed to endpoints (and consumed by external applications) or exported as files, for convenience. As a proof of concept, the “virtual human” data can be parameterized to include emotional traits in the bio-signals (happy, sad,...), generating meaningful variations in data for applications developers.



# Contents

<b>Contents</b>	<b>i</b>
<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vii</b>
<b>Listings</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Objectives . . . . .	2
<b>2 State of the Art</b>	<b>3</b>
2.1 Digital Twins . . . . .	3
2.1.1 Digital Twins Definition . . . . .	3
2.1.2 Digital Twin Characteristics . . . . .	4
2.1.3 Digital Twins Application . . . . .	4
2.2 Digital Twins in Healthcare . . . . .	5
2.2.1 Overview . . . . .	5
2.2.2 Selected case studies . . . . .	6
2.3 AutoGenerative algorithms and models . . . . .	12
2.3.1 Overview . . . . .	12
2.3.2 Wavelet-based model . . . . .	15
2.3.3 Generating electrocardiogram signals by deep learning . . . . .	15
2.4 Discussion . . . . .	17
<b>3 Use cases in computational physiology research</b>	<b>19</b>
3.1 Dissertation Application . . . . .	19
3.2 Personas . . . . .	20
3.2.1 Persona A . . . . .	20
3.2.2 Persona B . . . . .	20

3.2.3	Persona C . . . . .	21
3.3	Use cases of the system . . . . .	21
3.4	Quality attributes/non-functional requirements . . . . .	24
3.5	Scope limits and exclusions . . . . .	25
<b>4</b>	<b>System Architecture</b>	<b>27</b>
4.1	Architecture Overview . . . . .	27
4.2	Interactions between modules . . . . .	29
4.3	Domain Concepts . . . . .	29
4.4	Testing Requirements . . . . .	31
<b>5</b>	<b>System Implementation</b>	<b>33</b>
5.1	System deployment and implementation technologies . . . . .	33
5.2	Dataset Used . . . . .	34
5.3	Web Application . . . . .	34
5.4	Tasks Scheduler . . . . .	38
5.4.1	Create a Task . . . . .	39
5.4.2	Export simulation to file . . . . .	41
5.5	Data Acquisition . . . . .	41
5.6	Biosignals generation component . . . . .	42
5.6.1	Training . . . . .	42
5.6.2	Supervisor . . . . .	42
5.6.3	Node . . . . .	43
5.7	Web Server . . . . .	44
5.8	Data Integration . . . . .	48
5.9	Data Protection and Security . . . . .	49
<b>6</b>	<b>Results and validation</b>	<b>51</b>
6.1	Signals generation validation . . . . .	51
6.1.1	Are the synthetic signals (biologically) realistic? . . . . .	51
6.1.2	Does the signal degrade over time? . . . . .	54
6.2	Initial Prototype and Experimental use . . . . .	58
6.2.1	VIRHUS Integration . . . . .	58
6.2.2	Personalizing new emotions . . . . .	61
6.3	Usability Evaluation . . . . .	63
6.3.1	Sample . . . . .	64
6.3.2	Method . . . . .	64
6.3.3	Questionnaire Results . . . . .	65

<b>7</b>	<b>Conclusions</b>	<b>73</b>
7.1	Work Developed . . . . .	73
7.2	Future Work . . . . .	74
	<b>References</b>	<b>75</b>
	<b>Appendix A Usability Tasks</b>	<b>77</b>
	<b>Appendix B Observer Spreadsheet</b>	<b>79</b>
	<b>Appendix C SUS Questionnaire</b>	<b>81</b>
	<b>Appendix D QUIS Questionnaire</b>	<b>83</b>





# List of Figures

2.1	Digital Twin structure . . . . .	5
2.2	Digital Twin and SmartFit . . . . .	10
2.3	Autoencoders . . . . .	13
2.4	Generative Adversary Network . . . . .	14
2.5	Wasserstein GAN . . . . .	14
2.6	Deeplearning Workflow . . . . .	16
3.1	Use case Diagram . . . . .	22
4.1	Architecture Design . . . . .	27
4.2	Sequential Diagram of the dataflow . . . . .	29
4.3	Database Table Design . . . . .	30
4.4	Domain concept Diagram . . . . .	30
5.1	Technology Stack . . . . .	34
5.2	Template page . . . . .	35
5.3	IDE Tool Page . . . . .	36
5.4	Add Dialog . . . . .	36
5.5	Virtual Human Player . . . . .	36
5.6	Virtual Human Charts . . . . .	38
5.7	Multiple Virtual Humans . . . . .	38
5.8	Task Tool Page . . . . .	39
5.9	Task Creation . . . . .	40
5.10	Request Column . . . . .	40
5.11	Request Column . . . . .	41
5.12	Request Column . . . . .	42
6.1	MAE Generic model Results . . . . .	53
6.2	MSE Generic model Results . . . . .	53
6.3	MAE Individual model Results . . . . .	54
6.4	MSE Individual model Results . . . . .	54

6.5	MAE Individual model Results over time . . . . .	56
6.6	MSE Individual model Results over time . . . . .	56
6.7	MAE Generic model Results over time . . . . .	56
6.8	MSE Generic model Results over time . . . . .	57
6.9	External Project . . . . .	58
6.10	Virtual Human creation . . . . .	59
6.11	Virtual Human dropdown . . . . .	59
6.12	IDE Tool Virtual Human . . . . .	60
6.13	External Project running . . . . .	61
6.14	Uploading a CSV . . . . .	62
6.15	Training Signal . . . . .	63
6.16	New Task . . . . .	63
6.17	Emotions with no indicator . . . . .	66
6.18	Signals in text form . . . . .	66
6.19	Boxes clipping out . . . . .	67
1	Usability Tasks . . . . .	77
2	Observer Spreadsheet . . . . .	79
3	SUS Questionnaire . . . . .	81
4	QUIS Questionnaire . . . . .	83

# List of Tables

3.1	User Case and Description . . . . .	23
3.2	User Case and Description . . . . .	24
6.1	Signal Realism scores of the Generic model using MAE metric . . . . .	52
6.2	Signal Realism scores of the Generic model using MSE metric . . . . .	52
6.3	Signal Realism scores of the Individual model using MAE metric . . . . .	53
6.4	Signal Realism scores of the Individual model using MSE metric . . . . .	54
6.5	Signal Degradation scores of the Individual model using MAE metric . . . . .	55
6.6	Signal Degradation scores of the Individual model using MSE metric . . . . .	55
6.7	Signal Degradation scores of the Generic model using MAE metric . . . . .	57
6.8	Signal Degradation scores of the Generic model using MSE metric . . . . .	57
6.9	Usability test results . . . . .	65
6.10	Questionnaire for User Interface Satisfaction Results 1 . . . . .	69
6.11	Questionnaire for User Interface Satisfaction Results 2 . . . . .	70
6.12	System Usability Scale Results . . . . .	71



# Listings

5.1	Server request data structure . . . . .	43
5.2	Node response data structure . . . . .	44
5.3	Request data structure . . . . .	46
5.4	Response data structure . . . . .	46
5.5	Virtual Human Request data structure . . . . .	46
5.6	Action data structure . . . . .	47
5.7	Task request data structure . . . . .	47
5.8	IDE Tool data structure block . . . . .	48
5.9	Task data structure block . . . . .	49



# Introduction

Nowadays, the development of alpha monetary systems for bio informatics research requires the use of biomedical data, a specific type of synthetic data, authentic and realistic, where the idea evolves a specific pattern or calculations. This material sometimes is not available or comes with a price upfront, not to mention the General Data Protection Regulation (GDPR) which also slows down the data gathering process in most projects.

Health experts, on the other hand, tend to treat patients with high-tech systems and use historical data for their procedures. These systems are implemented with sophisticated measures to further identify the correct treatment of a patient. These systems are often described as biological systems based on machine learning systems that classifies data without a concrete definition of the inherent model. Biological systems whereas deep learning are algorithms that find dependencies and relations between attributes in big data, but they are not available in a first analysis. The ability to collect and store data of patients became possible, and the state of the art deep learning has begun its shift towards system health monitoring (SHM)[1] and prognostic health management.

In consequence, the need for personal data is required for training [2]. For a correct representation of the population, there is more noisy data than control data. Which may compromise the representation of several machine learning algorithms. Though obtaining data can be quite easy, personal data is not [3]. According to the Paris Law [4], the life cycle of participants considering personal data is usually around 5-8, within the range of 1-2 being the critical regime of the subject.

Following up on the problems detailed previously, this dissertation introduces an emerging technology called Digital Twins, providing benefits and related work in medical care, and proposes a solution to complement research and development projects.

## 1.1 MOTIVATION

The motivation behind this dissertation is to explore the concept of Digital Twins to structure a synthetic data generator, that can be comparable to real human data, to support

developers working in new applications. A system that can accept its output as input, users' datasets, and much more to create a flexible and accommodating feel. The idea behind the system is Digital Twins, and the goal is to put into practice and deploy a solution that properly represents the concept as a whole with tools that could complement R&D activities and small informatics medical based projects.

## 1.2 OBJECTIVES

After introducing Digital Twins and the fundamentals, the goals and contributions of the dissertation are:

- Develop methods to simulate realistic-enough biological signals of “virtual” humans
- Influence the signals generation by activating the expression of different emotions for certain periods.
- Propose and develop an integrated computational platform to manage the simulation of physiological data for “virtual humans”, leveraging the concept of "Digital Twin"
- Demonstrate that the simulations can be consumed by other software agents/applications towards a mHealth innovation “playground”.



## State of the Art

The scope of this dissertation intersects several areas such as data analysis of biological sources, an auto-generative process capable of generative artificial signals based on a pre-trained model, architecture system design to stream data to multiple users, and endpoint designs to complement multiple projects over the OSI protocols.

### 2.1 DIGITAL TWINS

The label Digital Twins is a buzzword that has recently appeared in monitoring systems, but the concept of twin replication has already existed priorly. In 1970, NASA created a system capable of mirroring the state of physically unreachable targets (spacecraft, satellites, etc.) and encountering the right solutions to unexpected problems in this system. A famous example of the NASA mirrored system was an incident during the Apollo 13 mission. During the trip to the moon, the aircraft suffered a Main B Bus undervolted, and an explosion of one of the oxygen tanks. With the already set up aircraft in Kennedy Space Center, Houston, engineers managed to find a solution and rescue the astronauts from space in time.

#### 2.1.1 Digital Twins Definition

Digital Twins has existed for decades, but the definition has constantly changed among the research community, and deviations began to appear generalizing Digital Twins itself. This section will define Digital twins, deviations, and the architectural scope of a Digital Twin.

Digital Twins is a digital copy of a physical entity, capable of tracking, predicting, and adapting to the current state while saving previous transitions of events [5]. This process is done by continuously aggregating data coming from the real world, followed by analyzing in real-time and forecasting important events, to finally provide a computerized model of the physical twin. These models provide a means of maintenance, allowing experts to adjust and correct when needed without physically changing the real twin [5].

### 2.1.2 Digital Twin Characteristics

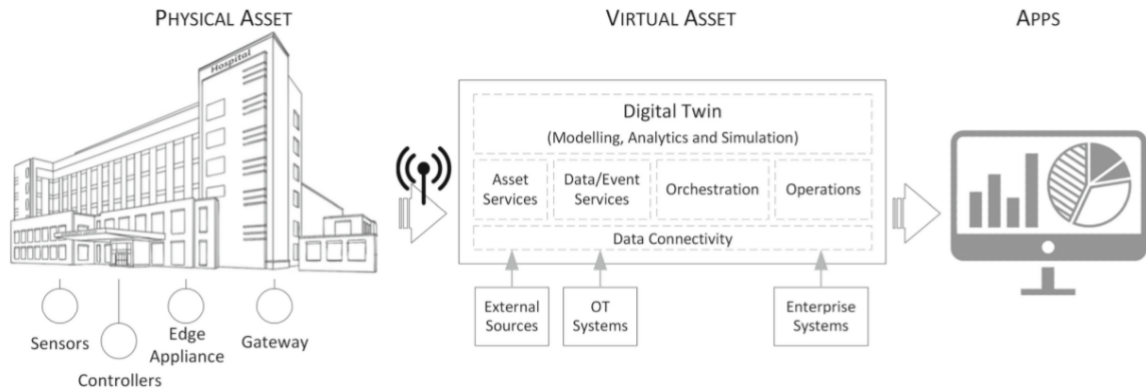
Digital twins heavily relies on historical data and deep learning methods to properly adjust and predict. The twin is constructed in way where the input of sensors are collected from a real-world entity. This allows the system to generate the physical entity in real time, offering insights of performance and potential problems. Essentially, Digital Twins are composed of multiple components [6]; input which represents the endless stream of raw data coming from the physical twin; Data which can be in any interpreted format, even binary code, sent by sensorial devices. The data is collected and stored in the historical component, often referred to as the persistency of the system, referred to as databases. The behavior, on the other hand, relies on the logical specifications when evaluating incoming data, for example, if the sensors say the physical is turned off, the virtual twin will evaluate the turned-off data and transit to the same state. The state diagram of the physical twin can often be redolent, in which machine learning techniques can create a model from the persistence component capable of identifying the current state.

In generative methods, a deviation emerged called Deep Digital Twins [3], which uses sensor data collections to create a probabilistic model of the physical twin. The sensor measurements and control inputs of the system are treated as random variables, creating probabilistic distributions. An advantage of using generative adversarial networks oriented would be from the self-adaptation provided, given how the Physical twin changes state through time, GAN networks can as well [3]. A similar deviation called Digital Shadow [7] was designed to be physically linked to the opposite twin through the Internet of Things paradigm, technically speaking, the digital twin continuously receives data from the sensors of the physical twin and adapts to the current behavior/state of the real twin. Its logical concept provides some insight into where the data should come from, where should it go, and how it should be processed. The following sections illustrates previous studies within the Digital Twin domain, from generic studies on the definition of Digital twins to studies that have applied the concept in healthcare. The purpose is to shed light on what exists and where this paper lands among them.

### 2.1.3 Digital Twins Application

Digital Twins heavily relies on data collection in many fields of study or operations that involve saving data[8]. Smart manufacturing for example uses the Digital Twin concept for data collecting and management of various resources throughout a product life cycle. By doing so, it proceeds by tracking the product quality, production process, and energy efficiency to later simulate and optimize providing predictive suggestions for debugging[8], and real-time simulations of the equipment-based and data-driven models. Another example is the automotive industry, where the safety of both consumers and passengers is the number one priority. With the help of Digital Twins, it's possible to perform collision testing, log anomalies, and evaluate the material used in each assignment. In post purchases, it is also possible to collect performance evaluations and log in real-time vehicles to better optimize the structure and performance of future models.

System implementation relies on a generic design of required components and an understanding of how these components communicate with one another. In a recent study done by Angelo Croatti[9], the paper is about the integration of digital twins and Multi-Agent Systems (MAS) technologies in healthcare. The authors of the paper drew out how digital twins would be used in healthcare.



**Figure 2.1:** Digital Twin structure from [9]

In theory, Digital Twins were this black box or grey area between the physical object and the user interface. They would also have software agents observe and act upon incoming reads of the real world. By doing so, these agents could create what they called mirror worlds and forecast actions and events within. A mirror world is a digital plane entirely built by software agents with the capacity of tracking physical objects like chairs, tables, walls, etc, and placing them in the mirror world accordingly.

## 2.2 DIGITAL TWINS IN HEALTHCARE

### 2.2.1 Overview

In the healthcare industry, initial implementations aimed at the maintenance of medical devices [10] performing optimizations in terms of patient diagnostics, energy consumption, and hospital lifecycle [10]. Digital Twins concept, on the other hand, is being used to visualize and predict the patient's health, allowing experts to predict the occurrences of diseases and the proper treatment considering the history of the patient, space, and activity [11]. This is a game-changer, especially considering the patient's history records. For example, a study was conducted using a fitness application. The goal was to further push the patient's well-being through suggestions, based on the exercises, the diet, and hours of sleep. The system could adapt and predict optimal treatments for the patient to do [10]. The data was collected through wearable sensors and sent to the fitness application SmartFit. The data captured were dynamically changed thus adapting the Digital Twins' status, so potential events are triggered, and predictions and suggestions can be made. The authors of this paper were able to develop an extension to SmartFit where they exploited the data being fed by the

wearable sensors and data collected by the personal trainers. Important note to include, the dataset collected had missing and/or noisy data, to cope with such problems they used KNN imputation strategy to ensure robustness and maximize the data informativeness.

Meanwhile, health experts heavily rely on intensive diagnostic practices to properly identify the problem at hand. To evaluate the diagnosis procedure in traditional medical practices and how it impacts the patient's quality of life. By doing so, they plan to introduce the emerging technology called Digital Twins in healthcare to help doctors and medical advisors prescribe the most correct treatment/prescription, a practice they call precision medicine. To further support their goal, they developed a model based on their initial ideas and adaptable smart systems in healthcare, systems where critical information is given on each available treatment or drug assisting medical advisors to treat their patients. They also drew out the internal structures of a Digital Twin to achieve their goals. On the developer end of systems, they require fresh sets of datasets which they cannot afford to test their systems for mistakes. Through a strategy called Generative Adversarial Networks which consists of two learning networks, the generative and discriminator. These two networks play a game of fooling the critic, the generative will play the role of "Artist" where they will attempt to draw realistic images and the discriminator's job is to tell the fake images from the real images. In each iteration, the knowledge in both starts to grow until the discriminator fails to tell the fake images from the real ones.

### **2.2.2 Selected case studies**

#### *Predict Individual Stress Levels in Extreme Environments*

The paper[12] illustrates a traditional style of collecting data, through sensorial devices attached to the participants. Before equipping the devices, they must undergo a calibration process that will increase the output accuracy and reduce the noise within the segments. In the study, the researchers concluded a connection between the human body's temperature and the level of stress, thus requiring temperature and humidity sensors to record the participant's body temperature and an ECG sensor over the heart to track the cardiac distress in cold temperatures, stimulating levels of stress. In need, developed a modular, washable smart textile undergarment equipped with integrated sensors and a central processing and communication unit. With the smart textile, they facilitated the sensor application on the human skin at medical valid positions and allowed a seamless health monitoring experience. Upon execution, the data acquisition environment allowed them to label generated data snapshots. Within the environment, the scenario, the physiological fingerprint as well as sensor value frequency and variance can be modified and were labeled 1027 data snapshots. Overall their results were interesting but required participants, equipment, and a team of researchers to simply create a dataset for stress predictions.

Stress is a health condition relating to physiological pain and if not treated properly, long-term damages[13] can occur to the individual. Though there is research backing long-term effects, identifying stress in real-time and the short-term effects were still yet to be discovered. With the technology that exists today, a team of chemists, medical practitioners, textile

designers, and engineers developed end-to-end remote environments to recreate extreme and dangerous scenarios. The users that would interact in these environments were given smart textile wearables that measure the vital quantities of body temperature, heart rate, skin temperature on their wrists and ankles, and breathing frequency. The figure below shows the textile wearables and the devices used to measure and send in real-time the users' biosignals to the system. The system would then collect all the raw output, filter out the noise and build a snapshot of the scenario that was carried out. Medical advisors in this particular study, had groups of eight undergo scenarios and within each one, a stress level estimation was determined. Before deploying the users into predefined scenarios, experts had to determine a scale of stress for each scenario.

The stress level scale was based on the behavior and appearance of the user during the scenario. Experts defined five stress levels: minimum, low, medium, high and maximum. Each contains a short description of what to expect at each level. In minimum stress, the user would express kindness and in a mindful state, sensors would indicate relaxation and smooth body motions. At low-stress levels, the users would express positive stress, and eustress, which correlates to executing a familiar or trained task. Medium stress levels would require the user to be at their highest state of attention and reach the maximum amount of tasks.

High-stress levels would require the user to be non-responsive following up with two physiological reactions behind hypothermia or overheating. Humans at this level are considered mentally unstable. Finally, the maximum level of stress included reactions such as they fight for survival, incapacity to react, and even coma. These levels were used to categorize the data snapshots.

In consequence, the authors avoided executing the scenarios with real-life patients and decided to mock their data snapshots, creating 1027 data snapshots. They were also not entirely clear on how the data was generated or what algorithms were used. Upon mocking and labeling their snapshots, the authors developed a user interface that displayed the ongoing scenarios and the participants involved. Each user is classified by the system by their current stress level and shows the output of each sensor when a user is selected.

In this paper, the authors used the Digital Twins concept to create an evaluation environment and a user interface to help medical experts understand how and when their patients underwent the effect of stress and at what level of stress they were. Despite explaining a traditional case scenario of data acquisition, the authors were able to develop and test a system capable of displaying the patient's stress level without involving undergoing real scenarios expressing severe stress levels.

#### *Generating electrocardiogram signals*

The goal of the following study[14] is to generate artificial electrocardiogram signals using deep learning which is associated with the transformation and extraction of features that attempt to create generative models for ECGs based on a pre-existing dataset of ECG segments. The proposed models were capable of generating ECGs containing three different heartbeat types: normal beat left bundle branch block beat and right bundle branch block beat. The

authors concluded that the artificial signals generated by the models were more realistic since deep artificial neural networks can discover intricate structures and characteristics of real ECGs instead of manually setting specific parameters for synthesis, but there were not satisfied which the overall results thus listing potential reasons to why their models performing poorly.

- A small number of samples within the training set, only containing 5000 samples of each category where the expected number of samples for training data is usually in the tens of thousands or hundreds of thousands.
- The length of the training samples was too short which would negatively impact simulations that would expand the training length. The maximum length used was 4 seconds, far less than the normal length of the normal training length.

Despite skipping the traditional approach of data acquisition, the researchers in this study were vastly limited to the quality of the dataset which was provided which negatively impacted their results.

#### *Human Digital Twin for Fitness Management*

Fitness and weight-loss programs are designed to help the average person at losing weight. These programs were built off of an endless amount of research and data collected over time. The problem with these programs resides in the static nature, the inability to adapt accordingly to the individual[15]. A recent study conducted by BARRICELLI[10], talks about the implementation of Digital Twins using SmartFit, a health monitoring system, to track the physical twin. The reason this paper was specifically picked was due to the idea of taking a preexisting system and incorporating Digital Twins, regardless of how the technology is used in that very system.

A new approach has recently appeared where data acquisition is possible and accessible through endpoint communications. The goal of the study[10] was to use an existing system called SmartFit, a software framework for supporting trainers and coaches in monitoring and managing athletes' fitness activity and results. SmartFit was designed to continuously capture measurements through IoT sensors embedded in wearable devices for continuous tracking. The data was initially treated as dynamic data describing the athlete's current state. The measurements collected which describe the participant's health are often incomplete or entirely forgotten, creating gaps within the dataset. The authors of this study decided to fill the gaps with KNN imputation strategy, which worked successfully if the feature vectors had the same type and took the same range of values, but since the health features whose types were often different from one another, and the range of variation were very different, they considered estimating the missing values by searching for the nearest KNNsample.

Unlike the traditional and historical approach to acquiring data, this method provides an entirely different perspective of how data can be collected and transform to help achieve the study's goals. For example, in the previous study, predicting stress levels, researchers could have used SmartFit as their source of data, providing their participants which a similar IoT device and extracting the live feed from the system, thus reducing the cost they used for

developing the devices and the system in which analyzed and aggregated the data to be later used.

SmartFit is a health monitoring system designed to help trainers track and monitor a team of athletes, capturing the athletes' performance and behavior over time. These measurements are recorded by wearable sensor devices and logging food consumption. The app is capable of aggregating and displaying the data for the trainers to identify any critical behaviors during the training session. SmartFit is also able to alert trainers using created rules aim to avoid injuries and bad habits during each session. The developers of SmartFit have exposed an endpoint to the data collected, allowing outside developers to extract and perform data analysis of the athletes efficiently. Using these endpoints, the authors were able to extend the SmartFit application with artificial intelligence that would help improve the athletes' performance by suggesting changes to their daily routine. Through the API, the authors were able to create DTs of each athlete by updating the digital twin whenever the physical twin changes state. The system would also allow trainers to conduct "what if" scenarios on their digital athletes, forecasting a state of their interest or forecasting the steps to achieve a state of interest. They initiate the research by creating a dataset consisting of snapshots of each athlete, each snapshot corresponding to a training session, and each session would be labeled by the respective trainer. The dataset would then be used to train four different multiclass classifiers and four different parameter settings.

The authors encounter several problems throughout the research, mainly involving the data collected from SmartFit. The athletes' sessions were manually recorded and sometimes forgotten throughout the session creating holes within the data. A workaround that was used during the project was to apply KNN imputation[16] to the dataset. However, the types of features used were quite different from one another and opted by searching for the k-impute nearest sample instead of the nearest feature.

The results of the paper showed that only the manual parameter( $W_{same}$ ) had some influence on the results and that KNN-based classifiers achieved the best results when  $W_{same} = 0.75$ .

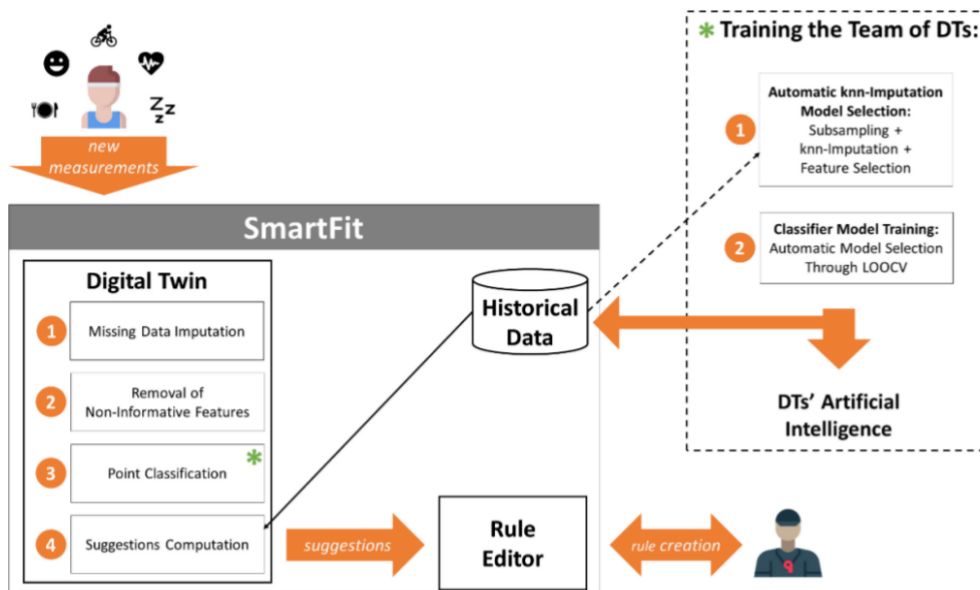


Figure 2.2: Digital Twin and SmartFit from [10]

### Deep digital twins for detection, diagnostics, and prognostics

The following paper[3], conducted by Wihan Booyse, addresses the Deep Digital Twins and comparisons with the Digital Twins concept and how it can be used in prognostics and health monitoring. The authors chose PHM due to its probabilistic nature and with DDT, experts could interrogate the possibilities and states of the twin given the historical health data.

Deep Digital Twins inherits the generic definition of Digital Twins which is a digital representation of a real-world object but instead uses deep generative models. These models undergo an intensive training phase, making it possible to simulate system responses by changing operational settings as well as comparing predicted behavior to the physical twin. Though the goal of Digital Twins is to simulate the entire system, DDT primarily focuses on the data collected from the sensors, creating a probabilistic model[3] of the system. All sensor measurements correspond to random variables of  $x$  and the inputs of the system by  $y$ . The variable of  $z$  represents the true hidden latent variables which is the approximation of a perfect distribution, that describes the behavior of the system. DDT relies on a clean and healthy dataset, especially for PHM, that is important: to capture the entire distribution of the training set; a large amount of noise or nonsensical observations; and to track the deterioration of a sensor which could impact the conditions of the DDT. Two types of deep generative models suit the implementation of DDT.

- Explicit probabilistic models.
- Implicit probabilistic models.



The difference between the two resides in the mathematical estimation of the perfect distribution. Explicit models can provide an estimation of a distribution that describes the system whereas Implicit models will only provide a ratio between the generated and observed data. Ultimately, both models provide a likelihood indicator, that the data belongs to the data distribution(training set).

### *Explicit Models*

To construct explicit models, the authors used a Variational Autoencoder[17] as the deep generative model. These models are from a group of statistical models that perform approximate assumptions in latent variable models[18]. VAEs are constructed by two neural networks, the encoder network, and the decoder network. The job of the encoder is to take the original data and compress it down into a lower dimension which is a representation of the input data. The output of the encoder are parameters that correspond to the posterior distribution of the latent variable model. The parameters are then fed through the decoder which attempts to decompress the latent representation to the original form. This results in a loss of data while attempting to minimize the mean squared error between the original data and the decompressed data.

### *Implicit Models*

DDT implicit models are based on Generative Adversarial Network, a duo neural network team, like VAEs, but both networks are in control over their parameters. The goal of both neural networks is to minimize their loss function  $L$  by constantly changing their parameters[19]. The opposing networks called the Discriminator and the Generator, are parameterized differently with a set of weights. In each interaction, the Generator formulates samples from a latent variable  $z$ . The samples are passed through the parametric distribution, generating output samples labeled as fake for the Discriminator to evaluate what is real and what is fake. The Generator will train over these results and a new iteration of samples is ready to be evaluated. As the loop continues, the distance between both real and fake distributions becomes smaller as samples generated will start to resemble the true distribution.

### *GearBox*

In practice, the authors used a dataset provided by Schmidt[20], Gearbox, which represents the effect of a broken tooth within a one-stage spur gear transmission, as a benchmark dataset to compare both explicit and implicit models and the generative results of each one. The comparisons showed that both models can generate realistic data to a certain degree but the behavior of implicit models showed more sensitivity to deviations, unlike the opposing model. In conclusion, both DDT implementations seem to be working well with the non-stationary nature of data.

### *CMAPSS*

CMAPSS dataset consists of simulated data of the entire life-cycle of multiple turbofan assets, based on the Commercial MODular Aero-Propulsion System Simulation developed

by NASA. Each sample is a time series signal of various sensor data and the operating scenario measured periodically. The dataset contains modes of failure correlating to module degradation. The goal of using CMAPSS was to test DDT ability to achieve the various modes of failure without ever seeing it in the first place.

According to the results, DDT was able to generate various modes of failure under discrete operating modes. Was also demonstrated that the implicit model was able to learn a representation of health being an excellent diagnostic tool.

The study despite not using medical measurements was able to demonstrate the possibilities of Deep Digital Twins in PHM. Instances that can generate synthetic datasets representing the life-cycle without ever learning beforehand any points of failure in the turbofan dataset. In addition, GAN was identified as a well-suited DDT's deep learning framework, especially in the Gearbox dataset compared to the VAE results.

## 2.3 AUTOGENERATIVE ALGORITHMS AND MODELS

### 2.3.1 Overview

The access and availability of data has failed to match increasing rise of startups worldwide, leading many to use the same data available to the public. To overcome the problem, the idea to generate new data became possible using mathematical methods, specifically machine learning, to train and generate identical data for self use.

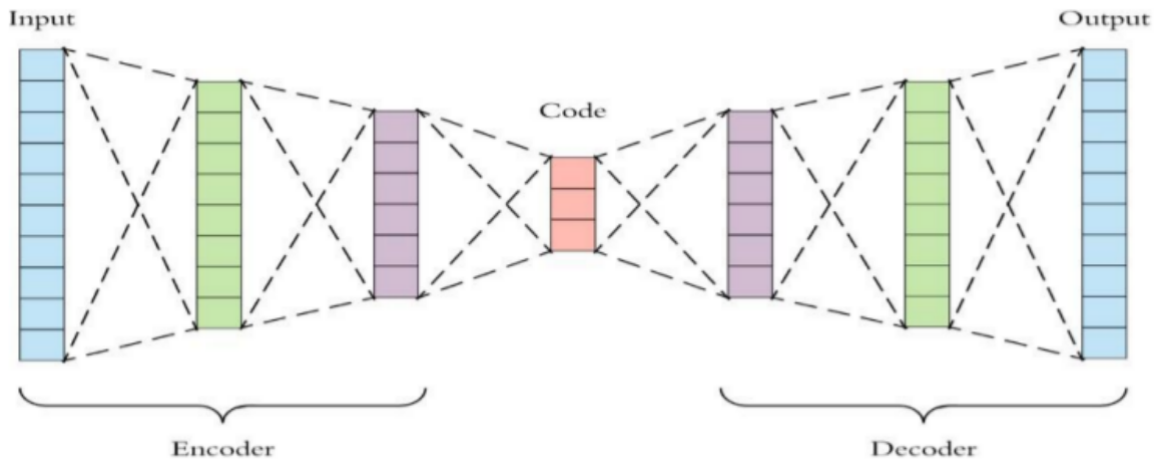
#### *ECGSYN*

Electrocardiogram (ECG) is a recording of the electrical activity of the heart, as it pumps the blood to the body. This recoding can be treated as a signal providing information about the occurrence of the electric impulses, their frequency and intervals. The name was cleverly chosen by developers Patrick McSharry and Gari Clifford, ECG stands for Electrocardiogram and SYN for synthesized. The application is based on differential equations models to produce features of the ECG, and the many defects which are common like beat-to-beat variation, respiratory sinus arrhythmia, and R-peak modulation. The output is mainly used to assess medical research and calibrate machines.

#### *Auto Encoders*

Autoencoder is an architecture of neural networks which reconstructs data. The architecture consists of two components: encoder and decoder. The encoder's job is to compress data down to a certain dimension while preserving the main features[21]. The compressed object is then used as input to the decoder component, which will extract and reconstruct the compressed object's representation. Autoencoders use a backpropagation algorithm to help reduce the noise and error of the results.

Initially, autoencoders were designed for image reconstruction but the idea of reconstructing good images into blurred-out versions made no sense. Instead, they were used to reconstruct corrupted images and even low-resolution images.



**Figure 2.3:** Autoencoders from [22]

### *Variational Auto Encoder*

Variational Auto Encoder (VAE)[21] is a framework to produce efficient approximations of maximum likelihood of maximum a posteriori estimations for the parameters of the decoder which creates a possibility to create a random process and generate artificial data. Using a probabilistic encoder,  $q(t|x)$ , VAE is capable of developing the distribution over latent variables  $t$  given data  $x$ . The same occurs with the decoder component, given latent code  $t$  it produces the distribution of the value of  $x$ . Just like the components of autoencoders, VAE's components are neural networks and the model should be trained using gradient-based methods and back propagation algorithms.

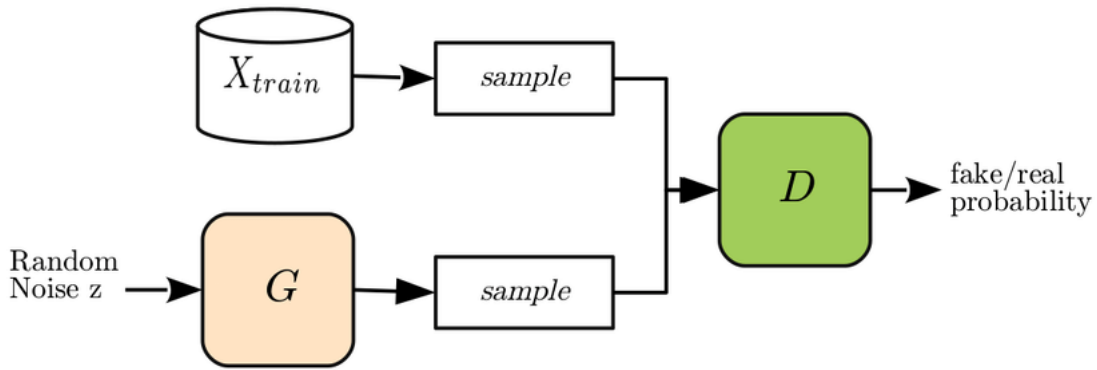
The findings on autoencoders are quite interesting as it aligns with the project's ideas when it comes to generating artificial signals through a pre-existing dataset. The in-depth study on autoencoders showed differences between the input and output and the effects it has on images. Since this dissertation focuses on biological signals, this outcome will not entirely reflect on the results.

### *Generative Adversary Networks*

Generative Adversary Networks are a common technique used in both semi-supervised and unsupervised learning. They excel through implicitly modeling high-dimensional distributions of data and are characterized as a pair of networks competing against each other. The pair of networks have distinctive names, one being the Generator, where the goal is to generate data, and the Discriminator which will try to compare the differences with the real data. Both networks are simultaneously trained[23] to the point where the Discriminator is not able to differentiate generated data from real data.

There are a set of conditions[23] that makes GANs an interesting approach to generative data

- The generator does not have direct access to real data, the only way of learning is through the discriminator's evaluations.



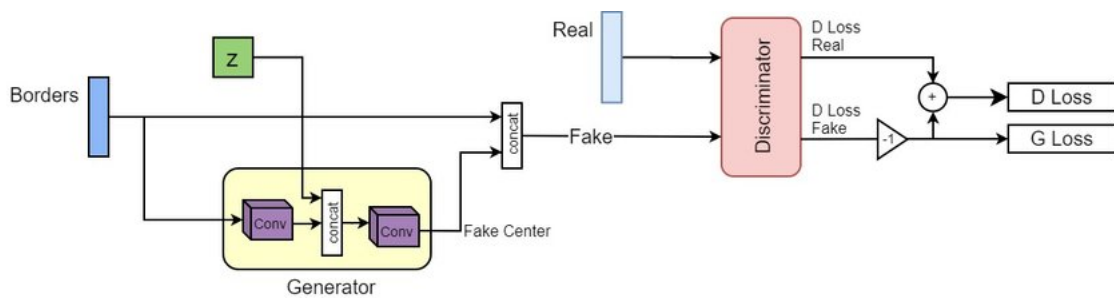
**Figure 2.4:** Generative Adversary Networks from[24]

- The discriminator has access to both generated and real stacks of data.
- The error signal is the binary response from the discriminator detailing if the data is real or fake.
- The error signal can be used to train the generator, to the point it generates identical data in which the discriminator can not tell if it's real or not.

GANs provides synthesizable data sample from random noise but are considered difficult to train due to vanishing gradients and would later require careful hyperparameter tuning and model selection for training. There are simpler models like the autoencoder referred to above and The Wasserstein GAN.

### *Wasserstein*

The Wasserstein GAN (WGAN) was first discovered to ease the training difficulty and avoid problems throughout the process[25]. The Figure below shows a flowchart of the WGAN and the differences between the discriminator of the WGAN and GAN. GAN discriminators affect the classification of real and fake samples, unlike the WGAN's discriminator which is represented by the EarthMover(EM) distance between both samples.



**Figure 2.5:** Wasserstein GAN from [26]

### *SpectroGAN*

SpectroGAN is a deviation structure[14] like GAN's consisting of two models: a generative model G and a discriminative model D that is made of deep neural networks. GANs were

designed to process images, 2-D like structures. Since ECG is a 1-D structure, in this paper[27], the authors used a short-term Fourier transform (STFT) over 1-D ECGs to obtain a 2-D array as the input of GAN. The only caveat is choosing the appropriate window size during STFT conversion since it is directly connected with obtaining a good resolution.

### *Wavelet GAN*

Wavelet transform is a different approach to analyzing 1-D arrays like audio and of course biosignals[14]. Unlike the STFT, the window is changed automatically according to the frequencies it focuses on: straightening the window size at high frequencies and widening at low frequencies. This allowed researchers to observe the signal’s characteristics on different scales. The function that defines WT of a signal is:

$$T(a, b) = 1/a \int f(t)\psi(t - b)/a$$

### **2.3.2 Wavelet-based model**

Wavelet-based model is an autoregressive model[14] that predicts the current value of a discrete-valued time series  $x = [x_1 x_2 \cdot \cdot \cdot x_T]$  using past samples:

$$p(x) = \sum p(Xt|x_1, \dots, x_{t-1})$$

where  $T$  is the length of the time series. The conditional probability distribution in figure 1 represents a dilated causal convolution with a large receptive field. The distribution is categorically resulting in a raw flexible audio waveform modeling.

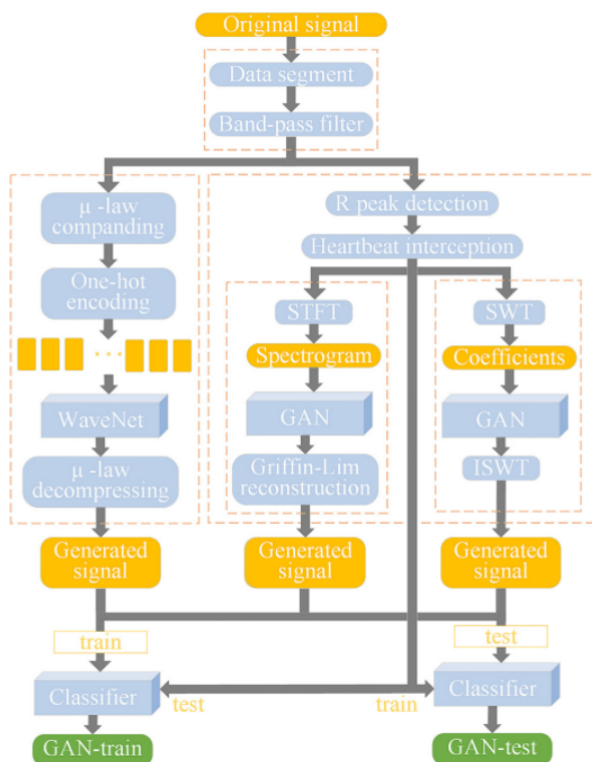
Its also possible to manipulate the distribution with inputs,  $h = [h_1 h_2 \cdot \cdot \cdot h_T]$ ,  $p(x | h)$  An example of this is in text-to-speech synthesis, with the help of input  $h$ , Wavenet can produce waveforms with specific characteristics like phoneme and speaker identity. After some more research on speech recognition, many showed that the Wavelet Transform and its variants were proven to be useful due to the efficacy of feature extraction and the accuracy of speech production models. and so Wavelet-based models in this dissertation were considered a strong candidate for synthetic signals.

### **2.3.3 Generating electrocardiogram signals by deep learning**

Electrocardiograms have over time become the cornerstone in detecting heart-related illnesses and anomalies. For scientists to continue research without using a physical heart, alternatives like historical databases, for example, PhysioNet, are used to further understand and solve today’s sicknesses. As humans evolve, so will the body. Thus yesterday’s recordings may not correspond to today’s problems, meaning that the numerous terabytes of samples of patients with heart issues will all expire. In the following paper[27], the authors detail an alternative to generating synthetic realistic ECGs using 3 unique methods of deep learning.

## Dataset

In the study, the authors used the MIT-BIH arrhythmia database containing 48 ECG samples, roughly around 30minutes long with a frequency rate of 360Hz. Each sample consists of leads, MLII and V5. For simplicity, they used MLII for the assignment. Dataset was broken down into 2 sets, training, and evaluation. The sets included 3 types of heartbeats: normal beat, left bundle branch block beat, and right bundle block beat. Important note, the dataset also contained other types of heartbeats, but due to their irregular nature to the normal beat, they were excluded.



**Figure 2.6:** Deeplearning Workflow from [27]

The Figure above shows the workflows of each method with the corresponding preparation required. The original signal undergoes a filtering process which creates independent segments of data and the associated label. A band-pass filter was used to eliminate segments that had a different frequency than the majority. The segments are then divided into 2 sets, training, and evaluation, to test the effectiveness of each implementation and interpret the results.

## WaveNet

WaveNet is based on autoregressive models, used primarily for creating audio samples like speech and music[28]. Constructed using convolutional layers without pooling layers. The output layer is a softmax layer used to categorize the distribution over  $\xi$ . The reason is primarily used in audio samples is its ability to generate long, high-quality signals, and since ECG holds a complex structure making this method is perfect for long-term signals.

WaveNet starts by consuming prepared ECGs as input. Following and training the convolutional layers, the softmax layer, output layer, create a waveform sample point. The sample point is then used to generate the following point until a robust signal is created.

## Results

The authors were able to conclude that their proposed methods were vastly more successful and complete compared to traditional algorithms. The only downside to using their models is the setting parameters that would change the heartbeat structure in traditional algorithms. Otherwise, the results of both GAN models were very successful hitting above 90% realism and can simulate abnormal beats time in time again.

## 2.4 DISCUSSION

Reflecting back, it was possible to draw an outline of the project. Yet, it was still unclear how the dataset should look like upon granted access, how it should be after the system has finished predicting the segments and how segments should be returned in real time. To further investigate these questions, started looking into several libraries that supported auto encoders, one which was the Keras of Google Tensorflow. This prompted a simple python program which instantiates an auto encoder and had the dataset loaded in memory. This caused a lot of mistakes during the execution, with main issue being the dataset itself. The dataset was complete in its nature, raw, containing the baseline and emotion in question. The only issue was how difficult it was to use as a training set and understanding how the auto encoder should perceive the data. The first step was to trim out the excessive data, the data points before and after the emotion timestamps. The goal behind this was to create specific sets of an emotion. Upon running it through the encoder, it failed to produce visible electrocardiograms, instead the end result was a segment of noise. After investigating further into parsing ECG segments, a python library called Neurokit appeared, allowing us to break down the segment into heartbeats, and the idea of predicting heartbeats instead of segments made much more sense and thus implemented a simple python program which took all four signals, parsing the ECG heart beats and trimming the other three according to the timestamps of each heartbeat. This made it possible to develop an entirely new dataset containing numpy arrays of each signal of each emotion. Making it also possible to create baselines of each patient of the dataset plus an aggregated set containing all of the patients' segment data.





# Use cases in computational physiology research

The following chapter illustrates a description of the target users, the scenarios in their daily routine, the fundamental connection between the target users and the system, use case and non functional requirements of the system.

## 3.1 DISSERTATION APPLICATION

The motivating scenario for the present work is related to the research and academic activities in IEETA research unit. The students and researchers often need to use biosignals samples to develop methods, for example, to test mobile applications use cases. A system that offers basic simulation of biosignals and provides a programmatic friendly access to them could be used by developers to test their applications with realistic data and also by teachers to create assignments and learning activities.

Regardless, there is an array of factors to consider when collecting physiological data, the first being the frequency of collection (the number of data points collected per second), critical timing to collect (an event in which the variable expresses data variation), cost (financial compensation, experiment planning, data collecting, equipment required, etc...), variability in test measurement (often presented as a coefficient of variation), and care need for standardization of test methods and their interpretation (e.g., referencing vs. a “gold-standard”).

Another important consideration concerning the use of biological signals is when such samples are obtained relative to the definition of variables and outcomes (e.g., are they concurrent or separated by time). The importance of critical timing in collecting various descriptive or risk factors is illustrated by the following example. A researcher wants to perform cardiac profiling to examine differences between emotions. To obtain cardiac signal samples, he/she performs a study, having the patient respond to a questionnaire followed up

by a video or audio expressing the emotion while wearing sensorial devices which capture the patient's pulse.

Storage of biological samples, as well as technical factors relating to their measurement, are normally constructed into datasets identified by the patient's unique identifier. These datasets are vastly used for the research and development of new technology and systems. The data is collected from raw data transmitters specifically placed on the patient to collect the biological signals in their rawest form. The data is later filtered removing potential noise and aggregated into a dataset.

## 3.2 PERSONAS

The purpose of this section is to identify the target audience, entities which will benefit the system provided in this dissertation.

### 3.2.1 Persona A

Joshua Macomb, a 53-year-old professor at University of Aveiro lectures a variety of classes in his academic field computer science. Joshua is known to be punctual with his lectures and extremely strict with evaluations. Joshua was recently assigned to lecture a class called Introduction to Software Engineering, a course that talks about software development patterns and module integration.

On the very first assignments, students were told to build a broker-client that would consume messages from a server, process the incoming data, and create events whenever a condition has been met. This requires the professor to organize, install and configure a broker messaging server with a producer client sending data to the server. Unfortunately, Joshua has 5 different courses to prepare for and does not have the time to prepare an environment and the security from malicious users.

Joshua is looking for a service that is compatible with the messaging protocol, creates dynamic topics with data customized by him, and assigns topics to each group of students.

The University of Aveiro has recently launched a project regarding the health of its football athletes. The project consists of evaluating the effects football has had over one year of playing football and figure out alternatives to the equipment used on the field.

### 3.2.2 Persona B

Candice Doe, a 21-year-old university student was assigned to implement the usability experience and front end of the system so coaches and medical experts could carefully observe their athletes over time. Candice downloaded a dataset consisting of multiple biosignals from PhysioNet to test the charts on the application, but she ran into multiple visual bugs and realized that the signals were at a completely different frequency than what the system was meant to work with. She has never changed the properties of a signal and is quite frustrated because she was not expecting this to happen days before the deadline.

Candice is searching for datasets that are compatible with the system requirements, containing multiple signals so she could present the front end to her colleagues on Monday.

### 3.2.3 Persona C

At Saint Marie Hospital, the psychiatry department administration has recently bought equipment for their staff. Some of the equipment requires periodic maintenance on the sensor readings and data processing. Joe, a 34-year-old medical technician, was assigned to calibrate these machines over the weekend. He would bring his laptop containing software applications that could diagnose the machine and report any anomalies to fix. Unfortunately, the new equipment had a different input port and a Bluetooth module installed.

Joe is willing to migrate or add to his toolbox new tools that could diagnose this newly purchased equipment at Saint Marie.

#### *Personas Conclusion*

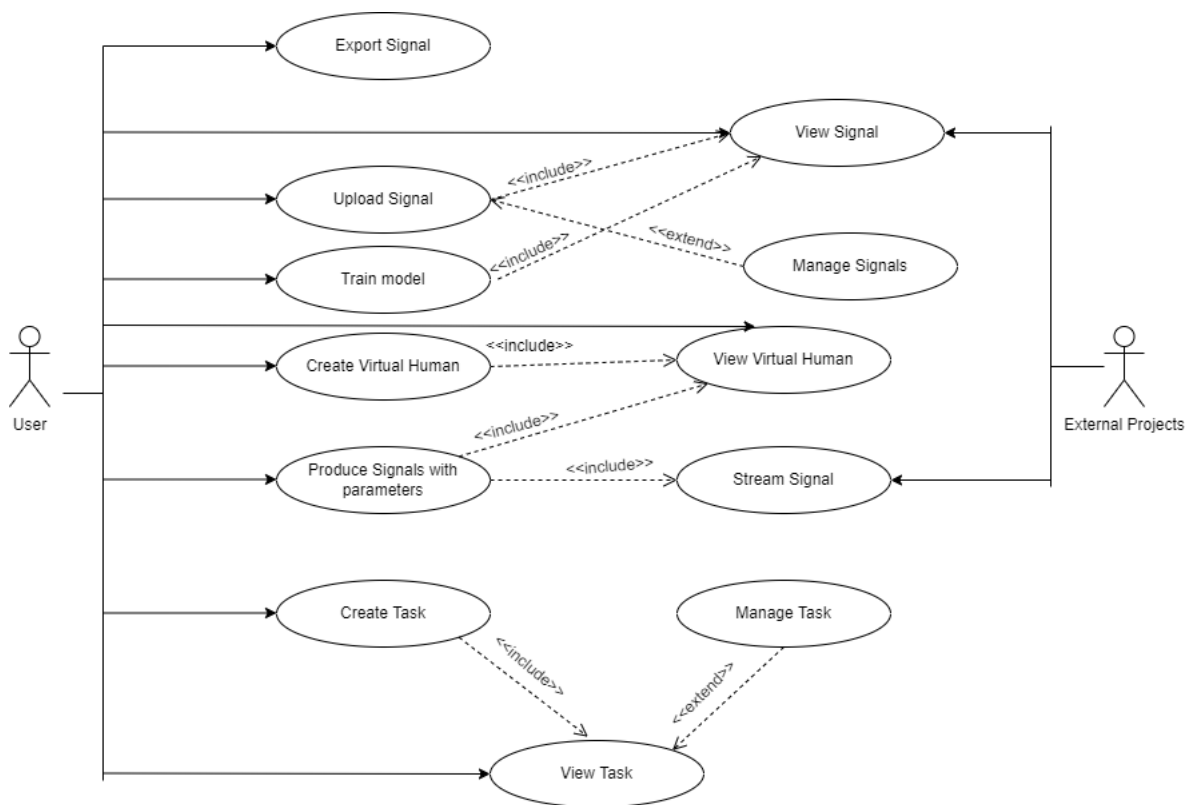
From the scenarios above, it was possible to extract critical problems that indicate a need for a:

- A system that provides data source links to feed in the user's target project.
- The data within being dynamic and customizable to the user's requirements.
- Compatible on multiple devices and operating systems.

### 3.3 USE CASES OF THE SYSTEM

The goal of the proposed system is to implement the Digital Twins concept, where the user is free to create their very own virtual humans using pre existing data or their own and predict the segments under a selected state. The system will also support third-party projects that require biological signals through implementing a service with the proper security to create and stream biological signals using customizable parameters which change the output of the signals.

The following use cases were designed to help experts in research and software development to interact with the system:



**Figure 3.1:** Use case Diagram

User Case	Description
Produce signals with parameters	With a set of tools available, the user will fill out a form to create a sample. This form will contain important information related to the structure and effect of the signal.
Export the signal	There will be an option for the user to download the created sample in a readable format like CSV.
Stream signal	Alongside the tools, there will be a monitor showing the results of the user's inputs and alterations during the stream of the signal.
Upload signal	There will be a place where users can upload samples of their own and have the system create samples derived from theirs.
View Signal	Allowing the user to view available signals.
Manage Signal	Allowing the user to modify or delete signals at their disposal.
Train model	An option where users can train models with the selected baseline.
Create Virtual Human	Users will have the ability to create virtual humans with a set of attributes.
View Virtual Human	Allowing users to view available virtual humans.
Stream Signal	An option where users can view the signal.
Create Task	Allowing users to create segments of a specific length and frequency sample without being present.
View Task	An option to view the Non-interactive signals generation.
Manage Task	Allowing users to pause, delete or duplicate tasks.

**Table 3.1:** User Case and Description

### 3.4 QUALITY ATTRIBUTES/NON-FUNCTIONAL REQUIREMENTS

The Non-functional requirements, the table below, will help cover all the remaining features of the system. Criteria that define the operation of the system and indirect expectations from the users.

Non-Functional Requirements	Description
Performance	Performance correlating the response times from the system. When the users instantiates a virtual humans, there should not be any interruptions or delays within the segment.
Portability	If the user upgrades their system, vertically or horizontally or migrate to entirely new environment the system should be able to operate equally.
Compatibility	It should matter what language or framework the user is using on their end, the system should be able to communicate properly.
Capacity	The system should offer storage to users for their segments and models.
Reliability	The system should be able to produce identical signals after extensive use.

**Table 3.2:** User Case and Description

### 3.5 SCOPE LIMITS AND EXCLUSIONS

The scope of the dissertation is ambitious and thus several topics will be addressed in a limited manner or excluded from the dissertation.

#### Monitoring

Monitoring will be excluded from the project due the complexity it will bring and unnecessary value. To display the Digital Twins concept, the system requires a simulation process and a management interface to guide the user to simulating their own segments.

#### Security

The requests and data that flows throughout the system belongs to the user and should use a certain level of security, but unfortunately this will delay to implementation process and in unnecessary manner as data security is not the scope of the project.

#### Bio Signals

Despite being the main source of data, the scope of this dissertation will focus on creating artificial signals with a sufficient degree for research and development purposes only, meaning the time-series, labels and sample frequency will be the main focus.





# System Architecture

## 4.1 ARCHITECTURE OVERVIEW

The proposed architecture provides a scalable and high available system that supports multiple requests in parallel. Using default datasets, nodes are able to generate unique signals to the user desire by first compressing the selected signal into a smaller dimension, load the selected model into memory and decompress it into a new signal. The following figure illustrates the main modules of the system.

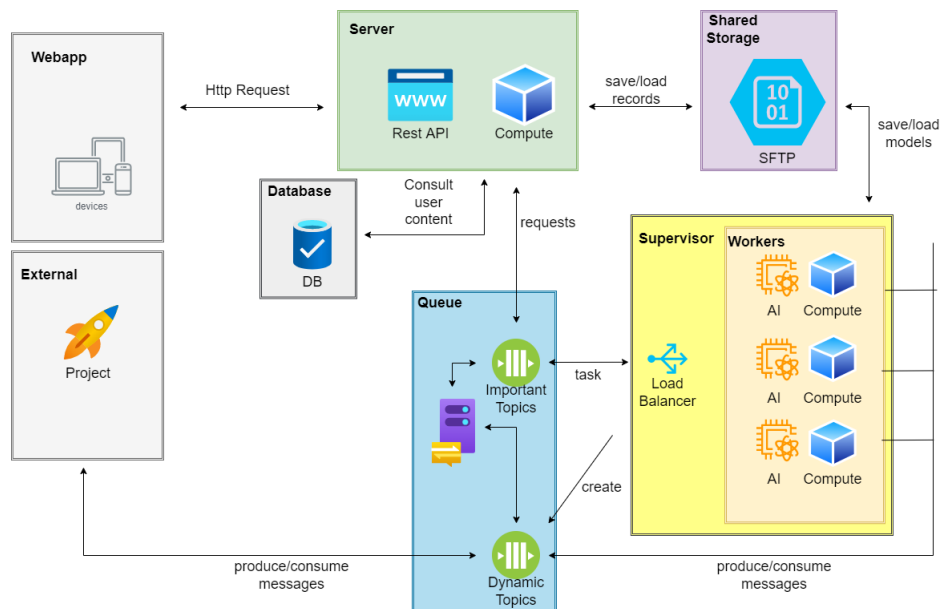


Figure 4.1: Architecture Design

## External

The external module is referred to users that will be interacting with the system communicating over the messaging server, the system should be flexible on where the requests are coming from and validate them before execution.

## Webapp

The server modules consist of multiple instances, one being the Web App, the frontier of the entire system, a place where all of the use cases are available to the user. Tools, monitoring, information on how to connect and download the virtual human's vital signs. The user will be able to manage their virtual humans on the fly from anywhere. The compute component is where all user input data is validated, stored and executed. Once validated, it gets broken down into tasks and sent to the messaging modules using a dedicated client.

## Database

The database module represents a cluster of database servers to provide users with high availability to their virtual humans and account information. The purpose of the cluster is to provide persistency to volatile data at a fast speed, this means whenever a user decides to an attribute of their virtual human, the system will not need to upload the entire virtual human into memory. For development, only one instance was used but in productions, it's recommended to deploy multiple instances and balance the workload between the nodes.

## Shared Storage

Shared storage allows multiple components to save a large amount of information, like generated samples and models that are related to the user. Having it shared allows the server to answer requests of the trained model, change virtual human parameters, etc and the work can easily consult without requesting help from the server.

## Queue

The queue module is a key piece to the system, this module allows other modules to communicate with each creating pattern of integration and organize the flow of data between them. There are 2 types of topics in this module:

- Static Topics, which permits modules to interact with each other.
- Dynamic topics, allowing clients outside of the system to download or tune real-time data from the virtual humans.

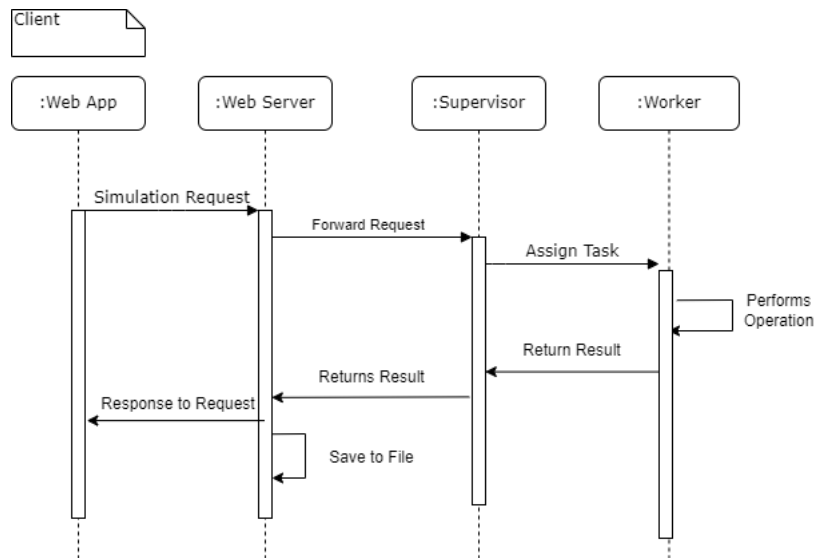
## Supervisor

The Supervisor module consist of 2 types of components, the supervisor instance, and worker nodes. The supervisor is in charge of distributing tasks coming from the server to its nodes fairly. Meaning there should not be nodes doing anything and other constantly receiving tasks. The Worker component is made up of artificial intelligence, digital twin, and logical computation which handles and sends the result of each task to the queue module.

### 4.2 INTERACTIONS BETWEEN MODULES

The interactions that occur between modules of the system are mainly due to the user's interactions with the system and optimization techniques which will improve response times to future requests, saving trained models, or keeping strips of data in memory ready to be shipped.

The interaction begins with the request to simulate a virtual human, this operation generates requests from the web app on the user's end, sending requests to the server. The server validates the origin and the structure of the request before parsing and sending it to the supervisor through the messaging system. Once the supervisor receives a request formatted by the server, it proceeds to select a node and assign the task to it. The node sends periodic updates to the topic of the request and ultimately sends the result. There are periodical tasks that are in charge of freeing resources and removing old messages from the system.

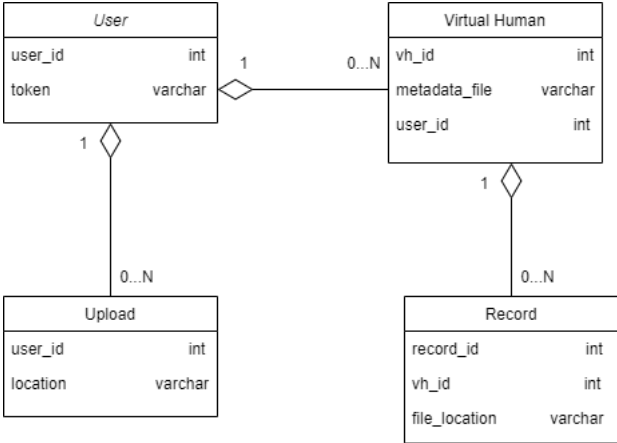


**Figure 4.2:** Sequential Diagram of the dataflow

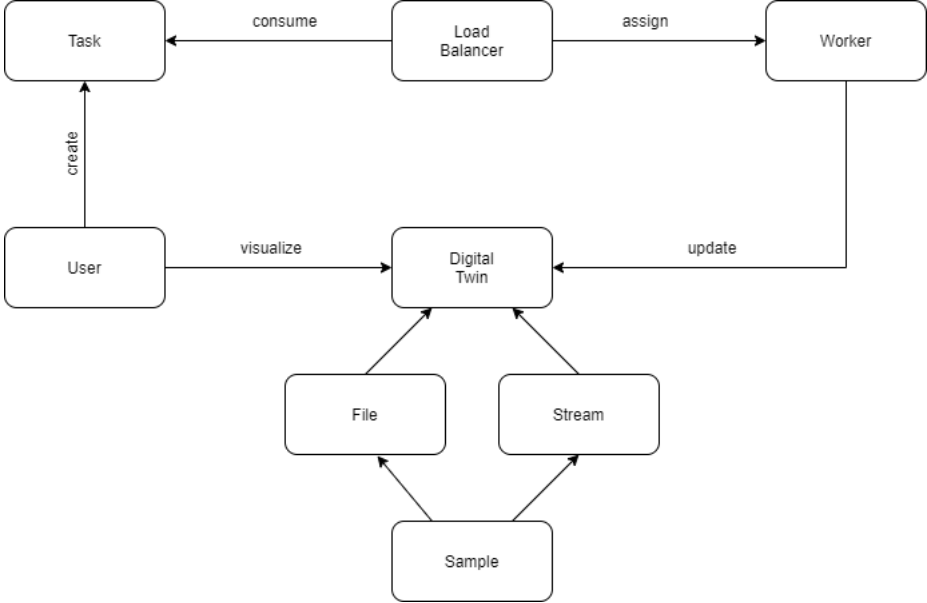
### 4.3 DOMAIN CONCEPTS

A researcher organizes various studies where its necessary to collect biological signals of their participants. Each case study may collect different types of signals, including ECG, and multiple states are recorded during each session, for examples the recording of emotions. Each

recording may be labelled differently, and associated to the patient. The information stated above can be organized and saved into a database so it can be used later when a new patient participates in an existing study or a new recording of an existing patient appears.



**Figure 4.3:** Database Table Design



**Figure 4.4:** Domain concept Diagram

#### 4.4 TESTING REQUIREMENTS

Testing Requirements are used to ensure the reliability and consistency of the system and the user's goals. The following table illustrates the set of testing requirements that align with the system requirements.

- Access the IDE Tool page from the web application using an unauthenticated session and validate the following page being the login operation page.
- Create a virtual human on the same page and validate the dropdown on the page containing the new virtual human.
- Open a virtual human and validate the player toolset becomes available.
- Open two virtual humans and validate that two sets of players are open on the screen.
- Open a virtual human, press play, pause, play, then stop and validate the behavior of the graph through multiple interactions. When the graph is paused, the data continues on-screen, when is stopped, the data on the screen is wiped out.
- Access the Task page from the web application using an unauthenticated session and validate the following page being the login operation page.
- Create a new task and validate the Task page containing the new task.



# System Implementation

This chapter illustrates the decisions and procedures taken to implement the system modules and the integration between them accordingly to the proposal in chapter 3.

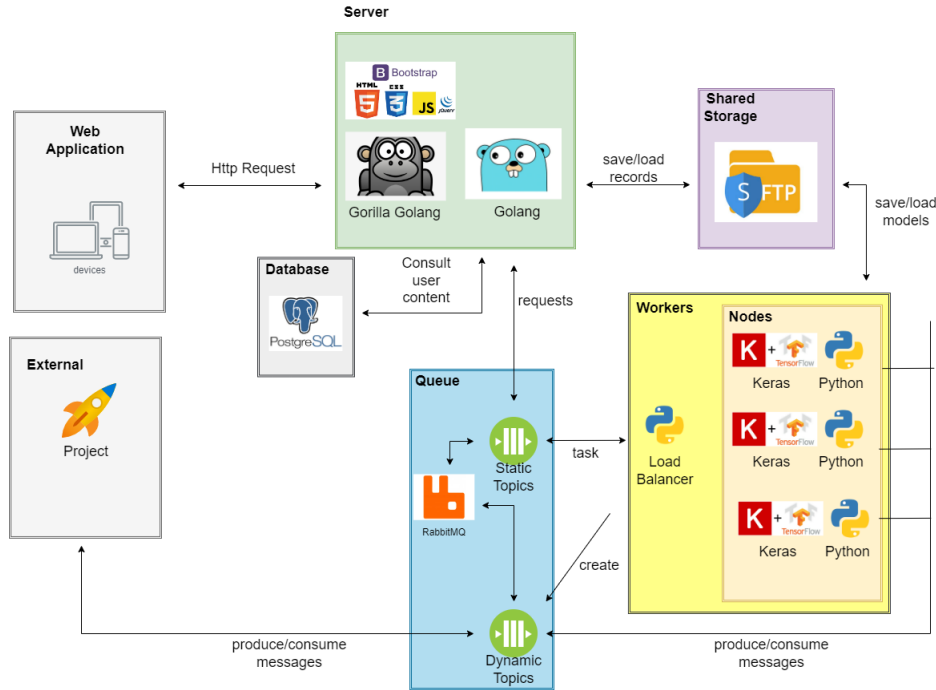
## 5.1 SYSTEM DEPLOYMENT AND IMPLEMENTATION TECHNOLOGIES

Each module has a unique role within the system, and each one requires a different set of technological stacks selected to support their implementation. In the figure below provides a visual understanding of the system and how each module communicate.

Each task consists of:

- **Server:** The server was implemented using Golang, which facilitates parallel executions, with the supportive libraries gorilla/sessions to deploy the web application and steadyway/amqp to instantiate rabbitmq clients.
- **Webapp:** The web application was developed using JQuery and Bootstrap where it allows the user to interact with the system in a responsive plane.
- **PostgreSQL:** Dockerized PostgreSQL to save the user's assets directories, metadata of each virtual human and user accounts.
- **RabbitMQ:** Dockerized RabbitMQ with GUI Management to setup static topics and allow data to flow throughout the system.
- **Supervisor:** The supervisor was implemented using python 3.7, with the help of the threading library, it was possible to receive and send data without pausing.
- **Worker:** The workers do not communicate with each other, only with the Supervisor. Developed in Python 3.7 using autoencoder library of Keras, they are able to predict the segments independently.
- **SharedStorage:** The shared storage allows the modules to use models trained by the user. This storage is used by the Server in case the user would like download the model, and the workers to predict new segments of data.

The modules must be able to communicate with each other, but does not require Internet access by any means. If the entire system were to be deployed in a private network, users would also need access to the same network to be able to use the system.



**Figure 5.1:** Technology Stack

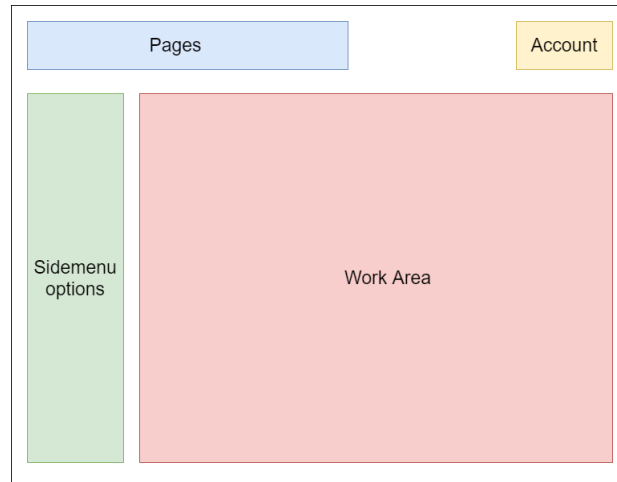
## 5.2 DATASET USED

For this demonstration of the digital twin concept, the decision on using historical data was made, and decided to use a dataset that was supplied by the supervisors of this dissertation. The dataset consists of records of patients recording 4 of their biological signals, Electrocardiogram, Electromyography, and Electrodermal activity. These signals were recorded under the consent of the patients and the research is transparent on the way it was recorded. The dataset is based on the differences between happy, fear, and neutral. Patients were assigned videos to watch during the experiment, beginning with a neutral base video. The goal behind this decision was to create a baseline that would correspond to the neutral emotion the patient was feeling. After the video ended, the expert that was presently orchestrating the experiment would change the video to express a different emotion. Once the video was done, the patient would be fed a neutral video to return to the original emotion. The process continued until all target emotions were recorded. This is one of the approaches discussed in chapter 2, where digital twins can emerge through historical data.

## 5.3 WEB APPLICATION

Users will be able to interact with the system through a designated webapp, available from the server. The interface design is based on the following template where users can navigate through the pages at the top of each page, a side menu showing subpages of each page, at the top right corner the account operations, and finally, the work area where the user can interact and observe the state of the system.





**Figure 5.2:** Template page

Every page in the webapp is only accessible to authenticated users, meaning the user would need to register or log in with an existing account. Whenever an unauthenticated user attempts to access any page of the webapp they are immediately redirected to the login page.

#### *Interactive web app for simulation setup and control*

The IDE Tool page is where users can create, simulate and observe their virtual humans in real-time. To access this page, users will first need to authenticate, then navigate to the top of the page and select IDE Tool option which will redirect you to. On the page, users will be prompted with a dropdown displaying all of their virtual humans created by them. Next to the dropdown are two buttons, Add and Open.

#### Add

The Add button when clicked will prompt the user a form requesting a name for the virtual human. Here the user enters a name identifying the virtual human, if the name is taken by another virtual human under the same user, the system will alert the user and return to the initial state.

If the name is not taken, the system proceeds to create a virtual human by creating an internal directory, registering the virtual human, and alerting the user of the system's success.

#### Open

The open button will load the virtual human on the user's screen in standby mode, waiting for further instruction. After loading, the user can observe the various operations that can be done to the virtual human.

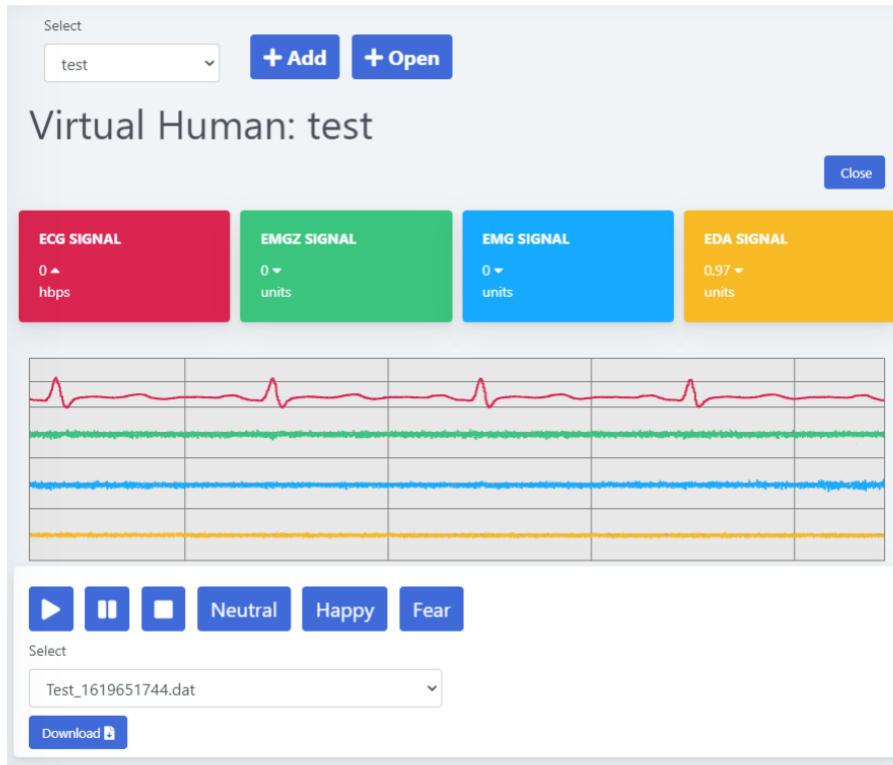


Figure 5.3: IDE Tool Page

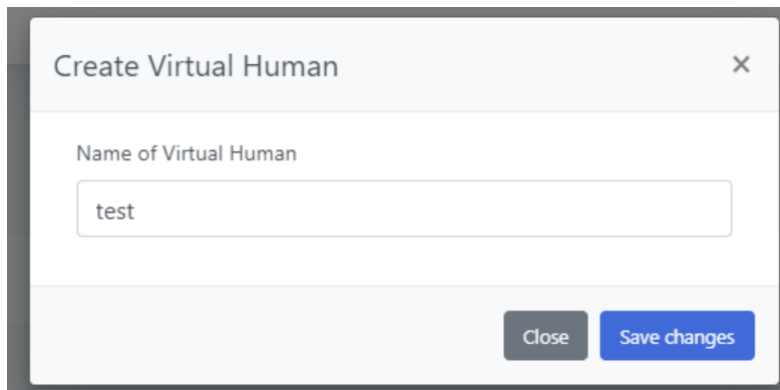


Figure 5.4: Add Dialog

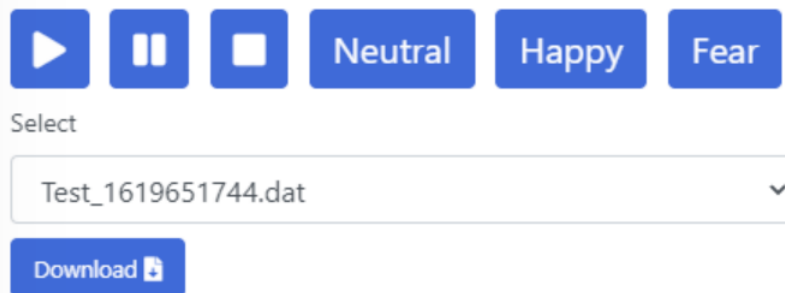


Figure 5.5: Virtual Human Player

## Play Button

The play button, when clicked, will tell the system to start the virtual human. Behind the scenes, and as mentioned in chapter 4, the system notifies the supervisor requesting blocks of data of the virtual human with the selected emotion. The supervisor then reads the request and launches a node to handle the request. Once the output is ready, the node sends it through the designated channels for the server to read, record to file, and send the block to the user's webapp.

## Pause Button

The pause button will simply pause the webapp from sending requests and updating the chart above. Behind the scenes, the server will not be aware of the pause because the system is designed to respond upon request and not keep track of any signal through the IDE Tool.

## Stop Button

The stop button is similar to the pause button, where it stops the webapp from sending requests to the system. Instead of pausing the chart, it cleans the chart, reset internal variables, and cleans the data queue. By resetting the variables, when the user presses play, the webapp will start with an initial request of the first block.

## Emotions

The emotion buttons allow the user to change the virtual human's current emotion. This will simply change a parameter in the requests so that the output data is the result of that particular emotion. Only one emotion can be selected at a time.

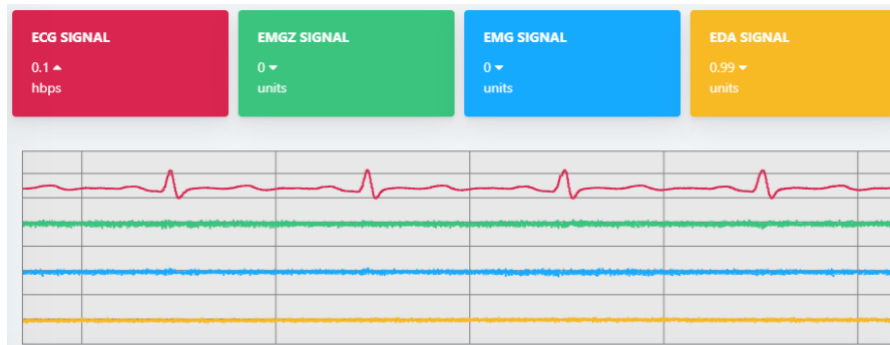
## Download Feature

Below the virtual human player, is a dropdown of recordings of the virtual human done by the user. The user can download these files by selecting the desired file and hitting the download button.

## Charts

Above the virtual human player are the charts that represent the virtual human's four biological signals. During the live feed, the webapp will constantly update the chart with

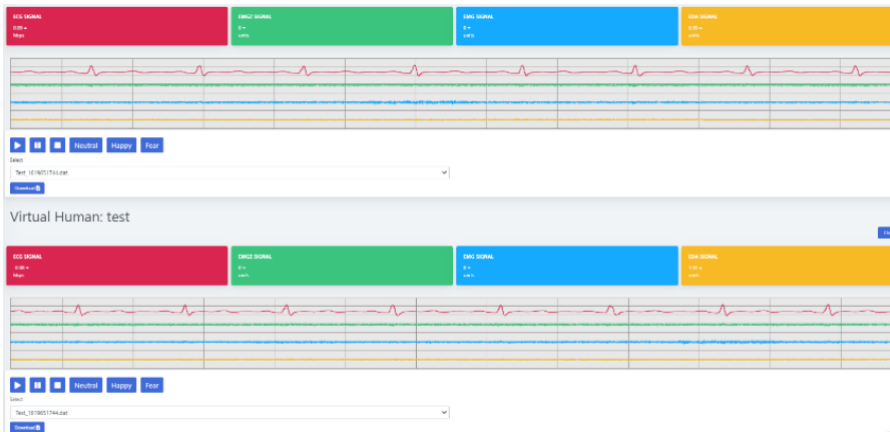
new data, providing real-time visual feedback to the user. This approach allows users to understand how the system work and the signals that are being produced on demand.



**Figure 5.6:** Virtual Human Charts

### Multi display

The virtual human displays were implemented dynamically, thus possible to display multiple virtual humans at once to the user, requirement in chapter 3. The user is allowed to open multiple virtual human instances, of the same or different virtual humans, and create records on demand. This feature is particularly useful to the user for real-time comparison between two virtual humans.

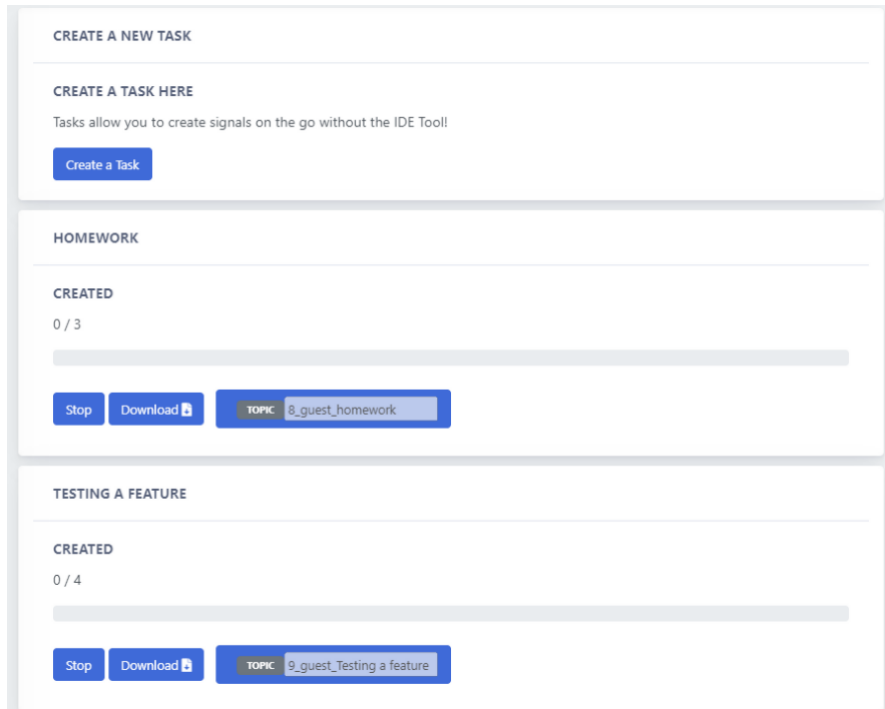


**Figure 5.7:** Multiple Virtual Humans

## 5.4 TASKS SCHEDULER

IDE Tool is designed to support real-time interactions with the system, meaning the user is required to be active on the webapp until the signal the goal is completed. If the user ever needed a record with substantially long length, in the IDE Tool, the user would need to stay toon until the end. With the Tasks scheduler, the user can schedule tasks in the system. This allows users to create multiple records far faster than IDE Tool and can route the data directly

to their projects through the messaging queue. Upon entering the page, the user will see in the work area all of the tasks that were created by them. At the very top will be the creation tool button allowing users, especially that have not created a task yet, to create a task.



**Figure 5.8:** Task Tool Page

Each task consists of:

- Name
- A status that indicates the operation within the system
- Stop button to interrupt the task
- Download button, to download the result once it's finished
- A dynamic channel that allows users to listen and receive the signal in real-time

#### 5.4.1 Create a Task

The Task feature allows users to tell the system to generate a record of their desire, defining the length, frequency, and virtual human. This feature is useful when the user wants to generate multiple signals at once. The user will need to click on the create a task button which will be redirected to a new interactive page. On this page, the user is presented with:

- Collection, containing the emotions with default parameters
- Tasks, which shows how the signal will look like
- Request, the summary of the task which is being made.

Users can drag and drop boxes that represent a portion of the signal. The box's parameters can be easily manipulated through the condition dropdown and the length in seconds.

In the Request column, users can define the name, the frequency of the signal and observe the length of the signal which changes according to the task column.

Collection	Tasks
Neutral Length: <input type="text" value="10"/> Condition: <input type="text" value="None"/>	Neutral Length: <input type="text" value="40"/> Condition: <input type="text" value="None"/>
	Neutral Length: <input type="text" value="10"/> Condition: <input type="text" value="None"/>

**Figure 5.9:** Task Creation

Request
Name <input type="text" value="Another test"/> <small>Name of the virtual human</small>
Frequency <input type="text" value="64"/> <small>Frequency in Hz of this Task</small>
Total Length <input type="text" value="50"/> <small>The total length in seconds of the current Task</small>

**Figure 5.10:** Request Column

After saving the task, the user is redirected back to the task page with the newly created task on their screen ready to be executed.

### 5.4.2 Export simulation to file

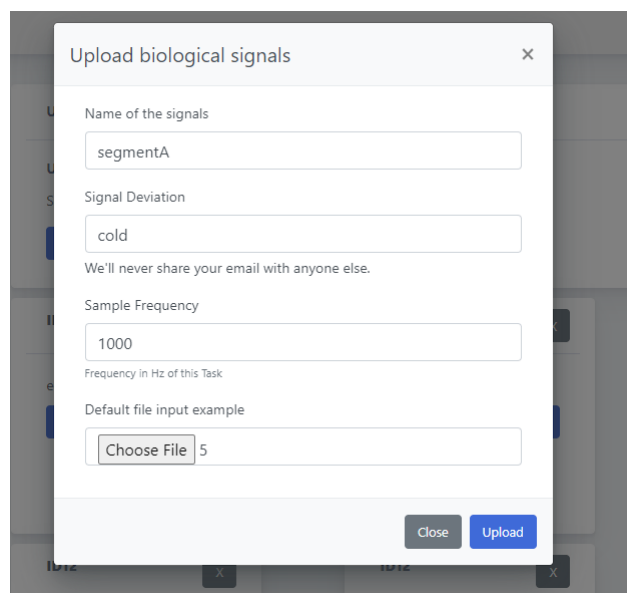
The file is created, by using either the IDE Tool or Tasks, is a CSV file composed of 4 columns:

- ECG
- EMG
- EMGZ
- EDA

This file can be accessed by clicking on the download button under a completed task or selecting a record from the dropdown on the IDE Tool page. The values in each column are in decimal format and can be parsed into floats.

## 5.5 DATA ACQUISITION

Data acquisition is one the main system requirements where the user can upload segments of data into VIRHUS platform and create new segments through the IDE Tool or the Task scheduler. This process can be done on the signals page where the user can upload their segments into VIRHUS. The user can identify the name, the condition, and sample frequency.



Upload biological signals

Name of the signals  
segmentA

Signal Deviation  
cold

We'll never share your email with anyone else.

Sample Frequency  
1000  
Frequency in Hz of this Task

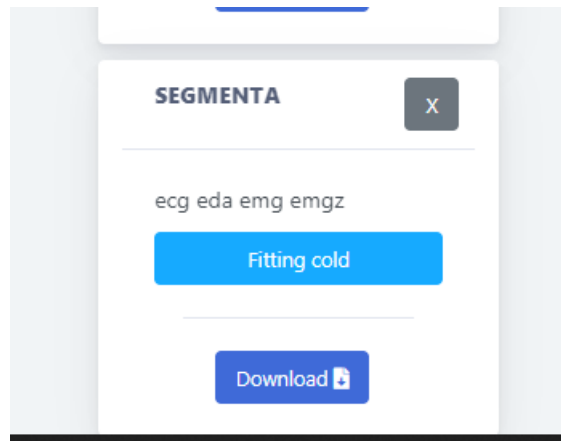
Default file input example  
Choose File 5

Close Upload

**Figure 5.11:** Request Column

To use the newly uploaded signal, the user will need to create and train a model on top of the segment. This action can be done on the signal page, selecting the segment and pressing the fit button, which will force the system to produce a new model for the user. While training the model the system will mark the specific segment with the word fitting indicating the user that it is training a new model. The user is allowed to train a model an infinite amount of times but not have more than one training session active.

Once the training has finished, a new model will appear in the library page. The user can also create new virtual humans on top of the new model and play them in the IDE Tool.



**Figure 5.12:** Request Column

## 5.6 BIOSIGNALS GENERATION COMPONENT

The biological generation component can be viewed as the digital twin component but does not check all of the boxes. When it comes to monitoring the real twin model adaptation, the component is not prepared to do such operations, where as its capable of training itself, receive input segments, and predicts output segments. It is composed of the Supervisor, nodes, and a program that trains the emotion models for signal generation. The training process is done separately from the signal generation as it takes up too much time and goes against the non-functional requirements in chapter 3.

### 5.6.1 Training

The training process is done by the server, this occurs when the user wants to create a new model using the selected baseline or retraining a model. The only time retraining a models makes sense when the user overwrites the baseline. The training is done by a program in python made by the author of the system, where the dataset used is filtered into a smaller, precise dataset. This is required because the dataset records were over thirty minutes long and had multiple emotions involved. The workaround to this was to create a new dataset by cropping out the unnecessary content from the sample and saving the emotion captured. The newly created dataset will then be used to create the models of each emotion, in this case, fear, happiness, and neutral. The program starts by loading the datasets into memory using a library called NeuroKit, a python library that focuses on medical signals and filters out any type of noise the signal might contain. After it instantiates the Sequential autoencoder from the Keras library and uses the clean signals as the training set. Once the training is finished, the program saves the model to storage where future nodes can load and perform predictions without the need to train the model all over again.

### 5.6.2 Supervisor

The supervisor is an instance within the system that operates to the server's request. These requests are transit from the server to the supervisor via the messaging queues of RabbitMQ, . The goal behind each request is to predict the desire emotion and comply to



the length of the request. Starts by listening to the designated channel for requests. Once a request arrives, the supervisor initiates a node, passing as arguments the incoming request and tunes in for more requests. This pattern allows the module to attend requests and generate output at the same time, paralyzing the operation. A request is composed of six parameters:

- Id, identifying the signal within the system.
- Length, the length of the signal in seconds
- Frequency, the frequency of the signal
- Signals, an array of biosignals that the node will need to make.
- Emotion, the emotion of the sign.
- Offset, the position of this request within the signal.

An example of a request look like the following:

```
1 {  
2   "id": "test_1634764865",  
3   "length": 10,  
4   "frequency": 1000,  
5   "signals": "ecg, emg, emgz, eda",  
6   "emotion": "neutral",  
7   "offset": 5  
8 }
```

**Listing 5.1:** Server request data structure

The id of each request is generated by the system, the length, sample frequency, signals and emotion are introduced by the user and the offset allows the system to order signal accordingly.

### 5.6.3 Node

The node component is the core piece of the system, generating biological signals to the user's request. Orchestrated by the supervisor, the python program utilizes Keras Autoencoder to read the emotion model from storage and predict an output. After generating the output, the node creates a response using the parameters of the request and attaching the output. A response contains four attributes: Data, the output generated by the node. Id, identifying the signal in question. Offset, the position of the output within the signal Emotion, the emotion present in this block of data.

An example of response looks like the following:

```
1 {
2   "Id": "test_1634764865",
3   "Data": [
4     [
5       [
6         "...",
7         -0.001753, -0.071964, -0.071964, -0.071964,
8         0.051148, -0.062068, -0.062068, -0.062068,
9         -0.023286, -0.042980, -0.042980, -0.042980,
10        -0.001354, -0.041946, -0.041946, -0.041946,
11        -0.033155, -0.062624, -0.062624, -0.062624,
12        -0.039346, -0.061383, -0.061383, -0.061383,
13        "..."]
14     ]
15   ],
16   "Emotion": "neutral",
17   "Offset": 5
18 }
19 }
```

**Listing 5.2:** Node response data structure

Similar to the request object, the response carries the id, emotion, offset, and the data in a three dimensional array.

## 5.7 WEB SERVER

The server component is charged with orchestrating the entire system, guaranteeing the functional and non-functional requirements are met. The component was entirely written in Golang, using the gorillas session library to save user sessions internally. Another important library used in Golang was the net.http library, which allowed the author to create not only the Web app but an API that would support the webapp requests. In the following table illustrates the endpoints of the API.

HTTP Method	PATH	Action
POST	/api/v1/login	Logs a user in and saves the session internally.
POST	/api/v1/logout	Deletes the session internally.
GET	/api/v1/data	Create a data request.
GET	/api/v1/vhs	Get all the virtual humans of a given account.
GET	/api/v1/vh	Get the metadata of the virtual human.
POST	/api/v1/vh	Create a virtual human.
POST	/api/v1/task	Create a task.
GET	/api/v1/recordings	Get records of a virtual human.
GET	/api/v1/downloadrecord	Get the entire file of the record.

## Login and Logout

The login and logout endpoints are where the user can authenticate and deauthenticate respectively. These methods were created to keep other operations limited to authenticated users, and all user-created content saved under the respective user.

## Data

The Get data endpoint is in charge of delivering data to the IDE Tool. The system differentiates the first request by the offset number, if the value is 0, this means the system needs to create a new ID and begin writing to a new file. The response from this endpoint is direct, indicating the id and data generated for the signal.

```

type ReqData struct {
    Id          string
    Timestamp   string
    Emotion     string
    Length      int
    Frequency   int
    Offset      int
}

```

**Listing 5.3:** Request data structure

```

type SendData struct {
    Id string
    Data []float64
}

```

**Listing 5.4:** Response data structure

## Get Virtual Humans

The Get Virtual Humans endpoint is responsible for returning all the virtual humans that belong to a given user. This is mainly used by the first dropdown in the IDE Tool, which lists all the virtual humans belonging to the authenticated user. The requests consist of the user's id, token, username, and message. The message attribute contains the reason for the request. The response structure is an array of the virtual human data structure.

```

type RequestPayload struct {
    Id      string
    Token   string
    User    string
    Msg     string
}

```

**Listing 5.5:** Virtual Human Request data structure

## Create Virtual Human

The Create Virtual Human endpoint is in charge of creating new virtual humans. This operation is done by the authenticated user through the IDE Tool, which tells the system to create a new virtual human object and add its details to the Database. The request structure used here is the RequestPayload structure in getVHs and the response structure is a simple 200 response code informing the webapp.

## Create Task

The Create Task endpoint is responsible for creating tasks within the system. This is used on the Task page by authenticated users. The request structure is composed of the signal identification, user information, and an array of actions. Action structure is what holds the necessary data for each box added to the request on the Task creation page. Attributes status, topic, current, path, and id are later added by the system. The response is a simple 200 status code to inform the webapp of the success.

```
type Action struct {
    Emotion    string
    Length     int
    Frequency  int
}
```

**Listing 5.6:** Action data structure

```
type Task struct {
    Id         string
    User       string
    Name       string
    Path       string
    Topic      string
    Status     string
    Current    int
    Total      int
    Actions    [] Action
}
```

**Listing 5.7:** Task request data structure

## Get Recordings

The Get Recording endpoint returns a list of recordings belonging to the authenticated user. So if a request had a recording id of another user, the request will fail. The request uses the RequestPayload structure, similar to how getVHs was implemented. The response of this endpoint is an array of strings.

## Download Record

The Download Record endpoint returns the request recording to the authenticated user. Just like Get Recording, the request for recordings of other users is impossible. The request uses the RequestPayload structure and the response is the requested file for the user to download. The web app converts the data into a CSV format and saves it on the user's device.

## 5.8 DATA INTEGRATION

The server will need to relay requests to the supervisor. As mentioned before there are designated channels to allow modules communicate with another, and in Golang there is a library called `streadway/amqp` that allows Golang to use AMQP protocol. Just like the supervisor and its nodes, the server also has its pair of clients: the consumer and producer.

### *Producer*

The producer is in charge of sending data into the designated channel, communicating with the supervisor. This allows the server to send data requests from the webapp using the `getData` and `createTask` endpoints. The data structure is the same one mentioned previously in the supervisor component.

### *Consumer*

The consumer is responsible for reading data blocks from the designated channel. There are two types of data blocks, data blocks generated by the IDE Tool and the blocks generated by the Task Page. The data structures used in this channel are the following.

```
type RabbitMQ_MSG struct {
    Data      [][][] float64
    Id        string
    Offset    int
    Emotion   string
}
```

**Listing 5.8:** IDE Tool data structure block

```

type RabbitMQ_Status struct {
    Data    [][][] float64
    Id      string
    Emotion string
    User    string
    Status  string
}

```

**Listing 5.9:** Task data structure block

## 5.9 DATA PROTECTION AND SECURITY

Data protection are mechanisms and procedures to ensure the privacy and integrity of the user's data. The data collected hold significant value to the organization and the user, so if the data were ever to be tampered with or accessed by third parties, the integrity dissipates and the user's trust is gone.

The datasets used, which holds no information of patients involved, within the system are completely inaccessible by the public. Datasets uploaded by users are also inaccessible but the respective can choose to remove them from the system. The system does not collect nor save users personal information and the segments uploaded from the users are not identifiable.

The records of virtual humans also follow the same philosophy where the authors are in control of their content. In this case with the records, the authors are allowed to view them and have them streamed to their projects through dynamic channels.

Each user has a token associated with their account, the generated string allows users to pull data through the messaging module without using the web application. The token helps the system identify the user that is acting on the messaging system and have them accountable for the action. Of course, the system strongly suggests not sharing the token.

User information, records location, and virtual human information are stored in the Database module. Sensitive data like passwords and tokens are encrypted in case a data breach occurs.





# Results and validation

The purpose of Digital Twins is to emulate a physical entity to a point where it can produce reactions to the current environment or state thus the data produced by the system must undergo several validations that properly meet the use case scenarios, from the usability of the web application to the easiness of integration in third-party projects.

The following chapter illustrates the tools and protocols used to validate the biological signals being produced by the Digital Twin, evaluate the usability of the web application with the target audience and integrate VIRHUS into third-party projects to help achieve their set of goals.

## 6.1 SIGNALS GENERATION VALIDATION

Even though producing biological signals is an important milestone to complete during the implementation phase, it is also important to understand if the very same synthetic signal contains a structure close enough to a real twin. There are two interesting hypotheses, regarding value and distance from the synthetic to the real signal, and the other questions about the quality over time. In other words, does the signal degrade over time? For each hypothesis, a strategy and protocol were designed to confirm the hypothesis and understand the nature of the signals.

### 6.1.1 Are the synthetic signals (biologically) realistic?

To prove the authenticity of the signals, a protocol of validation was used to calculate the distance between the real and data. Thus was required to isolate a real patient of the provided dataset from the others. Let's call the isolated patient ID10. Within the system, there were two approaches to producing synthetic data. First producing data using a model based on all of the patients, in other words, during the training process, the model considered the entire data to produce 4 unique models for each unique emotion, in these 12 unique models. This set of models was called generic models. The second approach was to use models based on a specific patient, producing similar signals based on that specific part of the dataset.

Name	ECG MAE	EMG MAE	EMGZ MAE	EDA MAE
GenF	0.0843	0.0787	0.112	0.825
GenH	0.0649	0.0614	0.128	1.436
GenN	0.066	0.0748	0.113	1.023

**Table 6.1:** Signal Realism scores of the Generic model using MAE metric

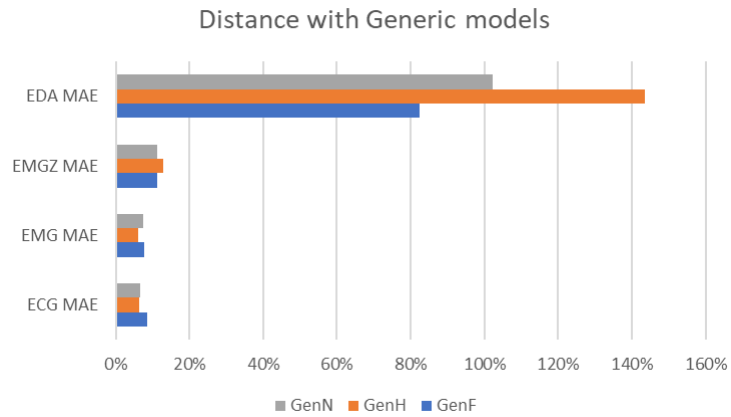
Name	ECG MSE	EMG MSE	EMGZ MSE	EDA MSE
GenF	0.0140	0.010	0.033	0.691
GenH	0.009	0.006	0.039	2.068
GenN	0.008	0.008	0.0319	1.0528

**Table 6.2:** Signal Realism scores of the Generic model using MSE metric

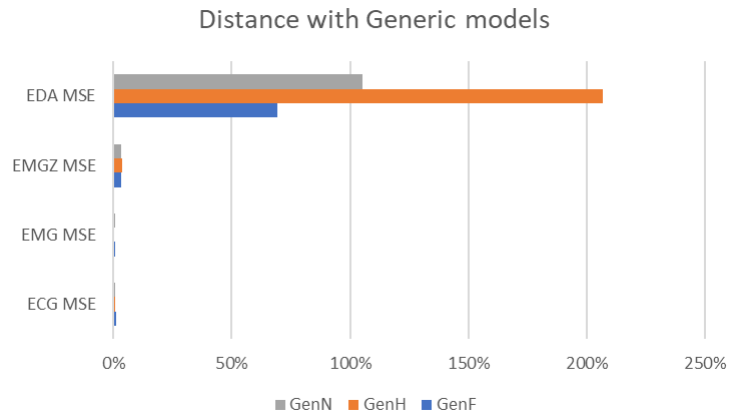
Validating both approaches will also answer the question if the aggregation of data provides more, less, or no significant value to the signal. The authenticity was proven undergoing the following protocol, the baseline of ID10 was used to produce a specific emotion (Happy, Fear, or Neutral) of 5 minutes and followed by comparing to ID10's emotional signal of 5minutes. The comparison used various error metrics in percentage to evaluate the distance between both signals. The higher the error was the further both signals were from each other.

The following table and graphs show a set of results for both the generic approach and the second approach.

The generic model, performance-wise, the model did roughly well providing ECG and EMG results below the 10% margin while EMGZ is slightly above.



**Figure 6.1:** MAE Generic model Results



**Figure 6.2:** MSE Generic model Results

Name	ECG MAE	EMG MAE	EMGZ MAE	EDA MAE
Fear	0.071	0.040	0.108	0.855
Happy	0.0627	0.056	0.129	1.453
Neutral	0.0665	0.085	0.125	1.043

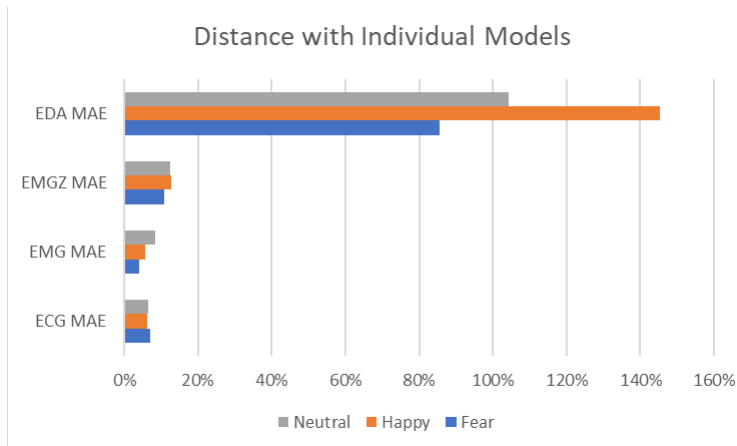
**Table 6.3:** Signal Realism scores of the Individual model using MAE metric

In the second approach with the individual model, the results were quite similar, especially with the EMGZ signal slightly crossing the 10% threshold.

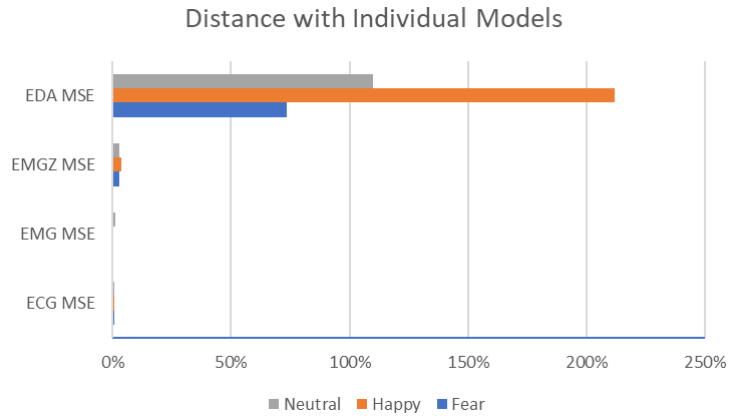
In both occurrences, it is easy to observe that predicting EDA signals is almost impossible producing error results surpassing the 80% threshold. Also, both show similar results regarding the other 3 signals, providing positive results. The error evaluation indicates that three of the tested signals were correctly modelled. The EDA signal presents higher error, which can be justified by the signal properties, or the amount of data used in the model definition.

Name	ECG MSE	EMG MSE	EMGZ MSE	EDA MSE
Fear	0.009	0.002	0.031	0.737
Happy	0.008	0.005	0.038	2.117
Neutral	0.007	0.012	0.031	1.100

**Table 6.4:** Signal Realism scores of the Individual model using MSE metric



**Figure 6.3:** MAE Individual model Results



**Figure 6.4:** MSE Individual model Results

### 6.1.2 Does the signal degrade over time?

To prove the existence of degradation within the synthetic signal, a new protocol was designed to validate and conclude the hypothesis. To do so, once again, the ID10 baseline was used to produce newly synthetic signals in six different lengths, 10 seconds, 60 seconds, 300 seconds, 600 seconds, 1200 seconds, and 3600 seconds. The values chosen were not based on any method and made it possible to observe the error distance over time. Of course, reintroducing both approaches, the generic model and individual model to see if training

Name	ECG MAE	EMG MAE	EMGZ MAE	EDA MAE
10s	0.079	0.041	0.115	0.840
60s	0.083	0.079	0.111	0.827
300s	0.075	0.039	0.108	0.855
600s	0.075	0.039	0.108	0.855
1200s	0.075	0.039	0.108	0.855
3600s	0.075	0.039	0.108	0.855

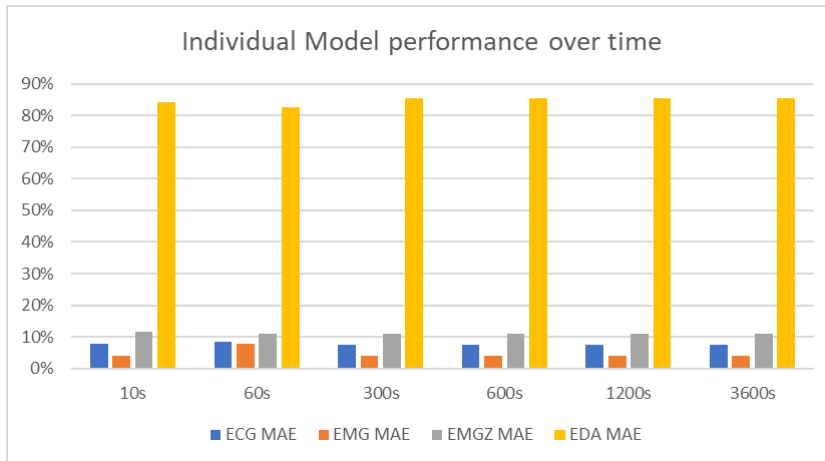
**Table 6.5:** Signal Degradation scores of the Individual model using MAE metric

Name	ECG MSE	EMG MSE	EMGZ MSE	EDA MSE
10s	0.012	0.002	0.031	0.708
60s	0.014	0.009	0.032	0.694
300s	0.011	0.002	0.030	0.736
600s	0.011	0.002	0.030	0.737
1200s	0.011	0.002	0.030	0.736
3600s	0.011	0.002	0.030	0.736

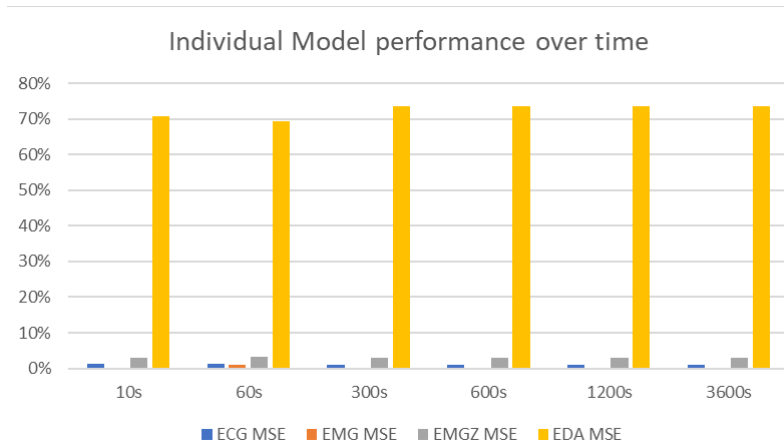
**Table 6.6:** Signal Degradation scores of the Individual model using MSE metric

models with the conjunction of a patient are better, worse, or insignificant over time. For comparisons, the same error metrics were used to evaluate the distance between the real to the synthetic signal.

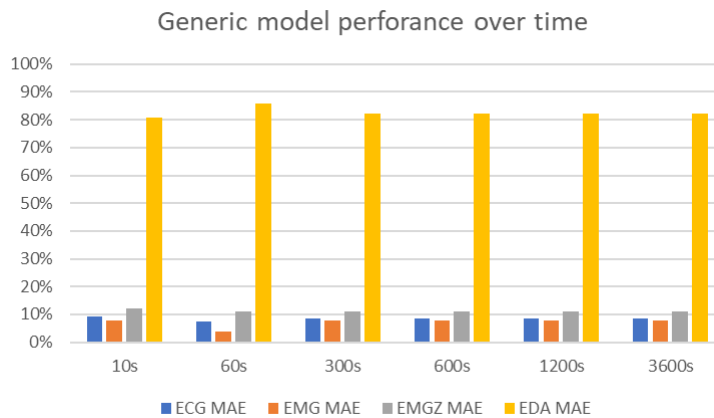
According to the results above, the signals maintains consistent throughout long periods of time, regardless if its generic or an individual model.



**Figure 6.5:** MAE Individual model Results over time



**Figure 6.6:** MSE Individual model Results over time



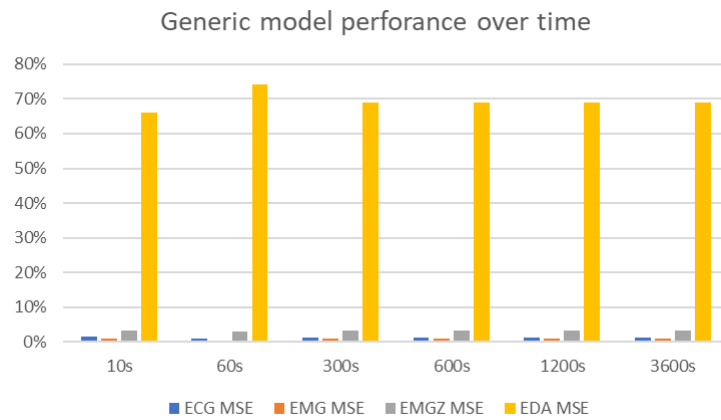
**Figure 6.7:** MAE Generic model Results over time

Name	ECG MAE	EMG MAE	EMGZ MAE	EDA MAE
10s	0.091	0.078	0.119	0.809
60s	0.075	0.039	0.109	0.858
300s	0.084	0.079	0.111	0.824
600s	0.084	0.079	0.111	0.824
1200s	0.084	0.079	0.111	0.824
3600s	0.084	0.079	0.111	0.824

**Table 6.7:** Signal Degradation scores of the Generic model using MAE metric

Name	ECG MSE	EMG MSE	EMGZ MSE	EDA MSE
10 seconds	0.015	0.009	0.034	0.661
60 seconds	0.011	0.002	0.030	0.741
300 seconds	0.014	0.009	0.032	0.689
600 seconds	0.014	0.009	0.032	0.689
1200 seconds	0.013	0.009	0.032	0.689
3600 seconds	0.014	0.009	0.032	0.689

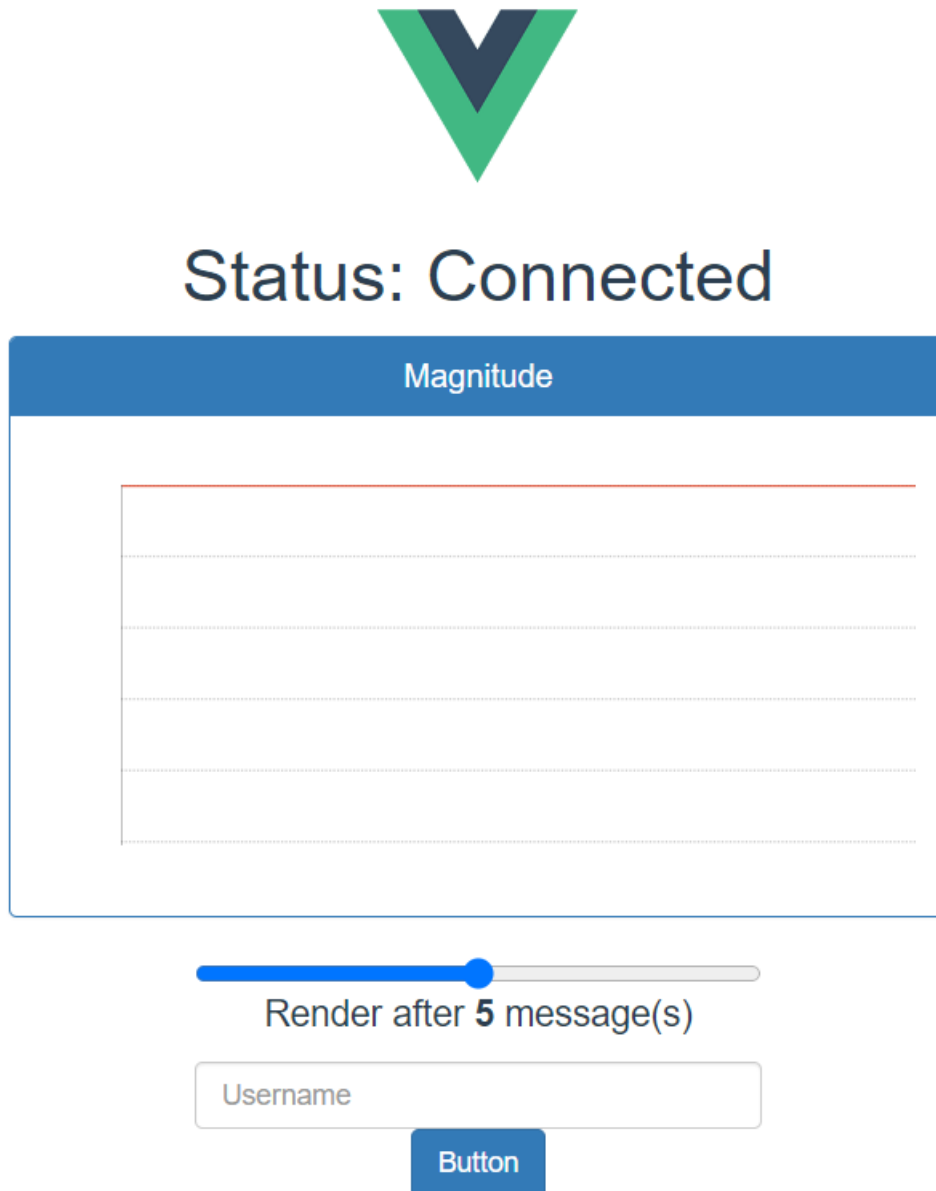
**Table 6.8:** Signal Degradation scores of the Generic model using MSE metric



**Figure 6.8:** MSE Generic model Results over time

## 6.2 INITIAL PROTOTYPE AND EXPERIMENTAL USE

Once the entire system was fully implemented and ready to be used, an additional effort was made to implement a simple graphical web application which serves as an example of how to integrate VIRHUS into external projects.



**Figure 6.9:** External Project

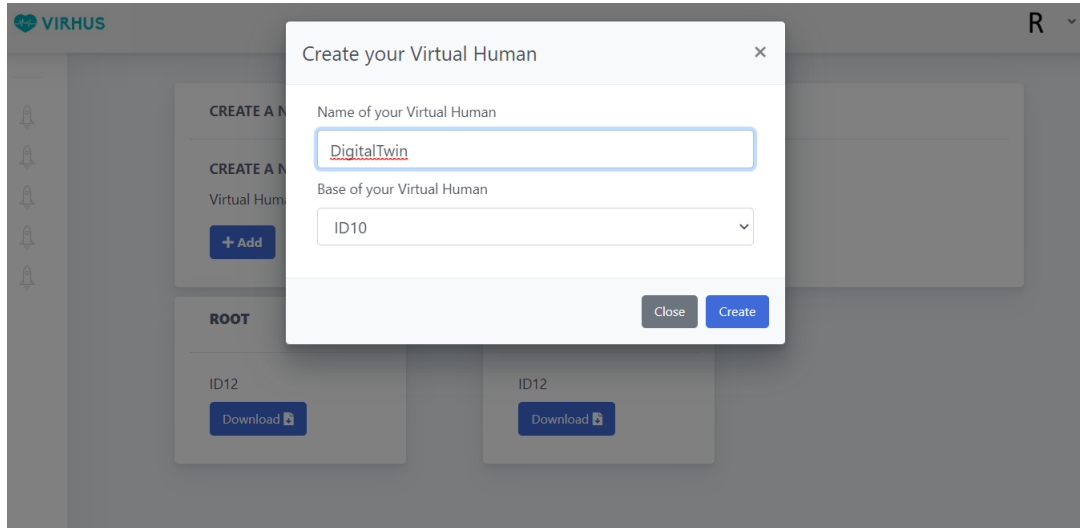
### 6.2.1 VIRHUS Integration

The design of this project is quite straightforward, with the primary goal to test out the real-time chart of Vue.js. For this to work, the target user will need to do the following steps:



### *Virtual Human Creation*

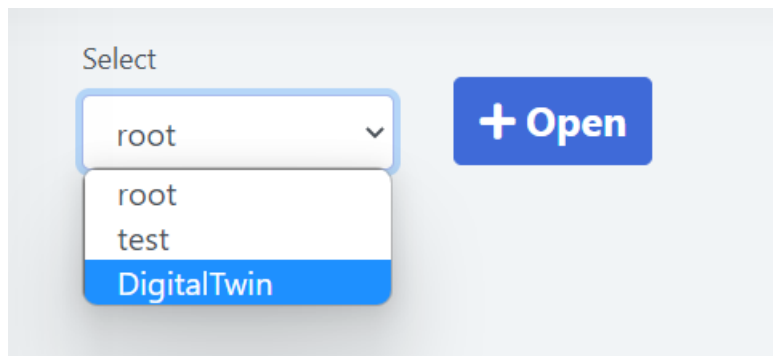
The user will need to head over to the virtual humans page on the VIRHUS website, click on the add button, and fill out the name of the desirable baseline to use. After hitting the create button, the user should be to see a new virtual human ready to be used.



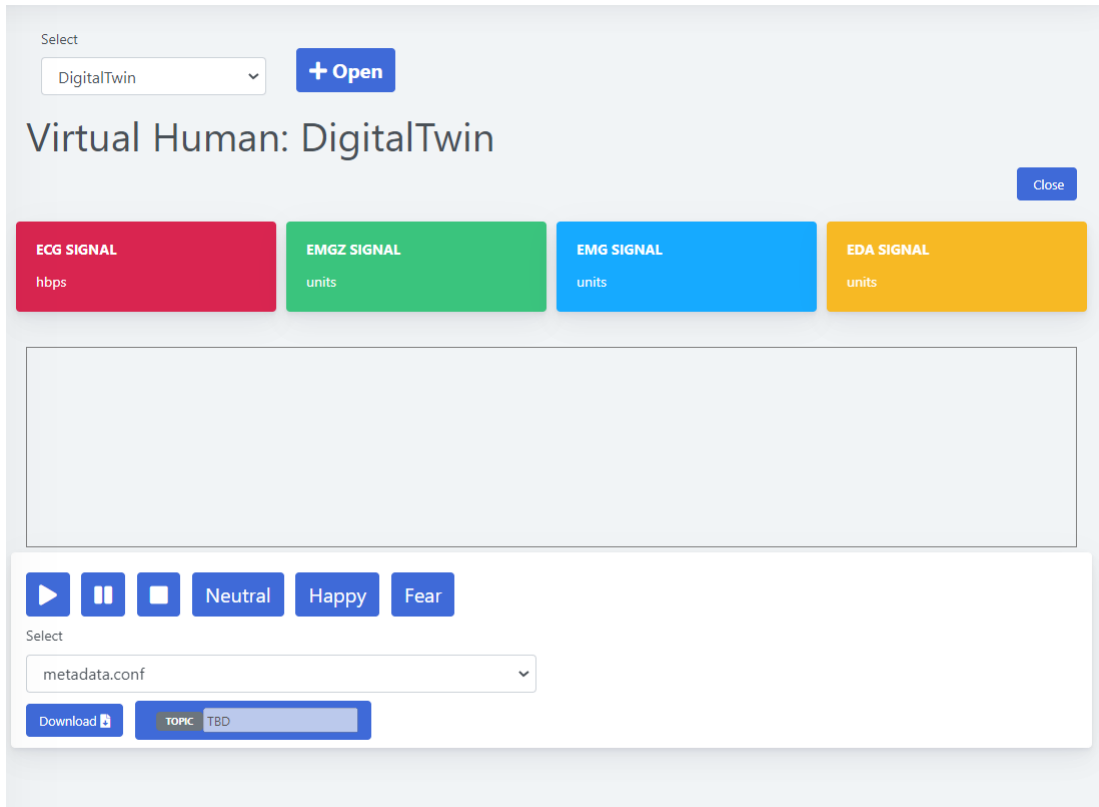
**Figure 6.10:** Virtual Human creation

### *Using the IDE Tool*

After creating the virtual human, the user can open it on the IDE Tool page by selecting from the drop-down the newly created virtual human and clicking open. This will open a graphical interface showing in real-time the Virtual human operating. Upon hitting play, the virtual human will start producing biological signals, this data can be accessed via recording or through the topic that appears once the user presses play.



**Figure 6.11:** Virtual Human dropdown



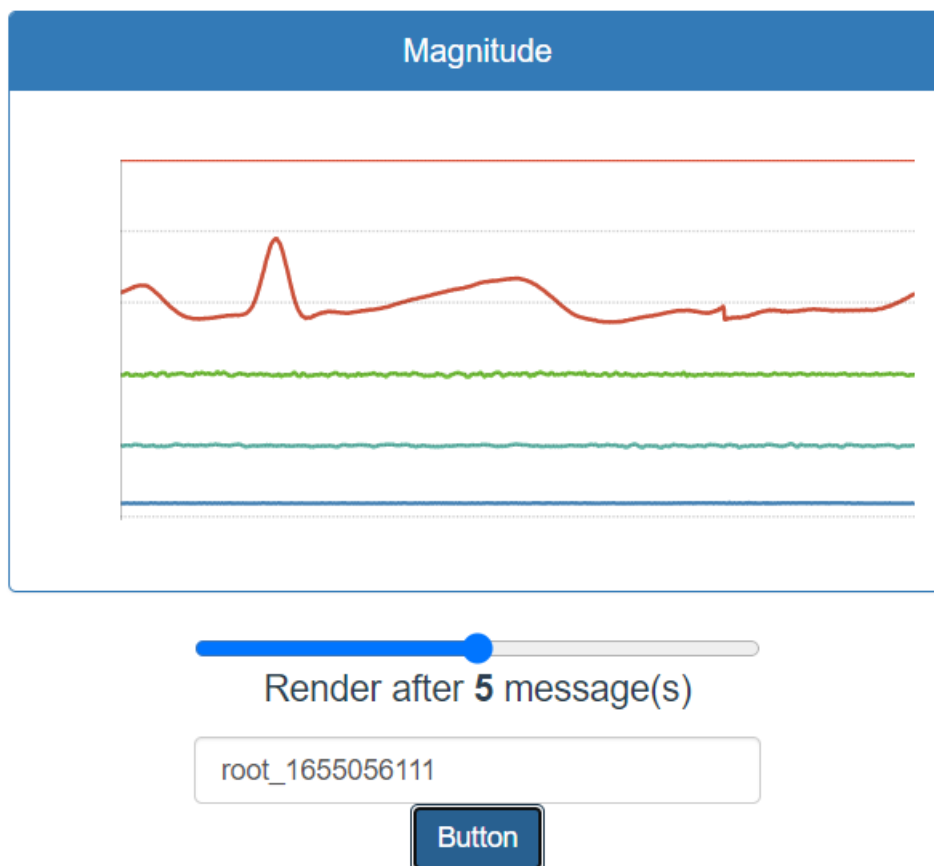
**Figure 6.12:** IDE Tool Virtual Human

### *External Project*

The external project is a web application in VueJs running on Nodejs. This allowed the project to create an AMQP client using existing libraries of Nodejs to fetch the biological signals of the virtual human. In the previous section, the topic printed in the IDE Tool will be used to connect and fetch in real-time the virtual human's signals and display them on the VueJs time series chart, demonstrating the simplicity it is to integrating VIRHUS into a basic project without excessive libraries or software.



# Status: Connected



**Figure 6.13:** External Project running

### 6.2.2 Personalizing new emotions

The system is also prepared to produce new signals with newly introduced emotions by the user. This approach allows users to create new case studies with virtual humans. To do so, the user would need to upload the dataset to the system, have it trained and perform task requests on it.

#### *Upload new data*

Uploading data is quite simple, users would need to prepare a CSV that represents the emotion or condition in which the signal encounters. This can be done on the Signal page on

the VIRHUS web application by clicking the upload button. The user would need to provide a name, condition, sample frequency, and the CSV file.

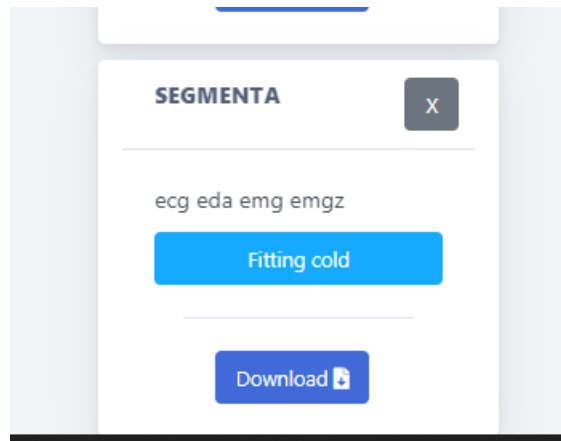
The image shows a modal dialog box titled "Upload biological signals" with a close button (X) in the top right corner. The dialog contains the following fields and text:

- Name of the signals:** A text input field containing "segmentA".
- Signal Deviation:** A text input field containing "cold".
- Privacy notice:** "We'll never share your email with anyone else."
- Sample Frequency:** A text input field containing "1000".
- Frequency in Hz of this Task:** A label below the sample frequency field.
- Default file input example:** A text input field containing "Choose File 5".
- Buttons:** "Close" (grey) and "Upload" (blue) buttons at the bottom right.

**Figure 6.14:** Uploading a CSV

### *Training model*

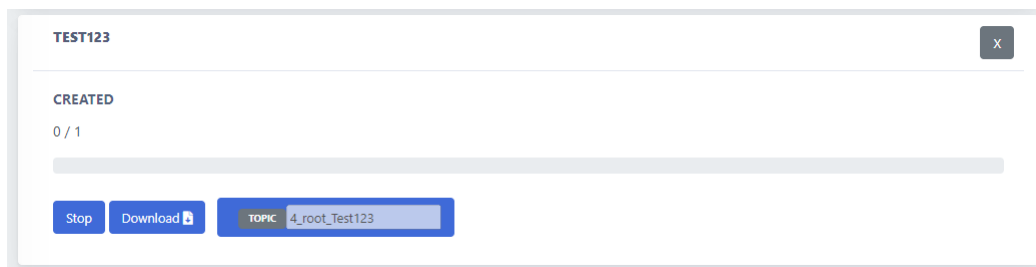
Training a model can be done on the signal page by clicking the train button on the desired signal. This operation will launch a process that will conduct the training and creation of the baseline component. Users can train multiple signals at once and train the same signal multiple times.



**Figure 6.15:** Training Signal

### *Using the Task feature*

The IDE tool is limited to producing signals under generic conditions and serves as a live demonstration to the user. The Task page, on the other hand, allows the creation of signals under any existing condition, be the newly created ones or generic conditions, and defines the length in seconds of the signal. Internally, the system will assign the request to an available worker and relay the results to the designated topic which will appear on the task page right next to the task.



**Figure 6.16:** New Task

## 6.3 USABILITY EVALUATION

It is important to understand if the web application meets the target user's standards in terms of usability. Thus a usability evaluation is required to assess any flaws within the web application's interface. The goal behind every usability test is to figure out potential changes according to how the user perceives the system. Is the terminology used correct, are the icons appropriate to the operation, are there too many clicks, or does the user get lost at a certain point?

The usability evaluation will carry out through in-person or online sessions where the evaluator will be present to help and record the session. Users will be given a consent form to fill out, a spreadsheet of tasks, and two post questionnaires. The evaluator sometimes referred to as the observer, will have an observation spreadsheet to record any significant responses or

questions the user might have during the session, record the time of each task, and attribute a level of difficulty according to the user. The following sections will discuss the tasks involved, the overall results, and the conclusion plus changes that were made.

### 6.3.1 Sample

The participants chosen for the usability experience were students from the University of Aveiro, studying software engineering, bioinformatics, or a medical course. Of the numerous applicants, only eleven were able to show up and participated. The eleven participants were between 19 and 27 years old and on average 23.4 years old. Nine of the students were software engineers, 1 from the medical course and another studying bioinformatics. The nine software engineers were familiar with the usability evaluation while the other two were not.

### 6.3.2 Method

In a real-life scenario, the user will start their experience with the system through the web application, creating virtual humans, uploading data and train virtual humans. The goal of the sessions is to produce these experiences and have the users relay to us their feedback on the system from various perspectives. The setup of each session is quite simple, all the user would need is their device with an internet connection to access the VIRHUS website. Since some of the users did not have a device ready or the device they had was inadequate, we allowed users to perform the session on a borrowed computer that contained the necessary material.

Each user was given a consent form and a 5minute explanation of what VIRHUS is about. Once the user was ready to begin, the participants were told to navigate to the VIRHUS website and explore the user interface for five minutes. During this time they were told to think aloud so that the observer could take notes during the event.

1. After exploring the website, the participant was allowed to ask any questions regarding their self-tour around the web application.
2. They were then prompted to create a virtual human with the given name and baseline. After they were told to search for the virtual human.
3. The following task required them to upload raw signals to the system. Here a file was prepared for them. After uploading they were asked to look for the uploaded signal and train a new model on top. They were asked if they could see if the model was being trained.
4. In the following task, they were told to create another virtual human, but this time using the baseline created by the training operation of the previous task.
5. After that, the next task told them to open the IDE Tool, search for their Virtual human of the previous task, and begin interacting with the interface. Pressing play, pause and stop. Changing the current emotion and downloading the current experience.
6. Next they were prompted to create a Task request in the Task user interface. Creating requests of a certain length, baseline, model, and sample frequency. Have them wait and download the result.

7. They were then asked to download the virtual human and asked to open the zip file and tell the observer that they were looking at it.
8. Finally, the participant was told to delete their entire progress, virtual humans, tasks, models, and signals.

Throughout the sessions, users were constantly asked or prompted to think out aloud which produced a lot of feedback. The following chapter illustrates the results, conclusions, and changes to the web application.

### 6.3.3 Questionnaire Results

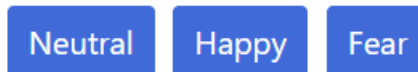
In all eleven sessions, the participants were able to complete all eight tasks that were given with an overall average time of four minutes and 18 seconds. Participants spent most of their time on both interfaces, IDE Tool, and Tasks, and both scored the lowest satisfaction score throughout the entire evaluation. The results of the other pages were quite similar to one another with a couple of minor changes required.

Task	User difficulty				Task Completion Time	N <sup>o</sup> needed help
	V. Easy	Easy	Mild	Difficult		
Task 2	3	8	0	0	14.44 seconds	0
Task 3	3	7	1	-	0	0
Task 4	3	8	0	0	16.23 seconds	0
Task 5	0	0	9	2	02:20.4	4
Task 6	0	4	7	0	1:13	2
Task 7	2	9	0	0	15.33	0
Task 8	8	3	0	0	13.2	0

**Table 6.9:** Usability test results

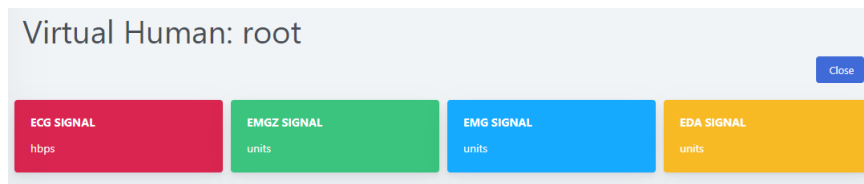
### IDE Tool interface

Task five talks about interacting with the IDE interface, telling the participant to create an instance of a virtual human and have it switch emotion to emotion. In parallel, they were told to explain what they were observing to the observer and point out any differences when changing the virtual human's emotion. As result, only a few participants were lost and/or could not explain what they were looking at. One of them kept clicking on the same emotion thinking the system was producing a different one, while another said that they spotted no difference upon changing between emotions.



**Figure 6.17:** Emotions with no indicator

The problem to why this task was rated the lowest was due to its visual feedback to tell the user which emotion was being produced in real-time. Another issue is the graphical interpretation, in other words, the signals being displayed on the screen. Participants were able to identify them as biological signals but the majority were only able to point out what ECG was, while the other three were beyond their knowledge.



**Figure 6.18:** Signals in text form

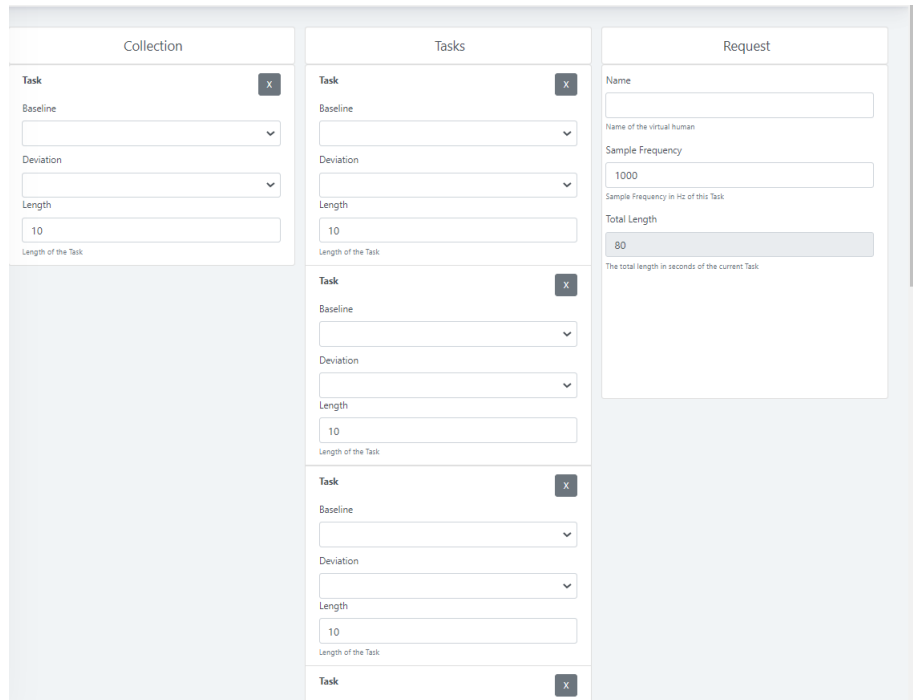
Concluding the IDE Tool, two changes were made to further improve the user experience:

- Providing visual feedback of the current emotion, participants immediately lost track of which emotion was being played after a couple of changes leading the user to an inconsistent perception of the signal being produced.
- Adding visual icons or images which request the biological signal. For example, the ECG would have a heart figure icon or image and EMG a muscle-like icon. In the future, users could associate signals with custom icons, showing a physical representation of the signal that they plan on uploading.

### Task interface

On task six, users were told to create a request using the signal baseline that they uploaded in a previous task. Upon completion, participants look satisfied with how fast the response of the system was to their task and urge to repeat the request but with more creativity. Unfortunately, after adding multiple snippets to the request, the user interface overflowed the page and made it difficult to organize a long list of snippets.





**Figure 6.19:** Boxes clipping out

In effort to remake the page with the idea to create lengthy signals while away from the system, the participants were asked how would they put a time series of signals together. The following are the summarized suggestions:

- The majority referenced a popular audio editor as a way to build signals with snippets. Maintaining the drag and drop idea, the left-hand sign would contain all of the baselines. Once a baseline was dragged into the middle of the screen, users could stretch or condense the snippet. Once the signal was out of frame, users could zoom in and out.
- Others suggested this be done through input forms, displaying a table and the rows would represent the snippets.

Conclusion the Task interface, and the way how signals are designed need to be changed, including more usable features like zooming in and out, less information in the snippets, and more. Given the cost and effort in making the Task interface, these changes would require more time and thus will remain as future work in this dissertation.

### *Assets Pages*

The results of the other pages were similar, the pages referenced in this topic are the library, signals, and virtual humans. Overall the participants found working with these pages simple and easy to use. The goal behind them is to show the user the current assets and the respective states of each asset in the system. During the usability test, and few minor problems were encountered. The first was the wording baseline, most participants were not aware of what this terminology meant and its role within the system. Another was when they were asked to download the virtual human and were confused when they got the metadata.

This occurred in task seven when participants were asked to tell the observers what they were looking at, and all of them said that they were expecting the records and the corresponding models.

In the post usability test, participants were given two questionnaires, the questionnaire for User Interface Satisfaction and the System Usability Scale. The goal behind both questionnaires is to gather the participant's opinions about the cosmetics used and reflect on a list of statements.

#### *Questionnaire for User Interface Satisfaction*

The questionnaire for User Interface Satisfaction also referred to as QUIS, is intended to gather the participant's perspective. The questionnaire is composed of a list of short and direct questions, each containing a pair of antithetical adjectives serving as the spectrum.

Through the questionnaire, participants pointed out that:

- System feedback was poor.
- There was no highlighting on the screen simplifying the task.
- System messages and reports were poor.
- Overall Reactions to the software were dull.

Regarding the system feedback, participants felt that their interactions were not properly represented during the usability test. A good example of this was the emotion button in the IDE Tool, users had to remember what emotion was being played instead of a color indicator. Participants felt that the application lacked system messages telling them when a certain task was done, for example training a model or completing a segment. Instead, users had to constantly refresh to page to see if the task was done.

Attribute	Scale	Rounded Result
Characters on the computer screen	hard to read 0 1 2 3 4 5 6 7 8 9 easy to read	7
Highlighting on the screen simplifies task	not at all 0 1 2 3 4 5 6 7 8 9 very much	0
Organizing of information on screen	confusing 0 1 2 3 4 5 6 7 8 9 very clear	8
Usability and User Interface		
Use of colors and sounds	poor 0 1 2 3 4 5 6 7 8 9 good	4
System feedback	poor 0 1 2 3 4 5 6 7 8 9 good	2
System messages and reports	poor 0 1 2 3 4 5 6 7 8 9 good	3
System response to errors	awkward 0 1 2 3 4 5 6 7 8 9 gracious	7
System clutter and UI noise	poor 0 1 2 3 4 5 6 7 8 9 good	8
Terminology and System Information		
Use of terms throughout system	inconsistent 0 1 2 3 4 5 6 7 8 9 consistent	7
Computer terminology is related to the task you are doing	never 0 1 2 3 4 5 6 7 8 9 always	5
Position of messages on screen	inconsistent 0 1 2 3 4 5 6 7 8 9 consistent	9
Messages on screen which prompt user for input	confusing 0 1 2 3 4 5 6 7 8 9 clear	9
Computer keeps you informed about what it is doing	never 0 1 2 3 4 5 6 7 8 9 always	5
Error messages	unhelpful 0 1 2 3 4 5 6 7 8 9 helpful	7

**Table 6.10:** Questionnaire for User Interface Satisfaction Results 1

Attribute	Scale	Rounded Result
Learning to operate the system	difficult 0 1 2 3 4 5 6 7 8 9 easy	7
Exploring new features by trial and error	difficult 0 1 2 3 4 5 6 7 8 9 easy	2
Remembering names and use of commands	difficult 0 1 2 3 4 5 6 7 8 9 easy	7
Tasks can be performed in a straight-forward manner	never 0 1 2 3 4 5 6 7 8 9 always	8
Help messages on the screen	unhelpful 0 1 2 3 4 5 6 7 8 9 helpful	8
Supplemental reference materials	confusing 0 1 2 3 4 5 6 7 8 9 clear	7
Overall Reactions to the software	terrible 0 1 2 3 4 5 6 7 8 9 wonderful	6
Overall Reactions to the software	difficult 0 1 2 3 4 5 6 7 8 9 easy	9
Overall Reactions to the software	frustrating 0 1 2 3 4 5 6 7 8 9 satisfying	7
Overall Reactions to the software	dull 0 1 2 3 4 5 6 7 8 9 stimulating	3
Overall Reactions to the software	rigid 0 1 2 3 4 5 6 7 8 9 flexible	7

**Table 6.11:** Questionnaire for User Interface Satisfaction Results 2

### *System Usability Scale.*

The System Usability Scale, also referred to as the SUS questionnaire[29], is used to gather the participant's opinions[29]. The way it works, each participant has ten different statements/affirmations to agree or disagree with. These statements are based on the overall satisfaction after using the application. Each statement has a contribution score, which is summed up and multiplied by 2.5[29]. The value represents the overall SUS score of the application.

The majority of participants agreed upon:

- Most people would learn to use this website every quickly
- This website is easy to use.
- The website is not unnecessarily complex.

Overall the results of the SUS questionnaire[30] did appear during the usability test, on part of several observations regarding the speed and timing of completing tasks. The opinion and numbers do align when it comes to the simplicity of the system.

Statement	Rounded Result	SUS Contribution
I needed to learn a lot of things before I could get going with this website.	3	3
I think that I would like use this website frequently.	3	1
I felt very confident using this website.	2	0
I found this website unnecessarily complex.	1	5
I found this website very cumbersome/awkward to use.	2	4
I think that I would need assistance to be able to use this website.	2	4
I found the various functions in this website were well integrated.	4	2
I would imagine that most people would learn to use this website every quickly	5	3
I thought there was too much inconsistency in this website	3	3
I believe this website is easy to use.	4	2
SUS Overall Score		67.5

**Table 6.12:** System Usability Scale Results



# Conclusions

Digital twins have been applied as a software concept to model physical entities. Usually digital twins imply the acquisition of the current state of the entity, a model of its behavior some reasoning about future conditions. In this dissertation project we looked into digital twins as an abstraction to model physiological signals of virtual humans. The proposed system VIRHUS was able to work with existing signal samples as "seeds" to generate synthetic signals.

## 7.1 WORK DEVELOPED

The system, VIRHUS, was complex, considering the stack implemented and the dataset used. Many questions arose for example, how could a patient's signals be predicted under the influence of a different emotion and how could that be measured to an acceptable degree, how can users be able to integrate the system into their own projects, how will data flow throughout the system, what the best approach, the dataset provided needs to be changed. After the research phase of this project, I came to many conclusions how the system should look like and how other projects can benefit VIRHUS externally. When came down to forecasting segments, autoencoders was the ideal approach because the models that were trained previously would hold the influence emotion and all the system had to do was predict a new segment with a provided baseline. From these observations, pages like Signals, Library and Virtual humans were required to show the user assets that are available.

The sample data (required to prepare the generative model ) provided came from my supervisor's previous case study regarding human emotions, was well complete. I was able to understand what a segment looks like, the meaning behind a baseline and picture how the end result should be to the user. Of course before training the models, I had to readjust the data, removing the baseline and noise.

From a developer point of view, I learnt a lot while implementing VIRHUS. The technical stack used was quite challenging to me, especially with the main server been written in Golang. The reason behind it was due to the research and curiosity I had at the time and also a great opportunity to learn another programming language. I ran into difficulties while

developing VIRHUS, especially the interactive design of the IDE and Task page. Mainly due to the drag and drop feature, where users could append endless amount of snippets. The idea behind this interface was how the web application was going to generate dynamic boxes while maintaining the original one and show on the right the current values. I also wanted users with no experience in bio informatics understand what a segment looks like and how one can create a segment. Submitting requests and receiving a CSV was not enough, so I went beyond and used Vuejs to present a real-time the signals and let the user be able to manipulate the emotions. The difficult part here was when the IDE needed more data, which would take time to generate and send, so a workaround was implemented to show the user that system is working by sending arrays of zeros to the application.

Overall, VIRHUS was able to achieve parts of the Digital Twins concept by providing a prediction of segments of real patients under a specific condition and adapting the models used in the predictions. Another important goal achieved was the ability to integrate VIRHUS into third party projects. This allows students to work on mobile/web applications to generate biological signals for their own use.

## 7.2 FUTURE WORK

During the research phase, I came across several papers talking about Generative Adversary Networks. This approach was at first, the path to go down as the results shown in the papers were quite impressive. After many attempts of understanding GANs, I decided to stick with auto encoders. In future iteration, would be interesting to incorporate GANs as the main segment generator and see its performance versus auto encoders. Regarding the process of creating models, it would be interesting to research a different approach in building a data set, in this case study, the dataset was broken down into heartbeat segments, which allowed NeuroKit to filter out noisy or incorrect heartbeats.

The example provided in this project used a RabbitMQ client to communicate directly to the messaging server. As future reference, a different approach to integrating VIRHUS would be interesting. For example, have the data throttle through web sockets or https requests to support platforms that can not use the client.

The web application presented in this dissertation has two interactive tools, the IDE and Task pages. At the moment, the pages according to the usability tests are perceived dull and required a better immersive experience. In the future, it would be interesting to lay out a completely different design for the task page, maybe use the participants suggestions as the requirements to the new design. For the IDE Tool, introduce animated objects within the scene to highlight the signals meanings.



# References

- [1] S. Khan and T. Yairi, "A review on the application of deep learning in system health management," *Mechanical Systems and Signal Processing*, vol. 107, pp. 241–265, 2018, ISSN: 0888-3270. DOI: <https://doi.org/10.1016/j.ymssp.2017.11.024>.
- [2] A. K. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, 2006, ISSN: 0888-3270. DOI: <https://doi.org/10.1016/j.ymssp.2005.09.012>.
- [3] W. Booyse, D. N. Wilke, and S. Heyns, "Deep digital twins for detection, diagnostics and prognostics," *Mechanical Systems and Signal Processing*, vol. 140, p. 106612, 2020, ISSN: 0888-3270. DOI: <https://doi.org/10.1016/j.ymssp.2019.106612>.
- [4] N. Pugno, M. Ciavarella, P. Cornetti, and A. Carpinteri, "A generalized paris' law for fatigue crack growth," *Journal of the Mechanics and Physics of Solids*, vol. 54, no. 7, pp. 1333–1349, 2006, ISSN: 0022-5096. DOI: <https://doi.org/10.1016/j.jmps.2006.01.007>.
- [5] L. F. Rivera, M. Jiménez, P. Angara, N. M. Villegas, G. Tamura, and H. A. Müller, "Towards continuous monitoring in personalized healthcare through digital twins," in *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '19, Toronto, Ontario, Canada: IBM Corp., 2019, pp. 329–335.
- [6] M. Ghita, S. Benhadou, and M. Hicham, "Digital twins development architectures and deployment technologies: Moroccan use case," *International Journal of Advanced Computer Science and Applications*, vol. 11, Jan. 2020. DOI: [10.14569/IJACSA.2020.0110260](https://doi.org/10.14569/IJACSA.2020.0110260).
- [7] R. Lutze, "Digital twin based software design in ehealth - a new development approach for health / medical software products," Jun. 2020. DOI: [10.1109/ICE/ITMC49519.2020.9198546](https://doi.org/10.1109/ICE/ITMC49519.2020.9198546).
- [8] J. Trauer, S. Schweigert-Recksiek, C. Engel, K. Spreitzer, and M. Zimmermann, "What is a digital twin? – definitions and insights from an industrial case study in technical product development," *Proceedings of the Design Society: DESIGN Conference*, vol. 1, pp. 757–766, May 2020. DOI: [10.1017/dsd.2020.15](https://doi.org/10.1017/dsd.2020.15).
- [9] A. Croatti, M. Gabellini, S. Montagna, and A. Ricci, "On the integration of agents and digital twins in healthcare," *Journal of Medical Systems*, vol. 44, p. 161, Aug. 2020. DOI: [10.1007/s10916-020-01623-5](https://doi.org/10.1007/s10916-020-01623-5).
- [10] B. Barricelli, E. Casiraghi, J. Gliozzo, A. Petrini, and S. Valtolina, "Human digital twin for fitness management," *IEEE Access*, vol. PP, pp. 1–1, Feb. 2020. DOI: [10.1109/ACCESS.2020.2971576](https://doi.org/10.1109/ACCESS.2020.2971576).
- [11] K. Bruynseels, F. Santoni de Sio, and J. van den hoven, "Digital twins in health care: Ethical implications of an emerging engineering paradigm," *Frontiers in Genetics*, vol. 9, Feb. 2018. DOI: [10.3389/fgene.2018.00031](https://doi.org/10.3389/fgene.2018.00031).
- [12] C. Scheuermann, T. Binderberger, N. von Frankenberg, and A. Dr. Werner, "Digital twin -a machine learning approach to predict individual stress levels in extreme environments," Sep. 2020. DOI: [10.1145/3410530.3414316](https://doi.org/10.1145/3410530.3414316).
- [13] N. Schneiderman, G. Ironson, and S. Siegel, "Stress and health: Psychological, behavioral, and biological determinants," *Annual review of clinical psychology*, vol. 1, pp. 607–28, Feb. 2005. DOI: [10.1146/annurev.clinpsy.1.102803.144141](https://doi.org/10.1146/annurev.clinpsy.1.102803.144141).
- [14] N. Wulan, W. Wang, P. Sun, K. Wang, Y. Xia, and H. Zhang, "Generating electrocardiogram signals by deep learning," *Neurocomputing*, vol. 404, May 2020. DOI: [10.1016/j.neucom.2020.04.076](https://doi.org/10.1016/j.neucom.2020.04.076).

- [15] P. Morgan, R. Callister, C. Collins, R. Plotnikoff, M. Young, N. Berry, P. McElduff, T. Burrows, E. Aguiar, and K. Saunders, “The shed-it community trial: A randomized controlled trial of internet- and paper-based weight loss programs tailored for overweight and obese men,” *Annals of behavioral medicine : a publication of the Society of Behavioral Medicine*, vol. 6, Nov. 2012. DOI: 10.1007/s12160-012-9424-z.
- [16] T. Hastie, R. Tibshirani, G. Sherlock, M. Eisen, P. Brown, and D. Botstein, “Imputing missing data for gene expression arrays,” *Technical report, Stanford Statistics Department*, vol. 1, Dec. 2001.
- [17] X. Hou, L. Shen, K. Sun, and G. Qiu, “Deep feature consistent variational autoencoder,” Mar. 2017, pp. 1133–1141. DOI: 10.1109/WACV.2017.131.
- [18] D. Kingma and M. Welling, “Auto-encoding variational bayes,” Dec. 2014.
- [19] W. Fedus, M. Rosca, B. Lakshminarayanan, A. Dai, S. Mohamed, and I. Goodfellow, “Many paths to equilibrium: Gans do not need to decrease adivergence at every step,” Oct. 2017.
- [20] S. Schmidt, “A cost-effective diagnostic methodology using probabilistic approaches for gearboxes operating under non-stationary conditions,” 2016.
- [21] V. Kovenko and I. Bogach, “A comprehensive study of autoencoders’ applications related to images,” in *IT&I Workshops*, 2020.
- [22] E. Academy, “Applied deep learning - part 3: Autoencoders,” 2022. [Online]. Available: <https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798>.
- [23] K. Wang, C. Gou, Y. Duan, L. Yilun, X. Zheng, and F.-Y. Wang, “Generative adversarial networks: Introduction and outlook,” vol. 4, pp. 588–598, Sep. 2017. DOI: 10.1109/JAS.2017.7510583.
- [24] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, “Logan: Evaluating privacy leakage of generative models using generative adversarial networks,” May 2017.
- [25] Y. Luo, “Eeg data augmentation for emotion recognition using a conditional wasserstein gan,” vol. 2018, Jul. 2018, pp. 2535–2538. DOI: 10.1109/EMBC.2018.8512865.
- [26] P. Ebner and A. Eltelt, “Audio inpainting with generative adversarial network,” Mar. 2020.
- [27] N. Wulan, W. Wang, P. Sun, K. Wang, Y. Xia, and H. Zhang, “Generating electrocardiogram signals by deep learning,” *Neurocomputing*, vol. 404, May 2020. DOI: 10.1016/j.neucom.2020.04.076.
- [28] A. oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” Sep. 2016.
- [29] P. Harper and K. Norman, “Improving user satisfaction: The questionnaire for user interaction satisfaction version 5.5,” Jan. 1993.
- [30] J. Brooke, “Sus: A quick and dirty usability scale,” *Usability Eval. Ind.*, vol. 189, Nov. 1995.

# Appendix A Usability Tasks

Task 1	Exploring the Interface Go to url <a href="http://www.virhus.pt">www.virhus.pt</a> and spend 5mins exploring the website. Please use the think aloud protocol while exploring	Difficult	[1] [2] [3] [4] [5]	Easy
Task 2	Create a Virtual Human Go to url <a href="http://www.virhus.pt/virtualhumans">www.virhus.pt/virtualhumans</a> Click on add in the welcome message area to create a virtual human Give it a name and choose the available baselines Click create.	Difficult	[1] [2] [3] [4] [5]	Easy
Task 3	Creating a model Go to the Signals page and upload the file given to you. Give it a name, signal deviation will be fear, 1000 Sample frequency and select the file from your local device. Click upload. Look for the signal and train the model. Optional Download the model. Go to the library and see if the model is there.	Difficult	[1] [2] [3] [4] [5]	Easy
Task 4	Create another virtual Human Go to Virtual Humans and create a new virtual human Type in a new name and select the newest baseline that was created. Click create.	Difficult	[1] [2] [3] [4] [5]	Easy
Task 5	Interact with IDE interface Go to the IDE page and select the newest virtual human you have created. Click Open. Hit the play button and the observer whats happening. Click on the other emotions, does it change anything? Stop the simulation and download the file.	Difficult	[1] [2] [3] [4] [5]	Easy
Task 6	Interacting with the Task Interface Go to the Task and create a new task with same baseline used previously. Name the task, leave it at 1000 Sample Frequency and click save. Once the task has ended, click download	Difficult	[1] [2] [3] [4] [5]	Easy
Task 7	Download the virtual human Go to virtual humans and download the newest virtual human Tell the observers what you have downloaded	Difficult	[1] [2] [3] [4] [5]	Easy
Task 8	Delete progress Go to virtual humans and delete the virtual humans created by you. Go to Tasks and delete the newest task created by you. Go to Library delete the trained model. Go to Signals and delete the upload signals.	Difficult	[1] [2] [3] [4] [5]	Easy

Figure 1: Usability Tasks



# Appendix B Observer Spreadsheet

Tasks	Nº clicks	Completed the task?	Time of execution mm:ss	Made mistakes?	Felt lost?	Asked for help?	Difficulty level
1		yes[ ] no[ ]	__:__	no[ ] few[ ] a lot[ ]	no[ ] a bit[ ] a lot[ ]	no[ ] yes[ ] where?	[1] [2] [3] [4] [5]
2		yes[ ] no[ ]	__:__	no[ ] few[ ] a lot[ ]	no[ ] a bit[ ] a lot[ ]	no[ ] yes[ ] where?	[1] [2] [3] [4] [5]
3		yes[ ] no[ ]	__:__	no[ ] few[ ] a lot[ ]	no[ ] a bit[ ] a lot[ ]	no[ ] yes[ ] where?	[1] [2] [3] [4] [5]
4		yes[ ] no[ ]	__:__	no[ ] few[ ] a lot[ ]	no[ ] a bit[ ] a lot[ ]	no[ ] yes[ ] where?	[1] [2] [3] [4] [5]
5		yes[ ] no[ ]	__:__	no[ ] few[ ] a lot[ ]	no[ ] a bit[ ] a lot[ ]	no[ ] yes[ ] where?	[1] [2] [3] [4] [5]
6		yes[ ] no[ ]	__:__	no[ ] few[ ] a lot[ ]	no[ ] a bit[ ] a lot[ ]	no[ ] yes[ ] where?	[1] [2] [3] [4] [5]
7		yes[ ] no[ ]	__:__	no[ ] few[ ] a lot[ ]	no[ ] a bit[ ] a lot[ ]	no[ ] yes[ ] where?	[1] [2] [3] [4] [5]
8		yes[ ] no[ ]	__:__	no[ ] few[ ] a lot[ ]	no[ ] a bit[ ] a lot[ ]	no[ ] yes[ ] where?	[1] [2] [3] [4] [5]

**Figure 2:** Observer Spreadsheet



# Appendix C SUS Questionnaire

		Strongly Disagree				Strongly Agree			
1	I needed to learn a lot of things before I could get going with this website.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	I think that I would like use this website frequently.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	I felt very confident using this website.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	I found this website unnecessarily complex.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	I found this website very cumbersome/awkward to use.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	I think that I would need assistance to be able to use this website.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7	I found the various functions in this website were well intergrated.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8	I would imagine that most people would learn to use this website every quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9	I thought there was too much inconsistency in this website	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
10	I believe this website is easy to use.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 3: SUS Questionnaire





# Appendix D QUIS Questionnaire

Screen				
Characters on the computer screen		hard to read	0 1 2 3 4 5 6 7 8 9	easy to read
Highlighting on the screen simplifies task		not at all	0 1 2 3 4 5 6 7 8 9	very much
Organizing of information on screen		confusing	0 1 2 3 4 5 6 7 8 9	very clear
Usability and User Interface				
Use of colors and sounds		poor	0 1 2 3 4 5 6 7 8 9	good
System feedback		poor	0 1 2 3 4 5 6 7 8 9	good
System messages and reports		poor	0 1 2 3 4 5 6 7 8 9	good
System response to errors		awkward	0 1 2 3 4 5 6 7 8 9	gracious
System clutter and UI noise		poor	0 1 2 3 4 5 6 7 8 9	good
Terminology and System Information				
Use of terms throughout system		inconsistent	0 1 2 3 4 5 6 7 8 9	consistent
Computer terminology is related to the task you are doing		never	0 1 2 3 4 5 6 7 8 9	always
Position of messages on screen		inconsistent	0 1 2 3 4 5 6 7 8 9	consistent
Messages on screen which prompt user for input		confusing	0 1 2 3 4 5 6 7 8 9	clear
Computer keeps you informed about what it is doing		never	0 1 2 3 4 5 6 7 8 9	always
Error messages		unhelpful	0 1 2 3 4 5 6 7 8 9	helpful
Learning				
Learning to operate the system		difficult	0 1 2 3 4 5 6 7 8 9	easy
Exploring new features by trial and error		difficult	0 1 2 3 4 5 6 7 8 9	easy
Remembering names and use of commands		difficult	0 1 2 3 4 5 6 7 8 9	easy
Tasks can be performed in a straight-forward manner		never	0 1 2 3 4 5 6 7 8 9	always
Help messages on the screen		unhelpful	0 1 2 3 4 5 6 7 8 9	helpful
Supplemental reference materials		confusing	0 1 2 3 4 5 6 7 8 9	clear
Overall Reactions to the software				
		terrible	0 1 2 3 4 5 6 7 8 9	wonderful
		difficult	0 1 2 3 4 5 6 7 8 9	easy
		frustrating	0 1 2 3 4 5 6 7 8 9	satisfying
		dull	0 1 2 3 4 5 6 7 8 9	stimulating
		rigid	0 1 2 3 4 5 6 7 8 9	flexible

Figure 4: QUIS Questionnaire