2024

Alistana Nutrition & Fitness Tracker (AFNT)

UFCFXK-30-3 – DIGITAL SYSTEMS PROJECT

ALI SUHAIL | 21072712

UWE Bristol University of the West of England

Notice

The project files and code can be viewed at:

https://github.com/alisuhail-amanu2/AFNT

The Kanban board for the project can be viewed at:

https://trello.com/invite/b/4r2w1F7l/ATTI108ef6ba22ae857e17384de5c89ba99117B65004/project-management

Acknowledgements

I extend my sincere appreciation to Martin Serpell for his invaluable guidance and support during the project's initial planning phase at UWE Bristol, before his retirement.

Furthermore, I would like to express my gratitude to Dr. Eman Qaddoumi for her continuous guidance and unwavering support throughout the module, which significantly contributed to the project's success.

Contents

Table of Figures

Table Content

1. Abstract

This research aims to address the global health challenge of high obesity rates by developing a secure fitness and nutritional tracker to assist individuals in achieving their health goals. The prevalence of obesity, particularly in developed countries, has led to widespread health and economic implications. This study seeks to raise awareness about the significance of good health, emphasizing the positive mental effects associated with fitness.

The primary objective is to create an accessible software solution that encourages health-conscious behaviour and to increase awareness about the benefits of exercising regularly which can help minimize the healthcare burden and lower the risks of diseases and illness. The relevance of this research is underscored by the urgent need to combat the obesity pandemic and understand its far-reaching consequences on both society and the economy.

The anticipated outcomes include fostering a global culture of health consciousness, where individuals comprehend the profound benefits of good health. The research also endeavours to produce an easy-to-use program, empowering users to navigate their fitness journey effectively. By promoting higher fitness levels, the research aspires to contribute to a healthier population, subsequently mitigating the risks of diseases and alleviating the strain on healthcare systems. This project serves as a comprehensive test of software development and planning skills, aimed at making a substantial impact on public health and well-being.

2. Introduction

2.1. The Problem

The global obesity pandemic is a pressing challenge, particularly in developed nations. According to The Guardian, around 38% of the global population, approximately 2.4 billion people, are classified as overweight or obese, and this trend is expected to worsen without effective intervention. By 2035, it's predicted that one in four individuals will be clinically obese (Campbell, 2023). This escalation in obesity rates poses significant health risks, burdening healthcare systems and impacting economic productivity (‘Obesity Consequences’, 2012). Key contributors to obesity include the consumption of high-calorie, fatty foods and insufficient physical activity (Wright and Aronne, 2012).

2.2. Potential Solution

This research aims to address this complex problem by promoting good health and fitness, which not only benefits physical well-being but also mental health. Regular exercise and a health-conscious lifestyle can boost self-confidence, attractiveness, and overall well-being, reducing the strain on individuals and healthcare systems. Embracing a healthy lifestyle can inspire others to prioritize their health, fostering a community focused on wellness and fitness.

2.3. Project Objective

The primary objective of this study is to develop the Alistana Fitness & Nutrition Tracker (AFNT) application, a comprehensive and free tool designed to promote health-conscious behaviours. AFNT aims to increase awareness about the benefits of regular exercise and a balanced diet, thereby reducing healthcare burdens and mitigating risks associated with obesity. The application will enable users to securely track workouts, nutrition, and body progress across mobile and desktop platforms using a unified software solution.

2.4. Project Phases Overview

The AFNT project will be structured into four distinct phases. Phase 1 will focus on developing the database management systems for AFNT, incorporating two main databases—one for updating the AFNT application and another for securely storing user fitness, meal, and body data. All user data will be kept private and accessible only to the respective user. Phase 2 will involve managing updates and establishing preset workouts and meals for the AFNT app through an admin-accessible website. Phase 3 will be the pivotal stage, centred around developing the AFNT Application—a cross-platform application enabling users to track workouts, meals, and body data. Finally, Phase 4 will extend Phase 3 by incorporating hardware in the form of an 'Arduino Watch,' which will measure additional body metrics such as heart rate, blood oxygen level, steps, and body temperature, syncing this data with Phase 3 for user access and logging. This holistic approach underscores the project's commitment to enhancing health awareness and providing a versatile solution for fitness and nutrition tracking.

2.5. Report Structure

The AFNT report is structured into ten detailed chapters, each covering essential aspects of the project development and evaluation. It begins with a comprehensive Literature Review (Chapter 3), exploring technological advancements, the benefits of an active lifestyle, and the evolution of health and fitness

tracking, among other topics. Chapter 4 focuses on Requirements, detailing the structure, categorization, and use case diagrams for the project's objectives. Methodology (Chapter 5) discusses the AFNT phases, sprint planning, and the use of Gantt charts and Trello Kanban for project management. Design (Chapter 6) delves into the specifics of database design, admin website wireframes, AFNT application design, and Arduino Watch components. Implementation (Chapter 7) covers the implementation of each project phase, including DBMS, admin management website, AFNT application, and Arduino Watch functionalities. Project Evaluation (Chapter 8) assesses the achievements and challenges in each phase and discusses the agile and Kanban methodologies used. The report concludes with Further Work and Conclusions (Chapter 9), outlining potential future directions for the project. Finally, References/Bibliography (Chapter 10) provides a list of sources used throughout the report for reference and credibility.

3. Literature Review

3.1. Technological Advancements and Human Lifestyle

Throughout human history, technological advancements have played a pivotal role in making daily activities more efficient and less labour-intensive. From the invention of tools by Homo habilis to the industrial revolution, innovations like the wheel, carts, and various modes of transportation have transformed the way people live and work (Woessner *et al.*, 2021). The Industrial Revolution further enhanced productivity and ushered in an era of electronic and telecommunications revolution, introducing household appliances that reduced manual labour. Simultaneously, advancements in medicine, spanning over two millennia, have significantly contributed to improved healthcare and increased life expectancy. The twentieth century witnessed breakthroughs such as vaccines, early disease diagnosis, and treatment innovations, resulting in a substantial rise in life expectancy to around 80 years.

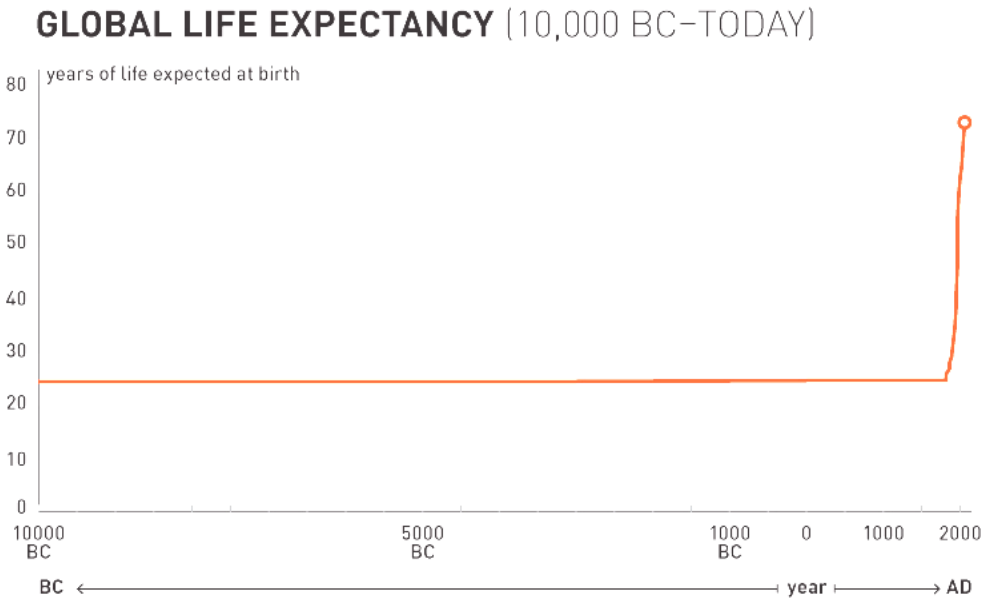


Figure 1: Life Expectancy 10,000 BC - Today (Cato Institute)

However, alongside these benefits, technological proliferation has resulted in a significant decline in incidental physical activity. Everyday activities like active transport and manual labour have been replaced or reduced by technological solutions. The advent of the internet, especially accessible through mobile devices, has further contributed to increased sedentary behaviour, with established associations between internet usage during leisure time and obesity. The overall reduction in physical activity, coupled with a surge in sedentary behaviours, has become a significant factor in the obesity epidemic. Despite technology's positive impact on healthcare and life expectancy, addressing the challenges posed by reduced physical activity remains crucial for promoting overall well-being (Woessner et al., 2021).

Another significant contributor to the rapid increase in obesity rates is the heightened caloric intake, particularly from sweetened beverages, as emphasized by Caballero (2007). These dietary changes, marked by increased consumption of energy-dense foods and a shift away from healthier options, play a substantial role in the current health crisis. The availability of low-cost, easily accessible, and energy-dense food items, combined with changes in dietary patterns, emerges as a prominent factor in the rising rates of obesity. Addressing dietary choices and promoting healthier eating habits are critical components of strategies aimed at combating the obesity epidemic (Caballero, 2007).

Fortunately, technology's ascent has spawned innovative tools for achieving a healthier lifestyle, including mobile phones, smartwatches, and a variety of health and fitness technologies. This market encompasses meditation and workout apps, wearables, connected home gym equipment, Wi-Fi-enabled bathroom scales, and more, offering solutions for weight loss, stress reduction, improved sleep, enhanced immunity, elevated mood, and better nutrition (Moscaritolo, 2024). Additionally, the COVID-19 pandemic accelerated the adoption of health and fitness apps, by gyms closing due to the pandemic, this forced individuals to maintain their well-being from the comfort of their homes. The iOS app market, as analysed by Pankush Kalgotra, Raja, and Sharda (2022), exceeded growth expectations by 29.9%, highlighting the increasing demand for health and fitness-related apps during and after the pandemic.

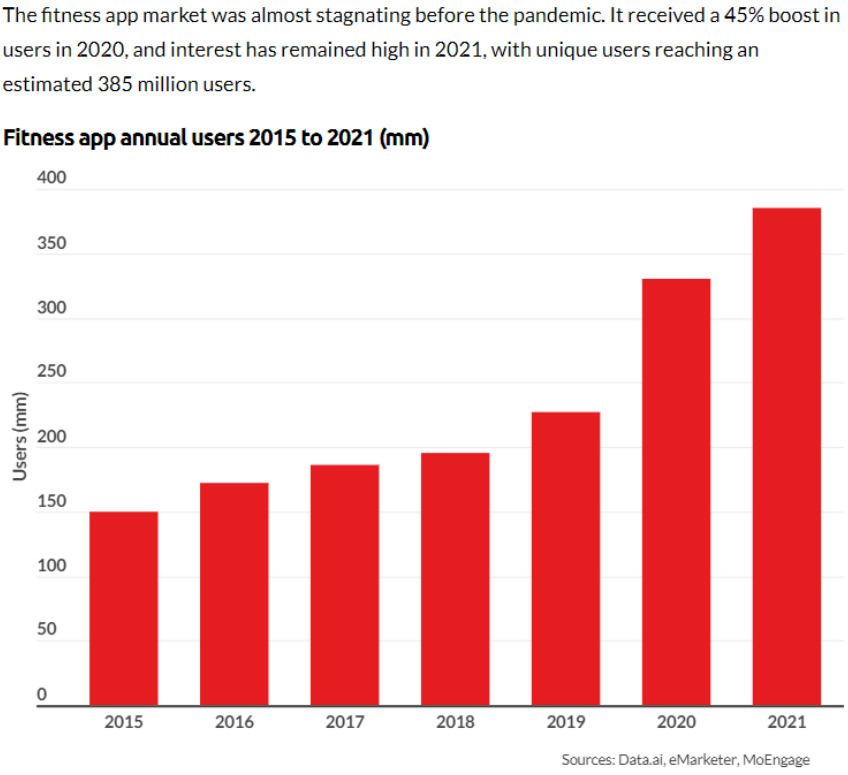


Figure 2: Fitness app annual users 2015-2021

3.2. Benefits of an Active Lifestyle

Engaging in regular exercise not only contributes to improved mental well-being, reducing feelings of anxiety and depression, as highlighted by the Mental Health Foundation (2015), but it also plays a pivotal role in weight management by aiding in the burning of excess calories and enhancing metabolism, according to Mayo Clinic (2023). Additionally, exercise has been shown to enhance brain function, safeguarding memory, and thinking skills, thereby promoting overall cognitive health (Godman, 2014). Beyond mental and cognitive benefits, regular physical activity significantly enhances sleep quality, facilitating quicker sleep onset and deeper sleep experiences ('How Can Exercise Affect Sleep? | Sleep Foundation', 2013). Moreover, exercise positively influences the immune system by promoting optimal circulation and facilitating the efficient movement of immune system cells and substances throughout the body ('How to boost your immune system - Harvard Health', 2014). This multifaceted impact underscores the holistic benefits of incorporating regular exercise into one's lifestyle.

3.3. Evolution and Rise of Health and Fitness Tracking

In the present era, digital and wearable health and fitness technologies seamlessly integrate into our daily lives, with smartphones acting as versatile fitness tracking devices. What sets today's technologies apart is their unparalleled personalization. Unlike the mass-oriented approaches of early 1900s entrepreneurs, modern wearables and health apps delve deep into personal tracking, monitoring everything from dietary habits and sleep patterns to movement frequency and body composition (Millington, 2018).

One of the biggest reasons for the high popularity of health and fitness tracking apps can be attributed to the preference for convenience and flexibility, diverging from traditional gym attendance. A study by Better UK (2020) identified reasons such as time constraints, low confidence, crowded gym environments, and familial obligations as factors influencing people to choose fitness apps over gym visits. Despite gyms offering various tools and fitness trainers, the associated expenses, including costly gym membership fees and personal trainers, make these options financially challenging for some (thefitnessgrp, 2023). Consequently, the cost-effectiveness and accessibility of fitness apps, coupled with the opportunity to adhere to expert guidelines, have led to a growing inclination towards utilizing virtual trainers for fitness training at home, accommodating diverse lifestyles and preferences.

Portability has been another transformative factor in the rapid popularity of fitness-tracking apps. In the late 1800s, Charles Wesley Emerson lamented the immobility of exercise equipment like dumbbells. Even in the late 20th century, health and fitness practices were confined mostly to gyms and homes (Millington, 2018). The breakthrough came with smartwatches like Pebble and Apple Watch, offering not just time-telling but also fitness tracking, app integration, and mobile payment capabilities. These multifunctional wearables evolved from niche fitness gadgets to mainstream devices, capturing consumer imagination. In 2009, James Park and Eric Friedman initiated a revolution with Fitbit, launching the Fitbit Classic—a wearable measuring steps, distance, and calories burned. By gamifying the impactful metric of daily steps, Fitbit mainstreamed the concept that anyone can measure health-affecting metrics, and technology can assist in monitoring. Over fourteen years, health and fitness apps burgeoned into a market worth over $8 billion in 2023, attracting nearly 400 million users in 2021 (‘Fitness App Revenue and Usage Statistics’, 2024).

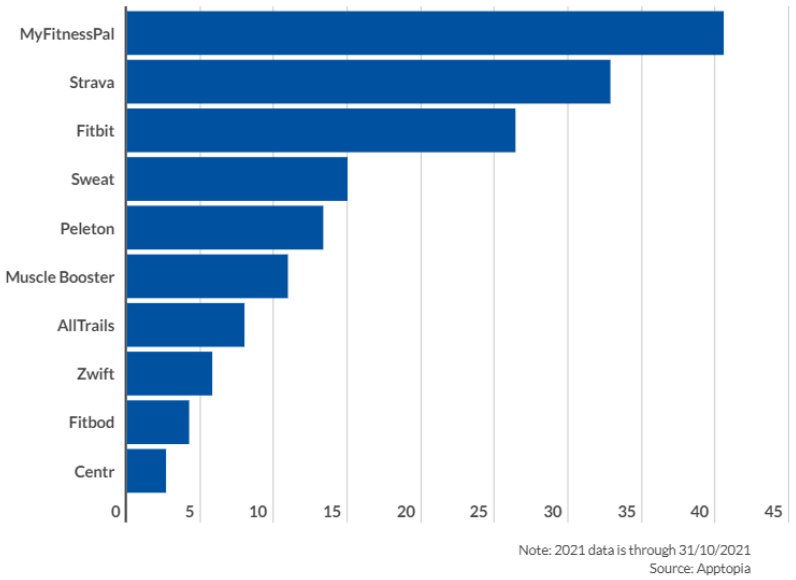


Figure 3: Fitness in-app purchase revenue by app 2021 ($mm)

Fitbit's success influenced major technology players like Apple and Google to enter the fitness app realm, propelled by the widespread adoption of iPhones and the App Store. The Apple Watch, renowned for gathering intricate biometric data, marked a transformative phase in health tracking with apps and wearables monitoring diverse health metrics, including heart rate, sleep patterns, and stress levels. Apple has extended its tracking capabilities beyond fitness to healthcare, unveiling products like the Apple Watch Series 4 and subsequent iterations. These devices not only track irregular heartbeats but also measure blood oxygen levels, have fall detection, and facilitate automatic emergency calls (‘Healthcare - Apple Watch’, 2017). These technologies demonstrate Apple's commitment to advancing technology for comprehensive healthcare solutions.

Fitness trackers, once simple step counters, have evolved into sophisticated health companions. They now feature comprehensive insights into various aspects of physical well-being, from heart rate monitoring and sleep tracking to GPS navigation. Motivating users through goal setting and progress tracking, fitness trackers incorporate gamification elements, adding fun and competition to the fitness journey. Beyond functionality, these trackers have become fashion-forward accessories, seamlessly integrating with everyday attire for constant monitoring of vital health metrics (Waghchoure, 2023).

3.4. Fitness and Health Tracker Varieties

3.4.1. Common Categories of Fitness and Health Trackers

Figure 4: Fitness App Types.

Activity Tracking: These apps utilize sensors in smartphones or wearable devices to monitor physical activities such as steps taken, distance covered, and calories burned. They provide users with real-time feedback on their activity levels and are exemplified by popular apps like Samsung Health (‘Samsung Health’, 2023).

Workout and Exercise: Designed to cater to various fitness levels and goals, workout and exercise apps offer a wide range of routines and fitness plans. Users can choose from strength training, cardio workouts, yoga sessions, and more, depending on their preferences and objectives.

Nutrition and Diet: Nutrition and diet apps play a crucial role in helping users maintain a balanced diet and healthy eating habits. They enable users to track their calorie intake, plan meals, and receive nutritional guidance based on their dietary preferences and goals. Well-known examples include MyFitnessPal (‘MyFitnessPal’, 2024) and Lose It! (‘Lose It!’, 2022).

Running and Cycling: These apps are tailored specifically for runners and cyclists, offering features such as GPS tracking, route planning, and pace monitoring. Apps like Strava are popular among outdoor enthusiasts for their ability to track and analyse performance metrics during runs and rides (‘Strava’, 2024).

Health Metrics Monitoring: These apps integrate seamlessly with wearable devices like Fitbit (‘Fitbit, 2024) or Garmin Connect (‘Garmin’, 2019) to monitor various health metrics such as heart rate, blood pressure, and sleep patterns. They provide users with valuable insights into their overall health trends and help them make informed decisions about their well-being.

These categories encompass a diverse array of fitness app types that cater to different user needs and preferences. While the mentioned categories cover a broad spectrum of fitness and health-related activities, there are also specialized apps (Figure 4) available for specific activities or health goals, highlighting the ever-evolving nature of digital health and fitness technology.

3.4.2. Evolving Trends in Fitness and Health Tracking

Wearable Technology: Leading fitness trackers and smartwatches, such as Fitbit and Apple Watch, utilize advanced sensors to monitor a range of health metrics, including heart rate, sleep patterns, and physical activity. These devices, equipped with GPS technology, can track outdoor activities, and measure additional factors like skin temperature and blood oxygen levels (‘Fitbit’, 2023).

Machine Learning and AI: My Fitness Pal employs a mobile app featuring Machine Learning (ML) algorithms for personalized workout and nutrition recommendations. Analysing users' workout history, biometric data, and relevant information allows the application to adapt and offer more effective, tailored suggestions (‘MyFitnessPal’, 2023).

Natural Language Processing (NLP): Virtual assistants like Apple’s Siri and Google Assistant utilize NLP technology to interpret and respond to voice commands, facilitating hands-free operation of fitness apps during workouts.

Computer Vision: In workout apps, Computer Vision technology ensures users maintain proper exercise form. By analysing video data, computer vision algorithms provide real-time feedback on exercise techniques, promoting correct form.

3.5. Effectiveness of Fitness Trackers

Wearable activity trackers have emerged as cost-effective tools to combat physical inactivity. A comprehensive review of 39 systematic reviews and meta-analyses, spanning diverse populations, demonstrated the positive impact of activity trackers on physical activity, body composition, and fitness, resulting in approximately 1800 extra steps per day, 40 additional minutes of walking, and about 1 kg reduction in body weight (Ferguson et al., 2022).

In a case study, the effectiveness of a wearable fitness tracker, specifically a Fitbit, was explored in a 36-week intervention program for an overweight, type 2 diabetic, geriatric subject. The integrated use of Fitbit reported qualitative improvements in active minutes, steps taken, miles walked, calorie

intake, sleep duration, and liquid consumption. The subject significantly increased daily walking distance from less than one mile to over 4.6 miles, nearing the recommended 5 miles per day goal. The technology facilitated continuous monitoring by the healthcare team, showcasing positive changes in exercise dedication and overall well-being (Thomas William Miller, 2017).

Baseline	Measurements	12 weeks	24 weeks	36 weeks
8.1	HbA1c	7.8	7.3	6.3
14	Average Active Minutes/day	23	48	64
<1000	Average Steps Taken/day	2991	6510	8931
<1.00	Average Miles Walked/day	1.39	3.03	4.15
NA	Calories Burned/day	1.571	2,581	2,758
NA	Sleep hours & minutes/night	6 hr 5 min	6 hr 52min	8 hr 8 min

Figure 5: Variables measured over a 36-week study for a Type 2 Diabetic patient using Fitbit (Thomas William Miller, 2017).

Moreover, another case study that involves a virtual fitness trainer app, assessed through the Situational Motivational Scale (SIMS) with 54 students, demonstrated effectiveness in motivating and engaging users in fitness activities. The app's virtual trainers positively influenced students' motivation, making the activities enjoyable and beneficial for their fitness levels. Respondents found the activities interesting, fun, and essential, highlighting the app's potential to enhance engagement and motivation in fitness-related endeavours (Mokmin and Nurullizam Jamiat, 2020).

In summary, the evidence suggests that wearable fitness trackers, exemplified by Fitbit and virtual trainer apps, play a crucial role in enhancing physical activity, motivating users, and positively impacting overall health and well-being.

3.6. Understanding the Risks Associated with Fitness Trackers

Fitness tracker apps, with their extensive data collection and sharing capabilities, introduce significant privacy and security risks. A survey of 11,000 mobile health apps, representing 5,000 developers, reveals that fitness and nutrition apps are particularly advanced in sharing user data via shared application programming interfaces (APIs) (‘Permissions on Android’, 2024). Sharing commonly occurs with data aggregators like Apple’s HealthKit, wearables, and directly between apps (Grundy, Held and Bero, 2017).

These apps serve diverse functions, accessing vast amounts of highly personal data, including location, text messages, and even camera or photo access (Olmstead, 2015). The inherent access to personal health information heightens privacy risks, with concerns about data being shared with third parties, including advertisers and data brokers (Grundy, Held and Bero, 2017)

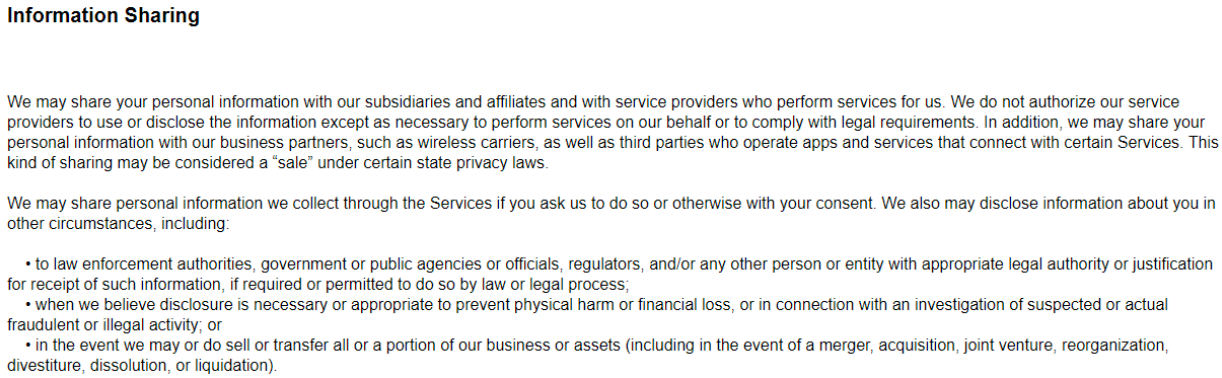
The risks extend to information leaks, manipulation, and loss, as demonstrated by Li's privacy threat model (Li, 2015). User profiling across multiple sites can lead to aggregated user profiles, monetized for marketing, or even exploited for identity fraud (Grundy, Held and Bero, 2017).

Mobile health apps, including fitness trackers, routinely request numerous permissions, indicating a broad spectrum of data access. The most common permissions relate to internet access, with implications for data transmission and sharing (Grundy, Held and Bero, 2017).

Common fitness tracker apps like Samsung Health and Fitbit apps have encountered challenges in usability and data privacy, which have been documented in various sources. Usability issues, including complex interfaces and inconsistent user experiences, missing features have been noted by

reviewers and users alike (‘Appconner’, 2021), along with inaccuracy in collected data in certain apps.

Fitbit's data collection policy outlines that personal data collected through its app, including fitness and health-related information, is used primarily for providing and improving its services, such as personalized recommendations and analytics (‘Fitbit Legal: Privacy Policy’, 2023). Similarly, Samsung Health's data collection policy states that user data, including fitness, nutrition, and biometric information, is utilized for enhancing user experience, research, and product development (‘Samsung Privacy’, 2024).



Information Sharing

We may share your personal information with our subsidiaries and affiliates and with service providers who perform services for us. We do not authorize our service providers to use or disclose the information except as necessary to perform services on our behalf or to comply with legal requirements. In addition, we may share your personal information with our business partners, such as wireless carriers, as well as third parties who operate apps and services that connect with certain Services. This kind of sharing may be considered a “sale” under certain state privacy laws.

We may share personal information we collect through the Services if you ask us to do so or otherwise with your consent. We also may disclose information about you in other circumstances, including:

- to law enforcement authorities, government or public agencies or officials, regulators, and/or any other person or entity with appropriate legal authority or justification for receipt of such information, if required or permitted to do so by law or legal process;
- when we believe disclosure is necessary or appropriate to prevent physical harm or financial loss, or in connection with an investigation of suspected or actual fraudulent or illegal activity; or
- in the event we may or do sell or transfer all or a portion of our business or assets (including in the event of a merger, acquisition, joint venture, reorganization, divestiture, dissolution, or liquidation).

Figure 6: Samsung Information Sharing Policy.

3.7. Ethical Considerations

In developing the AFNT app, ethical considerations play a crucial role in ensuring user trust and data security. The app must prioritize data privacy by complying with GDPR (‘Overview of UK GDPR’, 2016) and implementing secure storage and transmission measures. Intellectual property rights are respected, necessitating proper licensing for third-party content. Accessibility is a key focus, with the app designed to be inclusive and user-friendly for individuals with disabilities following WCAG guidelines (WCAG 2, 2018). Security measures are implemented for the DBMS and ensure encrypted data transfer from the Arduino watch. Hardware standards for the Arduino watch prioritize user safety and comfort. Battery optimization features guarantee prolonged operation, and both the website and app adhere to accessibility standards and offer a user-friendly interface. Compatibility across various mobile platforms and seamless integration with mapping APIs further enhance the app's ethical usability and accessibility.

3.8. Development Research Plan

Embarking on the AFNT Fitness Tracker development necessitates a thorough grasp of technical nuances in each project component. A comprehensive understanding of challenges, tasks, and effective management approaches is crucial. Due to the extensive nature of the AFNT project, it will be initially segmented into five planning stages, as outlined in Figure 7.

Framework Selection

1. Which frameworks are suitable for developing the AFNT Fitness Tracker App, considering factors like cross-platform support, ease of use, and the ability to provide superior user control?

Data Management for AFNT

2. How will AFNT data be managed to ensure user privacy, accessibility, and superior control over personal health information?

Custom Watch Components

3. What are the essential components required to create a custom watch for the AFNT project, emphasizing features like heart rate monitoring and data transfer, while offering users superior control?

User Interface Design

4. How will the user interface of the AFNT app be designed to provide superior user control, ensuring a seamless and personalized experience?

Integration with Health Data

5. How does the AFNT app plan to integrate health data from the custom watch, providing users with superior control over monitoring and managing their health metrics?

Security Measures

6. What security measures will be implemented in the AFNT app to protect user data and provide users with superior control over the privacy and security of their health and fitness information?

Figure 7: AFNT Initial Questions.

3.8.1. Success Criteria

Planning

1. What UML design tool to use?
2. How to design a UML Use Case Diagram?
3. How to design a UML Class Diagram?
4. How to design a UML Sequence Diagram?
5. How to design a sprint log?
6. How to design a log?
7. How do you plan each phase of the project?
8. How do you split objectives into smaller tasks?
9. What Methodology to follow?
10. What test method to follow?
11. What will be the success criteria?

Database Management System (DBMS)

1. How to normalize the databases appropriately?
2. How to set up a central database using a server?
3. How to set up a local database in the local machine?
4. How to connect the central and local database to the application.
5. What type of data to store?
6. How much data should AFNT manage?
7. How to plan and manage database queries?

Admin Management Website (AM)

8. What framework to use for a website?
9. How to make the website communicate with the database server?
10. How to design the website?
11. How to sync the React website to the app to log in/register users?
12. How to implement accessible features on the website?

AFNT Application

13. What language to use for the App?
14. What libraries/plugins to use for the UI design?
15. What libraries/plugins to use for map API?
16. How to design the app?
17. How to make the code more efficient?
18. How to implement more accessible features in the app?
19. How to get and post data from the DBMS?

Arduino Fitness Watch

20. What other categories of smartwatches should we consider for comparison with the Arduino watch?
21. How to efficiently learn C++ language and Arduino IDE?
22. What circuit boards to get and how to wire them?
23. What sensors to purchase?
24. How to store the watch data collected?
25. How to transmit the data to the application wireless/wired?
26. How to store watch data properly in the DBMS?

Figure 8: What I Need to Find Out.

After extensive research and consultations with supervisors, I've restructured the planning phase for AFNT into five key categories, as illustrated in Figure 8. The success criteria revolve around the development of a fully functional AFNT application compatible with both mobile and desktop platforms. It should adeptly track workouts, meals, and body data, and visualize body progress through graphs (Aiming to amalgamate multiple types of fitness apps into one, as depicted in Figure 4). Additionally, the Arduino watch should seamlessly connect and synchronize data with the AFNT app via Bluetooth, providing real-time heart rate, blood oxygen levels, and step count data.

3.8.2. Technical Knowledge

To facilitate diagramming, I've opted for Astah UML for its versatility, ease of use, and popularity in the industry (‘Powerful and Fast UML Diagramming Software - Astah’, 2023). Excel will be utilized for sprint, testing, and planning logs due to its spreadsheet functionality, ease of use and the wide range of planning templates it offers (‘Excel | Microsoft 365’, 2016).

The database architecture will be divided into two components: The Central Database Server (CDB) and the Local Database (LDB). In the LDB, all user personal data will be stored locally, ensuring that it remains exclusively accessible to the user. The database will be created using SQLite, a highly popular and easy-to-use database modelling tool (‘SQLite’, 2024). This approach aligns with AFNT's commitment to maintaining the security and privacy of user data. Conversely, the CDB will house predefined workouts, meals, and user login credentials, facilitating secure login and access to AFNT's user data. CDB will be stored in a MySQL server due to its reliability, scalability, performance, security features, cross-platform compatibility, and strong community support, making it suitable for various applications (‘SQL Server | Microsoft’, 2022).

For the AM Website, Python Flask was chosen for its simplicity and widespread use in the industry (‘Flask Documentation (3.0.x)’, 2024). Extensive online resources are available to support effective implementation. In developing the AFNT application and Graphical User Interface (GUI), Python programming language and the Kivy GUI Framework were used as it offers flexibility and cross-platform support, making it an ideal choice. It's open-source and user-friendly (‘Kivy: Cross-platform Python Framework for NUI’, 2024).

Regarding the Arduino Fitness Watch, components from Tiny Circuits were selected for their compact design and ease of assembly, resembling modular electronic components akin to Lego blocks (‘TinyCircuits - Maker of Tiny, Open-Source Electronics’, 2024). Tiny Circuits also boasts a supportive community and provides comprehensive documentation and basic examples to aid development (‘Tiny Circuit Projects’, 2022). All components were acquired at personal expense.

Product	SKU	Quantity	Total	Arrived
Wireling Adaptor TinyShield	ASD2022	1	£9.00	04/11/2024
Accelerometer TinyShield	ASD2511-R-A	1	£10.00	04/11/2024
5-Pin Wireling Cables - 50mm/1	ASR00022	2	£4.00	04/11/2024
TinyScreen+ (Processor, OLED & USB in one)	ASM2022	1	£15.00	04/11/2024
SanDisk Ultra 64GB microSD card	-	1	£8.45	20/01/2024
Pulse Oximeter Sensor Wireling	AST1041	1	£12.00	23/01/2024
Bluetooth Low Energy TinyShield (ST)	ASD2116-R	1	£15.00	23/01/2024
Accelerometer Wireling	AST1001	1	£5.00	23/01/2024
Lithium Ion Polymer Battery - 3.7V 290mAh	ASR00007	1	£5.00	23/01/2024
MicroSD TinyShield	ASD2201-R	1	£10.00	23/01/2024
Total			**£93.45**	

Figure 9: Arduino Watch Components.

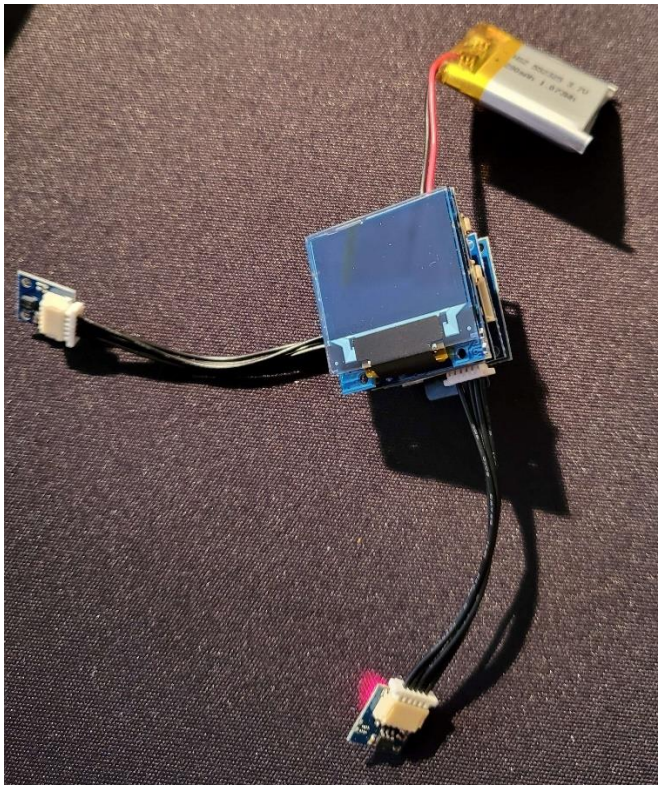


Figure 10: Assembly of All Arduino Watch Modules

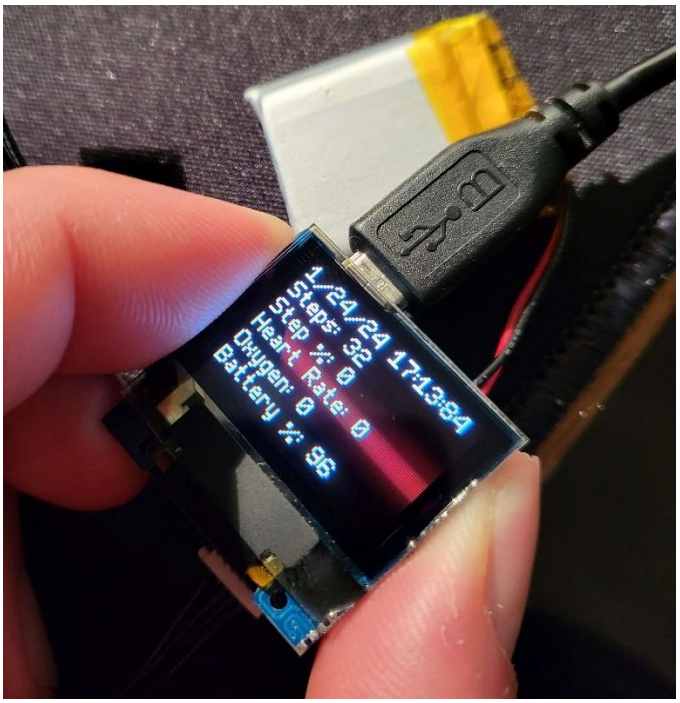


Figure 11: First Version of Arduino Watch Interface (24/01/2024).

3.8.3. AFNT Evaluation Criteria

In this study, we will evaluate two widely used fitness tracker apps, Samsung Health, and Fitbit, based on their performance and features, as illustrated in Figure 9. AFNT stands out from Samsung Health (Paired with Samsung Galaxy Watch 1) and Fitbit app (Paired with Fitbit Flex 2) in several aspects:

Features	Fitbit Flex 2 2016	Samsung Galaxy Watch 2018 (46mm)	Arduino Watch (Prototype)
App Name	Fitbit	Samsung Health	AFNT
User Login	Email	Email	Email
Make ID and Username	Yes	Yes	Yes
Data Storage	Local/Cloud	Local/Cloud	Local/Cloud
Data Synchronization	Yes	Yes	Yes
Data Modification	Yes	Yes	Yes
Personal Data Sharing	Limited Circumstances	May Utilize Aggregated and Anonymized User Data for Marketing/Advertising	No Data is Shared, Unless Requested
Desktop Support (Cross-Platform)	No	No	Yes
Goal Setting	Yes	Yes	Yes
Track Progress	Yes	Yes	Yes
BMI	Yes	Yes	Yes
Height	Yes	Yes	Yes
Weight Tracking	Yes	Yes	Yes
Water Intake	No	Yes	Yes
Body Fat Tracking	Yes	Yes	Yes
Skeletal Mass Tracking	No	No	Yes
Progress Reports	Yes	Yes	Yes
Step Count	Yes	Yes	Yes
Distance	Yes	Yes	No (OOS)
Floor Tracking	No	Yes	No (OOS)
Sleep Tracking	Limited	Yes	No (OOS)
Weight Progression	Yes	Yes	Yes
Caloric Analysis	Yes	Yes	Yes
Meal Tracking	Yes	Yes	Yes
Food Item Tracking	Yes	Yes	Yes
Nutrient Detail Analysis	Limited	Yes	Yes
Heart Rate Measurement	No	Yes	Yes
Heart Rate Log	No	Yes	Yes
Workout Tracking	Limited	Limited	Yes
Exercise Tracking	Limited	Limited	Yes
Water Resistant	Yes	Yes	No (OOS)
Swim Tracking	Yes	Yes	No (OOS)
Call and Text Features	Limited	Yes	No (OOS)
Touch Screen	No	Yes	No (OOS)
GPS	No	Yes	No (OOS)
Battery Capacity (mAh)	Replaceable coin cell	472	290
Weight w/o Strap (g)	13	64	17
Storage Capacity (gb)	No Storage	4-8	64

	Yes
	Limited
	No
	Out of Scope (OOS)

Figure 12: AFNT and Arduino Watch Evaluation Table.

Selection Rationale for Samsung Galaxy Watch and Fitbit Flex 2: The Samsung Galaxy Watch and Fitbit Flex 2 were chosen for practical reasons, based on my prior experience with the Samsung Galaxy Watch, which enables thorough testing of both devices. Moreover, the Fitbit Flex 2 was selected due to its fitness-focused design akin to the Arduino Watch, distinguishing it from multipurpose/flagship smartwatches like the Apple Watch or higher-end Samsung models. Its core features centre on tracking activities such as steps, distance, calories, and sleep patterns, with limited smartwatch functionalities such as call and text notifications. Moreover, Including the original Galaxy Watch allows for a comparison of the Arduino watch's capabilities against a more advanced smartwatch in terms of both hardware and software, while cost considerations also influenced the selection process.

Figure 13: Samsung Galaxy Watch (2018).

Figure 14: Fitbit Flex 2 (2016).

Privacy Measures: AFNT emphasizes data privacy by storing user data locally on the device (LDB), ensuring that personal information is not utilized for marketing, advertisement, or analytical purposes as AFNT has no way to access this data. It also incorporates a feedback mechanism for users to share data securely, a feature less emphasized in Samsung Health and Fitbit apps.

Cross-Platform Functionality: AFNT is designed to function seamlessly across both mobile and desktop platforms, providing users with flexibility in accessing their fitness and health data. This versatility contrasts with the mobile-centric approach of Samsung Health and Fitbit apps.

Customizable Workout Plans: Unlike Samsung Health and Fitbit apps, AFNT allows users to tailor and customize their workout plans according to their specific fitness objectives and preferences, fostering greater user engagement and motivation.

Enhanced User Control: AFNT offers users extensive control over their fitness, health, and nutrition data. Users can modify the nutritional details of food items, rate workouts, customize exercise aspects, allocate workouts and meals, and manage data synchronization across devices. This level of control enhances the user experience and personalization, surpassing what is currently offered by Samsung Health and Fitbit apps.

3.8.4. Requirements Gathering

The app aims to provide a seamless user experience with features like customizable workout plans, nutritional tracking, and real-time health data from the Arduino watch. User-friendly interfaces, security measures, and cross-platform compatibility are fundamental requirements. These requirements were gathered in the following ways:

- Regularly consulting mentors like Martin Serpell and Eman Qaddoumi for industry insights and best practices in fitness app development.
- Analysing UI, features, and functionalities of popular fitness apps like Samsung Health and Fitbit to guide AFNT app design decisions.
- Tracking personal fitness progress at the gym and understanding user expectations firsthand (Figure 15, Figure 16 and Figure 17).
- Conducting comprehensive online research to stay abreast of emerging trends and user demands in fitness app development.

Week 3

Wednesday 17/01/2024

Push	Workout	Status	Details
Bench Press	3x8, 40kg (10kg*2), 90s rest	10	3x8 35kg
DB Bench	3x10, 40kg (20kg*2), 90s rest	10	2x10 40kg 1x12 35kg
Overhead Press (Machine) OR	3x10, 22.5kg, 90s rest 3x10, 25kg (12.5kg*2) DBs, 90s rest	9 10	3x10 22.5kg OPM
Shoulder Raises	3x10, 12kg (6kg*2), 90s rest	9	
DB Front Delt Raise	3x14, 12kg (6kg*2), 90s rest	9	
Bicep Curl Superset Curved Bar Dumbbells	3x9, 20kg OR 3x9, 10kg DBs, 90s rest	10	
Lat Pulls (Machine)	3x10, 35kg, 90s rest	9	
Lat Pull Down	3x10, 45.5kg, 90s rest	9	
Chin Assist Dip Assist	3x8, 0kg, 90s rest, CA 3x8, 0kg, 90s rest, DA	8	1x5 CA 1x8 DA Did pullups

Friday 19/01/2024

Pull	Workout	Status	Details
Deadlift	3x10, 62.5kg (20kg + 1.25kg)	9	
BB Bent Over Row	3x10, 42.5kg (10kg + 1.25kg), 90s rest	10	
Chin Assist Dip Assist	3x5, 0kg, 90s rest, CA 3x8, 0kg, 90s rest, DA	10 9	
Lat Press	3x10, 44.5kg, 90s rest	9	
Bicep Curl Superset Curved Bar Dumbbells	3x9 20kg, CB 3x8 10kg DBs, 90s rest, DB	10	
Stiff Arm Pull Down FUNC	3x10, 18.5kg, 90s rest	9	
Tricep Curls	3x10, 13.5kg, 90s rest	10	2x10 13.75kg 1x10 11.50kg
Overhead Extension	3x8 8.5kg, 90s rest	10	
DB Front Delt Raise	3x14, 16kg (8kg*2), 90s rest	-	No Time
Leg Raises	3x20, 90s rest	10	
Delt Pulls	3x10, 59kg, 90s rest	-	No Machine

Saturday 20/01/2024

Leg	Workout	Status	Details
Back Squats	3x10, 50kg (15kg), 120s rest	9	1x10 30kg 1x10 35kg 1x10 40kg 1x10 45kg 1x10 50kg
Single Rack Carry Weighted Lunges	3x2 lengths, 32kg (16kg x 2) KBs 3x16, 28kg (14kg*2) DBs, 90s rest	-	
Leg Press	3x10, 95kg (20kg + 15kg + 2.5kg) DB, 90s rest	10	3x10 72.5kg
Prone Leg Curl	3x9, 32kg, 90s rest	-	
DB Incline Press	3x10, 40kg (20kg*2), 90s rest	10	
Calf Press	3x30, 90kg, 90s rest	9	1x30 80kg 1x30 87kg
Chin Assist Dip Assist	3x8, 0kg, 90s rest, CA 3x8, 0kg, 90s rest, DA	-	No Time
Leg Extension	3x10, 57lbs, 90s rest	10	3x10 57lbs

Figure 15: My workout tracking method using Excel Spreadsheet.

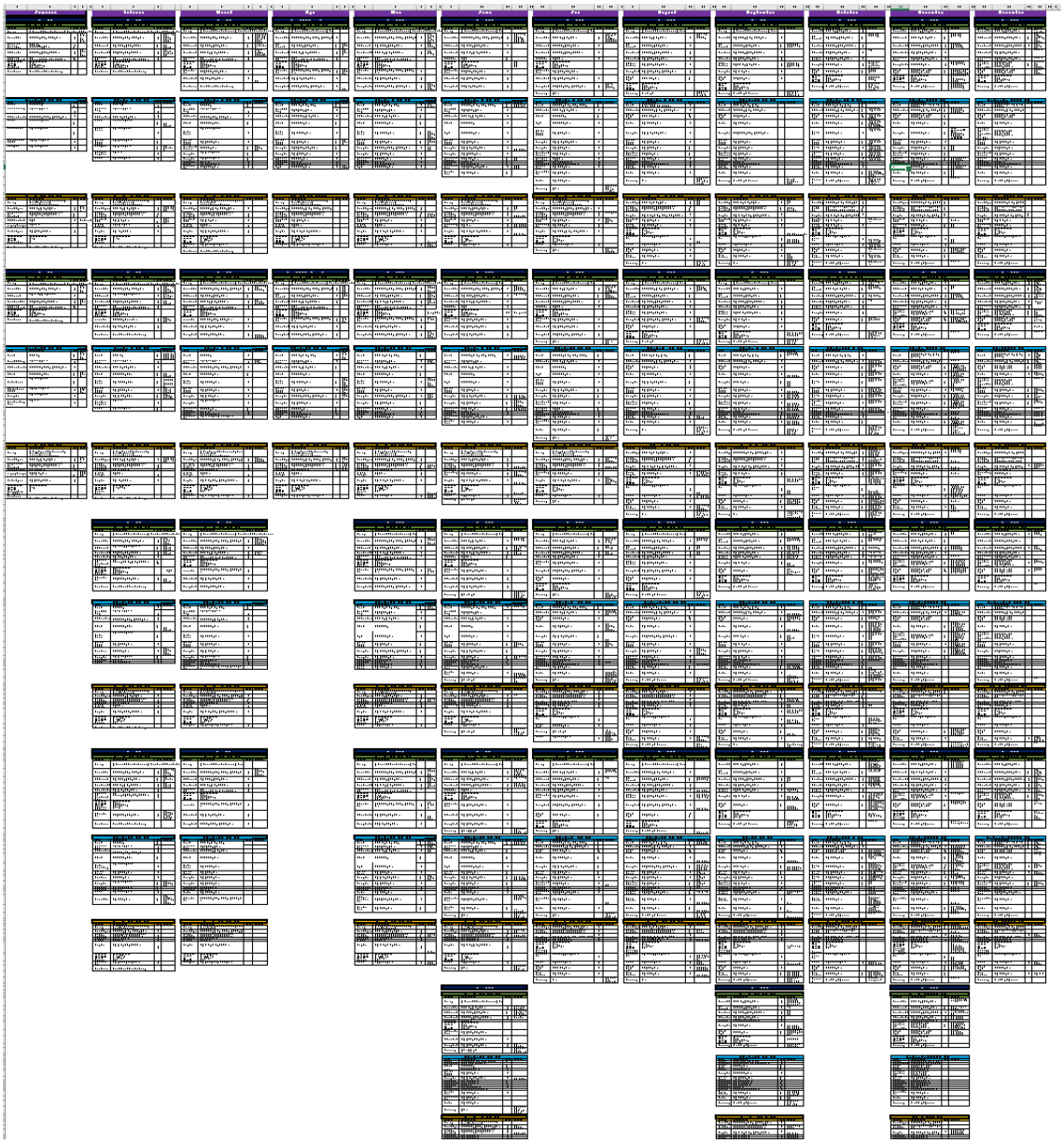


Figure 16: My workout data using Excel – 2023

Exercise	Type	19/01/2023
Bench Press	Pecs, Front Delts, Triceps	32.5
Incline DB Press	Pecs, Front Delts, Triceps	20.0
Overhead Press	Front/Middle Delts, Triceps, Traps	16.0
DB Shoulder Raises	Front/Middle Delts, Serratus	5.0
Bicep Curls	Biceps	5.5
Chin Assists	Biceps, Lats, Pecs, Rear Delts	37.8
Lat Pull downs	Lats, Traps, Biceps, Rear Delts, Rhomboids	32.0
Stiff Arm Pull Down	Rear Delts, Triceps, Lats, Rhomboids	10.2
Deadlifts	Quads, Glutes, Hamstrings, Core, Back, Traps	57.5
GHD	Hamstrings	5.0
Weighted Lunges	Quads, Glutes, Hamstrings, Calves	16.0
Back Squats	Lower Back, Quads, Glutes, Hamstrings	37.5
Total		**275.0**

Figure 17: My workout progress evaluation – 2023

4. Requirements

An essential aspect crucial to the success of any project lies in defining a comprehensive set of requirements. These requirements serve as concise outlines detailing the project's objectives and the expected performance of its deliverables. They can encompass both broad and specific descriptions of the system's behaviour and functionality.

ID	Objective	Description	Created
O1	Develop a Scalable and Secure Database Management System (DBMS)	Develop a scalable MySQL DBMS, including a central database for 2,000+ records and a local database for user data and watch data. Improve database performance and security and ensure GDPR compliance.	09/10/2023
O2	Build the Alistana Fitness & Nutrition Tracker (AFNT) Application	Develop the AFNT application with features for workout and nutrition tracking, body progress and measurement tracking. Should be connected to DBMS, Admin Management website (only for Admins), and Arduino watch.	09/10/2023
O3	Design a User-Centric Admin Management (AM) Website	Develop a responsive website with secure login and allows Admins to edit the central database (user login data and preset workout and meal data) and app push updates to the AFNT application. The website will prioritize a user-friendly design, encrypted communication, and security measures.	09/10/2023
O4	Design and develop a Fitness Watch using Arduino	Create an Arduino-based Fitness watch to measure blood oxygen level, heart rate and step count, and investigate ways of connecting the Arduino watch to the AFNT and store body data in the local database via AFNT.	16/10/2023
O5	Enhance Code Quality and Performance	Implement clean, maintainable code with 80% code coverage. Optimize application and website response times to under 2 and 3 seconds, respectively.	25/10/2023

Figure 18: AFNT Project Objectives.

In refining the final list of requirements, careful consideration was given to attributes such as traceability, testability, and consistency. Some requirements may represent distinct functionalities, contingent upon the completion of others, and they are noted for their dependencies. The subsequent sections will delineate the structured format of the requirements, categorizing them into non-functional and functional requirements. Additionally, a use case diagram, use case descriptions, class diagrams and the project's overall architecture diagram will be presented.

Note: Click Here for the complete list of project objectives, use case diagrams with descriptions, and requirements.

The requirement structure is specifically crafted to ensure clarity and facilitate efficient tracing and testing of project objectives or requirements. The requirement code follows a well-defined format as outlined below:

<***Requirement/Objective***><***Phase Component***><***Number***>

4.1.1. Requirement/Objective:

This component indicates whether the property is a **functional** requirement ('F'), a **non-functional** requirement ('NF'), or an overall project **objective** ('O').

4.1.2. Phase Component:

It signifies the specific phase of the project to which the requirement or objective pertains. Each phase is represented by a distinct code, such as 'A' for **App** development, 'D' for **Database**, or 'AW' for **Arduino Watch** development.

4.1.3. Number:

Used to differentiate between cases with the same requirement or objective and phase. It ensures unique identification within a specific phase and requirement/objective category.

For instance, adding meals and workouts are both functional requirements for the app development phase of the AFNT App. Despite being similar, they are assigned different numbers to distinguish them from each other effectively.

The table below succinctly summarizes all the codes and their meanings for easy reference and streamlined project management. The table below displays all the codes and their meanings for simplicity and easy reference.

Table 1: Requirement/Objective Abbreviations.

Requirement/ Objective	
O	Objective
F	Functional Requirement
NF	Non-Functional Requirement

Table 2: Phase Component Abbreviations.

Phase Component	
A	AFNT Application
AW	Arduino Watch
D	Database
DBR	Data Backup & Recovery
DS	Data Storage & Optimization
ES	Efficiency & Sustainability
P	Performance
PS	Privacy & Security
R	Reliability
TPS	Third-Party Service Integration
U	Usability
W	Admin Management Website

4.2. Requirement Categorization

The project's requirements are distributed across various phases, with each requirement accompanied by a concise description of its purpose. Additionally, each requirement is prioritized using the MoSCoW prioritization technique (Agile Business, 2022), which enables effective planning and management throughout the project lifecycle. MoSCoW prioritization was selected for its simplicity and suitability for both pre-planning and post-planning stages.

Furthermore, each requirement is tagged with a completion status to facilitate straightforward tracking of progress. Additionally, the date of creation or last update is recorded for each requirement, providing valuable insights into the project’s development over time.

Table 3: MoSCow Prioritization table.

Priority Level	Abbreviation	Meaning
Must have	M	Task that is crucial for the project.
Should have	S	Important task but not vital for the final project.
Could have	C	Wanted or desirable but not important for the project's overall goal.
Won't have	W	Out of scope.

Table 4: Requirement Table Overview.

Performance				
ID	**Summary**	**Priority**	**Status**	**Created**
NFP1	The AM website should load in under 5 seconds, ensuring optimal user experience	S	Complete	09/10/2023
NFP2	The AM website shall respond to user input within 200 milliseconds, providing a smooth and responsive interaction.	S	Partially Complete	09/10/2023
NFP3	The application shall load within a 5-second time frame, ensuring users can access the features promptly.	S	Complete	09/10/2023
NFP4	The Arduino watch should display heart rate, blood oxygen level, steps count in real time on the watch display.	M	Not Started	17/10/2023

Note: For further reading on project aims, objectives, and requirements, please navigate to Appendix A: Requirements.

4.3. Usecase Diagrams and Descriptions

An effective strategy for informing and crafting a set of requirements is through the creation of a use case-driven system. Use cases offer detailed insights into how the system will be utilized, encompassing information about users, usage scenarios, and the sequences and structures inherent in its functioning. The use of UML diagrams is prevalent in this context, as they are widely recognized as an industry standard. Many software developers rely on UML diagrams to articulate software design models, making familiarity with them a common expectation among software professionals (Alam, 2022).

For this project, three UML use case diagrams have been developed, each corresponding to a different aspect of the project: one for the AFNT Application, one for the Arduino Watch, and one for the Admin Management Website. These diagrams are designed to align with the project's requirements and objectives. Given the complexity of each diagram, detailed descriptions of the use cases have been provided to offer a deeper understanding, particularly for less detailed use cases. Additionally, certain use case descriptions encompass multiple related use cases to streamline the presentation. For instance, the "Manage workouts" use case may include sub-use cases such as "Add Workout," "Edit Workout," and "Delete Workouts," which are combined into a single use case description for clarity and simplicity.

4.3.1. Usecase Diagrams and Descriptions

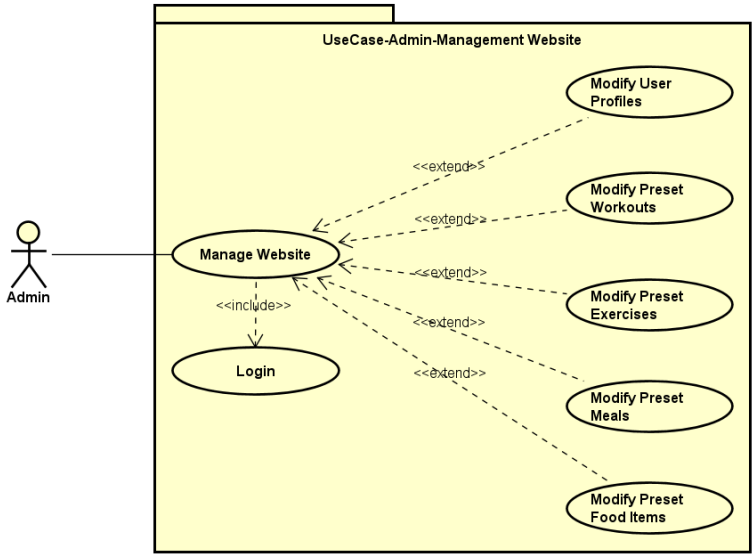


Figure 19: Usecase Admin Management Website.

Table 5: Admin Management Website Usecase description.

Use Case Diagram	Admin Management Website
Use Case Identifier	Manage Website and the <<extend>>
Goal	Admins and modify the Central Database and push updates to AFNT app
Priority	M
Updated	21/03/2024
Participating Actors	Admin
Pre-conditions	Admins needs to login, upon a successful login Admins are directed to the admin dashboard
Post-conditions	Admins need to navigate to the 'Edit Central DB' tab
Basic Flow of Events	1. Admin logs on the website 2. Upon a successful login, admins are redirected to the Admin Dashboard. 3. Admins can navigate to 'Edit Central DB' and can modify its contents (consists of presets meals and workout data, and user login data). 4. After implementing successful changes, admins can push this and the LocalDB will import the new centralDB updates when user logins successfully.
Alternative Flow	None

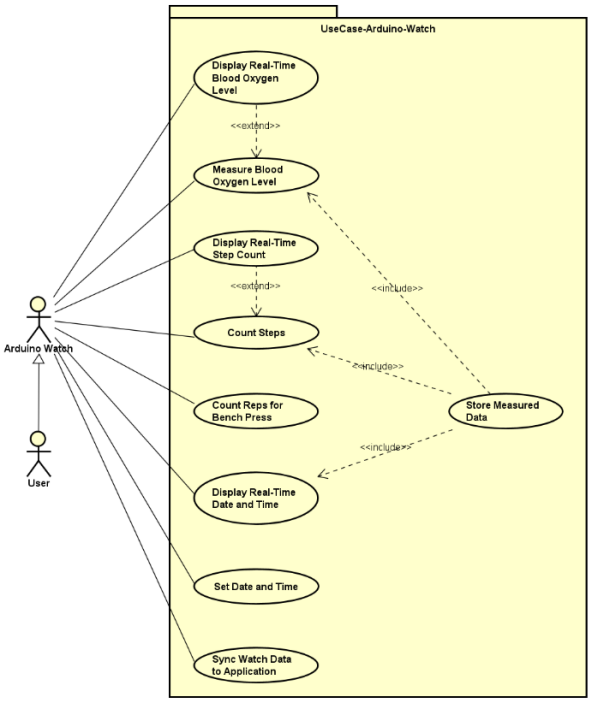


Figure 20: Arduino Watch Usecase Diagram.

Table 6: Store Measured Data and the <<extend>> Usecase description.

Use Case Diagram	Arduino Watch
Use Case Identifier	Store Measured Data and the <<extend>>
Goal	The Arduino Watch can store and measure body data (Step Count, Heart Rate, Blood Oxygen level, Count Reps, and current Date and Time)
Priority	M
Updated	21/03/2024
Participating Actors	User Admin
Pre-conditions	User must have an Arduino Watch which is turned on and straped properly on the wrist.
Post-conditions	
Basic Flow of Events	1. Turn on the Arduino Watch 2. Properly wear the Arduino Watch around the wrist. 3. Arduino Watch should start measuring real time heart rate, blood oxygen, step count and the time data and store this data in the microSD card, which can be accessed to view the data collected.
Alternative Flow	None

Table 7: Sync Watch Data to Application Usecase description.

Use Case Diagram	Arduino Watch
Use Case Identifier	Sync Watch Data to Application
Goal	The Arduino Watch shall send the collected user data to the AFNT App via Bluetooth, which can be accessed and viewed on the AFNT App live.
Priority	M
Updated	21/03/2024
Participating Actors	User Admin
Pre-conditions	User must have an Arduino Watch which is turned on and straped properly on the wrist.
Post-conditions	The Arduino Watch shall be linked to AFNT App via Bluetooth.
Basic Flow of Events	1. Turn on the Arduino Watch 2. Properly wear the Arduino Watch around the wrist. 3. On the AFNT App, go to 'Arduino Watch' tab and press sync. 4. The app should link with the Arduino Watch after successfully finding it, and the Arduino Watch shall start transmitting data to the AFNT App.
Alternative Flow	None

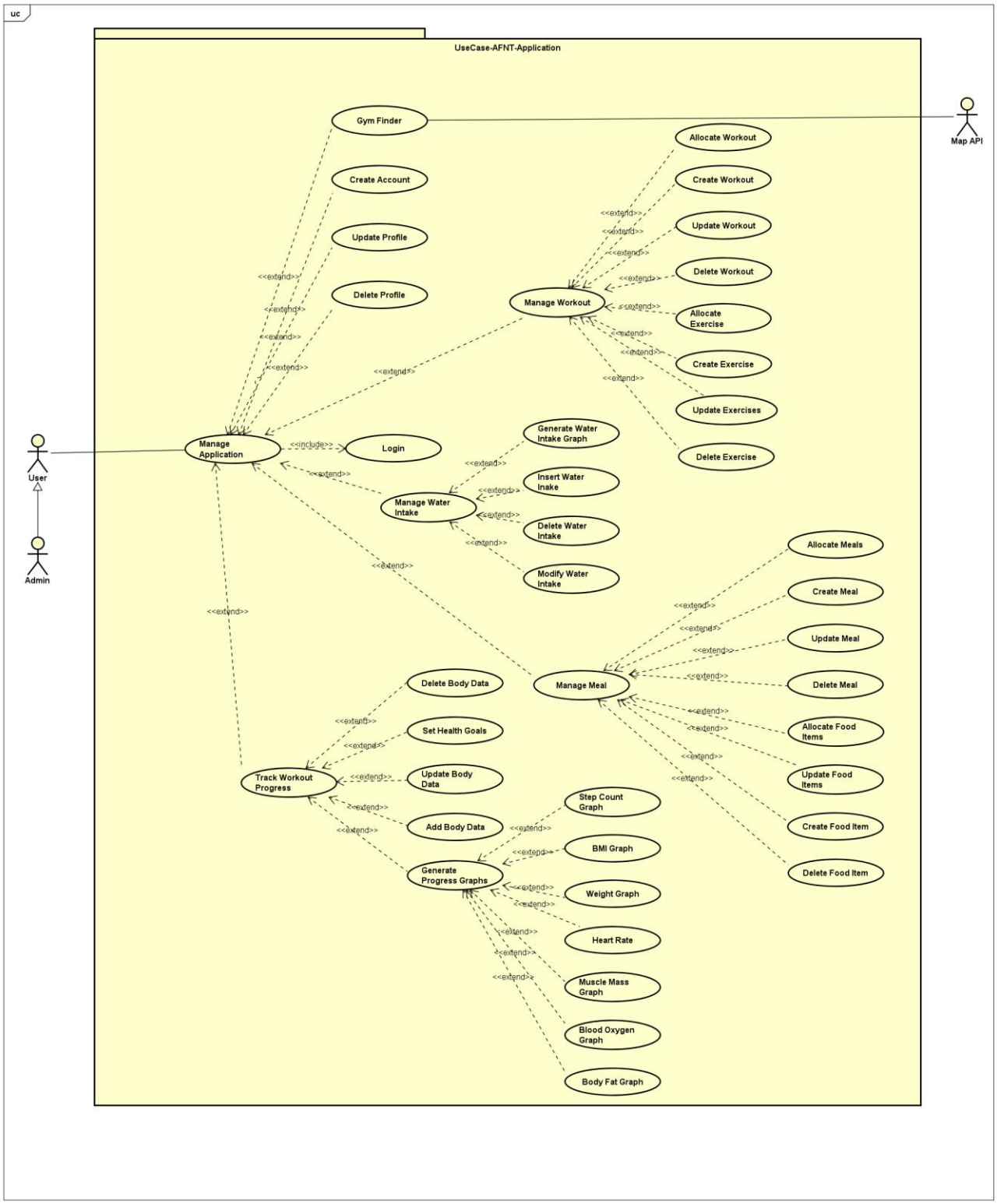


Figure 21: AFNT App Usecase Diagram.

Table 8: Manage Workout and the <<extend>> Usecase description.

Use Case Diagram	AFNT Application
Use Case Identifier	Manage Workout and the <<extend>>
Goal	Manage Workouts and Exercises (Add/Update/Delete)
Priority	M
Updated	21/03/2024
Participating Actors	User Admin
Pre-conditions	Details of Workouts and Exercises are stored in the LocalDB database and to access its contents and modify data, the User must be in the "Workout" tab.
Post-conditions	Details of Workouts and Exercises are displayed to the User/Admin
Basic Flow of Events	1. User navigates to the 'Workout' tab. 2. App displays all workouts planned for the current week. 3. User can change the date range to see workouts in the past and future 4. User can select the workout and view the exercises allocated in the selected workout. 5. The page includes buttons that allow user to allocate, create, edit, and delete workouts and exercises.
Alternative Flow	None

Table 9: Manage Meal and the <<extend>> Usecase description.

Use Case Diagram	AFNT Application
Use Case Identifier	Manage Meal and the <<extend>>
Goal	Manage Meals and Food Items (Add/Update/Delete)
Priority	M
Updated	21/03/2024
Participating Actors	User Admin
Pre-conditions	Details of Meals and Food Items are stored in the LocalDB database and to access its contents and modify data, the User must be in the "Meal" tab.
Post-conditions	Details of Meals and Food Items are displayed to the User/Admin
Basic Flow of Events	1. User navigates to the 'Meal' tab. 2. App displays all meals planned for the current week. 3. User can change the date range to see meals in the past and future 4. User can select the meal and view the food items allocated in the selected meal. 5. The page includes buttons that allow user to allocate, create, edit, and delete meals and food items.
Alternative Flow	None

Table 10: Manage Water Intake and the <<extend>> Usecase description.

Use Case Diagram	AFNT Application
Use Case Identifier	Manage Water Intake and the <<extend>>
Goal	Manage Water Intake (Add/Update/Delete)
Priority	M
Updated	21/03/2024
Participating Actors	User Admin
Pre-conditions	Details of Water Intake (ml) are stored in the LocalDB database and to access its contents and modify data, the User must be in the "Water Intake" tab.
Post-conditions	The screen contains a foam where the User can select the date and the amount of water drank. The use can also generate graphs of water intake per selected month/selected year
Basic Flow of Events	1. User navigates to the 'Water Intake' tab. 2. App displays a form for adding water intake in the selected date (default date is the current date). 3. User can change the date to update/add the water intake and click 'Submit' to add/update records. 4. The page also contains a button to delete all water intake data. 5. The page also contains a button to generate two water intake graphs, one for per selected month, and one for the selected year
Alternative Flow	None

Table 11: Gym Finder Usecase description.

Use Case Diagram	AFNT Application
Use Case Identifier	Gym Finder
Goal	Find nearest gym relative using user's location data.
Priority	M
Updated	21/03/2024
Participating Actors	User Admin MapAPI
Pre-conditions	User must consent to share their location
Post-conditions	The screen displays the nearest gym to users' location around a 10-mile radius
Basic Flow of Events	1. User navigates to the 'Gym Finder' tab. 2. App uses user's location data and fetches the nearest gyms to the user's location. 3. User can generate a path to the selected gym and can download/save the location and directions.
Alternative Flow	None

Table 12: Create Account Usecase description.

Use Case Diagram	AFNT Application
Use Case Identifier	Create Account
Goal	Users can create a new account.
Priority	M
Updated	21/03/2024
Participating Actors	User Admin
Pre-conditions	User must navigate to the 'Register' screen
Post-conditions	The user must enter their details like username, password, gender, dob email, phone number and postcode (optional)
Basic Flow of Events	1. User launches the AFNT app. 2. The app displays a login page, user needs to select the 'Register' button. 3. User needs to provide the details stated in post-conditions. 4. After entering valid data, then press 'Register'. 5. A success popup will be displayed, and the user can now login using the new account.
Alternative Flow	None

Table 13: Update Profile Usecase description.

Use Case Diagram	AFNT Application
Use Case Identifier	Update Profile
Goal	Users can update their profile (Add/Update)
Priority	M
Updated	21/03/2024
Participating Actors	User Admin
Pre-conditions	User must navigate to the 'Profile' screen by clicking the profile icon in the 'Dashboard' screen
Post-conditions	The user can modify details like username, password, gender, dob email, phone and postcode (optional)
Basic Flow of Events	1. User opens the AFNT app 2. User logs in. 3. User is sent to the dashboard screen. 4. User clicks on the profile icon and clicks 'Profile'. 5. User can now modify the account details stated in post-conditions
Alternative Flow	None

Table 14: Delete Profile Usecase description.

Use Case Diagram	AFNT Application
Use Case Identifier	Delete Profile
Goal	Users can delete their profile
Priority	M
Updated	21/03/2024
Participating Actors	User Admin
Pre-conditions	User must navigate to the 'Profile' screen by clicking the profile icon in the 'Dashboard' screen
Post-conditions	The user can delete their profile (and all the data associated with it)
Basic Flow of Events	1. User opens the AFNT app 2. User logs in. 3. User is sent to the dashboard screen. 4. User clicks on the profile icon and clicks 'Profile'. 5. There is a delete button in the bottom of the screen and clicking that will delete the profile and all the data associated with it. The user is then sent back to the login screen.
Alternative Flow	None

Table 15: Track Workout Progress and the <<extend>>

Use Case Diagram	AFNT Application
Use Case Identifier	Track Workout Progress and the <<extend>>
Goal	Users can track their workout progress
Priority	M
Updated	21/03/2024
Participating Actors	User Admin
Pre-conditions	User must navigate to the 'Body Stats' screen in the 'Dashboard' screen
Post-conditions	The user can monitor and modify their workout progress data like step count, BMI, weight, heart rate, muscle mass, blood oxygen level and body fat data.
Basic Flow of Events	1. User navigates to the dashboard screen. 2. User clicks on the 'Body Stats' tab. 3. Users can input their current body stat (weight, height, BMI etc.). User can also update and delete data. 4. User can also set health goals (i.e. set weight goal, step goal etc.)
Alternative Flow	None

Table 16: Generate Progress Graph and the <<extend>> Usecase description.

Use Case Diagram	AFNT Application
Use Case Identifier	Generate Progress Graphs and the <<extend>>
Goal	Users can generate graphs to track their workout progress
Priority	M
Updated	21/03/2024
Participating Actors	User Admin
Pre-conditions	User must navigate to the 'Body Stats' screen in the 'Dashboard' screen. Then select one of the generate graphs button.
Post-conditions	The user can generate graph data for step count, BMI, weight, heart rate, muscle mass, blood oxygen level and body fat data.
Basic Flow of Events	1. User navigates to the dashboard screen. 2. User clicks on the 'Body Stats' tab. 3. User needs to input the month and year and then select which body stat to generate a graph for. 4. The app will then display a monthly and yearly graph for the selected body stat.
Alternative Flow	None

In conclusion, the Requirements chapter provides a comprehensive overview of the AFNT project's objectives, functionalities, and performance expectations. By adhering to a structured format and utilizing prioritization techniques, the requirements are organized and categorized to facilitate efficient project management. The inclusion of use case diagrams and descriptions further enhances understanding and alignment with project goals. Overall, this chapter serves as a crucial reference point for understanding AFNT’s key goals and objectives.

5. Methodology

In this chapter, we will explore how agile principles and the Trello Kanban methodology were utilized in the AFNT project to ensure smooth development and delivery. The chapter will discuss the rationale behind selecting Trello as the project management tool and highlight the benefits of employing Kanban for project execution. It will outline the key phases of the AFNT app development and describe the division of the project into sprints, focusing on each sprint's objectives.

Additionally, the chapter will delve into the integration of Trello Kanban into the project, explaining the structure of the Kanban board and its role in organizing and tracking tasks. Lastly, it will discuss the advantages of using Trello Kanban, emphasizing its simplicity, scalability, transparency, and customization capabilities. Through these insights, the methodology chapter will provide a comprehensive understanding of the strategies and tools utilized in the AFNT project for effective project management.

5.1. AFNT Phases

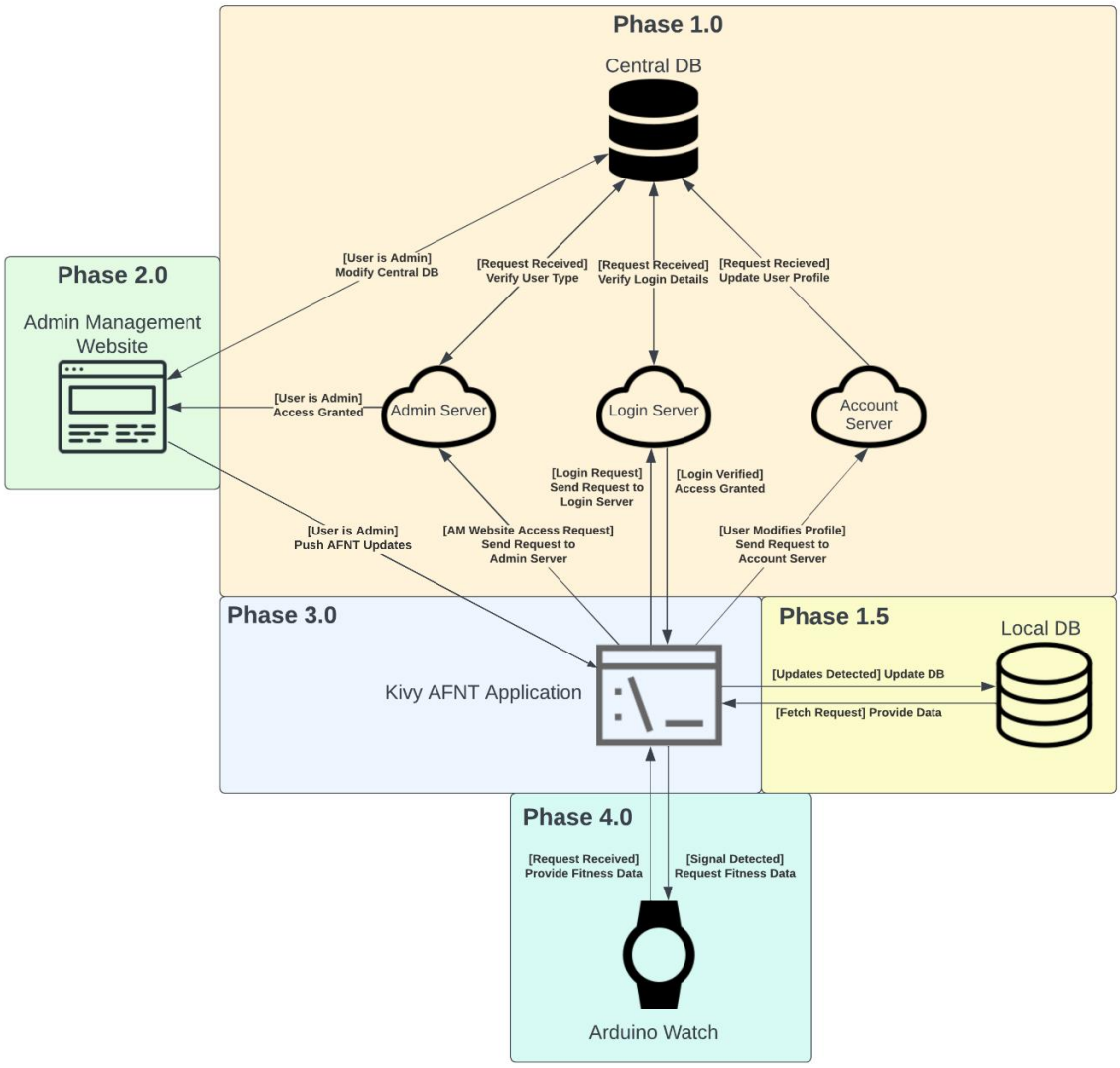


Figure 22: AFNT Project Phases.

The AFNT project is divided into four key phases:

Table 17: AFNT Phase Descriptions.

Phase 1	Phase 1 involves setting up the Central Database (CDB) within the Database Management System (DBMS). The CDB stores predefined workouts, meals, and user login credentials securely. Updates made in the CDB are synced to the user's Local Database (LDB) upon successful login.
Phase 1.5	The LDB is introduced to store user-specific workout, meal, and body statistics data locally. This data remains unaffected by updates from the CDB, ensuring user customization is preserved.
Phase 2	introduces the Admin Management Website (AM), allowing administrators to manage the CDB and push updates to the AFNT application.
Phase 3	focuses on developing the AFNT application, enabling users to track workouts, meals, and body data. Users can also modify their profiles, with changes reflected in the CDB.
Phase 4	Phase 4 extends Phase 3 by introducing the Arduino Fitness Watch, capable of monitoring health metrics and seamlessly transferring data to the AFNT application via Bluetooth or a wired connection.

5.2. Sprint Planning

Sprint planning is a crucial aspect of project management, especially in agile methodologies like the one employed in the AFNT Project. It involves breaking down the project into smaller, manageable chunks known as sprints, each focusing on specific objectives.

In the case of the AFNT Project, sprint planning has resulted in the division of the project into four distinct sprints, with each sprint dedicated to a particular phase of development. However, it's important to note that due to time constraints and priority considerations, Phase 2 of the project will be largely disregarded. Below are the details of each sprint and its corresponding focus:

Table 18: Sprint 1 – Planning & Design.

At Risk?	TASK NAME	PHASE TYPE	START DATE	END DATE	DURATION in days	STATUS	PRIORITY	COMMENTS
	SPRINT 1 – PLANNING & DESIGN		05/10/2023	31/12/2023	88			
Yes	Write down requirements	Planning	05/10/2023	16/10/2023	12	Complete	High	
Yes	Project Proposal	Planning	20/10/2023	25/10/2023	6	Complete	High	
Yes	Decide Arduino Watch Component	Planning	21/10/2023	26/10/2023	6	Complete	High	
Yes	Pre-Ethical Checklist	Planning	06/11/2023	06/11/2023	1	Complete	High	Received First Arduino Watch Parts on 04/11/2024
Yes	Ethical Checklist	Planning	06/11/2023	06/11/2023	1	Complete	High	
Yes	Research Kivy GUI	Planning	05/11/2023	02/12/2023	28	Complete	High	
No	Design AFNT Logo	Planning	14/12/2023	14/12/2023	1	Complete	Medium	
No	Create Usecase diagrams for Admin Management Website	Design	07/12/2023	15/12/2023	9	Complete	High	
No	Create Usecase diagrams for AFNT App	Design	07/12/2023	15/12/2023	9	Complete	High	
No	Create Usecase diagrams for Arduino Watch	Design	07/12/2023	15/12/2023	9	Complete	High	
Yes	Create LocalDB Schema	Design	16/12/2023	20/12/2023	5	Complete	High	
Yes	Create CentralDB Schema	Design	16/12/2023	20/12/2023	5	Complete	High	
No	Create AFNT Architecture Diagram	Design	21/12/2023	21/12/2023	1	Complete	High	
Yes	Setup Login Server	Server	21/12/2023	25/12/2023	5	Complete	High	
Yes Yes	Setup Admin Login Server Setup Account Server	Server Server	21/12/2023 21/12/2023	25/12/2023 25/12/2023	5	Complete Complete	High High	
Yes	Create Class diagram for AFNT App	Design	22/12/2023	26/12/2023	5	Complete	High	
Yes	Create Class diagram for Central and Local DB	Design	22/12/2023	26/12/2023	5	Complete	High	
No	Create AFNT UI Sketches	Design	27/12/2023	29/12/2023	3	Complete	High	Delayed to 16/01/2024
No	Create AMWebsite UI Sketches	Design	29/12/2023	30/12/2023	2	Complete	High	Delayed to 16/01/2024
No	Add Usecase descriptions	Design	30/12/2023	31/12/2023	2	Complete	High	Delayed to 16/01/2024

Table 19: Sprint 2 – Phase 1 Implementation & PIP

At Risk?	TASK NAME	PHASE TYPE	START DATE	END DATE	DURATION in days	STATUS	PRIORITY	COMMENTS
	SPRINT 2 – PHASE 1 IMPLEMENTATION & PIP		01/01/2024	31/01/2024	31			
Yes	[# 1] AMW Login Screen	Website	01/01/2024	03/01/2024	3	Complete	High	
Yes	Implement Central DB	Database	04/01/2024	10/01/2024	7	Complete	High	
Yes	Implement LocalDB	Database	04/01/2024	10/01/2024	7	Complete	High	Implemented Phase 1 and 1.5
Yes	[# 1] AFNT Login Screen	Application	11/01/2024	16/01/2024	6	Complete	High	
Yes	[# 1] AFNT Registration Screen	Application	16/01/2024	18/01/2024	3	Complete	High	
Yes	Create Poster	PIP	15/01/2024	21/01/2024	7	Complete	High	
Yes	Complete Video Demo	PIP	16/01/2024	22/01/2024	7	Complete	High	
Yes	Assemble Arduino Watch	Phase 4	23/01/2024	23/01/2024	1	Complete	High	Final Arduino Watch parts arrived on 23/01/2024
Yes	Test second batch of Arduino Watch	Phase 4	23/01/2024	23/01/2024	1	Complete	High	
Yes	Finish Literature Review	Report	15/01/2024	24/01/2024	10	Complete	High	
Yes	PIP Demo	PIP	24/01/2024	24/01/2024	1	Complete	High	

Table 20: Sprint 3 – Phase 3 Implementation.

At Risk?	TASK NAME	PHASE TYPE	START DATE	END DATE	DURATION in days	STATUS	PRIORITY	COMMENTS
	SPRINT 3 – PHASE 3 IMPLEMENTATION		01/02/2024	23/03/2024	52			
Yes	[#2] AFNT Dashboard	Application	01/02/2024	02/02/2024	2	Complete	High	
Yes	[#23] Water Intake Screen	Application	01/02/2024	02/02/2024	2	Complete	High	
Yes	[#26] Water Intake Plots Screen	Application	01/02/2024	02/02/2024	2	Complete	High	
Yes	[#24] Body Statistics Screen	Application	03/02/2024	06/02/2024	4	Complete	High	
Yes	[#25] Body Statistics Plots Screen	Application	03/02/2024	06/02/2024	4	Complete	High	
Yes	[#4, #8, #12, #13] Workout History Screen	Application	07/02/2024	20/02/2024	14	Complete	High	
Yes	Fix Github problem	Github	21/02/2024	21/02/2024	1	Complete	High	
Yes	[#5, #11, #16] Exercise Log Screen	Application	21/02/2024	22/02/2024	2	Complete	High	
Yes	[#8, #10, #11, #12] Workout Allocate Screen	Application	26/02/2024	01/03/2024	5	Complete	High	
Yes	[#10] Workout Create Screen	Application	02/03/2024	03/03/2024	2	Complete	High	
Yes	[#14] Exercise Create Screen	Application	03/03/2024	03/03/2024	1	Complete	High	
Yes	[#11, #16] Exercise Allocate Screen	Application	04/03/2024	08/03/2024	5	Complete	High	
Yes	[#11] Exercise Log Allocate Screen	Application	09/03/2024	09/03/2024	1	Complete	High	
No	Gym Finder Screen	Application	08/03/2024	10/03/2024	3	In Progress	Low	
Yes	Update Usecase Diagrams	Design	15/03/2024	20/03/2024	6	Complete	Medium	Travelled to Kuwait on 16/03/2024 due to urgent personal matter
Yes	Update Class Diagrams	Design	21/03/2024	22/03/2024	2	Complete	Medium	
Yes	Report Draft Submission to Supervisor	Report	20/03/2024	23/03/2024	4	Complete	High	

Table 21: Sprint 4 – Phase 4 Implementation.

At Risk?	TASK NAME	PHASE TYPE	START DATE	END DATE	DURATION in days	STATUS	PRIORITY	COMMENTS
	SPRINT 4 – PHASE 4 IMPLEMENTATION		24/03/2024	13/05/2024	51			
Yes	Implement Step Count Functionality	Phase 4	24/03/2024	25/03/2024	2	Complete	High	
Yes	Implement Blood Oxygen Functionality	Phase 4	24/03/2024	25/03/2024	2	Complete	High	
Yes	Implement Heart Rate Functionality	Phase 4	25/03/2024	25/03/2024	1	Complete	High	
Yes	Implement Date and Time Functionality	Phase 4	25/03/2024	25/03/2024	1	In Progress	High	
Yes	Arduino Bluetooth Data Transfer	Phase 4	31/03/2024	03/04/2024	4	In Progress	High	
Yes	Arduino Screen	Phase 3	04/04/2024	13/04/2024	10	In Progress	High	Return flight to UK on 13/04/2024
Yes	Meal History	Phase 3	15/04/2024	15/04/2024	1	In Progress	Medium	
Yes	Food Item Log	Phase 3	16/04/2024	17/04/2024	2	Not Started	Medium	
Yes	Meal Allocate Screen	Phase 3	17/04/2024	18/04/2024	2	Not Started	Medium	
Yes	Meal Create Screen	Phase 3	19/04/2024	20/04/2024	2	Not Started	Medium	
Yes	Final Report	Report	20/04/2024	24/04/2024	5	In Progress	High	

Note: Click Here to access the sprint file.

5.3. Gantt Chart

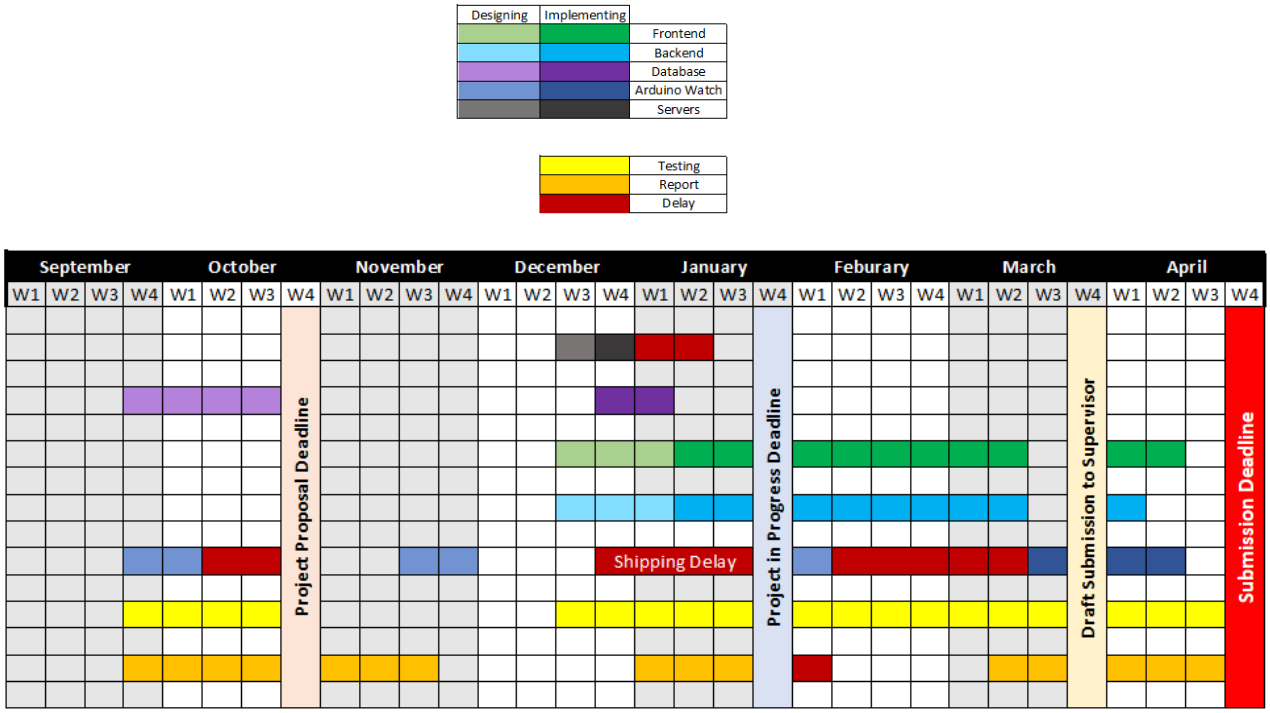


Figure 23: Gantt Chart.

Note: Click Here to access the Gantt chart file.

5.4. Trello Kanban Integration

Trello was chosen as the project management tool for its alignment with agile principles and its user-friendly interface, facilitating seamless tracking of tasks, progress, and priorities (Atlassian, 2019). This chapter explores the benefits of Trello Kanban methodology, its incorporation into the AFNT project, and its pivotal role in managing the project's advancement. Additionally, it outlines the essential phases of AFNT app development, providing an in-depth overview of the project's progression. Through the integration of agile methodologies, well-structured project phases, and proficient utilization of Trello Kanban, the AFNT project demonstrates a systematic and organized approach to software development.

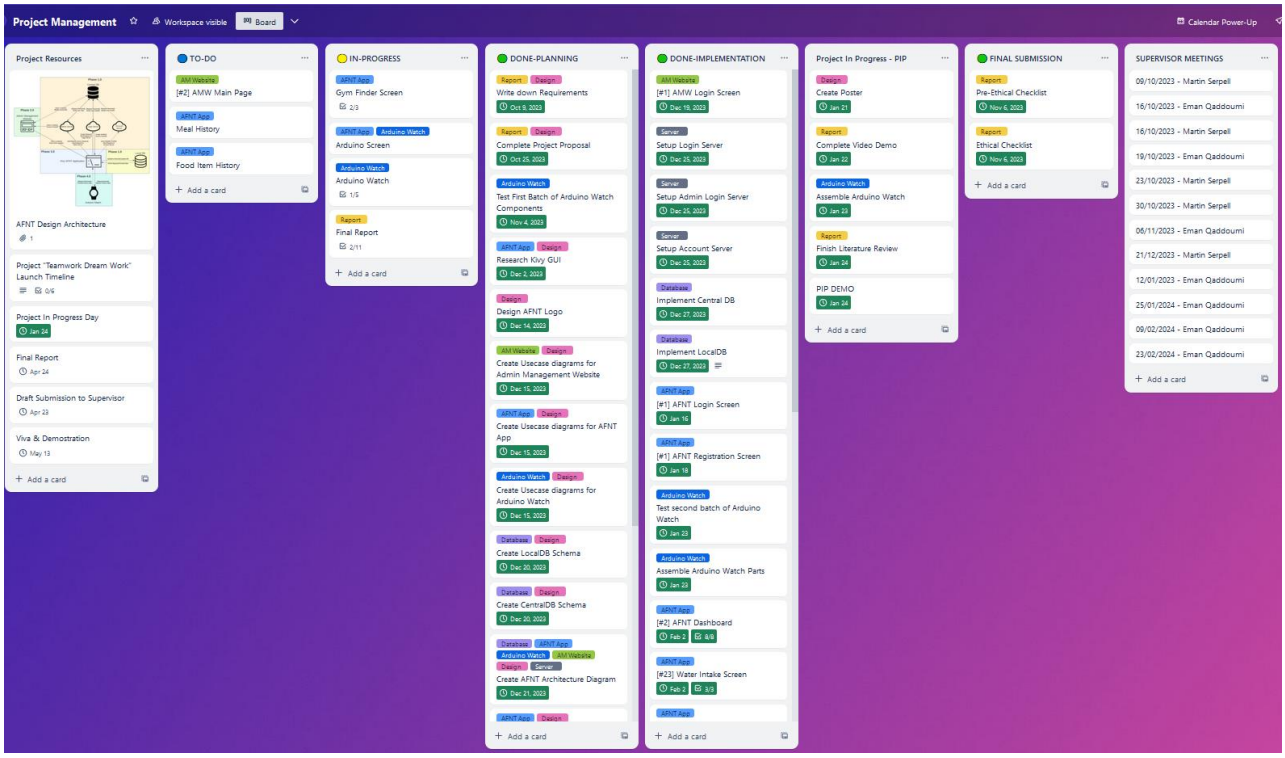


Figure 24: AFNT Project Kanban Board.

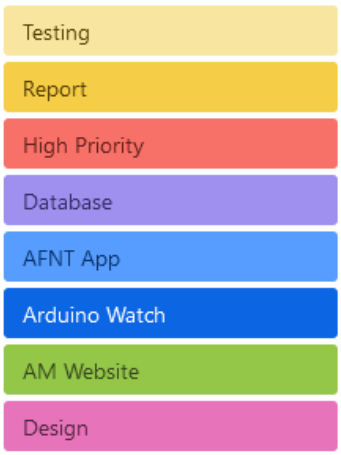


Figure 25: Project Labels on Trello.

Note: To access Trello dump data please navigate to Appendix B: AFNT Trello Dump.

The Kanban methodology operates on the fundamental principle of visualizing tasks and workflow on a board. This board is typically divided into columns representing different stages of the project. Each task or "card" progresses through these columns from initiation to completion. Trello, as a Kanban tool, provides a digital platform for teams to create boards, lists, and cards, facilitating collaboration and transparency in project management. Cards can be assigned to team members, labelled with priorities, and updated in real-time as progress is made.

In the case of the AFNT project, the Trello Kanban board consists of eight main columns with four columns specifically allocated to each sprint, as outlined in the section 5.2 each serving a specific purpose:

Table 22: Trello Kanban Column Descriptions.

PROJECT – RESOURCES	This column serves as a repository for all relevant and useful information, making it easy and convenient to access upcoming submission deadlines or supervisor meetings. It provides a centralized location for important project-related resources.
TO – DO	Tasks that have not yet been started or completed are placed in this column. It serves as a backlog of pending tasks that need to be addressed.
IN – PROGRESS	Tasks that are currently being worked on or are partially implemented are moved to this column.
SPRINT 1 – PLANNING & DESIGN	Once design/planning tasks are fully confirmed and planned, they are moved to this column. It signifies that the planning and design phase for these tasks is complete, and they are ready for implementation.
SPRINT 2 – PHASE 1 IMPLEMENTATION & PIP	This column is dedicated to Phase 1 (Database and Server) implementation. Including the submissions, deadlines, and demos related to the Project-In-Progress (PIP) session.
SPRINT 3 – PHASE 3 IMPLEMENTATION	Tasks that have been fully implemented from the AFNT app are moved to this column. It indicates that the implementation phase of these tasks is complete, and they are ready for testing or review.
SPRINT 4 – PHASE 4 IMPLEMENTATION	Tasks related to the Arduino watch development (Phase 4), final project submissions and demos, particularly those associated with the final deadline, are placed in this column. It ensures that all final deliverables are organized and accounted for as the project nears completion.
SUPERVISOR – MEETINGS	Important dates and details regarding supervisor meetings are noted in this column.

By utilizing these specific columns, the Trello Kanban board effectively organizes and tracks the progress of tasks throughout the project lifecycle, ensuring clarity, accountability, and efficiency in project management.

5.5. Trello Kanban Advantages

The decision to utilize Trello Kanban for the AFNT app was based on several factors:

Table 23: Trello Kanban Advantages.

Simplicity	Trello's user-friendly interface and intuitive design make it easy for team members to adopt and use effectively, without extensive training.
Scalability	Trello is highly scalable and can accommodate projects of various sizes and complexities. As the AFNT app project evolved, Trello could easily adapt to changing needs and requirements.
Transparency	Kanban methodology promotes transparency by providing visibility into the status of tasks and progress made. This transparency enhances accountability and ensures everyone is on the same page.
Customization	Trello allows for customization to fit the specific needs and preferences of the team. Boards, lists, and cards can be tailored to reflect the unique workflow and requirements of the AFNT app project.

The integration of Trello Kanban into the AFNT project offered an organized and effective project management strategy that aligned seamlessly with the adopted agile methodology. By leveraging Kanban boards and Trello's user-friendly interface, the project could visualize tasks, monitor progress, and prioritize activities with ease. The project's division into four distinct phases, each playing a vital role in the AFNT app's development, facilitated a structured approach, ensuring clarity and focus at every stage.

Moreover, the incorporation of use case and requirements gathering laid a solid framework for project design, aiding in the creation of wireframes and class diagrams, which will be further explored in the Desig chapter. Collectively, these elements, alongside Trello Kanban, agile methodology, and phased development, established a strong foundation for the successful implementation of the AFNT project.

6. Design

As previously outlined, the AFNT project is divided into four key phases: database management, website development, application development, and Arduino Watch hardware integration. Phase 1 focuses on establishing a robust DBMS. Phase 2 introduces the AM website for database administration, leading into Phase 3's development of the AFNT fitness tracking application. Phase 4 expands the project with the Arduino Watch, integrating modular components for health monitoring and data transfer to the AFNT app via Bluetooth or wired connection.

This chapter discusses detailed design documentation, including schemas, wireframes, diagrams, and specifications for each phase, providing a comprehensive overview of the project's structured approach to achieving its goals.

Note: Click Here to access all the design files (i.e. usecase, class, wireframes etc).

6.1. Phase 1: Database Design

Phase 1 focuses on establishing a Database Management System (DBMS) featuring a Central Database (CDB). The CDB is primarily tasked with storing predefined workouts, meals, and user login credentials to ensure secure access. Operating as an SQL Database Server, the CDB is exclusively accessible to administrators. Updates made to the CDB are synchronized with the user's Local Database (LDB) upon successful user login. It's important to note that the Central Database solely manages predefined workouts and meals provided to users by default and does not store any user-specific workout or meal data. Custom workouts and meals created by users remain unaffected by changes pushed from the CDB.

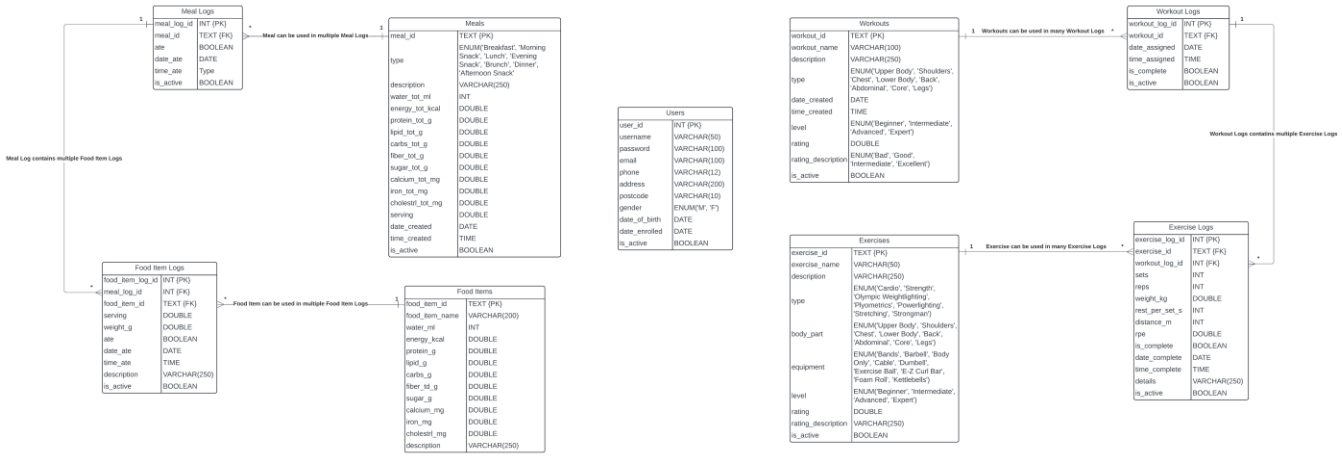


Figure 26: Central Database Schema.

Table 24: Central and Local Database Schema Descriptions.

Table Name	Description
Workouts	Workouts Contains details of various workouts, such as the workout name and type. For instance, a workout might be categorized as "legs."
Workout Logs	Stores records of individual workout sessions created using the definitions from the Workouts table. This allows for the creation of different workout logs based on the definitions provided. For example, the workout 'legs' can be allocated by user on any day of the week, and once the workout is completed, the user can set the completion status as "complete".
Exercises	Holds definitions for various exercises, with each exercise being associated with a specific workout. For example, the exercise "barbell squats" can be allocated to the "legs" workout. This dataset was imported from Kaggle (Pandit, 2022).
Exercise Logs	Exercise Log: Records detailed information for each exercise, including the number of sets, repetitions, and rest periods. This data corresponds to exercises defined in the Exercises table and assigned to specific workouts. For instance, the "barbell squats" exercise may be designated for the "legs" workout, with specifications such as 3 sets of 10 repetitions and a 90-second rest period between sets.
Meals	Meals Contains definitions for different meal types, such as "Joe's breakfast," categorized by meal type (e.g., breakfast). This data was imported from Kaggle (vinitshah0110, 2022).
Meal Logs	Utilizes meal definitions from the Meals table for allocation purposes. Users can select meals from this table and allocate them to specific dates, creating new meal logs. For example, "Joe's breakfast" can be allocated on 16th March 2024 and the user can set the meal status as "complete."
Food Items	Stores definitions and nutritional information for various food items. For instance, "egg" is a food item, and its nutritional contents are recorded in this table.
Food Item Logs	Allow users to allocate food items from the Food Items table to meals and specify serving sizes or quantities. For example, users can add the food item "egg" and adjust the serving size as needed.
Users	Stores personal information for each user, including username, encrypted password, gender, address, and contact details. Gender information is used for recommending workouts and meals tailored to the user's gender, address data helps in locating nearby gyms, and contact details are utilized for verification and two-factor authentication (currently out of scope).

It's important to note that the deletion of any preset meal or workout does not impact existing meal or workout logs. Rather than removing the record entirely from the database, the system updates the "is_active" column of the record to 0, indicating that it is no longer active. As a result, users will no longer be able to view or access this record, but the associated logs will remain intact in the database.

6.1.1. Local Database

Phase 1.5 expands the Database design by introducing the Local Database (LDB), dedicated to storing user-specific workout, meal, and body statistics data. Unlike the Central Database (CDB), LDB stores data locally, accessible only to the respective user. Utilizing SQLite, LDB comprehensively captures all user-related information, ensuring privacy and security.

Upon user login, any updates made in the Central Database are seamlessly synchronized with LDB, ensuring that users have access to the latest predefined meals and workout data. However, these updates do not impact user-specific custom workouts and meals, preserving their integrity. Even past user entries remain unaffected by CDB updates.

Furthermore, while both CDB and LDB share the same schema for fundamental components such as users, workouts, exercises, meals, and food items, LDB includes additional tables specifically tailored for collecting and managing body data. These tables will be elaborated in this section.

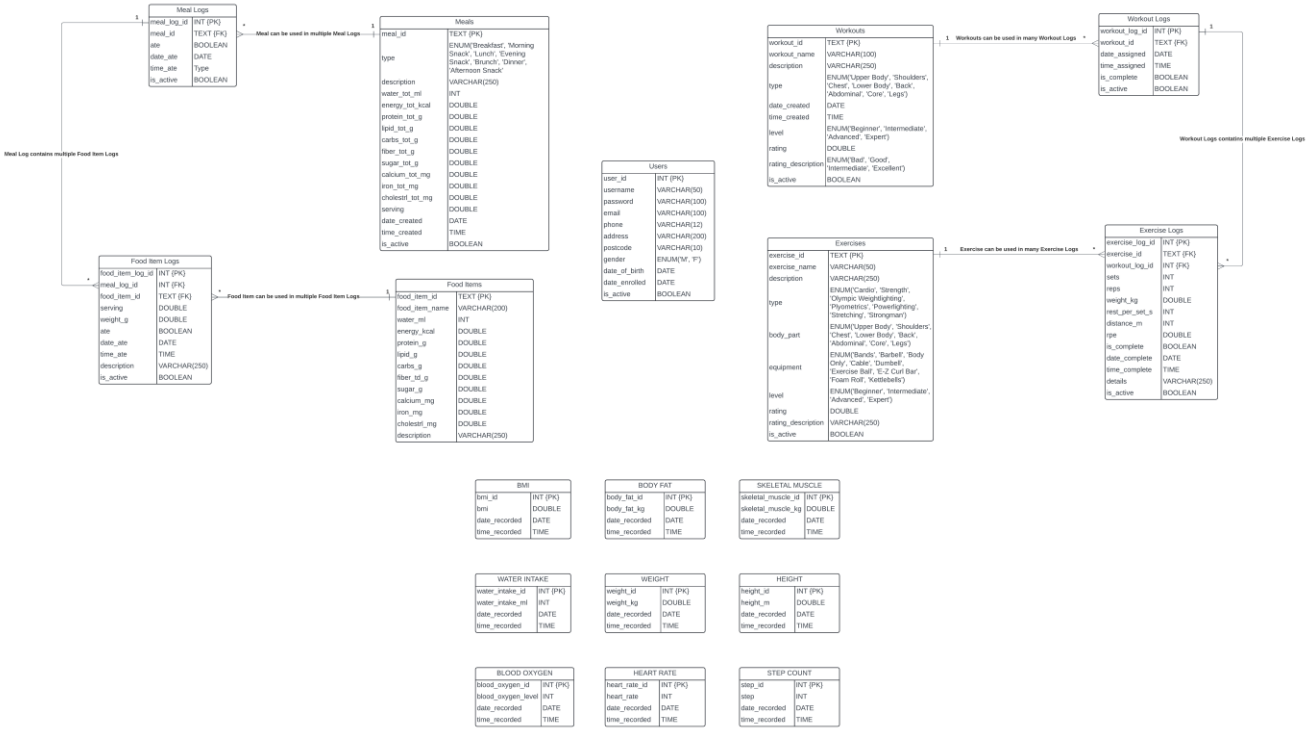


Figure 27: Local Database Schema.

Table 25: Local Database Schema Descriptions.

Table Name	Collection Method	Description
BMI	User Input or Automatically calculated based on provided weight and height data.	Stores BMI data and the date/time of the record creation. It is calculated using the formula ('BMI Calculator', 2024): round(weight / ((height/100) * (height/100), 1)
Body Fat	User Input or Automatically calculated based on the provided age and gender data.	Stores Body Fat data and the date/time of the record creation. It is calculated using the formula ('Body Fat Calculator', 2017): **Males**: round(1.20 * bmi) + 0.23 * age - 16.2, 1) **Females**: round(1.20 * bmi) + 0.23 * age - 5.4, 1)
Skeletal Muscle	User Input	Stores Skeletal Muscle (Muscle Mass) and the date/time recorded.
Water Intake	User Input	Stores the water intake in millilitres (ml) and the date/time recorded.
Weight	User Input	Stores the weight in kilograms (kg) and the date/time recorded.
Height	User Input	Stores the height in metres (m) and the date/time recorded.
Blood Oxygen Level	Arduino Watch (Pulse Oximeter Sensor)	Stores the blood oxygen in percentage (%) and the date/time recorded.
Heart Rate	Arduino Watch (Pulse Oximeter Sensor)	Stores the Heart Rate data in beats per minute (bpm) and the date/time recorded. This data will collected using the Arduino Watch's Pulse Oximeter Sensor.
Step Count	Arduino Watch (Accelerometer) and User Input	Stores the step count of user per day and the date recorded. This data can be collected using the Arduino Watch.

6.1.2. Servers

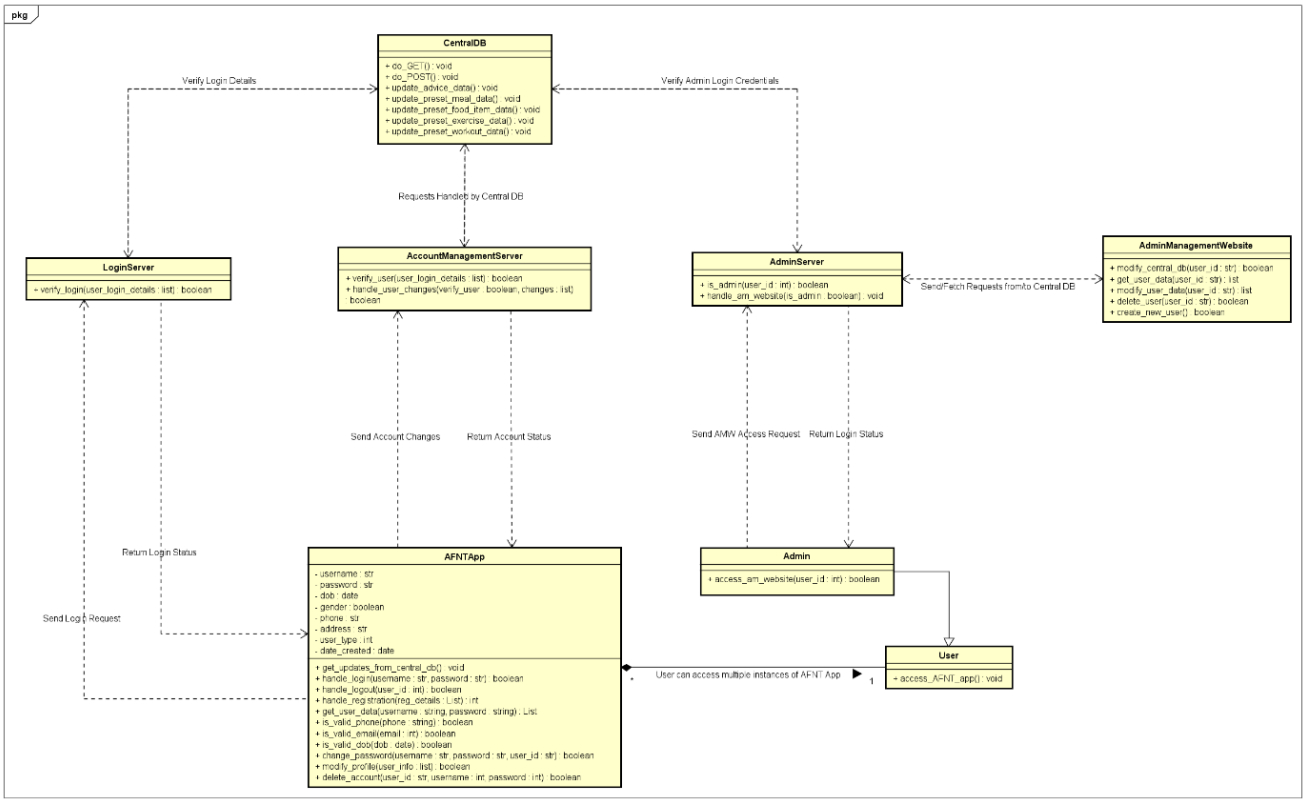


Figure 28: Central Database and Admin Website Class Diagram.

Table 26: Database Server Descriptions.

Class	Description
LoginServer	Responsible for authenticating user login credentials within the AFNT app.
AccountManagementServer	Manages user profile and personal details, facilitating any necessary updates or modifications.
AdminManagementServer	Validates admin login credentials and facilitates database changes to the Central Database (CDB).

6.2. Phase 2: Admin Management Website Design

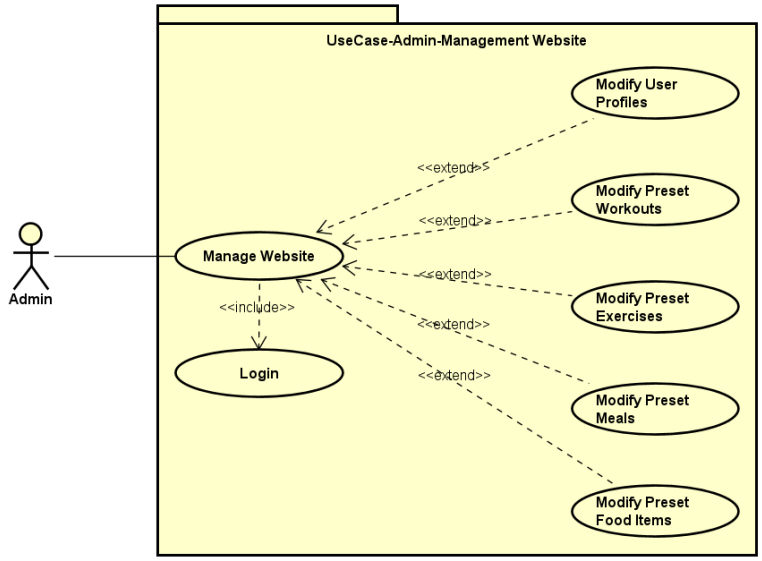


Figure 29: Admin Management Website Usecase Diagram.

Use Case Diagram	Admin Management Website
Use Case Identifier	Manage Website and the <<extend>>
Goal	Admins and modify the Central Database and push updates to AFNT app
Priority	M
Updated	21/03/2024
Participating Actors	Admin
Pre-conditions	Admins needs to login, upon a successful login Admins are directed to the admin dashboard
Post-conditions	Admins need to navigate to the 'Edit Central DB' tab
Basic Flow of Events	1. Admin logs on the website 2. Upon a successful login, admins are redirected to the Admin Dashboard. 3. Admins can navigate to 'Edit Central DB' and can modify its contents (consists of presets meals and workout data, and user login data). 4. After implementing successful changes, admins can push this and the LocalDB will import the new centralDB updates when user logins successfully.
Alternative Flow	None

Table 27: Admin Management Website Usecase Description.

Phase 2 introduces the Admin Management Website (AM), which serves as the centralized platform for administering the Central Database (CDB) and facilitating updates to the AFNT application. This web-based interface is exclusively accessible to administrators, empowering them to manage and modify CDB content effectively. To gain access to the AM website, Admins will have to sign in, and the login details will be validated using the Admin Management server that securely communicates with the CDB. After login details are successfully validated, they can access the AM website.

6.2.1. Admin Management Website Wireframes

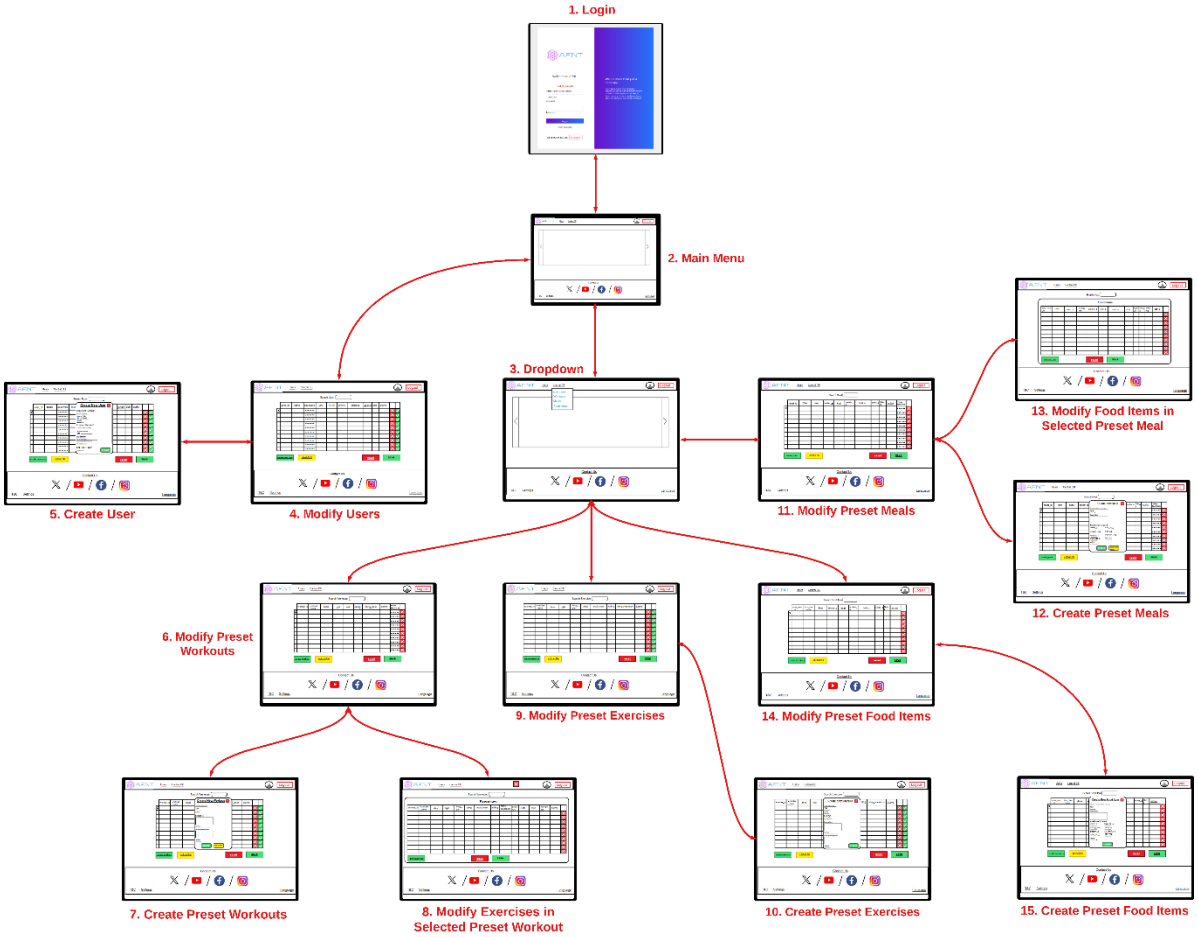


Figure 30: Admin Management Website Wireframes.

6.3. Phase 3: AFNT Application Design

Phase 3 of the AFNT project focuses on developing the AFNT application, which aims to provide users with a comprehensive set of features for tracking workouts, managing meals, and monitoring body data. This section will explore three key aspects of the application's design: wireframe exploration, application structure using the Kivy GUI framework, and class definitions.

Wireframes will outline the visual layout and user interface elements, while the application's architecture implemented with Kivy GUI will ensure a dynamic and responsive user experience. Finally, class definitions will encapsulate data and behaviour into reusable components, contributing to the application's functionality and modular design. Together, these components will provide insights into the AFNT application's design and implementation process.

Workout Data: Workout data encompasses all workout logs/sessions generated by users, including custom workouts and associated exercise logs. Each workout log is linked to specific exercises defined within custom workouts.

Meal Data: Meal data includes all meal logs/sessions created by users, covering custom meals and associated food item logs. Each meal log is associated with specific food items defined within custom meals.

Body Statistics Data: Body statistics data captures performance, growth, and progress metrics vital for users to track their fitness goals. Key metrics initially focused on by AFNT include Step Count, BMI, Body Fat, Skeletal Muscle, Heart Rate, and Blood Oxygen Level.

6.3.1. Application Wireframes

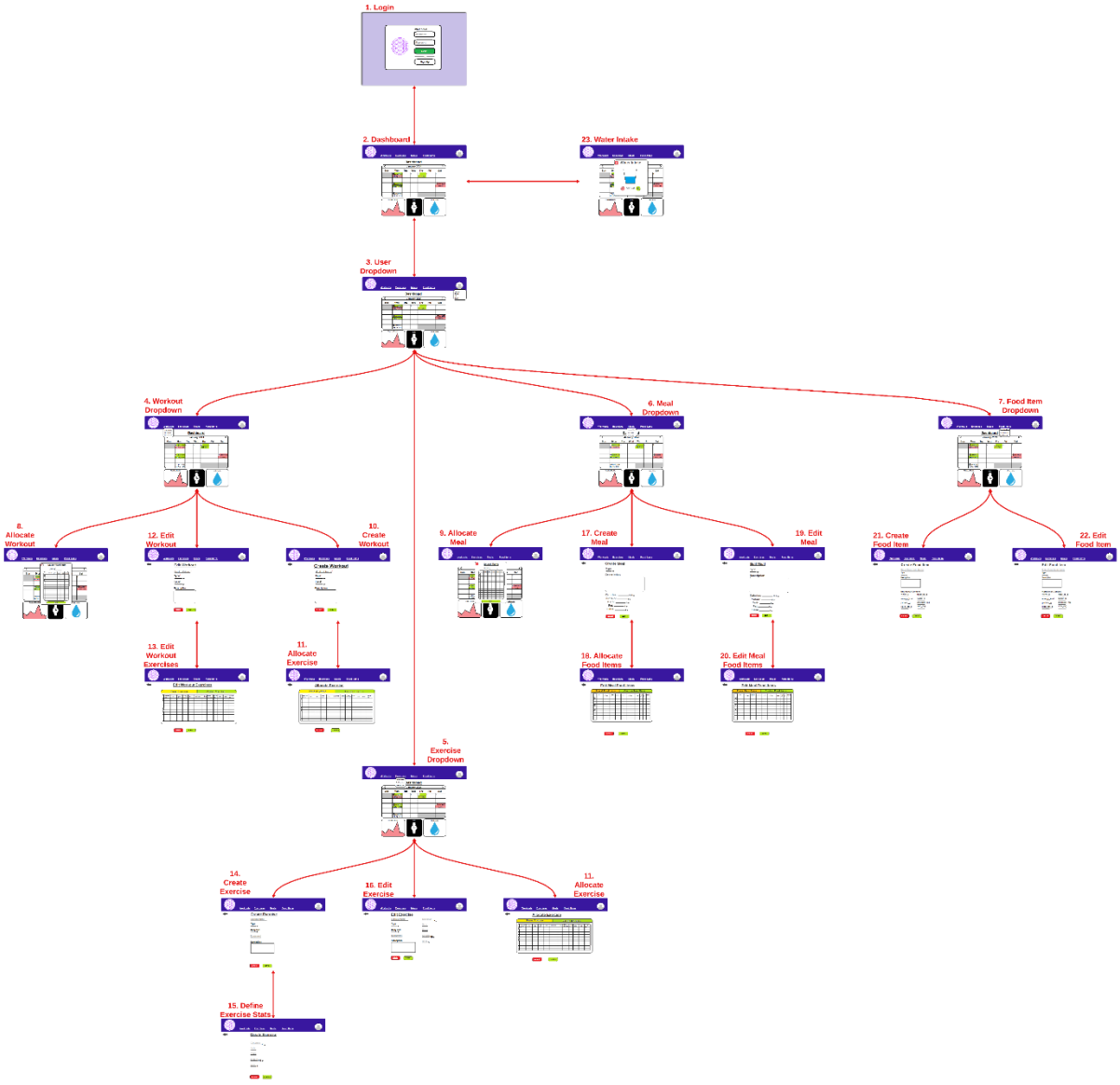


Table 28: AFNT App Wireframes.

Wireframes played a crucial role in the development of the AFNT application by providing a visual representation of its layout and structure. These skeletal outlines serve as blueprints for the app's user interface, making it easier to conceptualize and refine its functionality before proceeding with the actual development. By outlining the placement of various elements such as buttons, menus, and content sections, wireframes help ensure intuitive navigation and user-friendly interactions. Overall, wireframes are instrumental in streamlining the design process, reducing development time, and helping to ultimately enhance the user experience of the AFNT app.

6.3.2. Kivy GUI Class Structure

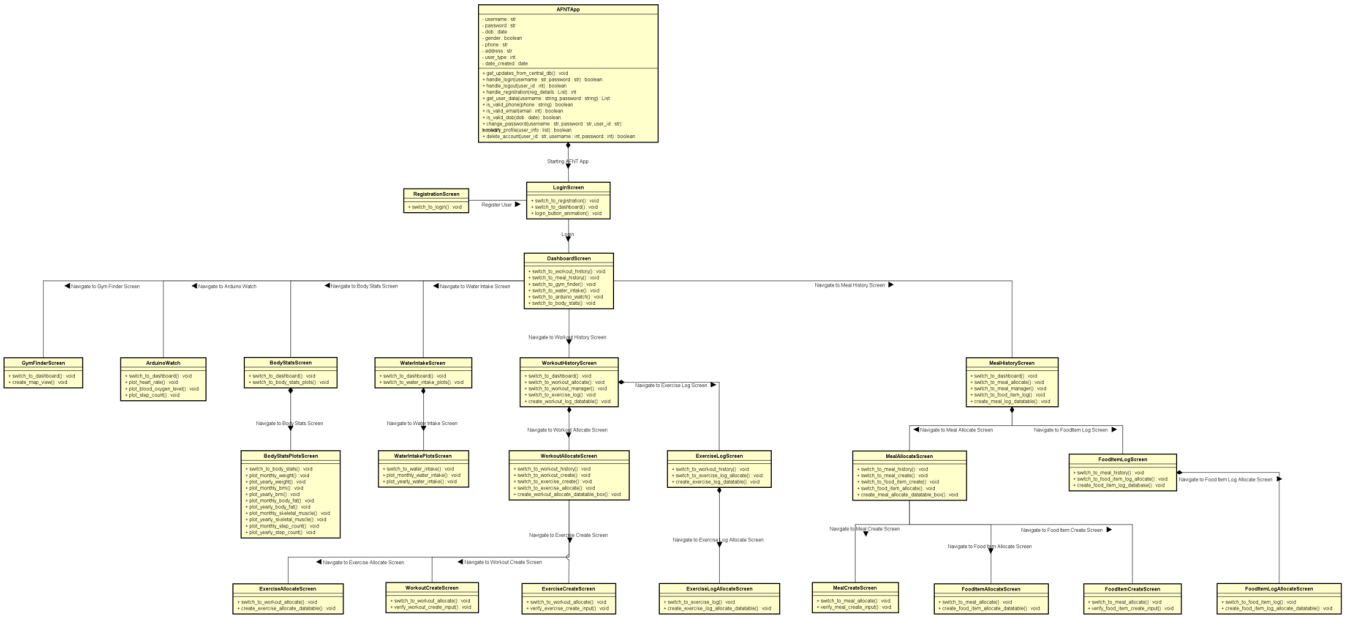


Figure 31: AFNT GUI Class Diagrams.

Using the wireframes described in 6.3.1, a comprehensive class diagram was devised leveraging the Kivy ScreenManager module to craft an intuitive and user-friendly UI coupled with an efficient screen management system. The class diagram encompasses 25 distinct Screen classes, each meticulously designed to execute specific tasks and functionalities, as elucidated in the following tables.

Note: The acronym ARMV will signify the ability to Add, Remove, Modify, or View a specific class component within the AFNT framework.

Table 29: AFNT Main Kivy GUI Class Definitions.

Screen Name	Pre-condition	Description
LoginScreen	Launch AFNT App	Managing login operation.
RegistrationScreen	Navigate from LoginScreen	Managing Registration operation.
DashboardScreen	Successfully Login	Contains options and buttons to navigate the AFNT App.
WaterIntakeScreen	Navigate from DashboardScreen	Responsible for managing Water Intake data (ARMV).
WaterIntakePlotsScreen	Select month and year then click 'Generate Graph'	Generates two water intake graphs, one for per selected month and one for selected year (average Water Intake per month).
BodyStatsScreen	Navigate from DashboardScreen	Responsible for managing Body Statistics data (ARMV).
BodyStatsPlotsScreen	Select month and year then click 'Generate Graph' on the selected Body Statistic	Generates two selected body stats graphs, one for per selected month and one for selected year.
GymFinderScreen	Navigate from DashboardScreen	Loads a map and displays the nearest gyms to user's address/location in a 10mile radius, range can be modified.

Table 30: AFNT Workouts and Exercises Kivy GUI class definitions.

Screen Name	Pre-condition	Description
LoginScreen	Launch AFNT App	Managing login operation.
WorkoutHistoryScreen	Navigate from DashboardScreen	Responsible for handling Workout Log data (ARMV).
ExerciseLogScreen	Select a Workout Log from WorkoutHistorySceen	Responsible for managing Exercise Log data (ARMV).
ExerciseLogAllocateScreen	Select 'Allocate Exercise' from ExerciseLogScreen	Responsible for allocating Exercise Logs to the selected Workout Log.
WorkoutAllocateScreen	Click 'Workout Manager' from WorkoutHistoryScreen	Responsible for allocating Workouts on selected date and time. Also manages Workout data (ARMV).
WorkoutCreateScreen	Click 'Create Workout' from WorkoutHistoryScreen	Responsible for creating Workouts.
ExerciseCreateScreen	Click 'Create Exercise' from WorkoutHistoryScreen	Responsible for creating new Exercises.

Table 31: AFNT Meals and Food Items Kivy GUI class definitions.

Screen Name	Pre-condition	Description
MealHistoryScreen	Navigate from DashboardScreen	Responsible for handling Meal Log data (ARMV).
FoodItemLogScreen	Select a Meal Log from MealHistorySceen	Responsible for managing Food Item Log data (ARMV).
FoodItemLogAllocateScreen	Select 'Allocate Food Item' from FoodItemLogScreen	Responsible for allocating Food Item Logs to the selected Meal Log.
MealAllocateScreen	Click 'Meal Manager' from MealHistoryScreen	Responsible for allocating Meals on selected date and time. Also manages Meal data (ARMV).
MealCreateScreen	Click 'Create Meal from MealHistoryScreen	Responsible for creating Meals.
FoodItemCreateScreen	Click 'Create Food Item' from MealHistoryScreen	Responsible for creating new Food Item.
MealPlotsScreen	Select month and year, then click 'Generate Graph' from MealHistoryScreen.	Responsible for generating meal plots.

6.3.3. Class Structure

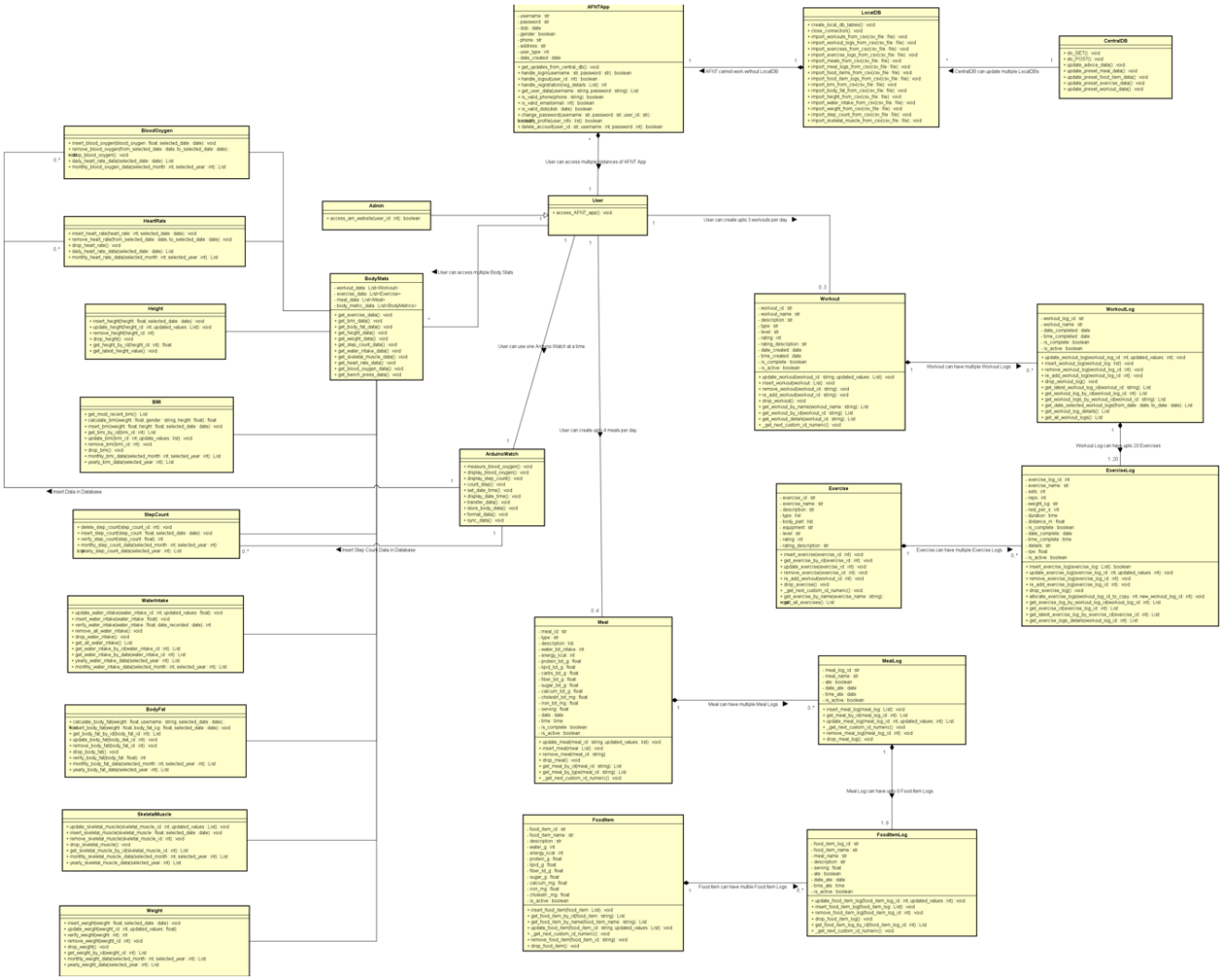


Figure 32: AFNT Class Diagram.

There are 24 classes dedicated to managing each aspect of the AFNT project. Their descriptions are provided in the below table.

Table 32: AFNT Class descriptions.

Class	Description
AFNTApp	Responsible for launching the AFNT App and handling login, registration, and account management requests
LocalDB	Responsible for handling importation of data and includes the database schema.
CentralDB	Responsible for handling database operations and containing the database schema.
User	Responsible for user operations
Admin	Responsible for CDB management and inherits all functionalities from the User class.
ArduinoWatch	Manages the data received from the Arduino watch. Also deals with connection/synchronising with the Arduino Watch.
Workout	Manages all database operations for Workouts table in LDB.
WorkoutLog	Manages all database operations for Workout Logs table in LDB. This class has a composite relationship with the Workout class.
Exercise	Manages all database operations for Exercises table in LDB.
ExerciseLog	Manages all database operations for Exercise Logs table in LDB. This class has a composite relationship with the Exercise and WorkoutLog class.
Meal	Manages all database operations for Meals table in LDB.
MealLog	Manages all database operations for Meal Logs table in LDB. This class has a composite relationship with the Meal class.
FoodItem	Manages all database operations for Food Items table in LDB.
FoodItemLog	Manages all database operations for Food Item Logs table in LDB. This class has a composite relationship with the MealLog and FoodItem class.
BodyStats	Fetches all body-related data from LDB.
BloodOxygen	Manages all functionalities related to Blood Oxygen.
HeartRate	Manages all functionalities related to Heart Rate. This data is only collected by the Arduino Watch.
Height	Manages all functionalities related to Height.
Weight	Manages all functionalities related to Weight.
BMI	Manages all functionalities related to BMI.
StepCount	Manages all functionalities related to Step Count.
WaterIntake	Manages all functionalities related to Water Intake.
BodyFat	Manages all functionalities related to Body Fat.
SkeletalMuscle	Manages all functionalities related to Skeletal Muscle.

6.4. Phase 4: Arduino Watch Design

Phase 4 expands on Phase 3 with the introduction of the Arduino Watch, capable of monitoring health metrics such as heart rate, blood oxygen level, and step count. The watch can transfer data to the AFNT application via Bluetooth or a wired connection. To understand the functionality of the Arduino Watch, it's important to understand the roles of the TinyDuino and TinyShield components.

6.4.1. TinyDuino and TinyShield

The TinyDuino Platform is a compact open-source electronics platform based on Arduino technology. It consists of a TinyDuino processor board and TinyShields, which add functionalities like sensors, communications, and displays. These boards stack together using 32-pin connectors without soldering, creating a customizable stack for various projects. The TinyDuino Processor Board, similar to an Arduino Uno but much smaller, uses the Atmel Atmega328P microcontroller and supports 14 digital I/O pins (6 PWM) and 6 analogue inputs. The main difference is TinyDuino's use of an 8MHz ceramic resonator instead of 16MHz, allowing operation down to 2.7 Volts, ideal for battery-powered applications (‘TinyDuino Overview’, 2024). Each TinyDuino module is designed to include essential circuitry for specific functions, optimizing size and cost.

The Arduino Watch is composed of 8 modular electronic components based on the TinyDuino platform, which can be connected like LEGO bricks. These modular components were selected for their compact size and portability, unlike larger standard Arduino boards and components that require soldering. Although these smaller modular components are more expensive, they save significant assembly and configuration time. This section will provide detailed descriptions of each modular component used in the Arduino Watch, along with their technical capabilities and schematics

Table 33: Arduino Watch Modular Electronic Components.

Modular Components	Usage in Arduino Watch
TinyScreen+ (Processor, OLED & USB) (ASD2431-R)	OLED display for the Arduino Watch
Lithium ION Polymer Battery (ASR00007)	3.7v 290mAh battery for the Arduino Watch
MicroSD TinyShield (ASD2201-R)	Storing data collected by the sensor wirelings
Wireling Adapter TinyShield (ASD2022)	A 4-channel Multiplexer, used to connect the sensor wirelings
Accelerometer Sensor Wirling (AST1001)	Step count measurement.
Accelerometer TinyShield (ASD2511-R-A)	Also used for step count measurement and connected to the Accelerometer Wireling
Pulse Oximeter Sensor Wireling	Heart rate, blood oxygen level, and body temperature measurement.
Bluetooth Low Energy TinyShield (ST)	Bluetooth 4.1, used to connect Arduino Watch to smartphones and desktops to transfer data to the AFNT apps

6.4.2. TinyScreen+ OLED Screen

The OLED screen used for the Arduino Watch measures 0.96 inches diagonally and has a resolution of 96x64 RGB pixels, capable of displaying 16-bit colour with vivid brightness and excellent colour reproduction, thanks to its OLED technology which eliminates the need for a backlight. Alongside the screen, there are four buttons positioned on the side to facilitate user input. The OLED display is driven by an SSD1331 driver, ensuring efficient and effective rendering of graphics and text on the screen (‘TinyScreen | TinyScreen’, 2024).

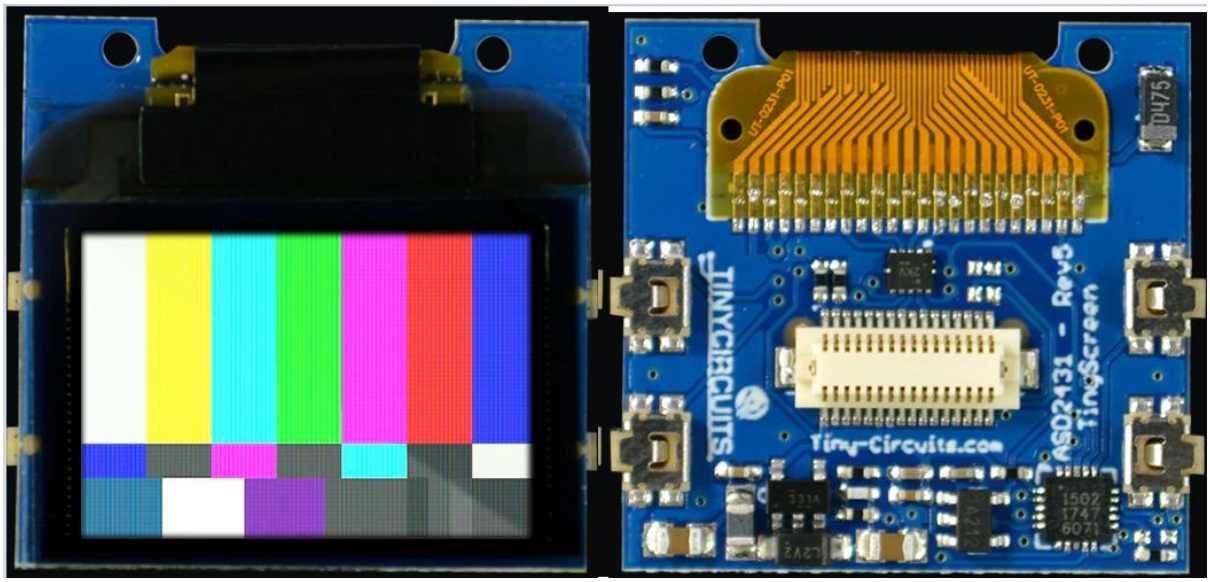


Figure 33: TinyScreen+ OLED (ASD2431-R).

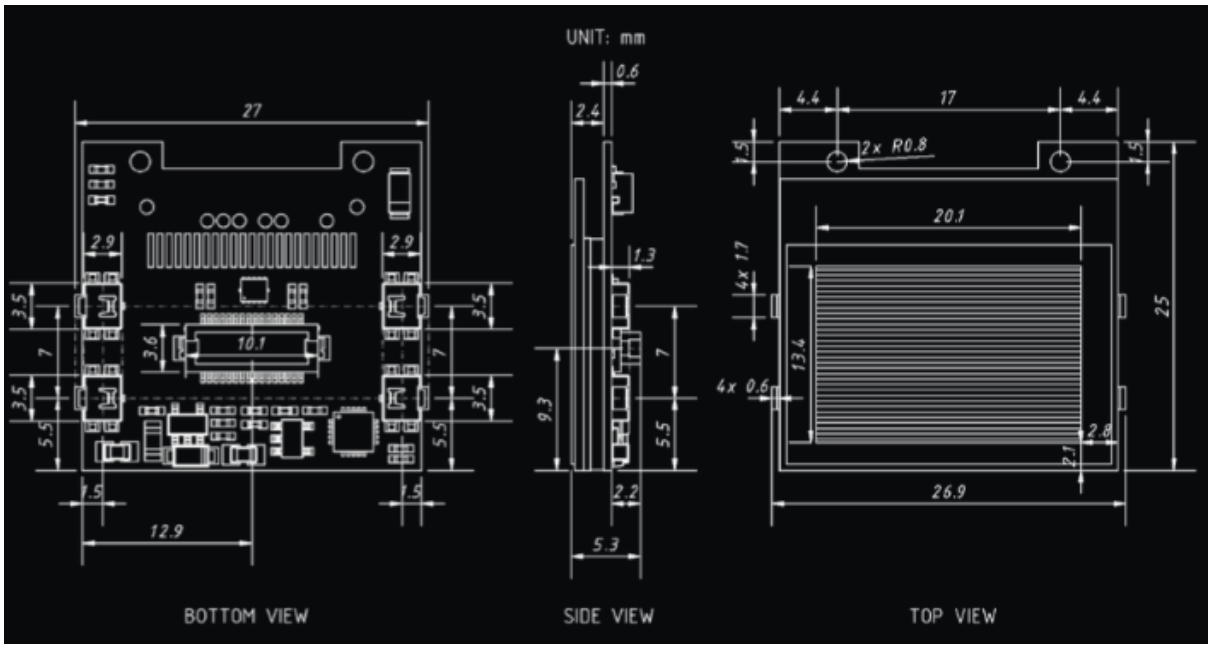


Figure 34: TinyScreen+ Dimensions.

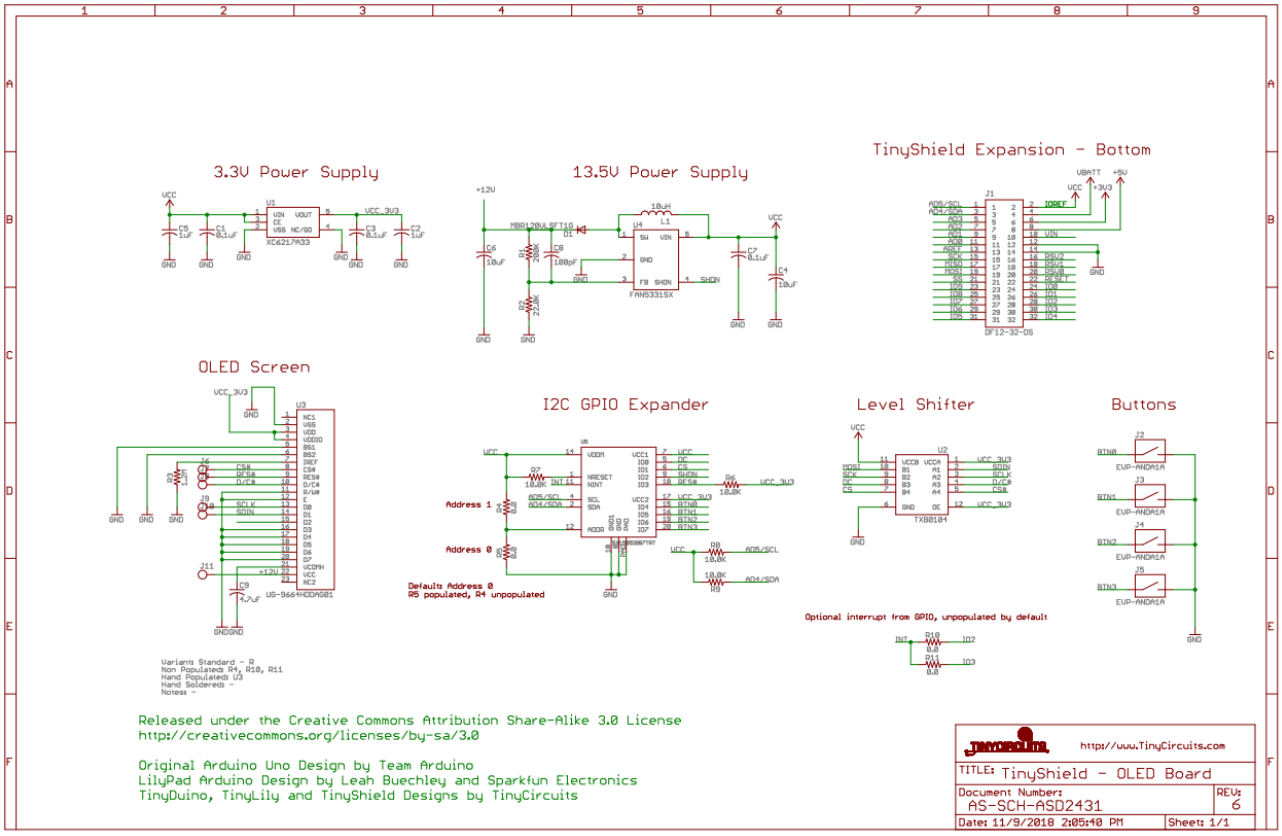


Figure 35: TinyScreen+ Schematic.

TinyScreen Specs

- 96x64 OLED display, 16-bit color depth
- 0.96" (24.4mm) viewable area
- Total Size: 1.02" x 0.98" (25.8mm x 25.0mm)
- SSD1331 display controller
- Software controllable backlight (OLED brightness)
- Power down mode
- Four push buttons along the sides (connected to IO pins)

TinyDuino Power Requirements

- Voltage: 3.0V - 5.5V
- Current: 20 - 45mA (depending on brightness), due to the current requirements, this board cannot be run using the TinyDuino coin cell option

Pins Used

SPI and I2C Interface used:

- **A5/SCL** - I2C Serial Clock line
- **A4/SDA** - I2C Serial Data line
- **11 - SCLK:** This signal is the serial SPI clock out of the TinyDuino and into the TinyScreen.
- **13 - MOSI:** This signal is the serial SPI data out of the TinyDuino and into the TinyScreen.
- Note: SPI Chip select and button inputs are handled by the on-board I2C GPIO expander.

Dimensions

- Dimensions: 25.8mm x 25mm (1.01 inches x .984 inches)
- Max Height (screen surface to bottom of TinyShield connector): 4.40mm (0.173inches)
- Weight: 3.2 grams (.113 ounces)

Figure 36: TinyScreen+ Specs.

6.4.3. Lithium Ion Polymer Battery

This rechargeable lithium-ion polymer (LiPo) battery makes it suitable for the Arduino Watch. It has a capacity of 290mAh at 3.7V, with a voltage range from 4.2V when fully charged to 3.0V when fully discharged. The battery is equipped with protection circuitry to prevent overcharging or deep discharge, enhancing safety and longevity (‘TinyCircuits | Lithium Ion Battery’, 2024).

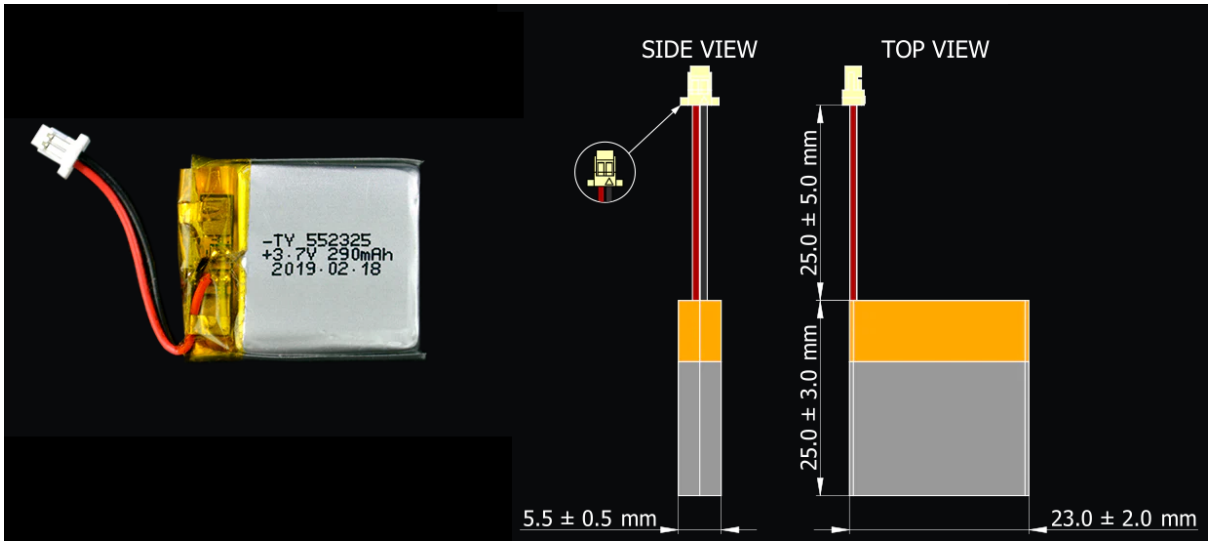


Figure 37: Lithium ION Battery (ASR00007) Dimensions.

- Nominal Voltage: 3.7V
- Nominal Capacity: 290mAh
- Max Charge Current: 290mA (1C)
- Max Discharge Current: 290mA (1C)
- Dimensions: 25.0±3.0mm x 23.0±2.0mm x 5.5±0.5mm (0.984±0.118" x 0.906±0.079" x 0.217±0.020")
- Wire Length: 25.0±5.0mm (0.984±0.197")
- Weight: 5.3 grams
- Battery Connector: JST SHR-02V-S-B
- Mating Connector: JST SM02B-SRSS-TB(LF)(SN)

Figure 38: Lithium ION Battery Specs.

6.4.4. MicroSD TinyShield

The TinyShield microSD Adapter enables the Arduino Watch to capture sensor data and save it onto a microSD card. Users can access this data by connecting the microSD card to the TinyDuino. The adapter is compatible with SD card support libraries available within the Arduino Software environment. The Arduino Watch will be using a 64GB microSD card.

This TinyShield includes level shifters and a local power supply, ensuring reliable and safe operation across the entire TinyDuino operating voltage range, up to 5V (‘TinyCircuits | MicroSD’, 2024).

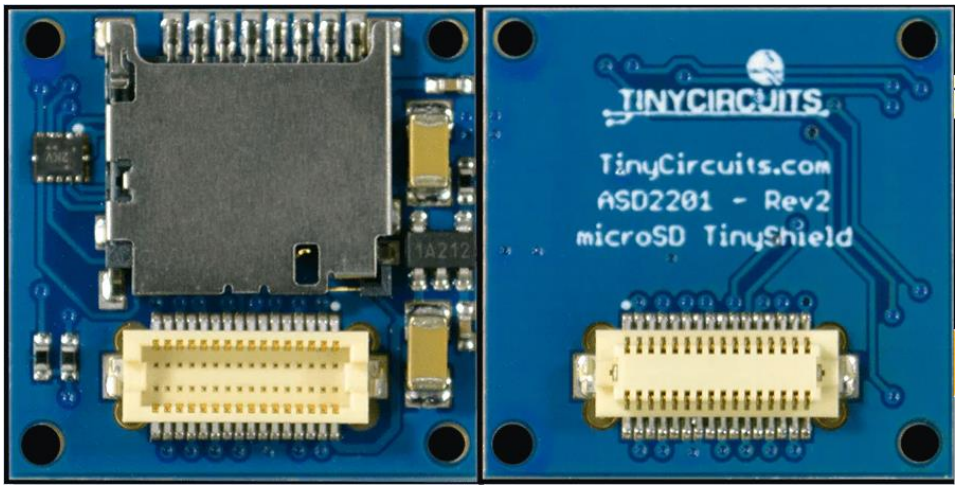


Figure 39: MicroSD TinyShield (ASD2201-R).

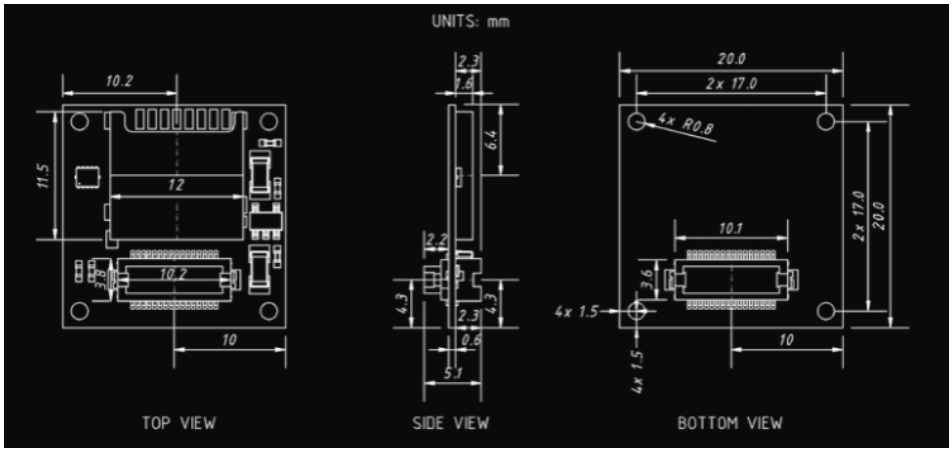


Figure 40: MicroSD TinyShield Dimensions.

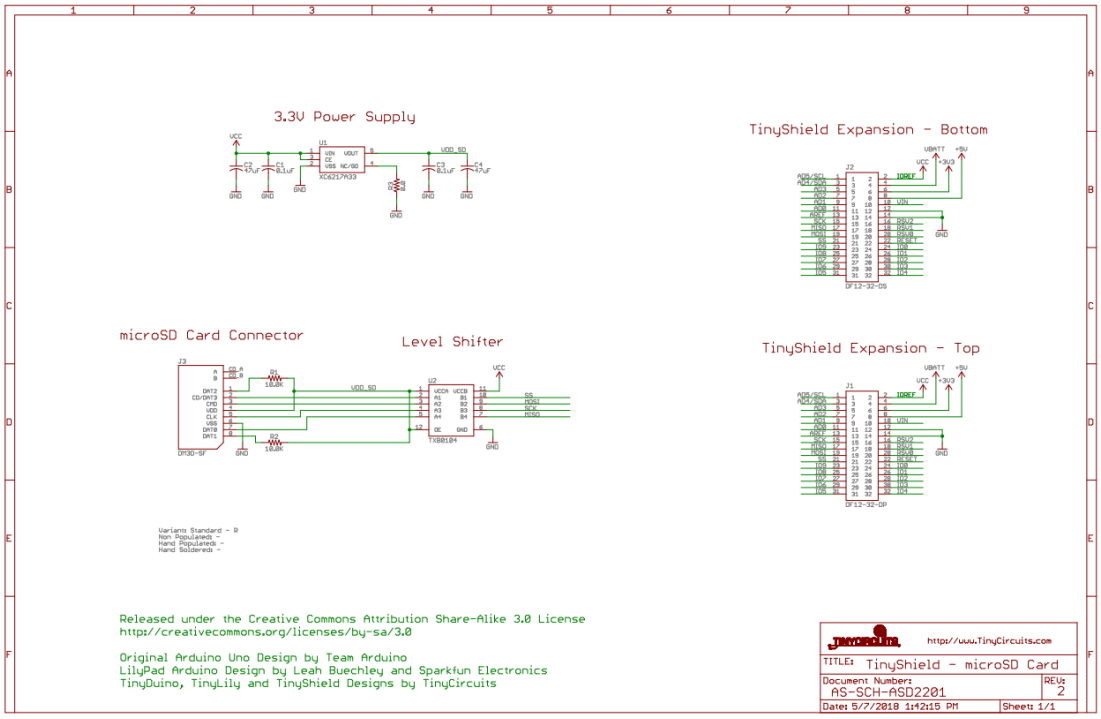


Figure 41: MicroSD TinyShield Schematic.

MicroSD Specs

- Uses standard Arduino SD Card Library
- Supports standard microSD cards and SDHC cards

TinyDuino Power Requirements

- Voltage: 3.0V - 5.5V
- Current: 100mA or more during SD card writes, depends on the microSD card being used. Because of this high current, the TinyDuino processor cannot be used with a coin cell.

Pins Used

SPI Interface used

- **10 - CS:** This signal SPI chip select for the microSD card
- **11 - SCLK:** This signal is the serial SPI clock out of the TinyDuino and into the microSD card.
- **12 - MISO:** This signal is the serial SPI data out of the microSD card and into the TinyDuino.
- **13 - MOSI:** This signal is the serial SPI data out of the TinyDuino and into the microSD card.

Dimensions

- 20mm x 20mm (.787 inches x .787 inches) Note: microSD car overhanges the edge by approx 3mm for easy removal
- Max Height (from lower bottom TinyShield Connector to upper top TinyShield Connector): 5.11mm (0.201 inches)
- Weight: 1.36 gram (.05 ounces)

Figure 42: microSD TinyShield Specs.

6.4.5. Wireling Adapter TinyShield

The Wireling Adapter TinyShield integrates a 4-channel multiplexer, along with onboard voltage regulation and level shifting capabilities, enabling connection of up to 4 Wirelings per Shield. The multiplexer feature helps to connect multiple I²C devices with the same address without encountering interference issues, which is essential for attaching wirelings such as the pulse oximeter and accelerometer to the TinyDuino (Arduino Watch).

Wireling Adapter TinyShields are designed to work seamlessly with TinyCircuits processors featuring a 32-pin connector (‘TinyCircuits | Wireling Adapter’, 2022).

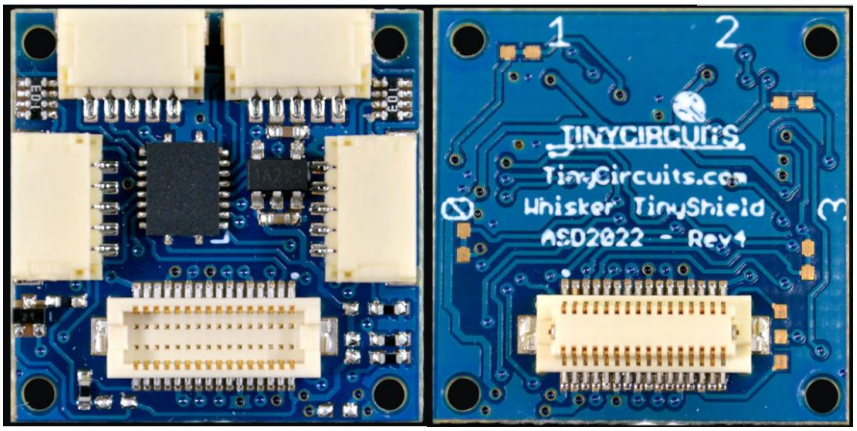


Figure 43: Wireling Adapter TinyShield (ASD2022).

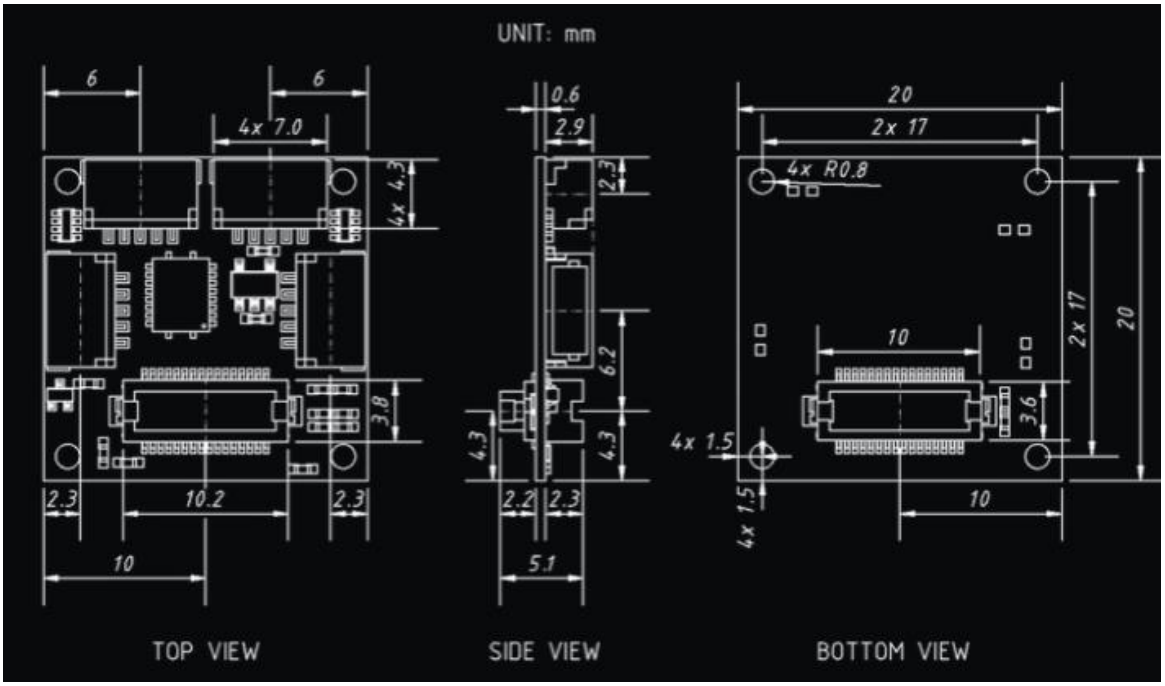


Figure 44: Wireling Adapter Dimensions.

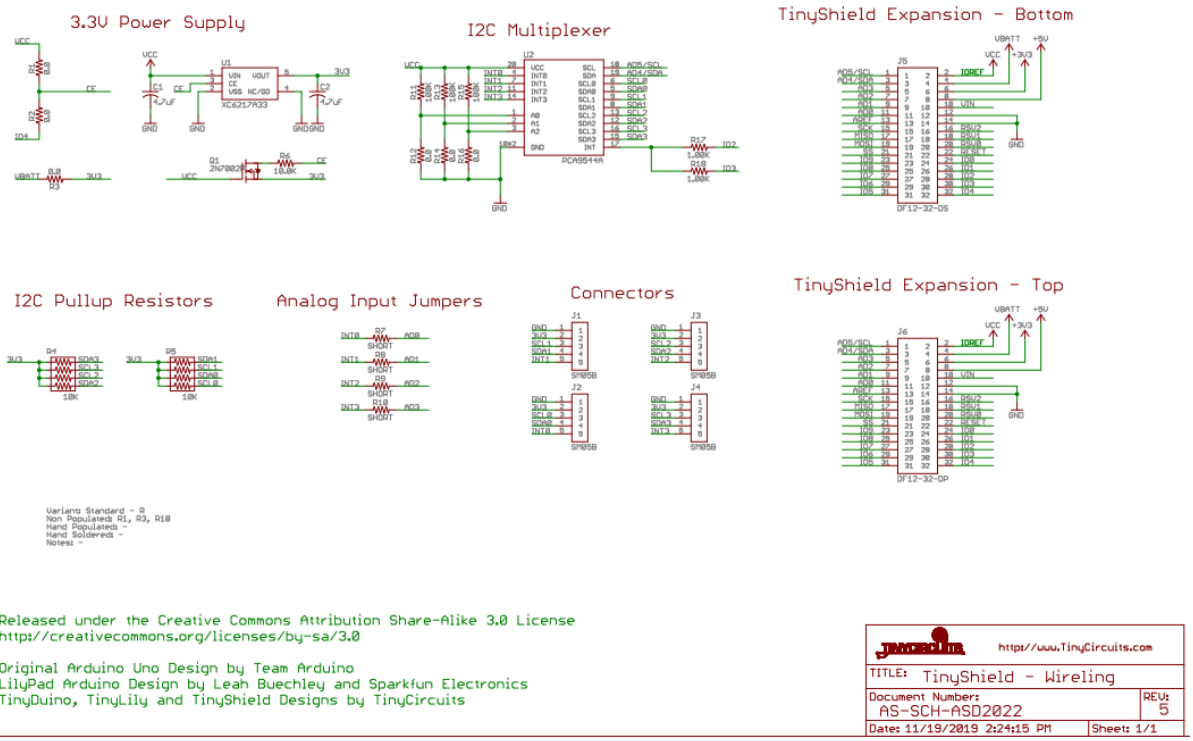


Figure 45: Wireling Adapter Schematic.

MULTIPLEXER Specs

- I²C Bus
- Four Active-Low Interrupt Inputs
- Active-Low Interrupt Output
- Three Address Pins

TinyDuino Power Requirements

- Voltage: 3.0V - 5.5V

Pins Used

- A5/SCL - I²C Serial Clock line
- A4/SDA - I²C Serial Data line
- INT - Optional IO2 or IO3 on TinyDuino
- IO4 - Power Control
- I²C Address used: 0x70

Dimensions

- 20mm x 20mm (.787 inches x .787 inches)
- Max Height (from the bottom of 32-pin expansion connector to top Wireling Connector [illegible])
- Weight: 1 gram (.04 ounces)

Figure 46: Wireling Adapter TinyShield Specs.

6.4.6. Accelerometer TinyShield and Sensor Wireling

This TinyShield contains the Bosch BMA250 3-axis accelerometer, known for its high performance and low power consumption. The BMA250 measures accelerations along three perpendicular axes, enabling the detection of tilt, motion, shock, and vibration, which is ideal for step counting in the Arduino Watch. This will be connected to the Arduino Watch via the Wireling Adapter TinyShield, which will be connected to the Accelerometer Wireling.

While the BMA250 is typically designed to operate at 1.8V, the Accelerometer TinyShield includes level shifters and a local power supply to guarantee reliable and safe operation across the entire TinyDuino operating voltage range, up to 5V (‘TinyCircuits | Accelerometer Shield’, 2024).

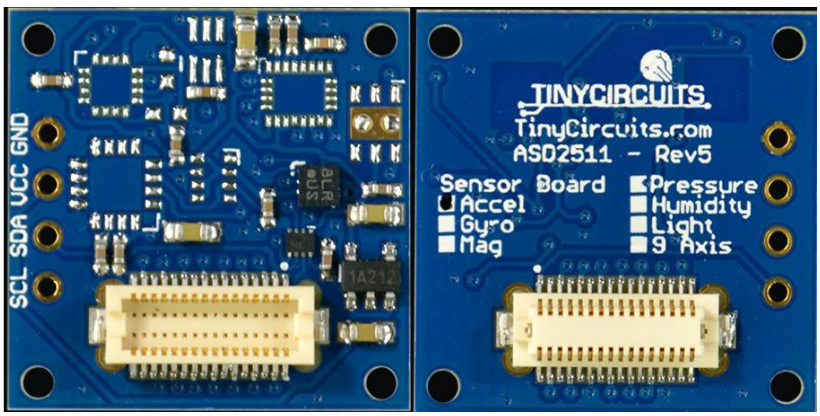


Figure 47: Accelerometer TinyShield (ASD2511-R-A).

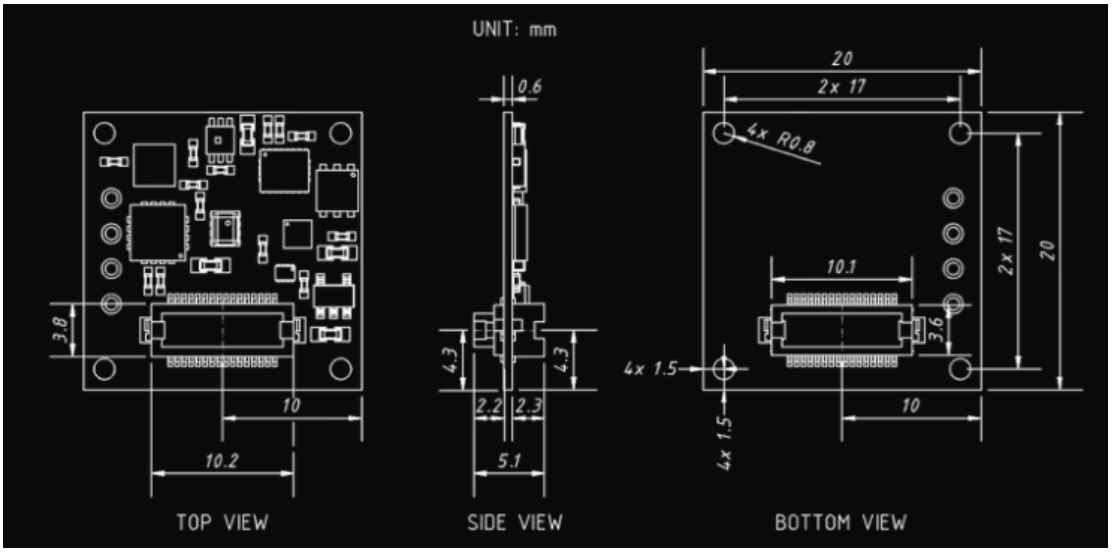


Figure 48: Accelerometer TinyShield Dimensions.

Figure 49: Accelerometer Wireling (AST1001).

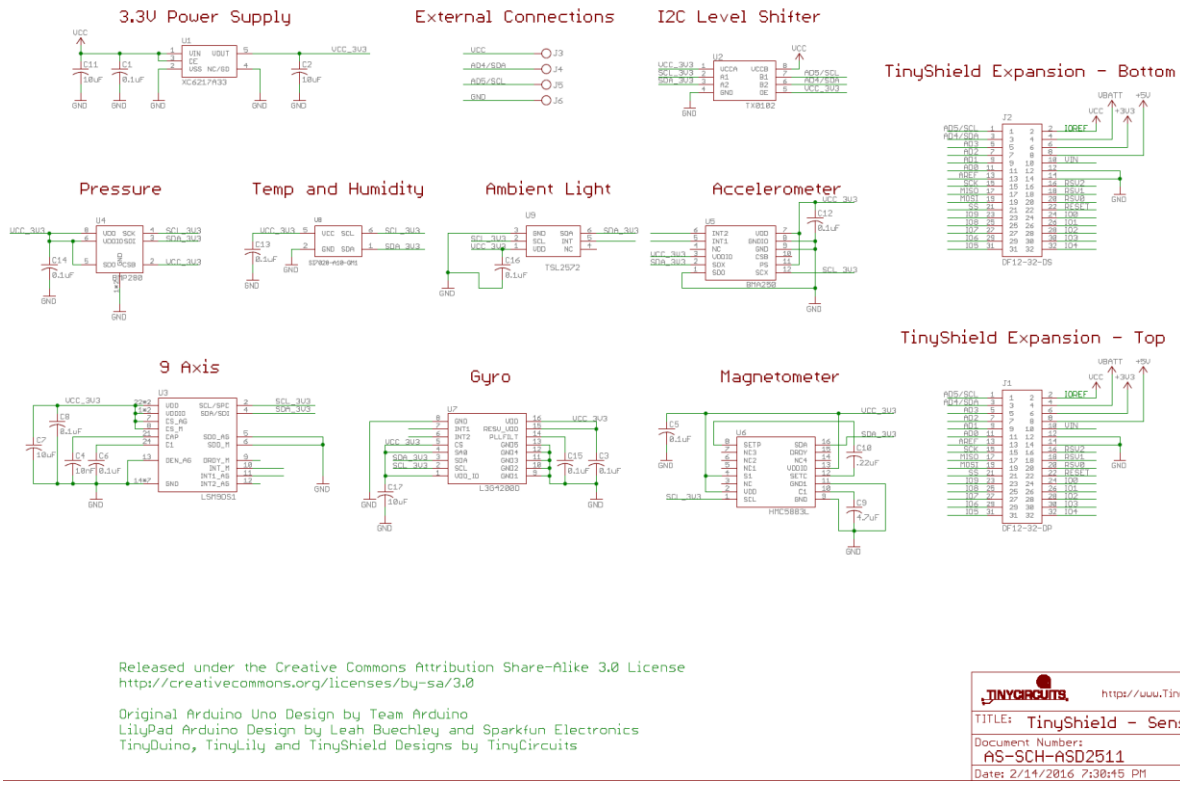


Figure 50: Accelerometer TinyShield Schematic.

Bosch BMA250 Accelerometer Specs

- 3-axis (X, Y & Z)
- Digital resolution: 10bit
- Resolution: 3.9mg
- Measurement ranges: +-2g, +-4g, +-8g, +-16g
- Sensitivity: 2g: 256LSB/g, 4g: 128LSB/g, 8g: 64LSB/g, 16g: 32LSB/g
- Zero-g offset (over lifetime): +-80mg
- Bandwidths: 1000Hz... 8Hz
- Low Power: 139uA @2kHz data rate

TinyDuino Power Requirements

- Voltage: 3.0V - 5.5V
- Current: 139uA (Normal Mode). Due to the low current, this board can be run using the TinyDuino coin cell option

Pins Used

- A5/SCL - I2C Serial Clock line
- A4/SDA - I2C Serial Data line

Dimensions

- 20mm x 20mm (.787 inches x .787 inches)
- Max Height (from lower bottom TinyShield Connector to upper top TinyShield Connector): 5.11mm (0.201 inches)
- Weight: 1 gram (.04 ounces)

Figure 51: Accelerometer TinyShield Specs.

6.4.7. Pulse Oximeter Sensor Wireling

This Wireling enables the measurement of heartbeat, blood oxygen saturation level, cardiogram data, and temperature using the MAX30101 pulse sensor. The sensor integrates internal LEDs, photodetectors, optical components, and low-noise electronics (‘TinyShield | Pulse Oximeter Sensor’, 2024). Making it suitable for the Arduino Watch to capture multiple types of physiological data with a single sensor.

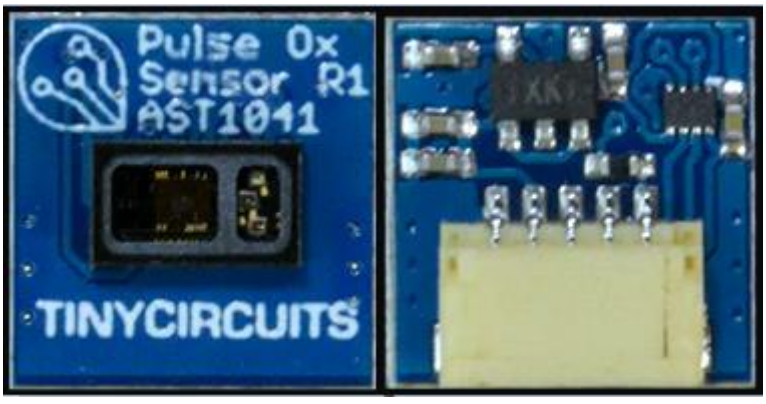


Figure 52: Pulse Oximeter Sensor Wireling (AST1041).

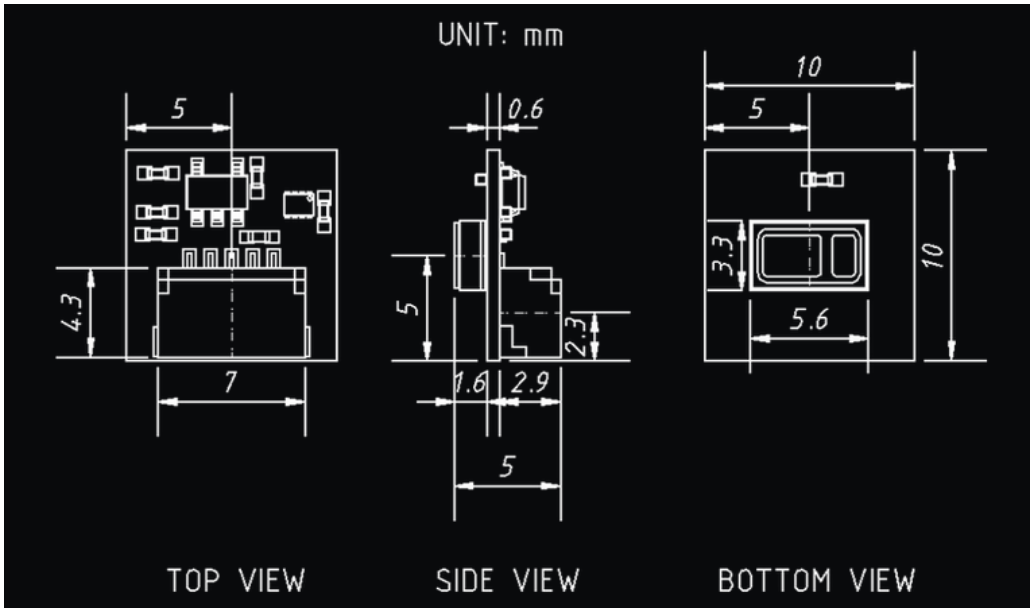


Figure 53: Pulse Oximeter Sensor Wireling Dimensions.

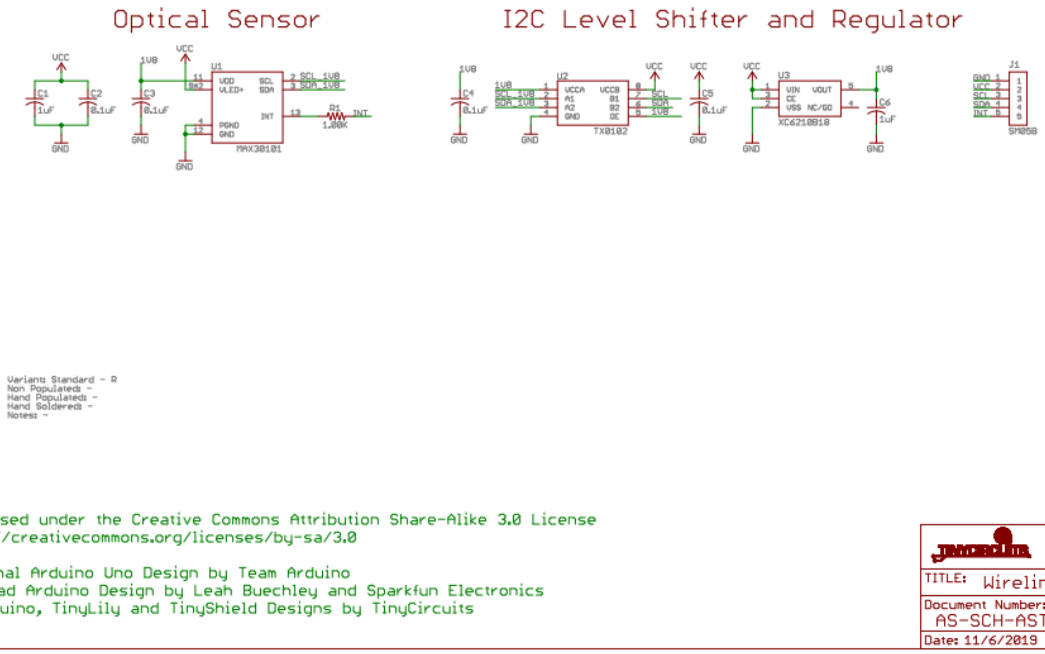


Figure 54: Pulse Oximeter Sensor Wireling Schematic.

MAX30101 Specs

- Heart-Rate Monitor and Pulse Oximeter Sensor in LED Reflective Solution
- Integrated cover glass for optimal, robust performance
- Programmable sample rate and LED Current for power savings
- Fast Data output capability: high sample rates
- Robust motion artifact resilience: high SNR

TinyDuino Power Requirements

- Voltage: 3.0V - 5.5V
- Power: < mW (Low-Power Mode)
- Current: 0.7uA, typ (Ultra-Low Shutdown Current)

Pins Used

- A5/SCL - I²C Serial Clock line
- A4/SDA - I²C Serial Data line

Dimensions

- 10mm x 10mm (.39 inches x .39 inches)
- Max Height (from the lower bottom of Wireling to upper top of pulse sensor) 5.11mm (0.20 inches)
- Weight: 1 gram (.04 ounces)

Figure 55: Pulse Oximeter Sensor Wireling Specs.

6.4.8. Bluetooth Low Energy TinyShield (ST)

The Bluetooth Low Energy (BLE) TinyShield combines master and slave modes into a single module. It enables the TinyDuino (Arduino Watch) system to connect with compatible iOS or Android devices using BLE, also known as Bluetooth 4.1. This technology is particularly advantageous for the watch, facilitating connectivity with low-power sensors and accessories to most modern smartphones.

The SPBTLE-RF0 module within BLE supports both master and slave modes, allowing the Arduino Watch to communicate with both BLE devices (i.e. smartphones) and other BLE modules or TinyShields operating in slave mode (‘TinyCircuits | Bluetooth Low Energy Shield (ST)’, 2024).

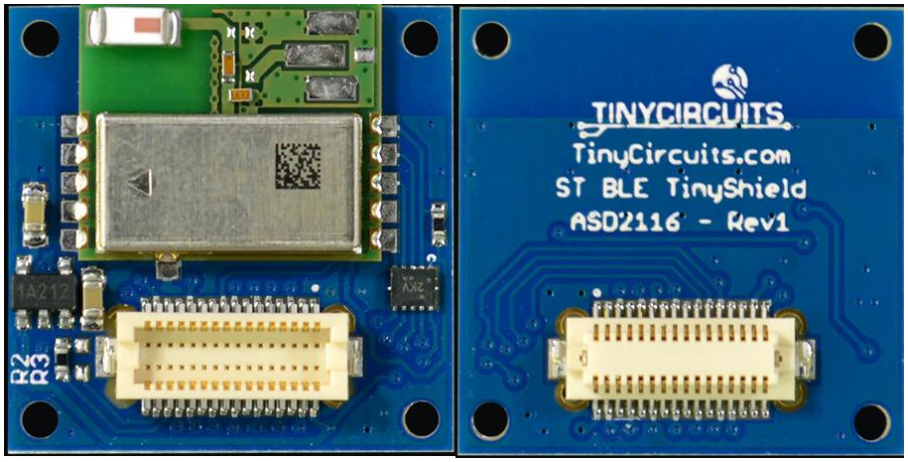


Figure 56: Bluetooth Low Energy TinyShield (ASD2116).

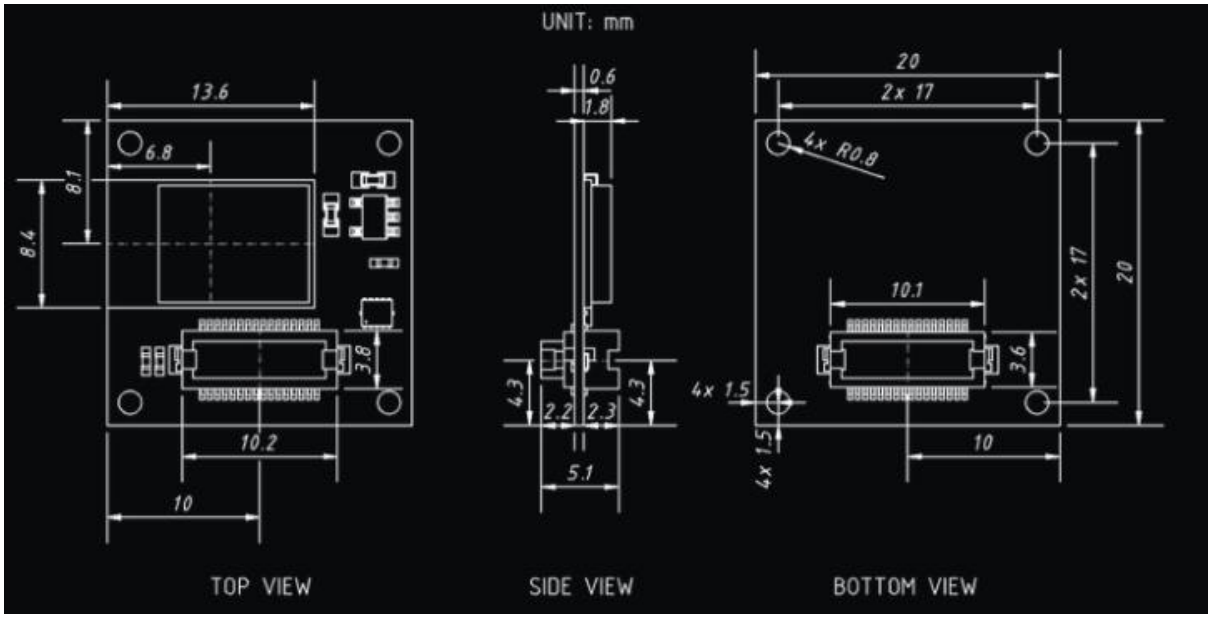


Figure 57: Bluetooth Low Energy TinyShield Dimensions.

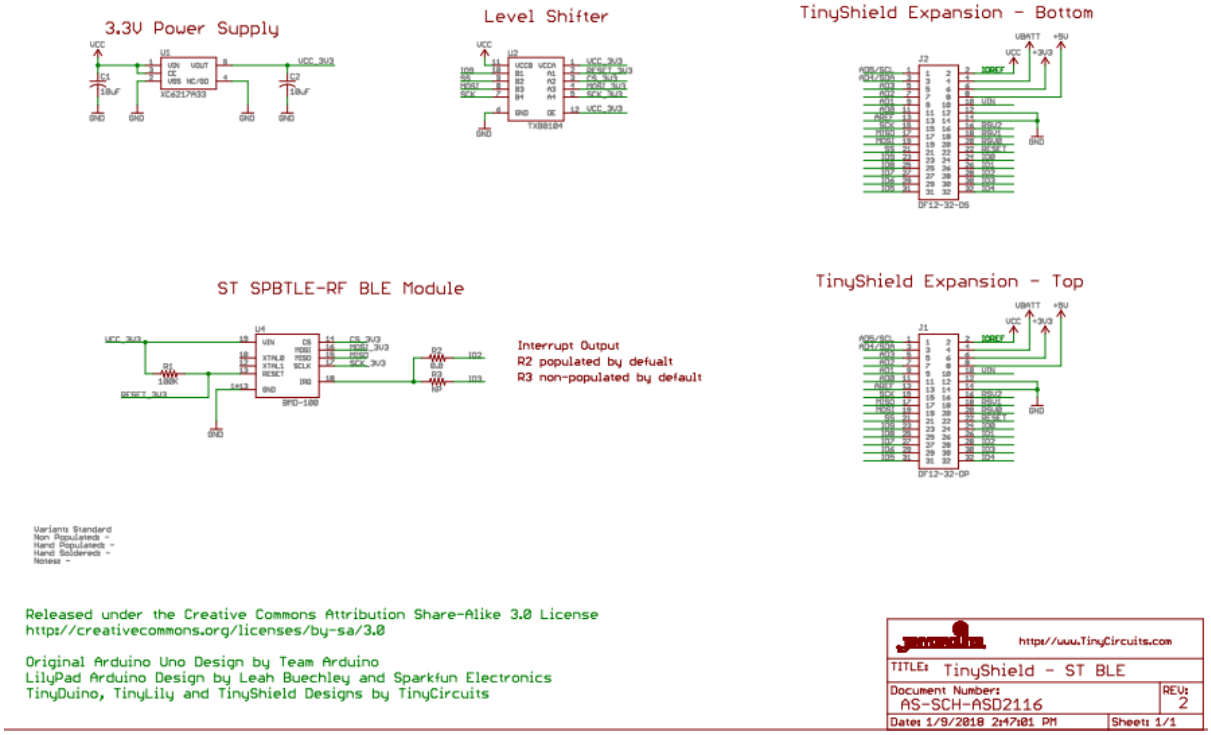


Figure 58: Bluetooth Low Energy TinyShield Schematic.

ST SPBTLE-RF0 Specs

- Fully Bluetooth v4.1 compliant
  - Supports master and slave modes
  - Multiple modes supported simultaneously
- Integrated Bluetooth Smart stack
  - GAP, GATT, SM, L2CAP, LL, RFPHY
  - Bluetooth Smart profiles
- Radio performance
  - Integrated chip antenna
  - TX power: +4 dBm
  - Receiver sensitivity: -88 dBm
  - Range: up to 20m
- Link Layer: AES-128 encryption and decryption

TinyDuino Power Requirements

- Voltage: 3.0V - 5.5V
- Current:
  - Transmit: 10.9mA (+4 dBm)
  - Receive: 7.3mA (Standard mode)
  - Down to 8.7µA average for 1s connection interval
  - Down to 14.5µA average for 500ms connection interval
  - Down to 26.1µA average for 250ms connection interval
  - Down to 119.9µA average for 50ms connection interval
  - Deep Sleep Mode: 1.7 µA
  - Due to the low current, this board can be run using the TinyDuino coin cell option

Pins Used

SPI Interface used

- **2 - SPI_IRQ:** This signal is the interrupt output from the SPBTLE-RF0 to the TinyDuino.
- **9 - BT_RESET:** This signal is the reset signal to the SPBTLE-RF0.
- **10 - SPI_CS:** This signal is the SPI chip select for the SPBTLE-RF0.
- **11 - MOSI:** This signal is the serial SPI data out of the TinyDuino and into the radio transceiver.
- **12 - MISO:** This signal is the serial SPI data out of the radio transceiver and into the TinyDuino.
- **13 - SPI_CLK:** This signal is the serial SPI clock out of the TinyDuino and into the radio transceiver.

Dimensions

- 20mm x 20mm (.787 inches x .787 inches)
- Max Height (from lower bottom TinyShield Connector to upper top TinyShield Connector): 5.11mm (0.201 inches)
- Weight: 1.49 grams (0.053 ounces)

Figure 59: Bluetooth Low Energy TinyShield Specs.

6.4.9. Arduino Watch Modular Component Stack

Figure 60: Arduino Watch Modular Component stack diagram.

In this chapter, we have explored the comprehensive design process of the AFNT project, spanning from database design to hardware integration for the Arduino Watch. Each phase plays a crucial role in enhancing the functionality and user experience of the AFNT system. By analyzing the design details such as database schema, website wireframes, application structure, and selection of modular components, this chapter establishes a solid foundation for the upcoming development and implementation stages discussed further in Chapter 7. The detailed documentation and design files provided for each phase offer valuable insights and guidance for the project's implementation.

7. Implementation

This chapter provides an overview of the implementation progress of the AFNT project across its four key phases. Each phase is discussed individually, detailing the technologies used and the extent of implementation achieved.

In terms of progress, Phase 1 has been predominantly implemented, with over 90% completion, focusing on database design and setup. Phase 3 and Phase 4 follow closely behind, with approximately 70% of their functionalities implemented, particularly in developing the AFNT application and integrating the Arduino Watch hardware. Phase 2, however, experienced minimal development due to time constraints and lower priority within the project's timeline. This chapter will delve into the specifics of each phase's implementation status and the technologies employed to complete AFNT project objectives and requirements.

7.1. Phase 1: DBMS Implementation

7.1.1. Central Database

The CDB serves as the repository for predefined workouts, meals, and user credentials, enabling secure login and access to AFNT user data. The CDB is implemented using MySQL Workbench due to its reliable performance, scalability, and robust data management capabilities, essential for handling critical user information securely. Its security features, including encryption options and user access controls, provide necessary protection for sensitive data stored in the database (‘SQL Server | Microsoft’, 2022).

Currently, only the `users` table is actively used in the CDB, with Phase 2 development remaining largely incomplete due to time constraints. Below are the examples of how the CDB tables were implemented.

```
CREATE TABLE `centraldb`.`exercises` (
  `exercise_id` INT NOT NULL AUTO_INCREMENT,
  `exercise_name` VARCHAR(150) NOT NULL,
  `description` VARCHAR(300) NULL,
  `type` ENUM('Strength', 'Cardio', 'Olympic Weightlifting', 'Plyometrics', 'Powerlifting
  `body_part` ENUM('Abdominals', 'Adductors', 'Chest', 'Hamstrings', 'Quadriceps', 'L
  `equiptment` ENUM('Body Only', 'Dumbbell', 'Exercise Ball', 'Kettlebells', 'Medicine
  `level` ENUM('Beginner', 'Intermediate', 'Advanced') NOT NULL,
  `rating` DOUBLE NULL,
  `rating_description` VARCHAR(45) NULL,
  `is_active` TINYINT NOT NULL,
  PRIMARY KEY (`exercise_id`),
  UNIQUE INDEX `exercise_id_UNIQUE` (`exercise_id` ASC) VISIBLE);
```

Figure 119: Create exercises table (CDB).

```sql
CREATE TABLE `centraldb`.`food_items` (
  `food_item_id` INT UNIQUE PRIMARY KEY,
  `food_item_name` VARCHAR(255) NOT NULL,
  `water_g` DECIMAL(5, 2),
  `energy_kcal` INT NOT NULL,
  `protein_g` DECIMAL(5, 2),
  `lipid_g` DECIMAL(5, 2),
  `carbs_g` DECIMAL(5, 2),
  `fiber_td_g` DECIMAL(5, 2),
  `sugar_g` DECIMAL(5, 2),
  `calcium_mg` DECIMAL(5, 2),
  `iron_mg` DECIMAL(5, 2),
  `cholestrl_mg` DECIMAL(5, 2),
  `gmwt_1` DECIMAL(5, 2),
  `gmwt_desc1` VARCHAR(255),
  `gmwt_2` DECIMAL(5, 2),
  `gmwt_desc2` VARCHAR(255),
  `is_active` BOOLEAN NOT NULL
);
```

Figure 123: Create food items table (CDB).

To view how all CDB tables were created, navigate to Appendix C: Creating Central Database.

For data, the food item and exercise datasets sourced from Kaggle serve as test samples. The exercises dataset contains nearly 3000 records (Pandit, N. 2022), while the food items dataset comprises approximately 8800 records (Vinit S., 2022). This data can populate the meals table and support the creation of both custom and preset workouts, allowing users to personalize their workout and meal data alongside their user profile information.

7.1.2. Local Database

SQLite was employed to implement the LDB in Phase 1.5 of the project. This choice was made to ensure data privacy, as SQLite stores data locally on the device rather than on a server, enhancing security.

The localDB schema includes tables for exercises, workouts, food items, meals, and body statistics, enabling comprehensive data management for user-specific information. Despite its advantages, SQLite has limitations, particularly in its lack of support for certain data types compared to more robust database systems. However, for the scope of this project, SQLite serves as an effective and efficient solution for managing local data storage and retrieval. The creation of specific tables within the localDB demonstrates the structured approach to organizing fitness and nutrition-related data for the AFNT application.

```python
"""
LocalDB Class that stores all User Workout, Meal and Body Data in a SQLite Database.
This class is responsible for creating localDB tables and intialising it and import data from the CSV file.
"""
class LocalDB():

    # Initializing LocalDB connection
    def __init__(self, local_db):
        self.local_db = local_db
        self.connection = sqlite3.connect(self.local_db)
        self.cursor = self.connection.cursor()
        print("\nLocalDB Connection Established!\n")
```

Figure 61: Initialization and connection to the LDB.

```
def create_local_db_tables(self):
    create_exercises = """
        CREATE TABLE IF NOT EXISTS exercises (
        exercise_id TEXT PRIMARY KEY NOT NULL,
        exercise_name TEXT NOT NULL,
        description TEXT,
        type TEXT,
        body_part TEXT NOT NULL,
        equipment TEXT,
        level TEXT,
        rating REAL,
        rating_description TEXT,
        is_active BOOLEAN NOT NULL
        )"""

    create_exercise_logs = """
        CREATE TABLE IF NOT EXISTS exercise_logs (
        exercise_log_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
        exercise_id TEXT NOT NULL,
        workout_log_id INTEGER NOT NULL,
        sets INTEGER,
        reps INTEGER,
        weight_kg REAL,
        rest_per_set_s INTEGER,
        duration TEXT,
        distance_m INTEGER,
        rpe REAL,
        is_complete BOOLEAN NOT NULL,
        date_complete DATE,
        time_complete TIME,
        details TEXT,
        is_active BOOLEAN NOT NULL,
        FOREIGN KEY (exercise_id) REFERENCES exercises(exercise_id),
        FOREIGN KEY (workout_log_id) REFERENCES workout_logs(workout_log_id)
        )"""
```

Figure 62: Create exercise tables (LDB).

```
create_workouts = """
    CREATE TABLE IF NOT EXISTS workouts (
    workout_id TEXT PRIMARY KEY NOT NULL,
    workout_name TEXT NOT NULL,
    description TEXT,
    type TEXT NOT NULL,
    date_created DATE,
    level TEXT,
    rating REAL,
    rating_description TEXT,
    time_created TIME,
    is_active BOOLEAN NOT NULL
    )"""

create_workout_logs = """
    CREATE TABLE IF NOT EXISTS workout_logs (
    workout_log_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    workout_id TEXT NOT NULL,
    date_assigned DATE NOT NULL,
    time_assigned TIME NOT NULL,
    is_complete BOOLEAN NOT NULL,
    date_completed DATE,
    time_completed TIME,
    is_active BOOLEAN NOT NULL,
    FOREIGN KEY (workout_id) REFERENCES workouts(id)
    )"""
```

Figure 63: Create workout tables (LDB).

```
create_food_items = """
    CREATE TABLE IF NOT EXISTS food_items (
    food_item_id TEXT PRIMARY KEY NOT NULL,
    food_item_name TEXT NOT NULL,
    water_g INTEGER,
    energy_kcal INTEGER,
    protein_g REAL,
    lipid_g REAL,
    carbs_g REAL,
    fiber_td_g REAL,
    sugar_g REAL,
    calcium_mg REAL,
    iron_mg REAL,
    cholestrl_mg REAL,
    gmwt_1 TEXT,
    gmwt_desc1 TEXT,
    gmwt_2 TEXT,
    gmwt_desc2 TEXT,
    is_active BOOLEAN NOT NULL
    )"""

create_food_item_logs = """
    CREATE TABLE IF NOT EXISTS food_item_logs (
    food_item_log_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    meal_log_id INTEGER NOT NULL,
    food_item_id TEXT NOT NULL,
    serving REAL,
    weight_g REAL,
    ate BOOLEAN NOT NULL,
    date_ate DATE,
    time_ate TIME,
    description TEXT,
    is_active BOOLEAN NOT NULL,
    FOREIGN KEY (food_item_id) REFERENCES food_items(food_item_id),
    FOREIGN KEY (meal_log_id) REFERENCES meal_logs(meal_log_id)
    )"""
```

Figure 64: Create food item tables (LDB).

```
create_meals = """
    CREATE TABLE IF NOT EXISTS meals (
    meal_id TEXT PRIMARY KEY NOT NULL,
    type TEXT NOT NULL,
    description TEXT,
    water_tot_g INTEGER,
    energy_tot_kcal INTEGER,
    protein_tot_g REAL,
    lipid_tot_g REAL,
    carbs_tot_g REAL,
    fiber_tot_g REAL,
    sugar_tot_g REAL,
    calcium_tot_mg REAL,
    iron_tot_mg REAL,
    cholestrl_tot_mg REAL,
    serving REAL NOT NULL,
    date_created DATE,
    time_created TIME,
    is_active BOOLEAN NOT NULL
    )"""

create_meal_logs = """
    CREATE TABLE IF NOT EXISTS meal_logs (
    meal_log_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    meal_id TEXT NOT NULL,
    ate BOOLEAN NOT NULL,
    date_ate DATE,
    time_ate TIME,
    is_active BOOLEAN NOT NULL,
    FOREIGN KEY (meal_id) REFERENCES meals(meal_id)
    )"""
```

Figure 65: Create meal tables (LDB).

```
create_bmi = """
    CREATE TABLE IF NOT EXISTS bmi (
    bmi_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    bmi REAL NOT NULL,
    date_recorded DATE NOT NULL,
    time_recorded TIME NOT NULL
    )"""

create_body_fat = """
    CREATE TABLE IF NOT EXISTS body_fat (
    body_fat_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    body_fat_kg REAL NOT NULL,
    date_recorded DATE NOT NULL,
    time_recorded TIME NOT NULL,
    body_fat_percent REAL NOT NULL
    )"""

create_skeletal_muscle = """
    CREATE TABLE IF NOT EXISTS skeletal_muscle (
    skeletal_muscle_id INTEGER PRIMARY KEY AUTOINCREMENT,
    skeletal_muscle_kg REAL NOT NULL,
    date_recorded DATE NOT NULL,
    time_recorded TIME NOT NULL
    )"""

create_height = """
    CREATE TABLE IF NOT EXISTS height (
    height_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    height_m REAL NOT NULL,
    date_recorded DATE NOT NULL,
    time_recorded TIME NOT NULL
    )"""

create_weight = """
    CREATE TABLE IF NOT EXISTS weight (
    weight_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    weight_kg REAL NOT NULL,
    date_recorded DATE NOT NULL,
    time_recorded TIME NOT NULL
    )"""

create_water_intake = """
    CREATE TABLE IF NOT EXISTS water_intake (
    water_intake_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    water_intake_ml REAL NOT NULL,
    date_recorded DATE NOT NULL,
    time_recorded TIME NOT NULL
    )"""

create_step_count = """
    CREATE TABLE IF NOT EXISTS step_count (
    step_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    step INTEGER NOT NULL,
    date_recorded DATE NOT NULL,
    time_recorded TIME NOT NULL
    )"""
```

Figure 66: Create body statistic tables (LDB).

The LDB similarly utilizes the exercises and food items datasets sourced from Kaggle, enabling administrators and users to generate and adjust preset workouts and meals as needed.

7.2. Phase 2: Admin Management Website Implementation

Phase 2 entails building the AMWebsite, a crucial component for database administration in the AFNT project. This phase centres on creating a secure and user-friendly interface to manage user credentials, workouts, meals, and other data stored in the CDB server.

Due to time constraints and delays associated with other university projects, along with its lower priority, phase 2 of the project was postponed to prioritize critical aspects. As of now, only the website login screen and authentication via a login server have been completed. Phase 2 employs Python Flask for both backend and frontend development, chosen for its simplicity and widespread adoption in the industry (‘Flask Documentation’, 2024). Abundant online resources are available to facilitate its effective implementation.

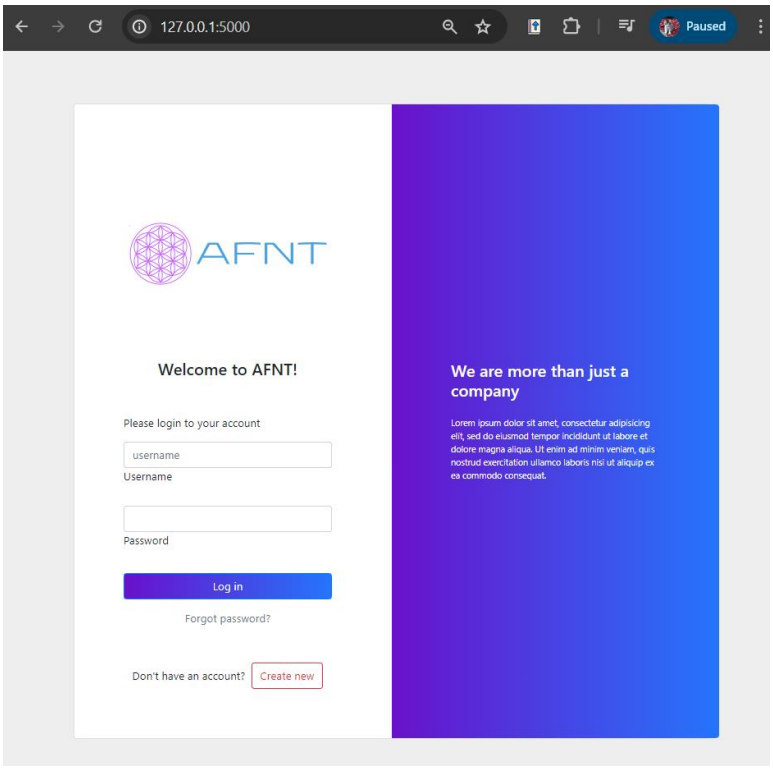


Figure 67: AM Website Login Screen.

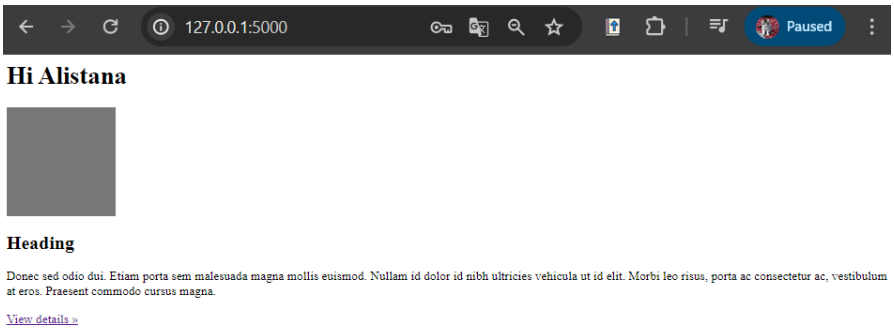


Figure 68: AM Website Login Success Screen (Incomplete Dashboard).

7.3. Phase 3: AFNT Application Implementation

In developing the AFNT application and Graphical User Interface (GUI), Python programming language and the Kivy GUI Framework were used as it offers flexibility and cross-platform support, making it an ideal choice. It's open-source and user-friendly (‘Kivy: Cross-platform Python Framework’, 2024). Please note that the AFNT App was implemented for the desktop version. Due to time constraints and the need to prioritize critical functionalities and requirements for the AFNT project, implementation and testing on smartphones were not conducted.

7.3.1. Kvlang Language

Kvlang is a declarative language used in Kivy for designing user interfaces (UIs) with a clear separation of logic and presentation. It allows developers to define the structure, appearance, and behaviour of UI components in a concise and readable format (Douglas, 2023). Kvlang simplifies the process of UI development by enabling rapid prototyping and easy customization of UI elements. By defining UI layouts and properties directly in Kvlang files, developers can create dynamic and interactive applications efficiently. This declarative approach enhances code readability and maintainability, making Kvlang a powerful tool for building cross-platform applications with Kivy.

```
<LoginScreen>:
    name: 'login_screen'
    FloatLayout:
        AsyncImage:
            source: 'img/afnt_login_background.jpg'
            allow_stretch: True
            keep_ratio: False

        MDCard:
            orientation: 'vertical'
            allow_stretch: False
            size_hint: None, None
            size: dp(300), dp(400)
            pos_hint: {"center_x": 0.5, "center_y": 0.5}

            BoxLayout:
                orientation: 'vertical'
                size_hint_x: None
                width: dp(250)
                spacing: dp(25)
                pos_hint: {'center_x': 0.5, 'center_y': 0.5}

                MDLabel:
                    id: welcome_label
                    text: 'Welcome Back!'
                    font_size: 40
                    bold: True
                    halign: 'center'
                    size_hint_y: None
                    height: self.texture_size[1]
                    padding_y: 15

                MDTextField:
                    mode: "round"
                    id: input_username
                    hint_text: "username"
                    helper_text_mode: "on_error"
                    helper_text: "enter valid username"
                    max_text_length: 10
                    icon_right: "account"
                    pos_hint: {'center_x': 0.5, 'center_y': 0.5}
                    size_hint_x: None
                    width: dp(250)
                    on_text: app.handle_username_input(self.text)
```

Figure 69: Kvlang example.

7.3.2. Kivy Material Design

In addition to leveraging Python and the Kivy GUI Framework, the project also incorporates KivyMD for its Material Design User Interface (MUI) components. KivyMD extends the capabilities of Kivy by providing a range of ready-to-use UI elements that align with Material Design guidelines, enhancing the visual appeal and usability of the AFNT application (‘KivyMD’, 2022). The utilization of MUI components not only streamlines the development process but also ensures consistency and modern aesthetics across different platforms. Moreover, the cross-platform nature of the AFNT app, facilitated by Kivy and KivyMD, enables seamless usage on various devices, including mobile platforms, thereby enhancing accessibility and user experience. This approach underscores the importance of leveraging frameworks that prioritize user interface design and cross-device compatibility in modern software development.

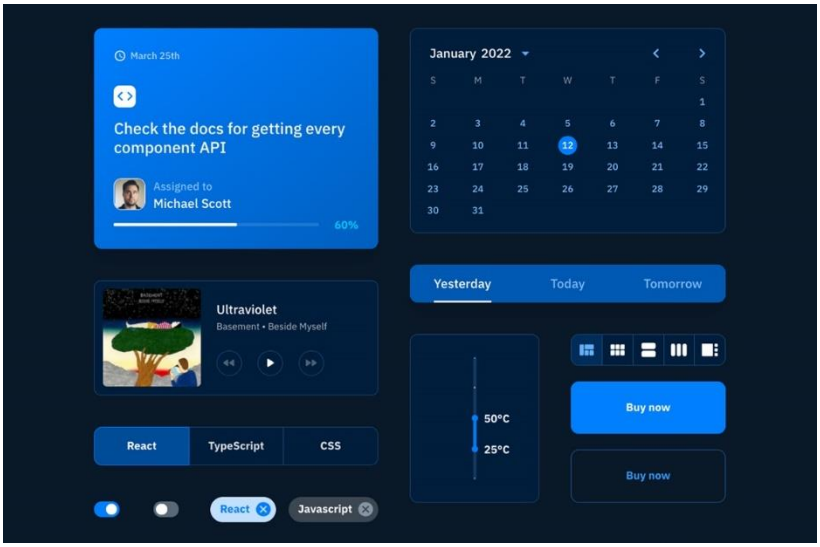


Figure 70: Example of MUI components.

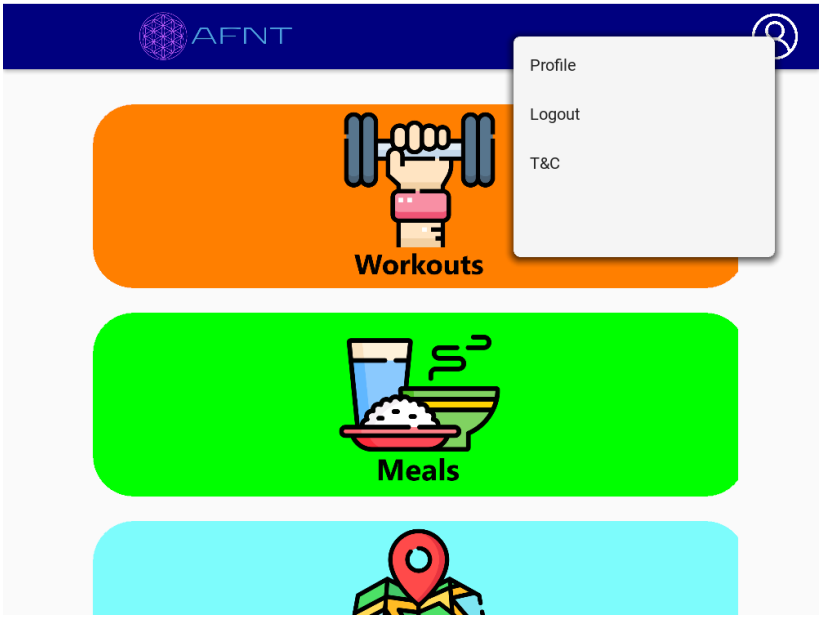


Figure 71: Utilizing MUI components in AFNT.

7.3.3. Kiny Screen Manager

In Kivy, the ScreenManager is essential for managing multiple screens within applications, allowing seamless transitions between different user interfaces. Each defined screen corresponds to specific functionality like login, registration, dashboard, workout history, and more. Developers utilize ScreenManager to dynamically switch between these screens based on user interactions or application logic, ensuring an intuitive user experience. By creating separate screen classes inherited from Kivy's Screen class, developers encapsulate the layout and behaviour for each screen and manage transitions effectively. ScreenManager also supports various transition effects, enhancing the visual appeal and usability of multi-screen applications.

```
WindowManager:
    LoginScreen:
    RegistrationScreen:
    DashboardScreen:
    BodyStatsScreen:
    BodyStatsPlotsScreen:
    WaterIntakeScreen:
    WaterIntakePlotsScreen:
    WorkoutHistoryScreen:
    WorkoutAllocateScreen:
    WorkoutCreateScreen:
    ExerciseCreateScreen:
    ExerciseLogScreen:
    ExerciseLogAllocateScreen:
    ExerciseAllocateScreen:
    GymFinderScreen:
    MealHistoryScreen:
    ArduinoWatchScreen:
    ArduinoWatchPlotsScreen:
    MealAllocateSceen:
    MealCreateScreen:
    FoodItemLogScreen:
    FoodItemLogAllocateScreen:
    FoodItemAllocateScreen:
```

Figure 72: AFNT Screen classes.

```
class DashboardScreen(Screen):
    def logout(self):
        self.manager.transition.direction = 'down'
        self.manager.current = 'login_screen'

    def switch_to_workout_history(self):
        self.manager.transition.direction = 'left'
        self.manager.current = 'workout_history_screen'
        plots_screen = self.manager.get_screen('workout_history_screen')
        plots_screen.clear_workout_log_datatable_box()
        plots_screen.create_workout_log_datatable()

    def switch_to_meal_history(self):
        self.manager.transition.direction = 'left'
        self.manager.current = 'meal_history_screen'
        plots_screen = self.manager.get_screen('meal_history_screen')
        plots_screen.clear_meal_log_datatable_box()
        plots_screen.create_meal_log_datatable()

    def switch_to_gym_finder(self):
        self.manager.transition.direction = 'left'
        self.manager.current = 'gym_finder_screen'

        plots_screen = self.manager.get_screen('gym_finder_screen')
        plots_screen.create_map_view()

    def switch_to_water_intake(self):
        self.manager.current = 'water_intake_screen'

    def switch_to_arduino_watch(self):
        self.manager.current = 'arduino_watch_screen'

    def switch_to_body_stats(self):
        self.manager.current = 'body_stats_screen'
```

Figure 73: Dashboard Screen class.

```
class SuccessPopup(Popup):
    def show_popup(self, title, para):
        content = Label(text=para)
        popup = Popup(title=title, content=content, size_hint=(None, None), size=(400, 200))
        popup.bind(on_dismiss=self.on_popup_dismiss)
        popup.open()

    def on_popup_dismiss(self, instance):
        pass
```

Figure 74: Basic Popup class example.

7.3.4. Major Components

To make it easier to cover each objective and requirement for Phase 3. This subchapter will categorize every requirement into 7 components (screens) as shown below:

Table 34: Phase 3 components.

Component	Func. Requirements	Non-Func. Requirements	Status
Login and Registration Screens	FA41 (Login) FA42 (Reg)	NFES2, NFR2, NFU2 NFU5-7	Complete
Profile Settings Screen	FA43-45		Not Complete
Workout Screens	FA1-14	NFA1-6, NFP13, NFES2, NFR2, NFU2 NFU5-7	Complete
Meal and Water Intake Screens	FA15-26	NFA8-14, NFES2, NFR2, NFU2 NFU5-7	Partially Complete
Gym Finder Screen	FA38	NFTPS1, NFES2, NFR2, NFU2 NFU5-7	Partially Complete
Body Statistics Screens	FA27-FA35	NFES2, NFR2, NFU2 NFU5-7	Complete
Arduino Watch Screens	FA36-37	NFES2, NFR2, NFU2 NFU5-7	Partially Complete

7.3.5. Login and Registration Screens

The login and registration screens utilize CDB for accessing and storing user login credentials.

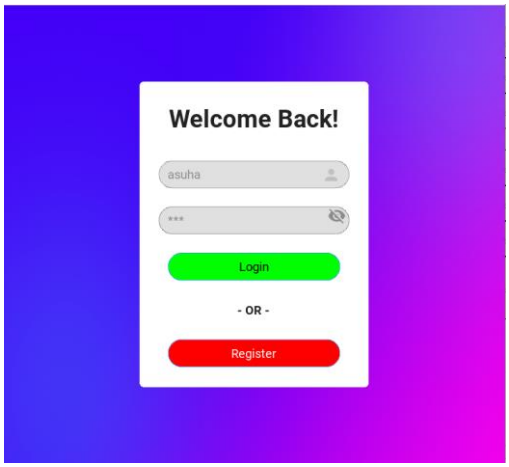


Figure 75: AFNT Login Screen.

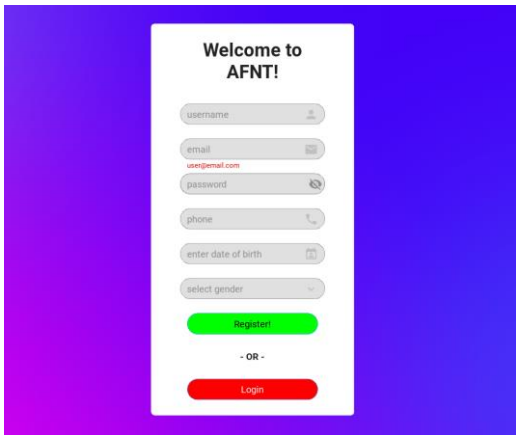


Figure 76: AFNT Registration Screen.

7.3.6. Workout Screens

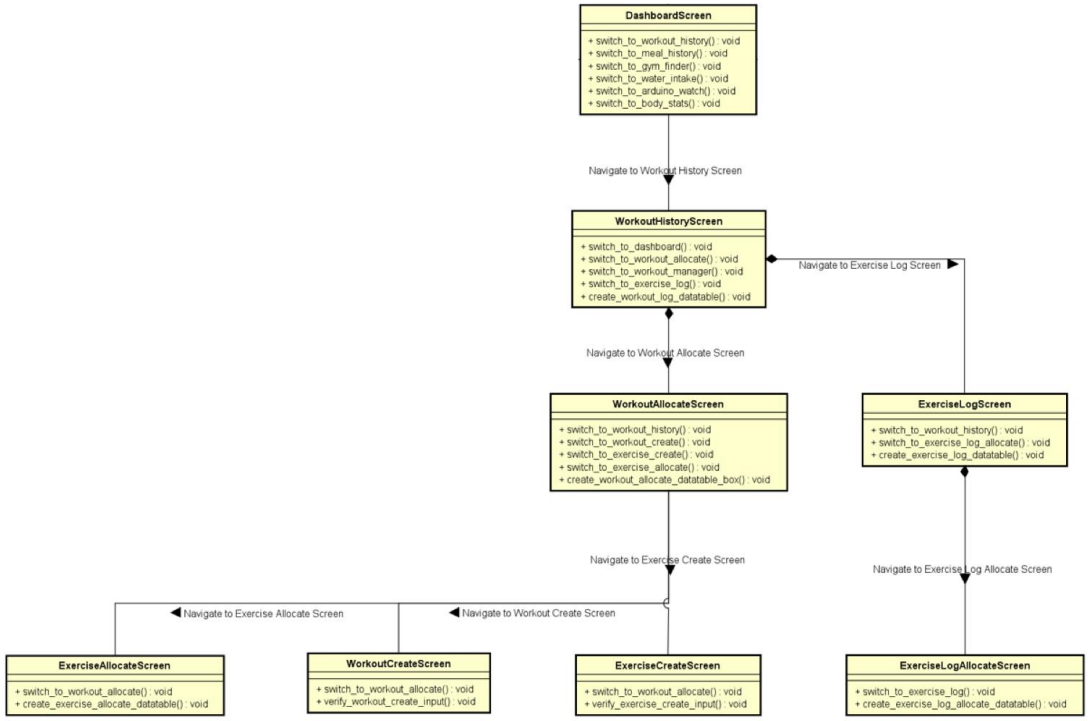


Figure 77: Workout Screens class diagram.

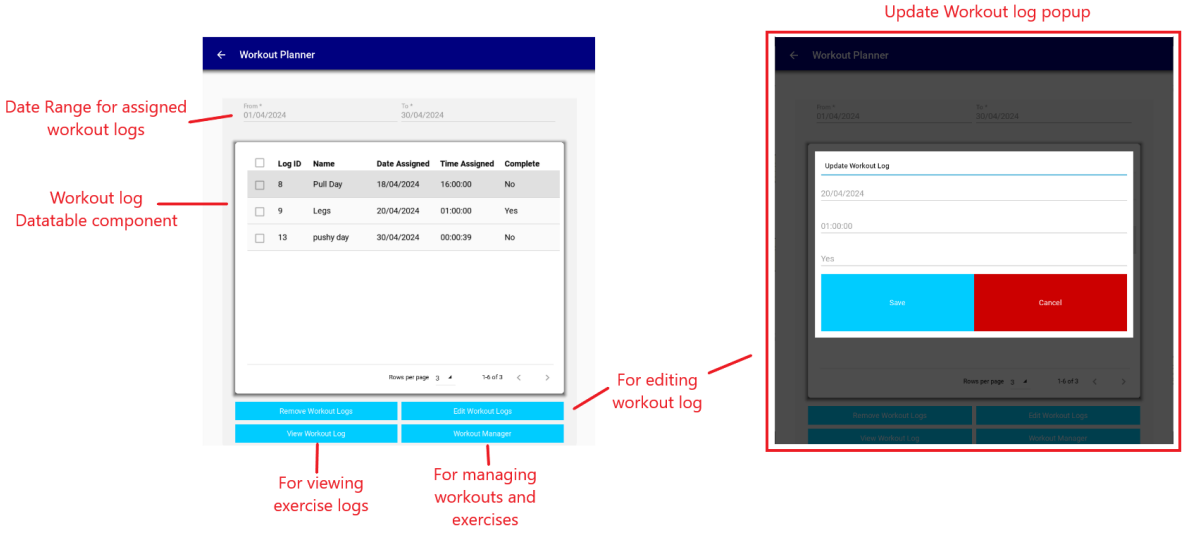


Figure 78: Workout History Screen.

Each of these screens covers specific requirements for workouts and exercises. For instance, the workout history screen class is responsible for managing the WorkoutLog class.

```python
def create_workout_log_datatable(self):
    screen = self.manager.get_screen('workout_history_screen')
    screen.ids.workout_log_date_from.text = self.from_selected_date
    screen.ids.workout_log_date_to.text = self.to_selected_date
    workout_log_datatable_box = self.ids.workout_log_datatable_box
    workout_log_data = self.workout_logs.get_date_selected_workout_logs(self.from_selected_date, self.to_selected_date)

    # Worrkout lod datatable displays data like the ID, workout name, date and time assigned, as well as the completion
    if workout_log_data:
        self.workout_log_datatable = MDDataTable(
            pos_hint={'center_x': 0.5, 'center_y': 0.5},
            size_hint=(0.95, 0.3),
            check=True,
            use_pagination=True,
            rows_num=6,
            pagination_menu_height='240dp',
            pagination_menu_pos="auto",
            background_color=[1, 0, 0, .5],
            column_data=[
                ["Log ID", dp(25)],
                ["Name", dp(25)],
                ["Date Assigned", dp(25)],
                ["Time Assigned", dp(25)],
                ["Complete", dp(25)]
            ],
            row_data=workout_log_data
        )
        self.workout_log_datatable.bind(on_check_press=self.rows_selected)
    else:
        self.workout_log_datatable = Label(text='No Workouts Recorded', color = 'red', font_size = "20sp", bold = True)

    workout_log_datatable_box.add_widget(self.workout_log_datatable)
```

Figure 79: Creating a Datatable using KivyMD.

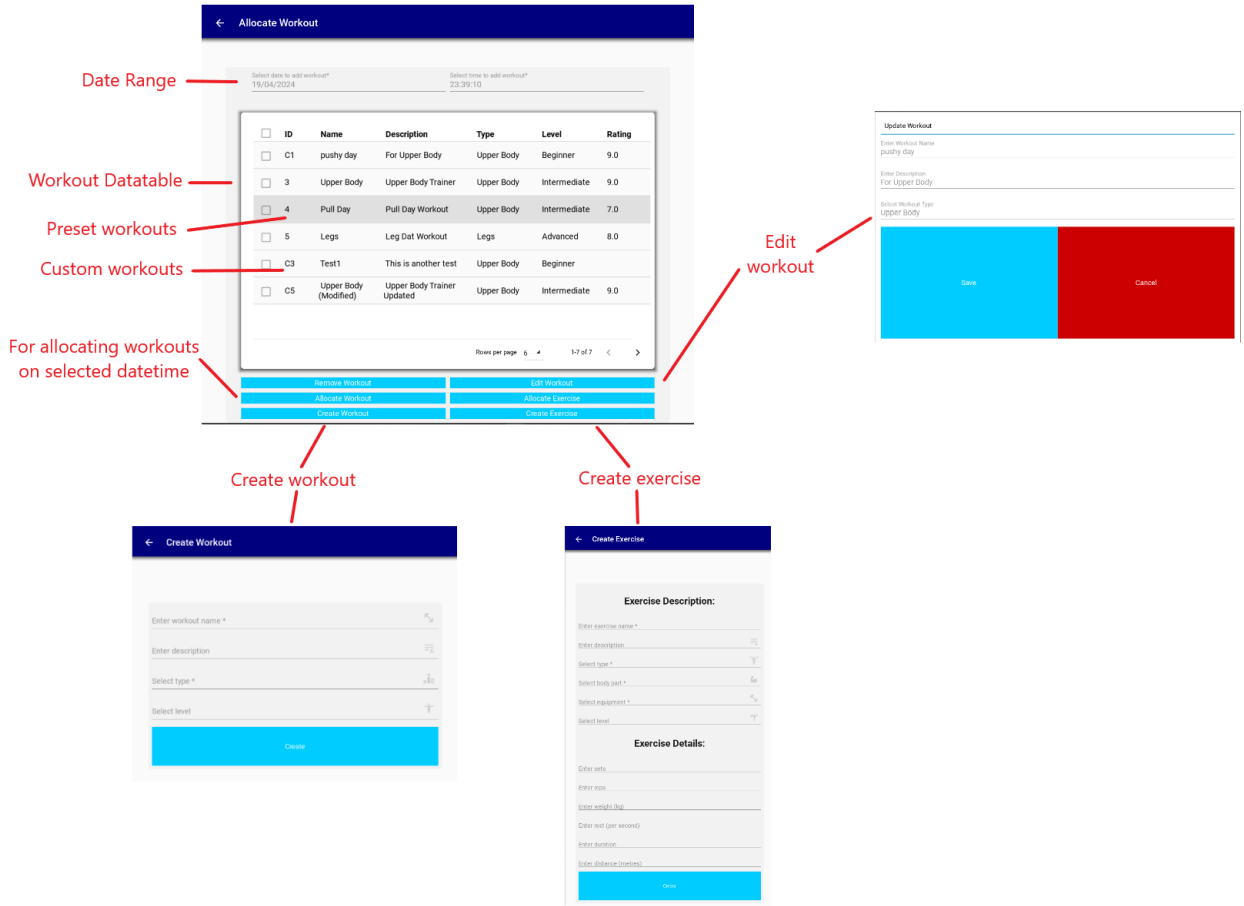


Figure 80: Allocate Workout and Create Exercise and Workout Screens.

Selecting `Workout Manager` in the Workout History Screen will direct the user to the Allocate Workout screen, where the user can manage the Workout and Exercise classes.

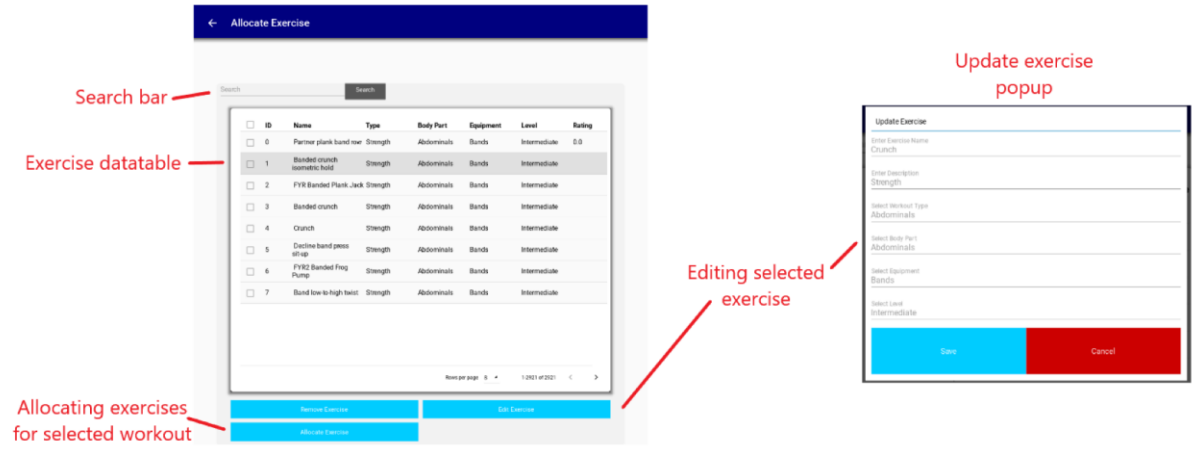


Figure 81: Allocate Exercise Screen.

To access this page a workout has to be selected in the Workout Allocate Screen, then select `Allocate Exercise`.

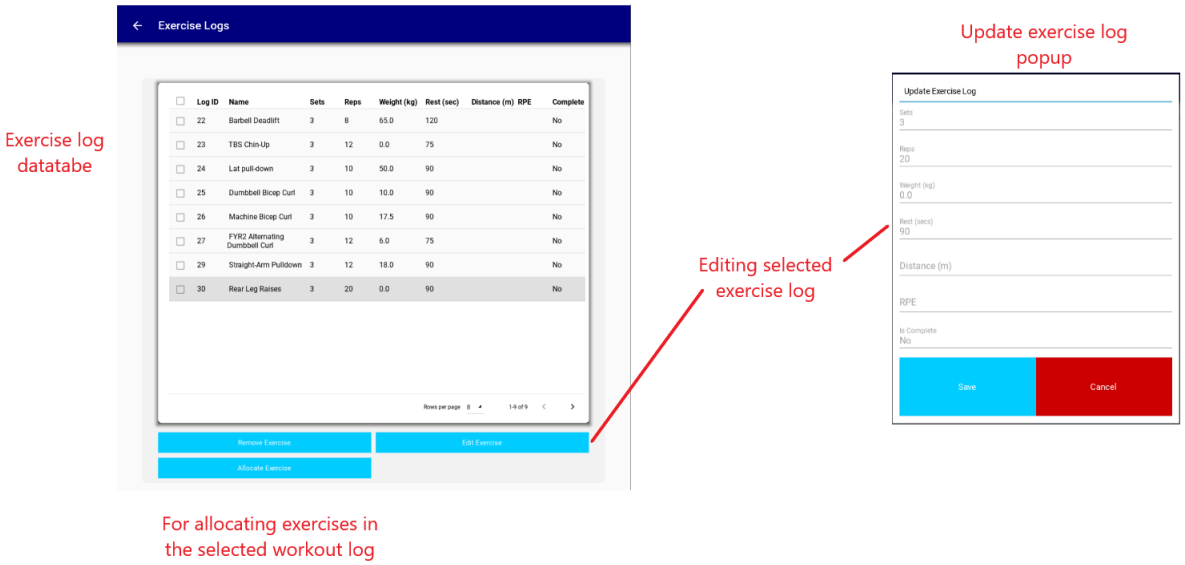


Figure 82: Exercise Logs Screen.

To access this screen, select a workout log in the Workout History Screen, then select `View Workout Log`.

7.3.7. Meal and Water Intake Screens

The meal screens remain incomplete due to time constraints and other project deadlines. Presently, the only partially finished meal screen is the Meal History Screen, responsible for displaying meal logs and supporting only the delete functionality.

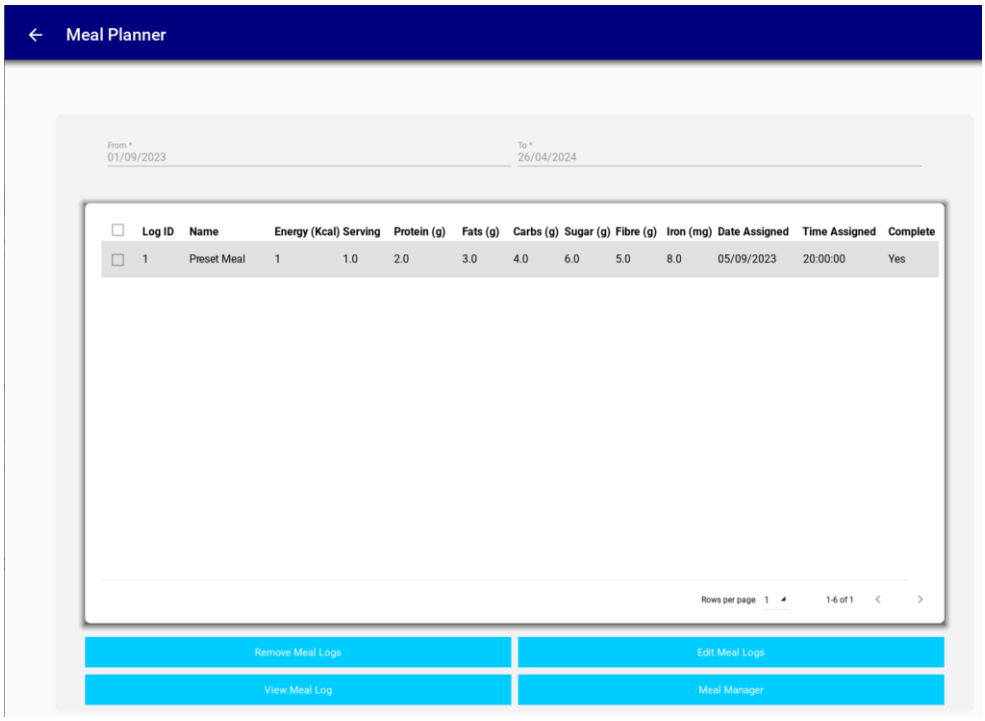


Figure 83: Meal History Screen.

Regarding water intake, the Water Intake Screen has been developed to monitor and track water intake. Additionally, this screen generates water intake plots for users using the matplotlib library within the Kivy framework.

```
def plot_monthly_water_intake(self, selected_month, selected_year):
    local_db = LocalDB('local_db.db')
    water_intake = WaterIntake(local_db.connection)
    try:
        box1 = self.ids.box1
        fig, ax = plt.subplots(figsize=(250, 150))
        canvas = fig.canvas
        water_intake_data = water_intake.monthly_water_intake_data(selected_month, selected_year)
        plt.bar(water_intake_data[0], water_intake_data[1], color='b', alpha=0.7)
        plt.title(f'Monthly Water Intake for {selected_month}/{selected_year}')
        plt.xlabel('Day of the Month')
        plt.ylabel('Water Intake (ml)')
        plt.axhline(y=2000, color='r', linestyle='--')
        plt.xticks(water_intake_data[2])
        plt.grid(axis='y')
        canvas = FigureCanvasKivyAgg(plt.gcf())
        box1.add_widget(canvas)

    except Exception as e:
        print("Error:", e)
    finally:
        local_db.close_connection()
```

Figure 84: Plotting Water Intake Monthly Graph.

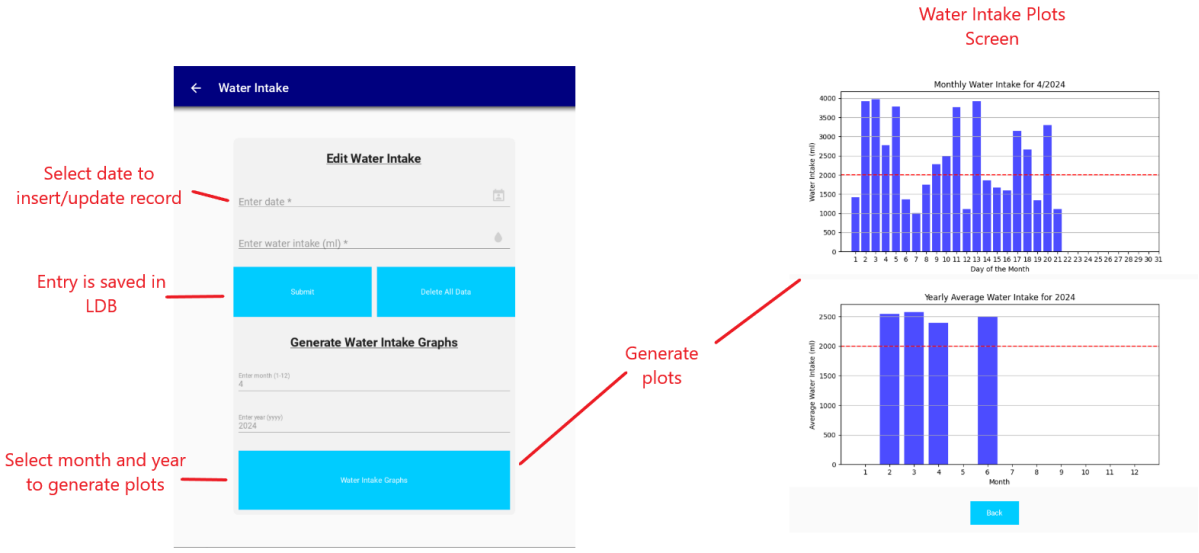


Figure 85: Water Intake & Plots Screens.

7.3.8. Gym Finder Screen

In the GymFinderScreen class, a map API is integrated to display a map view for locating nearby gyms. The map view utilizes the MapView widget within Kivy, with a default location set to Bristol, England. However, the functionality for searching for the nearest gym is not fully implemented due to time constraints.

```
# Create map view, currently default location is set to Bristol, England.
def create_map_view(self):
    mapview = MapView(zoom=15, lat=51.454514, lon=-2.587910)
    gym_finder_map = self.ids.gym_finder_map
    gym_finder_map.clear_widgets()
    gym_finder_map.add_widget(mapview)
```

Figure 86: Implementing MapAPI for Gym Finder Screen.

Figure 87: Gym Finder Screen.

7.3.9. Body Statistics Screens

Similar to the Water Intake Screen, the Body Statistics Screen enables users to input and monitor various body-related metrics such as weight, height, body fat, BMI, skeletal muscle, and steps taken.

Users can view this data through plots generated using the matplotlib library on the Body Statistics Plots Screen.

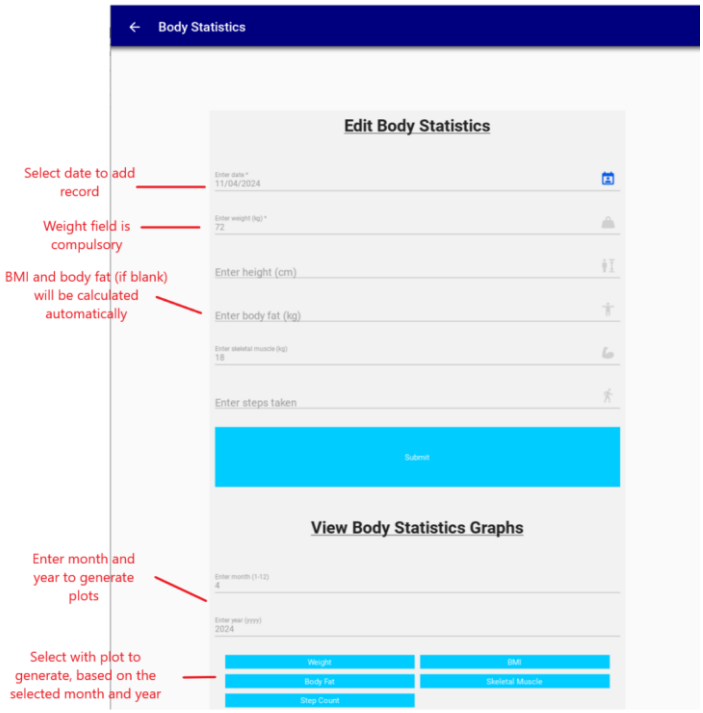


Figure 88: Body Statistics Screen.

The graphs displayed show monthly body statistics, followed by a yearly average graph at the bottom, providing comprehensive insights into the user's body metrics over time. It's important to note that the data displayed on the graph covers the period from February 1st to April 20th.

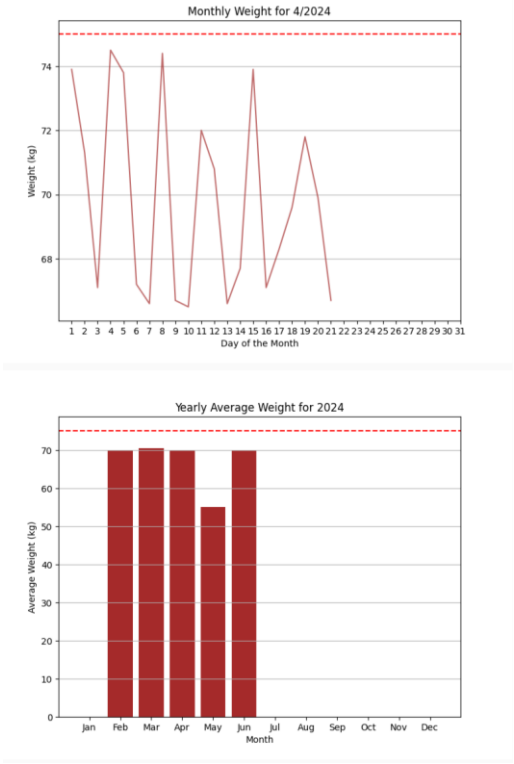


Figure 89: Monthly and Yearly Average Weight Graphs.

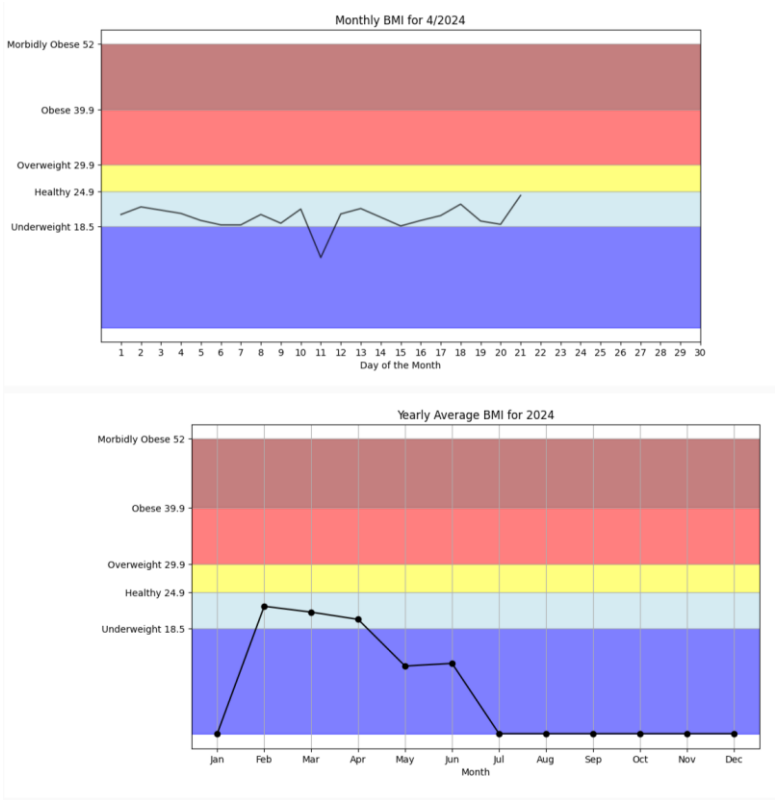


Figure 90: Monthly and Yearly Average BMI Graphs.

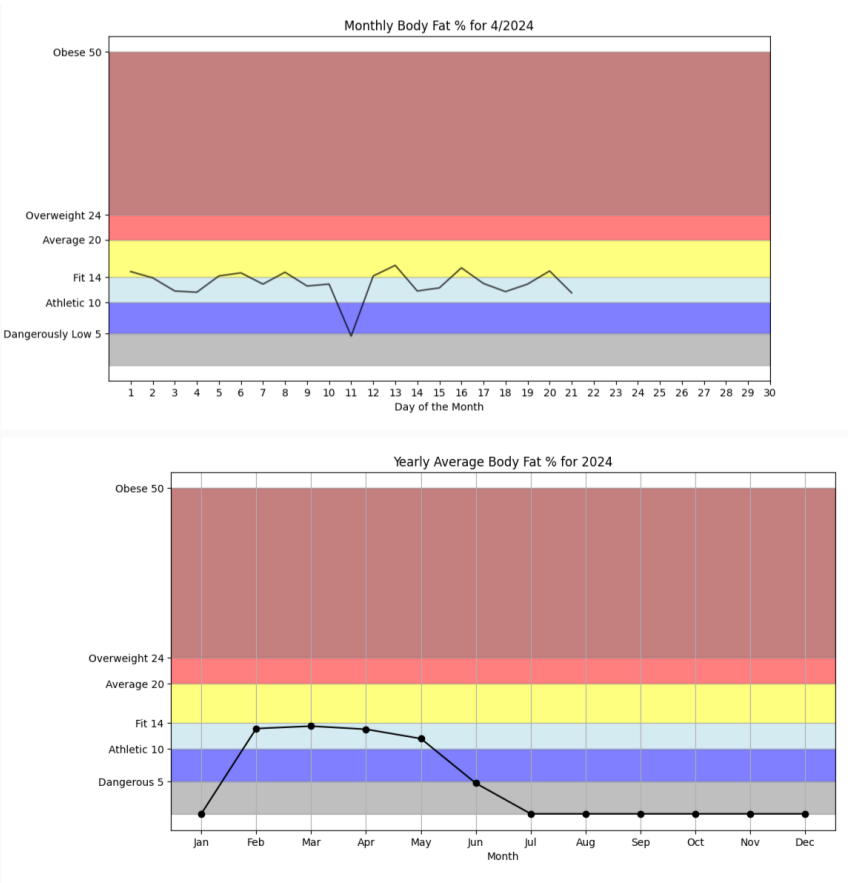


Figure 91: Monthly and Yearly Average Body Fat Graphs.

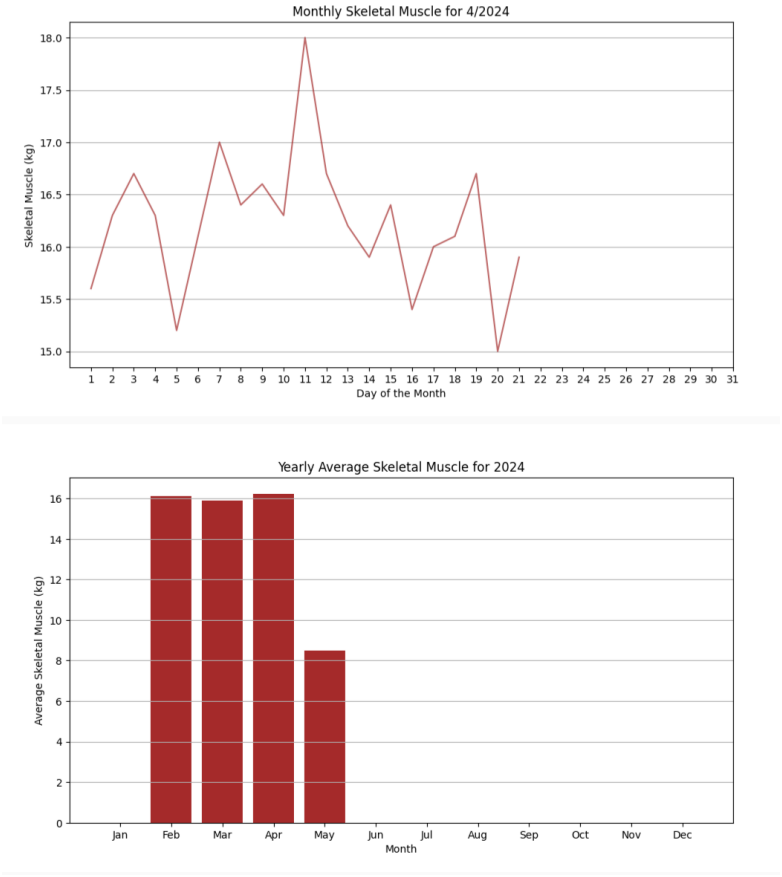


Figure 92: Monthly and Yearly Average Skeletal Muscle Graphs.

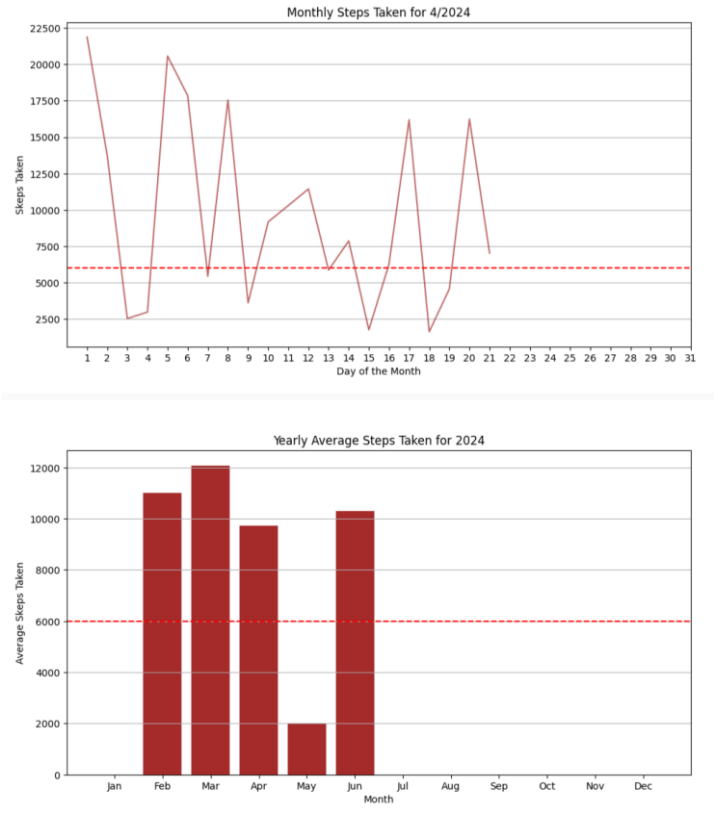


Figure 93: Monthly and Yearly Steps Graphs.

7.3.10. Arduino Watch Screens

The Arduino Watch Screens manages synchronization and communication with the Arduino Watch, facilitating the transfer of watch data into the app for users to view body data. Currently, the app can only access this data via a wired connection to the microSD card, where the data is read and displayed as plots in the AFNT app. Bluetooth functionality was not fully implemented due to time constraints and delays in receiving Arduino Watch parts like the Bluetooth LE TinyShield modular component.

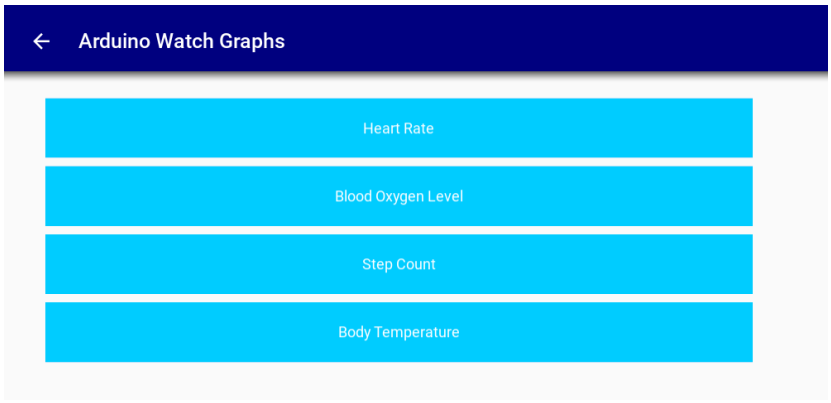


Figure 94: Arduino Watch Screen.

After successfully connecting the microSD card data to the computer, users can view the data in the form of mathplotlib graphs in the Arduino Watch Plots Screen as shown in the following plots:

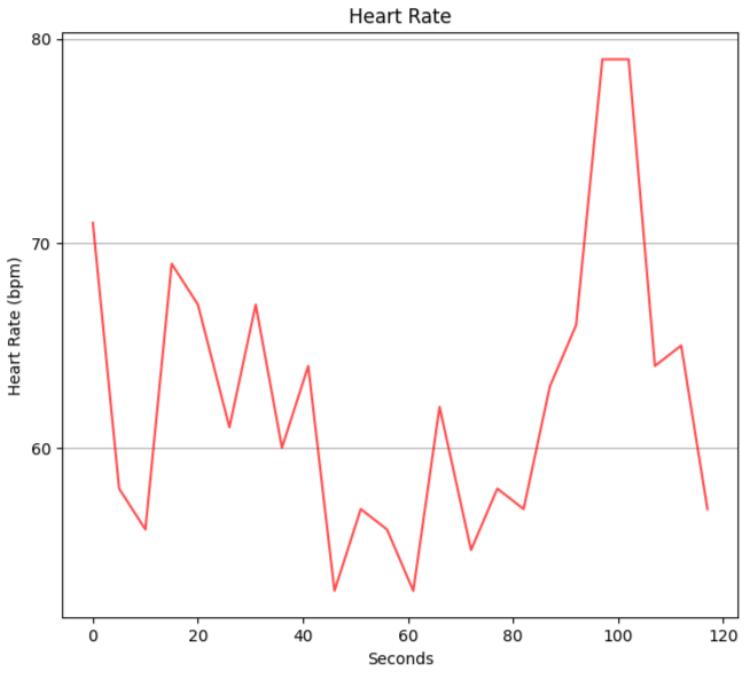


Figure 95: Heart Rate Plot.

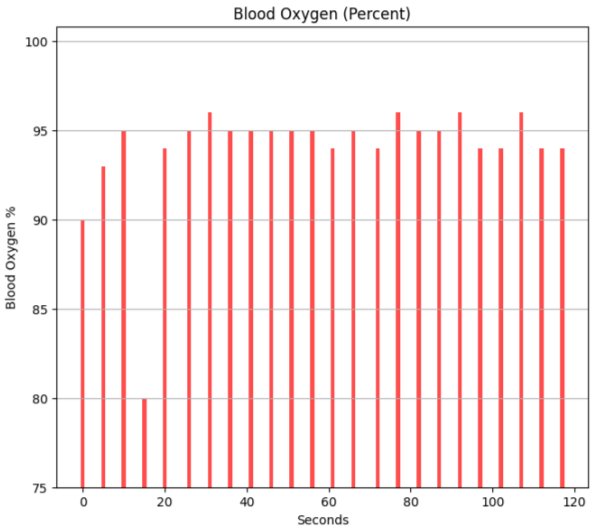


Figure 96: Blood Oxygen Level Plot.

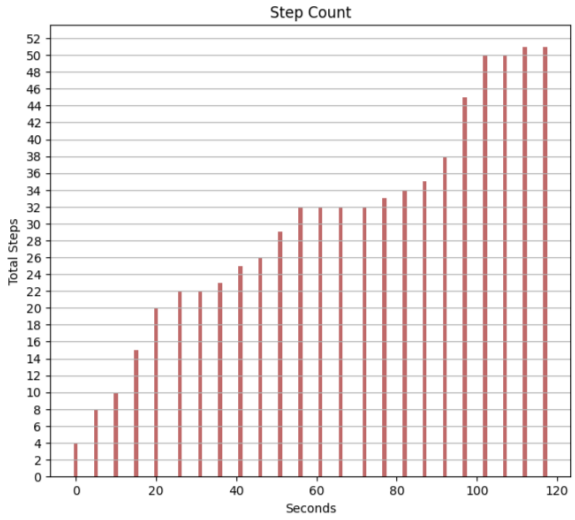


Figure 97: Step Count Plot.

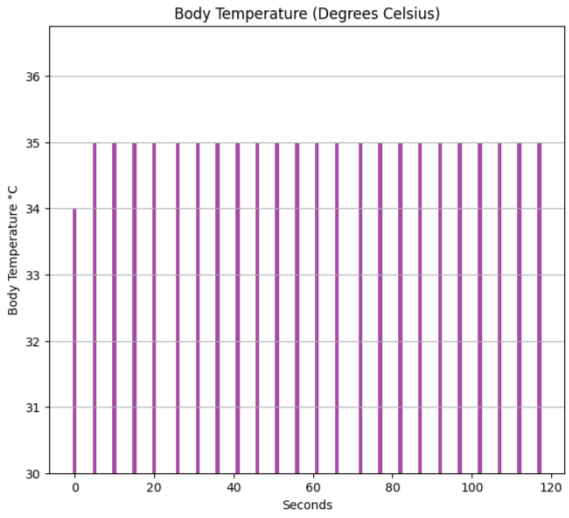


Figure 98: Body Temperature Plot.

To explore further details on how the Arduino Watch collects and stores this data, refer to section 7.4.

7.4. Phase 4: Arduino Watch Implementation

As previously discussed, the Arduino Watch comprises 8 modular electronic components interconnected like Lego blocks, as depicted in the diagram below. We will delve into how this stack was assembled and explore how the Arduino Watch captures, records, and presents physiological data collected through its sensors. Additionally, we will examine how the accelerometer detects and measures steps.

7.4.1. Stack Implementation and TinyShield Compatibility

Figure 60: Arduino Watch Modular Component stack diagram.

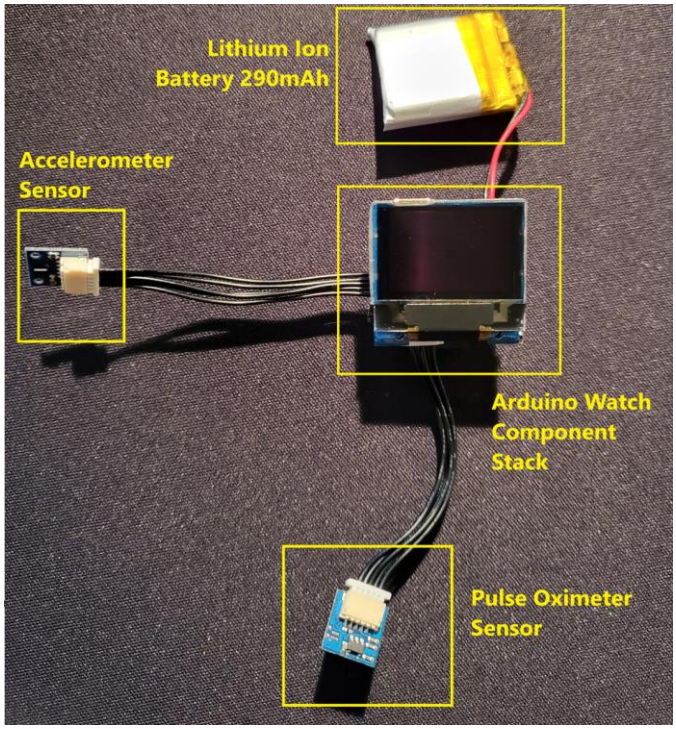


Figure 99: Arduino Watch component implementation.

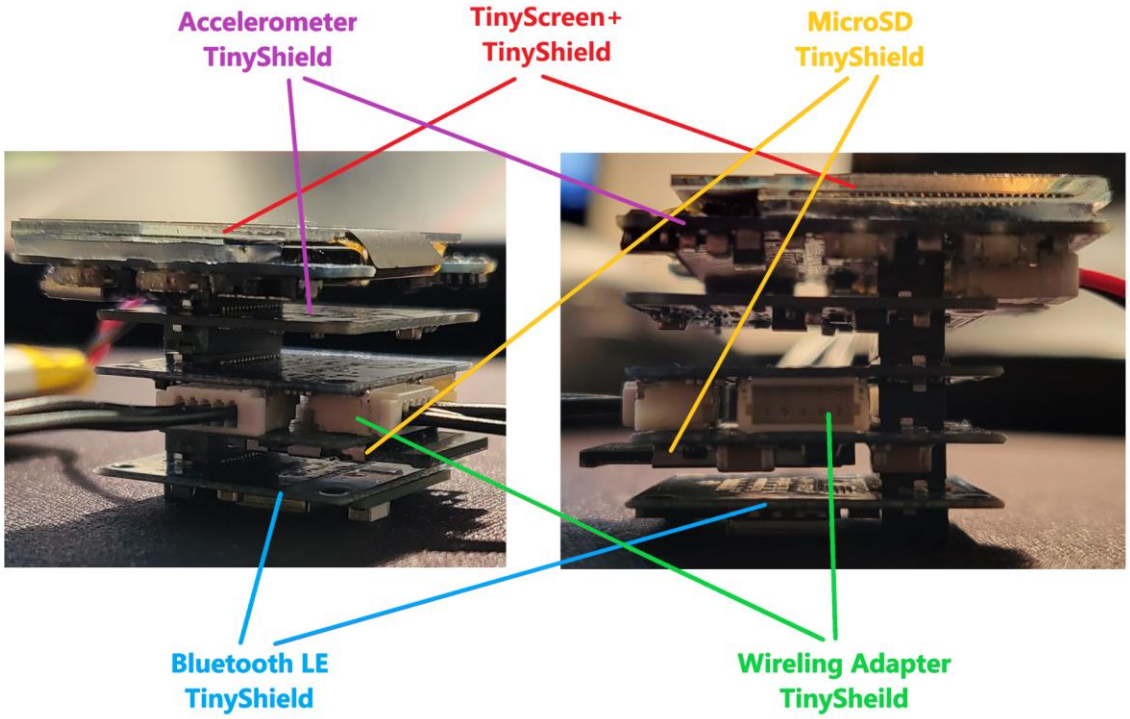
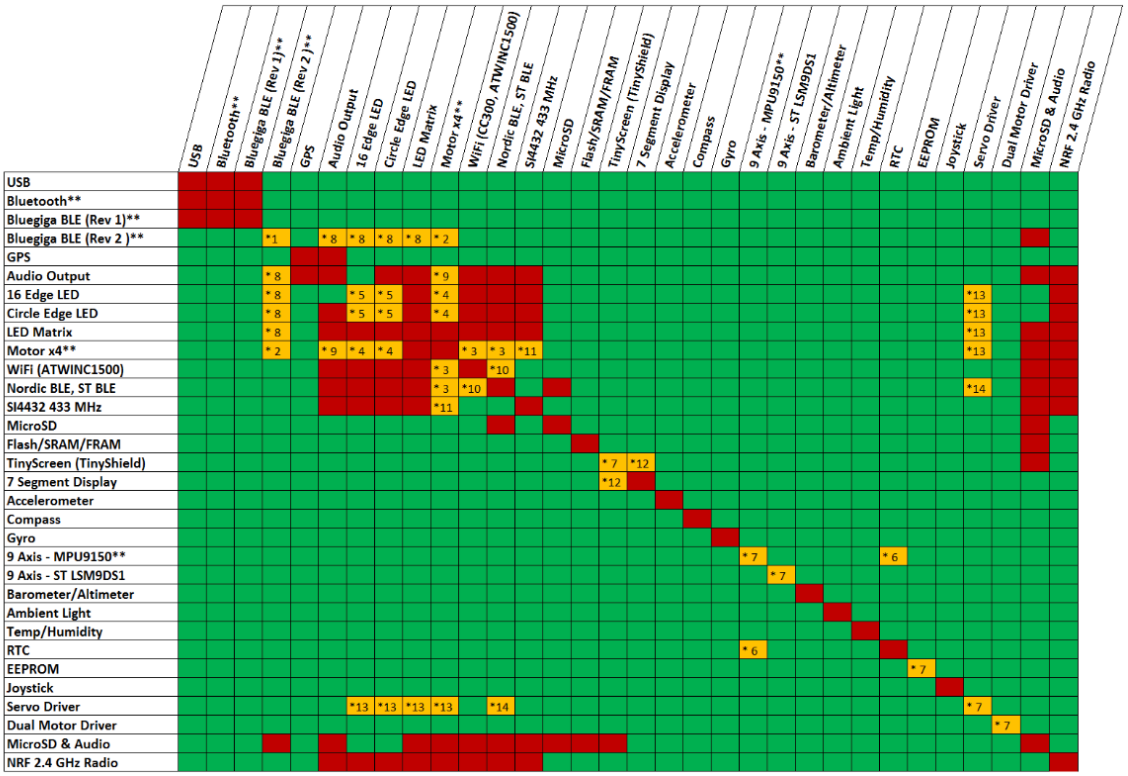


Figure 100: Implementing Arduino Watch stack.

The Arduino Watch stack can be assembled and rearranged by attaching the components together, and the order of the stack can be adjusted easily. This rearrangement does not impact the watch's behaviour or necessitate any modifications to the code, as the TinyDuino and TinyShields automatically adapt to the changes. However, if you change the sensor ports in the Wireling Adapter TinyShield, you may need to make minor code modifications to ensure that the sensor is assigned to the correct port and initialized properly for accurate data collection.

Table 35: TinyShield Compatibility. ('TinyDuino Overview', 2024)



Green: The two TinyShields can be used together on the same stack without issue.

Yellow: Special conditions.

Red: The two boards cannot be used together.

7.4.2. Pulse Oximeter Sensor Configuration

```
// Which port is each sensor connected to.
int pulseSensorPort = 2;
int accelSensorPort = 3;
```

Figure 101: Senor port allocations in the Wireling Adapter TinyShield.

```
// Function for checking the heart rate and blood oxygen percentage.
void checkPulse()
{
  if (pulseSensor.update()) {
    if (pulseSensor.pulseValid()) {
      beatAvg = pulseSensor.BPM();
      oxygenLevel = pulseSensor.oxygen();
      temperature = pulseSensor.temperature();
      SerialUSB.println("beatAvg, oxy, temp");
      SerialUSB.println(beatAvg);
      SerialUSB.println(oxygenLevel);
      SerialUSB.println(temperature);
    }
  }
}
```

Figure 102: checkPulse function.

The function checkPulse is used to retrieve data recorded by the pulse oximeter, providing information on the person's heart rate (BPM), oxygen level (%), and body temperature (°C). The sensor can also capture cardiogram data, although this functionality is currently not within the scope of the project.

7.4.3. Accelerometer Sensor Configuration and Step Calculation

For step data, the step calculation involves utilizing data from an accelerometer sensor, which measures acceleration along three axes: X, Y, and Z. These axes represent different spatial directions:

Axes: The X, Y, and Z axes correspond to different spatial directions relative to the orientation of the sensor. The X-axis typically represents horizontal movement from side to side, the Y-axis represents horizontal movement forward and backward, and the Z-axis represents vertical movement up and down (Volkwyn *et al.*, 2020).

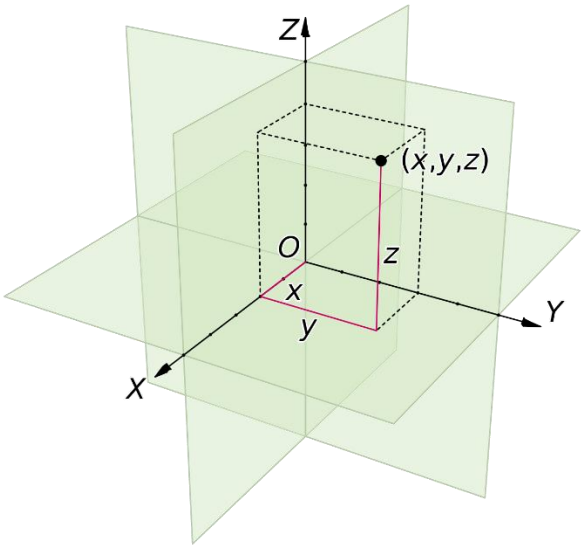


Figure 103: 3D Cartesian Coordinate System (Jorge S., 2018).

There is an origin labelled as O, along with axis lines X, Y, and Z as indicated by arrows. The tick marks on these axes are spaced one unit apart. A black dot represents a point located at coordinates x = 2, y = 3, and z = 4, denoted as (2, 3, 4).

Magnitude (of a Vector): In this context, magnitude refers to the overall acceleration experienced by the device. It is calculated using the accelerometer readings along the X, Y, and Z axes. The

magnitude (`A`) is computed as the square root of the sum of the squares of the individual axis values (`X`, `Y`, `Z`), mathematically represented as:

$$|A| = \sqrt{x_1^2 + y_1^2 + z_1^2}$$

Figure 104: Magnitude of a Vector formula (GeeksforGeeks, 2022).

Here, `X`, `Y`, and `Z` are the acceleration values along the respective axes.

The step detection algorithm uses changes in this acceleration magnitude (`A`) to detect when a step occurs. By analyzing the difference between the current magnitude value (`A`) and a previously recorded value (`B`), the algorithm can infer changes in movement indicative of steps. Thresholds and peak-to-peak ranges are used to distinguish meaningful step-like motions from other movements. When a significant change in acceleration aligns with predefined criteria for a step, such as specific ranges or patterns, the step count is updated to reflect the occurrence of a step. This approach translates physical movements, detected through acceleration data along multiple axes, into digital step counts within the program logic.

```
// Record the current time
lastSample = millis();

// Read new data from the accelerometer
accel_sensor.read();

// Calculate the magnitude (A) of the acceleration vector
int sum = pow(accel_sensor.X, 2);
sum += pow(accel_sensor.Y, 2);
sum += pow(accel_sensor.Z, 2);
int A = sqrt(sum);

// Compute the difference between the current and previous acceleration magnitudes
int difference = abs(A - sampleOld);

// Apply filtering to smooth out small changes
if (difference < 10) {
  difference = 0;
}
if (difference > 30) {
  difference = 30;
}

// Determine the sign of the change in acceleration magnitude
if (A - sampleOld > 0) {
  difference = sampleOld + difference;
} else {
  difference = sampleOld - difference;
}

// Store the difference in the acceleration buffer
aBuff[aBuffPos] = difference;

// Find the minimum and maximum values in the acceleration buffer
int Amin = 1000;
int Amax = 0;
for (int i = 0; i < amtSamples; i++) {
  int sampleVal = aBuff[i];
  if (sampleVal < Amin) Amin = sampleVal;
  if (sampleVal > Amax) Amax = sampleVal;
}

// Calculate the peak-to-peak value and set a threshold
int peakToPeak = Amax - Amin;
int threshold = (peakToPeak / 2) + Amin;

// Update the previous sample value
sampleOld = sampleNew;

// Check for a potential new step based on acceleration changes
bool newStep = false;
if (abs(sampleNew - difference) > precision) {
  sampleNew = difference;
  if (peakToPeak > 70 && sampleOld > threshold && sampleNew < threshold) {
    // Check if the step falls within a valid time interval
    if (validStepPattern) {
      if (millis() > lastStepTime + stepIntervalLow && millis() < lastStepTime + stepIntervalHigh) {
        newStep = true;
      } else {
        validStepPattern = false;
      }
    } else if (millis() > lastStepTime + stepIntervalLow && millis() < lastStepTime + stepIntervalHigh) {
      newStep = true;
      validStepPattern = true;
    }
    // Update the last step time and increment the step count
    lastStepTime = millis();
    if (newStep) {
      if (false) {
        stepAlert = true;
      }
      totalSteps++;
      // Display the total steps on the serial monitor
      SerialUSB.print("totalSteps:  ");
      SerialUSB.println(totalSteps);
      // Add the timestamp of the most recent step to the timestamps array
      stepTimestamps[firstMinusOne()] = millis();
```

Figure 105: Step calculation.

7.4.4. Logging Data to microSD Card

The Arduino Watch records all collected data, including the date and time, into a CSV file named `arduino.csv` at 5-second intervals.

```
// Check if it's time to log data (This logs data into the SD card every 5 seconds by default).
if (millis() - batteryTimer > FAST_DATA_INTERVAL)
{
  SerialUSB.print("Current millis(): ");
  SerialUSB.println(millis());
  SerialUSB.print("Previous batteryTimer value: ");
  SerialUSB.println(batteryTimer);
  SerialUSB.print("FAST_DATA_INTERVAL: ");
  SerialUSB.println(FAST_DATA_INTERVAL);

  // Log data here.
  createString(displayString, dataString, firstSD);
  validateSD(dataString, displayString, firstSD);

  // Update batteryTimer to the current millis() value.
  batteryTimer = millis();
}
```

Figure 106: Log data every 5 seconds in the Arduino Watch.

The `createString` function collects physiological data, formats it as a CSV string, and returns it. The `validateSD` function verifies that the watch is equipped with a microSD card containing the required CSV file.

```
// Function that writes body data collected by the Arduino Watch to the SD card (in arduino.csv).
void logData(String dataString, String displayString) {
  uint16_t data[ANALOG_COUNT];

  // Read all channels to avoid SD write latency between readings.
  for (uint8_t i = 0; i < ANALOG_COUNT; i++) {
    data[i] = analogRead(i);
  }

  if (DEBUG_MD) {
    SerialUSB.println("WRITING TO FILE!!");
  }

  if (firstSD) {
    // If it's the first time writing to the file, write column headings.
    file.println(displayString);
    // Set firstSD to false after writing column headings.
    firstSD = false;
  }

  // Append dataString (which contains the data row) to the file.
  file.println(dataString);

  if (DEBUG_MD) {
    SerialUSB.println("WRITING Complete!");
    SerialUSB.print("dataString: ");
    SerialUSB.println(dataString);
  }
}
```

Figure 107: logData function.

The `logData` function is responsible for writing the collected data into the `arduino.csv` file located on the microSD card.

Table 36: Logging watch data in `arduino.csv`.

epoch_time	datetime	step_count	step_goal_percent	heart_rate	oxygen_level	temperature_celsius	battery_percent
1712448005	24/04/2007 00:00	4	0	71	90	34	96
1712448010	24/04/2007 00:00	8	0	58	93	35	97
1712448015	24/04/2007 00:00	10	0	56	95	35	97
1712448020	24/04/2007 00:00	15	0	69	80	35	96
1712448025	24/04/2007 00:00	20	0	67	94	35	97
1712448031	24/04/2007 00:00	22	0	61	95	35	97
1712448036	24/04/2007 00:00	22	0	67	96	35	96
1712448041	24/04/2007 00:00	23	0	60	95	35	97
1712448046	24/04/2007 00:00	25	0	64	95	35	99
1712448051	24/04/2007 00:00	26	0	53	95	35	98
1712448056	24/04/2007 00:00	29	0	57	95	35	98
1712448061	24/04/2007 00:01	32	0	56	95	35	97
1712448066	24/04/2007 00:01	32	0	53	94	35	97
1712448071	24/04/2007 00:01	32	0	62	95	35	97
1712448077	24/04/2007 00:01	32	0	55	94	35	97
1712448082	24/04/2007 00:01	33	0	58	96	35	96
1712448087	24/04/2007 00:01	34	0	57	95	35	98
1712448092	24/04/2007 00:01	35	0	63	95	35	97
1712448097	24/04/2007 00:01	38	0	66	96	35	97
1712448102	24/04/2007 00:01	45	0	79	94	35	97
1712448107	24/04/2007 00:01	50	0	79	94	35	97
1712448112	24/04/2007 00:01	50	0	64	96	35	96
1712448117	24/04/2007 00:01	51	0	65	94	35	97
1712448122	24/04/2007 00:02	51	0	57	94	35	98

7.4.5. Watch Main Display

The watch display presents the current date and time along with information on steps, step goal percentage, blood oxygen level, heart rate, and battery level.

Figure 108: Arduino Watch display.

```
void displayData(unsigned long &screenClearTime)
{
    int totalSteps = getTotalSteps();
    int totalStepsPercent = percentOfDailyStepGoal(totalSteps);
    int batteryPercent = getBatteryPercent();

    if (rtc.getSeconds() == 0 && millis() - screenClearTime > 1000) {
        display.clearScreen();
        screenClearTime = millis();
    }

    // Turn on the display.
    display.on();

    display.setCursor(0, 0);
    display.print(rtc.getDay());
    display.print("/");
    display.print(rtc.getMonth());
    display.print("/");
    display.print(rtc.getYear());
    display.print(" ");
    display.print(rtc.getHours());
    display.print(":");
    display.print(rtc.getMinutes());
    display.print(":");
    display.print(rtc.getSeconds());

    display.setCursor(0, 10);
    display.print("Steps: ");
    display.println(totalSteps);

    display.setCursor(0, 20);
    display.print("Steps %: ");
    display.println(totalStepsPercent);

    display.setCursor(0, 30);
    display.print("Heart Rate: ");
    display.println(beatAvg);

    display.setCursor(0, 40);
    display.print("Oxygen Level: ");
    display.println(oxygenLevel);

    display.setCursor(0, 50);
    display.print("Battery %: ");
    display.println(batteryPercent);
}
```

Figure 109: Arduino Watch display data code.

7.4.6. Bluetooth Low Energy Functionality

The Bluetooth Low Energy (BLE) communication between the Arduino Watch and a smartphone has been tested and implemented. However, at this moment, the synchronization and transmission of body data from the Arduino Watch via BLE have not been implemented due to time constraints. The STBLE library is essential for enabling BLE functionality and can be customized to adjust BLE behaviour as needed.

```
UART service added successfully.
General Discoverable Mode.
Connected to device:49-46-56-EE-65-F4
```

Figure 110: Arduino Watch connecting to a smartphone.

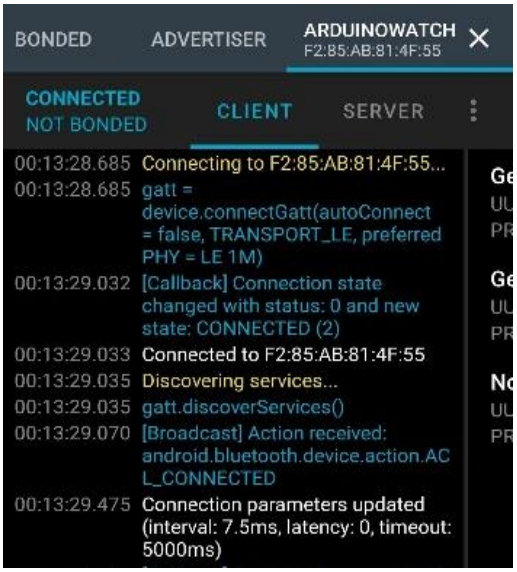


Figure 111: Andriod smartphone connecting to the Arduino Watch.

Once the connection between the Android smartphone and the Arduino Watch is established, data can be exchanged bidirectionally between the devices. The BLE functionality was verified using the 'nRF Connect' app, available for both iOS and Android platforms, which supports BLE communication and facilitates sending and receiving instructions to and from the Arduino Watch.

Figure 112: nRF Connect application.

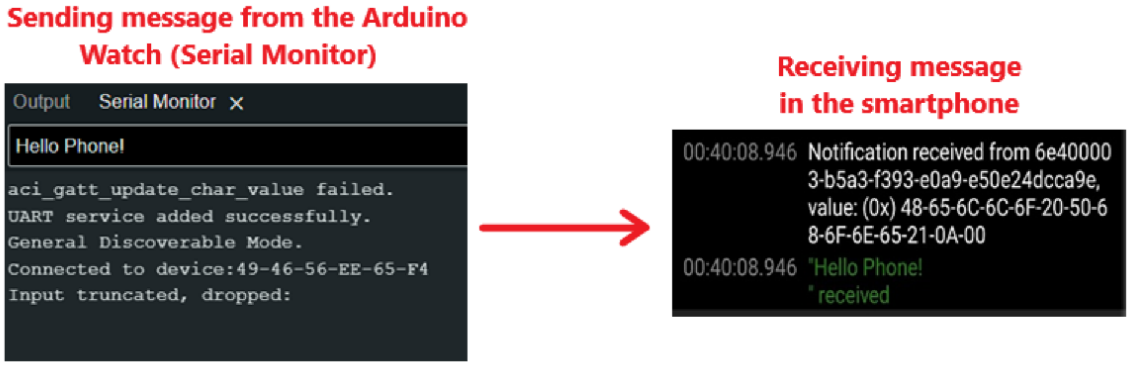


Figure 113: Sending a message from the Arduino Watch to the smartphone.

Figure 114: Receiving a message from the smartphone to the Arduino Watch.

This code has certain limitations; currently, only 20 characters of data can be sent at a time.

In summary, the AFNT project has made significant strides in implementing its core functionalities across multiple phases. Notably, Phase 1 has achieved over 90% completion, focusing on database design and setup, while Phase 3 and Phase 4 are approximately 70% implemented, emphasizing the development of the AFNT application and integration with the Arduino Watch. Despite challenges and constraints, such as time limitations and project priorities, the project has leveraged a range of technologies and frameworks to realize its objectives effectively. Moreover, this chapter underscores the importance of each phase's contributions to the overall success of the AFNT project and highlights areas for further exploration and refinement in subsequent project stages.

8. Project Evaluation

This chapter aims to evaluate the AFNT project by assessing the progress and achievements across its key phases. This evaluation encompasses a review of project objectives, agile development practices, successes, challenges encountered in each phase, and comprehensive testing strategies.

8.1. Requirements and Objectives

In summary, the AFNT project is driven by five main objectives. Objective 1 is nearly complete but requires additional data and testing for the CDB, while the LDB has been fully implemented. Objective 2 has achieved significant progress but lacks meal tracking, gym locater, cross-platform and administrative management features, leading to Phase 2 (O3) development, which was limited due to time constraints. Objective 4 has fulfilled most of its requirements, including connection with the AFNT app, but lacks BLE communication and robust synchronization techniques with the application, primarily due to time limitations. Finally, Objective 5 focuses on ensuring clean and manageable program code, a goal that has been achieved, although further optimization remains possible.

ID	Objective	Description	Created
O1	Develop a Scalable and Secure Database Management System (DBMS)	Develop a scalable MySQL DBMS, including a central database for 2,000+ records and a local database for user data and watch data. Improve database performance and security and ensure GDPR compliance.	09/10/2023
O2	Build the Alistana Fitness & Nutrition Tracker (AFNT) Application	Develop the AFNT application with features for workout and nutrition tracking, body progress and measurement tracking. Should be connected to DBMS, Admin Management website (only for Admins), and Arduino watch.	09/10/2023
O3	Design a User-Centric Admin Management (AM) Website	Develop a responsive website with secure login and allows Admins to edit the central database (user login data and preset workout and meal data) and app push updates to the AFNT application. The website will prioritize a user-friendly design, encrypted communication, and security measures.	09/10/2023
O4	Design and develop a Fitness Watch using Arduino	Create an Arduino-based Fitness watch to measure blood oxygen level, heart rate and step count, and investigate ways of connecting the Arduino watch to the AFNT and store body data in the local database via AFNT.	16/10/2023
O5	Enhance Code Quality and Performance	Implement clean, maintainable code with 80% code coverage. Optimize application and website response times to under 2 and 3 seconds, respectively.	25/10/2023

Figure 115: AFNT Objectives.

As for project requirements, the project currently encompasses a total of 117 requirements, which have undergone several reviews and revisions throughout the project's timeline. Among these requirements, 10 of these are categorized as "W" (won't have). Each requirement's completion contributes to the project's progress and goal achievement. A scoring system is employed where completed requirements score a full point, while those in progress or partially complete score 0.5, and requirements not yet implemented score 0. Using this scoring method, the overall progress of the AFNT project is estimated at around 74%. This calculation reflects the advancement of the project in meeting its objectives, though it does not consider the priority levels assigned to requirements based on the MoSCoW framework. Below is a table illustrating the current progress status of the requirements.

Table 37: Functional Requirement Progress.

Functional Requirements	Total Requirements	Not Implementing	Not Started	Partially Complete	Complete
Phase 1: DBMS	11	0	0	1	10
Phase 2: AM Website	10	7	1	1	1
Phase 3: AFNT Application	45	2	4	12	27
Phase 4: Arduino Watch	9	0	0	4	5

Table 38: Non-Functional Requirement Progress.

Non-Functional Requirements	Total Requirements	Not Implementing	Not Started	Partially Complete	Complete
Application	13	0	2	5	6
Performance	4	0	0	1	3
Efficiency & Sustainability	5	0	0	1	4
Data Storage & Optimization	1	0	0	0	1
Privacy & Security	4	0	1	3	0
Reliability	5	0	0	5	0
Usability	7	1	2	4	0
Data Backup & Recovery	2	0	0	0	2
Third-Party Service Integration	1	0	0	0	1

8.2. Agile and Kanban

Utilizing Trello for agile development greatly facilitated the identification, division, and implementation of tasks based on project requirements and phases, which were derived from UML diagrams and user stories. This approach enabled efficient tracking and management of tasks and requirements. The use of a Gantt chart and sprint planner further improved project management, allowing for iterative feedback and continuous enhancement. Overall, this experience has been invaluable, providing essential skills in managing large, long-term projects and enabling comfortable adaptation to industry-standard Kanban boards and agile methodologies.

8.3. Successes and Challenges in Each Phase

8.3.1. Phase 1

Phase 1 and 1.5 progressed smoothly in terms of development and meeting deadlines (Sprint 1), leveraging my extensive experience in database design and development gained throughout the course. The implementation of the LDB and CDB proceeded seamlessly. However, I encountered challenges with testing and acquiring data to populate the database. To address this, I sourced data samples from online platforms like Kaggle, which required adjustments to align with database requirements and ensure data security.

8.3.2. Phase 2

Following the completion of Phase 1, I transitioned into Phase 2 of the project, delving into server technologies and Python Flask. Despite facing time constraints and lower project priority, I initiated research and groundwork on server setups and Flask implementation. While unable to fully complete this phase, I successfully created wireframes for the website and made significant progress in website development, particularly by implementing the login functionality. However, delays in the design and planning phases of subsequent project stages, namely Phase 3 and the newly introduced Phase 4, posed challenges in sustaining momentum for Phase 2 development. Nevertheless, this experience provided valuable insights into server technologies and Python Flask, enriching my understanding of these domains.

8.3.3. Phase 3

Phase 3 encountered persistent delays attributed to iterative design changes and evolving requirements for the AFNT App. After finalizing the design, planning, and research on Kivy GUI, Phase 3 development commenced shortly before the Project in Progress week (Sprint 2), resulting in a significant three-week delay. Despite these setbacks, substantial progress was made between February and March (Sprint 3), with the completion of most Phase 3 requirements, including workout management, body data tracking, and water intake monitoring features. However, critical tasks such as nutrition tracking were left unfinished, and the gym tracking feature, though partially implemented with the integration of MapAPI, remained incomplete due to time constraints and the impending need to shift focus to Phase 4, which presented greater challenges as the project deadline drew near.

8.3.4. Phase 4

Phase 4 started positively with thorough research to acquire the necessary components for the Arduino Watch, tailored to meet project specifications. I opted for TinyCircuits and purchased their modular components due to their compact size and ease of assembly without soldering, despite the higher cost. This investment proved valuable as it provided me with substantial knowledge in Arduino development, an area I had not explored previously. The bulk of Arduino Watch implementation occurred from late March to mid-April (Sprint 4), which was challenging given the project deadline at the end of April and associated shipping delays. Despite these constraints, I rapidly quickly learned and implemented foundational body data tracking and recording functionalities outlined in Phase 4. However, due to limited familiarity with Bluetooth Low Energy (BLE) technology, I could not fully complete the BLE implementation or synchronization with the AFNT app. Nonetheless, I successfully established a basic connection between the Arduino Watch and my smartphone, enabling limited message exchange within the project scope.

8.4. Testing

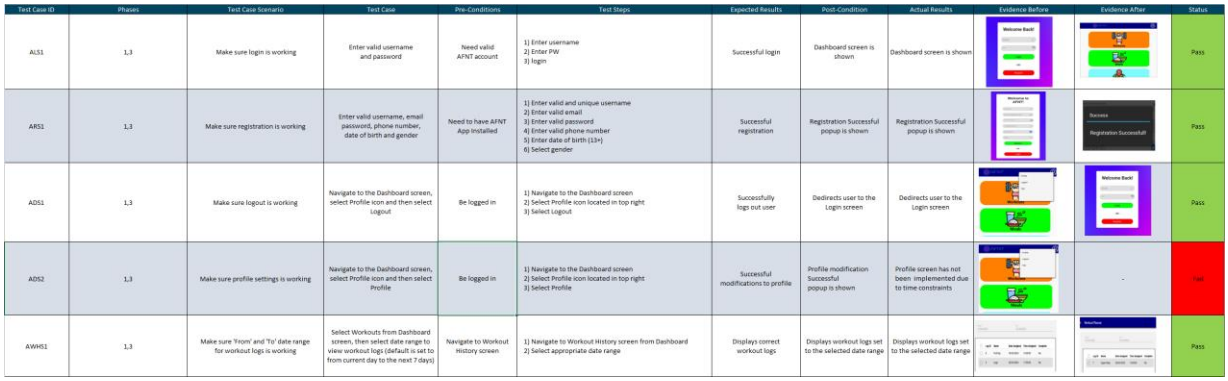
Note: **Click Here** to access the test sheets.

Phases 1, 3, and 4 were thoroughly tested using an open-source agile testing template ('Test Case Template', 2023). Each test case is uniquely identified by an ID that specifies the project phase, test scenario, pre and post-conditions, steps to replicate the test, and expected versus actual results. Additionally, evidence in the form of before and after screenshots accompanies each test case to provide comprehensive validation. Moreover, the test case ID format includes an abbreviation representing the type of test case and the specific project phase being tested, followed by a unique indexing number.

To identify tests for Phases 1 and 3 related to the DBMS and application: The test case ID format includes 'A' to denote the Application, followed by a screen name abbreviation and a corresponding number. For instance, 'AWIPS1' represents *<Application><Water Intake Plots Screen><ID 1>*.

Abbreviation	Description
LS	Login Screen
RS	Registration Screen
DS	Dashboard Screen
WHS	Workout History Screen
WAS	Workout Allocate Screen
EAS	Exercise Allocate Screen
CES	Create Exercise Screen
ELS	Exercise Log Screen
ELAS	Exercise Log Allocate Screen
MHS	Meal History Screen
GFS	Gym Finder Screen
BSS	Body Statistics Screen
BSPS	Body Statistics Plots Screen
WIS	Water Intake Screen
WIPS	Water Intake Plots Screen
AWS	Arduino Watch Screen

Figure 116: Test Case ID Abbreviations (Application).



Test Case ID	Phases	Test Case Scenario	Test Case	Pre-Conditions	Test Steps	Expected Results	Post-Condition	Actual Results	Evidence Before	Evidence After	Status
ALS1	1,3	Make sure login is working	Enter valid username and password	Need valid AFNT account	1) Enter username 2) Enter PW 3) login	Successful login	Dashboard screen is shown	Dashboard screen is shown			Pass
ARS1	1,3	Make sure registration is working	Enter valid username, email password, phone number, date of birth and gender	Need to have AFNT App installed	1) Enter valid and unique username 2) Enter valid email 3) Enter valid password 4) Enter valid phone number 5) Enter date of birth (13+) 6) Select gender	Successful registration	Registration Successful popup is shown	Registration Successful popup is shown			Pass
ADS1	1,3	Make sure logout is working	Navigate to the Dashboard screen, select Profile icon and then select Logout	Be logged in	1) Navigate to the Dashboard screen 2) Select Profile icon located in top right 3) Select Logout	Successfully logs out user	Dedirects user to the Login screen	Dedirects user to the Login screen			Pass
ADS2	1,3	Make sure profile settings is working	Navigate to the Dashboard screen, select Profile icon and then select Profile	Be logged in	1) Navigate to the Dashboard screen 2) Select Profile icon located in top right 3) Select Profile	Successful modifications to profile	Profile modification Successful popup is shown	Profile screen has not been implemented due to time constraints		-	Fail
AWHS1	1,3	Make sure 'From' and 'To' date range for workout logs is working	Select Workouts from Dashboard screen, then select date range to view workout logs (default is set to from current day to the next 7 days)	Navigate to Workout History screen	1) Navigate to Workout History screen from Dashboard 2) Select appropriate date range	Displays correct workout logs	Displays workout logs set to the selected date range	Displays workout logs set to the selected date range			Pass

Figure 117: Screenshot of Application Test Cases.

For identifying Arduino Watch tests: The test case ID format begins with 'AW' followed by a number. For example, 'AW5' denotes <*Arduino Watch*><*ID 5*>.

Due to time constraints and the incomplete BLE connection between the AFNT App and the Arduino Watch, Phase 4 testing did not include a comparison with devices like the Samsung Galaxy Watch or Fitbit Flex. Therefore, a comprehensive comparison was challenging without full implementation. To compensate for the lack of full meal-tracking functionality in AFNT, users can utilize Samsung Health's health-tracking features, while Google Maps can serve as an alternative for gym finder functionality.

Moreover, the evaluation of step counting accuracy and other body data metrics was not conducted due to prioritizing implementation over sensor accuracy during the project. Future efforts will focus on enhancing sensor accuracy and step-counting algorithms once foundational functionalities are robustly implemented. The Arduino Watch aims to evolve by becoming more compact and feature-rich, potentially incorporating GPS, phone call reception, and notifications beyond the project's initial requirements.

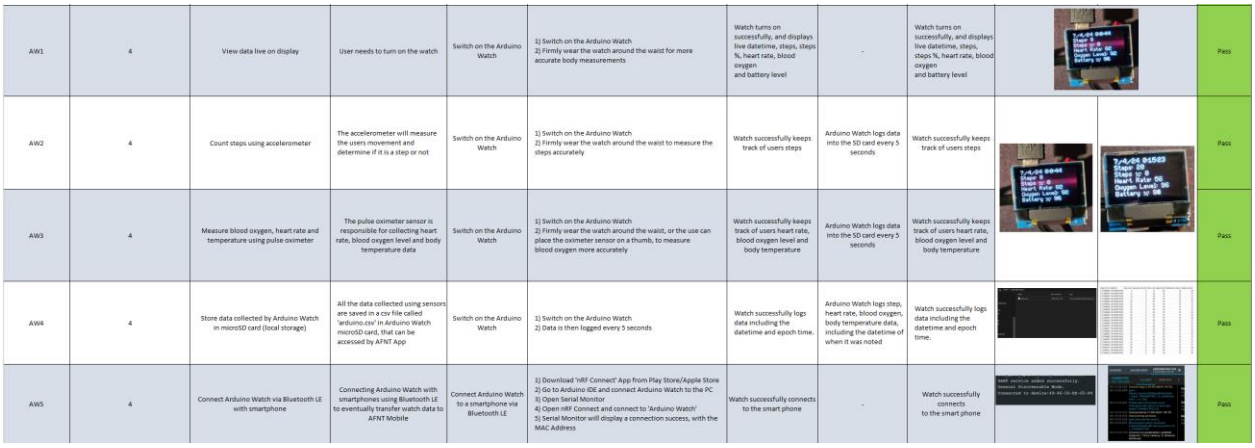


Figure 118: Screenshot of Arduino Watch Test Cases.

There are a total of 51 tests, with 43 dedicated to Phases 1 and 3, and the remaining 8 specific to the Arduino Watch. Out of these tests, 5 have failed primarily due to incomplete implementation of related requirements or functionalities. Since tests are designed based on implemented requirements, there are no tests for unimplemented functionalities. Despite the failures, the overall test results are promising, showcasing the project's progress in terms of functionality. The failed tests are attributed to incomplete features such as nutrition tracking, gym tracking, and the Arduino Watch's BLE functionality, which were constrained by time limitations.

Agile testing strategies, like the one used here, played a crucial role in the iterative development of the AFNT project by continuously assessing project functionalities against defined criteria. These tests help identify issues early, allowing for prompt resolution and ensuring that subsequent iterations incorporate necessary improvements. By integrating testing into each phase, agile methodologies promote flexibility and responsiveness to changing project requirements, ultimately enhancing the overall quality and functionality of the deliverables.

The evaluation of the AFNT project highlights significant progress and challenges across its phases. Agile methodologies, including Trello for task management and agile testing templates, have facilitated structured project execution. Objectives 1, 2, 4, and 5 have achieved substantial progress, with certain features requiring further development. Each project phase encountered unique challenges, such as data acquisition in Phase 1 and design delays in Phase 3. Testing has been integral in identifying areas for improvement, particularly in incomplete tasks. Moving forward, the project aims to address outstanding requirements and optimize functionalities while leveraging agile methodologies to refine the AFNT project.

9. Conclusion

Overall, the project successfully addressed the majority of critical requirements, achieving approximately 74% of its requirements. Most project phases were completed, although Phase 2 encountered delays and remains unfinished. Additionally, key functionalities like cross-platform compatibility and BLE integration. Despite these challenges, most requirements were addressed, and the majority of test cases passed successfully, with the main issues stemming from incomplete implementations due to time limitations and other personal matters. The AFNT app was briefly used by me for real-time logging of body intake data and workouts, underscoring its practicality. However, further testing by other users would enhance its reliability and functionality before final deployment.

To sum up, the AFNT project is aimed to raise awareness of obesity and promote healthier lifestyles through the AFNT app, which unfortunately has limited usage. However, the project’s idea sparked

discussions within my social circles and family, fostering increased awareness and prompting some individuals to adopt more active and health-conscious lifestyles. Despite not meeting all initial requirements, the app still offers a robust set of features that can aid users in achieving their fitness objectives. It combines diverse fitness and body data logging functionalities and integration with the Arduino Watch. The app provides comprehensive customization of workouts and body data in a single platform—a unique proposition in the fitness app landscape. Despite encountering challenges like time constraints and external issues, I am satisfied with the achieved results. The app's development yielded significant features that can benefit users in their fitness journeys. With further refinement and additional features, there is substantial potential for the app to assist a broader audience in achieving their fitness goals effectively.

9.1. Further Work

Moving forward, the AFNT project can prioritize implementing incomplete requirements due to time constraints, such as meal tracking and plotting, gym finder functionality, and BLE synchronization. Once these foundational requirements are addressed, the project can then advance to include more sophisticated features, such as:

- Incorporate access to fitness and health advice for users, along with instructional exercise videos for technique demonstrations.
- Enhance the user interface (UI) to optimize usability across various devices.
- Improve sensor accuracy and algorithms for precise body metrics, and add cardiogram tracking and sleep monitoring.

  Add advanced features on the Arduino Watch such as touchscreen, GPS, phone call reception, and notifications.

10. References / Bibliography

1. Campbell, D. (2023) *More than half of humans on track to be overweight or obese by 2035 – report the Guardian*. 2 March 2023 [online]. Available from: https://www.theguardian.com/society/2023/mar/02/more-than-half-of-humans-on-track-to-be-overweight-or-obese-by-2035-report [Accessed 22 January 2024].

2. Wright, S. and Aronne, L.J. (2012) Causes of obesity. *PubMed* [online]. 37 (5), pp. 730–732. Available from: https://link.springer.com/article/10.1007/s00261-012-9862-x [Accessed 22 January 2024].

3. Desjardins, J. (2020) *Animation: The World's Rapid Rise in Life Expectancy, in Just 13 Seconds Visual Capitalist*. 11 May 2020 [online]. Available from: https://www.visualcapitalist.com/rapid-rise-in-life-expectancy/ [Accessed 23 January 2024].

4. Woessner, M.N., Tacey, A., Ariella Levinger-Limor, Parker, A., Levinger, P. and Levinger, I. (2021) The Evolution of Technology and Physical Inactivity: The Good, the Bad, and the Way Forward. *Frontiers in Public Health* [online]. 9. Available from: https://www.frontiersin.org/articles/10.3389/fpubh.2021.655491/full?&utm_source=Email_to_authors_&utm_medium=Email&utm_content=T1_11.5e1_author&utm_campaign=Email_publication&field=&journalName=Frontiers_in_Public_Health&id=655491#F1 [Accessed 23 January 2024].

5. Caballero B. (2007) The Global Epidemic of Obesity: An Overview. *Epidemiologic Reviews* [online]. 29 (1), pp. 1–5. Available from: https://academic.oup.com/epirev/article/29/1/1/444345?login=false [Accessed 23 January 2024].

6. Moscaritolo, A. (2024) *The Ultimate Guide to Health and Fitness Tech for 2024 PCMag UK*. 5 January 2024 [online]. Available from: https://uk.pcmag.com/gallery/130809/the-ultimate-guide-to-health-and-fitness-tech [Accessed 23 January 2024].

7. Pankush Kalgotra, Raja, U. and Sharda, R. (2022) Growth in the development of health and fitness mobile apps amid COVID-19 pandemic. *DOAJ (DOAJ: Directory of Open Access Journals)* [online]. 8, p. 205520762211290-205520762211290. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9536106/ [Accessed 23 January 2024].

8. ‘Fitness App Revenue and Usage Statistics (2024)’ (2024) *Business of Apps*. 8 January 2024 [online]. Available from: https://www.businessofapps.com/data/fitness-app-market/ [Accessed 23 January 2024].

9. ‘How to look after your mental health using exercise’ (2015) *Mental Health Foundation*. 2015 [online]. Available from: https://www.mentalhealth.org.uk/explore-mental-health/publications/how-look-after-your-mental-health-using-exercise [Accessed 17 October 2023].

10. ‘7 great reasons why exercise matters’ (2023) *Mayo Clinic*. 2023 [online]. Available from: https://www.mayoclinic.org/healthy-lifestyle/fitness/in-depth/exercise/art-20048389 [Accessed 17 October 2023].

11. Godman, H. (2014) *Regular exercise changes the brain to improve memory, thinking skills - Harvard Health Harvard Health*. 9 April 2014 [online]. Available from: https://www.health.harvard.edu/blog/regular-exercise-changes-brain-improve-memory-thinking-skills-201404097110 [Accessed 17 October 2023].

12. ‘How Can Exercise Affect Sleep? | Sleep Foundation’ (2013) *Sleep Foundation*. 25 February 2013 [online]. Available from: https://www.sleepfoundation.org/physical-activity/exercise-and-sleep [Accessed 17 October 2023].

13. ‘How to boost your immune system - Harvard Health’ (2014) *Harvard Health*. 22 September 2014 [online]. Available from: https://www.health.harvard.edu/staying-healthy/how-to-boost-your-immune-system [Accessed 17 October 2023].

14. www.facebook.com/thefitnessgrp (2023) *How Much Does a Gym Membership Cost in the UK (2023) Thefitnessgrp.co.uk*. 19 July 2023 [online]. Available from: https://www.thefitnessgrp.co.uk/how-much-does-a-gym-membership-cost/ [Accessed 25 January 2024].

15. ‘Top excuses people use for not going to the gym | Better’ (2023) *Better.org.uk*. 2023 [online]. Available from: https://www.better.org.uk/content_pages/top-gym-excuses [Accessed 25 January 2024].

16. Millington, B. (2018) *A brief history (and a look into the future) of fitness technology The Conversation*. 11 January 2018 [online]. Available from: https://theconversation.com/a-brief-history-and-a-look-into-the-future-of-fitness-technology-89884 [Accessed 23 January 2024].

17. ‘Healthcare - Apple Watch’ (2017) *Apple (United Kingdom)*. 2017 [online]. Available from: https://www.apple.com/uk/healthcare/apple-watch/ [Accessed 23 January 2024].

18. Waghchoure, E. (2023) *The Evolution of Fitness Trackers: Your Personal Health Companion Medium*. 16 September 2023 [online]. Available from: https://medium.com/@elijah.ndspl/the-evolution-of-fitness-trackers-your-personal-health-companion-5745e41ee9e2 [Accessed 23 January 2024].

19. 'Samsung Health | Apps & Services | Samsung UK' (2023) *Samsung uk*. 21 November 2023 [online]. Available from: https://www.samsung.com/uk/apps/samsung-health/ [Accessed 26 January 2024].

20. 'Calorie Tracker & BMR Calculator to Reach Your Goals | MyFitnessPal' (2024) *Myfitnesspal.com*. 2024 [online]. Available from: https://www.myfitnesspal.com/ [Accessed 26 January 2024].

21. 'Lose It! - Calorie counting made easy' (2022) *Loseit.com*. 2022 [online]. Available from: https://www.loseit.com/ [Accessed 26 January 2024].

22. 'Strava | Running, Cycling & Hiking App - Train, Track & Share' (2024) *Strava.com*. 2024 [online]. Available from: https://www.strava.com/features [Accessed 26 January 2024].

23. 'The Fitbit App' (2024) *Fitbit.com*. 2024 [online]. Available from: https://www.fitbit.com/global/uk/technology/fitbit-app [Accessed 26 January 2024].

24. Garmin International (2024) *Garmin Connect | Free Online Fitness Community Garmin Connect*. 2024 [online]. Available from: https://connect.garmin.com/ [Accessed 26 January 2024].

25. 'Nike Training Club App. Home Workouts & More.' (2023) *Nike.com*. 2023 [online]. Available from: https://www.nike.com/ntc-app [Accessed 24 January 2024].

26. (Beachbodyondemand, 2023) *Beachbodyondemand.com*. 2023 [online]. Available from: https://www.beachbodyondemand.com/?locale=en_US [Accessed 24 January 2024].

27. 'Permissions on Android' (2024) *Android Developers*. 2024 [online]. Available from: https://developer.android.com/guide/topics/permissions/overview#normal-dangerous [Accessed 24 January 2024].

28. Olmstead, K. (2015) *Apps Permissions in the Google Play Store Pew Research Center: Internet, Science & Tech*. 10 November 2015 [online]. Available from: https://www.pewresearch.org/internet/2015/11/10/apps-permissions-in-the-google-play-store/ [Accessed 24 January 2024].

29. Grundy, Q., Held, F. and Bero, L. (2017) Tracing the Potential Flow of Consumer Data: A Network Analysis of Prominent Health and Fitness Apps. *Journal of Medical Internet Research* [online]. 19 (6), pp. e233–e233. Available from: https://www.jmir.org/2017/6/e233/#ref7 [Accessed 24 January 2024].

30. Li, J. (2015) A Privacy Preservation Model for Health-Related Social Networking Sites. *Journal of Medical Internet Research* [online]. 17 (7), pp. e168–e168. Available from: https://www.jmir.org/2015/7/e168/ [Accessed 24 January 2024].

31. 'Appconner - Best Apps & Stories' (2021) *Appconner*. 2021 [online]. Available from: https://www.appconner.com/app-samsung-health?utm_source=bing&utm_medium=cpc&utm_campaign=%e6%96%b0213b_afcapp_uk_jay%209.30%200.1*&utm_term=samsung%20health&utm_content=1830_samsung%20health* [Accessed 26 January 2024].

32. 'Overview of UK GDPR' (2016) *Practical Law*. 2016 [online]. Available from: https://uk.practicallaw.thomsonreuters.com/Document/I968a9c521afe11e89bf099c0ee06c731/View/FullText.html?navigationPath=Search%2Fv1%2Fresults%2Fnavigation%2Fi0ad740160000018d3e479a10da2c4879%3Fppcid%3Dc2504c02e0aa4c198285633427e0550a%26Nav%3DKNOWHOW_UK%26fragmentIdentifier%3DI968a9c521afe11e89bf099c0ee06c731%26parentRank%3D0%26startIndex%3D1%26contextData%3D%2528sc.Search%2529%26transitionType%3DSearchItem&listSource=Search&listPageSource=4be1b00a2501e5d28b058acb57e263c2&list=KNOWHOW_UK&rank=2&sessionScopeId=09fc82fad034c2252f14fe02939c428e76892af50a3a0a55be4a0ac57ed2c67c&ppcid=c2504c02e0aa4c198285633427e0550a&originationContext=Search%20Result&transitionType=SearchItem&contextData=(sc.Search)&comp=pluk [Accessed 25 January 2024].

33. WCAG 2 (2018) *WCAG 2 Overview Web Accessibility Initiative (WAI)*. 2018 [online]. Available from: https://www.w3.org/WAI/standards-guidelines/wcag/ [Accessed 25 January 2024].

34. 'Powerful and Fast UML Diagramming Software - Astah' (2023) *Astah*. September 2023 [online]. Available from: https://astah.net/products/astah-uml/ [Accessed 25 January 2024].

35. 'Free Online Spreadsheet Software: Excel | Microsoft 365' (2016) *Microsoft.com*. 2016 [online]. Available from: https://www.microsoft.com/en-us/microsoft-365/excel?ocid=ORSEARCH_Bing [Accessed 25 January 2024].

36. 'SQLite Home Page' (2024) *Sqlite.org*. 2024 [online]. Available from: https://www.sqlite.org/index.html [Accessed 25 January 2024].

37. ‘SQL Server Downloads | Microsoft’ (2022) *Microsoft.com*. 2022 [online]. Available from: https://www.microsoft.com/en-GB/sql-server/sql-server-downloads [Accessed 25 January 2024].

38. ‘Welcome to Flask — Flask Documentation (3.0.x)’ (2024) *Palletsprojects.com*. 2024 [online]. Available from: https://flask.palletsprojects.com/en/3.0.x/ [Accessed 25 January 2024].

39. ‘Kivy: Cross-platform Python Framework for NUI’ (2024) *Kivy.org*. 2024 [online]. Available from: https://kivy.org/ [Accessed 25 January 2024].

40. ‘TinyCircuits - Maker of Tiny, Open Source Electronics’ (2024) *TinyCircuits*. 2024 [online]. Available from: https://tinycircuits.com/ [Accessed 25 January 2024].

41. ‘TinyCircuits Projects’ (2022) *TinyCircuits*. 24 January 2022 [online]. Available from: https://tinycircuits.com/blogs/learn [Accessed 25 January 2024].

42. Business, A. (2022) *Chapter 10: MoSCoW Prioritisation Agilebusiness.org*. 2022 [online]. Available from: https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html [Accessed 22 March 2024].

43. Alam, M. (2022) *7 Crucial UML Diagram Advantages IdeaScale*. 16 March 2022 [online]. Available from: https://ideascale.com/blog/uml-diagram-advantages/ [Accessed 22 March 2024].

44. Atlassian (2019) *What is kanban? Atlassian*. 2019 [online]. Available from: https://www.atlassian.com/agile/kanban#:~:text=Kanban%20is%20one%20of%20the,for%20teams%20of%20all%20sizes. [Accessed 22 March 2024].

45. Pandit, N. (2022) *Gym Exercise Dataset Kaggle.com*. 2022 [online]. Available from: https://www.kaggle.com/datasets/niharika41298/gym-exercise-data/data [Accessed 23 March 2024].

46. vinitshah0110 (2022) *Food Composition Kaggle.com*. 2022 [online]. Available from: https://www.kaggle.com/datasets/vinitshah0110/food-composition [Accessed 23 March 2024].

47. ‘Body Fat Calculator’ (2017) *Calculator.net*. 2017 [online]. Available from: https://www.calculator.net/body-fat-calculator.html [Accessed 23 March 2024].

48. ‘BMI Calculator’ (2024) *Calculator.net*. 2024 [online]. Available from: https://www.calculator.net/bmi-calculator.html [Accessed 23 March 2024].

49. ‘TinyDuino Overview - TinyCircuits Docs’ (2024) *Tinycircuits.com*. 2024 [online]. Available from: https://learn.tinycircuits.com/TinyDuino_Overview/ [Accessed 20 April 2024].

50. ‘OLED Shield | TinyScreen | TinyCircuits.com’ (2024) *TinyCircuits*. 2024 [online]. Available from: https://tinycircuits.com/products/tinyscreen?variant=13748964423 [Accessed 18 April 2024].

51. ‘Lithium Ion Polymer Battery - 3.7V 290mAh | TinyCircuits.com’ (2024) *TinyCircuits*. 2024 [online]. Available from: https://tinycircuits.com/products/lithium-ion-polymer-battery-3-7v-290mah?_pos=4&_sid=64d75c1d1&_ss=r [Accessed 18 April 2024].

52. ‘MicroSD Shield | TinyShield | TinyCircuits.com’ (2024) *TinyCircuits*. 2024 [online]. Available from: https://tinycircuits.com/products/microsd-tinyshield?_pos=3&_sid=33c073989&_ss=r [Accessed 18 April 2024].

53. ‘TinyCircuits | Wireling Adapter’ (2022) *TinyCircuits*. 2022 [online]. Available from: https://tinycircuits.com/products/wireling-adapter-tinyshield?_pos=14&_sid=924b00554&_ss=r [Accessed 18 April 2024].

54. ‘TinyCircuits | Accelerometer Shield’ (2024) *TinyCircuits*. 2024 [online]. Available from: https://tinycircuits.com/products/accelerometer-tinyshield?_pos=3&_sid=5fb80dca9&_ss=r [Accessed 18 April 2024].

55. ‘TinyCircuits | Accelerometer Sensor’ (2024) *TinyCircuits*. 2024 [online]. Available from: https://tinycircuits.com/products/accelerometer-wireling-bma250?_pos=2&_sid=97eec2ab4&_ss=r [Accessed 18 April 2024].

56. ‘TinyCircuits | Pulse Oximeter Sensor’ (2024) *TinyCircuits*. 2024 [online]. Available from: https://tinycircuits.com/products/pulse-oximetry-sensor-wireling [Accessed 18 April 2024].

57. ‘TinyCircuits | Bluetooth Low Energy Shield (ST)’ (2024) *TinyCircuits*. 2024 [online]. Available from: https://tinycircuits.com/products/bluetooth-low-energy-tinyshield [Accessed 18 April 2024].

58. Douglas, J. (2023) *Introduction to Python Visual Programming with kvlang Medium*. 25 February 2023 [online]. Available from: https://medium.com/@joshua.d1434/introduction-to-python-visual-programming-with-kvlang-a0960bdfd776 [Accessed 20 April 2024].

59. ‘KivyMD 2.0.1.dev0 documentation’ (2022) *Readthedocs.io*. 2022 [online]. Available from: https://kivymd.readthedocs.io/en/latest/index.html [Accessed 19 April 2024].

60. Volkwyn, T.S., Bor Gregorcic, Airey, J. and Linder, C. (2020) Learning to use Cartesian coordinate systems to solve physics problems: the case of ‘movability’. *European journal of physics* [online]. 41 (4), pp. 045701–045701. Available from: https://iopscience.iop.org/article/10.1088/1361-6404/ab8b54/meta [Accessed 20 April 2024].

61. Jorge Stolfi (2018) *Cartesian coordinate system Wikipedia*. 16 April 2024 [online]. Available from: https://en.wikipedia.org/wiki/Cartesian_coordinate_system#/media/File:Coord_system_CA_0.svg [Accessed 20 April 2024].

62. GeeksforGeeks (2022) *Magnitude of a Vector GeeksforGeeks*. 30 January 2022 [online]. Available from: https://www.geeksforgeeks.org/magnitude-of-a-vector/ [Accessed 20 April 2024].

63. ‘Test Case Template’ (2023) *ProjectManager*. 8 November 2023 [online]. Available from: https://www.projectmanager.com/templates/test-case-template [Accessed 21 April 2024].

Appendix A: Requirements

1. Project Aim

To design and implement a software solution focused on helping users achieve their fitness objectives. This involves developing a user-friendly platform for creating and customising workout routines, monitoring nutrition and workout progress, and offering health and fitness guidance. The main aim is to build an engaging and efficient fitness tool that encourages users to live healthier, more active lives.

2. Project Objectives

Please note all requirements will be marked by MoSCow Prioritization technique.

ID	Objective	Description	Created
O1	Develop a Scalable and Secure Database Management System (DBMS)	Develop a scalable MySQL DBMS, including a central database for 2,000+ records and a local database for user data and watch data. Improve database performance and security and ensure GDPR compliance.	09/10/2023
O2	Build the Alistana Fitness & Nutrition Tracker (AFNT) Application	Develop the AFNT application with features for workout and nutrition tracking, body progress and measurement tracking. Should be connected to DBMS, Admin Management website (only for Admins), and Arduino watch.	09/10/2023
O3	Design a User-Centric Admin Management (AM) Website	Develop a responsive website with secure login and allows Admins to edit the central database (user login data and preset workout and meal data) and app push updates to the AFNT application. The website will prioritize a user-friendly design, encrypted communication, and security measures.	09/10/2023
O4	Design and develop a Fitness Watch using Arduino	Create an Arduino-based Fitness watch to measure blood oxygen level, heart rate and step count, and investigate ways of connecting the Arduino watch to the AFNT and store body data in the local database via AFNT.	16/10/2023
O5	Enhance Code Quality and Performance	Implement clean, maintainable code with 80% code coverage. Optimize application and website response times to under 2 and 3 seconds, respectively.	25/10/2023

3. Functional Requirements

The functional requirements are divided into 4 categories: Database, Website, Application and Arduino Watch. These categories may be divided into further sub-categories to make them more readable, making it easier to track progress.

3.1. Database

Database Management System (DBMS)				
ID	**Summary**	**Priority**	**Status**	**Created**
Central Database (CDB)				
FD1	The central database should be stored in a MySQL Server.	M	Complete	25/10/2023
FD2	Should store preset meals, food items, exercises, and workouts with attributes such as meal_id, food_item_id, exercise_id, and workout_id.	M	Complete	11/10/2023
FD3	Store user information in CDB with attributes such as user, email, password hash, type (User or Admin), gender, phone, address, and date created.	M	Complete	25/10/2023
FD4	Support queries to retrieve preset meals, food items, exercises, and workouts based on various filters such as meal type, food category, exercise category, and workout difficulty.	M	Complete	25/10/2023
FD5	Provide data synchronization capabilities to update the local database with the latest preset data by comparing the timestamps of the local and central databases.	M	Partially Complete	25/10/2023
Local Database (LDB)				
FD6	The local database should be stored in the user drive using SQLite.	M	Complete	25/10/2023
FD7	Store both preset and custom (combined as one) in tables: meals, meal_logs food_items, food_item_logs. exercises, exercise_logs workouts, workout_logs, heart rate, blood oxygen level, and step count.	M	Complete	16/10/2023
FD8	Store user information in LDB with attributes such as user, email, password hash, type (User or Admin), gender, phone, address, and date created.	M	Complete	16/10/2023
FD9	Support updates to custom meals, workouts, exercises, and health metrics by allowing users to add, modify, and delete records.	M	Complete	09/10/2023
FD10	Store health metrics from the Arduino watch including step count, heart rate and blood oxygen level.	M	Complete	17/10/2023

FD11	Store other metrics such as height (meters), weight (kg), BMI, skeletal muscle (kg), body fat (%) and water intake (ml),	M	Complete	26/10/2023

3.2. Website

Admin Management Website (AM)				
ID	**Summary**	**Priority**	**Status**	**Created**
FW1	The AFNT App shall link to the AM website for Admin login.	S	Not Started	09/10/2023
FW2	The website should only be accessible to Admins and can be accessed Online or via AFNT App.	S	Partially Complete	09/10/2023
FW3	The website shall allow Admin users to log in securely with their credentials.	S	Complete	09/10/2023
FW6	The website shall allow admins to modify user personal details (i.e., profile picture, name, age, gender, dob, email, password, phone, address, and postcode.	C	Not Implementing	09/10/2023
FW7	The website shall allow admins to edit their own profiles (profile picture, name, age, gender, dob, email, password, phone, address, and postcode).	C	Not Implementing	09/10/2023
FW8	The website should allow admins to add/modify/delete preset workout data	M	Not Implementing	09/10/2023
FW9	The website should allow admins to add/modify/delete preset exercise data	M	Not Implementing	09/10/2023
FW10	The website should allow admins to add/modify/delete preset meal data	M	Not Implementing	09/10/2023
FW11	The website should allow admins to add/modify/delete preset food item data	M	Not Implementing	09/10/2023
FW12	The website shall allow admins to push database updates to the AFNT application.	M	Not Implementing	10/10/2023

3.3. Application

Alistana Fitness & Nutrition Tracker Application (AFNT)				
ID	**Summary**	**Priority**	**Status**	**Created**
Login and Registration				
FA41	The application shall allow users to login using their username and password credentials.	M	Complete	17/10/2023
FA42	The application shall allow users to register, and they must provide details such as username, email, password, phone number, email address, address, date of birth and gender.	M	Complete	17/10/2023
User Profile				
FA43	The application shall allow users to modify their personal data such as profile picture, username, email, password, phone number, email address, address, date of birth and gender.	M	Partially Complete	17/10/2023
FA44	The application shall allow users to delete their personal data.	M	Partially Complete	17/10/2023
FA45	The application shall allow users to delete their account and all the data associated with it.	M	Partially Complete	17/10/2023
Workout Tracker				
FA1	The application shall allow users to create their own workout plans.	M	Complete	09/10/2023
FA2	The application shall allow users to customize their own workout plans.	M	Complete	09/10/2023
FA3	The application shall allow users to choose pre-designed workout plans based on their goals.	M	Complete	09/10/2023
FA4	The application shall allow users to receive suggested workouts based on fitness goals.	S	Partially Complete	09/10/2023
FA5	The application shall allow users to track their workouts by recording exercises, sets, reps, and weights (For custom exercises).	M	Complete	09/10/2023
FA6	The application shall allow users to rate their performance for each exercise (For both preset and custom).	M	Complete	09/10/2023
FA7	The application shall allow users to rate their performance for each workout (For both preset and custom).	M	Complete	09/10/2023

FA8	The application shall allow users to view their workout logs (For both preset and custom).	M	Complete	09/10/2023
FA9	The application shall allow users to view their exercise logs (For both preset and custom).	M	Complete	09/10/2023
FA10	The application shall allow users to display graphs of each exercise progress (i.e., Bench press max weight every month).	M	Partially Complete	09/10/2023
FA11	The application shall allow users to display graphs of cardio-related exercises (i.e., Step/Distance for treadmill session) by manually inputting the information.	M	Complete	23/10/2023
FA12	The application shall allow users to display graphs of cardio-related exercises (i.e., Step/Distance for treadmill session) provided by the Arduino watch.	M	Complete	23/10/2023
FA13	The application shall produce charts for weight in kilograms (per day).	M	Complete	26/10/2023
FA14	The application shall allow users to set goals for workouts.	C	Not Started	24/10/2023
Nutrition Tracker				
FA15	The application shall allow users to select preset food items.	M	Complete	09/10/2023
FA16	The application shall allow users to select preset meals.	M	Partially Complete	09/10/2023
FA17	The application shall allow users to create custom food items and define their nutritional contents (i.e., calories, protein, carbs, fats etc.)	M	Partially Complete	09/10/2023
FA18	The application shall allow users to create custom meals and add food items (Both custom and preset) to the custom meal.	M	Partially Complete	26/10/2023
FA19	The application shall allow users to modify custom food items.	M	Partially Complete	09/10/2023
FA20	The application shall allow users to modify custom meals.	M	Partially Complete	09/10/2023
FA21	The application shall allow users to input daily water intake in millilitres (per day).	M	Complete	09/10/2023
FA22	The application shall allow users to generate a daily calorie intake graph based on a selected date range.	M	Not Started	24/10/2023
FA23	The application shall allow users to generate a daily nutritional content intake (i.e., Carbs, fat, protein intake per	M	Not Started	24/10/2023

	day) graph based on a selected date range.			
FA24	The application shall allow users to generate a daily water intake graph based on a selected date range.	M	Complete	24/10/2023
FA25	The application shall allow users to set goals for nutritional intake.	C	Not Started	25/10/2023
FA26	The application shall allow users to set goals for their water intake	C	Partially Complete	25/10/2023
Body Measurements Tracker				
FA27	The application shall allow users to input their daily weight in kilograms (per day).	M	Complete	26/10/2023
FA28	The application shall allow users to input their daily height in meters (per day).	M	Complete	26/10/2023
FA29	The application shall allow users to input Skeletal muscle data in kilograms (per day).	M	Complete	26/10/2023
FA30	The application shall allow users to input Body fat data in percentage (per day).	M	Complete	26/10/2023
FA31	The application shall use weight and height data and calculate user's BMI (per day).	M	Complete	26/10/2023
FA32	The application shall produce charts for weight in kilograms (per day).	M	Complete	26/10/2023
FA33	The application shall produce graphs for height change.	M	Complete	26/10/2023
FA34	The application shall produce charts for BMI change.	M	Complete	26/10/2023
FA35	The application shall allow users to set goals for step counts/distance covered per day.	M	Complete	26/10/2023
FA36	The application shall allow users to generate a heart rate graph provided by the Arduino watch.	M	Complete	26/10/2023
FA37	The application shall allow users to generate a blood oxygen chart provided by the Arduino watch.	M	Complete	26/10/2023
Gym Locator				
FA38	The application shall allow users to get a list of the nearest gyms based on a mapping API.	S	Partially Complete	24/10/2023
Health & Nutrition Advice				
FA39	The application shall allow users to access health-related advice.	W	Not Implementing	09/10/2023
FA40	The application shall allow users to access fitness-related advice (i.e., correct exercise forms etc.).	W	Not Implementing	09/10/2023

3.4. Arduino Watch

Arduino Fitness Watch				
ID	**Summary**	**Priority**	**Status**	**Created**
FAW9	The watch shall display the current date and time.	M	Complete	16/10/2023
FAW1	The watch shall have an oximeter module attached to measure heart rate and blood oxygen levels in real time.	M	Complete	16/10/2023
FAW2	The watch shall have an accelerometer module attached to measure steps in real-time.	M	Complete	16/10/2023
FAW3	The watch shall also use the accelerometer to count reps for the bench press in real-time.	M	Partially Complete	18/10/2023
FAW4	The watch shall have a microSD storage unit to store body measurements in real-time every 5 seconds.	M	Complete	18/10/2023
FAW5	The watch shall compile, and store measured data in a CSV file type.	M	Complete	18/10/2023
FAW6	The watch shall use a Bluetooth (low energy) module to transfer data to the application.	M	Partially Complete	18/10/2023
FAW7	The watch shall also use a wired connection via a micro-USB port to transfer watch data to the application.	M	Partially Complete	18/10/2023
FAW8	The application shall process and store watch data in the local database.	M	Partially Complete	18/10/2023

4. Non-Functional Requirements

The non-functional requirements are divided into 9 categories: Application, Performance, Efficiency & Sustainability, Data Storage Optimization, Privacy & Security, Reliability, Usability, Data Backup & Recovery, and Third-Party Service Integration. These categories may be divided into further sub-categories to make them more readable, making it easier to track progress.

4.1. Application:

Alistana Fitness & Nutrition Tracker Application (AFNT)				
ID	**Summary**	**Priority**	**Status**	**Created**
Workout Tracker				
NFA1	Users can add an unlimited number of custom exercises	M	Complete	25/10/2023
NFA2	Users can add an unlimited number of custom workouts	M	Complete	25/10/2023
NFA3	Users cannot modify preset workouts and exercises.	M	Complete	25/10/2023
NFA4	Users can allocate three workouts per day.	M	Complete	25/10/2023
NFA5	Users can choose up to 20 exercises (incl. custom exercises) per workout.	M	Complete	25/10/2023
NFA6	Users cannot modify preset workouts and exercises.	M	Complete	25/10/2023
Nutrition Tracker				
NFA8	Users can add an unlimited number of custom food items	M	Partially Complete	25/10/2023
NFA9	Users can add an unlimited number of custom meals	M	Partially Complete	25/10/2023
NFA10	Users cannot modify preset food items and meals.	M	Partially Complete	25/10/2023
NFA11	Users can modify meal logs and food item serving size in that meal (Can be both custom or preset meal and food item.)	M	Partially Complete	26/10/2023
NFA12	Users can only select up to 4 meals (Morning, Afternoon, Evening, and Dinner) per day.	M	Not Started	25/10/2023
NFA13	Users can choose up to 8 food items (incl. custom food items) per meal.	M	Not Started	25/10/2023
NFA14	Users cannot modify the nutrition content of preset meals and food items.	M	Partially Complete	25/10/2023

4.2. Performance:

Performance				
ID	**Summary**	**Priority**	**Status**	**Created**
NFP1	The AM website should load in under 5 seconds, ensuring optimal user experience	S	Complete	09/10/2023
NFP2	The AM website shall respond to user input within 200 milliseconds, providing a smooth and responsive interaction.	S	Partially Complete	09/10/2023
NFP3	The application shall load within a 5-second time frame, ensuring users can access the features promptly.	S	Complete	09/10/2023
NFP4	The Arduino watch should display heart rate, blood oxygen level, steps count in real time on the watch display.	M	Complete	17/10/2023

4.3. Efficiency & Sustainability:

Efficiency & Sustainability				
ID	**Summary**	**Priority**	**Status**	**Created**
NFES1	The AM website should have a memory usage of no more than 500 MB and a CPU usage of no more than 10% under normal operating conditions.	C	Complete	09/10/2023
NFES2	The application should have a memory usage of no more than 250 MB and a CPU usage of no more than 5% under normal operating conditions.	C	Complete	25/10/2023
NFES3	The watch should collect and store data with a latency of no more than 100 milliseconds and a storage usage of no more than 50 MB.	C	Complete	24/10/2023
NFES4	The watch should transfer data to the application with a latency of no more than 200 milliseconds and a data transfer rate of at least 1 Mbps.	C	Partially Complete	24/10/2023
NFES5	Ensure database consistency and integrity by implementing ACID (Atomicity, Consistency, Isolation, Durability) properties in database transactions.	C	Complete	25/10/2023

4.5. Data Storage Optimization:

Data Storage & Optimization				
ID	**Summary**	**Priority**	**Status**	**Created**
NFDS1	Implement efficient data storage techniques such as indexing, partitioning, and normalization to ensure that the average database query time is no more than 200 milliseconds for read queries and no more than 500 milliseconds for write queries.	S	Complete	09/10/2023

4.6. Privacy & Security:

Privacy & Security				
ID	**Summary**	**Priority**	**Status**	**Created**
NFPS1	The system shall implement features such as data minimization, purpose limitation, data portability, and the right to be forgotten, to comply with GDPR guidelines.	C	Partially Complete	09/10/2023
NFPS2	The system shall implement role-based access control, secure authentication and authorization mechanisms, and regular security audits to ensure data privacy and security for user personal and health data.	W	Partially Complete	09/10/2023
NFPS3	User data, both at rest and in transit, shall be encrypted using industry-standard encryption algorithms (e.g., AES-256) to ensure data is stored and transmitted safely.	C	Partially Complete	24/10/2023
NFPS4	The Bluetooth or wired data transfer from the Arduino watch to the app shall use secure protocols (e.g., TLS/SSL) and industry-standard encryption algorithms (e.g., AES-256) to ensure the data is transferred securely and encrypted.	S	Not Started	24/10/2023

4.7. Reliability:

Reliability				
ID	**Summary**	**Priority**	**Status**	**Created**
NFR1	The AM website will have an uptime of at least 99.9% by implementing robust error handling and failover mechanisms.	S	Partially Complete	10/10/2023
NFR2	The application shall implement robust error handling to ensure that any errors are gracefully handled, and logged for analysis and that the application recovers without crashing.	M	Partially Complete	24/10/2023

NFR3	The database and server shall implement robust error handling to automatically recover from failures and backup mechanisms to ensure data is backed up at least once a day and can be restored within 24 hours in case of data loss.	M	Partially Complete	09/10/2023
NFR4	The Arduino hardware shall be constructed with materials that meet industry standards for wearables.	C	Partially Complete	16/10/2023
NFR5	The smartwatch shall have a power management system that optimizes battery usage to ensure at least 12 hours of continuous operation on a single charge under normal usage conditions.	C	Partially Complete	16/10/2023

4.8. Usability:

Usability				
ID	**Summary**	**Priority**	**Status**	**Created**
NFU1	The AM website shall follow WCAG 2.1 AA compliance, as verified by automated testing tools.	C	Partially Complete	09/10/2023
NFU2	The application shall have a user-friendly interface with a System Usability Scale (SUS) score of at least 70, indicating good usability.	C	Partially Complete	24/10/2023
NFU3	The AM website UI shall have a navigation menu that is clearly visible and accessible from all pages, and a user flow that requires no more than three clicks to reach any page.	S	Partially Complete	25/10/2023
NFU4	The website and application shall implement features such as screen reader compatibility, keyboard navigation, and text alternatives for non-text content to enhance accessibility for users with disabilities.	W	Not Implementing	25/10/2023
NFU5	The application shall synchronize health data from the smartwatch to the app by pressing a sync button, ensuring that the data is up-to-date and accurate.	M	Not Started	16/10/2023
NFU6	The application shall display clear and informative error messages in case of data transfer issues from the watch and provide a troubleshooting guide in the help section of the app.	S	Not Started	16/10/2023
NFU7	The application should be compatible with mobile platforms, providing seamless functionality and usability across various mobile devices. The responsiveness and compatibility standards should adhere to modern design principles, facilitating	S	Partially Complete	16/10/2023

	accessibility and usability for users on different devices.			

4.9. Data Backup & Recovery:

Data Backup & Recovery				
ID	**Summary**	**Priority**	**Status**	**Created**
NFDBR1	The system shall have an automated backup and recovery system that creates daily backups of the database and stores them in a secure location, with the ability to restore data within 24 hours in case of data loss.	M	Complete	09/10/2023
NFDBR2	The program's source code shall be backed up regularly to GitHub, with automated daily backups to OneDrive for additional redundancy. The backup system should support version control, allowing for the recovery of specific versions of the code if needed.	M	Complete	09/10/2023

4.10. Third-Party Service Integration:

Third-Party Service Integration				
ID	**Summary**	**Priority**	**Status**	**Created**
NFTPS1	The system shall integrate with mapping APIs to provide location-based services such as gym locator. The integration should be seamless, with real-time data synchronization and minimal latency.	C	Complete	09/10/2023

Appendix B: AFNT Trello Dump

Card	Card Name	Due Date	Due Complete
TO-DO	Food Item Log	21/04/2024	FALSE
	Meal Create Screen	21/04/2024	FALSE
	Food Item Create Screen	21/04/2024	FALSE
	Food Item Allocate Screen	21/04/2024	FALSE
IN-PROGRESS	[#2] AMW Main Page	01/03/2024	FALSE
	Gym Finder Screen	15/03/2024	FALSE
	Arduino Bluetooth Data Transfer	20/04/2024	FALSE
	Meal Allocate Screen	21/04/2024	FALSE
	Final Report	24/04/2024	FALSE
SPRINT 4 PHASE 4 IMPLEMENTATION	Arduino Watch	12/04/2024	TRUE
	Meal History Screen	13/04/2024	TRUE
	Arduino Screen	20/04/2024	TRUE
SPRINT 3 PHASE 3 IMPLEMENTATION	[#2] AFNT Dashboard	01/02/2024	TRUE
	[#23] Water Intake Screen	01/02/2024	TRUE
	[#26] Water Intake Plots Screen	01/02/2024	TRUE
	[#24] Body Statistics Screen	05/02/2024	TRUE
	[#25] Body Statistics Plots Screen	05/02/2024	TRUE
	[#4, #8, #12, #13] Workout History Screen	19/02/2024	TRUE
	Fix GitHub problem	20/02/2024	TRUE
	[#5, #11, #16] Exercise Log Screen	21/02/2024	TRUE
	[#8, #10, #11, #12] Workout Allocate Screen	29/02/2024	TRUE
	[#10] Workout Create Screen	02/03/2024	TRUE
	[#14] Exercise Create Screen	02/03/2024	TRUE
	[#11, #16] Exercise Allocate Screen	07/03/2024	TRUE
	[#11] Exercise Log Allocate Screen	08/03/2024	TRUE
	Report Draft Submission to Supervisor	22/03/2024	TRUE
SPRINT 2 PHASE 1 IMPLEMENTATION & PIP	[#1] AMW Login Screen	03/01/2024	TRUE
	Implement Central DB	10/01/2024	TRUE
	Implement LocalDB	10/01/2024	TRUE
	[#1] AFNT Login Screen	16/01/2024	TRUE
	[#1] AFNT Registration Screen	18/01/2024	TRUE
	Create Poster	21/01/2024	TRUE
	Complete Video Demo	21/01/2024	TRUE
	Assemble Arduino Watch Parts	22/01/2024	TRUE
	Test second batch of Arduino Watch	22/01/2024	TRUE
	Finish Literature Review	24/01/2024	TRUE
	PIP DEMO	24/01/2024	TRUE
SPRINT 1 PLANNING & DESIGN	Write down Requirements	09/10/2023	TRUE
	Complete Project Proposal	25/10/2023	TRUE
	Test First Batch of Arduino Watch Components	04/11/2023	TRUE
	Pre-Ethical Checklist	06/11/2023	TRUE
	Ethical Checklist	06/11/2023	TRUE

	Research Kivy GUI	02/12/2023	TRUE
	Design AFNT Logo	14/12/2023	TRUE
	Create Usecase diagrams for Admin Management Website	15/12/2023	TRUE
	Create Usecase diagrams for AFNT App	15/12/2023	TRUE
	Create Usecase diagrams for Arduino Watch	15/12/2023	TRUE
	Create LocalDB Schema	20/12/2023	TRUE
	Create CentralDB Schema	20/12/2023	TRUE
	Create AFNT Architecture Diagram	21/12/2023	TRUE
	Setup Login Server	25/12/2023	TRUE
	Setup Admin Login Server	25/12/2023	TRUE
	Setup Account Server	25/12/2023	TRUE
	Create Class diagram for AFNT App	26/12/2023	TRUE
	Create Class diagram for Central and Local DB	26/12/2023	TRUE
	Create AFNT UI Sketches	16/01/2024	TRUE
	Create AM Website UI Sketches	16/01/2024	TRUE
	Add Usecase descriptions	15/01/2024	TRUE
SUPERVISOR-MEETINGS	09/10/2023 - Martin Serpell	09/10/2023	TRUE
	16/10/2023 - Eman Qaddoumi	16/10/2023	TRUE
	16/10/2023 - Martin Serpell	16/10/2023	TRUE
	19/10/2023 - Eman Qaddoumi	19/10/2023	TRUE
	23/10/2023 - Martin Serpell	23/10/2023	TRUE
	30/10/2023 - Martin Serpell	30/10/2023	TRUE
	06/11/2023 - Eman Qaddoumi	06/11/2023	TRUE
	21/12/2023 - Martin Serpell	21/12/2023	TRUE
	12/01/2023 - Eman Qaddoumi	12/01/2023	TRUE
	25/01/2024 - Eman Qaddoumi	25/01/2024	TRUE
	09/02/2024 - Eman Qaddoumi	09/02/2024	TRUE
	23/02/2024 - Eman Qaddoumi	23/02/2024	TRUE
	27/03/2024 - Eman Qaddoumi	27/03/2024	TRUE
	19/04/2024 - Eman Qaddoumi	19/04/2024	TRUE
	23/04/2024 - Eman Qaddoumi	23/04/2024	TRUE

Appendix C: Creating Central Database

```
CREATE TABLE `centraldb`.`exercises` (
  `exercise_id` INT NOT NULL AUTO_INCREMENT,
  `exercise_name` VARCHAR(150) NOT NULL,
  `description` VARCHAR(300) NULL,
  `type` ENUM('Strength', 'Cardio', 'Olympic Weightlifting', 'Plyometrics', 'Powerlifting
  `body_part` ENUM('Abdominals', 'Adductors', 'Chest', 'Hamstrings', 'Quadriceps', 'L
  `equiptment` ENUM('Body Only', 'Dumbbell', 'Exercise Ball', 'Kettlebells', 'Medicine
  `level` ENUM('Beginner', 'Intermediate', 'Advanced') NOT NULL,
  `rating` DOUBLE NULL,
  `rating_description` VARCHAR(45) NULL,
  `is_active` TINYINT NOT NULL,
  PRIMARY KEY (`exercise_id`),
  UNIQUE INDEX `exercise_id_UNIQUE` (`exercise_id` ASC) VISIBLE);
```

Figure 119: Create exercises table (CDB).

```
CREATE TABLE `centraldb`.`meals` (
    `meal_id` INT PRIMARY KEY UNIQUE AUTO_INCREMENT,
    `type` ENUM('Breakfast', 'Morning Snack', 'Lunch', 'Afternoon Snack',
          'Brunch', 'Dinner', 'Evening Snack') NOT NULL,
    `description` VARCHAR(300),
    `water_tot_g` INT,
    `energy_tot_kcal` INT,
    `protein_tot_g` DOUBLE,
    `lipid_tot_g` DOUBLE,
    `carbs_tot_g` DOUBLE,
    `fiber_tot_g` DOUBLE,
    `sugar_tot_g` DOUBLE,
    `calcium_tot_mg` DOUBLE,
    `iron_tot_mg` DOUBLE,
    `cholestrl_tot_mg` DOUBLE,
    `serving` DOUBLE NOT NULL,
    `date_created` DATE NOT NULL,
    `time_created` TIME NOT NULL,
    `is_active` TINYINT(1) NOT NULL
);
```

Figure 120: Create meals table (CDB).

```
CREATE TABLE `centraldb`.`users` (
  `user_id` INT PRIMARY KEY UNIQUE AUTO_INCREMENT,
  `username` VARCHAR(50) UNIQUE NOT NULL,
  `password` VARCHAR(50) NOT NULL,
  `type` ENUM('Admin', 'User') NOT NULL,
  `email` VARCHAR(150) NOT NULL,
  `country_code` INT,
  `phone` VARCHAR(20) NOT NULL,
  `address` VARCHAR(255),
  `postcode_zipcode` VARCHAR(20),
  `gender` ENUM('Male', 'Female') NOT NULL,
  `dob` DATE NOT NULL,
  `date_enrolled` DATE NOT NULL,
  `time_enrolled` TIME NOT NULL,
  `is_active` BOOLEAN NOT NULL
);
```

Figure 121: Create users table (CDB).

```
CREATE TABLE `centraldb`.`workouts` (
  `workout_id` INT UNIQUE PRIMARY KEY,
  `workout_name` VARCHAR(150) UNIQUE NOT NULL,
  `description` VARCHAR(255),
  `type` ENUM('Abdominals', 'Adductors', 'Chest', 'Hamstrings',
    'Quadriceps', 'Lats', 'Shoulders', 'Triceps') NOT NULL,
  `date_created` DATE NOT NULL,
  `level` ENUM('Beginner', 'Intermediate', 'Advanced') NOT NULL,
  `rating` INT,
  `rating_description` VARCHAR(100),
  `time_created` TIME NOT NULL,
  `is_active` BOOLEAN NOT NULL
);
```

Figure 122: Create workouts table (CDB).

```
CREATE TABLE `centraldb`.`food_items` (
  `food_item_id` INT UNIQUE PRIMARY KEY,
  `food_item_name` VARCHAR(255) NOT NULL,
  `water_g` DECIMAL(5, 2),
  `energy_kcal` INT NOT NULL,
  `protein_g` DECIMAL(5, 2),
  `lipid_g` DECIMAL(5, 2),
  `carbs_g` DECIMAL(5, 2),
  `fiber_td_g` DECIMAL(5, 2),
  `sugar_g` DECIMAL(5, 2),
  `calcium_mg` DECIMAL(5, 2),
  `iron_mg` DECIMAL(5, 2),
  `cholestrl_mg` DECIMAL(5, 2),
  `gmwt_1` DECIMAL(5, 2),
  `gmwt_desc1` VARCHAR(255),
  `gmwt_2` DECIMAL(5, 2),
  `gmwt_desc2` VARCHAR(255),
  `is_active` BOOLEAN NOT NULL
);
```

Figure 123: Create food items table (CDB).

```
CREATE TABLE `centraldb`.`workout_logs` (
  `workout_log_id` INT PRIMARY KEY AUTO_INCREMENT,
  `workout_id` INT NOT NULL,
  `date_assigned` DATE NOT NULL,
  `time_assigned` TIME NOT NULL,
  `is_complete` BOOLEAN NOT NULL,
  `date_completed` DATE,
  `time_completed` TIME,
  `is_active` TINYINT(1) NOT NULL,
  FOREIGN KEY (`workout_id`) REFERENCES `centraldb`.`workouts`(`workout_id`)
);
```

Figure 124: Create workout logs table (CDB).

```
CREATE TABLE `centraldb`.`meal_logs` (
  `meal_log_id` INT PRIMARY KEY UNIQUE AUTO_INCREMENT,
  `meal_id` INT NOT NULL,
  `ate` BOOLEAN NOT NULL,
  `date_ate` DATE NOT NULL,
  `time_ate` TIME NOT NULL,
  `is_active` BOOLEAN NOT NULL,
  FOREIGN KEY (`meal_id`) REFERENCES `centraldb`.`meals`(`meal_id`)
);
```

Figure 125: Create meal logs table (CDB).

```
CREATE TABLE `centraldb`.`food_item_logs` (
  `food_item_log_id` INT PRIMARY KEY UNIQUE AUTO_INCREMENT,
  `meal_log_id` INT NOT NULL,
  `food_item_id` INT NOT NULL,
  `serving` DOUBLE NOT NULL,
  `weight_g` DOUBLE,
  `ate` BOOLEAN NOT NULL,
  `date_ate` DATE,
  `time_ate` TIME,
  `description` VARCHAR(300),
  `is_active` BOOLEAN NOT NULL,
  FOREIGN KEY (`meal_log_id`) REFERENCES `centraldb`.`meal_logs`(`meal_log_id`),
  FOREIGN KEY (`food_item_id`) REFERENCES `centraldb`.`food_items`(`food_item_id`)
);
```

Figure 126: Create food item logs table (CDB).

```
CREATE TABLE `centraldb`.`exercise_logs` (
  `exercise_log_id` INT PRIMARY KEY UNIQUE AUTO_INCREMENT,
  `exercise_id` INT NOT NULL,
  `workout_log_id` INT NOT NULL,
  `sets` INT,
  `reps` INT,
  `weight_kg` DOUBLE,
  `rest_per_set_s` INT,
  `duration` TIME,
  `distance_m` INT,
  `rpe` INT,
  `is_complete` BOOLEAN NOT NULL,
  `date_complete` DATE,
  `time_complete` TIME,
  `details` VARCHAR(300),
  `is_active` BOOLEAN NOT NULL,
  FOREIGN KEY (`exercise_id`) REFERENCES `centraldb`.`exercises`(`exercise_id`),
  FOREIGN KEY (`workout_log_id`) REFERENCES `centraldb`.`workout_logs`(`workout_log_id`)
);
```

Figure 127: Create exercise logs table (CDB).

```
CREATE TABLE `centraldb`.`advices` (
    `advice_id` INT PRIMARY KEY AUTO_INCREMENT,
    `name` VARCHAR(100) UNIQUE NOT NULL,
    `type_of_advice` ENUM('Health', 'Fitness', 'Nutrition') NOT NULL,
    `description` TEXT,
    `solution` TEXT,
    `reference_link` VARCHAR(255),
    `image_link` VARCHAR(255),
    `date_added` DATE,
    `time_added` TIME,
    `is_active` TINYINT(1) NOT NULL
);
```

Figure 128: Create advices table (CDB).