Name: Xingchen Chen
Matriculation number: S2269664
MEng Honours Project Phase 2 Report
<IoT for Health: An Integrated System for Monitoring Physiological Parameters and Activity Recognition>
16 January 2024

Mission Statement

Project Title: Smartphone interface for peripheral sensor

Project Definition

The mission of this project is to develop an advanced IoT system that integrates IMU sensors for accurate human activity recognition, along with blood pressure and SpO2 sensors. The primary objective is to create a comprehensive solution that provides a holistic view of an individual's physical state, enabling the early detection of potential health issues. By combining multiple sensors, this IoT system aims to deliver real-time monitoring and analysis, facilitating proactive healthcare management and timely intervention.

Preparatory Tasks

1. Collect information about sensors, Arduino, Android, Bluetooth, and machine learning concepts.
2. Install the necessary software and tools to set up the development environment.

Main Tasks

1. Develop the IoT system using the MAX30102 sensor, Arduino, and the Nordic Thingy 52, with a focus on the hardware component.
2. Collect and analyze data from the IMU sensors, blood pressure, and SpO2 sensors to obtain accurate measurements and insights.
3. Build the Android application that will serve as the interface for the IoT system. Integrate the functionalities of the sensors and ensure communication between the smartphone and external devices.
4. Test and debug the system to ensure its functionality and reliability.
5. Create comprehensive documentation and user manuals that provide clear instructions and guidelines for users to understand and effectively utilize the IoT system.

Scope for Extension

This project has potential for future expansion and enhancement, which may include:
1. Addition of supplementary sensors to gather additional health-related data.
2. Integration with other devices or platforms to enhance interoperability and data-sharing capabilities.

Background Knowledge

Successful completion of this project requires proficiency and experience in: IoT and embedded systems, sensor utilization, human activity recognition, machine learning concepts and models, data analysis methods, programming languages(C++, Java and Kotlin), bluetooth protocols and communication.

Acceptance Criteria

1. The IoT system integrates and facilitates seamless communication between the sensors, Arduino, and Android smartphones.
2. The system accurately collects data from IMU, blood pressure, and SpO2 sensors, ensuring precise measurements of physical activity and health parameters.
3. The Android application offers an intuitive and user-friendly interface.
4. Rigorous testing and debugging are executed to ensure the functionality and reliability of the IoT system under diverse scenarios and conditions.
5. The IoT system demonstrates optimized performance by efficiently processing data, ensuring prompt and accurate real-time monitoring and analysis.

Resources

Sensors, android phone, computer, etc.

Location

Edinburgh

Abstract

This thesis explores the development of a health monitoring system integrating Android applications and Firebase cloud services to enhance healthcare delivery, focusing on secure user authentication and online storage and access of health data. The system, designed for both patients and doctors, offers advanced data visualization tools, improving healthcare decision-making. The project also employs multi-threading, the Strassen algorithm and the Eigen library for optimized computational processes in machine learning algorithms. The study underscores the pivotal role of technology in healthcare, suggesting future enhancements in advance personalized medicine and preventive care.

Declaration of Originality

I declare that this thesis is my
original work except where stated.

.....

Statement of Achievement

In the second phase of this project, I have significantly contributed to two main domains: the development of Android applications and the exploration of machine learning acceleration techniques.

For the Android development segment, I built upon the Phase 1 foundation, enhancing the pre-existing application that measures physiological data (heart rate, blood oxygen, and physical activity). The application was augmented with user registration and login/logout functionality and Google login capabilities using Firebase's API. I transitioned all data storage from local to Firebase cloud storage, ensuring a secure and efficient data management system. Recognizing the need for effective communication between patients and healthcare providers, I developed a complementary Doctor app. This application, linked with Firebase, visually represents patients' physiological and activity data through bar graphs and scatter plots. It also features user authentication capabilities, including registration, login/logout functionalities and Google sign-in. Rigorous testing confirmed the seamless operation of both applications: the patient app displays and uploads sensor data to the cloud, while the Doctor app updates in real time.

On the machine learning acceleration front, the crux of the research focused on matrix multiplication optimization, a fundamental operation in model computation. Within the Visual Studio 2022 environment, I investigated several optimization techniques. Firstly, I examined the efficacy of multithreading using a thread pool against a baseline non-optimized approach. Subsequently, I explored the Strassen algorithm's performance in contrast to conventional methods. Finally, I tested the synergy of the high-performance Eigen matrix library with the Strassen algorithm. These experiments were successful, demonstrating marked performance enhancements, particularly with large-scale matrix operations, and validating the potential of these optimizations for broader applications in machine learning and other computationally intensive domains.

This phase's achievements not only manifest in the successful implementation and validation of the developed applications and optimization techniques but also set a robust precedent for future endeavours in enhancing computational efficiency within the field of machine learning.

Contents

List of Symbols

A Unit of electric current in the International System of Units.

V Unit of electric potential difference in the International System of Units.

Ω Unit of electrical resistance in the International System of Units.

Glossary

CNN Convolutional neural networks.

IMU Inertial Measurement Unit.

IoT Internet of Things.

ML Machine learning.

NLP Natural language processing.

RNN Recurrent Neural Network.

Chapter 1

Introduction

1.1 Phase 1 Recap: Comprehensive IoT Health-Tracking System

The primary goal of Phase 1 was to develop a reliable, effective, and user-friendly health-tracking system. This involved creating a solution that not only tracks physiological parameters like heart rate and SpO2 levels but also recognises user activities through advanced sensor integration and machine learning techniques. The system aimed to provide immediate and insightful health feedback to users[1].

1.1.1 System Development and Integration:

- **Hardware Integration:** The hardware setup for measuring heart rate and SpO2 level was built on the Arduino development board, utilising the MAX30102 sensor and HC-06 Bluetooth module for data acquisition and connectivity. This hardware setup was meticulously configured to ensure accurate and reliable measurements of heart rate and SpO2 levels. The hardware setup for human activity recognition leverages the advanced integrated IMU(inertial measurement unit) sensor Nordic Thingy:52 with a built-in Bluetooth Low Energy (BLE) module. With two IMU sensors placed on the upper and lower body of the human and a high data transmission rate of 25Hz, it ensures accurate recognition results. Users could seamlessly connect the sensors to their Android phones via Bluetooth, receiving real-time physiological data via the intuitive app interface.
- **Machine Learning Model Design:** During Phase 1, a deep dive into machine learning(**ML**) algorithms was conducted. A Convolutional Neural Network (**CNN**) was chosen for its ability to recognise patterns from the raw data provided by the Inertial Measurement Units (**IMU**). After feeding the acceleration data and gyroscope data into the ML model, up to 14 human activities can be recognised. The model was trained, tested, and optimised and reached an accuracy of 97%.
- **Android Application:** The development of the Android application was a pivotal part of Phase 1, serving as the user interface for the health-tracking system. The app was primarily coded in Java and Kotlin, and the IDE used for development was Android Studio. The user interface was

designed with a focus on simplicity and ease of use, ensuring that users could navigate through the app intuitively and access their health data effortlessly. Users were provided with real-time updates of their physiological parameters, such as heart rate and SpO2 levels, as well as their current activity status. The application also included a history feature, allowing users to view a histogram of recorded activities and their durations, giving them insight into their health and activity trends over time. The application included an algorithm to provide personalised health advice based on the user's heart rate, SpO2 readings, and recognised activities, making the health information actionable.

1.1.2 Conclusion:

Phase 1 of the project marked a significant step forward in personal health monitoring technology. By successfully integrating sophisticated hardware with advanced software capabilities, the project has laid a solid foundation for accessible, real-time health monitoring and advice. The system not only provides vital physiological data and activity recognition but also interprets this information to offer personalised, actionable health advice, thereby empowering users in their health management journey.

1.2 Motivations for Phase 2: Addressing Phase 1 Limitations

While Phase 1 established a strong foundation, integrating various sensors and pioneering machine learning techniques for real-time health insights, it illuminated several areas ripe for enhancement. The journey from Phase 1 to Phase 2 is not simply a continuation but a transformative leap propelled by the lessons learned and the ambitious vision to surmount the initial system's limitations. Recognizing the imperfections of Phase 1 was instrumental in shaping a comprehensive roadmap for Phase 2, characterized by its bold objectives to elevate user experience, system robustness, and operational efficiency.

Outlined below are the key improvement areas identified from Phase 1 and the corresponding objectives that define the essence of Phase 2:

1.2.1 Transitioning to an Online System

Phase 1 Limitation:

In Phase 1, the health-tracking system primarily relied on local data storage, confining all user data to individual devices. This approach, while initially straightforward and seemingly efficient, harboured critical vulnerabilities. The most significant was the high risk of data loss; whether due to device changes, malfunctions, or damage, users were perilously close to losing their valuable health data. This risk was not merely about losing historical information but also about interrupting ongoing health monitoring and potentially undermining long-term health management strategies.

Phase 2 Enhancement:

To decisively overcome these challenges, Phase 2 is set to implement a sophisticated online data storage solution, likely utilizing Firebase or other equivalent cloud services. This fundamental shift to cloud-based storage is poised to revolutionize the system by not only safeguarding against data loss but also enabling data syncing and access across multiple devices. Users will benefit from a continuous, device-independent health monitoring experience, bolstered by stringent security protocols and the flexibility of scalable cloud infrastructure. This enhancement is aimed at delivering an uninterrupted and enriched user experience, ensuring that every individual's health data is consistently accessible, secure, and integrated within the broader health management ecosystem.

1.2.2 Enhancing Multi-user Accessibility

Phase 1 Limitation:

The initial phase of our health-tracking system was designed with a single-user model. This approach, while functional for individual users, significantly curtailed the system's applicability in broader, multi-user scenarios. Environments such as family health management, group fitness activities, or patient monitoring in a healthcare setting were beyond the system's reach due to the inability to accommodate multiple user profiles and manage diverse health data streams effectively.

Phase 2 Enhancement:

In response to the limitations identified in Phase 1, Phase 2 will introduce user authentication with login, logout, and registration functionalities.
Each user can create and maintain a unique account, ensuring that their health data and insights are personalized, secure, and isolated from other users. Users can track their health data over time with individual profiles, allowing for longitudinal health monitoring and analysis.
This enhancement will allow multiple users to maintain individual profiles and health records, making the app more versatile and user-friendly. With a comprehensive user authentication system and multi-user support, the health-tracking system will benefit a wider range of individuals, families, and healthcare settings, ensuring that more people can take advantage of personalized, continuous health monitoring and insights.

1.2.3 Enabling Doctor Access

Phase 1 Limitation:

The lack of healthcare provider access in Phase 1 meant missed opportunities for integrating personal health data into broader health management and treatment plans.

Phase 2 Enhancement:

To bridge this gap and maximize the potential of the health-tracking system in professional healthcare settings, Phase 2 will focus on developing a dedicated interface for healthcare providers. This enhancement involves several key developments. Firstly, healthcare providers will be able to access patient data securely and conveniently, with adherence to privacy laws and regulations. Access will be contingent on patient consent, ensuring respect for patient privacy and autonomy. Then, the interface will be designed to provide a comprehensive view of the patient's health data, including historical trends, real-time metrics, and predictive insights. It will be intuitive and user-friendly, allowing healthcare providers to easily navigate and interpret the information.
Phase 2's healthcare provider interface significantly enhances clinical utility by enabling secure access to health data, fostering improved care and patient outcomes.

1.2.4 Optimizing Machine Learning Training Speed

Phase 1 Limitation

The initial phase was challenged by slow training times and less-than-ideal performance of the machine learning models. The need for enhanced computational strategies and model optimization became evident to ensure the system's effectiveness and user satisfaction.

Phase 2 Enhancement

A part of Phase 2 is committed to accelerating the machine learning training process through using more efficient algorithms. Adoption of more efficient algorithms specifically designed to speed up the training process without compromising the model's performance or accuracy.

These enhancements are aimed at not only overcoming the limitations encountered in Phase 1 but also at expanding the system's capabilities and applications. By addressing these imperfections, Phase 2 will advance the project towards a more integrated, efficient, and versatile health management solution.

1.3 Phase 2 Advancements

Phase 2 of this project marks a significant stride in both the development of Android applications and the advancement of machine learning acceleration techniques. Building on the groundwork in Phase 1, I have enhanced the existing patient application, which monitors physiological data, with user authentication capabilities using Firebase's API. This includes integrating user registration, login/logout functionality, and Google login options, as well as transitioning data storage from local devices to Firebase cloud services for increased security and accessibility.

In tandem, I developed a Doctor app, creating a bridge between patients and healthcare providers by enabling real-time access to patient data through Firebase. This application presents data via intuitive graphs, providing essential insights at a glance. With rigorous testing, both applications demonstrated flawless functionality, with patient data synchronously reflected across platforms, ensuring up-to-date information is always available for both patients and doctors.

Additionally, I have delved into optimizing machine learning processes, focusing on the pivotal operation of matrix multiplication. Within Visual Studio 2022, my experimentation covered a range of optimization techniques. I began by leveraging multithreading with a thread pool to enhance performance over the standard approach. Following this, I implemented the Strassen algorithm, observing its superiority in computational efficiency compared to traditional methods. My exploration culminated by integrating the high-performance Eigen library with the Strassen algorithm, which yielded significant improvements, particularly in handling large-scale matrix operations.

These efforts in Phase 2 have not only solidified the functionality and reliability of the developed applications but have also illustrated the profound potential of computational optimizations in machine learning. The successful implementation of these advanced techniques sets a strong foundation for future projects to build upon, aiming to further elevate the efficiency and effectiveness of health monitoring systems and machine learning operations.

Chapter 2

Literature Review

2.1 Introduction

In the dynamic and interconnected realm of health technology, sophisticated data storage solutions and rapid advancements in computational techniques have significantly reshaped the landscape. This literature review seeks to provide an in-depth exploration of health-tracking technologies, with a pronounced emphasis on the transformative impact of online data storage and management, as well as the critical role of machine learning and accelerating machine learning training processes. As we explore the evolution and current state of these technologies, their significance in enhancing the effectiveness and adaptability of health monitoring becomes clear.

This review is commenced with a historical overview of health-tracking technologies, tracing their journey from inception to today's sophisticated, integrated systems. Special attention is given to the revolution in data management, as the shift from local to cloud-based storage has substantially mitigated risks and opened new possibilities for health data analytics, remote monitoring, and improved system resilience.

The review then shifts to machine learning's role in health monitoring. It examines the array of techniques used, their application in real-world scenarios, and the impact of accelerating the training process on system responsiveness and predictive power. By examining strategies like parallel processing and algorithm optimization, including Strassen's algorithm and GPU acceleration, the review highlights the ongoing efforts to enhance model training efficiency and overall system performance.

This literature review transcends a mere research synthesis; it weaves the narrative of health-tracking technology's evolution, current challenges, and future potential. It emphasizes the critical need for robust, scalable online data storage and rapid machine learning training, marking them as essential for the next generation of health monitoring systems.

2.2 Historical Development of Health-Tracking Technologies

2.2.1 Early Beginnings of Health-Tracking Technologies

The inception of health-tracking technologies predates the digital era, originating from the fundamental human desire to understand and enhance physical health and performance. Initially, these technologies were mechanical and simple in their functionality, reflecting the technological constraints of their time.

Mechanical Pedometers

One of the earliest health-tracking devices is the pedometer[2], with concepts dating back to Leonardo da Vinci and practical devices developed in the 17th century[3]. These mechanical pedometers estimated walking distance using a simple lever or pendulum mechanism that advanced a counter with each step taken. Early adopters wore these devices around their waist or carried them in pockets, using them to gauge physical activity levels. Despite their innovative design, these early pedometers were often limited in accuracy and durability[4].

Early Wearable Devices

Beyond step counting, early wearable devices focused on single health metrics like pulse rate or body temperature. Devices such as pulse watches became tools for individuals, particularly athletes and patients with heart conditions, to monitor their physiological states. These wearables were predominantly mechanical, requiring manual operation and often limited to the information they could provide[5].

Simplicity and Limitations

The simplicity of these early devices made them accessible to a broad audience but also meant they were limited in functionality and precision. They provided no means for data storage or in-depth analysis and were prone to inaccuracies due to mechanical failures or human errors. However, their existence marked the beginning of the quantified self-movement and set the foundation for future advancements in personal health monitoring technologies[5].

Cultural and Technological Significance

These initial health-tracking devices served as both technological innovations and cultural milestones, reflecting a growing interest in personal health and the beginnings of the wearable technology market. They heralded a new era of health consciousness and paved the way for continuous innovation in the field of personal health monitoring[6].

2.2.2 Technological Milestones in the Evolution of Wearables

The trajectory of wearable health technologies is punctuated by key technological advancements that significantly transformed their capabilities and applications.

From Mechanical to Electronic

The evolution from mechanical to electronic systems marks a pivotal milestone in wearable technology. This shift introduced electronic components, replacing mechanical counters and paving the way for improved accuracy, digital displays, and interactive functionalities[7].

Introduction of Heart Rate Monitoring

Incorporating heart rate monitoring into wearable devices was a significant breakthrough, transitioning from bulky clinical equipment to portable monitors for personal use. This advancement expanded the scope of wearable devices to include vital sign monitoring during various activitiesc[8].

GPS and Environmental Sensing

The addition of GPS technology and environmental sensors enhanced the utility of wearables, enabling location tracking, route mapping, and the collection of environmental data, thereby enriching the context and analysis of health and fitness data[9], [10].

2.2.3 Early Integration of Technology

The foundation for today's sophisticated health-tracking systems was laid by the initial integration of digital interfaces and sensors into wearable devices.

Digital Interfaces

The transition to digital interfaces represented a leap forward from mechanical displays, offering users improved interaction with their devices and laying the groundwork for subsequent advancements in user interface design, including touchscreens and mobile app integration[11], [12].

First Generation Sensors

Early wearable devices incorporated basic motion detectors and health metric sensors, setting a precedent for the wide array of health data that modern wearables can monitor. Despite their limitations, these first-generation sensors were crucial in demonstrating the potential of wearables to track a variety of health-related metrics[11], [12].

2.3 Current State of the Art in Health-Tracking Technologies

2.3.1 Advanced Sensor Technology

Modern health-tracking technologies are characterized by advanced sensor technologies, significantly broadening the scope and accuracy of personal health monitoring.

Innovative Sensor Types: Devices now incorporate a range of sensors such as 3D accelerometers, photoplethysmographs (PPGs), and bioimpedance sensors. For example, the Apple Watch's PPG sensor provides sophisticated heart rate monitoring and atrial fibrillation detection[13], [14].

Integration in Diverse Environments: Sensors are integrated into various environments, including smartwatches and smart clothing. Smart fabrics in athletic wear provide real-time feedback on performance and form[15]. Moreover, devices like the Dexcom G6 allow diabetics to continuously monitor glucose levels, showcasing the advanced capabilities and benefits of modern sensors in health management[16].

2.3.2 Modern Wearable Devices

The sophistication and variety of modern wearable devices have significantly enhanced personal health monitoring and user interaction.

Comprehensive Health Tracking: Contemporary wearables like the Samsung Galaxy Watch monitor a wide array of health metrics, including stress levels and sleep patterns, and offer suggestions for improving health and well-being[17].

Enhanced User Experience: Devices such as the Garmin Venu 2 enhance user experience with features like high-resolution touchscreens, voice commands, and detailed health reports[17].

Healthcare Integration: Wearable devices are increasingly integrating directly with healthcare systems, exemplifying the Withings ScanWatch's ability to detect and communicate critical health conditions to healthcare providers[17].

2.3.3 Mobile Health Applications and Integration

Mobile health applications serve as central platforms for holistic health management, integrating data from various sources and offering comprehensive health insights[18], [19].

Holistic Health Platforms: Apps like MyFitnessPal and Apple's HealthKit aggregate data from wearables and other health devices to provide a comprehensive view of an individual's health, covering aspects from nutrition to physical activity and mental health[20].

Personalized Health Insights: Utilizing machine learning and big data, apps can offer personalized health insights and predictions. The Flo app, for instance, provides personalized cycle tracking and health insights using AI algorithms[21].

Telehealth and Remote Monitoring: Integration with telehealth services has become more prevalent, with platforms like Teladoc Health offering remote monitoring and virtual consultations by integrating with wearable devices, making healthcare more accessible[22].

The current landscape of health-tracking technologies reflects a robust and sophisticated field, offering a comprehensive approach to health monitoring and management. These technologies continue to evolve, promising even more integrated, predictive, and personalized solutions in the future.

2.4 Transformative Impact of Online Data Storage and Management

The shift from local to cloud-based storage systems has significantly transformed the landscape of health-tracking technologies, offering enhanced data accessibility, security, and analysis capabilities[23].

2.4.1 Shift to Cloud-Based Systems

The transition to cloud-based systems has brought about a paradigm shift in how health data is stored, accessed, and utilized, marking a turning point in health monitoring technology[24], [25].

Enhanced Accessibility and Continuity:

Cloud-based systems such as Google Cloud Healthcare API and Amazon Web Services provide secure, continuous access to health data, enabling users to monitor their health over time and across different devices[26]. For example, the Fitbit platform leverages cloud storage to sync and store user data seamlessly[27].

Robust Security Measures:

Platforms like Microsoft Azure offer healthcare organizations HIPAA-compliant cloud storage solutions, ensuring end-to-end encryption and stringent access controls to protect sensitive health data[28].

Real-Time Data Syncing and Backup:

Services such as iCloud Health ensure that users' health data is continuously backed up and updated in real-time, facilitating timely health monitoring and decision-making[29].

2.4.2 Advantages and Innovations

The adoption of cloud-based storage has introduced numerous advantages and innovations in health-tracking technologies, significantly enhancing data management and analytic capabilities.

Scalability and Flexibility:

Cloud solutions like AWS enable health-tracking applications to effortlessly accommodate increasing amounts of data and users, eliminating the need for extensive physical infrastructure[30].

Real-Time Monitoring and Predictive Analytics:

Healthcare providers are adopting cloud-based platforms for real-time patient monitoring and predictive analytics, improving patient outcomes and care efficiency. Philips' eICU program is a prime example of how cloud technology is being used to offer advanced monitoring services in critical care[31].

Deep Insights Through Big Data Analytics:

IBM Watson Health utilizes cloud storage and computing to analyze diverse health datasets, contributing to advances in personalized medicine and population health management by providing deeper insights and more accurate health predictions[32].

Integration with Telemedicine and Remote Care:

Cloud-based health data is central to telemedicine platforms, enabling remote consultations, monitoring, and care management. Services like Teladoc and Doctor on Demand rely on cloud storage to integrate patient health data with video consultations and other remote care tools[33].

Health Research and Collaboration:

Initiatives like the All of Us Research Program demonstrate the role of cloud storage in facilitating large-scale health research and collaboration, aiming to advance precision medicine through the analysis of health data from over a million participants[34].
In summary, the adoption of cloud-based data storage and management represents a major leap forward in health technology, offering scalability, real-time monitoring, and deep insights through big data analytics. As these systems continue to evolve, they promise to further revolutionize health monitoring, making it more reliable, insightful, and integral to personal and public health management.

2.5 Machine Learning's Role in Health Monitoring

2.5.1 Evolution of Machine Learning Applications

Machine learning (ML) has significantly evolved in the context of health monitoring, from basic data classification to sophisticated predictive modeling, enhancing personalization and accuracy in health care.

Early Applications: Initially, machine learning in health monitoring was employed for simple pattern recognition tasks using algorithms such as decision trees and linear regression. These methods were foundational in analyzing and interpreting health data[35].

Growth of Sophisticated Models: As computational capabilities expanded, so did the complexity of ML models. Deep learning, with architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (**RNN**), has become particularly influential, allowing for the analysis of complex patterns in large datasets[36], [37].

Predictive Capabilities and Personalization: Contemporary ML applications focus heavily on predictive analysis and personalized health insights. These models use historical data to forecast potential health issues and customize recommendations based on individual patient profiles, thereby improving the effectiveness of health interventions[35].

2.5.2 Case Studies of Successful Implementation

Real-world case studies illustrate the impactful role of machine learning in enhancing health monitoring and management.

Predicting Patient Deterioration: Hospitals utilize ML models to predict patient deterioration, enhancing care responsiveness. Tools like the Rothman Index integrate various health metrics to provide early warnings and improve patient outcomes[38].

Wearable Health Monitors: Wearable devices such as the Apple Watch use ML algorithms to detect irregular heart rhythms and potential falls, providing timely alerts to users and health professionals[13].

Managing Chronic Conditions: ML models are increasingly used in managing chronic conditions, exemplified by Medtronic's closed-loop insulin pumps for diabetes management. These systems adjust treatment in real-time, based on continuous glucose monitoring, demonstrating the personalized approach facilitated by machine learning[39].

Mental Health Applications: Applications like Mindstrong and Talkspace employ ML to analyze user interactions and provide insights into mental health conditions, facilitating timely therapeutic interventions and support[40].
Through these case studies, the profound and diverse impacts of machine learning in health monitoring are evident, showcasing its potential in advancing care quality, predictive accuracy, and personalized treatment. As machine learning technology continues to evolve, its applications in health monitoring are expected to expand further, offering even more sophisticated tools for health care and management.

2.6 Accelerating Machine Learning Training

The acceleration of machine learning (ML) training processes is a critical area of focus across various industries, driving significant improvements in fields such as health monitoring, autonomous systems, and predictive analytics. Rapid and efficient training of ML models is essential to handle the increasing data volumes and complexity, enabling timely insights and innovations. Despite the substantial benefits, accelerating ML training presents challenges including computational demands, data management, and maintaining model accuracy.

2.6.1 Challenges in Accelerating Machine Learning Training

Accelerating the training of machine learning models is essential across various domains but comes with significant challenges related to data complexity, computational resources, and model generalization[41], [42].

Data Complexity

High-dimensional and diverse datasets pose a significant challenge in machine learning. As data grows in volume and complexity, it requires sophisticated preprocessing and feature engineering to make it suitable for training efficiently. The variability in data, especially in areas like health monitoring, adds layers of complexity that can slow down the training process and demand more computational power[43].

Computational Resources

The computational demand for training complex ML models is substantial. The need for high processing power, memory, and storage to handle extensive datasets and maintain the model's parameters can become a bottleneck in the training process. Access to GPUs, TPUs, and distributed computing environments is often necessary, posing a challenge, especially for those with limited resources[44].

Overfitting and Generalization

Ensuring that models generalize well to new data while speeding up the training process is a significant challenge. Overfitting occurs when a model learns the training data too well, including its noise and peculiarities, and performs poorly on unseen data. Techniques like regularization, cross-validation, and early stopping are employed to combat overfitting, but they need to be carefully balanced with efforts to accelerate training[41].

2.6.2 Innovative Solutions for Acceleration

To address the challenges in machine learning training, a variety of innovative solutions have been developed, focusing on parallel computing, efficient algorithms, transfer learning, and network optimization techniques.

Parallel Computing and GPUs

Parallel Computing: Utilizes methods that allow for simultaneous processing of operations, significantly speeding up data processing and model training. Techniques such as distributed computing and multi-threading enhance parallelism within machines and across networks[45], [46].
GPUs: Graphics Processing Units are instrumental in accelerating machine learning training. Their ability to perform multiple operations concurrently makes them particularly effective for the matrix and vector computations that are common in machine learning[47].

Efficient Algorithms

Optimization Techniques: Improvements in optimization algorithms, such as Stochastic Gradient Descent (SGD) and its variants, have enhanced the efficiency of the training process. These techniques optimize the path to convergence, reducing the time required to train models[48].
Algorithmic Efficiency: Advances in algorithms improve the construction and training of models. These include better initialization methods, adaptive learning rates, and avoidance of local minima, contributing to quicker and more effective training.

Transfer Learning and Pre-trained Models

Transfer Learning: Involves using a model trained on one task as the starting point for another task. By leveraging the features learned from related tasks, transfer learning can drastically reduce training time and data requirements[49].
Pre-trained Models: Utilizing models that have been pre-trained on large datasets allows practitioners to bypass the initial phase of training. Instead, models are fine-tuned on specific tasks, reducing overall training time while leveraging learned features[50].

Network Pruning and Quantization

Network Pruning: This technique involves removing non-significant parameters from a model to reduce its complexity and size. Pruning results in a more compact model that requires less time to train and is more efficient to run[41].
Quantization: Reduces the precision of the model's parameters to decrease the model size and increase training and inference speed. Quantization is beneficial in deploying models in resource-constrained environments[41].
These solutions collectively represent a comprehensive approach to accelerating the training of machine learning models. By improving computational efficiency, optimizing algorithms, and reducing model complexity, these techniques contribute to faster, more efficient, and scalable machine learning processes.

2.6.3 Case Studies and Applications

Accelerated machine learning training has made significant impacts across various sectors. Here are notable examples demonstrating this impact:

Image Recognition

In the domain of image recognition, accelerated training has enabled rapid improvements in model accuracy. **Google's Inception model**, trained on ImageNet, is a prime example where faster training cycles have markedly advanced computer vision capabilities[51].

Natural Language Processing (NLP)

Accelerated training in NLP has led to the development of complex models like **BERT**, which can now be trained more quickly and efficiently, advancing language understanding and generation capabilities significantly[52].

Health Monitoring

In health monitoring, accelerated training enables the rapid deployment of predictive models. A notable case is the implementation of machine learning models to predict patient deterioration in ICUs, leading to more timely and effective interventions.

2.6.4 Future Implications and Directions

The acceleration of machine learning training has far-reaching implications for various sectors, promising transformative changes in data processing and utilization.

Real-time Data Processing

As training times decrease, real-time data processing and analysis become more feasible. This is crucial for applications like autonomous driving or real-time fraud detection, where the ability to quickly retrain and update models is vital for maintaining performance[53].

Accessibility and Democratization

Faster training times democratize access to machine learning, making it feasible for smaller organizations and individual researchers to develop and deploy advanced models. This shift is likely to lead to a broader adoption and innovation in machine learning across diverse sectors[54].

Innovations in Health Monitoring

For health monitoring, accelerated training promises more adaptive and personalized healthcare solutions. Rapid model updates can better reflect new health trends, patient data, or emerging conditions, leading to improved care and health management[55].

2.7 Summary

This comprehensive literature review encapsulates the dynamic evolution of health-tracking technologies, underscored by the transition from rudimentary mechanical devices to sophisticated systems integrated with advanced sensor technology and modern wearables. It highlights the transformative shift to cloud-based data management and the pivotal role of machine learning, which together drive the efficacy and personalization of health monitoring. The review also acknowledges the challenges and innovations in accelerating machine learning training, essential for handling increasing data volumes and complexity.

Overall, the review charts a trajectory of continuous innovation and adaptation in health-tracking technologies, promising enhanced, predictive, and personalized healthcare solutions in an increasingly digital world.

Chapter 3

System Requirement

3.1 System Function Overview

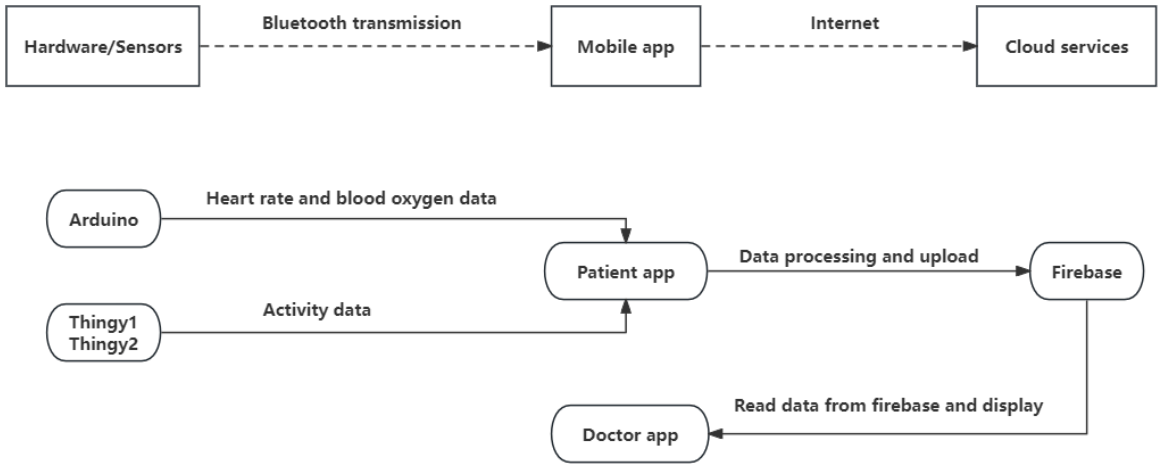

Figure 3.1: System function structure

Phase 2 of the project builds on the foundation set by Phase 1, mainly introducing significant upgrades with the addition of cloud capabilities and a dedicated doctor's interface.

The whole system comprises three main components: hardware (sensors), mobile applications (Doctor and Patient apps), and cloud services (Firebase[56]).

First, sensors are connected to mobile phones(Patient app) via Bluetooth, including a MAX30102 sensor controlled by Arduino for heart rate and blood oxygen level measurements and two Thingy:52 motion sensors for human activity recognition.

The Patient app receives data from the sensors and displays the user's physiological data. It then uploads the data to Firebase, which stores the physiological data of all users categorized by username, date, and data type.

Firebase serves as the central data repository, managing and storing patient data securely.

The Doctor app retrieves patient data from Firebase, displaying it in graphical formats such as bar charts and scatter plots, providing physicians with insights into patients' physiological conditions.

Figure3.1 illustrates the function block diagram for the project.

3.1.1 Patient App Development in Phase 2

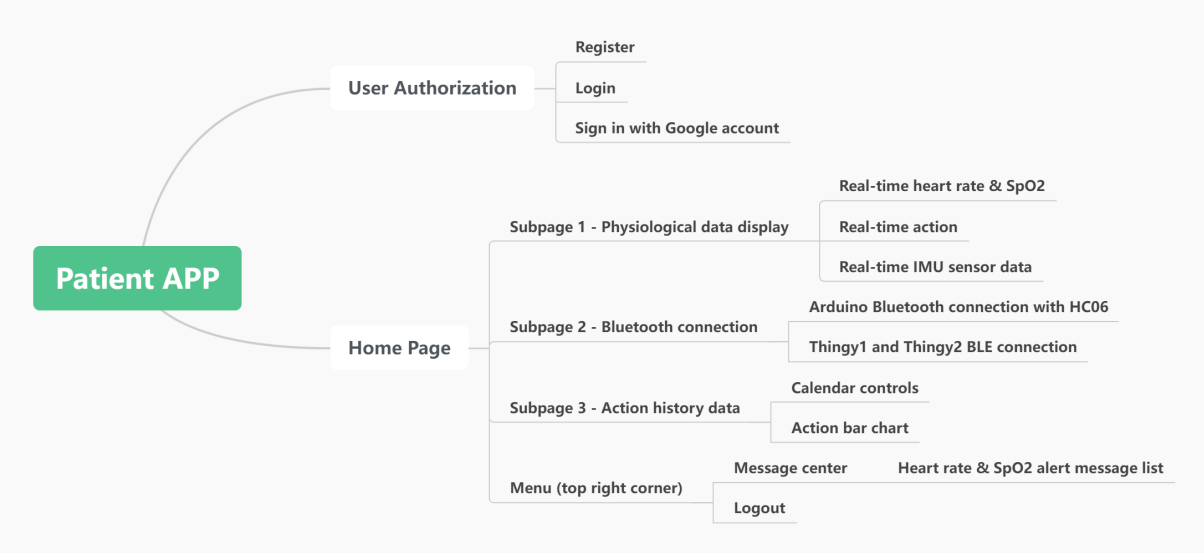

Figure 3.2: Patient app structure

The Patient App in Phase 2 evolves from its Phase 1 iteration, incorporating enhancements for improved user interaction and data management.

Core Functionalities: The app continues to perform its primary functions established in Phase 1, such as measuring physiological data, displaying this information, issuing health prompts, and logging activity history.

Authentication Features: A key addition is the integration of user authentication features. Users must register or log in before accessing the health monitoring functionalities, ensuring a personalized and secure experience. The app facilitates seamless user account switching during usage.

Cloud-Based Data Management: Transitioning from local to cloud-based storage, the app now sends user data directly to Firebase's cloud services. This shift enhances data safety, accessibility, and enables real-time data synchronization and analysis.

Figure3.2 illustrates the function diagram for the patient app.

3.1.2 Doctor App Development in Phase 2

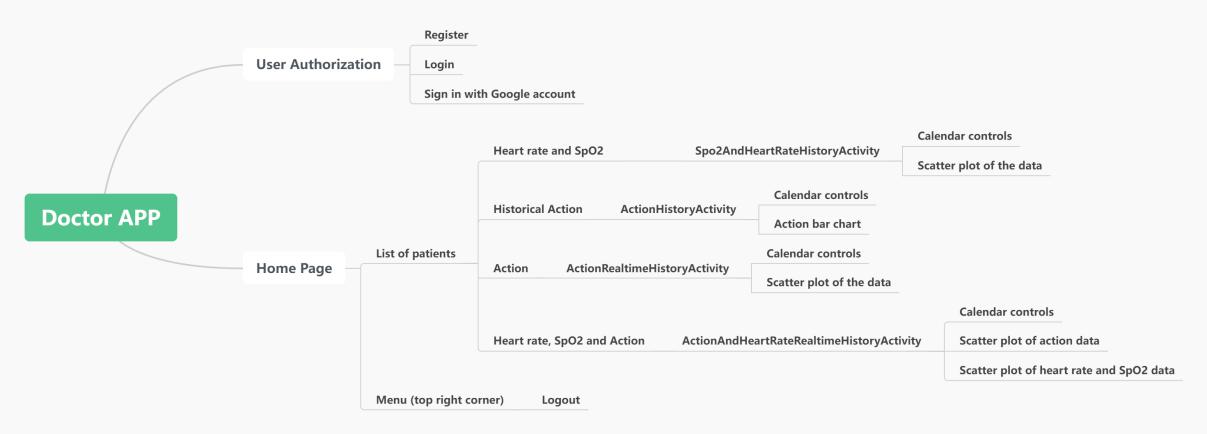

Figure 3.3: Doctor app structure

The Doctor App, a new development in Phase 2, is designed to provide doctors with an efficient means to access and analyze patient data.

User Authentication: Like the Patient App, the Doctor App features user registration, login, and logout functionalities, ensuring secure access for each medical professional.

Patient Data Visualization: Upon logging in, doctors are presented with a list of patients. The app allows access to detailed health data with the following key features:

- Heart Rate and SpO2 Over Time: Graphical representation of heart rate and blood oxygen levels over time.
- Duration of Each Activity: Information on the duration of each type of physical activity undertaken by the patient.
- Activity Over Time: Graphical representation of the patient's activity over time.
- Heart Rate, SpO2 and Activity Over Time: An integrated view of heart rate, blood oxygen, and activity data over time for in-depth analysis.

Integration with Firebase: The Doctor App is directly connected to Firebase cloud services for data retrieval. This ensures that doctors have access to the most current patient data for analysis.

Figure3.3 illustrates the function diagram for the doctor app.

3.2 Development Tools

In Phase 2, new development tools were introduced to enhance the system's capabilities. Highlighted below are the key development tools that stand out in Phase 2.

3.2.1 Firebase

In Phase 2, Firebase plays a crucial role, particularly through its Authentication and Realtime Database services.

Firebase Authentication

Firebase Authentication provides a comprehensive backend solution for user identity verification in applications. It supports various authentication methods including email and password, phone numbers, and federated identity providers like Google and Facebook. Adhering to OAuth 2.0 and OpenID Connect standards, it ensures seamless integration with other Firebase services and custom backends, offering secure and scalable user identity management.

Firebase Realtime Database

Firebase Realtime Database is a cloud-hosted NoSQL database that enables the storage and real-time synchronization of data in JSON format across connected clients. It maintains data availability even when offline, syncing local changes with remote updates upon reconnection. Ideal for collaborative applications, this database allows direct access from client devices and is equipped with robust security features, ensuring data integrity and protection.

3.2.2 Visual Studio 2022

Visual Studio 2022 emerges as a cornerstone in Phase 2 for C++ development, offering enhanced performance and handling of large codebases, sophisticated debugging tools for efficient problem resolution, and advanced code analysis to maintain high-quality standards. Its intuitive interface, coupled with integration capabilities with modern technologies and frameworks, streamlines the development workflow. This IDE not only accelerates the development process but also fosters team collaboration, making it an indispensable tool for building robust, efficient, and scalable software solutions.

The machine learning model in this project is developed using TensorFlow. Given that TensorFlow's core is written in C++, Visual Studio 2022 (VS2022) has been chosen as the primary development tool for this project.

3.2.3 Git and GitHub

Git and GitHub are highlighted as essential tools in the development tool section for their roles in version control and collaborative development.

Git - Version Control: Git excels in tracking and managing code changes, enabling multiple parallel development workflows. Its branching feature allows for organized and structured coding processes.

GitHub - Collaboration and Code Management: As a cloud-based repository hosting service, GitHub centralizes code collaboration. It enhances code quality through features like pull requests and code reviews. Additionally, GitHub facilitates CI/CD integration, ensuring efficient and error-free deployments.

Documentation and Issue Tracking: GitHub also supports comprehensive project documentation and provides an effective system for issue tracking, adding transparency and organization to the development process.
Git and GitHub are indispensable for efficient version control and collaborative software development, offering an organized and transparent environment for managing the project's codebase.

3.3 Summary

This section encapsulates the significant advancements realized in Phase 2 of the health-tracking project. Central to this phase is the integration of comprehensive cloud services and the debut of a dedicated interface for doctors, marking a pivotal evolution in the project's scope and capabilities.
Key developments include an expanded system architecture that encompasses hardware sensors, patient and doctor mobile applications and Firebase's central role in secure and real-time data management. This phase is distinguished by the adoption of Firebase for robust user authentication and data synchronization, the utilization of Visual Studio 2022 for sophisticated C++ development, and the implementation of Git and GitHub for efficient version control and collaborative software development.
Phase 2 represents a significant leap in enhancing data security, improving user interaction, and bolstering the system's overall efficiency. This chapter provides a comprehensive overview of these enhancements, highlighting the project's commitment to innovation and user-centric design in health monitoring technology.

Chapter 4

Design and Implementation

4.1 Introduction

The implementation part of the project divides into two primary domains: the development of the mobile applications and the acceleration of machine learning processes.

4.1.1 App Development

The project delves into constructing a comprehensive mobile application. Key features include the implementation of Firebase Authentication to ensure secure user access, the integration of the Firebase Realtime Database for real-time data handling, and the development of the Doctor app. The Doctor app is tailored to provide healthcare professionals with an insightful and seamless user experience.

4.1.2 Machine Learning Acceleration

This segment focuses on enhancing computational efficiency in the health monitoring system. Emphasis is placed on applying Strassen's algorithm, recognized for its efficient matrix multiplication, crucial in machine learning computations. Additionally, the project explores the use of thread pooling to facilitate multithreaded operations, with the goal of significantly improving the performance and speed of machine learning algorithms.

4.2 Android Application Development

4.2.1 User Authorization in Android Application Development

The development of the user authorization system is integral to the Android applications in Phase 2, focusing on the initial user experience.

User Authorization Interface: At the launch of the app, users encounter the user authorization screen. This interface is designed to be clear, intuitive, and user-friendly, providing a welcoming first point of interaction.

Email and Password Authentication:

- Users can register or log in using an email address and password. The system ensures each email is unique to prevent duplicate accounts.
- Error messages are displayed for attempts to register with an existing email or incorrect password entries during login, guiding users towards successful authentication.

Google Account Integration:

- The app includes a feature for one-tap registration and login using Google accounts, streamlining the user experience.
- This integration offers ease of access and enhances user convenience, expanding the app's accessibility.

Transition Post-Login: Upon successful authentication, the app confirms the success and smoothly transitions the user to the functional areas, ensuring a positive and seamless user experience from the start. To logout, the button on the top-right corner can help to switch users.

Figure4.1to Figure4.8 show the user authorization system for the patient application.

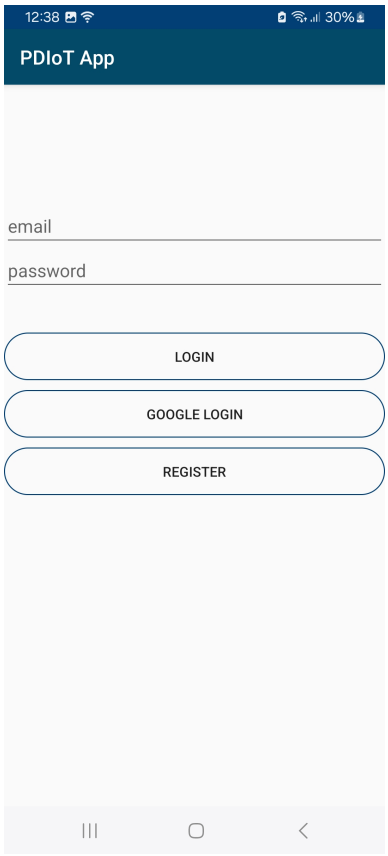

Figure 4.1: Login page

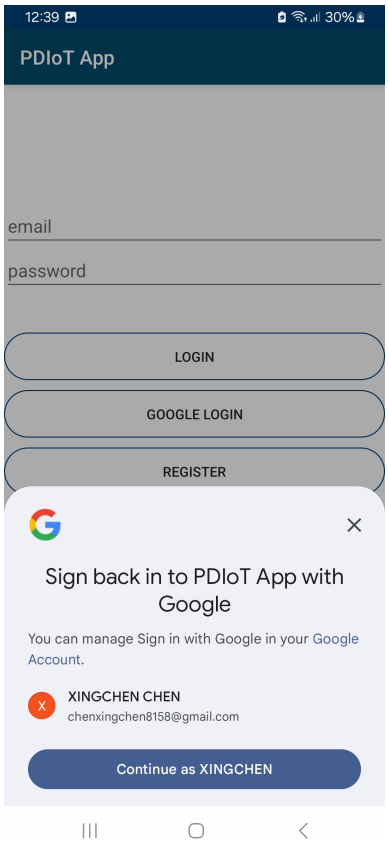

Figure 4.2: Login with Google

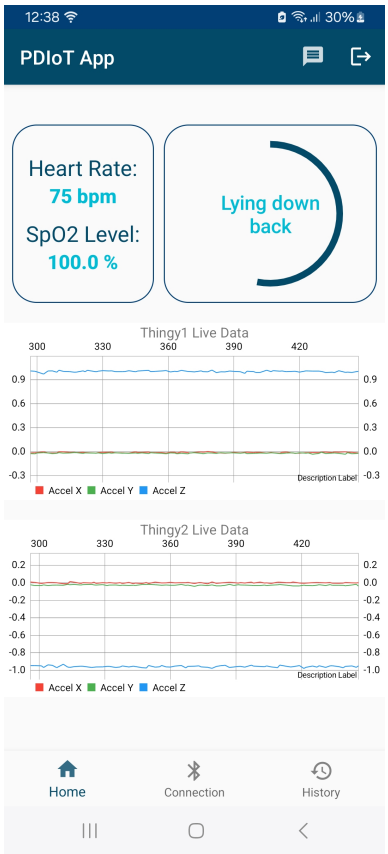

Figure 4.3: Home page with logout function

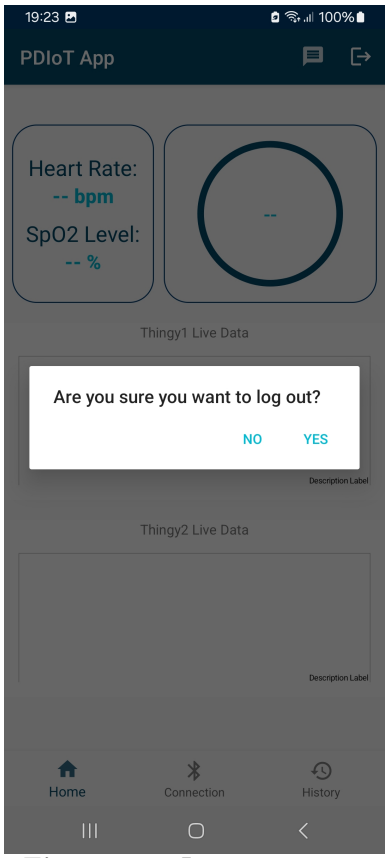

Figure 4.4: Logout message

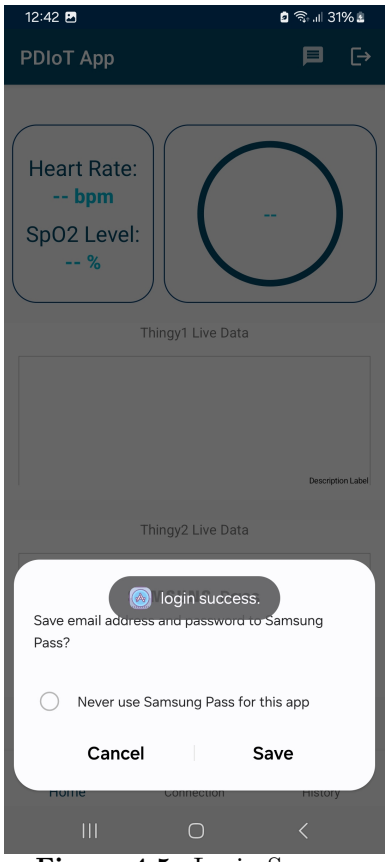

Figure 4.5: Login Success

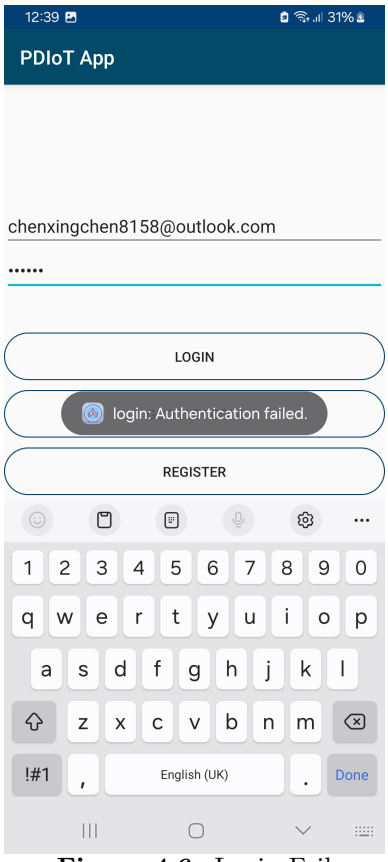

Figure 4.6: Login Fail

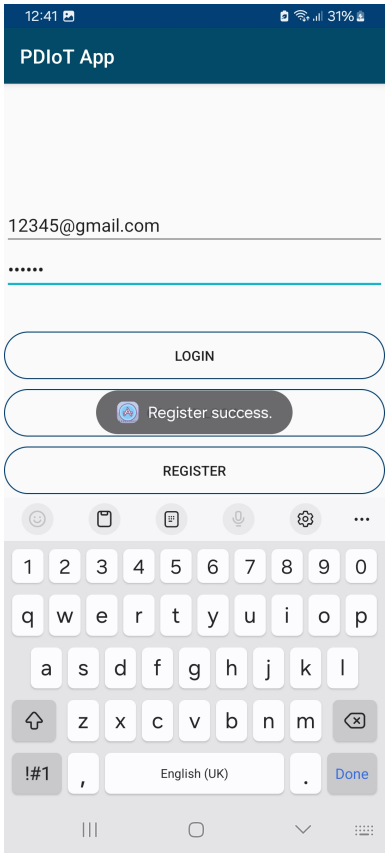

Figure 4.7: Successful registration

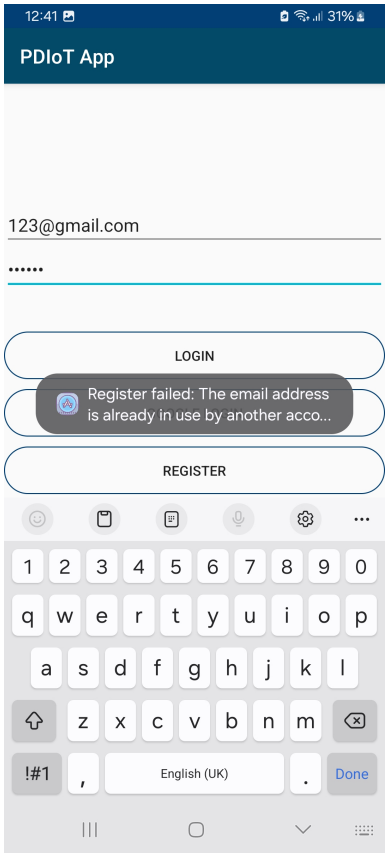

Figure 4.8: Registration failed

4.2.2 User Authentication implementation

This section details the implementation of user authentication methods in the Android application, including Google Sign-In, email/password login, logout, and registration functionalities.
The block diagram of Login is shown in Figure 4.9.

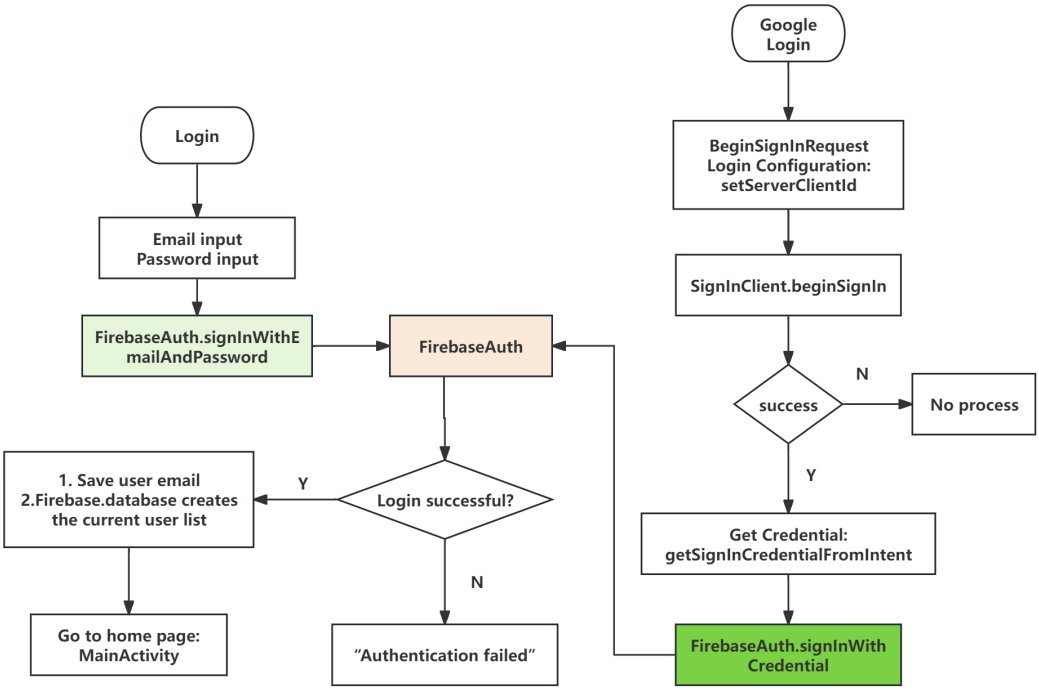

Figure 4.9: Block diagram for Login function

Google Sign-In (`googleLogin`):

- Initiates Google Sign-In using `oneTapClient`, a `SignInClient` instance.
- On success, attempts to launch the Google Sign-In intent.
- On failure to find saved credentials, logs the error and retains the signed-out UI.

Email/Password Sign-In (`login`):

- Retrieves and logs the user's email and password from the UI.
- Authenticates using Firebase's `signInWithEmailAndPassword` method.
- On success, logs the event, displays a success message, and calls `onLoginSuccess` to update the UI.

- On failure, logs the error and displays a failure message.

The block diagram of Register is shown in Figure 4.10.

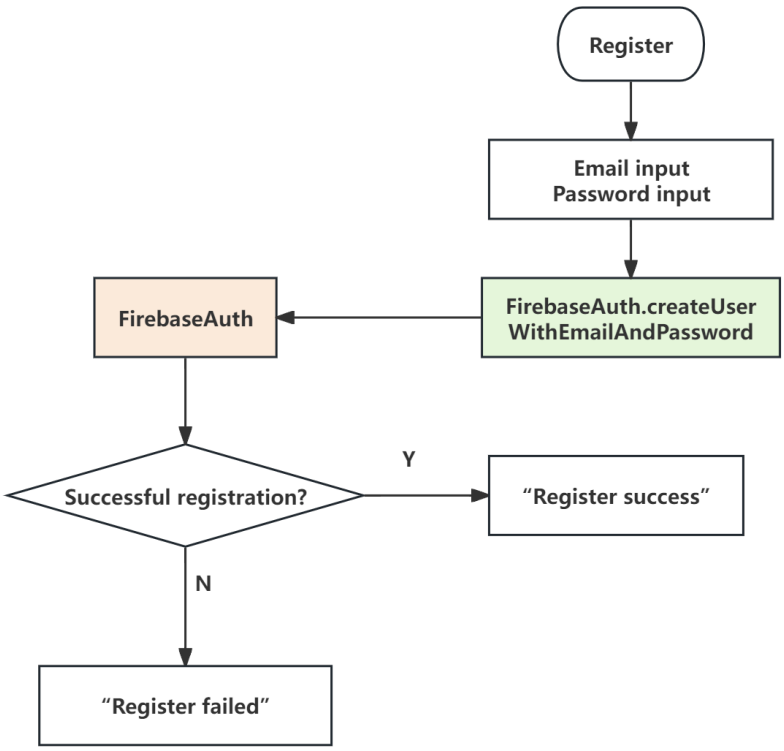

Figure 4.10: Block diagram for Register function

Email/Password Registration (`register`):

- Similar to the login function, retrieves and logs email and password.
- Registers a new user using Firebase's `createUserWithEmailAndPassword`.
- On successful registration, logs the success, shows a success message, and updates the UI.
- On failure, logs and displays an error message.

Logout Functionality (`logout`):

- Simply executes Firebase's `signOut` method to log out the current user.

These functions collectively manage the app's user authentication process, offering a well-rounded approach to handling sign-ins, sign-outs, and user registrations, leveraging Firebase's authentication services and Google's One Tap sign-in.

4.2.3 Data Storage Structure Design in Firebase Realtime Database

The Firebase Realtime Database for the Android application is designed with a hierarchical structure for efficient data organization and retrieval.

User-Specific Classification:

- Data is categorized at the top level by individual users, identified by a unique `userInfo` containing details like the user's email.
- This approach ensures personalized data handling and secure access to each user's health information.

Date-Level Organization:

- Data under each user is further organized by dates, allowing tracking of health metrics over time.
- Date entries provide a daily record of the user's health and activities.

Health Data Categories: Within each date entry, the data is segmented into four key categories:

1. **Action:** Duration of each type of activity throughout the day.
2. **ActionRealtime:** The specific activity the user was engaged in at each recorded moment.
3. **AlertMsg:** Alert messages generated, related to health warnings or notifications.
4. **Spo2AndHeartRate:** Heart rate and blood oxygen (SpO2) levels recorded at various times during the day.

The structure of the database is shown in Figure 4.11. This structured approach enables a comprehensive and chronological view of health metrics for each user, supporting the application's functionality in activity tracking, health monitoring, and alert generation. It forms an integral part of the backend architecture for efficient management and real-time updates of health data.

Figure 4.11: The structure of the database

4.2.4 DatabaseManager Implementation

Data List Initialization

Cloud and Local List Initialization: On the user's first login of the day, the system initializes four types of data lists in the cloud. These lists correspond to different health data categories such as `AlertMsg`, `Spo2AndHeartRate`, `ActionRealtime`, and `Action`. Simultaneously, the application creates corresponding empty lists locally to mirror the cloud-based data. This ensures readiness for managing and displaying the user's health data.
`addUserinfoToDb(userInfoBean: UserInfoBean)` Updates user information in the database, setting data in a specific path based on the user ID.

Data Synchronization on Subsequent Logins: If the user logs in again during the same day, the application synchronizes the local lists with the cloud-based lists. This synchronization process copies data from the cloud to the local lists, maintaining the latest health data in the application.

The initialization of the database is shown in Figure 4.12.

Data Reporting and List Management in Android Application

Spo2 and Heart Rate Data Management:

- `todaySpo2AndHeartRateList` stores heart rate and blood oxygen (SpO2) data.
- The application updates this list locally after every 20 data points.
- Subsequent synchronization with the cloud ensures up-to-date cloud storage.

Human Activity Recognition Data Management:

- `todayActionRealtimeList` contains real-time human activity recognition results.
- It follows a similar update process: local updating and cloud synchronization after every 20 entries.

Activity Duration Data Management:

- `todayActionList` includes the duration of each activity.
- Updates occur when the user exits the activity tracking page, reflecting total activity duration.
- Both local and cloud lists are updated during each session.

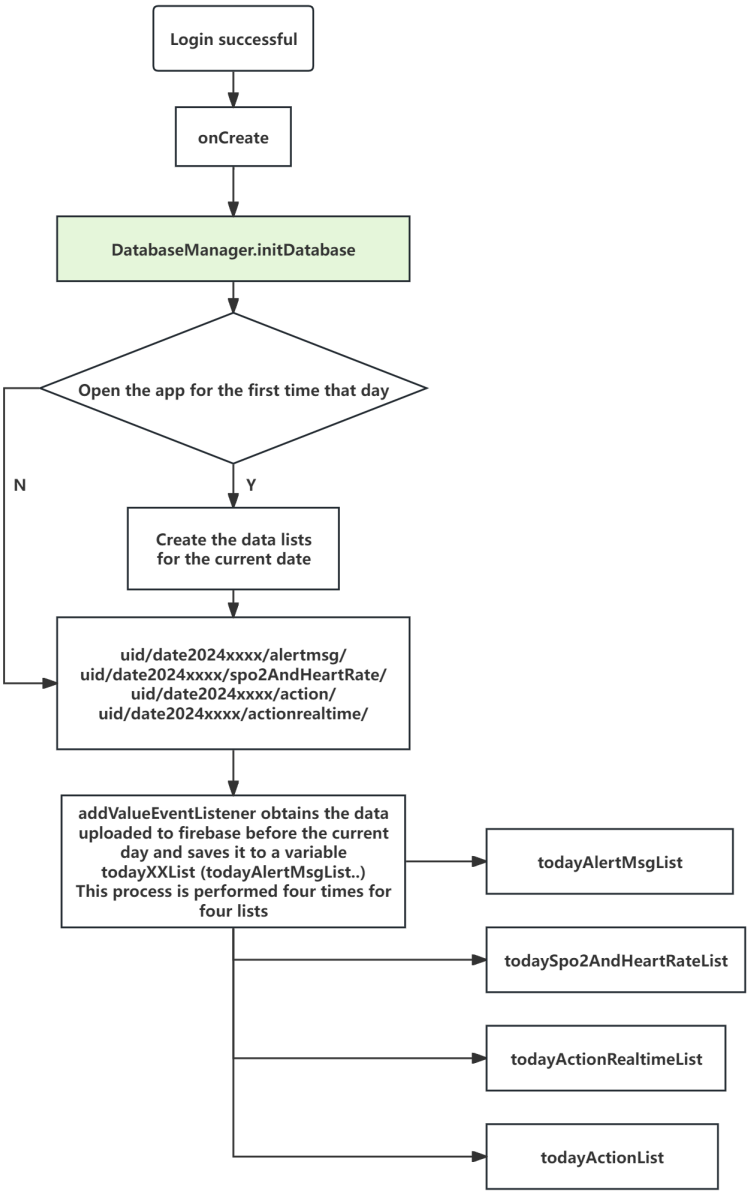

Figure 4.12: The initialization of the database

Alert Message Data Management:

- `todayAlertMsgList` stores alert messages, typically generated for abnormal heart rate or SpO2 levels.
- Each new alert prompts an immediate update to both the local and cloud lists.

The uploading of the data is shown in Figure 4.13 and 4.14.

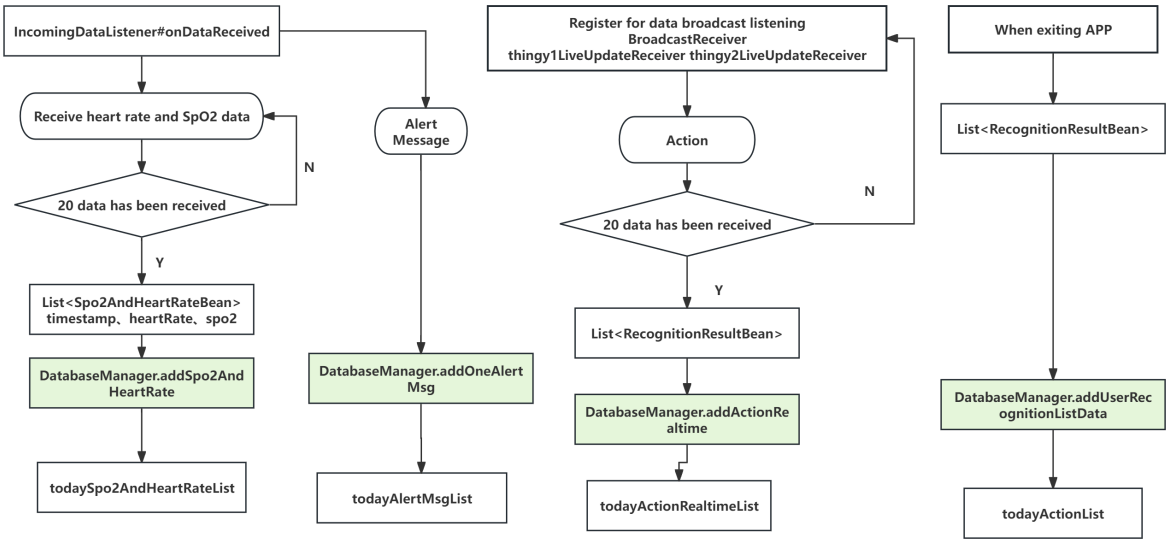

Figure 4.13: The uploading of data from sensor to app

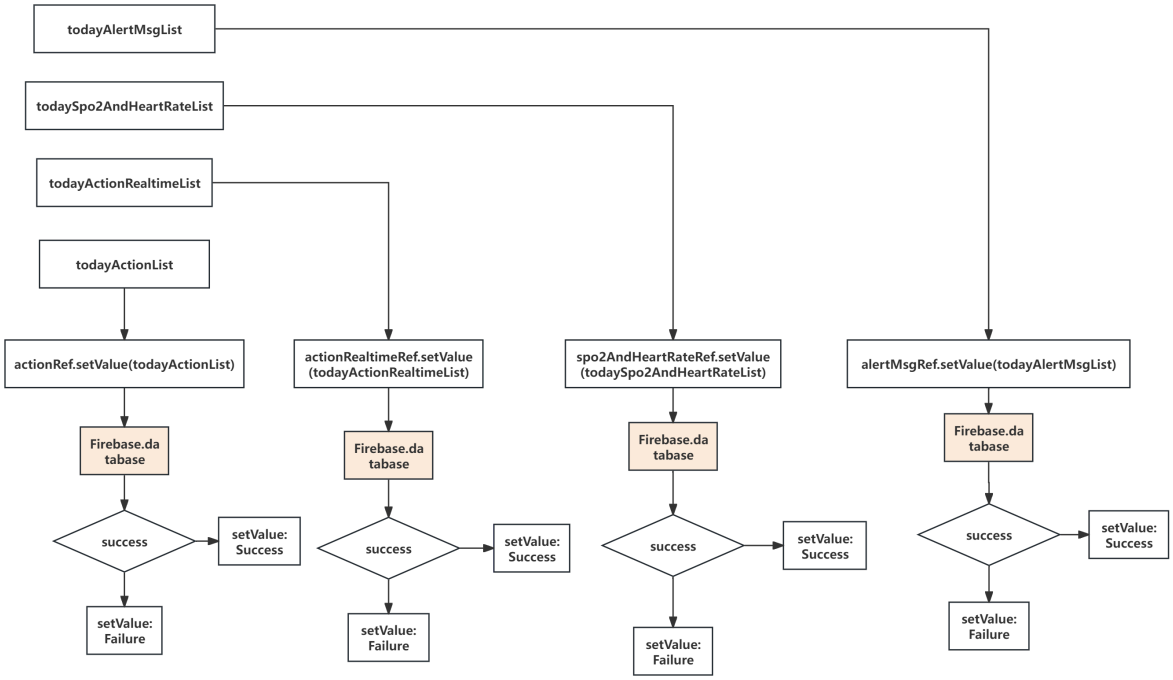

Figure 4.14: The uploading of data from app to cloud

4.2.5 Doctor App Main Interface Design

After successful registration and login, the Doctor App transitions to its main interface, designed to facilitate easy access and analysis of patient health data.

Patient List Display:

- The main interface features a list of user emails, each representing a different patient, allowing doctors to select and view specific health data.
- Selection of a patient's email opens detailed health information related to that user.

Data Visualization Pop-up Windows: Upon clicking a patient's email, the app provides four types of health data visualizations:

1. **Heart Rate and SpO2 Over Time:** Displayed as a scatter plot with heart rate in green dots and SpO2 levels in red dots.
2. **Activity Over Time:** Activity intensity and type are shown in a scatter plot, using different values and colors for various activities.
3. **Duration of Each Activity:** A summary view showing the time duration for various activities in bar chart.
4. **Combined Heart Rate, SpO2, and Activity:** This view correlates all data for comprehensive analysis.

Scatter Plot Representations:

- Scatter plots for heart rate and SpO2 provide visual insights into the patient's vitals over time.
- Activity data is differentiated by intensity values and colors, enabling easy identification of various activities, such as:
  - Sitting (various postures), Standing, Lying down: Represented with specific values like 2.0f for sitting and 1.0f for lying down.
  - Walking: 4.0f, Running: 6.0f, Stair activities: 5.0f.
  - Desk work: 2.0f, General movement: 4.0f.

Calendar-Based Data Selection: In the Doctor App, each data visualization page includes an interactive calendar interface. This feature allows doctors to select specific dates, enabling them to view and analyze patient health data corresponding to those dates. This calendar functionality not only enhances the user experience by providing an intuitive method for navigating historical health data but also proves essential in clinical settings for effective patient health management and progress tracking.

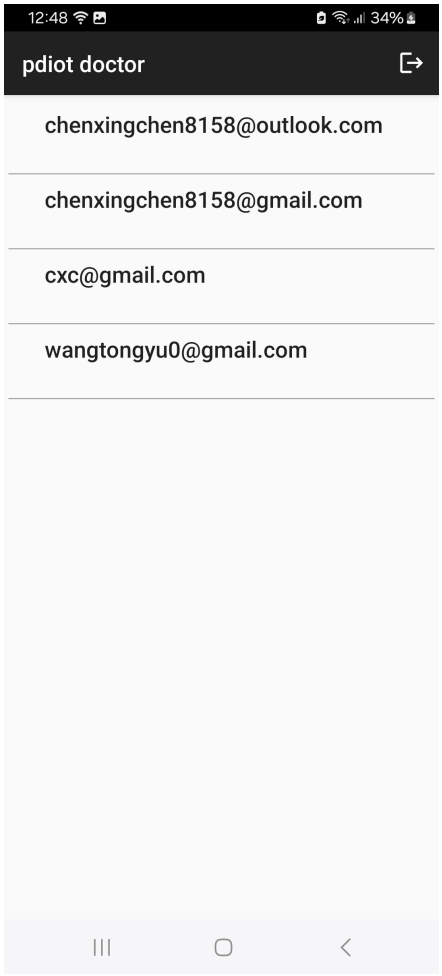

Figure 4.15: Home page

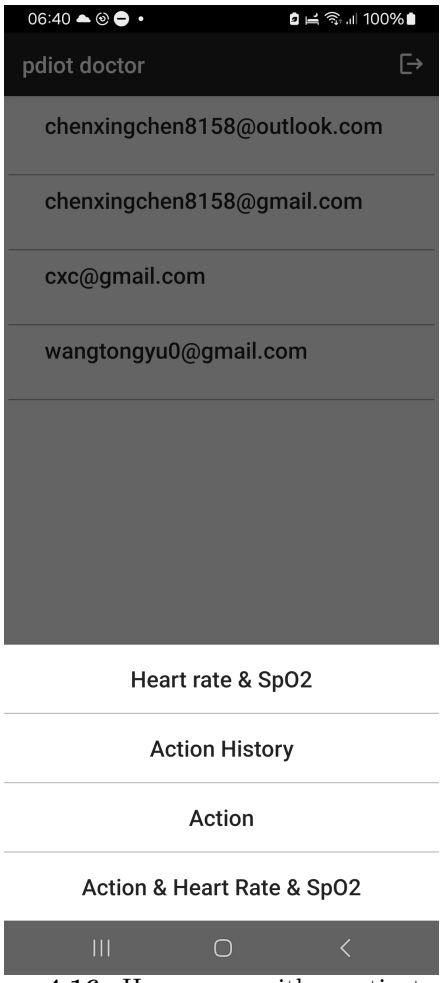

Figure 4.16: Home page with a patient selected

This design approach ensures that healthcare professionals can efficiently access, interpret, and analyze patient data through the Doctor App. The scatter plots and color-coded activities facilitate quick understanding and effective clinical decision-making.

Figure4.15 and Figure4.16 shows the home page of the doctor app.
Figure4.17 to Figure4.20 show the data visualizayion in the doctor application.

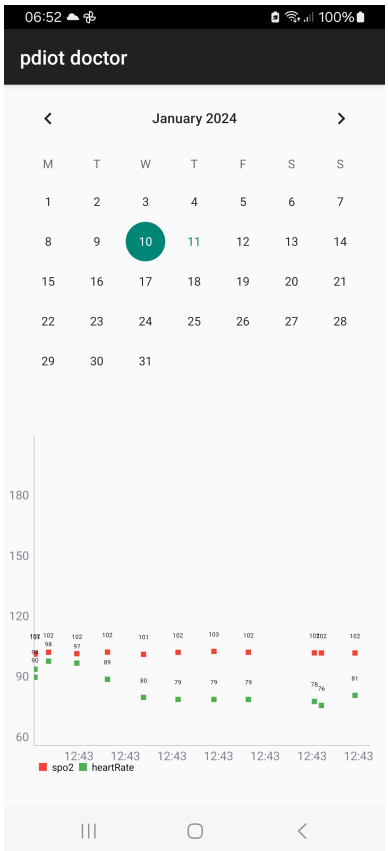

Figure 4.17: Heart rate + SpO2

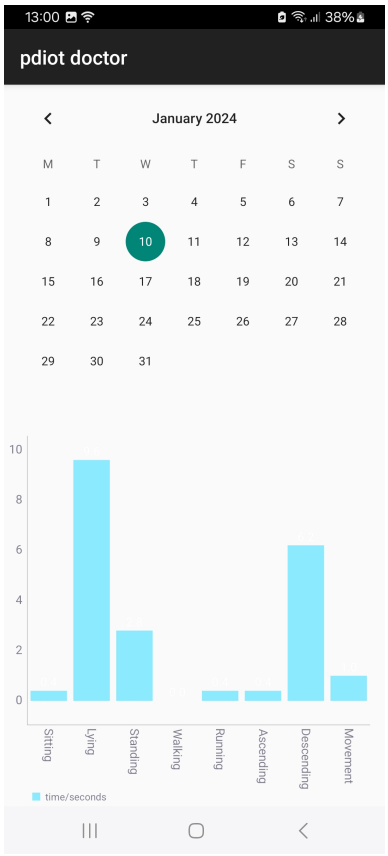

Figure 4.18: Action duration history

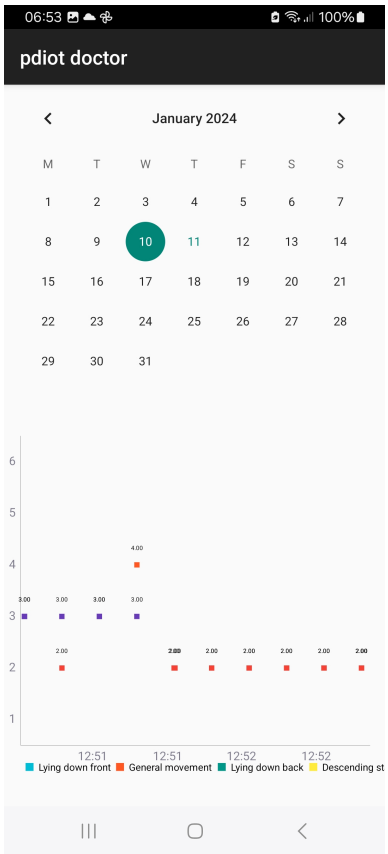

Figure 4.19: Action

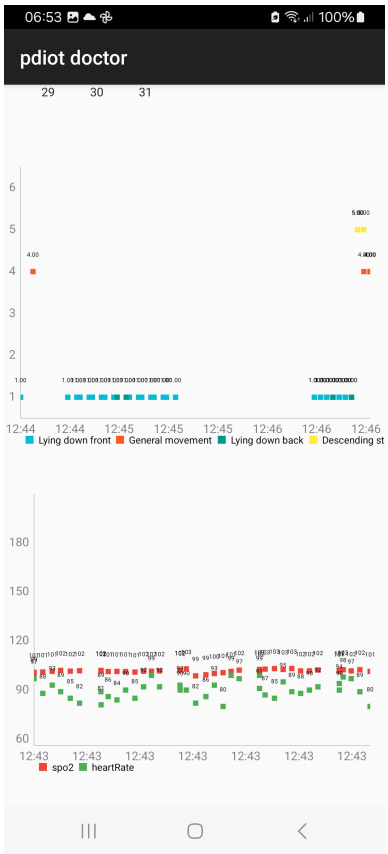

Figure 4.20: Heart rate + SpO2 + Action

4.2.6 Implementation of the Doctor App

Home Page

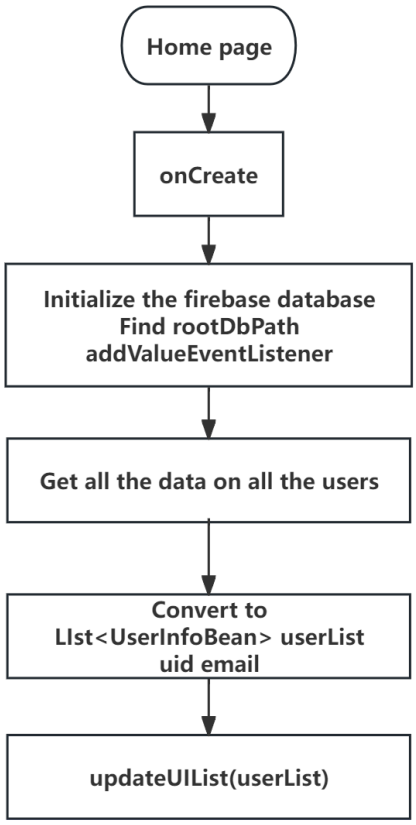

Figure 4.21: Initialization

The ‘MainActivity‘ class in the Doctor App of an Android application showcases the integration with Firebase Database, user interface interactions, and data management for patient information.

Firebase Database Integration:

- Integrates with Firebase Realtime Database to retrieve user information.
- Implements `ValueEventListener` to listen for data changes and update the user interface.

User List Management:

- Maintains a list of `UserInfoBean` to store and display patient information in a `ListView`.

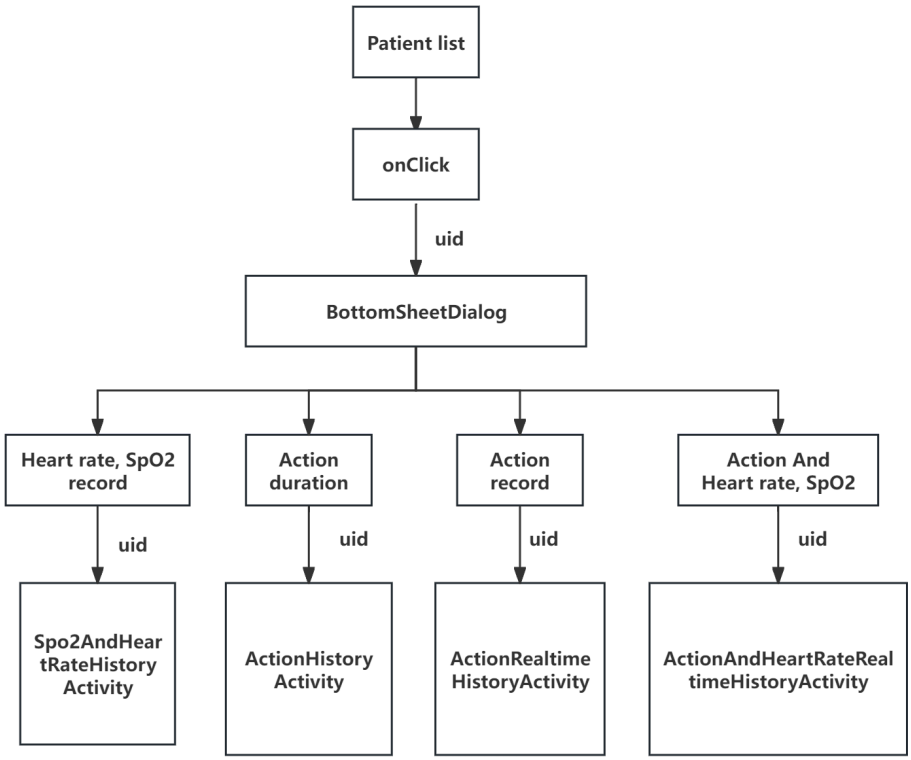

Figure 4.22: Home page

UI Interaction and Data Visualization:

- Enables clickable items in the `ListView` that open a `BottomSheetDialog` for detailed patient data views.
- Launches activities for Spo2, heart rate history, and activity data based on user selection.

Menu and Logout Functionality:

- Features an options menu with a logout functionality, which includes a confirmation dialog and Firebase sign-out.

Dynamic UI Updates:

- Dynamically updates the patient list in the UI using the `updateUIList` function.
- Custom views for each list item display the patient's email.

Figure4.21 shows the initialization of the doctor application. Figure4.22 shows the home page structure of the doctor application.

Patient Data Visualization

The doctor's app for the health-tracking system incorporates three pivotal activities - `Spo2AndHeartRateHistoryActivity`, `ActionHistoryActivity`, and `ActionRealtimeHistoryActivity` - to display comprehensive patient data.

Spo2AndHeartRateHistoryActivity

- Displays SpO2 and heart rate data history.
- Includes a calendar for selecting specific dates.
- Uses a scatter chart with distinct colors for SpO2 and heart rate.
- Retrieves and visualizes data from Firebase based on selected dates.

ActionHistoryActivity

- Focuses on the duration of various patient activities.
- Dynamically loads `HistoryFragment` for historical data display.
- Uses patient's UID to fetch specific data from Firebase.

ActionRealtimeHistoryActivity

- Presents a real-time view of patient activities.
- Incorporates a calendar view and a scatter chart for data visualization.
- Different colors and heights on the chart represent various activities.
- Fetches and categorizes data from Firebase.

The structures for these functions are shown in Figure4.23, Figure4.24 and Figure4.25.
Integrated Workflow:

- **Data Fetching and Synchronization:** Utilizes Firebase for personalized data retrieval based on patient UID.
- **User Interface and Interaction:** Maintains UI consistency with calendar views and scatter charts across different data views.
- **Custom Data Visualization:** Each activity provides unique insights through customized visual representations.
- **Modular and Interconnected Design:** Offers focused data analysis while ensuring a cohesive user experience.

These activities collectively enhance the doctor's ability to analyze and interpret health data, facilitating informed medical decision-making.

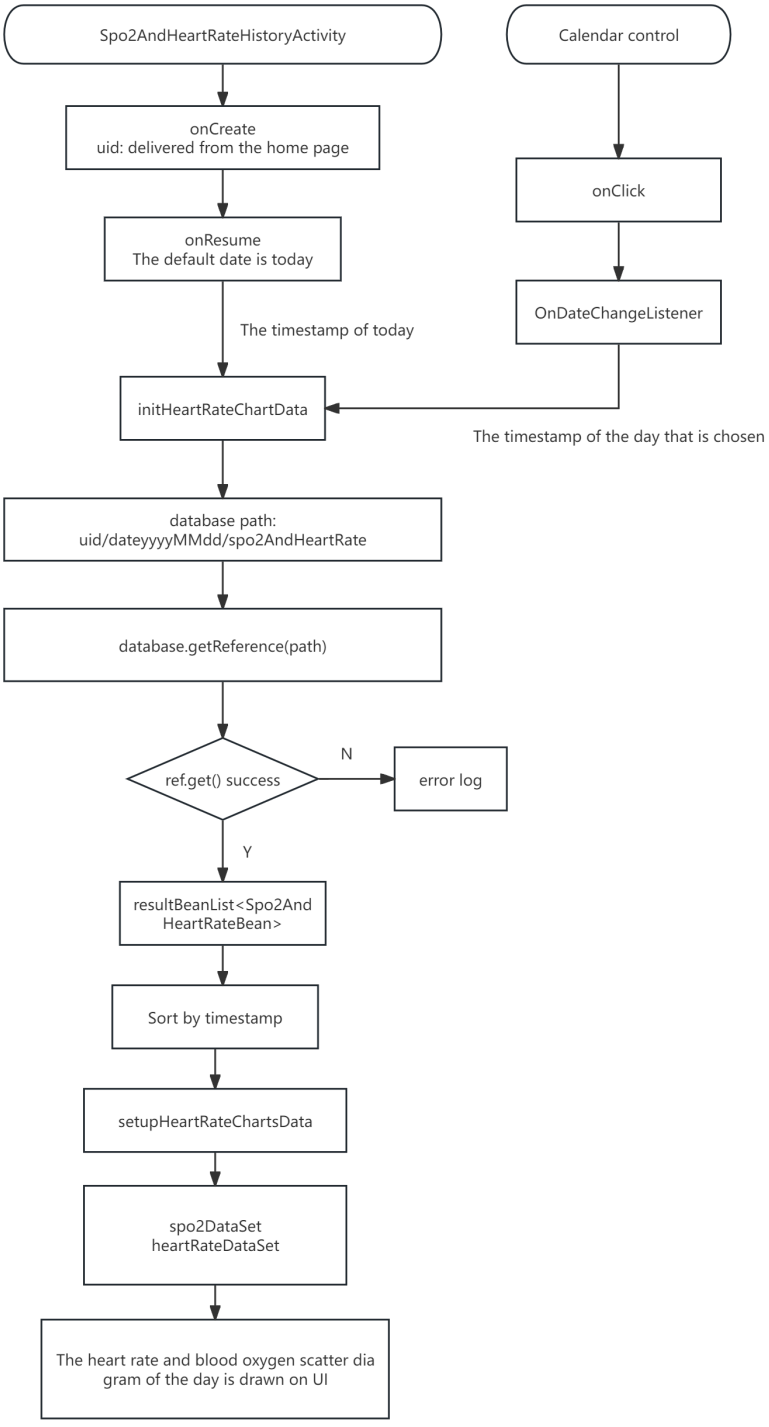

Figure 4.23: Structure for Spo2 And HeartRate History

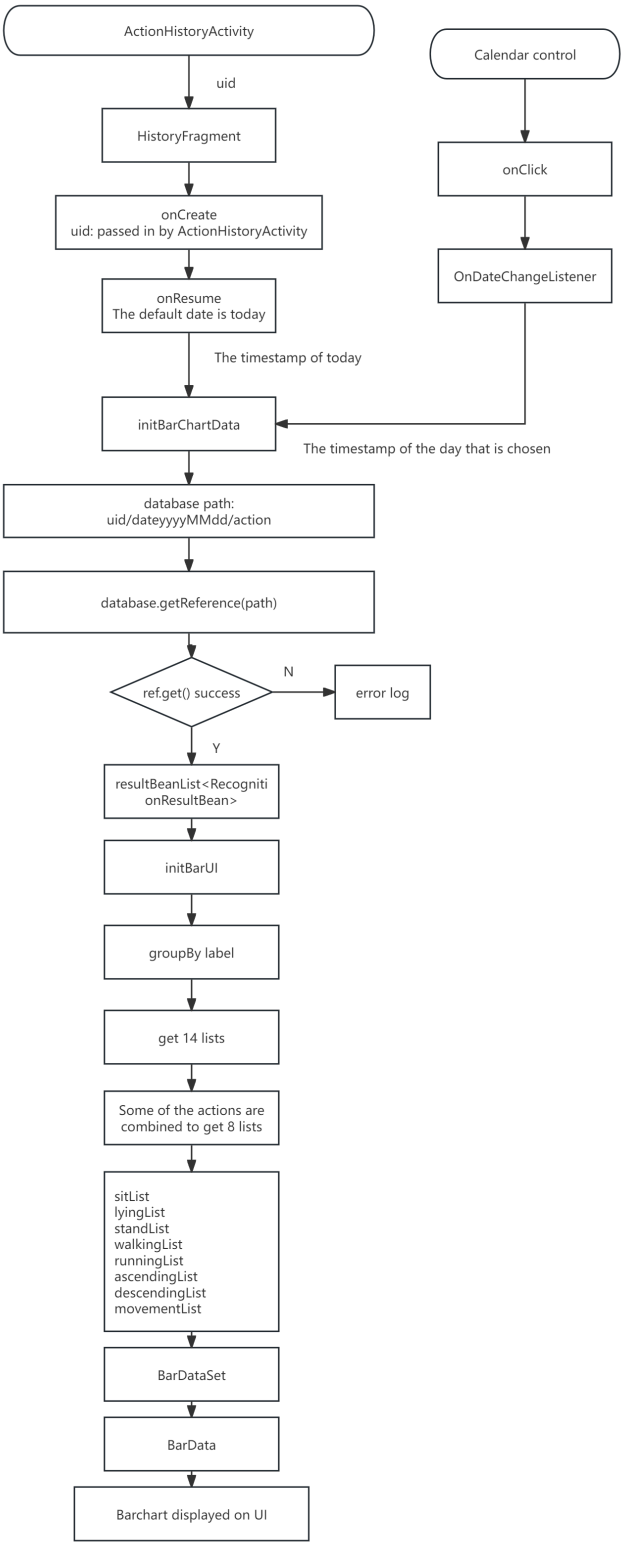

Figure 4.24: Structure for Action History

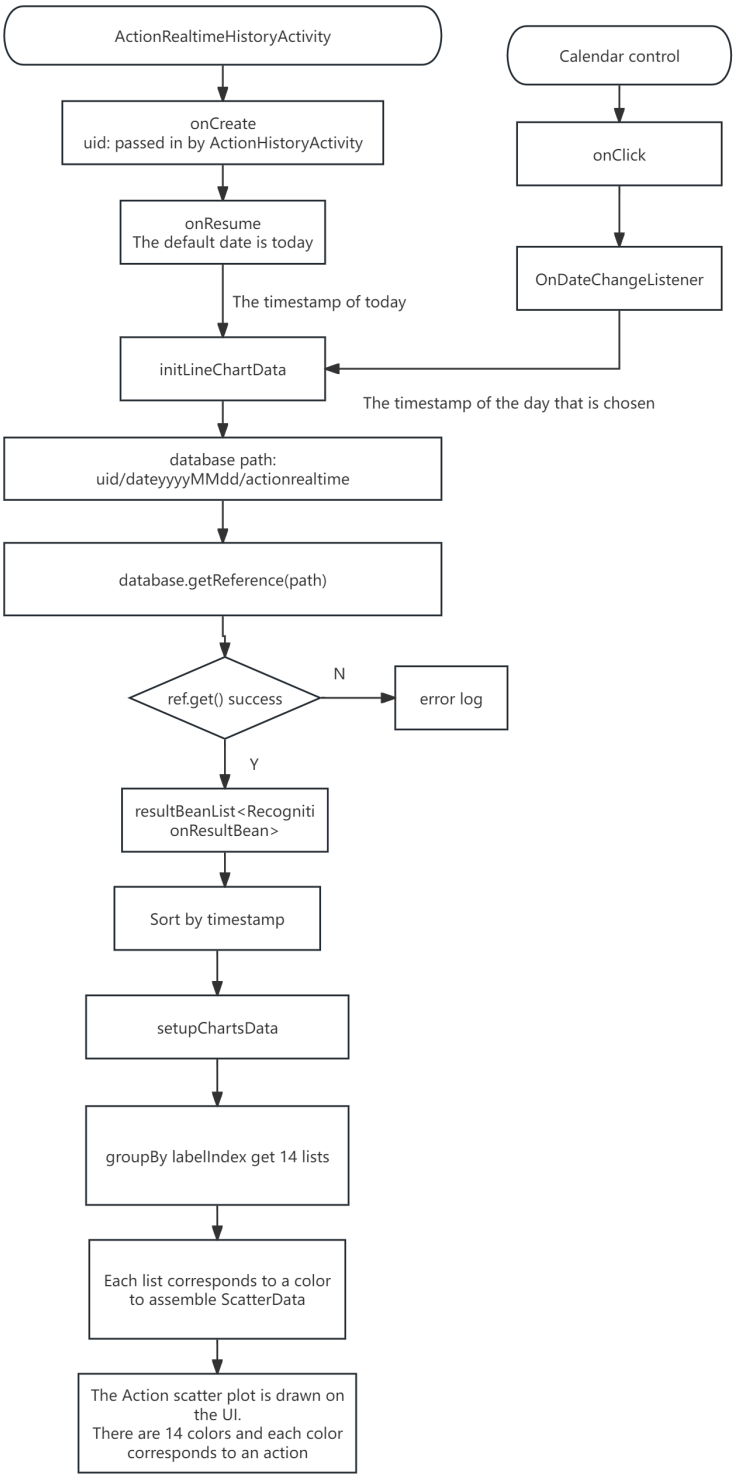

Figure 4.25: Structure for Action History by time

4.3 Machine Learning Acceleration

In the dynamic field of machine learning, the acceleration of algorithms has become a focal point, pivotal for advancing the capabilities and efficiency of models in handling vast and complex datasets. The crux of accelerating these algorithms often boils down to optimizing core computational processes, among which matrix multiplication stands out as especially critical. This fundamental operation is not just a mere step in the algorithmic process but rather the backbone of numerous machine learning applications, from neural network training to large-scale data analytics. The efficiency of matrix multiplication directly influences machine learning models' overall speed and scalability, making its optimization a key to unlocking faster and more efficient learning processes.

However, optimizing matrix multiplication presents its own set of challenges and necessities. As the scale of data and the complexity of models continue to grow, traditional matrix multiplication methods begin to falter, becoming a bottleneck that hinders further advancements. The need to address this bottleneck is not just about improving an individual operation but about unleashing machine learning algorithms' full potential.

In addressing these challenges, I mainly tried 3 innovative approaches: the utilization of multi-threading techniques, the implementation of the Strassen algorithm and the Eigen library. Multi-threading leverages the power of modern multi-core processors to conduct parallel computations, drastically reducing the time required for matrix operations in large-scale scenarios. This method aligns well with the demands of contemporary machine learning tasks, where efficient distribution and processing of computations can significantly boost algorithmic performance.

Conversely, the Strassen algorithm represents a paradigm shift in matrix multiplication. By employing a divide-and-conquer strategy, it reduces the computational complexity compared to conventional methods. While it introduces a trade-off in terms of numerical precision, the Strassen algorithm opens up new avenues for speeding up matrix multiplication, thereby contributing to the acceleration of machine learning algorithms.

4.3.1 Multi-threading Method

Multi-threading is a computational technique that allows a program to execute multiple threads or sequences of instructions concurrently. In the context of modern computing, where processors often have multiple cores, multi-threading enables these cores to run different threads simultaneously, thereby enhancing the overall computational efficiency. This approach is particularly relevant in tasks that can be parallelized, where different segments of a task can be executed simultaneously without waiting for other segments to complete.

In the realm of machine learning, and specifically in matrix multiplication, multi-threading can play a pivotal role. Matrix multiplication, a core operation in many machine learning algorithms, often involves large datasets and complex calculations that can be computationally intensive. By applying multi-threading to matrix multiplication, the task can be divided and processed across multiple threads. Each thread handles a part of the matrix operation, allowing for simultaneous computations. This division not only accelerates the processing time but also optimizes the use of available computational resources.

For instance, when multiplying two large matrices, the operation can be split into smaller tasks where subsets of the matrices are multiplied in separate threads. This parallel processing approach significantly reduces the time required for large-scale matrix operations, a common scenario in machine learning tasks like training neural networks or processing large datasets. The efficiency gains from multi-threading in matrix multiplication are thus a critical factor in accelerating machine learning algorithms, enabling them to handle larger datasets and more complex models with greater speed and efficiency.

Implementation of Multi-threading for Matrix Multiplication

The implementation of multi-threading in matrix multiplication was achieved using C++ with standard threading libraries. The central component of this approach is the 'ThreadPool' class, which manages a pool of worker threads to perform tasks concurrently. This class allows tasks to be enqueued and executed by available threads, efficiently utilizing the multi-core capabilities of modern processors.

ThreadPool Class

The 'ThreadPool' class is designed to create a specified number of threads upon instantiation. These threads wait for tasks to be queued. Upon receiving a task, a thread executes it and then returns to a waiting state. The class ensures proper synchronization using mutexes and condition variables, handling the dynamic allocation of tasks to threads and safely managing concurrent access to resources.

Key Components:

- **Constructor** (`ThreadPool(int numThreads)`): Initializes the pool with a specified number of worker threads.
- **Destructor** (`~ThreadPool()`): Signals all threads to stop and waits for them to finish.
- **enqueue Method**: Allows adding tasks (functions) to the task queue.

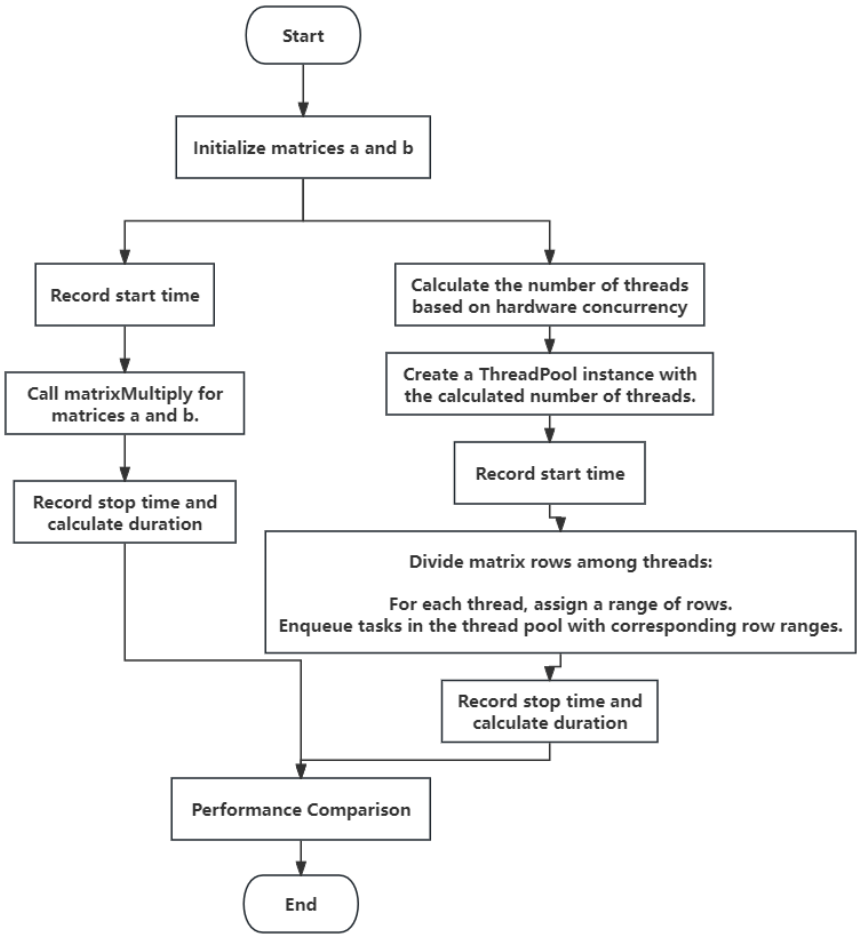

Figure 4.26: Block diagram for Multi-threading code for Matrix Multiplication

Matrix Multiplication Functions

Two matrix multiplication functions were implemented: ‘matrixMultiply‘ for the sequential approach and ‘matrixMultiplyParallel‘ for the multi-threaded approach.

The ‘matrixMultiply‘ function follows the standard algorithm for matrix multiplication, iterating through rows and columns to compute the product.

In contrast, ‘matrixMultiplyParallel‘ utilizes the ‘ThreadPool‘ to divide the matrix multiplication task into smaller sub-tasks, each processed by a different thread. The matrix is divided row-wise among the available threads, with each thread computing a portion of the result matrix. This division enables parallel computation, significantly reducing the time required for large matrix multiplications.

matrixMultiply: Performs sequential matrix multiplication.
matrixMultiplyWorker: A worker function for parallel matrix multiplication, processing a portion of the matrix.
matrixMultiplyParallel: Organizes parallel matrix multiplication using the thread pool, dividing the workload among multiple threads.

Performance Measurement

The performance of both sequential and parallel matrix multiplication was measured using high-resolution clocks. The code was executed multiple times to calculate the average duration for both methods. These measurements highlighted the efficiency gain achieved through multi-threading, demonstrating a substantial reduction in computation time for the parallel approach compared to the sequential one. Finally, the program outputs the timing details for both sequential and parallel multiplication and the computed performance improvement.

The block diagram of the Multi-threading method is shown in Figure 4.26.

4.3.2 The Strassen Algorithm Approach

The Strassen Algorithm[57], introduced by Volker Strassen in 1969, revolutionized the way matrix multiplication is performed, particularly impacting the field of computational complexity theory. This algorithm deviates from the standard approach of matrix multiplication, which involves a straightforward yet computationally intensive process of element-wise multiplication and summation. The standard method, also known as the naïve algorithm, has a computational complexity of $O(n^3)$ for multiplying two $n \times n$ matrices.
In contrast, the Strassen Algorithm employs a divide-and-conquer strategy, breaking down the large matrix multiplication problem into smaller sub-problems. While this might seem a modest improvement at first glance, it significantly reduces the computational complexity to approximately $O(n^{2.81})$. This reduction is achieved through the use of additional additions and subtractions, which are computationally less expensive than multiplications.

Details of the Strassen Algorithm

The Strassen Algorithm involves recursively dividing each matrix into four submatrices until the base case (typically when the matrices are sufficiently small) is reached. At each level of recursion, instead of performing eight matrix multiplications as in the standard approach, the Strassen Algorithm performs seven multiplications on these submatrices, along with several additions and subtractions. These operations form the core of the Strassen method, reducing the overall number of multiplication operations.

The key steps in the Strassen Algorithm include computing intermediate matrices from the sum and difference of submatrices, performing seven multiplications on these intermediate matrices, and then combining the results to form the final product matrix.

The Strassen algorithm operates by partitioning the input matrices A and B into equally sized block matrices, which allows the multiplication to be performed in a divide-and-conquer manner. The partitioning of a matrix A would look like the following:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Where each A_{ij}, B_{ij}, and C_{ij} are $\frac{n}{2} \times \frac{n}{2}$ matrices. The naive approach to matrix multiplication would compute the product matrix C as:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

This standard approach requires eight multiplications of matrix blocks to compute the C_{ij} matrices.
However, Strassen's algorithm reduces the number of multiplications by introducing new matrix combinations M_k that can be computed using only seven multiplications:

$$
\begin{aligned}
M_1 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22}) \cdot B_{11} \\
M_3 &= A_{11} \cdot (B_{12} - B_{22}) \\
M_4 &= A_{22} \cdot (B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12}) \cdot B_{22} \\
M_6 &= (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\end{aligned}
$$

The product matrix C can then be expressed in terms of the M_k matrices as:

$$
\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{bmatrix}
$$

This process continues recursively until the submatrices cannot be divided further, which is typically when they reduce to single elements, at which point the standard multiplication is applied.
If the original matrices A and B were not of size $2^n \times 2^n$, the matrices can be expanded by adding zero rows and columns to meet this condition. These added zeros will not affect the multiplication but will be removed from the final product matrix C to obtain the result of the original matrix size.

Computational Complexity Analysis using the Master Theorem

To analyze the computational complexity of the Strassen Algorithm, we can apply the Master Theorem[58]. The Master Theorem provides a method to determine the running time of divide-and-conquer algorithms in terms of their recursive occurrences. In the case of the Strassen Algorithm, the recurrence relation can be expressed as:

$$
T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)
$$

Here, $T(n)$ represents the time complexity of multiplying two $n \times n$ matrices, $7T\left(\frac{n}{2}\right)$ accounts for the seven multiplications of submatrices of size $\frac{n}{2} \times \frac{n}{2}$, and $O(n^2)$ represents the time complexity for the additional additions and subtractions.
Applying the Master Theorem, we find that the Strassen Algorithm falls into Case 1, where the critical exponent is less than the logarithm base 2 of 7 (i.e., $\log_2 7 \approx 2.81$). Thus, the time complexity of the Strassen Algorithm is:

$$
O(n^{\log_2 7}) \approx O(n^{2.81})
$$

This computational complexity is a significant improvement over the $O(n^3)$ complexity of the standard matrix multiplication method, particularly for large matrices. While the practical implementation of the

Strassen Algorithm also considers factors like the overhead of recursion and the threshold at which it becomes more efficient than the standard method, its theoretical efficiency has made it a cornerstone in the study of fast matrix multiplication algorithms.

Code Implementation of the Strassen Algorithm

The block diagram of this function is shown in Figure 4.27 and Figure 4.28.

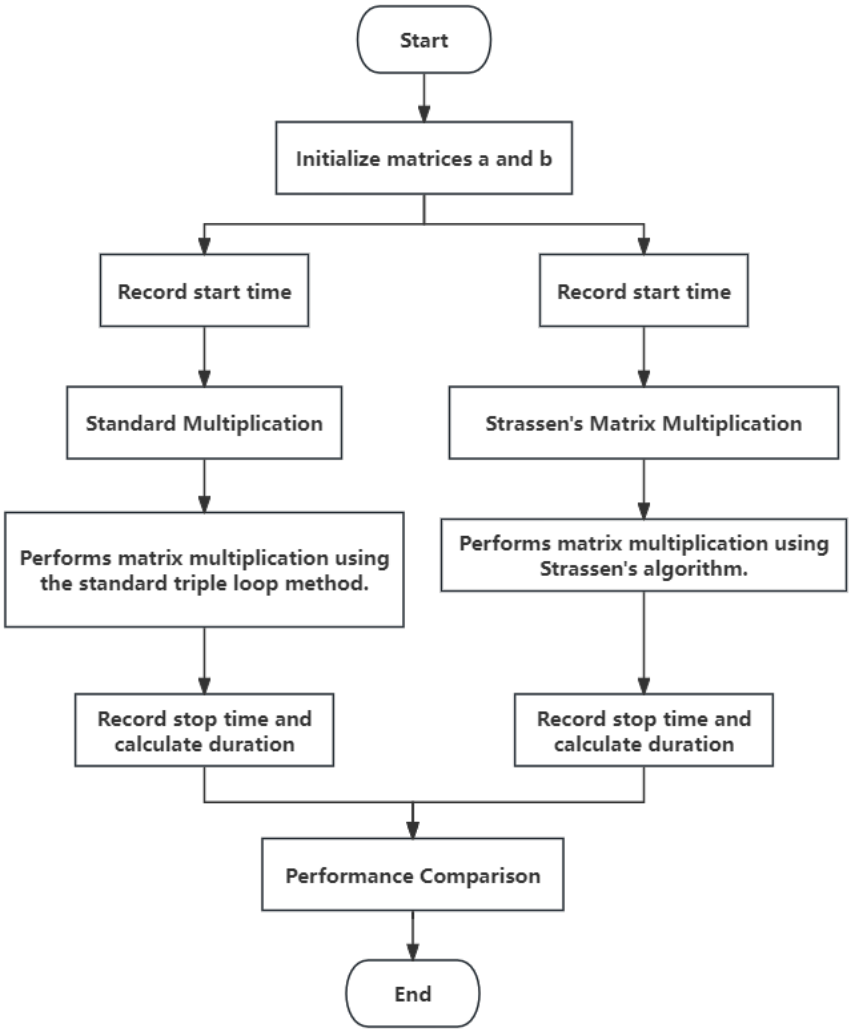

Figure 4.27: Block diagram for Strassen Algorithm for Matrix Multiplication

This C++ program demonstrates two methods of matrix multiplication: the standard approach and Strassen's algorithm. It allows users to compare the performance of these two methods by calculating the time taken to multiply two randomly generated matrices of a specified size. The key components of

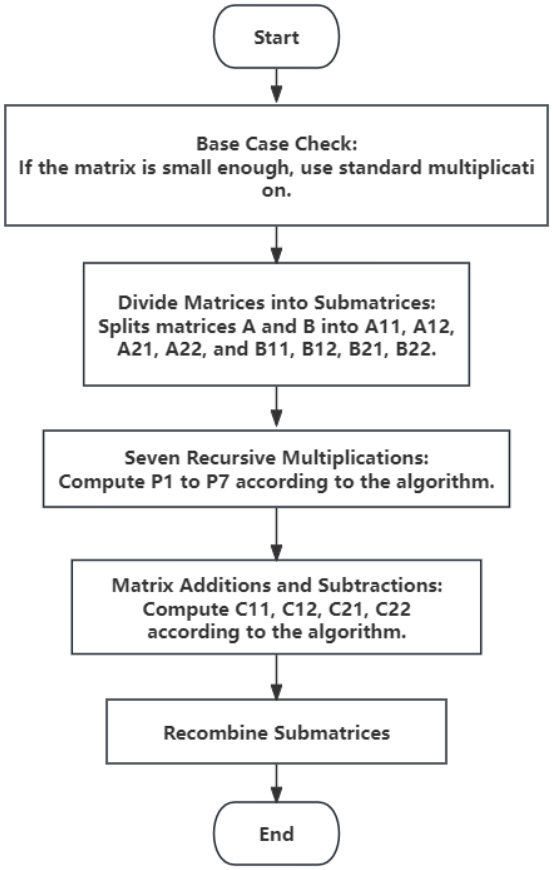

Figure 4.28: Block diagram for Strassen Algorithm

the program include matrix generation, standard multiplication, Strassen's multiplication, and performance measurement.

The program commences by prompting the user to input the size for the matrices. Utilizing a random number generator, it then generates two matrices filled with random integers ranging from 1 to 10.

Then the program will do the matrix multiplication calculation using `standardMultiply` and `strassenMultiply`

Post-computation, the program measures and compares the execution times of both methods using a high-resolution clock. These times are presented in microseconds, offering a direct comparison of the efficiency between the standard and Strassen's matrix multiplication algorithms.

4.3.3 Enhancing Matrix Multiplication with Eigen and Strassen's Algorithm

Eigen[59] is a high-level C++ library renowned for its efficiency in linear algebra, matrix operations, and numerical computations. Its optimizations, such as expression templates and loop unrolling, enable performance comparable to hand-written code. Eigen supports various matrix sizes and types, from small to large dense matrices. Its user-friendly API facilitates elegant, type-safe code, reducing bugs and maintenance.

Motivation for Combining Eigen with Strassen's Algorithm

While Eigen provides highly optimized routines for matrix operations, there is always a quest for pushing the boundaries of computational efficiency. The Strassen algorithm, with its lower asymptotic complexity for matrix multiplication, presents an opportunity for further optimization. When dealing with large matrices, the difference in complexity — $O(n^3)$ for the standard method versus approximately $O(n^{2.81})$ for the Strassen algorithm — can lead to non-trivial performance improvements.
The motivation for integrating Eigen with the Strassen algorithm stems from the desire to leverage the best of both worlds: the sophisticated optimizations of Eigen and the theoretically faster matrix multiplication of the Strassen algorithm. While Eigen's internal optimizations are highly effective for small to medium-sized matrices, the Strassen algorithm's approach can outpace the traditional methods as the matrix size grows.
Combining Eigen with Strassen's algorithm involves using Eigen for the base cases of multiplication within the Strassen recursion. This hybrid approach hypothesizes that for matrices beyond a certain size, the Strassen algorithm can reduce the number of recursive multiplications needed, while Eigen can perform these multiplications swiftly and accurately. The combination is expected to be particularly effective for matrices that are not power-of-two sizes, where Eigen's ability to handle arbitrary sizes with minimal overhead can be advantageous.

Code Implementation of Matrix Multiplication Using Strassen's Algorithm and Eigen Library

The block diagram is shown in Figure 4.29.

Main Function Workflow In the `main` function, the program sets off by defining a predetermined size for the matrices to be multiplied. It then generates two matrices of this defined size, filled with random values, utilizing Eigen's functionality. Following this, the program embarks on performing matrix multiplication through both the Strassen-Eigen hybrid and Eigen's standard method.
For each methodology, the program meticulously measures the execution time utilizing the high-resolution clock from the `<chrono>` library. It initiates a timer prior to the multiplication and halts it immediately after, calculating the elapsed time in milliseconds. This time measurement is instrumental in comparing the performance of the two distinct methods.

Performance Comparison and Result Validation Post the execution time calculations for both methods, the program computes the percentage improvement of the Strassen-Eigen method over Eigen's standard method. This computation provides a quantitative measure of the efficiency gain attributed to Strassen's algorithm in combination with Eigen.
Moreover, the program ascertains the accuracy of the Strassen-Eigen method by contrasting its outcome with that of Eigen's standard method. It verifies if the resulting matrices from the two methodologies align within a marginal threshold, accommodating any slight numerical variances. This step is crucial to ensure the Strassen-Eigen method's proficiency in not just enhancing performance but also in maintaining computational precision.

4.4 Summary of Design and Implementation

This chapter focuses on the design and implementation aspects of the project, divided into two primary areas: the development of mobile applications and the acceleration of machine learning processes.

In app development, key features include Firebase Authentication for secure user access, integration of Firebase Realtime Database for efficient data management, and the development of the Doctor app, tailored for healthcare professionals. The Patient app also undergoes enhancements for user interaction and data management.

For machine learning acceleration, the project explores Strassen's algorithm for efficient matrix multiplication and the use of thread pooling to facilitate multithreaded operations. These approaches aim to improve the performance and speed of machine learning algorithms.

The Android Application Development section delves into user authorization, Firebase data storage structure, and the DatabaseManager implementation. It emphasizes user-friendly interfaces, streamlined authentication processes, and effective data synchronization and management.

The Doctor App's main interface design is outlined, focusing on ease of access to patient data and effective data visualization techniques. It integrates with Firebase for realtime data retrieval and offers a comprehensive view of patient's health data through various visual formats.

The chapter concludes with a detailed discussion of machine learning acceleration techniques. It covers implementing multi-threading for matrix multiplication and integrating the Strassen algorithm with the Eigen library, highlighting their efficiency in computational processes.

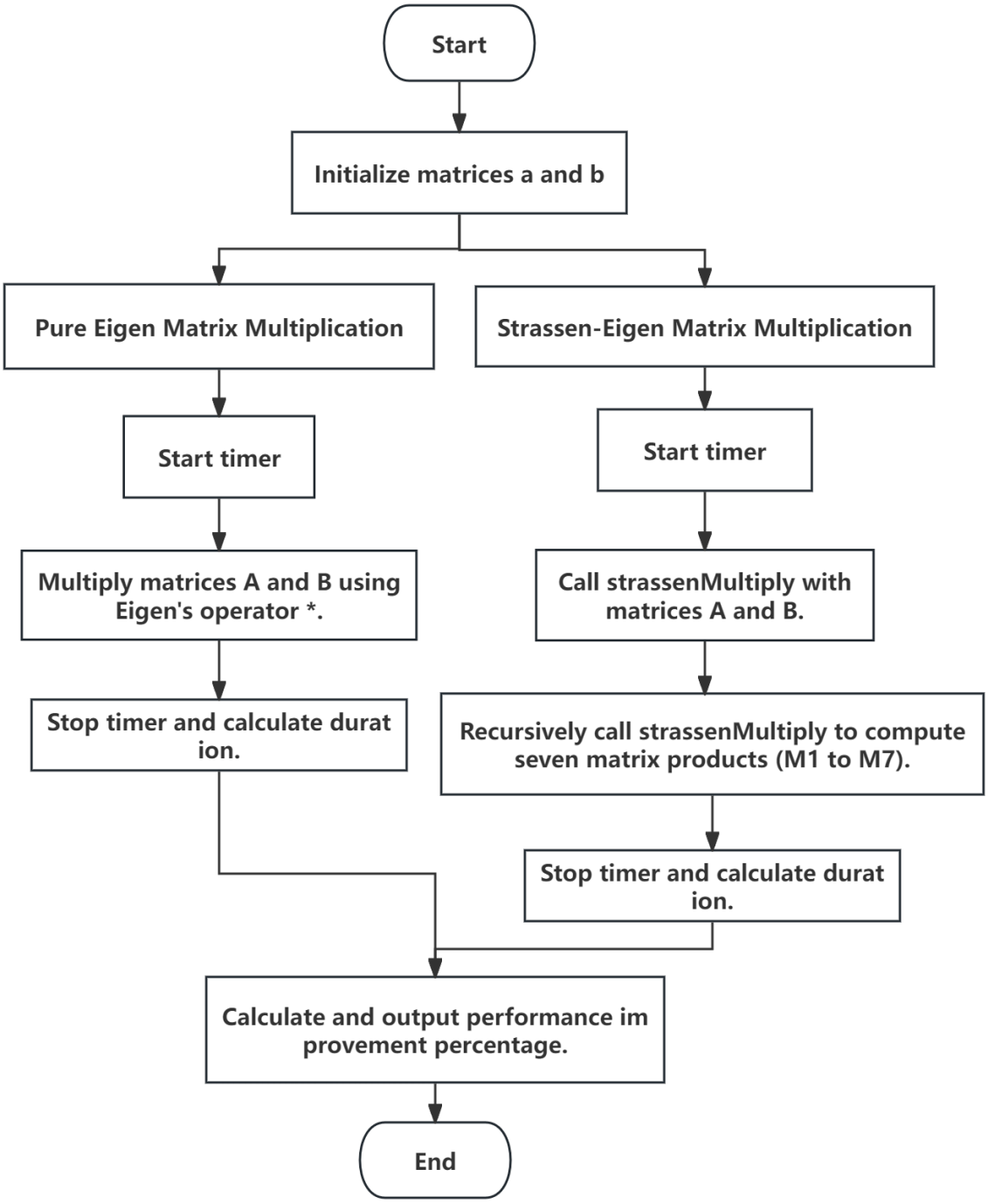

Figure 4.29: Block diagram for Strassen Algorithm with Eigen

Chapter 5

Testing and Discussion

5.1 Testing for Application

The testing phase is conducted to ensure the functionalities of all parts of the system.

5.1.1 Testing User Authentication

Test Cases

- **Registration Process:**
  - Test Case 1: Successful account creation.
  - Test Case 2: Registration with an already used email.
- **Login Process:**
  - Test Case 3: Login with correct credentials.
  - Test Case 4: Incorrect password entry.
  - Test Case 5: Google One Tap sign-in functionality.
- **Logout Process:**
  - Test Case 6: Successful logout.

Results

All tests passed successfully.

5.1.2 Testing Database Integration

Test Cases

- Test Case 1: Storing user health data.

- Test Case 2: Retrieving Heart rate and SpO2 data for display.
- Test Case 3: Retrieving Action historical data.
- Test Case 4: Updating existing records.
- Test Case 5: Error handling for data update failures.
- Test Case 6: Real-time data synchronization across devices.

Results

All tests passed successfully.

5.1.3 Testing Doctor App Display Functionality

Test Cases

- Test Case 1: Clarity and readability of user interface elements.
- Test Case 2: Navigation and user interaction flow.
- Test Case 3: Accuracy of data representation in graphs and charts.

Results

All tests passed successfully.

5.2 Performance Testing

5.2.1 Objective

The objective of the performance testing is to compare the execution times of the standard matrix multiplication method and the improved approaches. The tests are designed to identify the specific scenarios in which each multiplication strategy excels, as well as to subtly elucidate the nuanced advantages offered by the hybrid approach when juxtaposed with conventional techniques.

5.2.2 Test Environment

The performance testing was conducted using Visual Studio 2022 on an Alienware m15 Ryzen Edition R5. This system is equipped with an AMD Ryzen 9 5900HX processor with 16 CPUs, operating at approximately 3.3GHz, and has 16 GB of RAM. The operating system used was Windows 10 Home 64-bit.

Table 5.1: Comparison of Sequential and Parallel Matrix Multiplication Performance

	Sequential		Parallel	
Matrix Size	Avg. Time (ms)	Performance	Avg. Time (μs)	Performance
64	7874	-	10245.3	-30.116%
128	89667.3	-	33832.3	62.2691%
256	503793	-	103107	79.5338%
512	4.07391e+06	-	598269	85.3146%
1024	3.11788e+07	-	5.09379e+06	83.6627%

Table 5.2: Comparison of Naive Method vs. Strassen Algorithm for Matrix Multiplication

	Naive Method		Strassen Algorithm	
Matrix Size	Avg. Time (ms)	Performance	Avg. Time (μs)	Performance
64	7082	-	7790	-10%
128	88523	-	84097	5%
256	504874	-	398850	21%
512	1024000	-	662528	35.3%
1024	28192000	-	14462496	48.7%

Table 5.1 shows the comparison of computational times and performance improvements between sequential and parallel matrix multiplication methods across different matrix sizes. Table 5.2 displays the comparing results between the Naive method and the Strassen Algorithm for matrix multiplication. Table 5.3 outlines the comparison for the Eigen Library and the Eigen Library plus the Strassen Algorithm.

5.3 Discussion about experiment results

5.3.1 Multithreading Method

Observations

- **Small Matrices (Size 64)**: Parallel computation was less efficient than sequential, indicated by a -30.116% performance improvement. This is attributed to the overhead of thread management in parallel processing overshadowing the computation time for small matrices.

Table 5.3: Comparison of Eigen vs. Eigen Combined with Strassen Method

	Eigen (ms)		Strassen Algorithm+Eigen (ms)	
Matrix Size	Avg. Time	Improvement	Avg. Time	Improvement
32	0.8363	-	1.0883	-30%
64	6.6091	-	6.309	-5%
128	61.9153	-	80.4187	5%
256	476.74	-	391.461	20%
512	3204.87	-	2988.51	10%
1024	24311.2	-	18596.7	26%
2048	200153	-	123197	40%

- **Larger Matrices (Sizes 128, 256, 512, 1024)**: As the matrix size increased, parallel processing became significantly more efficient, with performance improvements ranging from 62.2691% to 85.3146%. This showcases the effectiveness of parallel computing in handling larger, more computation-intensive tasks.

Key Insights The observed results suggest that the efficiency of parallel processing is highly contingent on the size of the computation task. While overhead costs in multithreading impede its effectiveness for smaller tasks, its benefits become pronounced for larger-scale computations.

The analysis indicates that parallel processing, specifically for matrix multiplication, is highly beneficial for large matrix sizes, offering substantial performance improvements. It is crucial to consider the computational load and matrix size when choosing between sequential and parallel processing to optimize performance.

5.3.2 Strassen Algorithm

Observations

- For a **matrix size of 64**, the Naive Method is more efficient than the Strassen Algorithm. This is evident from the negative performance difference of -10%, suggesting that the Strassen Algorithm's overhead is not justifiable for small matrices.
- As the **matrix size increases** (128, 256, 512, 1024), the Strassen Algorithm begins to outperform the Naive Method. The performance improvement increases with the size, reaching up to 48.7% for a matrix size of 1024.

Key Insights

The analysis indicates that while the Strassen Algorithm is less efficient for smaller matrices due to its recursive overhead, it becomes increasingly beneficial for larger matrices. This shift in efficiency aligns with the theoretical expectations of the algorithm.

The choice between using the Naive Method and the Strassen Algorithm should be based on matrix size. For smaller matrices, the Naive Method might be more efficient, while for larger matrices, the Strassen Algorithm offers significant performance improvements. These results underscore the importance of empirical evaluation in selecting the most suitable algorithm for matrix multiplication tasks.

5.3.3 Strassen Algorithm with Eigen Library

The experimental results, as summarized in Table 5.3, provide insightful observations on the performance of the Eigen library alone versus its combination with the Strassen algorithm for matrix multiplication. The key findings from this comparison are discussed below:

Performance at Different Matrix Sizes The results demonstrate a clear trend: as the matrix size increases, combining the Strassen algorithm with Eigen tends to offer more significant performance improvements. For smaller matrices (sizes 32 and 64), the Strassen-Eigen method shows a decrease in performance, indicated by a negative improvement percentage. This can be attributed to the overhead associated with the recursive nature of the Strassen algorithm, which does not compensate for the performance gains in handling smaller matrices.

Increased Efficiency for Large Matrices For larger matrices, specifically from size 256 onwards, the Strassen-Eigen method outperforms the standard Eigen method. The improvement percentage increases with the matrix size, reaching up to 40% for a matrix of size 2048. This improvement suggests that the divide-and-conquer strategy of the Strassen algorithm becomes more effective as the matrix size grows, reducing the number of multiplication operations required and, hence, the overall computation time.

Optimal Use Cases Based on these results, the optimal use case for the Strassen-Eigen method appears to be for large matrix sizes, where the computational savings from the reduced complexity of the Strassen algorithm outweigh the overhead introduced by its recursive structure. The standard Eigen method is preferable for smaller matrices due to its optimized performance and lower overhead.

The experimental analysis reveals that while the Eigen library is highly efficient for standard matrix multiplication, integrating the Strassen algorithm can lead to substantial performance improvements for large matrices. This suggests a hybrid approach in practical applications, where the choice of algorithm is tailored based on the size of the matrix to achieve optimal performance.

5.4 Extended discussion

How to adapt Strassen's algorithm for non-power-of-two matrices

Method

1. **Size Determination**: Compute the nearest size that is a power of two, which is greater than the largest dimension of the original matrices.
2. **Matrix Padding**: Expand the original matrices to the power-of-two size by filling additional rows and columns with zeros, embedding the original matrices in the top-left corner.
3. **Multiplication Execution**: Apply Strassen's algorithm to the padded matrices. The algorithm's recursive division will now work correctly due to the power-of-two dimensions.
4. **Result Extraction**: After multiplication, extract the relevant section from the resulting matrix that matches the expected dimensions of the product.

Analysis

The adaptation of Strassen's algorithm comes with several considerations:

- **Computational Overhead**: Padding adds extra elements to the matrices, which can lead to increased computation time, especially for matrices close in size to the next power of two.
- **Memory Implications**: The larger padded matrices consume more memory, potentially leading to resource constraints for very large matrices.

Padding enables the use of Strassen's algorithm for any matrix size but is most beneficial for very large matrices. It is crucial to consider the trade-offs in computational and memory overhead against the performance gains for each specific case.

5.5 Summary

The chapter on Testing and Discussion in the thesis focuses on validating system functionalities, performance evaluations, and discussions on algorithm optimizations.

Testing encompassed user authentication, database integration, and the Doctor app's display functionality. Scenarios included registration, login/logout processes (including Google One Tap), and data storage/retrieval. All tests successfully demonstrated the system's robustness and reliability.

Performance tests compared standard and improved matrix multiplication methods using a high-performance computing environment. Findings revealed that while standard methods suffice for smaller matrices, parallel processing and the Strassen algorithm significantly outperform them in larger matrix computations. The Strassen-Eigen hybrid approach was particularly effective for large matrices. The discussion extended to adapting Strassen's algorithm for non-standard matrix sizes

Chapter 6

Impact and Exploitation

6.1 Introduction

This project, spanning across two distinct phases, traces a compelling journey from its nascent conceptualization to the realization of a comprehensive health monitoring system. Mirroring a typical product development cycle, the journey entailed rigorous research, innovative development, and meticulous validation, culminating in the production of an application that significantly enhances user interaction with health data.

6.2 Research and Development

Phase 1: Establishing the Foundation

- **Objective:** Initial focus on developing a foundational health-monitoring system, with emphasis on wearable sensor integration for real-time health metrics monitoring.
- **Research Efforts:** Exploration of technologies for effective data acquisition and user interaction, particularly in the context of wearable health sensors.
- **Development Achievements:** Creation of a system for collecting and displaying real-time health data including heart rate, SpO2 level and human activity, establishing a base for future enhancements.

Phase 2: Advancing Capabilities

- **Expanding the Scope:** Phase 2 aimed to address the limitations of Phase 1 and introduce advanced functionalities.
- **Transition to Cloud-Based Solutions:** Integration of Firebase for cloud-based data management, enhancing reliability, accessibility, and security.

- **Android Application Development:** Development of patient and doctor apps with new features like user authentication and cloud data synchronization.
- **Machine Learning Optimization:** While significant in Phase 2, machine learning optimization is not the primary focus of this impact discussion.

This journey from Phase 1 to Phase 2 reflects a comprehensive approach to product development, adhering to industry standards in innovation, user-centric design, and technological advancements. The shift to a sophisticated, cloud-integrated solution underlines the commitment to addressing healthcare monitoring and data management challenges.

6.3 Validation and Production

Phase 1: Initial Testing and Validation

- **Accuracy of MAX30102 Readings:** Comparative tests with sensors from different manufacturers using two libraries showcased reliable and consistent Heart Rate and SpO2 level readings.
- **Machine Learning Model Testing:** Implementation of 5-fold LOSOXV strategy and iterative optimization based on validation results. Two models - for upper body and lower body - were tested, with the upper body model achieving 97% accuracy. The lower body model excelled in recognizing activities like standing and walking.
- **Real-Time Recognition Evaluation:** Tests involving individuals performing various activities while wearing sensors, assessing the real-time accuracy of both models, leading to the development of a hybrid model.

Phase 2: Advanced Testing and Production

- **User Authentication Testing:** Comprehensive testing of the registration, login, and logout processes, including scenarios like incorrect password entries and Google sign-in.
- **Database Integration Validation:** Verification of the integration with Firebase, including tests for storing, retrieving, and updating health data.
- **Doctor App Functionality Tests:** Evaluation of the doctor app's display functionality, ensuring the accuracy and clarity of the graphical data representations.
- **Machine Learning Acceleration Performance Testing:** Rigorous performance testing of various matrix multiplication methods in Visual Studio 2022, demonstrating significant improvements in computational efficiency.

The rigorous testing and validation processes across both phases of the project underscore the commitment to developing a reliable and efficient health monitoring system. Phase 1 laid a strong foundation

with accurate sensor readings and initial machine learning models, while Phase 2 enhanced the system with robust user authentication, cloud-based data management, and advanced computational optimizations. These collective efforts have led to the realization of a sophisticated and impactful health monitoring solution.

6.4 Academic, Economic, and Societal Impact of the Project

Academic Contributions

- **Health Technology Innovation:** This project contributes significantly to academic research in health technology. It pioneers in integrating wearable sensor technology with advanced data processing for real-time health monitoring.
- **Advancements in Machine Learning:** The exploration and application of optimization techniques like the Strassen algorithm and multithreading in machine learning offer valuable insights for academic research, setting a new benchmark for computational efficiency.
- **Interdisciplinary Research Impetus:** The project blends fields like biomedical engineering, computer science, and data analytics, stimulating interdisciplinary research and collaborations.

Economic Impact

- **Cost-Efficiency in Healthcare:** By enabling early detection of health issues and continuous monitoring, the project offers potential cost savings in the healthcare sector, reducing the need for frequent clinical visits.
- **Technological Development:** The project's advancements in Android app development and cloud-based solutions present economic opportunities for tech companies in health technology.
- **Investment Attraction:** Demonstrating successful implementation of advanced technologies, the project can attract investments into health tech research and startups.

Societal Benefits

- **Enhanced Patient Care:** By providing continuous and accurate health monitoring, the system contributes to improved patient care, enabling timely medical interventions.
- **Accessible Health Monitoring:** The development of user-friendly mobile applications makes health monitoring accessible to a broader population, promoting preventive healthcare.
- **Educational Value:** The project serves as an educational model, inspiring future innovations in health technology and raising awareness about the importance of health monitoring.

The project stands at the intersection of technological innovation and healthcare improvement, marking a significant stride in academic research with tangible economic and societal benefits. It not only showcases the potential of integrating technology in healthcare but also sets the stage for future advancements that can revolutionize patient care and health monitoring practices.

6.5 Future Work

Machine Learning for Heart Rate and SpO2 Data

Objective: Implementation of machine learning algorithms for pattern recognition in heart rate and blood oxygen data to identify potential health risks.

Approach: Employ time-series analysis and advanced machine learning models such as LSTM networks or CNNs to analyze physiological data.

Outcome Expectation: Capability to detect health risks like arrhythmia or cardiovascular issues through anomaly detection in physiological data.

Integration: Integration of these algorithms into the existing app for real-time user feedback and alerts.

Integration of Additional Physiological Sensors

Objective: Expand the scope of monitored health metrics by adding sensors for ECG and other vital signs.

Approach: Collaborate with medical device companies for sensor integration, focusing on Bluetooth-enabled devices.

Outcome Expectation: A comprehensive health monitoring system providing a detailed view of the user's health.

Development of iOS and Web Applications

Objective: Extend the app's reach by developing versions for iOS and web platforms.

Approach: Use cross-platform development tools for mobile applications and modern web frameworks for the web application.

Outcome Expectation: Enhanced accessibility, catering to a broader user base and providing platform flexibility.

Optimization of Machine Learning in TensorFlow

Objective: Enhance machine learning computation efficiency by integrating optimized algorithms into TensorFlow.

Approach: Adapt and apply the optimization techniques tested in Visual Studio 2022, such as multithreading, the Strassen algorithm, and the Eigen library, into TensorFlow.

Outcome Expectation: Accelerated training and inference times for machine learning models, facilitating quicker model deployment and updates.

Research and Development: Continuous exploration and testing of cutting-edge computational methods to maintain technological leadership.

6.6 Reflection on Project Outcomes

Meeting Project Objectives

- **Achievement of Technical Goals:** The project successfully met its key technical objectives, including the development of Android applications for both patients and doctors, integration with Firebase for secure data handling, and the exploration of machine learning optimization techniques.

Lessons Learned

- **Importance of User-Centric Design:** Developing applications for patients and doctors emphasized the need for user-friendly and intuitive interfaces, underscoring the importance of a user-centric approach in technology development.
- **Value of Iterative Testing:** The project highlighted the necessity of rigorous and iterative testing processes. It provided insights into the effectiveness of each development stage, ensuring reliability and functionality.
- **Interplay of Different Disciplines:** The project's multifaceted nature, involving aspects of software development, data science, and healthcare, illustrated the benefits and challenges of working at the intersection of different disciplines.
- **Adaptability and Problem-Solving:** Encountering and overcoming various challenges, such as optimizing machine learning algorithms and integrating multiple technologies, reinforced the importance of adaptability and creative problem-solving in complex projects.
- **Scope for Continuous Improvement:** The project's journey from concept to realization highlighted the ongoing nature of technological advancement, suggesting that while current objectives were met, there is always room for further enhancement and innovation.

Reflecting on the project's journey, it is evident that while the primary objectives were successfully met, the development process offered invaluable lessons. These lessons, ranging from the importance of user-centric design to the value of interdisciplinary collaboration, not only enriched the project experience but also provided a roadmap for future endeavors in similar domains. The project stands as a testament to the power of technology in enhancing healthcare, paving the way for future innovations in this vital field.

6.7 Conclusion

This project represents a substantial advancement in health monitoring, integrating Android application development with machine learning optimization. The successful implementation of patient and doctor applications and Firebase integration has significantly improved health data management and accessibility. The exploration of machine learning acceleration, mainly through multithreading and the Strassen algorithm, has enhanced computational efficiency and set a new benchmark in the field. This interdisciplinary endeavour, merging software development with healthcare and data science, promises a transformative impact on patient care and health data analytics, showcasing the potential of technological innovation in advancing healthcare solutions.

Chapter 7

Conclusion

Phase 2 of this project has been marked as a transformative step in health monitoring, skillfully blending advanced computational strategies with bespoke Android application development. Building upon the foundational achievements of Phase 1, the scope and efficacy of the health monitoring system have been significantly expanded, offering a sophisticated solution that aligns with modern healthcare needs.

Revolutionizing Patient-Doctor Interaction: The development of patient and doctor applications, integrated with Firebase, has represented a pivotal advancement in health data management. The patient app, with its real-time synchronisation of physiological data, along with the doctor app providing comprehensive and insightful health data analysis, has worked to streamline the healthcare process. This innovative approach enhances patient engagement with their health data and empowers healthcare professionals with actionable insights, fostering improved healthcare outcomes.

Computational Efficiency in Machine Learning: A standout achievement of Phase 2 has been the successful implementation of advanced computational techniques, particularly in matrix multiplication, which is central to machine learning algorithms. Computational efficiency has been significantly optimised by employing methods like the Strassen algorithm and parallel processing. This breakthrough is crucial for processing extensive health data and complex algorithmic models, propelling the overall capabilities of the health monitoring system to new heights.

Future Trajectory and Continuous Innovation: Looking forward, the project sets the stage for further innovations. Plans to implement machine learning for detailed physiological data analysis, expand the sensor array, and develop cross-platform applications illustrate a commitment to continuous improvement. The integration of optimised computational techniques into TensorFlow heralds a new era in machine learning, promising faster model updates and enhanced system capabilities.

In conclusion, the completion of Phase 2 signifies not merely the culmination of a project but the beginning of a new chapter in health monitoring technology. The symbiosis of software development and

computational algorithms showcased in this project paves the way for groundbreaking advancements in healthcare. Reflecting on this journey, it is evident that an era is being ushered in where health technology is not only advanced but also more accessible, user-friendly, and impactful in everyday healthcare practices.

Acknowledgements

I extend my deepest appreciation to my advisor, Dr. Jiabin Jia, whose unwavering support and insightful guidance have been instrumental to the success of my research. His expertise and profound understanding have significantly shaped and enhanced this work, for which I am incredibly grateful.

I also wish to express my sincere gratitude to Dr. Shiwei Wang, the project examiner, for his time and valuable feedback on this thesis.

Lastly, I am thankful to the University of Edinburgh for the exceptional resources and facilities it has provided. The enriching learning environment and opportunities offered by this esteemed institution have been vital to my academic journey. My experience here has profoundly impacted my growth and the completion of this project.

References

[1] Chen, X.: *Iot for health: An integrated system for monitoring physiological parameters and activity recognition*, MEng Honours Project Phase 1 Report, Matriculation number: S2269664, Jul. 2023.

[2] Bravata, D. M., Smith-Spangler, C., Sundaram, V. *et al.*, 'Using Pedometers to Increase Physical Activity and Improve HealthA Systematic Review,' *JAMA*, vol. 298, no. 19, pp. 2296–2304, Nov. 2007, ISSN: 0098-7484. DOI: 10.1001/jama.298.19.2296. eprint: https://jamanetwork.com/journals/jama/articlepdf/209526/jrv70022_2296_2304.pdf. [Online]. Available: https://doi.org/10.1001/jama.298.19.2296.

[3] Freud, S.: 'Leonardo da Vinci', 1st ed. (Routledge, 2013). DOI: 10.4324/9781315851051.

[4] Bassett, D. R. and John, D., 'Use of pedometers and accelerometers in clinical populations: Validity and reliability issues,' *Physical Therapy Reviews*, vol. 15, no. 3, pp. 135–142, 2010. DOI: 10.1179/1743288X10Y.0000000004. eprint: https://doi.org/10.1179/1743288X10Y.0000000004. [Online]. Available: https://doi.org/10.1179/1743288X10Y.0000000004.

[5] Amin, T., Mobbs, R., Mostafa, N., Sy, L. and Choy, W., 'Wearable devices for patient monitoring in the early postoperative period: A literature review,' *Mhealth*, vol. 7, p. 50, 2021. DOI: 10.21037/mhealth-20-131.

[6] Shin, G., Jarrahi, M. H., Fei, Y. *et al.*, 'Wearable activity trackers, accuracy, adoption, acceptance and health impact: A systematic literature review,' *Journal of Biomedical Informatics*, vol. 93, p. 103 153, 2019, ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2019.103153. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046419300711.

[7] Li, X., Dunn, J., Salins, D. *et al.*, 'Digital health: Tracking physiomes and activity using wearable biosensors reveals useful health-related information,' *PLOS Biology*, vol. 15, no. 1, pp. 1–30, Jan. 2017. DOI: 10.1371/journal.pbio.2001402. [Online]. Available: https://doi.org/10.1371/journal.pbio.2001402.

[8] Achten, J. and Jeukendrup, A. E., 'Heart rate monitoring,' *Sports Medicine*, vol. 33, no. 7, pp. 517–538, 1st Jun. 2003. DOI: 10.2165/00007256-200333070-00004. [Online]. Available: https://doi.org/10.2165/00007256-200333070-00004.

[9] Cummins, C., Orr, R., O'Connor, H. and West, C., 'Global positioning systems (gps) and microtechnology sensors in team sports: A systematic review,' *Sports Medicine*, vol. 43, no. 10, pp. 1025–1042, 1st Oct. 2013. DOI: 10.1007/s40279-013-0069-2. [Online]. Available: https://doi.org/10.1007/s40279-013-0069-2.

[10] Aughey, R. J., 'Applications of gps technologies to field sports,' *International Journal of Sports Physiology and Performance*, vol. 6, no. 3, pp. 295–310, 2011. DOI: 10.1123/ijspp.6.3.295. [Online]. Available: https://journals.humankinetics.com/view/journals/ijspp/6/3/article-p295.xml.

[11] Pantelopoulos, A. and Bourbakis, N. G., 'A survey on wearable sensor-based systems for health monitoring and prognosis,' *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 1, pp. 1–12, 2009.

[12] Soh, P. J., Vandenbosch, G. A., Mercuri, M. and Schreurs, D. M.-P., 'Wearable wireless health monitoring: Current developments, challenges, and future trends,' *IEEE microwave magazine*, vol. 16, no. 4, pp. 55–70, 2015.

[13] Sañudo, B., De Hoyo, M., Muñoz-López, A., Perry, J. and Abt, G., 'Pilot study assessing the influence of skin type on the heart rate measurements obtained by photoplethysmography with the apple watch,' *Journal of Medical Systems*, vol. 43, pp. 1–8, 2019.

[14] Raja, J. M., Elsakr, C., Roman, S. *et al.*, 'Apple watch, wearables, and heart rhythm: Where do we stand?' *Annals of translational medicine*, vol. 7, no. 17, 2019.

[15] Castano, L. M. and Flatau, A. B., 'Smart fabric sensors and e-textile technologies: A review,' *Smart Materials and structures*, vol. 23, no. 5, p. 053 001, 2014.

[16] Castorino, K., Polsky, S., O'Malley, G. *et al.*, 'Performance of the dexcom g6 continuous glucose monitoring system in pregnant women with diabetes,' *Diabetes Technology & Therapeutics*, vol. 22, no. 12, pp. 943–947, 2020.

[17] Reeder, B. and David, A., 'Health at hand: A systematic review of smart watch uses for health and wellness,' *Journal of biomedical informatics*, vol. 63, pp. 269–276, 2016.

[18] Martínez-Pérez, B., De La Torre-Díez, I. and López-Coronado, M., 'Mobile health applications for the most prevalent conditions by the world health organization: Review and analysis,' *Journal of medical Internet research*, vol. 15, no. 6, e120, 2013.

[19] Sama, P. R., Eapen, Z. J., Weinfurt, K. P., Shah, B. R. and Schulman, K. A., 'An evaluation of mobile health application tools,' *JMIR mHealth and uHealth*, vol. 2, no. 2, e3088, 2014.

[20] Darby, A., Strum, M. W., Holmes, E. and Gatwood, J., 'A review of nutritional tracking mobile applications for diabetes patient use,' *Diabetes technology & therapeutics*, vol. 18, no. 3, pp. 200–212, 2016.

[21] Grieger, J. A. and Norman, R. J., 'Menstrual cycle length and patterns in a global cohort of women using a mobile phone app: Retrospective cohort study,' *Journal of Medical Internet Research*, vol. 22, no. 6, e17109, 2020.

[22] Custodio, V., Herrera, F. J., López, G. and Moreno, J. I., 'A review on architectures and communications technologies for wearable health-monitoring systems,' *Sensors*, vol. 12, no. 10, pp. 13 907–13 946, 2012.

[23] Sharma, S., Chen, K. and Sheth, A., 'Toward practical privacy-preserving analytics for iot and cloud-based healthcare systems,' *IEEE Internet Computing*, vol. 22, no. 2, pp. 42–51, 2018.

[24] Butpheng, C., Yeh, K.-H. and Xiong, H., 'Security and privacy in iot-cloud-based e-health systems—a comprehensive review,' *Symmetry*, vol. 12, no. 7, p. 1191, 2020.

[25] Kaur, P. D. and Chana, I., 'Cloud based intelligent system for delivering health care as a service,' *Computer methods and programs in biomedicine*, vol. 113, no. 1, pp. 346–359, 2014.

[26] Bisong, E. and Bisong, E., 'An overview of google cloud platform services,' *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, pp. 7–10, 2019.

[27] Diaz, K. M., Krupka, D. J., Chang, M. J. *et al.*, 'Fitbit®: An accurate and reliable device for wireless physical activity tracking,' *International journal of cardiology*, vol. 185, pp. 138–140, 2015.

[28] Wilder, B.: 'Cloud architecture patterns: using microsoft azure'. (" O'Reilly Media, Inc.", 2012).

[29] Arif, H., Hajjdiab, H., Al Harbi, F. and Ghazal, M.: 'A comparison between google cloud service and icloud,' in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, IEEE, pp. 337–340.

[30] Bahmani, A., Alavi, A., Buergel, T. *et al.*, 'A scalable, secure, and interoperable platform for deep data-driven health management,' *Nature communications*, vol. 12, no. 1, p. 5757, 2021.

[31] Pollard, T. J., Johnson, A. E., Raffa, J. D., Celi, L. A., Mark, R. G. and Badawi, O., 'The eicu collaborative research database, a freely available multi-center database for critical care research,' *Scientific data*, vol. 5, no. 1, pp. 1–13, 2018.

[32] Strickland, E., 'Ibm watson, heal thyself: How ibm overpromised and underdelivered on ai health care,' *IEEE Spectrum*, vol. 56, no. 4, pp. 24–31, 2019.

[33] Uscher-Pines, L. and Mehrotra, A., 'Analysis of teladoc use seems to indicate expanded access to care for patients without prior connection to a provider,' *Health Affairs*, vol. 33, no. 2, pp. 258–264, 2014.

[34] Us Research Program Investigators, A. of, 'The "all of us" research program,' *New England Journal of Medicine*, vol. 381, no. 7, pp. 668–676, 2019.

[35] Worden, K. and Manson, G., 'The application of machine learning to structural health monitoring,' *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1851, pp. 515–537, 2007.

[36] Kattenborn, T., Leitloff, J., Schiefer, F. and Hinz, S., 'Review on convolutional neural networks (cnn) in vegetation remote sensing,' *ISPRS journal of photogrammetry and remote sensing*, vol. 173, pp. 24–49, 2021.

[37] Sherstinsky, A., 'Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network,' *Physica D: Nonlinear Phenomena*, vol. 404, p. 132 306, 2020.

[38] Robert, N., 'How artificial intelligence is changing nursing,' *Nursing management*, vol. 50, no. 9, p. 30, 2019.

[39] Boughton, C. K. and Hovorka, R., 'New closed-loop insulin systems,' *Diabetologia*, vol. 64, pp. 1007–1015, 2021.

[40] Pfotenhauer, S. M., Frahm, N., Winickoff, D., Benrimoh, D., Illes, J. and Marchant, G., 'Mobilizing the private sector for responsible innovation in neurotechnology,' *Nature biotechnology*, vol. 39, no. 6, pp. 661–664, 2021.

[41] Wang, H., Qu, Z., Zhou, Q. *et al.*, 'A comprehensive survey on training acceleration for large machine learning models in iot,' *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 939–963, 2021.

[42] Lim, R., 'Methods for accelerating machine learning in high performance computing,' *University of Oregon–Area-2019-01*, 2019.

[43] Ratner, A. J.: 'Accelerating machine learning with training data management'. (Stanford University, 2019).

[44] Esmaeilzadeh, H. and Park, J., 'Machine learning acceleration,' *IEEE Micro*, vol. 39, no. 5, pp. 6–7, 2019.

[45] Quinn, M. J.: 'Parallel computing theory and practice'. (McGraw-Hill, Inc., 1994).

[46] Golub, G. H. and Ortega, J. M.: 'Scientific computing: an introduction with parallel computing'. (Elsevier, 2014).

[47] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E. and Phillips, J. C., 'Gpu computing,' *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[48] Zinkevich, M., Weimer, M., Li, L. and Smola, A., 'Parallelized stochastic gradient descent,' *Advances in neural information processing systems*, vol. 23, 2010.

[49] Weiss, K., Khoshgoftaar, T. M. and Wang, D., 'A survey of transfer learning,' *Journal of Big Data*, vol. 3, no. 1, p. 9, 28th May 2016. DOI: 10.1186/s40537-016-0043-6. [Online]. Available: https://doi.org/10.1186/s40537-016-0043-6.

[50] Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N. and Huang, X., 'Pre-trained models for natural language processing: A survey,' *Science China Technological Sciences*, vol. 63, no. 10, pp. 1872–1897, 1st Oct. 2020. DOI: 10.1007/s11431-020-1647-3. [Online]. Available: https://doi.org/10.1007/s11431-020-1647-3.

[51] Kornblith, S., Shlens, J. and Le, Q. V.: 'Do better imagenet models transfer better?' In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[52] Devlin, J., Chang, M., Lee, K. and Toutanova, K., 'BERT: pre-training of deep bidirectional transformers for language understanding,' *CoRR*, vol. abs/1810.04805, 2018. arXiv: 1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805.

[53] Talip, M. S. A., Razak, M. Z. A., Mohamad, M., Khairuddin, A. S. M., Izam, T. and Azizan, A., 'Enhancing traffic management with embedded machine learning for vehicle detection,' *IEEE*, 2023. DOI: 10.1109/ICM60448.2023.10378908. [Online]. Available: https://dx.doi.org/10.1109/ICM60448.2023.10378908.

[54] Kyrkou, C., 'Yolopeds: Efficient real-time single-shot pedestrian detection for smart camera applications,' *IET Computer Vision*, 2020. DOI: 10.1049/iet-cvi.2019.0897. [Online]. Available: https://dx.doi.org/10.1049/iet-cvi.2019.0897.

[55] Guignard, F.: 'On Spatio-Temporal Data Modelling and Uncertainty Quantification Using Machine Learning and Information Theory'. (Springer, 2023). DOI: 10.1007/978-3-030-95231-0. [Online]. Available: https://dx.doi.org/10.1007/978-3-030-95231-0.

[56] *Firebase*, https://firebase.google.com/, Accessed: 2023-10-01.

[57] Huss-Lederman, S., Jacobson, E. M., Tsao, A., Turnbull, T. and Johnson, J. R.: 'Implementation of strassen's algorithm for matrix multiplication,' in *Proceedings of the 1996 ACM/IEEE Conference on Supercomputing*, ser. Supercomputing '96, (IEEE Computer Society, 1996), 32–es, ISBN: 0897918541. DOI: 10.1145/369028.369096. [Online]. Available: https://doi.org/10.1145/369028.369096.

[58] Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C.: 'Introduction to Algorithms', Second. (MIT Press and McGraw–Hill, 2001), pp. 73–90, ISBN: 0-262-03293-7.

[59] *Eigen: C++ template library for linear algebra*, https://eigen.tuxfamily.org/index.php?title=Main_Page, Accessed: 2023-09-11.

Appendix A

The Gantt chart of Phase 2

FigureA.1is the gantt chart that guided the development of phase 2.

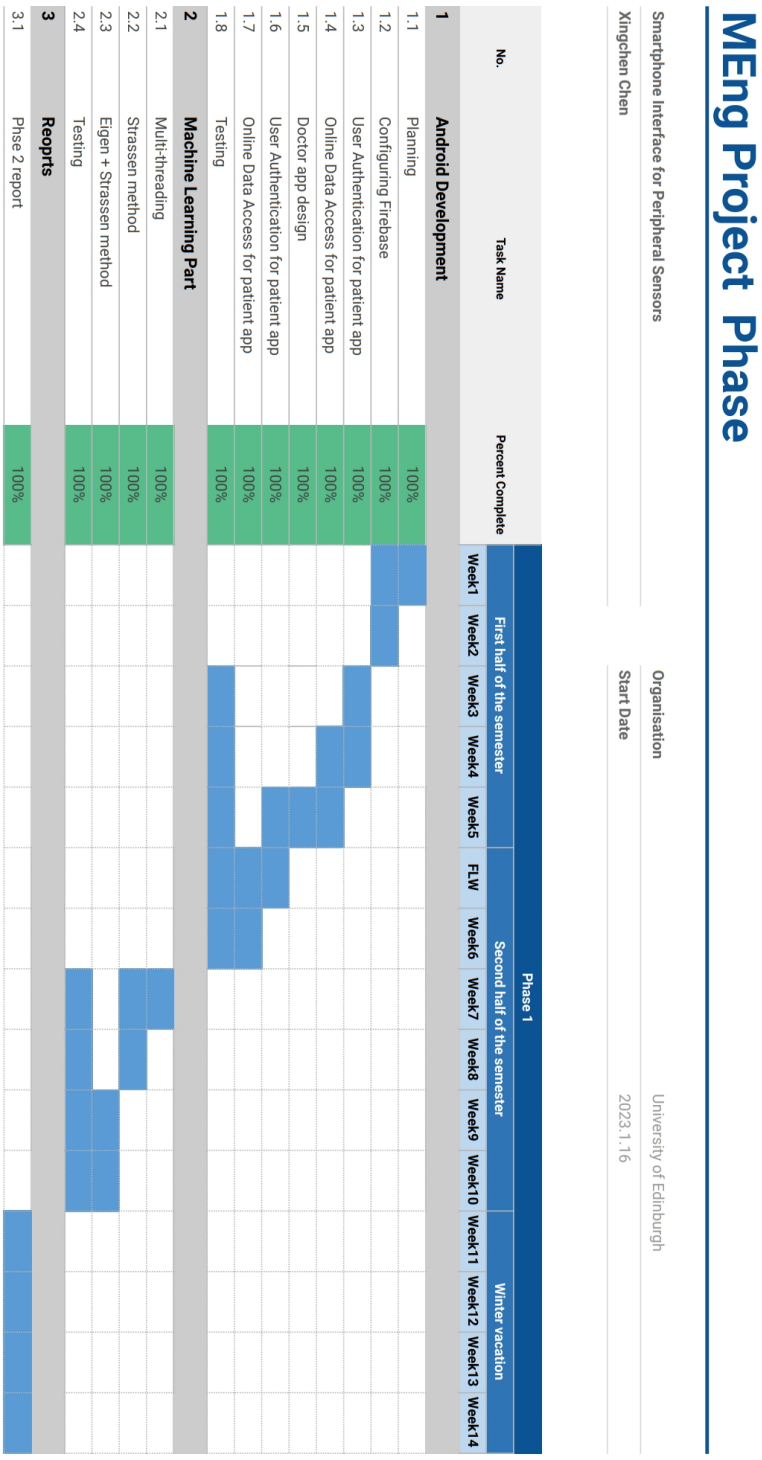

Figure A.1: The Gantt chart of Phase 2