Al Imam Mohammad Ibn Saud Islamic University

College of Computer and Information Sciences

Information Technology Department

ProGear Smart Bag

By:

Name	**ID**	**Department**
Sara Alrahma	444005991	IT
Haya bin Samih	444005442	IT
Hessa Almaarik	444005586	IT

Supervisor:
T.Shatha Alkhaldi

Project Submitted in Partial Fulfillment for the Degree of B.Sc. in "Information Technology"

First - 2026

Al Imam Mohammad Ibn Saud Islamic University

College of Computer and Information Sciences

Information Technology Department

ProGear Smart Bag

By:

Name	**ID**	**Department**
Sara Alrahma	444005991	IT
Haya bin Samih	444005442	IT
Hessa Almaarik	444005586	IT

Supervisor Name	T.Shatha Alkhaldi
Date	
Signature	

Declaration

We **Sara Alrahma**, **Haya Bin Samih**, and **Hessa Almaarik** being members of final year project group number **17**, declare that this report contains only work completed by members of our group except for information obtained in a legitimate way from literature, company or university sources. All information from these other sources has been duly referenced and acknowledged in accordance with the University Policy on Plagiarism.

Furthermore, we declare that in completing the project, the individual group members had the following responsibilities and contributed in the following proportions to the final outcomes of the project:

Student ID	**Responsibilities**[1]	**Contribution**[2] **%**	**Signature**
444005991	Team Member	33.3%	[illegible]
444005442	Team Member	33.3%	[illegible]
444005586	Team Member	33.3%	[illegible]

[1] Write down your responsibilities in the project
[2] Must add to 100%

Acknowledgment

First and foremost, we would like to express our deepest gratitude to Almighty ALLAH for granting us the strength, patience, and ability to complete this graduation project.

We would also like to extend our sincere appreciation to our supervisor, **T.Shatha Alkhaldi** for her continuous guidance, insightful feedback, and valuable support throughout the development of this project. Her encouragement and expertise had a significant impact on improving the quality of our work.

Our thanks are also extended to the College of Computer and Information Sciences at Imam Mohammad ibn Saud Islamic University for providing us with the knowledge, resources, and academic environment that enabled us to grow throughout our years of study.

Finally, we would like to express our heartfelt appreciation to our families and friends for their unwavering support, motivation, and understanding during this journey. Their encouragement played a crucial role in helping us overcome challenges and complete this project successfully.

Abstract

Powerlifters depend on several essential training items—such as lifting belts, wrist wraps, and knee sleeves—to stay safe and perform properly during heavy workouts. However, forgetting or misplacing any of these items is a common issue that can interrupt the athlete's routine and affect overall performance. Based on this problem, our project introduces the ProGear Smart Bag, an IoT-based solution designed to help athletes stay organized and ensure that all their gear is ready before training.

The system uses an ESP32 microcontroller connected to HX711 load-cell sensors to measure the actual weight inside the bag and compare it with the expected weight set by the user. If any difference appears, the system immediately detects that an item might be missing. The hardware communicates with the mobile application using Bluetooth Low Energy (BLE) to provide real-time updates.

The mobile application was built using Flutter, supported by Supabase (PostgreSQL) as the backend for authentication, data storage, and activity tracking. We followed an Agile development approach, which allowed us to gradually improve the system through continuous testing and feedback.

Overall, the ProGear Smart Bag offers a practical and user-friendly solution that enhances organization, reduces forgotten equipment, and supports a safer and more efficient training experience.

Keywords:
Smart Bag, Powerlifting, IoT, ESP32, HX711, BLE, Flutter, Supabase, Equipment Tracking

Abstract (in Arabic)

يعتمد لاعبو القوة على مجموعة من الأدوات الأساسية مثل حزام الرفع، اللفافات، وواقيات الركب لضمان الاستقرار والأمان أثناء التمارين الثقيلة. ومع ذلك، يواجه الكثير منهم مشكلة نسيان بعض المعدات قبل الذهاب لصالة التدريب، مما قد يعرّضهم للخطر أو يؤثر على جودة التمرين. ومن هنا جاءت فكرة مشروع ProGear Smart Bag بهدف مساعدة الرياضيين على التأكد من توفر جميع أدواتهم قبل بدء التدريب.
يعتمد النظام على متحكم ESP32 المتصل بحساسات وزن HX711 لقياس وزن محتويات الشنطة ومقارنته بالوزن المتوقع الذي يحدده المستخدم مسبقًا. وفي حال وجود نقص في الوزن، يقوم النظام باكتشاف ذلك مباشرة. يتم نقل البيانات من الهاردوير إلى التطبيق باستخدام تقنية Bluetooth Low Energy (BLE) لعرض القراءات بشكل لحظي.
تم تطوير التطبيق باستخدام Flutter ليوفّر واجهة سهلة وواضحة، وتم استخدام منصة Supabase كخادم خلفي لتخزين البيانات، إدارة الحسابات، وتسجيل التنبيهات والأنشطة. اعتمدنا منهجية Agile خلال التطوير مما ساعدنا على تحسين النظام بشكل مستمر بناءً على الاختبارات والملاحظات.
بشكل عام، يقدم مشروع ProGear Smart Bag حلاً عمليًا وسهل الاستخدام يساعد لاعبي القوة على تنظيم أدواتهم، تقليل نسيان المعدات، وتعزيز سلامة وجودة التدريب.

List of Abbreviations

API	Application Programming Interface
BLE	Bluetooth Low Energy
DB	Database
ESP32	Espressif Systems 32-bit Microcontroller
HX711	Load Cell Amplifier Module
IoT	Internet of Things
JSON	JavaScript Object Notation
PID	Product Identifier
PK	Primary Key
RLS	Row-Level Security
RPC	Remote Procedure Call
SDK	Software Development Kit
SQL	Structured Query Language
UI	User Interface
UUID	Universally Unique Identifier
UX	User Experience

Table of Contents

Declaration ... ***iii***

Acknowledgment ... ***iv***

Abstract ... ***v***

Abstract (in Arabic) ... ***vi***

List of Abbreviations ... ***vii***

Table of Contents ... ***viii***

List of Figures ... ***x***

List of Tables ... ***xii***

1 Introduction ... ***2***

1.1 Introduction ... 2

1.2 Problem Definition ... 2

1.3 Project Scope ... 3

1.4 Local Impact ... 4

1.5 Global Impact ... 5

1.6 Aims and Objectives ... 6

1.7 Alternative Solutions ... 7

1.8 Method / Approach ... 8

1.9 Project Timeline ... 8

1.10 Report Structure ... 10

1.11 Summary ... 11

2 Literature Review ... ***15***

2.1 Introduction ... 15

2.2 Background ... 15

2.2.1 Subsections ... 4

2.3 Related Work ... 18

2.3.1 Subsections ... 4

2.4 Summary ... 20

3 System Analysis and Design ... ***22***

3.1 Introduction ... 22

3.2 System Perspective ... 22

3.3 Requirements Elicitation Techniques ... 23

3.4 System Requirement ... 29

3.4.1 Functional Requirement ... 29

3.4.2 Non-Functional Requirement ... 29

3.5 User Interface Prototype ... 31

3.6 System Design ... 32

3.6.1 Architectural Design 36
3.6.2 Class Diagram 38
3.6.3 Sequence Diagram 38
3.6.4 Database Design (if any) 47

3.7 Summary 48

4 Implementation 50

4.1 Introduction 50

4.2 Programming Languages and Tools 52

4.3 User Interface Implementation 54

4.4 Database Implementation (if any) 59

4.5 Packages and Classes Description 62

4.6 Procedures Description 64

4.7 Summary 64

5 Testing and Results 70

5.1 Introduction 70

5.2 System Testing 70
5.2.1 Unit Testing 71
5.2.2 Integration Testing 71
5.2.3 User Acceptance Testing 73
5.2.4 Test Cases 74

5.3 Discussion 76

5.4 Summary 81

6 Conclusion and Future Work 85

6.1 Conclusion 85

6.2 Future Work 85

References 89

Appendix 93

A. Code Snippets 94

B. Presentation Slides 96

C. Miscellaneous 99

D. Meetings 102

E. User Manual 106

List of Figures

Figure 1 Agile Methodology....10
Figure 2 System Flowchart....10
Figure 3 Timeline....10
Figure 4 Iot-Based Smart Bag System Overview....17
Figure 5 Question 1....24
Figure 6 Question 2....25
Figure 7 Question 3....25
Figure 8 Question 4....25
Figure 9 Question 5....26
Figure 10 Question 6....26
Figure 11 Question 7....26
Figure 12 Question 8....27
Figure 13 Question 9....27
Figure 14 Question 10....27
Figure 15 Question 11....28
Figure 16 Question 12....28
Figure 17 Question 13....28
Figure 18 Question 14....29
Figure 19 Use Case Diagram....30
Figure 20 Welcome Screen....32
Figure 21 Sign in Screen....33
Figure 22 BT Instructions Screen....33
Figure 23 Searching | Failed Connection Screen....34
Figure 24 Connection Success Screen....34
Figure 25 Item Settled Screen....35
Figure 26 Reset Weight Screen....35
Figure 27 Home Dashboard Screen....36
Figure 28 Architectural Design....37
Figure 29 Class Diagram....38
Figure 30 App Initialization and Home Dashboard....39
Figure 31 Bluetooth Connection and Data Streaming....40
Figure 32 Reset Expected Weight....41
Figure 33 Home Header Activity Navigation....42
Figure 34 Battery Low Notification....43
Figure 35 Weight Delta Alert....43
Figure 36 User Login....44
Figure 37 User Registration....45
Figure 38 Reset Password....45
Figure 39 Hardware Weight & BLE Streaming....46
Figure 40 ER Diagram....47
Figure 41 Login....56
Figure 42 Bluetooth Scan & Connect....56
Figure 43 Home Dashboard....57
Figure 44 Reset Weight....57
Figure 45 Activity Center....58
Figure 46 Settings....58

Figure 47 Pseudocode 1 65
Figure 48 Pseudocode 2 65
Figure 49 Pseudocode 3 66
Figure 50 Pseudocode 4 67
Figure 51 Pseudocode 5 67
Figure 52 Pp 1 96
Figure 53 Pp 2 96
Figure 54 Pp 3 97
Figure 55 Pp 4 97
Figure 56 Pp 5 98
Figure 57 Pp 6 98
Figure 58 form 1 99
Figure 59 form 2 99
Figure 60 form 3 99
Figure 61 form 4 99
Figure 62 form 5 99
Figure 63 form 6 100
Figure 64 form 7 100
Figure 65 form 8 100
Figure 66 form 9 100
Figure 67 form 10 100
Figure 68 form 11 100
Figure 69 form 12 101
Figure 70 form 13 101
Figure 71 form 14 101
Figure 72 form 15 101
Figure 73 form 16 101
Figure 74 meeting 1 102
Figure 75 meeting 2 102
Figure 76 meeting 3 102
Figure 77 meeting 4 102
Figure 78 meeting 5 103
Figure 79 meeting 6 103
Figure 80 meeting 7 103
Figure 81 meeting 8 103
Figure 82 meeting 9 104
Figure 83 meeting 10 104
Figure 84 meeting 11 104
Figure 85 meeting 12 104
Figure 86 meeting 13 105
Figure 87 meeting 14 105
Figure 88 UM1 106
Figure 89 UM2 106
Figure 90 UM3 106
Figure 91 UM4 107
Figure 92 UM5 107
Figure 93 UM6 107
Figure 94 UM7 108

List of Tables

Table 1 Comparison Table for Alternative Solutions 7
Table 2 Team Responsibility 11
Table 3 Related Work 19
Table 4 Esp32_Controller 60
Table 5 Weight Sensor 61
Table 6 Packages and Classes 64
Table 7 ESP32 Hardware Component 71
Table 8 BLE Component 72
Table 9 Mobile Application Component (Flutter) 72
Table 10 Integration Between ESP32 & Mobile App 73
Table 11 Feedback and Observations 75
Table 12 Test Case 1 76
Table 13 Test Case 2 77
Table 14 Test Case 3 77
Table 15 Test Case 4 78
Table 16 Test Case 5 78
Table 17 Test Case 6 78
Table 18 Test Case 7 79
Table 19 Test Case 8 79
Table 20 Test Case 9 79
Table 21 Test Case 10 80
Table 22 Test Case 11 80
Table 23 Test Case 12 80

Chapter One: Introduction

1 Introduction

1.1 Introduction

Powerlifting is a strength-based sport that requires athletes to use specific equipment on a daily basis, such as lifting belts, wrist wraps, knee sleeves, chalk, and lifting shoes. These items play a major role in stability, technique, and injury prevention, especially during heavy lifts. Because of this, most powerlifters depend on having all their equipment with them every time they train. However, in reality, many athletes struggle with organizing their gear, especially when they carry multiple items in one bag or move between home, the gym, and competitions. It is very common for an athlete to arrive at the gym only to realize that an important item was forgotten at home, and this often forces them to change their plan or skip certain exercises, which affects their overall progress and confidence.

Even though technology today provides many tracking solutions, most of them are not designed for athletes or the fast-paced environment of the gym. Solutions like RFID tags, QR codes, or GPS trackers work well in logistics, warehouses, and travel management, but they usually require extra devices, manual scanning, internet connection, or expensive equipment. Powerlifters, on the other hand, need something quick, simple, and automatic—something that tells them immediately whether their bag is complete without needing to check each item one by one. This gap is the main motivation behind our project.

To solve this real problem, we created the ProGear Smart Bag, which focuses on practicality and ease of use. Instead of tracking each item separately, our system uses a more direct and reliable method: measuring the weight of the bag. The hardware setup uses an ESP32 microcontroller connected to dual HX711 load-cell sensors to detect even small weight changes. The system compares the current measured weight with the “expected weight” that the athlete sets when all equipment is inside the bag. If any item is removed or forgotten, the weight difference is detected immediately. The communication between the hardware and the mobile application happens through Bluetooth Low Energy (BLE), which allows fast, stable, and wireless updates without requiring internet or a complicated setup.

The mobile application was developed using Flutter, which helped us build a simple, clear, and user-friendly interface that works on both Android and iOS. The system is supported by Supabase as the backend, allowing us to store activity logs, manage notifications, and handle user authentication securely. Through the app, users can monitor real-time weight readings, reset the expected weight whenever they update their gear, receive alerts when something is missing, and review their activity history.

Throughout the development of the ProGear Smart Bag, we followed the Agile methodology, which allowed us to work in small steps, test continuously, and improve the system based on feedback. This approach helped us refine the hardware readings, improve stability, and create an application that actually fits the needs of powerlifters and athletes.

Overall, this chapter introduces the problem of equipment mismanagement among powerlifters, explains the importance of having a reliable way to verify athletic gear, reviews limitations of existing solutions, and presents the concept behind the ProGear Smart Bag as a practical and athlete-centered approach to solving a real challenge in training routines.

1.2 Problem Definition

In an ideal training environment, powerlifters would always begin their sessions fully organized, with every piece of equipment already prepared inside their sports bag. Items such as lifting belts, wrist wraps, knee sleeves, chalk, and lifting shoes would be consistently available and ready to use. This level of preparation would allow athletes to start their workouts smoothly, follow their training plans without delays, and maintain a consistent routine that supports steady progress.

In reality, maintaining this level of organization is much harder than it sounds. Powerlifters often juggle multiple responsibilities—such as school, work, and personal commitments—making it easy for small details to be overlooked. Packing a sports bag becomes a quick, routine task that depends mostly on memory. Because athletes carry several items, and some of them are small or used irregularly, it is common for something to be forgotten without noticing. This issue is made even more likely when athletes train at different times, switch gyms, or pack their gear in a rush.

Traditional sports bags do not provide any form of verification or assistance, so athletes often realize something is missing only after arriving at the gym. Even though forgetting one item may seem like a minor inconvenience, it can interrupt the training flow. A workout planned around specific lifts may need to be changed or delayed, and the athlete may need extra time to adjust their routine or find alternatives. These repeated interruptions create frustration, reduce training efficiency, and make the preparation process feel inconsistent.

Although technology is increasingly present in sports, most available tracking solutions are built for other fields—like logistics or inventory management—and do not match the simple, fast-paced environment of athletic training. Systems such as RFID tags or QR codes require manual scanning or additional hardware, which adds effort instead of reducing it. What powerlifters need is a solution that is automatic, simple, and fits naturally into their existing workflow without requiring them to change how they pack or use their bag.

The ProGear Smart Bag aims to bridge this gap by introducing an automated approach based on real-time weight monitoring. Instead of tracking each item individually, the system detects missing equipment by comparing the bag's measured weight with the expected weight set by the user. This method provides a practical and low-effort way for athletes to confirm that their equipment is complete before leaving for the gym. By simplifying the preparation process and reducing preventable mistakes, the ProGear Smart Bag helps athletes stay more organized, consistent, and prepared for their training sessions.

1.3 Project Scope

The scope of the ProGear Smart Bag project centers on developing a functional prototype that applies Internet of Things (IoT) technology to help powerlifters manage their essential training equipment more efficiently. IoT systems are widely known for enabling continuous monitoring and real-time data synchronization, which makes them suitable for practical, everyday applications [3]. In this project, the smart bag integrates HX711 load sensors to detect weight variations associated with missing equipment, along with Bluetooth Low Energy (BLE) communication to transmit real-time readings between the hardware and the mobile application. BLE is particularly well-suited for low-power and short-range communication, making it ideal for portable sports-related systems [4].

To ensure accurate and consistent weight measurements, the system uses load cells combined with the HX711 amplifier module, a common choice in embedded measurement systems due to its stability and precision [5]. All sensor data is transmitted to a cross-platform mobile application built using Flutter, a modern framework known for supporting high-performance applications across multiple operating systems with a unified codebase

[6]. Through the app, athletes can monitor live readings, review equipment status, and receive immediate alerts when a discrepancy suggests that an item may be missing.
The project scope includes hardware integration, sensor calibration, BLE communication logic, mobile application development, local data processing, and controlled testing conducted by the project team. Testing scenarios are designed to verify system responsiveness, weight-change detection accuracy, Bluetooth stability, and overall usability for individual powerlifters.
However, several features fall outside the scope of this prototype. These include cloud synchronization, GPS tracking, long-term analytics, team-based multi-user management, and voice assistant integrations. Additionally, BLE performance may vary in high-interference environments, which is a natural limitation of the communication protocol [7]. Despite these constraints, the ProGear Smart Bag successfully demonstrates a clear proof-of-concept that can be expanded in future versions to support more advanced capabilities.

1.4 Local Impact

The ProGear Smart Bag provides meaningful value for athletes in Saudi Arabia, particularly powerlifters who depend on multiple essential items during every training session. Many athletes struggle with keeping their gear organized, and forgetting even one item can cause unnecessary delays or disrupt their routine. By helping users confirm that all required equipment is packed before they arrive at the gym, the system supports smoother training sessions and reduces preventable interruptions, which aligns with studies highlighting the importance of proper preparation and readiness in athletic performance [8].
As the fitness culture continues to grow in Saudi Arabia, supported by national initiatives under Vision 2030 to promote sports participation and healthier lifestyles, solutions that combine technology with daily training routines have become increasingly relevant [9]. The ProGear Smart Bag fits naturally within these efforts by offering an affordable, accessible, and athlete-friendly tool that strengthens preparation habits and encourages consistent training behavior.
Furthermore, by reducing how often athletes misplace or unnecessarily re-purchase gear, the system indirectly supports more sustainable consumption practices. This aligns with

broader sustainability guidelines that emphasize minimizing waste and promoting more conscious use of personal equipment in everyday activities [10].

1.5 Global Impact

The ProGear Smart Bag tackles a global challenge commonly experienced by strength athletes: the difficulty of consistently keeping track of essential training equipment such as belts, wraps, and sleeves. This issue is not limited to local gyms—it affects powerlifters and fitness enthusiasts across different countries, especially as training environments become more fast-paced and individualized [11]. Forgetting or misplacing equipment often leads to interrupted sessions, reduced training quality, or the need to repurchase items, which is a recurring global concern among athletes.

By offering an IoT-based, portable, and low-cost alternative, the ProGear Smart Bag provides a practical solution compared to more expensive tracking technologies such as RFID-based systems or smart storage lockers, which are not always accessible to individual athletes or small gyms around the world [12]. The system's real-time monitoring and immediate alerts support better training preparedness, making it valuable in regions where athletes rely heavily on personal equipment and do not have institutional support or advanced tracking tools.

Furthermore, the project aligns with the global shift toward smart fitness technology, where digital solutions are increasingly used to enhance efficiency, reduce human error, and support safer training practices. By helping users maintain consistent equipment management, the system also contributes to sustainability efforts by reducing equipment loss, unnecessary replacement purchases, and overall material waste—a growing global concern across the sports and fitness industry [13].

1.6 Aims and Objectives

The aim of this project is to develop a smart bag system designed specifically for powerlifters, using IoT technology to ensure that all essential training equipment is present before each session. The system helps athletes stay organized, avoid unnecessary training delays, and prepare with confidence by detecting missing items in real time.

Objectives

- Monitor training equipment automatically by using load sensors connected to a microcontroller to identify any missing items before the athlete begins training.
- Develop a cross-platform mobile application that enables athletes to register their equipment and view the real-time status of their bag.
- Provide instant notifications to the user whenever weight discrepancies are detected, indicating that an item may be missing.

1.7 Alternative Solutions

Ensuring the readiness and availability of essential powerlifting gear before training sessions is a critical factor in supporting athletes' safety and performance. Powerlifters rely heavily on specialized equipment such as lifting belts, wrist wraps, and knee sleeves, making efficient gear tracking a fundamental requirement. Practical and scalable solutions are necessary to assist powerlifters in managing their equipment effectively and minimizing disruptions during training. Several alternatives have been considered, ranging from manual checklists to RFID tagging systems and smart locker technologies. Each solution differs in terms of technological feasibility, operational practicality, and cost-efficiency. This section presents a comparison of these solutions, focusing on their applicability to the needs of individual powerlifters and small teams.

Table 1 Comparison Table for Alternative Solutions

Solution	Features	Costs	Benefits	Drawbacks
Manual Gear Checklist	Powerlifters manually verify each essential gear item, such as belts and wraps, before training sessions.	No initial setup cost; only manual effort required.	Easy to implement; does not require technology.	Highly time-consuming; prone to human error; inefficient for handling multiple gear items [14].
RFID Tagging System	Gear items are tagged with RFID chips and scanned pre- and post-training.	High initial cost for tags, scanners, and system integration.	Quick verification; enables semi-automated checks for multiple items.	Tags can detach; requires line-of-sight scanning; high setup costs not suitable for individuals [14].

Smart Locker System	Lockers with embedded sensors automatically detect the presence of gear.	Very high installation and maintenance costs.	Secure, fully automated detection with minimal user effort.	Lacks portability; fixed infrastructure not suitable for dynamic gym settings; very expensive [15].
ProGear Smart Bag (Proposed)	A portable IoT-based smart bag using weight sensors and BLE for real-time monitoring and alerts.	Moderate cost: ESP32, HX711 sensors, BLE modules, and mobile app development.	Portable; highly accurate in detecting missing gear; real-time alerts; scalable for individual or small team use.	Requires sensor calibration; detection is based only on weight changes; environmental conditions may affect performance.

Compared to traditional and infrastructure-heavy alternatives, the ProGear Smart Bag presents a balanced and targeted solution for powerlifters seeking efficient equipment management. Unlike manual checklists, which are prone to human error and consume valuable preparation time, the smart bag offers automated, real-time tracking of essential gear such as lifting belts, wrist wraps, and knee sleeves. In contrast to RFID systems, which require costly infrastructure and are vulnerable to scanning errors, the ProGear Smart Bag utilizes weight sensors and Bluetooth connectivity to achieve reliable gear verification without the need for direct item tagging. Moreover, compared to smart locker systems that lack portability and involve substantial installation expenses, the smart bag remains lightweight, flexible, and accessible for individual athletes and small teams. Its moderate setup costs, ease of deployment, and real-time alerting capabilities make it an ideal solution for enhancing training efficiency, improving preparedness, and supporting the evolving needs of powerlifters in dynamic gym environments.

1.8 Method / Approach

selected for its effectiveness in managing projects characterized by dynamic requirements, frequent feedback loops, and the need for tight integration between hardware and software components. Agile methodology supports iterative and incremental development, allowing for continuous refinements to both the IoT hardware modules and the cross-platform mobile application. Given the project's real-time monitoring requirements and the critical role of timely feedback in powerlifting environments, Agile provides an ideal framework for ensuring system adaptability, user-centered improvements, and functional reliability [16]. The Agile process for the ProGear Smart Bag is customized to the specific needs of powerlifters and their training conditions. The methodology begins with a comprehensive requirements analysis phase, identifying essential functionalities such as real-time gear weight detection, Bluetooth-based communication, and immediate mobile alerting. During this phase, appropriate hardware components—such as HX711 load sensors, HC-05 Bluetooth modules, and ESP32 microcontrollers—are selected. In parallel, Flutter is chosen as the development platform for building a cross-platform mobile application with an intuitive and user-friendly interface tailored for athletes. Following requirements analysis, the system design phase involves developing detailed architectural diagrams illustrating the interaction among the sensors, microcontroller, Bluetooth module, and the mobile application. Wireframes are also created to plan a seamless user experience, ensuring that athletes can easily register and monitor their gear. Hardware components are individually tested for functionality, and communication protocols are designed to ensure accurate and reliable data exchange between devices.

During the development and implementation phase, the load sensors are integrated with the microcontroller, and the mobile application is developed in parallel. Development is organized into multiple sprints, each followed by continuous testing and evaluation. This approach enables early identification of hardware communication issues, Bluetooth latency challenges, and user interface inefficiencies, which are then immediately addressed through refinements. Testing and evaluation play a crucial role throughout the project. Unit tests are conducted on both the hardware modules and the mobile application. Integration testing is then performed to ensure that the components work cohesively, accurately detecting missing equipment and providing real-time alerts. Real-world scenarios, such as adding or removing powerlifting equipment from the bag, are simulated to validate the system's responsiveness and reliability under practical conditions.

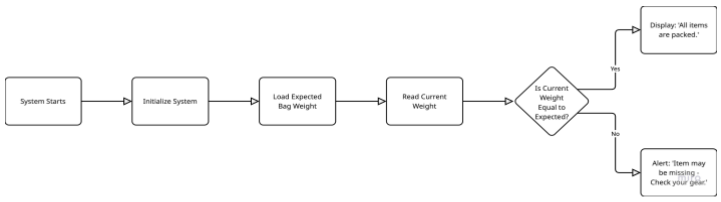
Finally, during the review and optimization phase, real-world feedback is collected from athletes who use the prototype during training sessions. Performance metrics, such as notification accuracy and response time, are analyzed. Based on user feedback and testing outcomes, refinements are applied to improve system responsiveness, minimize latency, and ensure the robustness of the ProGear Smart Bag in dynamic gym environments. Through this iterative development approach, the project aims to deliver a fully functional prototype aligned with the specific needs and expectations of powerlifters.

AGILE
1 Requirements
2 Design
3 Development
4 Testing
5 Deployment
6 Review

Figure 1 Agile Methodology

System Flowchart:

The flowchart below Figure 2 System Flowchart illustrates the operational logic of the ProGear Smart Bag. The system automatically initializes, reads the current weight of the bag, and compares it to a predefined expected weight. If the values do not match, an alert is sent to the user through the mobile application.

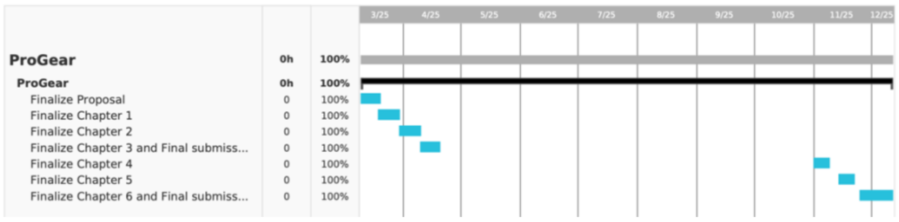


System Starts
Initialize System
Load Expected Bag Weight
Read Current Weight
Is Current Weight Equal to Expected?
Yes
Display: 'All items are packed.'
No
Alert: 'Item may be missing - Check your gear.'

Figure 2 System Flowchart

1.9 Project Timeline

The project timeline, as shown in Figure 3 Timeline provides a detailed schedule of the project's key tasks, including their durations, start and end dates, and the dependencies between them. This Gantt chart ensures that the project progresses in an organized manner, meeting the proposed deadlines efficiently.



ProGear	**0h**	**100%**
ProGear	**0h**	**100%**
Finalize Proposal	0	100%
Finalize Chapter 1	0	100%
Finalize Chapter 2	0	100%
Finalize Chapter 3 and Final submiss...	0	100%
Finalize Chapter 4	0	100%
Finalize Chapter 5	0	100%
Finalize Chapter 6 and Final submiss...	0	100%

Figure 3 Timeline

Team Member’s Responsibilities’:

The success of the ProGear Smart Bag project depends on the collective contributions of all team members, with tasks assigned based on individual strengths and areas of expertise. This structured distribution promotes smooth progress throughout all stages of the project, including planning, system development, testing, and documentation. Each member plays a key role in achieving the project’s goals and ensuring its successful delivery. The detailed allocation of responsibilities is presented Table 2 Team in below.

Table 2 Team Responsibility

Task	**Responsibility**		
	Sara	Haya	Hessa
Assist in refining the proposal and articulating its significance	✓	✓	
Contribute to structuring Chapter 1 and clarifying project objectives and research scope		✓	✓
Assist in developing Chapter 2 and summarizing foundational concepts	✓		✓
Support the writing of Chapter 3 and prepare for final submission		✓	
Assist in drafting Chapter 4 and integrating relevant case studies	✓	✓	
Contribute to Chapter 5 by analyzing results and interpreting findings			✓
Support the completion of Chapter 6 and finalize the overall document	✓		✓

1.10 Report Structure

This report documents the full development process of the ProGear Smart Bag system. Chapter 1, Introduction and Project Overview, presents the background and motivation behind the project, highlighting the challenges athletes—especially powerlifters—face when organizing and remembering essential training equipment. It also introduces the problem statement, defines the project scope, and clarifies that the solution focuses on weight-based detection rather than GPS or item-specific identification. Additionally, the chapter discusses the local and global impact of the proposed solution, outlines the project's aim and objectives, compares alternative approaches such as RFID systems and manual checklists, and explains the rationale behind choosing an IoT, weight-sensor-based method. The Agile methodology is introduced, along with a high-level timeline that guided the development process.

Chapter 2, Literature Review and Related Work, explores previous studies and existing technologies related to smart bags, IoT-based monitoring systems, and training-gear management. It presents theoretical foundations such as weight-sensing principles, Bluetooth communication, and mobile-based alert mechanisms. By analyzing examples like smart luggage systems, gym inventory management tools, and RFID-based solutions, the chapter identifies a clear gap in the market for a lightweight, affordable, athlete-focused smart bag. These insights inform the design decisions and help justify the technical direction taken in the development of ProGear Smart Bag.

Chapter 3, System Analysis and Design, describes the system's structure and its role within the athletic training environment. It outlines the functional requirements—including real-time weight monitoring, Bluetooth connectivity, and alert notifications—and the non-functional requirements such as usability, portability, and system reliability. The chapter includes the interface prototypes for the mobile application, showing how users interact with the system. The architecture is explained in detail, covering the integration between the load sensors, ESP32 microcontroller, BLE communication, and the Flutter mobile app. Supporting diagrams—including use case diagrams, class diagrams, and sequence diagrams—illustrate the system workflow and logic from both hardware and software perspectives. Chapter 4, System Implementation, provides a detailed walkthrough of how the hardware and software components were developed and integrated. This includes wiring the load cells with HX711 modules, configuring the ESP32 firmware to read and transmit real-time weight data, and implementing a custom BLE protocol. On the software side, the chapter describes the Flutter UI implementation, state-management structure, and how the app handles device connection, data parsing, and alert generation. Screenshots and code snippets are included only for critical components to demonstrate the system's core logic without overwhelming the reader.

Chapter 5, Testing and Results, documents the evaluation process of the prototype. Various test scenarios—such as detecting missing items, verifying Bluetooth communication stability, and measuring weight-reading consistency—are conducted to validate system performance. The chapter presents results, observations, and key findings, supported by tables, graphs, and screenshots. Any issues encountered during testing are discussed, along with steps taken to refine the system based on feedback and repeated trials.

Chapter 6, Conclusion and Future Work, summarizes the achievements of the project and reflects on the effectiveness of the ProGear Smart Bag as a practical solution for powerlifters. The chapter highlights how the system met its objectives and improved the process of tracking essential gear. Future work suggestions are also outlined, such as exploring cloud integration, expanding sensor capabilities, supporting multiple users, and enhancing data analytics for long-term training insights. These improvements offer opportunities for scaling the prototype into a more advanced and commercially viable product.

1.11 Summary

This chapter discussed the importance of properly managing powerlifting equipment to avoid training disruptions that occur when essential gear is forgotten or misplaced. It explained how this issue affects athletes' preparation and highlighted the need for a solution that can automatically verify equipment readiness without relying on manual checks.

Several existing approaches were reviewed, including manual checklists, RFID tagging systems, and smart locker technologies. While each provides some level of support, they are often limited by high costs, setup requirements, or the need for direct user interaction. None of these methods offer a lightweight, portable, and fully automated solution that suits individual athletes. To fill this gap, the proposed ProGear Smart Bag uses IoT technology that combines weight sensors with Bluetooth communication to detect missing items and instantly notify the user through a mobile application. The system follows the Agile methodology, allowing continuous refinement and improvements based on feedback—making it more flexible than traditional fixed design approaches.

The next chapter presents the literature review, which further explores the problem and examines previous technological solutions relevant to smart monitoring and athletic gear management.

Chapter Two: Literature Review

2 Literature Review

2.1 Introduction

This chapter presents a detailed review of the scientific, technical, and academic foundations that support the development of the ProGear Smart Bag. While Chapter 1 introduced the general idea, motivation, and problem context, this chapter expands on that foundation by providing the background knowledge needed to understand how the system operates and why certain technologies were selected.

The chapter is divided into two major sections. The first section, Background, explains the key concepts that form the basis of the system's design. These include the Internet of Things (IoT) and its role in creating interconnected smart devices, sensor-based monitoring systems used for detecting physical changes such as weight variations, Bluetooth communication technologies that enable wireless data transmission between hardware and mobile devices, and modern mobile application development frameworks such as Flutter. Understanding these concepts helps clarify the reasoning behind the project's technical decisions and shows how each component contributes to solving the main problem.

The second section, Related Work, examines previous studies, systems, and technologies that have attempted to address similar challenges in tracking personal equipment, sports gear management, and smart bag solutions. This includes reviewing approaches such as RFID-based tracking, smart luggage designs, gym inventory management tools, and wearable IoT systems. By analyzing the strengths and limitations of existing solutions, this chapter identifies gaps—such as lack of portability, high costs, or reliance on manual steps—that still affect athletes today. These gaps directly highlight the need for a lightweight, affordable, and user-friendly solution like the ProGear Smart Bag.

Overall, Chapter 2 aims to position the project within the broader context of smart sports technologies. It establishes the relevance of the problem, supports the design decisions made by the team, and demonstrates how the proposed solution aligns with current trends while addressing limitations found in previous work.

2.2 Background

In strength-based sports such as powerlifting, bodybuilding, and Olympic lifting, athletes rely heavily on specialized gear—including lifting belts, wrist wraps, supportive shoes, and other accessories—to enhance performance and prevent injuries. Research indicates that proper sports equipment significantly reduces injury risks and improves biomechanical efficiency during high-load exercises [17]. For example, lifting belts help increase intra-abdominal pressure, providing essential spinal stability during compound movements [18]. Forgetting or misusing this equipment can lead to technical errors and increase the likelihood of strain-related injuries.

Traditionally, athletes have depended on manual checklists or personal memory to pack their essential gear. Although simple, these methods are highly error-prone, especially during stressful situations such as competition days or while traveling [19]. Factors such as fatigue, time pressure, and distractions can cause athletes to overlook important items, which negatively impacts training quality, mental focus, and overall readiness. Studies in sports preparation emphasize the role of consistent equipment routines, noting that missing gear can directly disrupt an athlete's physical performance and psychological confidence [20].

This issue becomes even more complex in team-based or institutional environments, where coaches or facility managers are responsible for ensuring that multiple athletes have all required equipment. Without automated support, verifying gear readiness for several individuals is time-consuming, inconsistent, and inefficient [21]. Addressing this logistical challenge requires a technological solution that is scalable, reliable, and capable of delivering real-time feedback.

As smart technologies continue to expand, systems built on the "Internet of Things (IoT)" have introduced new possibilities for automating physical asset tracking. IoT is defined as "a network of physical objects embedded with sensors, software, and other technologies that enable communication and data exchange over the internet" [22]. In the sports field, IoT-based solutions are increasingly adopted for athlete monitoring, biometric wearables, and equipment management.

A core component of IoT systems is the sensor, defined as "a device that detects or measures a physical property and converts it into an interpretable signal" [23]. Within the ProGear Smart Bag, sensors are used to detect gear presence based on weight variations. The specific type of sensor integrated in the system is a load cell, which is "a transducer that converts mechanical force—such as weight or pressure—into an electrical output signal" [24]. Load cells are commonly used in precision weighing systems and logistics applications.

To convert the analog signals generated by load cells, the system employs the HX711 module, described as “a 24-bit analog-to-digital converter (ADC) designed for weigh-scale and industrial control applications” [25]. Its precision, stability, and power efficiency make it suitable for compact embedded systems.

The intelligence of the smart bag relies on an embedded system, defined as “a dedicated computer system designed to perform specific control functions within a larger mechanical or electrical setup” [26]. In this project, the embedded system is represented by the ESP32 microcontroller, which processes sensor data, compares weight values to predefined thresholds, and communicates feedback to the user.

Communication between the embedded system and the mobile application is handled via Bluetooth Low Energy (BLE)—a short-range wireless technology optimized for low power consumption, seamless pairing, and fast data exchange [27].

On the software side, the user interface is developed using Flutter, an “open-source UI software development kit for building cross-platform applications from a single codebase” [28]. The Flutter app provides real-time equipment status, displays alerts for missing items, and ensures an intuitive user experience tailored for athletes.

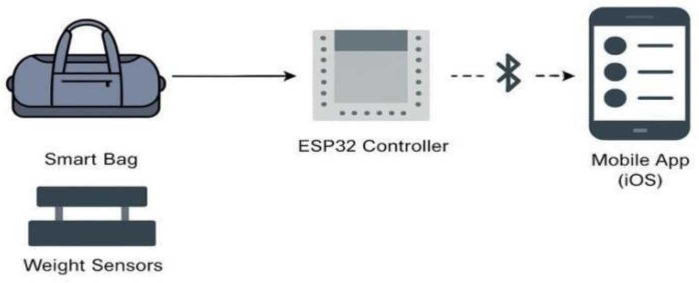


Figure 4 Iot-Based Smart Bag System Overview

As illustrated in Figure 4 Iot-Based Smart Bag System Overview , the ProGear Smart Bag architecture integrates load sensors inside the bag to continuously detect the weight of essential training gear. These sensors transmit raw weight readings to the ESP32 microcontroller, where the data is processed and compared against predefined thresholds for each registered item. When the system identifies a discrepancy—such as a missing or significantly under-weight item—the ESP32 initiates a Bluetooth Low Energy (BLE) communication request to the user’s smartphone. Once received, the mobile application displays an immediate on-screen alert and can also trigger optional sound or vibration notifications.

This real-time feedback loop ensures that athletes verify their gear before leaving for training, reducing the chances of forgotten equipment. Beyond improving logistical organization, the system also supports athletes’ mental focus and readiness by removing the uncertainty associated with manual packing routines—an important factor in maintaining confidence and consistency in high-performance training environments.

2.3 Related Work

Studies related to sensor technologies and wireless communication via Bluetooth are commonly explored across various applications, including health, sports performance, and equipment management. However, no prior studies have combined these technologies to specifically address detecting missing sports equipment inside a portable gym bag. This need is especially significant for athletes who require a reliable and effective way to ensure that they have all their essential gear. Several projects can be grouped into three categories: those using the same solution for the same problem, those using the same solution for different problems, and those using different solutions for the same problem. In the first category, Bousselmi et al. [29] introduced a wireless sensor system for real-time performance monitoring in sports using load cells. The system focused on collecting performance data rather than detecting or tracking gear. Similarly, Hettiarachchi [30] presented an IoT-based gym tracker that monitored user body weight using load sensors and Wi-Fi, offering value for health tracking but not for sports gear management. The discussion here shows that while both studies used load-based sensing, their objectives diverged from the ProGear Smart Bag's goal of inventory verification and gear readiness. These solutions validate the feasibility of load sensors in athletic environments but reveal a gap in their application for item detection and alerting within personal bags.

In the second category, Robotique. Tech [31] developed a weighing scale using the HX711 and ESP32 modules for weight monitoring via Bluetooth. Although the technical approach is similar to ProGear's, the application was entirely health-related and did not involve any form of equipment or inventory tracking. Nordic Semiconductor's D-Fetch Gym planner [32] used Bluetooth Low Energy (BLE) to monitor gym equipment usage, contributing to gym management systems. However, the system did not extend its functionality to tracking items inside an individual's gym bag. The discussion here highlights that these projects effectively implemented similar hardware and communication technologies, but their purposes—personal health tracking or facility management—differed from the individual, mobile, and gear-specific focus of the ProGear Smart Bag. This reinforces the novelty of using such components for item-specific, portable gear verification.

In the third category, Singh and Nigam [33] proposed an RFID-based smart school bag that alerted students when items were missing using tagged components. Lenovo's smart backpack [34] integrated sensors, motion detection, and solar charging for convenience but lacked any mechanism to verify the presence of specific items through weight. Smart locker systems also offer gear verification using embedded sensors but are confined to static environments and are unsuitable for athletes on the move. The discussion for this group reveals that while these approaches do address item detection, they rely on different mechanisms like RFID or motion sensors, which introduce practical limitations. RFID requires tagging every item, which is unrealistic for varied and non-standard sports gear. Locker systems are not portable, and motion detection does not ensure completeness of gear. Therefore, the ProGear Smart Bag addresses these shortcomings by offering a mobile, weight-based, and tag-free solution for real-time gear verification, optimized for athletic environments.

In conclusion, although existing systems provide valuable insights through the use of similar technologies or by addressing related challenges, none have specifically targeted the issue of missing sports gear in a portable, user-friendly format. The ProGear Smart Bag addresses this gap by combining load sensors and Bluetooth communication to deliver a real-time, practical solution designed specifically for athletes' daily needs.

Table 3 Related Work

Study/System	Domain	Technology Used	Limitations	Relevance to ProGear
Bousselmi et al. [29]	Sports Performance Monitoring	Load Cells, Wireless Sensor Network	Focused on performance metrics, not gear tracking	Uses same tech for same field, but different purpose
Hettiarachchi [30]	Fitness Tracking	Load Sensors, IoT (Wi-Fi)	Tracks user weight only; no equipment-level tracking	Similar technology but limited to personal health
Robotique.tech [31]	Personal Health Monitoring	HX711, ESP32, Bluetooth	Designed for weight monitoring only; no object differentiation	Demonstrates relevant tech, but different application
Nordic Semiconductor [32]	Gym Equipment Management	Bluetooth Low Energy (BLE)	Tracks usage of machines, not personal gear	Useful for facility management,

				not portable gear tracking
Singh and Nigam [33]	Education / School Supplies	RFID, Microcontroller	Requires tagging every item; not scalable for gym environments	Same problem (missing items), different and less practical solution
Lenovo [34]	Smart Backpack / Personal Use	Motion Sensors, Solar Charging	No weight verification; complex for sports application	Shares tracking goal, lacks needed functionality for gym gear

2.4 Summary

This chapter presented a comprehensive review of the foundational research and existing technological solutions relevant to the development of the ProGear Smart Bag system. The background section highlighted the growing demand for smart, automated tracking tools in both personal and athletic environments, emphasizing the limitations of traditional manual management methods, which are often inefficient and prone to human error.

The related work analysis examined a range of existing solutions—from RFID-enabled school bags to large-scale IoT inventory systems—and identified their strengths as well as their shortcomings. Despite advancements in smart tracking technologies, the review revealed a clear gap: current solutions do not offer a portable, affordable, and athlete-focused method for monitoring essential training gear.

By addressing this gap, the ProGear Smart Bag introduces a practical and adaptable approach that integrates IoT-based weight sensing with a user-friendly mobile interface. This contribution is significant, as it enhances equipment readiness, reduces the risk of forgotten items, and supports more efficient training routines.

The next chapter will shift from theoretical foundations to technical implementation, detailing the system's architecture, use case models, and core functional components.

Chapter Three: System Analysis and Design

3 System Analysis and Design

3.1 Introduction

In Chapter 2, several existing projects and research studies were examined to understand how different technologies have been utilized to address the issue of forgotten items. The review covered systems that aimed to prevent item loss inside bags as well as solutions that, although targeting different contexts, employed similar technologies such as weight sensors, embedded controllers, and mobile alert mechanisms. Among the examined works were smart bag prototypes that used load-cell–based weight detection, systems integrating sensor–mobile communication, and mobile applications designed to notify users when essential objects were missing.

The literature review provided a comparative analysis of these solutions in terms of hardware selection, communication protocols, user-interaction models, and overall system functionality. Through this analysis, common strengths and limitations were identified, along with gaps that existing systems had not yet addressed—particularly the lack of real-time feedback, limited portability, and insufficient user-centered design. These insights directly influenced the development of the ProGear Smart Bag by highlighting the need for a more reliable, responsive, and athlete-oriented solution.

This chapter presents the system analysis and design of the ProGear Smart Bag. This stage serves as a foundational step where the system's overall structure is defined based on user requirements, project objectives, and technical constraints. The chapter outlines how the system is conceptualized, how its components interact, and how it is architected to achieve the intended functionality with consistency and accuracy. The goal is to ensure that all hardware and software components operate cohesively and support the implementation phase effectively.

The chapter includes a detailed specification of the system's functional and non-functional requirements. It introduces the use-case diagram that illustrates user interaction with the system, followed by descriptions of the system's internal data flow. In addition, the chapter presents the full system architecture, covering both the hardware side—such as sensors, amplifiers, and the microcontroller—and the software side, including the mobile application interface and Bluetooth communication logic. The chapter concludes with design considerations that play a critical role in ensuring the reliability, usability, and real-world performance of the ProGear Smart Bag.

3.2 System Perspective

The ProGear Smart Bag is an independent and fully self-contained system. It was developed from the ground up to address a specific problem faced by athletes: forgetting essential sports equipment. The system does not rely on or extend any pre-existing platforms, applications, or commercial products. It was designed with a unique combination of features tailored for the sports environment, including real-time detection using weight sensors and direct communication with a custom-built mobile application. There are no dependencies on third-party systems, and the architecture was developed to be standalone, ensuring full control over both hardware and software aspects. Although our system is original in its design and not derived from any existing product, there are projects that share similar goals or use related technologies. One such project is the Lenovo, "Smart Backpack System,"[34] which was reviewed in Chapter 2. This project also aims to prevent users—specifically school children—from forgetting items in their bags. It uses a set of sensors connected to an Android application via Bluetooth to alert the user if something is missing. While both systems share the common goal of detecting forgotten items and utilize Bluetooth-based communication with a mobile interface, there are key differences in their design, target audience, and functionality. The "Smart Backpack System" targets a global market focused on children's education and parental assistance. Its design revolves around helping young students avoid forgetting school supplies, with a strong emphasis on external alerts for parents or guardians. On the other hand, the ProGear Smart Bag is aimed at individual users, particularly athletes, who need to independently verify that all of their gear is packed before heading to training or competition. Additionally, while the school bag project may rely on detecting the presence of specific items through manual setup or RFID-like detection, our system uses real-time weight measurement and customizable thresholds, offering more flexibility and fewer dependencies on item-specific tags or shapes. These distinctions highlight the unique direction of our system. Even though similar ideas exist in other domains, the ProGear Smart Bag remains a distinct and self-contained solution that was created to fulfill a different use case, with its own technical design and user experience considerations

3.3 Requirements Elicitation Techniques

Requirements elicitation is the process of identifying and gathering user needs, expectations, and constraints to develop a system that effectively addresses real-world problems. This process is essential for ensuring that the final system aligns with user requirements, thereby increasing usability, acceptance, and overall effectiveness. In the context of developing a smart gym bag designed for powerlifting athletes, two primary elicitation techniques were applied: questionnaires and brainstorming.

Brainstorming is a group creativity technique used to generate a wide range of ideas related to potential solutions. It encourages open communication and diverse viewpoints and assists in identifying system features that may not emerge through direct questioning [35]. This technique was conducted internally among the development team to explore system goals, hardware considerations, and realistic use-case scenarios.

The questionnaire method [36] was selected due to its structured and scalable nature, enabling the efficient collection of quantitative data from a broad user base. It also facilitates standardization of responses and supports targeted distribution to specific user groups. Questionnaires are particularly effective for reaching communities such as athletes, where diverse insights can be gathered at low cost and with minimal time requirements.

For this study, a structured questionnaire was developed to gather insights regarding user needs, preferred features, and expectations for the proposed system. The questionnaire was distributed through multiple channels, including WhatsApp community groups, university student clubs, and fitness-related groups, ensuring access to individuals who actively engage in powerlifting or strength training. A total of 153 responses were collected, providing a rich and diverse dataset that supported a comprehensive understanding of user requirements.

Question 1

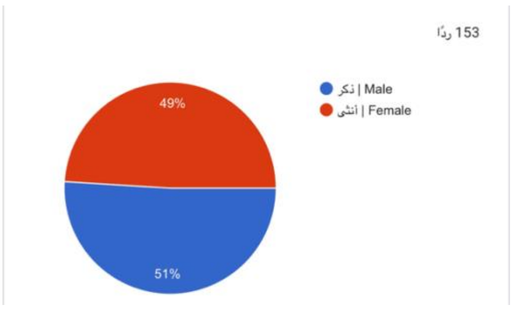


Figure 5 Question 1

This question identifies the gender distribution of respondents. Results were nearly evenly split, with 49% male and 51% female. This balance indicates that the interest in smart gym solutions is shared across genders.

Question 2

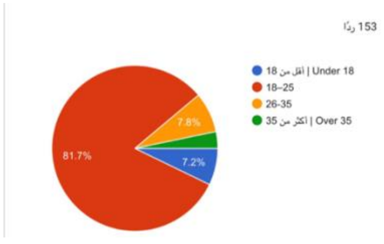


Figure 6 Question 2

This question asks about the age group of respondents. The majority (81.7%) were between 18–25 years old, followed by smaller percentages in other age groups. This shows that the primary target audience is young adults, likely active and open to fitness technology.

Question 3

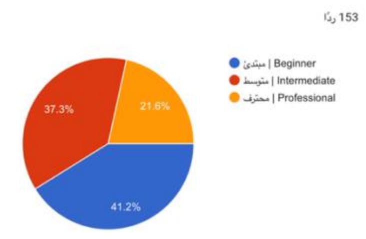


Figure 7 Question 3

This question explores the respondents' experience level in powerlifting. 41.2% identified as professional, 37.3% as intermediate, and 21.6% as beginners. These results suggest that users across all levels of experience could benefit from a smart gym bag, with a strong portion already deeply involved in the sport.

Question 4

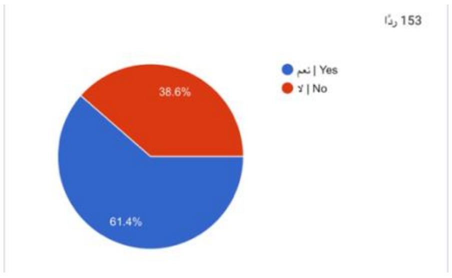


Figure 8 Question 4

This question examines whether respondents currently use a dedicated gym bag. 61.4% answered "Yes", while 38.6% answered "No". This indicates a large portion of users may not have an optimized solution for organizing their gear—highlighting the need for an intelligent alternative.

Question 5

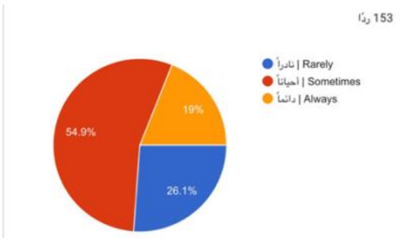


Figure 9 Question 5

This question investigates how often users forget to bring gym equipment. 54.9% answered "Sometimes", 26.1% "Always", and only 19% "Rarely". These results reveal that forgetting items is a common problem that affects many users.

Question 6

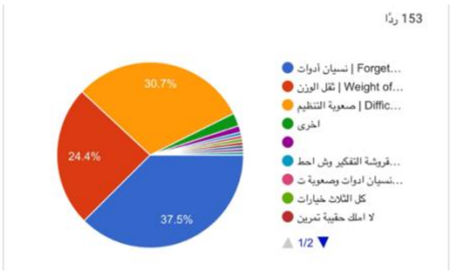


Figure 10 Question 6

This question aims to understand the most common issue when preparing a gym bag. The leading problems were difficulty organizing (37.5%), forgetting equipment (30.7%), and bag weight (24.4%). This underscores the importance of smart organization and reminder features in gym bags.

Question 7

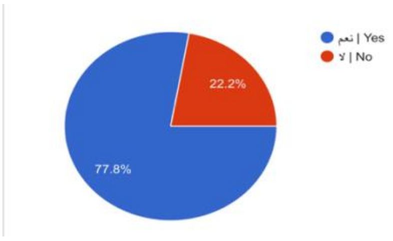


Figure 11 Question 7

This question assesses whether forgetting items has caused workout disruptions. A significant majority (77.8%) reported that it has impacted their training sessions, reinforcing the relevance of our smart bag's alert function.

Question 8

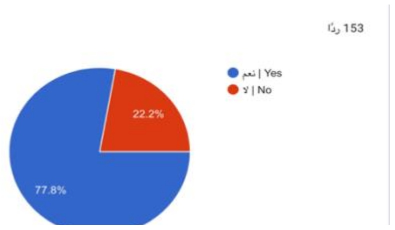


Figure 12 Question 8

This question explores whether respondents have used smart fitness products like smartwatches or workout apps. 77.8% said "Yes", indicating high familiarity with and acceptance of smart technology in the fitness domain.

Question 9

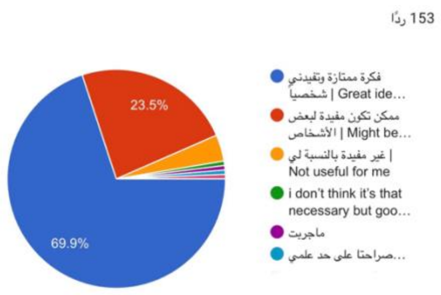


Figure 13 Question 9

This question gauges interest in the concept of a smart gym bag that alerts users about forgotten equipment. Many participants expressed strong interest, with the most common response being that it's a great idea that would personally benefit them.

Question 10

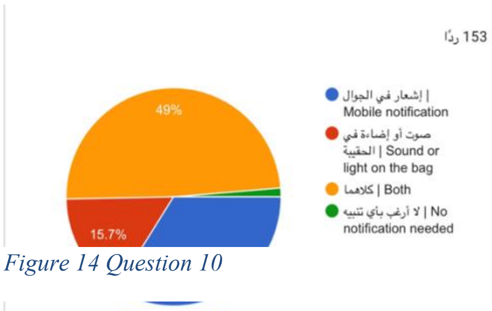


Figure 14 Question 10

This question explores preferred notification methods. 49% preferred mobile notifications, 34% preferred both mobile and physical alerts, and only 2 respondents preferred no notifications. This shows that combining mobile and visual/auditory alerts would meet most users’ expectations.

Question 11

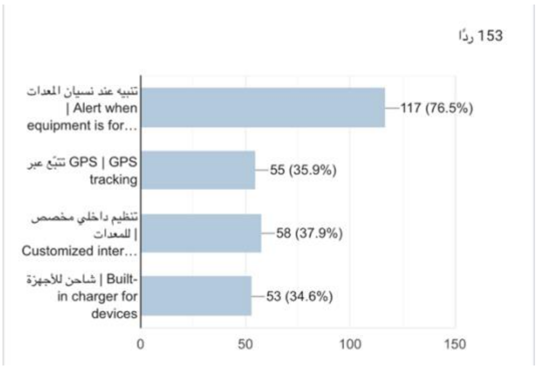


Figure 15 Question 11

This question identifies which feature users find most valuable in a smart gym bag. The most valued feature was “alert when equipment is forgotten” (76.5%), followed by internal organization (37.9%), GPS tracking (35.9%), and built-in device charging (34.6%). These results validate our feature prioritization.

Question 12

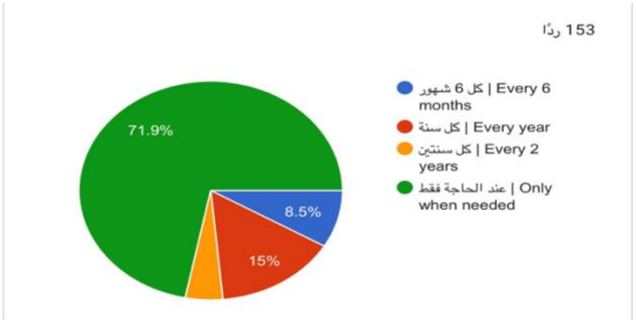


Figure 16 Question 12

This question examines how often users replace their gym bags. 71.9% said they replace it only when needed, indicating a preference for durable and reliable products, which the smart gym bag should address.

Question 13

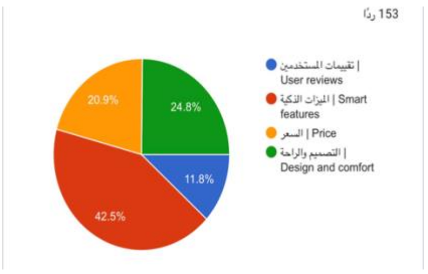


Figure 17 Question 13

This question asks what factors would motivate a user to buy a smart gym bag. The top factors were smart features (42.5%) and design/comfort (24.8%), followed by price (20.9%) and user reviews (11.8%). This highlights the importance of innovation and usability in the purchasing decision.

Question 14

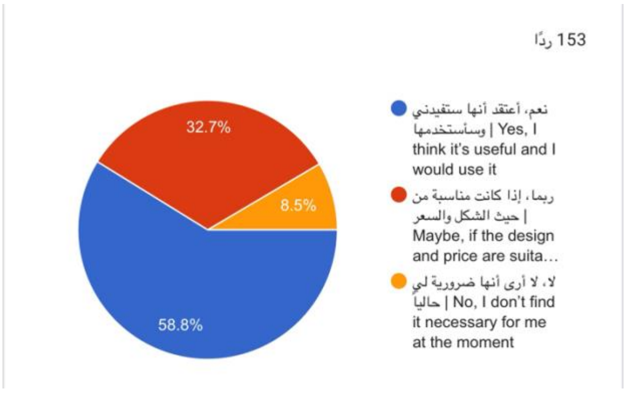


Figure 18 Question 14

This final question asks whether users would consider using a smart gym bag with alert features. A clear majority expressed interest, suggesting strong market potential and alignment with user needs.

In conclusion, the questionnaire responses provided valuable insights into the needs and preferences of our target users—primarily young adults actively engaged in powerlifting. A significant number of respondents reported frequent issues with forgetting equipment, disorganized bags, and the lack of reliable storage solutions. The idea of a smart gym bag that offers alerts and better organization was met with strong interest and support, especially when combined with user-preferred notification methods like mobile alerts. Moreover, features such as item tracking, internal compartmentalization, and smart functionality ranked high in user priority. These findings reinforce the relevance and potential impact of our project, guiding us to develop a solution that aligns with user expectations and solves real pain points in their fitness routines.

3.4 System Requirement

3.4.1 Functional Requirement

Functional requirements are defined as specific descriptions of the behavior or functions a system must perform under certain conditions [37]. These requirements specify the essential operations the system must be able to perform, such as data processing, user interaction, and communication.

The functional requirements of the ProGear Smart Bag system include:

1. The system shall allow the user to register or log in to the mobile application.
2. The system shall establish a Bluetooth connection between the smart bag and the mobile application.
3. The system shall measure the total weight of the items placed in the smart bag using internal sensors.
4. The system shall transmit the measured weight from the smart bag to the mobile application in real time.
5. The system shall compare the current weight of the smart bag with a predefined expected weight and notify the user if a decrease is detected, indicating that one or more items may be missing.

Use Case Diagram

A use case diagram is a visual representation that describes how users interact with a system. It illustrates the system's functionality from the user's perspective, showing the relationships between different actors (such as users or external systems) and the system's use cases. According to Kotonya and Sommerville, use cases describe interactions between external actors and the system to achieve a specific goal [37].

The primary goal of the use case diagram is to capture the dynamic behavior of the system and provide a high-level overview of what the system is expected to do. It helps stakeholders understand system boundaries, functionalities, and how each component or actor contributes to system operations.

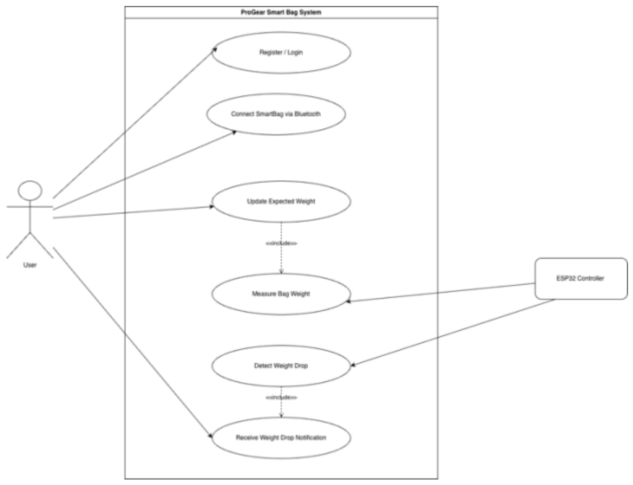


Figure 19 Use Case Diagram

The Use Case Diagram of the ProGear Smart Bag system illustrates the main interactions between the user and the system's internal hardware component, the ESP32Controller. The user can register or log into the mobile application and establish a Bluetooth connection with the smart bag. Once connected, the system automatically initiates a sequence of operations starting with measuring the total bag weight using internal sensors. This measurement is then sent to the mobile application. The controller analyzes the data to detect any drop in weight, which may indicate that an item is missing from the bag. If such a drop is detected, the system includes a use case to notify the user through the mobile application. The ESP32Controller actor is responsible for executing all internal system logic, such as weight measurement, data transmission, and drop detection, while the User actor interacts with the system at the beginning and receives alerts if needed. All use cases are encapsulated within the system boundary of the ProGear Smart Bag, clearly separating external and internal operations.

3.4.2 Non-Functional Requirement

Non-functional requirements refer to criteria that describe how the system performs a function, rather than what the function is [37]. These aspects focus on how the system behaves in terms of stability, accessibility, and protection of user data.

Reliability:

- The system shall function consistently during regular use without frequent crashes or failures.
- The system shall handle sensor readings accurately to avoid false alerts or missed detections.

Availability:

- The system shall maintain a stable Bluetooth connection within typical indoor training ranges (approximately 10 meters).
- The mobile application shall reflect updates within a few seconds of detecting gear status changes.

Security:

- The system should use basic encryption for Bluetooth communication to minimize interception risks.
- The application shall allow only authorized users to access gear status and history.

3.5 User Interface Prototype

The user interface (UI) of a system plays a crucial role in shaping the user experience, as it represents the point of interaction between the user and the system's features. A well-designed UI enhances usability, accessibility, and user satisfaction, ultimately improving the overall system efficiency and effectiveness. According to Benyon (2014), the user interface is where interaction happens, and its quality directly impacts how users perceive and interact with the system [38]. For this project, the UI prototype was designed using Figma, a web-based collaborative design tool that allows teams to create, share, and test interface designs in real time [39]. Figma is especially effective for UI/UX design due to its support for interactive prototyping, component-based design, and cloud collaboration. The following screens were developed as part of the mobile application interface for the ProGear Smart Bag system. Each screen is labeled with a descriptive name and a short explanation of its purpose and key features, demonstrating how the interface directly supports the system's requirements.

1.Welcome Screen

Introduces the user to the application with a clean, friendly interface and offers options to Sign in or Register.

Key Features:

- Brand identity display (logo and app name)
- Access to authentication flow

Related Requirements:

- FR1: "The system shall allow the user to register or log in to the mobile application"

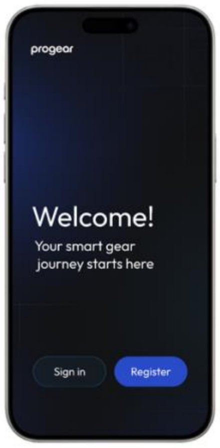


Figure 20 Welcome Screen

3.Sign In Screen

Allows the user to securely enter their credentials and access their account.

Key Features:

- Input fields for username and password
- Password visibility toggle
- Forgot password option
- Link to create a new account

Related Requirements:

- FR1: User authentication and access

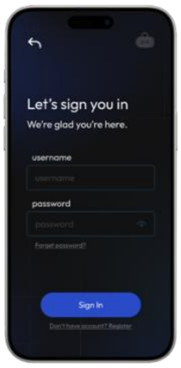


Figure 21 Sign in Screen

3.Bluetooth Instructions screen

Guides the user through the steps to connect the smart bag with their phone via Bluetooth.

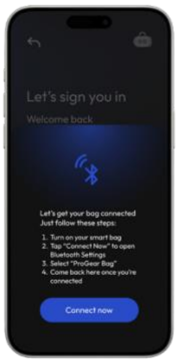
Key Features:

Step-by-step connection guide

- Interactive Connect Now button

Related Requirements:

- FR2: "The system shall establish a Bluetooth connection between the smart bag and the mobile application"



Let's get your bag connected
Just follow these steps:

1. Turn on your smart bag
2. Tap "Connect Now" to open Bluetooth Settings
3. Select "ProGear Bag"
4. Come back here once you're connected

Figure 22 BT Instructions Screen

4.Searching / Failed Connection Screen

Indicates the app is actively searching for the smart bag via Bluetooth.

Key Features:

- Animated feedback
- Retry option if connection fails

Related Requirements:

- FR2: Connection initiation and feedback

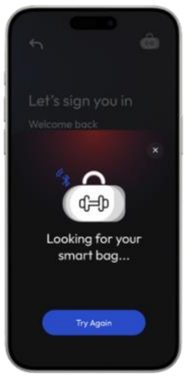


Figure 23 Searching | Failed Connection Screen

5. Connection Success Screen

Confirms that the smart bag is now connected to the mobile application.

Key Features:

- Confirmation checkmark animation
- Navigation to the next step

Related Requirements:

FR2: Confirmation of Bluetooth pairing

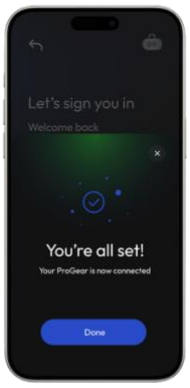


Figure 24 Connection Success Screen

6. Items Settled Screen

Allow the user to define the expected total weight of the items they usually carry. This is used as a baseline for future comparisons.

Key Features:

- Instructional message to guide the user
- "Done" button to confirm and save expected weight

Related Requirements:

- FR3: Weight measurement
- FR5: Compare and detect missing items

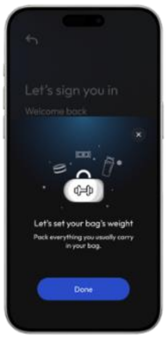

Let's sign you in
Welcome back
Let's set your bag's weight
Pack everything you usually carry in your bag.
Done

Figure 25 Item Settled Screen

7. Reset Weight Screen

Gives the user the ability to reset or update their expected bag weight by packing their usual items again.

Key Features:

- Instruction to pack the bag
- "Cancel" and "Done" buttons for flexibility
- Updates the baseline weight used for comparisons

Related Requirements:

- FR3: Weight measurement
- FR5: Compare and detect missing items

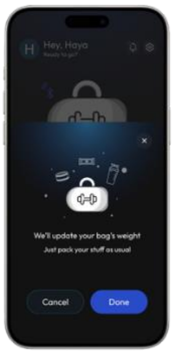


Figure 26 Reset Weight Screen

8. Home Dashboard Screen
Main control panel showing real-time information about the smart bag
Key Features:

- Connected status display
- Weight status bar (real-time sensor data)
- Visual alert for missing items
- Battery level indicator
- Reset expected weight button

Related Requirements:

- FR2: Bluetooth status
- FR3: Weight measurement
- FR4: Real-time data transmission
- FR5: Compare and detect missing items

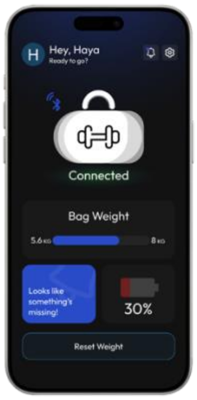


Figure 27 Home Dashboard Screen

3.6 System Design

3.6.1 Architectural Design

Architectural design is considered a critical phase in system development, as it defines the overall structure and interaction of hardware and software components. As per IEEE Std 1471-2000, "architectural design is the fundamental organization of a system, embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" [40]. In our proposed system, ProGear Smart Bag, the architecture consists of both hardware and software components that work together to deliver smart gear tracking functionality. The hardware components include the ESP32 microcontroller, which acts as the processing unit, HX711 load cell amplifier modules, and load sensors, which measure the weight of the equipment. On the software side, the system includes a mobile application, responsible for displaying weight data, sending notifications, and offering user interaction capabilities. To facilitate communication between these components, wireless connectivity via Bluetooth is utilized. The ESP32 collects weight data from the sensors and sends it to the mobile application, forming a seamless data exchange process. This connection serves as the interface between the hardware and software layers, ensuring real-time synchronization and user awareness of missing or forgotten items.

The importance of architectural design lies in its ability to define a scalable, maintainable, and efficient framework for system development. A well-structured architecture simplifies implementation, supports future enhancements, and ensures that system requirements—such as real-time feedback, accuracy, and usability—are met [41][42]. Furthermore, establishing clear interfaces between components reduces the likelihood of system integration issues and enhances overall system reliability [43].

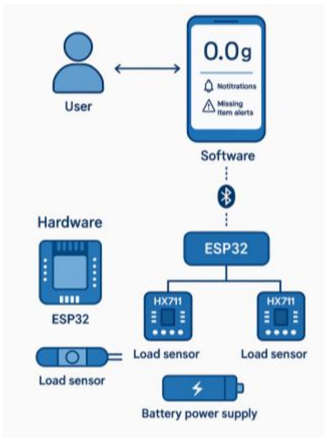


Figure 28 Architectural Design

This diagram illustrates the architectural design of the ProGear Smart Bag, showcasing the interaction between hardware and software components. At the hardware level, the system uses two load sensors connected to HX711 modules for weight amplification and digital conversion. These modules are linked to an ESP32 microcontroller, which acts as the central processing unit, managing sensor data and handling communication.

Through Bluetooth connectivity, the ESP32 transmits the collected weight data to the mobile application, which is part of the software layer. The application provides a user-friendly interface that allows the user to view the total weight, receive real-time notifications, and detect missing items based on weight thresholds. The system is powered by a battery, which makes it easy for athletes to carry and use the smart bag anywhere without needing to plug it into a power source. This portability is important for users who are constantly moving between training locations.

The design of the system is divided into clear layers — one for the hardware components like the sensors and microcontroller, and another for the software like the mobile app. This layered design makes the system easier to understand, develop, and expand in the future. For example, new features can be added to the app without changing the hardware. Also, this structure ensures that all parts of the system can communicate smoothly with each other, which helps the bag accurately detect and report missing items.

3.6.2 Class Diagram

A class diagram is a structural diagram in the Unified Modeling Language (UML) that describes the static structure of a system by showing its classes, attributes, operations, and the relationships among objects [44]. It plays a critical role in object-oriented analysis and design by offering a clear blueprint of how software components are defined and interact [45]. This diagram is particularly valuable during the early phases of system modeling, as it helps in identifying responsibilities, understanding class collaborations, and ensuring that the system design aligns with its functional requirements [46]. Furthermore, class diagrams aid in communication among developers and stakeholders, making the development process more structured and maintainable [47]. In the context of the ProGear Smart Bag system, the class diagram models the essential components—such as the User, ESP32Controller, and Weight Sensor—and shows how they collaborate to enable functionalities like authentication, data collection, and automated notifications. This representation serves as a foundation for implementation and ensures a consistent understanding of the system's core architecture [48].

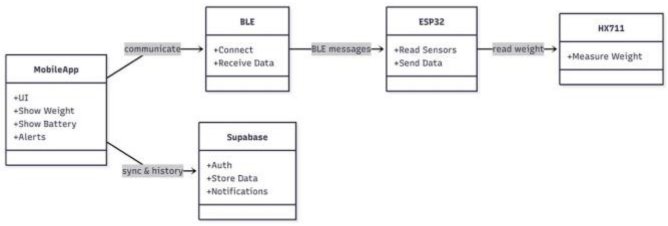


Figure 29 Class Diagram

The class diagram of the ProGear Smart Bag system illustrates the structural components that enable its smart, interconnected functionality by showing how data flows from the physical sensor to the mobile application and cloud services. At the core of the system, the ESP32Controller class functions as the primary processing unit responsible for handling sensor communication, reading values from the HX711 module, and transmitting processed weight data. It interacts directly with the WeightSensor class, which represents the HX711 load-cell interface responsible for capturing the actual weight readings through its readWeight() operation. The ESP32Controller then forwards this data using Bluetooth Low Energy (BLE), modeled through a communication layer that enables the controller to send and receive messages from the MobileApp class. The MobileApp class encapsulates all user-facing functionality, including showing real-time weight, displaying battery level, sending alerts, and managing Bluetooth connection states. Beyond local communication, the MobileApp also synchronizes user data with Supabase, represented as a backend class responsible for authentication, storing historical weight information, and managing notification records. This interconnected set of classes reflects a cohesive architecture in which the sensor provides raw measurements, the ESP32 processes and transmits them, the mobile app interprets and displays them, and Supabase ensures secure data persistence and user management. Together, these classes create a seamless and reliable smart-bag experience centered around accurate monitoring and responsive user alerts.

3.6.3 Sequence Diagram

The Sequence Diagram is a standardized diagram type in the Unified Modeling Language (UML) that models the flow of messages exchanged between system components or objects over time [49]. It focuses on the chronological sequence of operations, showing how different system elements interact to carry out specific functionalities. Sequence diagrams are particularly useful for representing real-time behavior, method calls, and system logic in a way that is both visual and easy to trace. The importance of using sequence diagrams lies in their ability to clarify complex process flows, highlight object interactions, and validate the behavioral logic of a system. According to IBM Developer [50], sequence diagrams are essential in software design as they allow stakeholders to understand how tasks are performed across components and help developers ensure the correct execution order of operations. Additionally, they support documentation and testing by outlining precise interaction scenarios between objects.

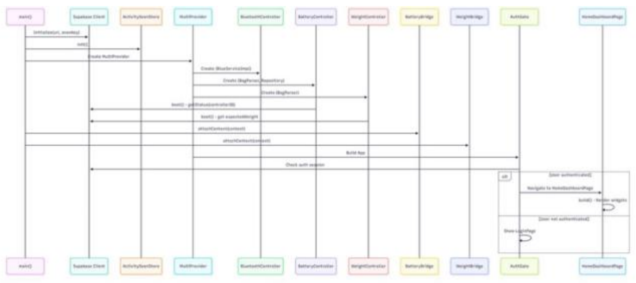


Figure 30 App Initialization and Home Dashboard

The sequence diagram for the "App Initialization and Home Dashboard Flow" illustrates how the ProGear application starts, prepares its internal components, and determines whether the user should be directed to the Home Dashboard or the Login Page. The process begins when the main() function initializes the application by configuring the Supabase client using the required URL and API key. After the initialization, the system creates multiple controllers and providers—such as the BluetoothController, WeightController, BatteryController, and their associated bridges and parsers. Each component is instantiated to ensure the system can communicate with the smart bag, read weight and battery values, and manage application state.

As the initialization continues, each controller attaches the required application context, allowing them to operate correctly throughout the app lifecycle. Once all components are ready, the system proceeds to check the current authentication session through the AuthGate. At this point, an alt frame is used to represent two alternative outcomes: if the user is already authenticated, the flow navigates directly to the HomeDashboardPage, where the app finishes building the UI and renders the necessary widgets. If the user is not authenticated, the system instead displays the LoginPage, prompting the user to sign in before accessing any functionality.

This sequence ensures a structured and reliable startup process, guaranteeing that all essential system components are fully initialized before the user interacts with the application.

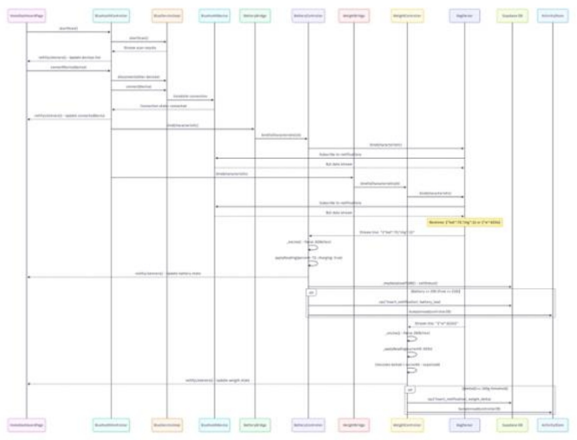


Figure 31 Bluetooth Connection and Data Streaming

The sequence diagram for the “Bluetooth Connection and Data Streaming” flow illustrates how the mobile application discovers the ProGear Smart Bag, establishes a Bluetooth Low Energy (BLE) connection, and continuously receives real-time weight and battery data from the ESP32 controller. The process begins when the HomeDashboardPage initiates a scan request, prompting the BluetoothController to start scanning for nearby devices. As scan results are streamed back, the app updates the device list for the user.

When the user selects the ProGear device, the system disconnects any previous connections and establishes a new one through the BlueService implementation. Once connected, characteristics related to weight and battery readings are bound and subscribed to, enabling continuous BLE notifications. The incoming data stream is then parsed by the respective bridges: BatteryBridge processes battery percentage and status, while WeightBridge extracts the raw weight readings.

After parsing, the BatteryController and WeightController update their states and notify the UI. Additional logic is applied, such as checking low-battery thresholds or calculating weight differences to detect missing items. If specific conditions are met—such as critically low battery or a significant drop below the expected weight—the system triggers remote procedure calls (RPCs) to Supabase to store alerts and update the bag's status. This coordinated flow ensures accurate, real-time monitoring of the smart bag through seamless Bluetooth communication.

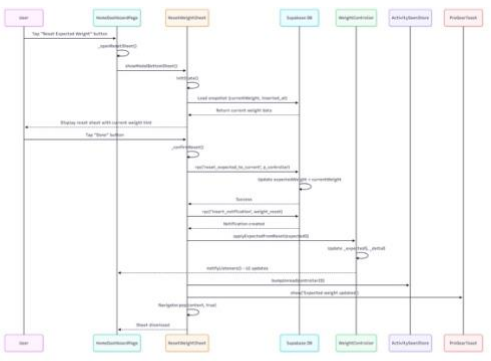


Figure 32 Reset Expected Weight

The sequence diagram for the "Reset Expected Weight" flow illustrates how the system updates the bag's reference weight when the user chooses to recalibrate it. The process begins when the user taps the "Reset Expected Weight" button on the Home Dashboard, prompting the app to open a reset bottom sheet displaying the current measured weight as a hint. Once the sheet is initialized, it retrieves the latest weight snapshot from the database and presents it to the user.

When the user confirms by pressing "Done," the reset process is triggered. The app calls a remote procedure (RPC) to update the expected weight in the Supabase database so that the new expected value matches the current measured weight. After the database successfully applies the change, a notification record is created to log that the weight has been reset.

The updated expected weight is then passed to the WeightController, which recalculates internal variables and notifies all listeners to refresh the UI accordingly. Following the update, the bottom sheet closes automatically, and the app displays a confirmation toast indicating that the expected weight has been successfully updated.

This sequence ensures a smooth and accurate recalibration process by coordinating user input, backend synchronization, controller updates, and user interface feedback in a clear and structured flow.

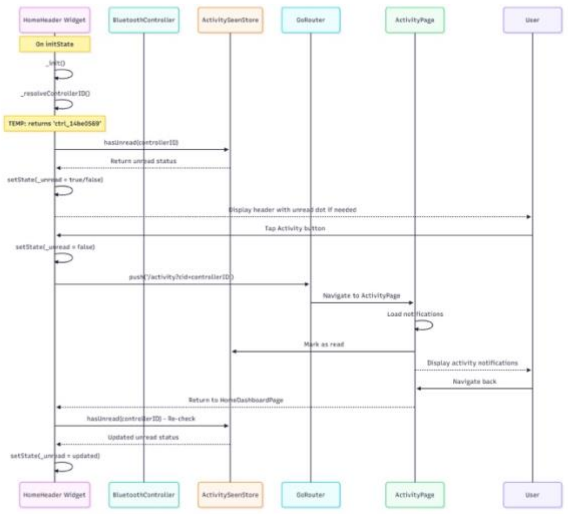

HomeHeader Widget
BluetoothController
ActivitySeenStore
GoRouter
ActivityPage
User
On initState
_init()
_resolveControllerID()
TEMP: returns 'ctrl_14be0569'
hasUnread(controllerID)
Return unread status
setState(_unread = true/false)
Display header with unread dot if needed
Tap Activity button
setState(_unread = false)
push('/activity?cid=controllerID')
Navigate to ActivityPage
Load notifications
Mark as read
Display activity notifications
Navigate back
Return to HomeDashboardPage
hasUnread(controllerID) - Re-check
Updated unread status
setState(_unread = updated)

Figure 33 Home Header Activity Navigation

The sequence diagram illustrates the process of displaying and updating the unread activity status within the Home Header of the ProGear application. The flow begins when the HomeHeaderWidget initializes and requests the associated controller ID through the BluetoothController. Once the controller ID is resolved, the widget sends a request to ActivitySeenStore to check whether the user has any unread activity notifications. The response is returned as either true or false, allowing the UI to update and display a notification indicator when necessary.

When the user taps the Activity button in the header, the system navigates to the ActivityPage via GoRouter and loads all notification records for the associated controller. As part of this process, ActivitySeenStore marks the viewed notifications as read. After reviewing the activities, the user navigates back to the HomeDashboardPage.

Upon returning, the widget triggers a second unread-status check to ensure the header accurately reflects the updated read state. The ActivitySeenStore provides the new unread status, and the HomeHeaderWidget updates the UI accordingly. This flow ensures that notification indicators are always synchronized with the user's interaction and that the system provides real-time feedback on unread activity.

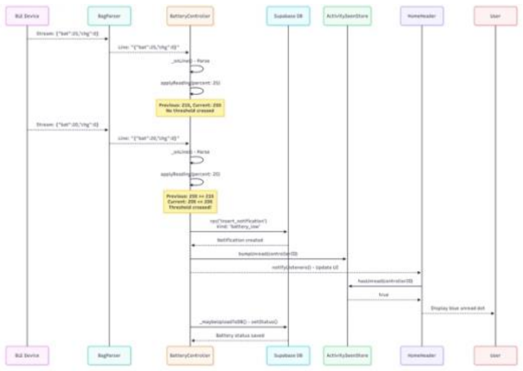


Figure 34 Battery Low Notification

The sequence diagram for the "Battery Low Notification" flow illustrates the interaction between the BLE Device, BagParser, BatteryController, Supabase Database, ActivitySeenStore, HomeHeader, and the User. The process begins when the BLE device streams battery percentage data to the system, which is then parsed by the BagParser and forwarded to the BatteryController. The controller compares the newly received percentage with the previous stored value to determine whether a threshold has been crossed. An alt frame is used to represent two possible outcomes: if the battery level remains above the threshold, no further action is taken; however, if the level drops to or below the defined limit, the system triggers a low-battery notification. Once triggered, the BatteryController communicates with the Supabase database to insert a notification record, after which ActivitySeenStore updates the unread status. The HomeHeader then reflects this change by displaying an unread indicator for the user. This structured sequence highlights how the system continuously monitors battery levels and ensures that the user is promptly informed when charging becomes necessary.

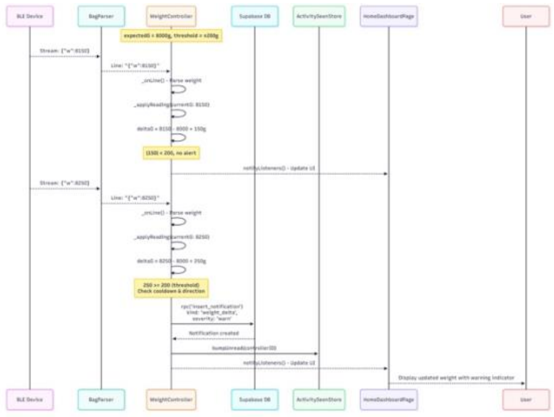


Figure 35 Weight Delta Alert

The sequence diagram for the “Weight Delta Alert” flow illustrates how the system processes incoming weight readings from the BLE device and determines whether a weight-change alert should be triggered. When a new weight value is streamed, the BagParser forwards the raw reading to the WeightController, which parses the value and updates the current measured weight. The system then calculates the difference between the expected weight and the newly received value. If the delta remains below the defined threshold, no alert is triggered and the UI simply updates the displayed bag weight. However, when the difference exceeds the threshold limit, the sequence enters an alternative path where the system validates alert conditions such as cooldown timing and weight-change direction. Once confirmed, a notification record is inserted into the database to indicate a weight-delta event, and the unread notification count is updated accordingly. The UI then reflects the change by displaying the updated weight along with a warning indicator, ensuring that the user is immediately informed of a potential missing or added item.

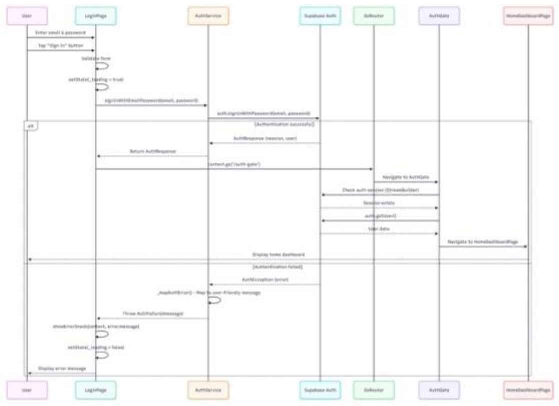

User
LoginPage
AuthService
Supabase Auth
GoRouter
AuthGate
HomeDashboardPage
Enter email & password
Tap "Sign In" button
Validate form
setState(_loading = true)
signInWithEmailPassword(email, password)
auth.signInWithPassword(email, password)
alt
[Authentication successful]
AuthResponse (session, user)
Return AuthResponse
context.go('/auth-gate')
Navigate to AuthGate
Check auth session (StreamBuilder)
Session exists
auth.getUser()
User data
Navigate to HomeDashboardPage
Display home dashboard
[Authentication failed]
AuthException (error)
_mapAuthError() - Map to user-friendly message
Throw AuthFailure(message)
showErrorSnack(context, error.message)
setState(_loading = false)
Display error message

Figure 36 User Login

The sequence diagram for the “User Login” flow illustrates the interaction between the user interface, the authentication service, and the navigation components when a user attempts to sign in. In this scenario, the process begins when the user enters their email and password on the LoginPage and taps the “Sign In” button. The page validates the form, shows a loading state, and forwards the credentials to the AuthService, which in turn calls Supabase Auth to perform the actual authentication. An alt frame is used to represent two alternative outcomes. In the successful branch, Supabase returns a valid session and user data, the AuthService passes the response back, and the router navigates to the AuthGate, which verifies the session and then redirects the user to the HomeDashboardPage. In the failure branch, Supabase returns an error, the AuthService maps this error to a user-friendly message, and the login page displays the message while removing the loading state. This structured sequence clarifies how the system validates credentials, handles errors gracefully, and controls access to the main application.

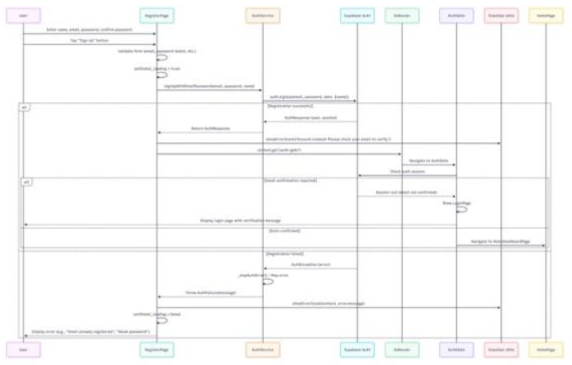


Figure 37 User Registration

The sequence diagram for the User Registration use case illustrates how new users create an account. The process begins when the user enters their personal details and submits the registration form. The system validates the input and sends the registration request to the authentication service. An alternative (alt) frame is used to represent different outcomes: if registration is successful, the system may require email verification before allowing access; if auto-confirmation is enabled, the user is redirected directly to the home screen; if registration fails, an error message is shown. This sequence ensures that account creation is handled securely and consistently.

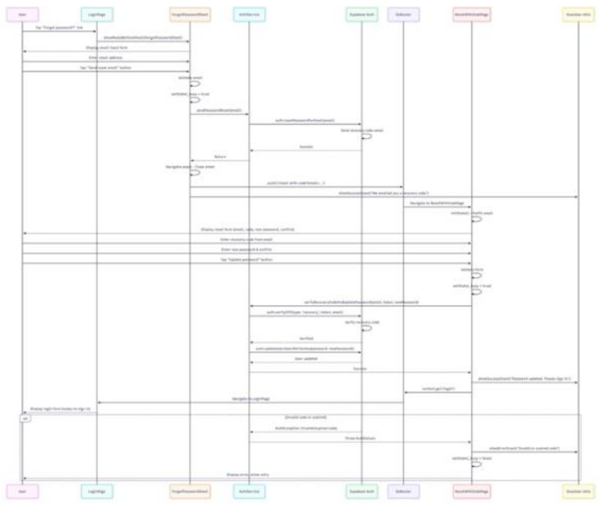
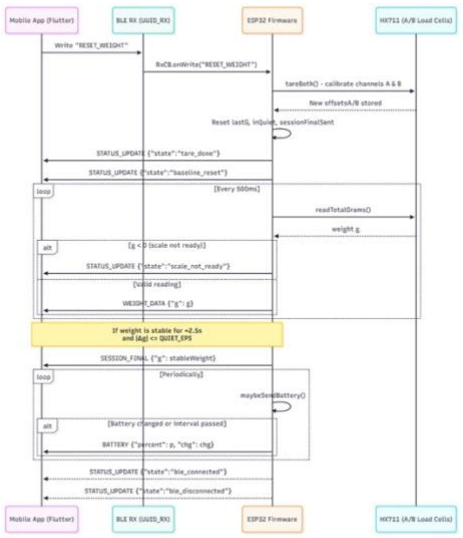


Figure 38 Reset Password

The sequence diagram for the "Password Reset" use case shows how users recover access to their accounts. The process begins when the user selects the "Forgot password" option and submits their email address. The system sends a recovery code to the user’s email and navigates them to the reset page. The user enters the recovery code and a new password, which is then verified and updated through the authentication service. If the process succeeds, the user is redirected to the login page with a success message; if the code is invalid or expired, an error message is shown. This flow ensures a secure and user-friendly password recovery mechanism.


Mobile App (Flutter)
BLE RX (UUID_RX)
ESP32 Firmware
HX711 (A/B Load Cells)
Write "RESET_WEIGHT"
RxCB.onWrite("RESET_WEIGHT")
tareBoth() - calibrate channels A & B
New offsetsA/B stored
Reset lastG, inQuiet, sessionFinalSent
STATUS_UPDATE {"state":"tare_done"}
STATUS_UPDATE {"state":"baseline_reset"}
loop
[Every 500ms]
readTotalGrams()
weight g
alt
[g < 0 (scale not ready)]
STATUS_UPDATE {"state":"scale_not_ready"}
[Valid reading]
WEIGHT_DATA {"g": g}
If weight is stable for ~2.5s
and |Δg| <= QUIET_EPS
SESSION_FINAL {"g": stableWeight}
loop
[Periodically]
maybeSendBattery()
alt
[Battery changed or interval passed]
BATTERY {"percent": p, "chg": chg}
STATUS_UPDATE {"state":"ble_connected"}
STATUS_UPDATE {"state":"ble_disconnected"}
Mobile App (Flutter)
BLE RX (UUID_RX)
ESP32 Firmware
HX711 (A/B Load Cells)
Figure 39 Hardware Weight & BLE Streaming

The sequence diagram for the Hardware Weight & BLE Streaming flow illustrates how the mobile application communicates with the ESP32 firmware and the HX711 load cell sensors to reset, measure, and continuously stream weight data. The process begins when the mobile app sends a RESET_WEIGHT command through the BLE RX characteristic, which is received by the ESP32 firmware. Upon receiving this command, the firmware triggers the tare operation on both HX711 channels to recalibrate the load cells and store new offset values, effectively resetting the baseline weight. The ESP32 then sends status updates back to the mobile application to indicate that the tare and baseline reset operations have been completed. After the reset, the firmware enters a periodic measurement loop where it reads the total weight from the HX711 sensors at fixed intervals. If the scale is not ready, a corresponding status message is sent; otherwise, valid weight readings are transmitted to the mobile app through BLE. Once the system detects that the weight has remained stable for a defined duration and within an acceptable delta threshold, a final stable weight value is sent to mark the end of the measurement session. In parallel, the firmware periodically checks the battery state and sends battery updates when changes are detected or after a defined interval. This sequence highlights the real-time interaction between the mobile app, BLE communication layer, ESP32 firmware, and hardware sensors to ensure accurate weight measurement, stability detection, and continuous status reporting.

3.6.4 Database Design (if any)

A database is an organized collection of structured data that allows for efficient storage, retrieval, and management of information within a system [51]. In our Smart Bag project, the database serves as a backbone for maintaining critical information such as user accounts, real-time weight measurements, and notification logs. The use of a relational database enhances data integrity, minimizes redundancy, and ensures consistency across components like users, sensors, controllers, and notifications. By implementing entity relationships and foreign key constraints, we ensure that each part of the system communicates seamlessly, and data is accurately tracked, even in complex interactions. Utilizing a database in this project is essential to enable scalability, traceability, and reliable synchronization between hardware and software layers [52].

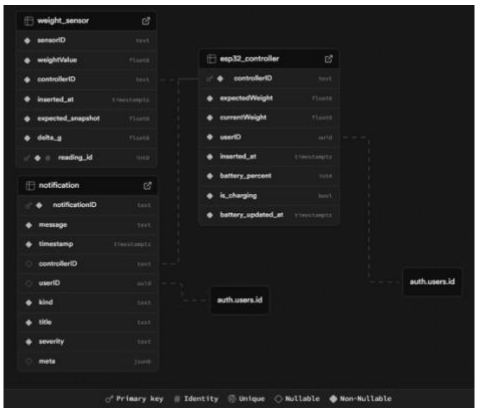


Figure 40 ER Diagram

The ER diagram illustrates how the main components of the Smart Bag system are connected and how data is stored and linked together. The User table stores basic information about each user, including their login details and email. Each user is connected to exactly one ESP32 Controller, which is the device inside the bag that monitors weight. The ESP32 Controller stores both the expected and current weight and is linked to one WeightSensor, which reads the actual weight from the bag. When the system detects a weight change or an issue, it creates a Notification. This notification is connected to both the user who should receive it and the controller that generated it. This structure helps the system track who owns each bag, what is happening inside it, and alert the right user when something important happens. It also ensures that data is organized and easy to retrieve or update when needed.

3.7 Summary

Chapter 3 provided a detailed and structured overview of the system analysis and design phase for the ProGear Smart Bag. This chapter played a crucial role in translating the conceptual idea into a clear and technically feasible solution by outlining all the essential elements required for system development. It began with a description of the system's overall perspective, clarifying its role within the user environment and how it interacts with external components and users. The chapter then focused on the requirements elicitation techniques, highlighting the use of questionnaires and brainstorming sessions to gather insights directly from the target audience—powerlifting athletes. These methods enabled the identification of user preferences, pain points, and expectations, which were then translated into a comprehensive list of system requirements. These requirements were categorized into functional requirements—such as equipment detection and notification features—and non-functional requirements—such as usability, performance, and reliability. This distinction ensured a complete understanding of both what the system should do and how it should perform .A user interface prototype was then introduced to visualize how users would interact with the mobile application, ensuring that the design supports an intuitive and seamless user experience. Following this, the system's design architecture was thoroughly discussed, presenting both the high-level architectural design and the detailed class and sequence diagrams that map the logical structure and behavior of the system. These diagrams clarified the internal processes and the flow of data within the system, bridging the gap between analysis and implementation. The chapter also included a database design section, where applicable, to manage and store essential data effectively. Overall, this chapter laid the foundation for the implementation phase by providing a clear blueprint of how the ProGear Smart Bag is expected to function in the real world. It ensured that each design decision was informed by actual user needs and grounded in practical feasibility, increasing the likelihood of delivering a successful and impactful solution for athletes who seek smarter and more organized training routines.

Chapter Four: Implementation

4 Implementation

4.1 Introduction

This chapter documents how the system designed in Chapter 3 was implemented as a working prototype of the ProGear Smart Bag. In the previous chapter, the problem was defined, the functional and non-functional requirements were finalized, and the overall architecture was presented with its main components: load sensors, the HX711 amplifier, the ESP32 microcontroller, Bluetooth Low Energy (BLE), and a Flutter-based mobile application. Load sensors (load cells) are transducers that convert the mechanical force of the equipment weight into electrical signals. The HX711 is a 24-bit analog-to-digital converter specifically designed for weigh-scale applications, used to digitize the load-cell output. The ESP32 is a low-power embedded microcontroller with integrated Bluetooth capabilities, responsible for reading sensor data, applying calibration and filtering, and managing communication with the mobile device. Bluetooth Low Energy (BLE) is a short-range wireless communication standard used to transmit the processed readings from the ESP32 to the smartphone. Flutter is an open-source user interface software development kit (SDK) that enables building cross-platform mobile applications from a single Dart codebase; it is used here to implement the ProGear mobile interface.

The chapter moves from describing "how the system should work" to explaining "how the system was actually built", covering the concrete implementation choices, configurations, and integration steps that enabled real-world operation. At a high level, the implementation consists of two main tracks that were developed in parallel and then integrated:

1.Hardware and firmware implementation.

Two load cells were wired to HX711 amplifiers, which were then connected to the ESP32. The firmware, written in C/C++, initializes the peripherals, reads and filters the raw weight signals, maintains an "expected weight" baseline, and detects significant weight drops that may indicate missing items. A lightweight BLE service publishes status messages, weight readings, and battery information to the mobile application. Practical issues such as calibration drift, sensor noise when the bag is moved, numerical stability for very small changes, and power consumption were explicitly addressed in the firmware logic.

2.Mobile application implementation.

A cross-platform application was developed using Flutter and the Dart programming language to guide the user through Bluetooth connection, display live readings, and raise alerts. The application implements the screens prototyped in Chapter 3 (connection/search/success screens, items-settled screen, reset-weight screen, and dashboard). Simple but robust state-management mechanisms keep the user interface reactive to BLE updates, while guardrails handle common edge cases such as connection loss, stale readings, or out-of-range values.

To ensure traceability and alignment with the system requirements, the remainder of this chapter is organized into the following sections:

- **Programming Languages and Tools:**

Describes the use of C/C++ for the ESP32 firmware and Dart/Flutter for the mobile application, as well as the development environments (Arduino IDE, Visual Studio Code, and Android Studio) that supported the implementation workflow.

- **User Interface Implementation:**

Presents the final mobile screens and interaction flows and briefly explains how each interface element relates to the functional requirements.

- **Database Implementation:**

Explains how persistent data—such as the expected-weight baseline and configuration parameters—is stored on the ESP32 (non-volatile memory) and within the mobile application.

- **Packages and Classes Description:**

Summarizes the main software modules and classes in both the firmware and the Flutter project, clarifying their responsibilities and interactions.

- **Procedures Description:**

Highlights the most critical procedures and algorithms, including calibration, missing-item detection, BLE message handling, and battery-status estimation.

- **Summary:**

Concludes the chapter by tying the implementation decisions back to the system requirements and preparing the discussion for the testing and results presented in Chapter

The implementation goal is to realize the core scenario end-to-end: once the smart bag is connected, the system measures the total weight, compares it with the stored baseline, and immediately notifies the user through the mobile application if a significant drop is detected. Each implemented component maps directly to the functional requirements defined earlier—Bluetooth pairing and status reporting, real-time weight measurement and transmission, and automatic detection and notification of potential missing equipment.
From a practical perspective, the implementation prioritizes responsiveness and clarity for athletes preparing quickly before a training session; therefore, the user interface emphasizes a single main status indicator, a clear connection flow, and a straightforward reset action. On the firmware side, simple and robust filtering techniques are preferred over complex models to keep readings stable while the bag is handled. Detection logic is based on a transparent threshold relative to the baseline set during the "items settled" step, reducing the risk of misclassifying natural fluctuations as missing items. The following sections describe the toolchain, interfaces, and embedded logic that collectively deliver the core functionality of the ProGear Smart Bag.

4.2 Programming Languages and Tools

Programming Languages

The implementation of the ProGear Smart Bag system required multiple programming languages, each selected for its suitability to a specific system component. The combination of these languages supported reliable communication among the mobile application, the embedded controller inside the bag, and the optional backend services.

- **Dart (Flutter Framework)**

Dart was used to develop the ProGear mobile application. As the underlying language of the Flutter framework, Dart enables building reactive and cross-platform interfaces from a single codebase. This made it suitable for displaying real-time weight readings, battery levels, and Bluetooth connection status. Flutter's rendering engine allowed smooth transitions and instantaneous UI updates upon receiving new data from the hardware.

- **SQL (PostgreSQL – Supabase)**

Structured Query Language (SQL) was employed to manage the cloud database hosted on the Supabase platform. SQL enabled the creation of relational tables (e.g., device logs and notifications), enforcement of Row-Level Security (RLS), implementation of triggers, and validation of stored data. PostgreSQL's reliability in handling structured information ensured accurate tracking of device records and historical measurements.

- **Arduino C++ (ESP32 Firmware)**

The firmware of the embedded system was written in Arduino C++. This language allowed direct control of the ESP32 microcontroller, which is a low-power embedded processor with built-in Wi-Fi and Bluetooth Low Energy (BLE). Arduino C++ was used to read the analog signals from the HX711 load-cell amplifiers, apply calibration and filtering, monitor battery levels, detect significant weight drops, and transmit the processed data to the mobile application through BLE. The ESP32 also stored the baseline "expected weight" using its non-volatile memory.

Software Tools

- **Flutter SDK**

Flutter provided the foundation for building the mobile interface. Its widget-based architecture enabled the creation of adaptive screens, material components, and reactive views that update automatically with new BLE data.

- **Supabase Platform**

Supabase served as the backend service when cloud functionality was required. It offered authentication, secure PostgreSQL storage, near real-time synchronization, and Remote Procedure Calls (RPCs). These features supported long-term logging and safe data retrieval.

- **Bluetooth Low Energy (BLE)**

BLE acted as the communication layer between the ESP32 and the mobile application. BLE is a low-energy wireless protocol suitable for small, frequent data transmissions. Custom components were implemented inside the application for scanning, connecting, exchanging commands, and decoding JSON messages received from the firmware. Key modules included: BluetoothController, BlueServiceImpl, BagParser, BatteryBridge, WeightBridge.

- **GitHub**

GitHub was used for version control, repository management, documentation storage, and branch-based development. It ensured proper tracking of firmware and application updates.

- **Jira**

Jira supported project planning and team task allocation. It facilitated sprint management, progress tracking, and documentation of software- and hardware-related issues.

- **Visual Studio Code & Android Studio**

Visual Studio Code was used mainly for editing Flutter and firmware code, while Android Studio provided emulators, device debugging tools, and performance optimization features.

- **Figma (UI/UX Prototyping)**

Figma was used to design the interface layouts, wireframes, and prototypes before implementation. This ensured consistent visual structure across all application screens.

Hardware Tools

- **ESP32 Microcontroller**

The ESP32 functioned as the main controller of the smart bag system. It processed sensor readings, executed the detection logic, managed BLE communication, and stored configuration values. Its built-in BLE module eliminated the need for external wireless hardware.

- **HX711 Load-Cell Amplifier**

The HX711 is a 24-bit precision analog-to-digital converter specialized for weigh-scale applications. It amplifies and digitizes the small voltage changes from the load cells. The firmware-controlled gain settings, reading cycles, offset calibration, and noise filtering.

- **Load Cells (Weight Sensors)**

Two load cells were installed to measure the total weight of items inside the bag. These sensors convert mechanical force into electrical signals, enabling detection of small weight differences. Their outputs feed directly into the HX711 amplifiers to obtain stable measurements.

- **Battery & Power Components**

A rechargeable battery powered the ESP32 and sensors. Power stability, voltage regulation, and battery-level monitoring were essential to guarantee correct operation during use.

- **Prototyping Tools (Breadboard, Wiring, Connectors)**

During development, a breadboard and modular wiring were used for testing sensor placement, verifying circuit behavior, and adjusting hardware connections before final integration into the bag enclosure.

4.3 User Interface Implementation

Tools and Programming Languages

The user interface (UI) for the ProGear Smart Bag mobile application was developed as a cross-platform Flutter application, utilizing the following core technologies that enabled responsive layouts, real-time updates, and seamless interaction with the smart bag hardware

- **Dart (Flutter Framework)**

Dart was used as the primary programming language for building the UI of the mobile application. Its reactive model allowed the interface to update instantly when new weight or battery data is received. Dart's clean syntax and strong integration with Flutter supported the creation of modern, smooth, and responsive screens suitable for both iOS and Android devices.

- **Flutter SDK**

Flutter served as the main toolkit for implementing all UI components. Through its widget-based architecture, we developed:

- Material and custom UI widgets (buttons, dialogs, cards, bottom sheets)
- Responsive interfaces that adapt automatically to different screen sizes
- Navigation using Flutter's built-in routing
- Smooth animations and transitions
  Flutter's unified codebase significantly improved development speed and ensured visual consistency across platforms.

- **Provider (State Management)**

Provider was used to manage real-time UI updates across the application. Weight readings, battery status, and Bluetooth connection states are all updated through Provider, ensuring that the user always sees accurate, real-time information without delays or screen reloads.

- **Custom BLE Services (UI Layer)**

Although BLE communication is handled at the application logic level, the UI integrates directly with BLE state streams to:

- Display live weight values
- Update battery percentage
- Show connection status indicators
  This real-time feedback is essential for monitoring the smart bag's status during use.

- **Supabase Authentication (For Login UI Only)**

Supabase Auth was used to implement the login interface. The UI communicates with Supabase to validate credentials and manage user sessions. While the rest of the application operates offline through BLE, authentication is required for secure account access.
Below are the main user interface screens of the ProGear Smart Bag mobile application, each accompanied by its purpose and detailed description.

1.Login Screen

Screenshot:

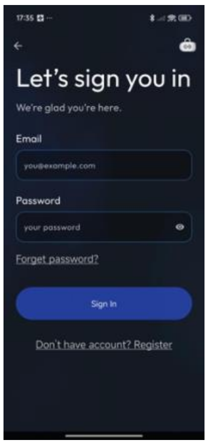


Figure 41 Login

Description:
This screen allows the user to securely sign in using Supabase Authentication. It includes input validation, error messages, and clean spacing to ensure a smooth and simple login flow. The design focuses on quick access and clarity, supporting secure entry into the app.

2.Bluetooth Scan & Connect Screen

Screenshot:

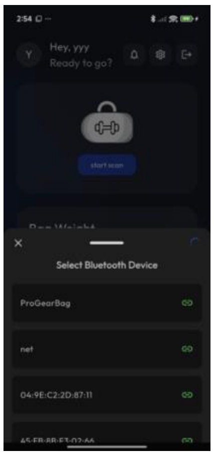


Figure 42 Bluetooth Scan & Connect

Description:

This screen enables scanning for nearby ESP32 devices, displaying their names, availability, and connection progress. Real-time indicators inform the user whether the device is connecting, connected, or unavailable. This interface is essential for establishing the communication link with the smart bag.

3.Home Dashboard Screen

Screenshot:

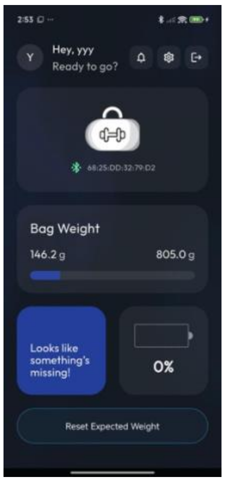


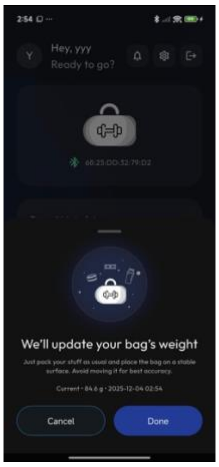
Figure 43 Home Dashboard

Description:

The dashboard displays the live weight, expected baseline weight, battery status, and connection state. Through Provider state management, all values update instantly as soon as BLE packets are received. This screen serves as the main monitoring hub for the user.

4.Reset Expected Weight Screen

Screenshot:



We'll update your bag's weight

Just pack your stuff as usual and place the bag on a stable surface. Avoid moving it for best accuracy.

Current · 84.6 g · 2025-12-04 02:54

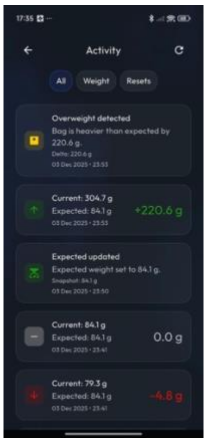
Figure 44 Reset Weight

Description:

This bottom-sheet interface allows the user to reset the expected baseline weight after placing equipment inside the bag. It shows the current measured weight and includes confirmation steps to prevent accidental resets. Haptic and visual feedback enhance user confidence.

5.Activity Center Screen

Screenshot:



17:35

Activity

All Weight Resets

Overweight detected
Bag is heavier than expected by 220.6 g.
Delta: 220.6 g
03 Dec 2025 · 23:53

Current: 304.7 g
Expected: 84.1 g
03 Dec 2025 · 23:53
+220.6 g

Expected updated
Expected weight set to 84.1 g.
Snapshot: 84.1 g
03 Dec 2025 · 23:50

Current: 84.1 g
Expected: 84.1 g
03 Dec 2025 · 23:41
0.0 g

Current: 79.3 g
Expected: 84.1 g
03 Dec 2025 · 23:41
-4.8 g

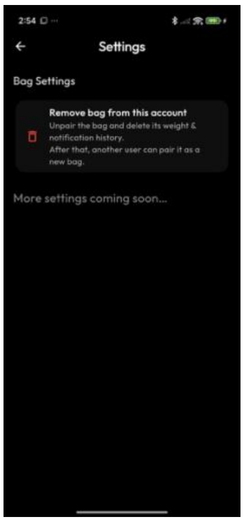
Figure 45 Activity Center

Description:

The Activity Center displays a chronological log of system notifications such as weight drops and resets. Each entry includes the timestamp and severity color coding. This screen helps users review past alerts and understand their interaction history with the smart bag.

6.Profile / Settings Screen

Screenshot:



2:54

Settings

Bag Settings

Remove bag from this account
Unpair the bag and delete its weight & notification history.
After that, another user can pair it as a new bag.

More settings coming soon...

Figure 46 Settings

Description:

This screen contains account settings and logout options. It ensures proper session management using Supabase Auth and provides access to personal account information.

Flutter and Dart were selected because they provide a fast, responsive, and cross-platform solution that is ideal for real-time interfaces. Provider ensures instant updates for dynamic values like weight and battery, while BLE services integrate smoothly with Flutter. Supabase Auth was chosen for its secure and lightweight authentication system suitable for mobile apps.

4.4 Database Implementation (if any)

Tools and Programming Languages Used:

The database component of the ProGear Smart Bag system was implemented using Supabase PostgreSQL, a cloud-hosted relational database solution that provides built-in real-time capabilities, authentication integration, and advanced security through Row-Level Security (RLS). The database is tightly integrated with both the mobile application (Flutter/Dart) and the ESP32 firmware, enabling synchronized updates for weight readings, expected-weight logic, notifications, and battery status.

Reasons for Choosing Supabase PostgreSQL:

- Strong Real-Time Sync:

Supabase provides real-time listeners that instantly reflect changes in weight readings, notifications, and controller status. This ensures that the mobile app always displays the most recent sensor data without requiring manual refresh.

- Security & RLS:

Supabase allows custom Row-Level Security (RLS) policies, ensuring that each user can only access their own smart bag controller and notifications. This is essential for a multi-user environment.

- Structured Data & Integrity:

Unlike NoSQL solutions, PostgreSQL ensures strong consistency and supports triggers, constraints, and stored procedures, which are necessary for accurate synchronization of weight data and automatic logic execution.

- Cloud-Hosted & Scalable:

Supabase provides a managed PostgreSQL instance with automatic scaling, making it suitable for IoT systems with growing data volumes such as continuous weight-sensor readings.

Source and Structure of the Dataset:

The ProGear Smart Bag uses a structured relational schema consisting of three main tables. These tables store:

- Smart bag controller metadata
- Real-time and historical weight readings
- System-generated notifications

Each new sensor reading or user-triggered action produces a structured record stored under its related controller, maintaining a full activity and weight-history log.

Database Schema

The system's schema consists of three core tables designed to maintain accurate synchronization between the hardware, mobile app, and cloud backend.

1.esp32_controller

Stores metadata for each physical smart bag controller.

Column	Type	Description
controllerID	TEXT (PK)	Unique identifier representing the ESP32 device
userID	UUID (FK)	Owner of the controller (from Supabase Auth)
expectedWeight	DOUBLE PRECISION	Target weight set by the user
currentWeight	DOUBLE PRECISION	Most recent weight reported by the ESP32
battery_percent	INTEGER	Battery level percentage (0–100)
is_charging	BOOLEAN	Whether the controller is charging
inserted_at	TIMESTAMPTZ	Timestamp of the last update

Table 4 Esp32_Controller

2.weight sensor

Stores detailed historical weight readings from the ESP32 sensors.

Column	Type	Description

reading_id	BIGINT (PK)	Auto-increment unique reading identifier
controllerID	TEXT (FK)	Links reading to a specific controller
weightValue	DOUBLE PRECISION	Raw weight value from the ESP32
expected_snapshot	DOUBLE PRECISION	Expected weight at the moment of reading
delta_g	DOUBLE PRECISION	Difference between expected and actual weight
inserted_at	TIMESTAMPTZ	Timestamp of the reading

Table 5 Weight Sensor

Trigger Logic (Smart Weight Synchronization)

A database-level trigger ensures that weight readings sent from the ESP32 remain synchronized with the controller's state.

Trigger Name:

fn_weight_sensor_on_insert()

Execution:

Runs before inserting a new record into weight_sensor.

Responsibilities:

- Copies the current expectedWeight into expected_snapshot
- Calculates delta_g (difference between expected and actual weight)
- Updates esp32_controller. currentWeight
- Maintains consistency between hardware readings and stored data

Benefits:

- The mobile app always displays the most updated weight
- Notifications fire instantly when a weight drop occurs
- All weight-related logic stays accurate and centralized

Row-Level Security (RLS Policies)

Supabase requires explicit RLS configuration to control access to data. RLS ensures each user interacts only with their own controller and activity history.

RLS Goals:

- A user can only read and update their own controllers
- A user can only insert readings for controllers they own
- Notifications are restricted to their respective owners

- Prevents cross-user data exposure across all tables

Examples of Implemented RLS Policies:

- SELECT from esp32_controller only where userID = auth.uid()
- INSERT into weight_sensor allowed only when controller belongs to auth.uid()
- SELECT notifications only where userID = auth.uid()

This provides a secure multi-user environment aligned with best practices for IoT systems.

RPC Functions (Cloud API Interface)

To simplify mobile-app logic and reduce direct SQL queries, several RPC functions were created.

Key RPC Procedures:

1. set_expected_weight(p_controller, p_value)

Updates expectedWeight for a given controller.

2.reset_expected_to_current(p_controller)

Sets expectedWeight = currentWeight. Used by the mobile app during the Reset Expected Weight operation.

3.insert_weight_reading(p_sensor, p_weight, p_controller)

Used by the app to insert a new weight reading event.

4.set_battery_status(p_controller, p_percent, p_charging)

Updates current battery percentage and charging status.

5.fetch_weight_history(p_controller)

Returns paginated historical readings for the Activity Center screen.

The database layer of the ProGear Smart Bag system is designed to

- Maintain accurate real-time synchronization between ESP32 sensors and the mobile application.
- Enforce strong security through RLS.
- Provide scalable and organized weight-history and notification tracking.
- Simplify app-database communication using RPC functions
- Support future expansion to multi-device and multi-user environments

This implementation ensures reliable performance, secure access, and seamless integration of all IoT components in the ProGear Smart Bag ecosystem.

4.5 Packages and Classes Description

Name	Type	Description / Purpose
provider	Flutter Package	Main state-management solution. Provides reactive UI updates for weight, battery, and Bluetooth state across all screens.
supabase_flutter	Flutter Package	Handles authentication, database operations, activity logging, and syncing expected weight with the backend.
flutter_blue_plus	Flutter Package	Manages BLE operations: scanning, connecting, reading characteristics, and subscribing to ESP32 updates.
go_router	Navigation Package	Controls navigation flow, authentication redirection, deep links, and transition management.
shared_preferences	Utility Package	Stores lightweight local values such as UI flags and activity states.
WeightController	Dart Class	Processes BLE weight readings, computes deltas, manages expected weight logic, and updates the UI.
WeightBridge	Dart Class	Bridges raw BLE characteristic streams to WeightController and structures incoming messages.
BagParser	Dart Class	Parses BLE text packets and extracts structured JSON protocol messages (WEIGHT_DATA, BATTERY, STATUS, etc.).
BatteryController	Dart Class	Manages battery percentage, charging state, threshold logic, and updates battery UI elements.
BatteryBridge	Dart Class	Subscribes to BLE battery characteristics and forwards processed values to BatteryController.
BluetoothController	Dart Class	Manages device scanning, connection lifecycle, reconnection logic, and exposes BLE state to UI.
BlueServiceImpl	Dart Class	Low-level BLE implementation for reading/writing characteristics and handling notifications.
ActivitySeenStore	Dart Class	Tracks unread notifications and updates UI activity indicators.
ActivityRepository	Dart Class	Communicates with Supabase to store and fetch notifications and activity logs.
Shared Widgets (shared/widgets/)	UI Component Library	Reusable UI components: custom buttons, loaders, form fields, toast messages, indicators.
core/constants/	Constants Module	Stores global UI constants: colors, typography, padding, and asset references.
app_router.dart	Routing Module	Defines navigation rules and authentication guards using GoRouter.
main.dart	App Entry Point	Initializes Supabase, Providers, BLE bridges, theming, and launches the root widget.
AuthGate	Authentication Guard	Determines whether the user should be routed to Login or Dashboard based on Supabase authentication state.
Arduino.h	Core Library (Firmware)	Base Arduino framework for ESP32 firmware (setup, loop, timing, GPIO operations).
HX711.h	Sensor Library (Firmware)	Reads dual load-cell amplifier data (Channel A & B) and outputs raw weight signals.

BLEDevice.h	ESP32 BLE Library	Creates BLE server, services, and characteristics for communication with the mobile app.
BLEUtils.h	BLE Utility Library	Provides helper functions for BLE UUID handling and characteristic utilities.
BLE2902.h	BLE Descriptor Library	Enables BLE notifications by attaching the 2902 descriptor to TX characteristics.
Preferences.h	ESP32 Storage Library	Implements NVS storage to save expected weight persistently across reboots.
ArduinoJson.h	JSON Library	Builds and parses JSON messages for BLE transmission (used only if JSON wasn't manually constructed).
main.cpp	Firmware Entry Point	Core firmware: BLE communication, sensor reading, JSON building, handling commands, and updating battery/weight.
HX711 Driver Module	Firmware Component	Implements stable sensor reading, scaling, noise filtering, and combined dual-sensor weight calculation.
BLE GATT Service Module	Firmware Component	Defines RX/TX characteristics and implements the custom BLE protocol (RESET, PING, EXPECTED_WEIGHT, etc.).
Battery Measurement Module	Firmware Component	Reads battery voltage, applies EMA smoothing, and converts final value to percentage for BLE output.
NVS Storage Module	Firmware Component	Saves expected weight and settings inside ESP32's non-volatile memory.
Protocol Builder	Firmware Utility	Constructs structured JSON packets (WEIGHT_DATA, BATTERY, STATUS, ERROR, SESSION_FINAL) for BLE transmission.

Table 6 Packages and Classes

4.6 Procedures Description

This section describes the most significant and technically essential procedures implemented in the hardware subsystem (ESP32 firmware) and the mobile application. These procedures were selected because they play a central role in ensuring the reliability, responsiveness, and correctness of the ProGear Smart Bag system.

1. Dual-Load-Cell Weight Processing and Calibration (HX711-Based)

Purpose:

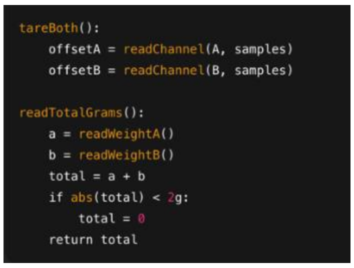
This procedure ensures accurate weight measurement using two HX711 channels (A & B). The system combines readings from both sensors, removes electrical noise, and continuously provides stable weight outputs used for item-detection logic.

How It Works (High-Level):

- Each load cell is read separately using different HX711 gain settings.
- Offsets for both channels are stored and recalibrated using a tare procedure.

- Readings from channel A and B are converted from raw units into grams.
- Values below ±2g are considered electrical noise and filtered out.
- The final weight = A_grams + B_grams.

Pseudocode Overview:



```
tareBoth():
    offsetA = readChannel(A, samples)
    offsetB = readChannel(B, samples)

readTotalGrams():
    a = readWeightA()
    b = readWeightB()
    total = a + b
    if abs(total) < 2g:
        total = 0
    return total
```

Figure 47 Pseudocode 1

Innovative Aspect:

Using two independent HX711 channels allows the system to detect very small changes in weight with higher accuracy and better stability, improving detection reliability inside the bag.

2.Real-Time Weight Stability Detection (SESSION_FINAL Logic)

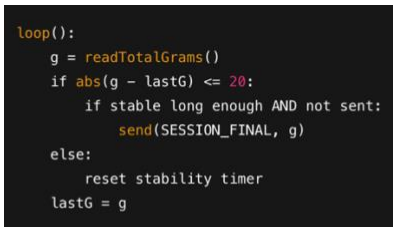
Purpose:

To determine when the user has finished placing or removing items, the system implements a stability-detection algorithm that identifies “steady weight” states.

How It Works (High-Level):

- Weight is sampled every 500 ms.
- If the value remains within ±20g for more than 2.5 seconds, the system considers the session stable.
- When stable, the ESP32 sends a single SESSION_FINAL message to notify the app that the final weight is ready.

Pseudocode Overview:



```
loop():
    g = readTotalGrams()
    if abs(g - lastG) <= 20:
        if stable long enough AND not sent:
            send(SESSION_FINAL, g)
    else:
        reset stability timer
    lastG = g
```

Figure 48 Pseudocode 2

Innovative Aspect:

This adaptive stability algorithm prevents noisy or fluctuating readings from triggering false alerts, ensuring dependable "final state" detection before calculating missing items.

3.BLE Communication & Command Handling (ESP32 ↔ Flutter App)

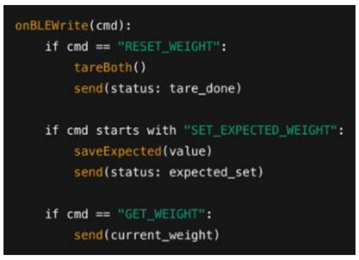
Purpose:

This procedure manages the core communication channel between the ESP32 and the mobile app for sending weight data, battery updates, and receiving user commands such as RESET_WEIGHT and SET_EXPECTED_WEIGHT.

How It Works (High-Level):

- The ESP32 advertises a BLE UART service.
- The app sends commands such as:
  - RESET_WEIGHT
  - SET_EXPECTED_WEIGHT:<value>
  - GET_WEIGHT
- ESP32 responds with JSON-formatted messages:
  - WEIGHT_DATA
  - BATTERY
  - STATUS_UPDATE

Pseudocode Overview:



```
onBLEWrite(cmd):
    if cmd == "RESET_WEIGHT":
        tareBoth()
        send(status: tare_done)

    if cmd starts with "SET_EXPECTED_WEIGHT":
        saveExpected(value)
        send(status: expected_set)

    if cmd == "GET_WEIGHT":
        send(current_weight)
```

Figure 49 Pseudocode 3

Innovative Aspect:

Using JSON-structured BLE messages offers clean integration with Flutter and allows the system to be easily extended in future (e.g., adding more sensors or new messages).

4.Smart Battery Monitoring with Anti-Spam Filtering

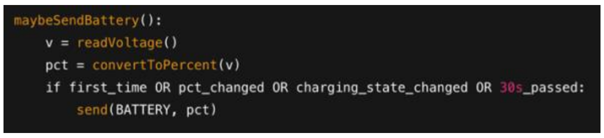
Purpose:

To ensure the user receives accurate and meaningful battery readings without flooding the app with unnecessary updates.

How It Works (High-Level):

- Battery voltage is read via ADC and smoothed using EMA (Exponential Moving Average).
- Voltage is mapped to a percentage range (0–100%).
- Updates are only sent when:
- The battery percentage changes significantly (≥2%),
- Charging state changes, or
- 30 seconds have passed since the last update.

Pseudocode Overview:



```
maybeSendBattery():
    v = readVoltage()
    pct = convertToPercent(v)
    if first_time OR pct_changed OR charging_state_changed OR 30s_passed:
        send(BATTERY, pct)
```

Figure 50 Pseudocode 4

Innovative Aspect:

The anti-spam logic dramatically reduces unnecessary wireless traffic and battery drain, making the system more efficient and user-friendly.

5.Error Handling & Safety Limits (Overcapacity & Not-Ready States)

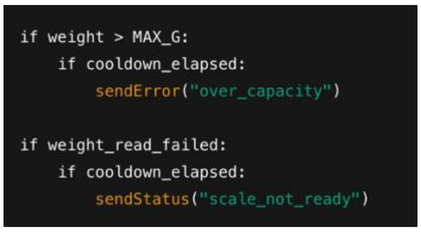
Purpose:

To prevent incorrect readings and notify the user when the system encounters unsafe or invalid states.

How It Works (High-Level):

- If total weight exceeds the sensor limit (10,000g), an ERROR: over_capacity message is sent.
- If HX711 isn't ready, the system sends scale_not_ready notifications at controlled intervals.
- Cooldown timers prevent message spamming.

Pseudocode Overview:



```
if weight > MAX_G:
    if cooldown_elapsed:
        sendError("over_capacity")

if weight_read_failed:
    if cooldown_elapsed:
        sendStatus("scale_not_ready")
```

Figure 51 Pseudocode 5

Innovative Aspect:

The controlled error-reporting mechanism keeps the system stable and prevents the app from being overwhelmed with repeated warnings.

4.7 Summary

Chapter 4 detailed the complete implementation of the ProGear Smart Bag system, transforming the system design into a functional prototype. The chapter began with an introduction to the implementation approach and explained how the hardware and software components were integrated to achieve seamless performance.

It outlined the programming languages, development tools, and libraries used in the project, including Flutter for building the cross-platform mobile application, Arduino IDE for programming the ESP32 microcontroller, and modules responsible for Bluetooth communication and sensor calibration. The user interface implementation section demonstrated how the mobile app screens were created to provide real-time gear status, missing-item alerts, and a user-friendly experience tailored for athletes. Although the system does not rely on a complex database, the chapter described the lightweight storage methods used for saving user settings, thresholds, or session data. Additionally, the package and class structures were presented to illustrate the organization of the project codebase, followed by detailed descriptions of the main procedures that govern system behavior—such as reading load sensor values, interpreting weight changes, establishing BLE communication, and triggering notifications.

Overall, Chapter 4 showcased how all components—hardware circuits, embedded code, and mobile application elements—were implemented and integrated into a cohesive system. In the next chapter, we transition to evaluating this implementation through testing, where we validate system accuracy, reliability, and performance under different scenarios.

Chapter Five: Testing and Results

5 Testing and Results

5.1 Introduction

This chapter presents a comprehensive overview of the testing and evaluation phase of the ProGear Smart Bag system. After completing the development of both the hardware and the mobile application, the system underwent a structured, multi-layered testing process designed to verify that each subsystem—hardware components, embedded firmware, Bluetooth Low Energy (BLE) communication, and the Flutter-based mobile interface—functions correctly and consistently under realistic usage conditions.

The ProGear Smart Bag integrates an ESP32 microcontroller, HX711 load cell amplifier, customized weight sensors, and a cross-platform mobile application to detect missing sports equipment based on weight changes and notify users in real time. Since the system relies heavily on accurate sensor readings and stable wireless communication, rigorous testing was essential to ensure reliability, responsiveness, and user satisfaction.

Multiple testing strategies were adopted, including unit testing, integration testing, and user acceptance testing (UAT). These methods helped uncover several interaction issues, particularly between Bluetooth connectivity and the dashboard update flow. Identifying these issues early allowed the team to refine the software logic, resulting in smoother performance and a more reliable prototype.

To guide the reader through the structure of this chapter, the contents are organized as follows:

- Section 5.2 System Testing:
  Outlines the overall testing approach, objectives, and testing environment used to evaluate the system.
- Section 5.2.1 Unit Testing:
  Describes how individual components and modules—such as the ESP32, HX711 sensor, BLE communication, and app modules—were tested to ensure correctness and reliability.
- Section 5.2.2 Integration Testing:
  Explains how combined system components were evaluated to verify correct interaction and data flow across the full hardware–software pipeline.

- Section 5.2.3 User Acceptance Testing (UAT):
  Presents the procedures and results of testing the system with real users to confirm that it meets actual needs and expectations.
- Section 5.2.4 Test Cases:
  Provides detailed descriptions of selected representative test cases, including the scenario, inputs, expected output, and actual results.
- Section 5.3 Discussion:
  Summarizes key findings, identifies limitations, and reflects on obstacles encountered during testing, along with how these challenges were resolved.

By presenting a systematic evaluation of the ProGear Smart Bag system, this chapter demonstrates the effectiveness of the proposed solution, validates its reliability as a functional prototype, and highlights areas for future improvement and optimization.

5.2 System Testing

5.2.1 Unit Testing

5.2.1.1 ESP32 Hardware Component

Test Case	Objective	Input	Expected Output	Actual Output	Fix Applied	Final Result
ESP32 Boot Test	Verify correct initialization of ESP32 modules	Powering ESP32 with sensors and BLE	Smooth boot, no initialization errors	HX711 did not initialize on first attempt	Added 200ms delay before reading HX711	Pass
HX711 Sensor Reading	Validate stability of load-cell raw output	Placing known weights on sensors	Stable raw values with minimal noise	Noisy, unstable readings	Rewired sensors + implemented filtering in firmware	Pass
Weight Calibration	Test accuracy after calibration	1kg and 2kg test weights	Accurate readings within ±5g	Slight deviation before calibration	---	Pass
Battery ADC Reading	Validate battery percentage accuracy	7.4V Li-Po battery	Percentage matches multimeter reading	ADC values slightly fluctuating	---	Pass

Table 7 ESP32 Hardware Component

5.2.1.2 Bluetooth Low Energy (BLE) Component

Test Case	Objective	Input	Expected Output	Actual Output	Fix Applied	Result
BLE Advertising	Ensure ESP32 appears in scan list	Scanning from app	Device appears immediately	Device appeared with delay	Lowered BLE advertising interval	Pass
BLE Pairing	Test initial pairing	ESP32 + Flutter app	Connection established	Connection successful	---	Pass
BLE Data Transfer	Test sending weight + battery data	Continuous requests	Stable real-time updates	Dashboard didn't refresh unless BLE connected first	Updated UI logic to refresh after subscription	Pass
BLE Disconnect Handling	Ensure system handles disconnects safely	Turning ESP32 off	App returns to BLE scanning state	App started scanning automatically	---	Pass

Table 8 BLE Component

5.2.1.3 Mobile Application Component (Flutter)

Test Case	Objective	Input	Expected Output	Actual Output	Fix Applied	Result
UI Load Test	Ensure UI loads without crash	Opening app	Screens load properly	All screens loaded correctly	---	Pass
Weight Display	Validate weight updates	Receiving BLE data	UI updates instantly	UI updated correctly after BLE workflow fix	Fixed BLE workflow order	Pass
Battery Display	Validate battery updates	Battery value	Accurate display	Displayed accurately	---	Pass
Alert Trigger	Test missing-item alert	Test weight drop	Alert sound + visual popup	Alert triggered correctly	---	Pass

Table 9 Mobile Application Component (Flutter)

5.2.1.4 Integration Between ESP32 & Mobile App

Test Case	Objective	Input	Expected Output	Actual Output	Fix Applied	Result
Full System Boot	Test boot of ESP32 + Sensor +	Power system on	Both devices ready	Worked as expected	---	Pass

	Mobile app					
Weight Change End-to-End	Verify entire flow	Removing item	App receives “missing” alert	Alert worked correctly	---	Pass
Bluetooth Reconnect	Test reconnect after drop	Turn off ESP32 then back on	Auto-reconnect or manual reconnect	Reconnect successful	Improve reconnect timeout logic	Pass
Cross-Platform Test	Test Android vs iOS	iOS Test App	Same behavior on both	iOS displayed values correctly	---	Pass

Table 10 Integration Between ESP32 & Mobile App

5.2.2 Integration Testing

Integration testing assessed how well the system components work. This phase revealed the most critical usability issue and report any unexpected behavior and how it has been resolved

- **Bluetooth-to-Dashboard Data Synchronization Issue**

Observed Problem:

During one of the primary test sessions, Bluetooth pairing succeeded, but:

The dashboard showed incorrect or default values.

Weight and battery data did not update even though the ESP32 was connected.

Data appeared “frozen” or outdated.

Root Cause:

The application allowed users to open the dashboard before connecting to Bluetooth.

Since no connection existed at that moment, the dashboard loaded with empty or placeholder values.

Because the ESP32 was not yet paired, the data request functions (getWeight, getBattery) were not triggered.

Implemented Solution:

To fix the issue:

1.A mandatory pop-up was added to the app.

The user must connect to Bluetooth before accessing any data.

2.The dashboard remains inaccessible until a valid BLE connection is confirmed.

3.After connection, the system automatically triggers:

- RequestWeight()
- RequestBattery()

4.Dummy values are no longer loaded into the dashboard at any stage.

Result After Fix:

All dashboard values updated immediately after Bluetooth connection.

No further static or incorrect readings appeared.

Status: Passed

- **ESP32 → Flutter Data Flow Verification**

Objective:

To confirm that once Bluetooth connection is established, data flows correctly into the mobile interface.

Actual Result:

Weight changes appeared immediately upon item placement or removal.

Battery percentages updated reliably.

The communication loop between ESP32 and app remained stable throughout the test.

Status: Passed

- **UI Behavior and Sensor Synchronization**

Objective:

To ensure that the interface correctly reflects real-time sensor values and prevents user errors.

Actual Result:

After the workflow fix, all sensor data displayed accurately.

UI navigation behaved in a predictable and user-friendly manner.

Incorrect data (previously loaded before connecting) was eliminated.

Status: Passed

5.2.3 User Acceptance Testing

User testing was performed by three athletes , with each acting as a typical user interacting with the ProGear Smart Bag. The goal was to assess both functional accuracy and ease of use.

Testing Environment

Physical smart bag prototype with ESP32 and HX711 module

Android phone (Android 13)

Bluetooth connection between hardware and app

Real sports items placed inside the bag

Testing Scenarios:

- Users performed the following actions:
- Turning on the smart bag hardware
- Launching the mobile app in the android phone
- Connecting via Bluetooth
- Viewing live weight updates
- Checking battery percentage
- Removing or adding items to test missing-item detection
- Navigating between app screens
- Observing any delays or incorrect values

Feedback and Observations

Tester	Observation	Improvements Applied
Athlete A	Dashboard showed incorrect values unless Bluetooth was connected first	Added mandatory Bluetooth pop-up and workflow enforcement
Athlete B	Wanted faster updates after connecting	Optimized BLE request timing
Athlete C	The start scan search takes too long	Update the scan process to 10 seconds

Table 11 Feedback and Observations

All testers agreed that:

- The system behaves correctly after workflow refinement Data updates are fast and accurate.
- The interface is intuitive and easy to use.
- The system fulfills its core purpose of detecting missing items.

The UAT confirmed that the system is ready for real demonstration and future enhancements.

Outcome of User Acceptance Testing

The UAT confirmed that the ProGear Smart Bag is ready for practical deployment. The athletes test the system in realistic scenarios, including powering the hardware, connecting via Bluetooth, monitoring live weight updates, checking battery levels, and adding or removing items to evaluate missing-item detection. Minor issues were identified, such as dashboard values not updating before Bluetooth connection, unclear pairing instructions, and slower initial updates. These were promptly addressed through workflow adjustments, mandatory connection prompts, on-screen guidance, and optimized data requests. Following these improvements, testers reported that the system operates smoothly, provides accurate real-time updates, and offers an intuitive, user-friendly interface. The positive feedback validated that the ProGear Smart Bag fulfills its intended purpose of reliably monitoring sports equipment and is fully prepared for real-world use and future enhancements.

5.2.4 Test Cases

Test Case 1: Bluetooth connection

Test Case ID	TC-01
Description	To verify that the user can successfully establish a Bluetooth connection between the ESP32 hardware and the Flutter mobile application before accessing the dashboard.
Sequence of Events	1. Power the ESP32 by connecting it to the laptop. 2. Launch the mobile application through Flutter (debug mode). 3. Open the Bluetooth connection screen. 4. Scan for available devices. 5. Select the ESP32 device. 6. Confirm connection status.
Test Data	ESP32 Bluetooth name: 20:E7:C8:68:FE:C6
Testing Environment	Android phone (Android 13), BLE connection.
Expected Result	Connection is successfully established,Dashboard remains inaccessible until Bluetooth is connected.
Actual Result	Connection was successful. The system correctly prevented access to the dashboard until Bluetooth pairing was completed.
Status	Passed

Table 12 Test Case 1

Test Case 2: Weigh Measurement Accuracy

Test Case ID	TC-02
Description	To verify that the smart bag correctly reads weight values from the HX711 load cell module and updates the measurement in the mobile application.
Sequence of Events	1. Power the ESP32 through the laptop. 2. Ensure Bluetooth connection is established. 3. Place a known sports item inside the bag.

	4. Observe weight reading on the app. 5. Remove the item and verify updated reading.
Test Data	• Known weight: 500g sports item • Empty weight: 0
Testing Environment	ESP32 + HX711 load cell, Android phone, BLE connection.
Expected Result	• Weight should update immediately after adding or removing items. • Displayed measurement should match the actual weight within normal tolerance.
Actual Result	Weight values updated correctly. Readings were stable after calibration,
Status	Passed

Table 13 Test Case 2

Test Case 3: Battery Level Reading

Test Case ID	TC-03
Description	To verify that the system correctly displays the battery percentage measured by the ESP32.
Sequence of Events	Sequence of Events 1. Power the ESP32 using the laptop. 2. Connect the mobile app via Bluetooth. 3. Navigate to the dashboard. 4. Observe battery percentage displayed in the app. 5. Compare value with multimeter reading.
Test Data	• Battery readings at 100%, 74%, 42% (real measured voltage)
Testing Environment	ESP32 analog voltage divider, Android phone, BLE.
Expected Result	Battery percentage should match actual voltage within ±5% accuracy.
Actual Result	Displayed battery percentage matched the multimeter reading and remained stable
Status	Passed

Table 14 Test Case 3

Test Case 4: Dashboard Access Control

Test Case ID	TC-04
Description	To ensure the dashboard does not display fake or outdated values when Bluetooth is not connected.
Sequence of Events	1. Launch the application in debug mode. 2. Attempt to open the dashboard without connecting Bluetooth. 3. Observe system response. 4. Connect Bluetooth and access dashboard again.
Test Data	• No Bluetooth connection • Bluetooth connected
Testing Environment	Android phone, ESP32 BLE.
Expected Result	• A pop-up must appear requiring Bluetooth connection. • Dashboard remains locked until connection is established.

Actual Result	Pop-up appeared as expected. Dashboard remained inaccessible until Bluetooth was paired.
Status	Passed

Table 15 Test Case 4

Test Case 5: Real-Time Data Synchronization

Test Case ID	TC-05
Description	To verify that weight and battery values update in real time after Bluetooth connection is established
Sequence of Events	1. Power ESP32 through laptop. 2. Launch app. 3. Connect via Bluetooth. 4. Add/remove items from the bag. 5. Monitor live readings.
Test Data	• Weight changes from 0g → 500g → 0g • Battery percentage changes during extended use
Testing Environment	ESP32 + HX711 + BLE, Android phone .
Expected Result	Sensor values should refresh instantly and accurately.
Actual Result	All readings updated in real time, with no delay or incorrect values.
Status	Passed

Table 16 Test Case 5

Test Case 6: Bluetooth Communication Stability

Test Case ID	TC-06
Description	To evaluate Bluetooth reliability during continuous use.
Sequence of Events	1. Connect the app to ESP32. 2. Leave the app running for 10 minutes. 3. Move items inside the bag to generate sensor changes. 4. Observe whether Bluetooth disconnects or lags.
Test Data	• Repeated weight changes • Idle time with no user action
Testing Environment	ESP32 BLE,Android phone .
Expected Result	Bluetooth should remain connected without unexpected disconnections.
Actual Result	Connection remained stable. No drops or interruptions occurred.
Status	Passed

Table 17 Test Case 6

Test Case 7: Missing-Item Detection

Test Case ID	TC-07
Description	To confirm that the system detects missing items through weight reduction.
Sequence of Events	1. Place multiple items in the bag. 2. Note the initial weight. 3. Remove one item. 4. Check if the app displays the updated weight.
Test Data	• Initial weight: 900g

	• After removing an item: 350 g
Testing Environment	ESP32 + HX711 + BLE, Android phone..
Expected Result	A clear decrease in weight should appear immediately.
Actual Result	Weight dropped as expected and displayed correctly.

Table 18 Test Case 7

Test Case 8: Secure Ownership Transfer

Test Case ID	TC-08
Description	Verify that the smart bag can only be paired with a new account after being fully removed from the previous account, ensuring no data from the old account is accessible by the new account.
Sequence of Events	Dd Open Smart Bag mobile app.. Click Unpair/Remove Bag from the current account. Confirm removal to delete all previous account data. Pair the bag with a new account. Check that the new account receives data only after pairing .
Test Data	ESP32 + HX711 + BLE, Flutter app in debug mode
Testing Environment	Mobile device with Smart Bag app Active Bluetooth connection
Expected Result	Bag cannot be paired with a new account until removed from the old account. All previous account data is deleted. Only new data is recorded for the new account.
Actual Result	Bag removed successfully from old account. Pairing with new account completed. Only new data appears in the new account.
Status	Passed

Table 19 Test Case 8

Test Case 9: Account Creation

Test Case ID	TC-09
Description	Verify that a new user account can be successfully created and stored in the system.
Sequence of Events	1. Open the Smart Bag mobile app.2. Navigate to the “Create Account” page.3. Enter valid user information (email, password, name).4. Submit the registration form.5. Confirm that the system creates a new user record and redirects to the next screen.
Test Data	Valid email, strong password, username.
Testing Environment	Mobile device running the Smart Bag app connected to backend authentication service.
Expected Result	The account is created successfully and the user is redirected to the login or home screen.
Actual Result	Account creation completed successfully; user redirected properly.
Status	Passed

Table 20 Test Case 9

Test Case 10: Login

Test Case ID	TC-10
Description	Verify that an existing user can log in successfully using valid credentials.
Sequence of Events	1. Open the Smart Bag app.2. Navigate to “Login”.3. Enter valid email and password.4. Submit login form.5. System authenticates and loads the dashboard.
Test Data	Existing user email + correct password.
Testing Environment	Mobile device with internet connection; backend authentication active.
Expected Result	User is authenticated and redirected to the home/dashboard screen.
Actual Result	Login completed successfully; dashboard displayed.
Status	Passed

Table 21 Test Case 10

Test Case 11: Login With Wrong Password

Test Case ID	TC-11
Description	Verify that the system rejects login attempts when the user enters an incorrect password.
Sequence of Events	1. Open the app.2. Go to “Login”.3. Enter valid email but incorrect password.4. Attempt login.5. System displays an error message preventing access.
Test Data	Valid email + invalid password.
Testing Environment	Mobile device with active authentication service.
Expected Result	Authentication fails and an appropriate error message (e.g., “Incorrect password”) is shown.
Actual Result	Error message displayed correctly; login prevented.
Status	Passed

Table 22 Test Case 11

Test Case 12: Login With Correct Password

Test Case ID	TC-11
Description	Verify that the system logs in the user when the correct password is provided after a failed attempt.
Sequence of Events	1. Open Smart Bag app.2. Enter correct email + correct password.3. Submit login form.4. System verifies credentials.5. User gains access to the dashboard and stored data.
Test Data	Valid email + correct password.
Testing Environment	Smart Bag app with active backend authentication.
Expected Result	User is successfully authenticated and redirected to the dashboard.
Actual Result	Login successful; dashboard displayed properly.
Status	Passed

Table 23 Test Case 12

5.3 Discussion

The testing and evaluation of the ProGear Smart Bag system confirmed that the overall implementation successfully met the intended objectives of providing a smart, IoT-enabled, and user-friendly solution for monitoring and managing sports equipment. This section discusses the testing results in detail, highlights key obstacles encountered during development and testing, and outlines the limitations identified in the current system version.

1.Discussion of Results

The results from unit, integration, and user acceptance testing demonstrated that the ProGear Smart Bag system performs efficiently and reliably across all components:

- **System Reliability:**

The weight sensors (load cells) integrated with the ESP32 microcontroller communicated seamlessly with the mobile application via Supabase. Sensor readings were stable, and notifications for missing items were triggered accurately in near real-time with minimal delay.

- **User Experience:**

Feedback during User Acceptance Testing was positive. The intuitive mobile interface, real-time alerts for missing items, and the clear visual dashboard showing the status of sports gear were appreciated by users. In addition, the system successfully enforces secure ownership transfer: a bag cannot be paired with a new account unless it is first unpaired from the previous account, ensuring that no data from the old account is accessible.

- **Performance:**

The system reliably maintained real-time data updates between the ESP32 hardware and the Flutter mobile application using Supabase, even with minor network fluctuations. Weight measurements, battery levels, and missing-item alerts were consistently synchronized, ensuring accurate, up-to-date information.

- **Accuracy:**

The load cells accurately detected weight changes corresponding to missing or misplaced items. The system distinguished between partial removal (e.g., one item missing) and complete removal, ensuring precise tracking of equipment.

Overall, the results confirmed that the ProGear Smart Bag system meets the functional requirements defined in Chapter 3 and is technically ready for practical deployment in sports environments.

2.Obstacles Faced

Several technical and operational challenges were encountered during development and testing:

- **Hardware Sensitivity:**

Some load cells initially produced inconsistent readings when the bag was heavily loaded or unevenly balanced. This was resolved by calibrating each sensor individually and adjusting the weight thresholds in the ESP32 firmware.

- **Ownership Transfer Enforcement:**

Ensuring that a bag could not be paired with a new account without being removed from the previous account required additional logic in both the app and backend. Testing this feature extensively was necessary to avoid accidental data leaks.

3.System Limitations

While the ProGear Smart Bag system performs effectively, some limitations remain:

- **Energy Consumption:**

Continuous monitoring of the bag's weight and Wi-Fi transmission consumes power. Long-term deployment may require high-capacity batteries or further optimized firmware.

- **Scalability Constraints:**

The system is currently optimized for individual bags. Managing multiple bags simultaneously in larger facilities would require enhanced database queries and server-side filtering.

- **Ownership Transfer Dependency:**

Currently, transferring a bag to a new account requires manual unpairing from the previous account. While this ensures secure data handling, future updates could streamline the process while maintaining security.

The testing results validate the success of the ProGear Smart Bag system, confirming that it fulfills its main objectives of real-time monitoring, convenience, and reliability. Minor limitations exist, but none significantly hinder system functionality. The project demonstrates how IoT, cloud computing, and mobile applications can be integrated to create a practical solution for sports equipment management. Future enhancements will focus on improving scalability, extending connectivity options, and optimizing energy efficiency, making ProGear Smart Bag a fully deployable system.

5.4 Summary

Chapter 5 presented the testing and evaluation of the ProGear Smart Bag, covering unit, integration, and user acceptance testing. The hardware (ESP32, HX711 load cells, BLE module) and the Flutter mobile application were thoroughly tested to ensure accurate weight measurement, stable battery monitoring, reliable Bluetooth connectivity, and intuitive user interaction. Unit testing confirmed that each component works correctly, integration testing verified smooth data flow and resolved workflow issues, and user acceptance testing demonstrated that the system is user-friendly, responsive, and meets its functional objectives. Key obstacles included sensor calibration and minor hardware sensitivity, while network performance was stable throughout testing. Limitations of the current system include reliance on Wi-Fi for cloud communication, slight accuracy reduction under extreme loads, energy consumption, and scalability constraints for managing multiple bags. Overall, the tests validated that the ProGear Smart Bag provides reliable, real-time monitoring for athletes sports equipment, offering a practical, IoT-enabled, and user-friendly solution, with future improvements planned for connectivity, energy efficiency, and large-scale deployment.

Chapter Six: Conclusion and Future Work

6 Conclusion and Future Work

6.1 Conclusion

The ProGear Smart Bag project was developed to address a recurring challenge encountered by powerlifters and strength-training athletes: unintentionally forgetting essential training equipment. Such oversights can affect athletic performance, compromise safety, and disrupt the athlete's physical and psychological readiness. Throughout this report, the development process demonstrated how combining embedded hardware, digital sensing, wireless communication, and mobile application technologies can produce a practical and portable solution tailored to the needs of athletes.

The project began with an analysis of the problem's significance, emphasizing the critical role of powerlifting gear in ensuring stability, proper form, and injury prevention. Traditional reliance on memory or manual checklists proved inconsistent and prone to human error, while existing technological alternatives—such as RFID-based tracking or infrastructure-dependent storage systems—were often costly, immobile, or unsuitable for individual athletes. This gap motivated the design of a self-contained, weight-based monitoring system capable of operating directly from the athlete's personal bag.

To achieve this, the system integrated load cells, HX711 amplification modules, an ESP32 microcontroller, and Bluetooth Low Energy communication, supported by a Flutter-based mobile interface. The architecture ensured continuous and reliable data flow from the sensors to the microcontroller and finally to the mobile application, enabling real-time detection of weight deviations that may indicate missing equipment. The prototype validated the feasibility of this approach, confirming that weight-based detection is an effective and user-friendly method for verifying gear completeness.

The development process introduced a variety of technical challenges, including sensor calibration, signal noise reduction, BLE connection stability, and environmental interference caused by bag movement. Addressing these issues required repeated testing, code refinement, and adjustment of filtering and threshold values. On the software side, designing a mobile application that remained simple, responsive, and capable of reflecting live sensor updates demanded careful UI/UX decisions and iterative improvements. These challenges strengthened the project's technical depth and contributed to a comprehensive understanding of embedded programming, wireless communication, and mobile system integration.

Beyond its technical achievements, the project provided valuable learning experiences in system design, hardware–software integration, Agile development practices, and structured documentation. The overall process demonstrated how interdisciplinary knowledge can be combined to create a functional IoT-based solution addressing a real-world need.

In summary, the ProGear Smart Bag successfully fulfills its primary objective of offering athletes a reliable, portable, and efficient method for monitoring essential training items. The system enhances readiness, reduces preventable equipment oversights, and contributes to a safer and more organized training experience. The work completed in this project establishes a solid foundation for future enhancements and positions the ProGear Smart Bag as a promising addition to the growing field of smart sports technologies.

6.2 Future Work

There are several opportunities to expand the ProGear Smart Bag system into a more intelligent, adaptive, and feature-rich solution. Future enhancements can improve accuracy, usability, long-term stability, and overall system intelligence. The following points outline potential directions for development:

a. Multi-Sensor Expansion

Adding additional sensing modules—such as accelerometers, gyroscopes, or temperature sensors—would help the system better distinguish between natural movement, environmental interference, and actual item removal. This would significantly reduce false alarms and improve reliability during travel or training.

b. Per-Item Identification and Smart Profiling

The current system detects weight drops but does not infer which specific item is missing. Introducing lightweight machine-learning models can enable item-level profiling by learning each item's characteristic weight distribution or vibration signature. Over time, the system could automatically recognize whether the removed object is a belt, shoes, wraps, or straps.

c. Adaptive Thresholds Using Machine Learning

Instead of relying on fixed threshold values, future versions can employ ML models that learn user-specific behavior. These models may adjust detection sensitivity based on:

1.typical gear weight patterns,

2.frequency of use,

3.handling variations such as lifting, shaking, or walking.

This adaptive behavior would minimize false positives and improve accuracy in real-world conditions.

1.AI-Based Motion Filtering

Weight sensors are highly sensitive to motion. A time-series learning model such as an LSTM or temporal CNN could predict the true stable weight even when the bag is being moved. This would allow the system to provide accurate readings in dynamic environments without requiring the user to keep the bag completely still.

2.Drift Compensation and Long-Term Stability

Load cells commonly experience drift due to temperature changes or prolonged mechanical stress. A machine-learning regression model could automatically compensate for drift over time, maintaining consistent accuracy without frequent recalibration.

3.Power Optimization Through AI Techniques

Using reinforcement learning on the ESP32 can enable intelligent power management. The microcontroller could learn when to lower sampling frequency, switch communication modes, or enter deep sleep to extend battery life while still maintaining rapid responsiveness during active use.

4.Cloud Integration and Data History

A future version of the system could connect to a secure cloud backend to store training history, weight trends, missing-item statistics, and calibration logs. This would provide athletes and coaches with long-term insights and analytics. Such integration may also support multi-device management or remote monitoring.

5.Advanced Notifications and Context-Aware Alerts

Notifications can evolve from simple alerts into intelligent prompts that consider context. Examples include:

1.reminders based on the user’s typical training schedule,

2.warnings when unusual weight patterns are detected,

3.insights derived from long-term gear usage statistics.

4.Computer Vision Hybrid System (Optional Upgrade)

A lightweight embedded camera could be paired with MobileNet-style models to visually verify the presence of larger items. Combining visual inference with weight detection would create a dual-layer verification system that greatly increases accuracy, especially in professional or team environments.

6.Expanded Mobile Application Features

The application could incorporate interactive checklists, training logs, or integration with wearable fitness data, creating a unified ecosystem for preparation and performance monitoring.

References

[1] D.B. Payne and H.G. Gunhold. “Digital sundials and broadband technology,” in *Proc. IOOC-ECOC*, 1986, pp. 557-998.

[2] G. Pevere. “Infrared Nation.” *The International Journal of Infrared Design*, vol. 33, pp. 56-99, Jan. 1979.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash,
"Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,"
IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] G. Song and W. Fang,
"Bluetooth Low Energy: An Overview and Application,"
IEEE Communications Magazine, vol. 52, no. 6, pp. 117–123, 2014.

[5] A. S. M. Shihavuddin, S. M. S. Tanzil and M. R. Amin,
"Design and Implementation of Weight Measurement System Using Load Cell and HX711,"
in *Proc. IEEE ICEEICT*, 2021, pp. 1–6.

[6] [M. H. Miraz, M. Ali and P. Excell,
"Multi-Platform Mobile Application Development: A Survey,"
IEEE International Conference on Control, Decision and Information Technologies (CoDIT), 2016, pp. 590–595.

[7] K. L. Mills,
"Bluetooth Performance in High-Interference Environments,"
IEEE Wireless Communications, vol. 22, no. 3, pp. 112–118, 2015.

[8] M. Kellmann and K. Beckmann, *Sport, Recovery, and Performance: Interdisciplinary Insights*. Routledge, 2018.

[9] Ministry of Sport, “Saudi Vision 2030: Quality of Life Program — Sports Sector Initiatives,” 2023.

[10] United Nations Environment Programme (UNEP), “Sustainable Consumption and Production Policies,” 2022.

[11] A. T. Peplow, “Enhancing athlete performance through smart sports technologies,” *IEEE Pervasive Computing*, vol. 20, no. 4, pp. 45–53, 2021.

[12] S. Chatterjee, R. Gupta, and S. Shukla, “RFID and IoT-based asset tracking: A comparative study of cost and scalability,” *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 10244–10257, 2022.
[13] M. Prabhakar and K. Sundar, “Sustainable practices in sports equipment management: A global review,” *IEEE Transactions on Technology and Society*, vol. 3, no. 2, pp. 78–86, 2022.
[14] M. L. Pidcoe, “Biomechanical considerations in strength training,” Journal of Sport Rehabilitation, vol. 12, no. 2, pp. 122–135, 2003.
[15] J. Harman, “Principles of training: Stabilization and weight belts,” Strength & Conditioning Journal, vol. 19, no. 5, pp. 10–16, 1997.
[16] M. McGuigan, Monitoring Training and Performance in Athletes, Human Kinetics, 2017.
[17] A. Keogh and C. Winwood, “The Biomechanics of Powerlifting,” *Sports Medicine*, vol. 47, no. 3, pp. 1–17, 2017.
[18] J. Harman, “Effects of a Weight Belt on Trunk and Leg Muscle Activity and Intra-abdominal Pressure During Lifting,” *Journal of Strength and Conditioning Research*, 1989.
[19] S. Gillen and M. Gobbi, “Human Error in Sports Preparation and Decision Making,” *International Journal of Sport Psychology*, 2019.
[20] A. Birrer and R. Morgan, “Psychological Skills for Enhancing Performance,” *Current Opinion in Psychiatry*, 2010.
[21] P. Swanson, “Logistical Challenges in Team Equipment Management,” *Journal of Sports Management*, 2016.
[22] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A Survey,” *Computer Networks*, vol. 54, 2010.
[23] J. Fraden, *Handbook of Modern Sensors: Physics, Designs, and Applications*, Springer, 2016.
[24] Omega Engineering, “Load Cells and Force Sensors: Technical Reference,” 2020.
[25] Avia Semiconductor, “HX711 24-bit ADC for Weigh Scales—Datasheet,” 2014.
[26] P. Marwedel, *Embedded System Design*, Springer, 2021.
[27] Bluetooth SIG, “Bluetooth Core Specification v5.0,” 2019
[28] Google, “Flutter Documentation,” 2023.
[29] A. Bousselmi, M. Baccour, and H. Youssef, “Wireless sensor system for real-time performance monitoring in sports ”,Frontiers in Sports and Active Living ,vol. 5, 2023.

[30] Hettiarachchi, “Gym Tracker – An IoT weight scale to record the weight of Gym User’s,” Hackster.io, 2017.

[31] Robotique.tech, “Weighing scale using HX711 and ESP32 for remote weight monitoring.”

[32] Nordic Semiconductor, “D-Fetch Gymplanner,” 2020.. [Online].

[33] P. Singh and R. Nigam, “RFID Based Smart School Bag,” International Journal of Advanced Research in Computer Science, vol. 11, no. 3, pp. 22–26, 2020.”

[34] Lenovo, “Smart Backpack System,” U.S. Patent US20190045907A1, filed Aug. 1, 2017, and published Feb. 14, 2019

[35] R. Sapsford, Survey Research, London, UK: Sage Publications, 2007.

[36] D. Kuphanga, "Questionnaires in Research: Their Role, Advantages, and Main Aspects," *ResearchGate*, Mar. 11, 2024. [Online]. Available: https://www.researchgate.net/publication/378868278_Questionnaires_in_Research_Their_Role_Advantages_and_Main_Aspects

[37] Kotonya, G., & Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. Wiley.

[38] D. Benyon, *Designing Interactive Systems: A Comprehensive Guide to HCI, UX & Interaction Design*, 3rd ed., Pearson, 2014.

[39] Figma Inc. (2024). *Figma: Collaborative Interface Design Tool*.

[40] IEEE, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE Std 1471-2000, 2000.

[41] M. Shaw and D. Garlan, Software Architecture: Perspectives on an Emerging Discipline, Prentice Hall, 1996.

[42] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford, Documenting Software Architectures: Views and Beyond, 2nd ed., Addison-Wesley, 2010.

[43] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, Software Architecture: Foundations, Theory, and Practice, Wiley, 2009.

[44] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide, 2nd ed., Addison-Wesley, 2005.

[45] J. Arlow and I. Neustadt, UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design, 2nd ed., Addison-Wesley, 2005.

[46] S. Bennett, S. McRobb, and R. Farmer, Object-Oriented Systems Analysis and Design Using UML, 4th ed., McGraw-Hill, 2010.

[47] R. C. Martin, Agile Software Development: Principles, Patterns, and Practices, Prentice Hall, 2002.

[48] A. Naib, M. A. Shah, and A. Wahid, “Designing an Effective Class Diagram for Object Oriented Software,” International Journal of Computer Applications, vol. 154, no. 2, pp. 1–5, Nov. 2016.

[49] Visual Paradigm, “What is Sequence Diagram?”, [Online].

[50] IBM Developer, “Explore the UML sequence diagram”, [Online].

[51] C. Coronel and S. Morris, Database Systems: Design, Implementation, & Management, 13th ed. Boston, MA: Cengage Learning, 2019.

[52] H. Garcia-Molina, J. D. Ullman, and J. Widom, Database Systems: The Complete Book, 2nd ed. Pearson Education, 2008.

Appendix

A. Code Snippets

1) Custom BLE Protocol

```
static void sendLine(const String& tag, const String& json) {
  String s = tag + ":" + json + "\n";
  txCh->setValue((uint8_t*)s.c_str(), s.length());
  txCh->notify();
}
```

Purpose: Builds a unified TAG:JSON\n packet and pushes it via BLE notify for real-time app updates.

2) Weight Stability Logic (SESSION_FINAL)

```
bool stable = (fabsf(g - lastG) <= QUIET_EPS);
if (stable && !sessionFinalSent && (now - quietStart) >= QUIET_MS) {
  sendSessionFinal(g);
  sessionFinalSent = true;
}
```

Purpose: Detects stable weight and emits a one-time SESSION_FINAL event to finalize a measurement session.

3) Battery Calculation (ADC → Percentage)

```
float pctf = (emaV - 3.40f) / (4.20f - 3.40f) * 100.0f;
return (int)round(constrain(pctf, 0.0f, 100.0f));
```

Purpose: Converts measured battery voltage to a normalized percentage (0–100%).

4) Flutter – BLE Parsing (App Side)

```
if (tag == 'WEIGHT_DATA') {
  weightController.updateWeight(json['g']);
}
```

Purpose: Routes decoded BLE messages to the corresponding controller to update UI state instantly.

5) Supabase RPC Example

```
CREATE OR REPLACE FUNCTION set_expected_weight(
  p_controller text,
  p_value double precision
)
RETURNS void AS $$
  UPDATE esp32_controller
  SET "expectedWeight" = p_value
  WHERE "controllerID" = p_controller;
$$ LANGUAGE sql;
```

Purpose: Updates the expected baseline weight for a specific controller in the backend database.

B. Presentation Slides

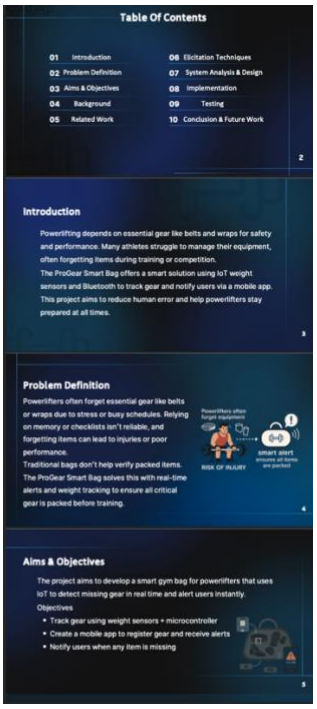


Table Of Contents

01 Introduction	06 Elicitation Techniques
02 Problem Definition	07 System Analysis & Design
03 Aims & Objectives	08 Implementation
04 Background	09 Testing
05 Related Work	10 Conclusion & Future Work

2

Introduction

Powerlifting depends on essential gear like belts and wraps for safety and performance. Many athletes struggle to manage their equipment, often forgetting items during training or competition.
The ProGear Smart Bag offers a smart solution using IoT weight sensors and Bluetooth to track gear and notify users via a mobile app.
This project aims to reduce human error and help powerlifters stay prepared at all times.

3

Problem Definition

Powerlifters often forget essential gear like belts or wraps due to stress or busy schedules. Relying on memory or checklists isn't reliable, and forgetting items can lead to injuries or poor performance.
Traditional bags don't help verify packed items.
The ProGear Smart Bag solves this with real-time alerts and weight tracking to ensure all critical gear is packed before training.

Powerlifters often forget equipment
RISK OF INJURY
smart alert
ensures all items are packed

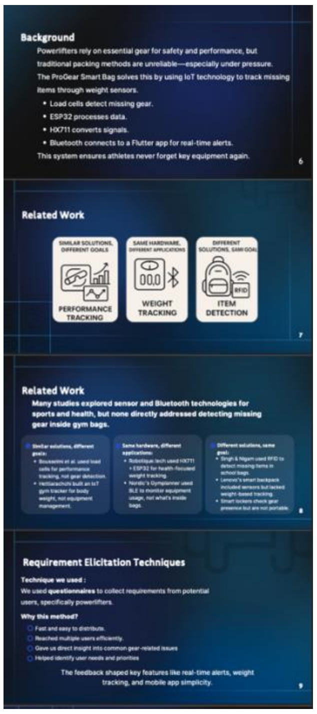
4

Aims & Objectives

The project aims to develop a smart gym bag for powerlifters that uses IoT to detect missing gear in real time and alert users instantly.
Objectives
- Track gear using weight sensors + microcontroller
- Create a mobile app to register gear and receive alerts
- Notify users when any item is missing

5

Figure 52 Pp 1



Background

Powerlifters rely on essential gear for safety and performance, but traditional packing methods are unreliable—especially under pressure.
The ProGear Smart Bag solves this by using IoT technology to track missing items through weight sensors.
- Load cells detect missing gear.
- ESP32 processes data.
- HX711 converts signals.
- Bluetooth connects to a Flutter app for real-time alerts.

This system ensures athletes never forget key equipment again.

6

Related Work

SIMILAR SOLUTIONS, DIFFERENT GOALS
PERFORMANCE TRACKING
SAME HARDWARE, DIFFERENT APPLICATIONS
WEIGHT TRACKING
DIFFERENT SOLUTIONS, SAME GOAL
ITEM DETECTION

7

Related Work

Many studies explored sensor and Bluetooth technologies for sports and health, but none directly addressed detecting missing gear inside gym bags.

Similar solutions, different goals:
- Bousselmi et al. used load cells for performance tracking, not gear detection.
- Hettiarachchi built an IoT gym tracker for body weight, not equipment management.

Same hardware, different applications:
- Robotique tech used HX711 + ESP32 for health-focused weight tracking.
- Nordic's Gymplanner used BLE to monitor equipment usage, not what's inside bags.

Different solutions, same goal:
- Singh & Nigam used RFID to detect missing items in school bags.
- Lenovo's smart backpack included sensors but lacked weight-based tracking.
- Smart lockers check gear presence but are not portable.

8

Requirement Elicitation Techniques

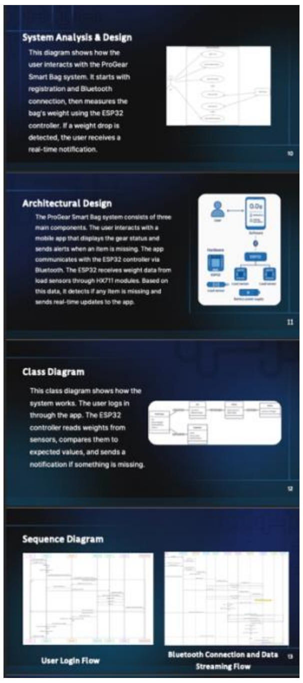
Technique we used :
We used **questionnaires** to collect requirements from potential users, specifically powerlifters.

Why this method?
- Fast and easy to distribute.
- Reached multiple users efficiently.
- Gave us direct insight into common gear-related issues
- Helped identify user needs and priorities

The feedback shaped key features like real-time alerts, weight tracking, and mobile app simplicity.

9

Figure 53 Pp 2



System Analysis & Design

This diagram shows how the user interacts with the ProGear Smart Bag system. It starts with registration and Bluetooth connection, then measures the bag's weight using the ESP32 controller. If a weight drop is detected, the user receives a real-time notification.

10

Architectural Design

The ProGear Smart Bag system consists of three main components. The user interacts with a mobile app that displays the gear status and sends alerts when an item is missing. The app communicates with the ESP32 controller via Bluetooth. The ESP32 receives weight data from load sensors through HX711 modules. Based on this data, it detects if any item is missing and sends real-time updates to the app.

11

Class Diagram

This class diagram shows how the system works. The user logs in through the app. The ESP32 controller reads weights from sensors, compares them to expected values, and sends a notification if something is missing.

12

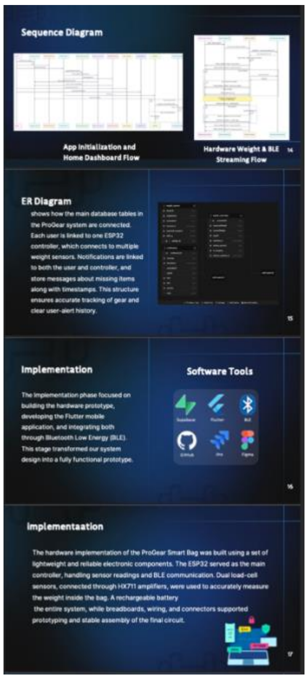
Sequence Diagram

User Login Flow

Bluetooth Connection and Data Streaming Flow

13

Figure 54 Pp 3



Sequence Diagram

App Initialization and Home Dashboard Flow

Hardware Weight & BLE Streaming Flow

14

ER Diagram

shows how the main database tables in the ProGear system are connected. Each user is linked to one ESP32 controller, which connects to multiple weight sensors. Notifications are linked to both the user and controller, and store messages about missing items along with timestamps. This structure ensures accurate tracking of gear and clear user-alert history.

15

Implementation

The implementation phase focused on building the hardware prototype, developing the Flutter mobile application, and integrating both through Bluetooth Low Energy (BLE). This stage transformed our system design into a fully functional prototype.

Software Tools

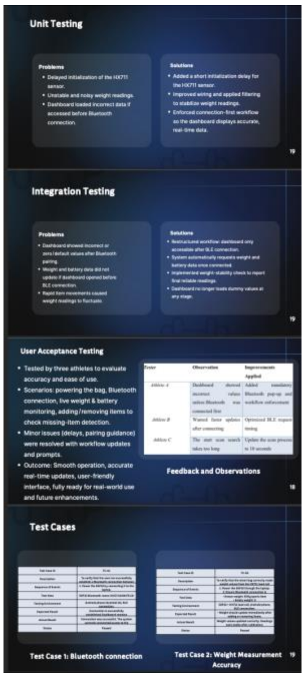
16

Implementaation

The hardware implementation of the ProGear Smart Bag was built using a set of lightweight and reliable electronic components. The ESP32 served as the main controller, handling sensor readings and BLE communication. Dual load-cell sensors, connected through HX711 amplifiers, were used to accurately measure the weight inside the bag. A rechargeable battery
the entire system, while breadboards, wiring, and connectors supported prototyping and stable assembly of the final circuit.

17

Figure 55 Pp 4



Unit Testing

Problems

- Delayed initialization of the HX711 sensor.
- Unstable and noisy weight readings.
- Dashboard loaded incorrect data if accessed before Bluetooth connection.

Solutions

- Added a short initialization delay for the HX711 sensor.
- Improved wiring and applied filtering to stabilize weight readings.
- Enforced connection-first workflow so the dashboard displays accurate, real-time data.

19

Integration Testing

Problems

- Dashboard showed incorrect or zero/default values after Bluetooth pairing.
- Weight and battery data did not update if dashboard opened before BLE connection.
- Rapid item movements caused weight readings to fluctuate.

Solutions

- Restructured workflow: dashboard only accessible after BLE connection.
- System automatically requests weight and battery data once connected.
- Implemented weight-stability check to report final reliable readings.
- Dashboard no longer loads dummy values at any stage.

19

User Acceptance Testing

- Tested by three athletes to evaluate accuracy and ease of use.
- Scenarios: powering the bag, Bluetooth connection, live weight & battery monitoring, adding/removing items to check missing-item detection.
- Minor issues (delays, pairing guidance) were resolved with workflow updates and prompts.
- Outcome: Smooth operation, accurate real-time updates, user-friendly interface, fully ready for real-world use and future enhancements.

Tester	Observation	Improvements Applied
Athlete A	Dashboard showed incorrect values unless Bluetooth was connected first	Added mandatory Bluetooth pop-up and workflow enforcement
Athlete B	Wanted faster updates after connecting	Optimized BLE request timing
Athlete C	The start scan search takes too long	Update the scan process to 10 seconds

Feedback and Observations

18

Test Cases

Test Case ID	TC-01
Description	[illegible]
Sequence of Events	[illegible]
Test Data	[illegible]
Testing Environment	[illegible]
Expected Result	[illegible]
Actual Result	[illegible]
Status	Passed

Test Case 1: Bluetooth connection

Test Case ID	TC-02
Description	[illegible]
Sequence of Events	[illegible]
Test Data	[illegible]
Testing Environment	[illegible]
Expected Result	[illegible]
Actual Result	[illegible]
Status	Passed

Test Case 2: Weight Measurement Accuracy

19

Figure 56 Pp 5

Hardware

20

Mobile App Interfaces

21

Mobile App Interfaces

21

Mobile App Interfaces

21

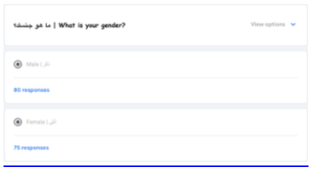
Figure 57 Pp 6

C. Miscellaneous

Questionnaire

You can access the questionnaire using the following link:

 https://forms.gle/PsrdYzi7iVa4tQNGA



ما هو جنسك؟ | What is your gender?

View options

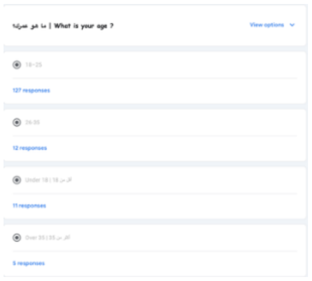
Male | ذكر

80 responses

Female | أنثى

75 responses

Figure 58 form 1



ما هو عمرك؟ | What is your age ?

View options

18–25

127 responses

26-35

12 responses

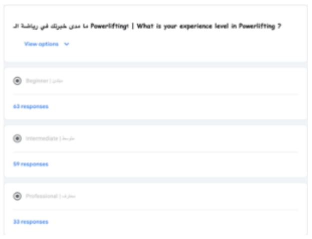
Under 18 | أقل من 18

11 responses

Over 35 | أكثر من 35

5 responses

Figure 59 form 2



ما مدى خبرتك في رياضة الـ Powerlifting؟ | What is your experience level in Powerlifting ?

View options

Beginner | مبتدئ

63 responses

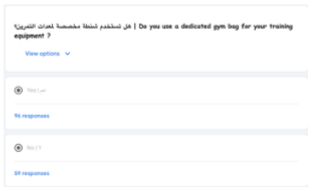
Intermediate | متوسط

59 responses

Professional | محترف

33 responses

Figure 60 form 3



هل تستخدم شنطة مخصصة لمعدات التمرين؟ | Do you use a dedicated gym bag for your training equipment ?

View options

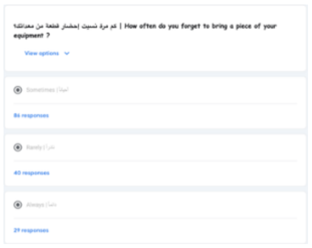
Yes | نعم

96 responses

No | لا

59 responses

Figure 61 form 4



كم مرة نسيت إحضار قطعة من معداتك؟ | How often do you forget to bring a piece of your equipment ?

View options

Sometimes | أحياناً

86 responses

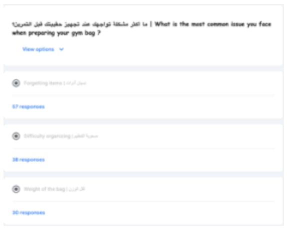
Rarely | نادراً

40 responses

Always | دائماً

29 responses

Figure 62 form 5


ما اكثر مشكلة تواجهك عند تجهيز حقيبتك قبل التمرين؟ | What is the most common issue you face when preparing your gym bag ?

View options

Forgetting items | نسيان أغراض

57 responses

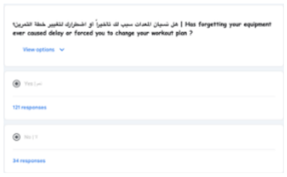
Difficulty organizing | صعوبة التنظيم

38 responses

Weight of the bag | ثقل الوزن

30 responses

Figure 63 form 6


هل نسيان المعدات سبب لك تاخيراً او اضطرارك لتغيير خطة التمرين؟ | Has forgetting your equipment ever caused delay or forced you to change your workout plan ?

View options

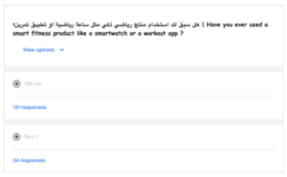
Yes | نعم

121 responses

No | لا

34 responses

Figure 64 form 7


هل سبق لك استخدام منتج رياضي تقني مثل ساعة رياضية او تطبيق تمرين؟ | Have you ever used a smart fitness product like a smartwatch or a workout app ?

View options

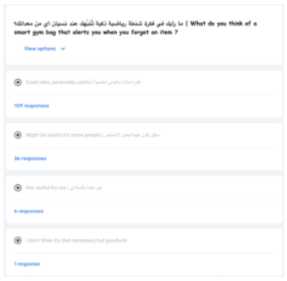
Yes | نعم

121 responses

No | لا

34 responses

Figure 65 form 8


ما رايك في فكرة شنطة رياضية ذكية تُنبّهك عند نسيان اي من معداتك؟ | What do you think of a smart gym bag that alerts you when you forget an item ?

View options

Great idea, personally useful | فكرة ممتازة وتفيدني شخصياً

109 responses

Might be useful for some people | ممكن تكون مفيدة لبعض الأشخاص

36 responses

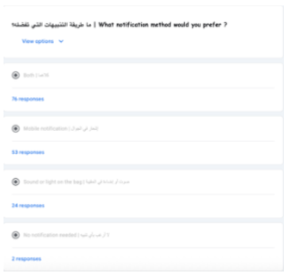
Not useful for me | غير مفيدة بالنسبة لي

6 responses

i don't think it's that necessary but goodluck

1 response

Figure 66 form 9


ما طريقة التنبيهات التي تفضلها؟ | What notification method would you prefer ?

View options

Both | كلاهما

76 responses

Mobile notification | إشعار في الجوال

53 responses

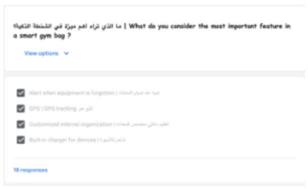
Sound or light on the bag | صوت أو إضاءة في الحقيبة

24 responses

No notification needed | لا أرغب بأي تنبيه

2 responses

Figure 67 form 10

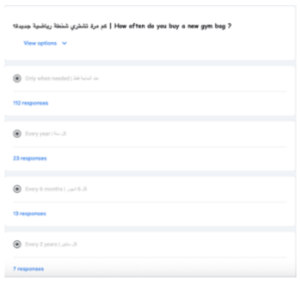

ما الذي تراه اهم ميزة في الشنطة الذكية؟ | What do you consider the most important feature in a smart gym bag ?

View options

☑ Alert when equipment is forgotten | تنبيه عند نسيان المعدات
☑ GPS | GPS tracking تتبع عبر
☑ Customized internal organization | تنظيم داخلي مخصص للمعدات
☑ Built-in charger for devices | شاحن للأجهزة

18 responses

Figure 68 form 11



كم مرة تشتري شنطة رياضية جديدة؟ | How often do you buy a new gym bag ?

View options

Only when needed | عند الحاجة فقط

112 responses

Every year | كل سنة

23 responses

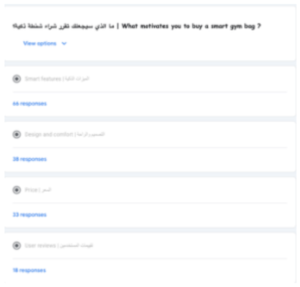
Every 6 months | كل 6 أشهر

13 responses

Every 2 years | كل سنتين

7 responses

Figure 69 form 12



ما الذي سيجعلك تقرر شراء شنطة ذكية؟ | What motivates you to buy a smart gym bag ?

View options

Smart features | الميزات الذكية

66 responses

Design and comfort | التصميم والراحة

38 responses

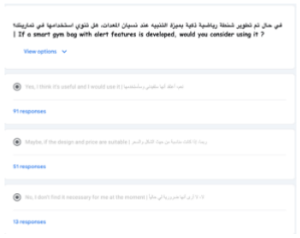
Price | السعر

33 responses

User reviews | تقييمات المستخدمين

18 responses

Figure 70 form 13



في حال تم تطوير شنطة رياضية ذكية بميزة التنبيه عند نسيان المعدات، هل تنوي استخدامها في تمارينك؟
| If a smart gym bag with alert features is developed, would you consider using it ?

View options

Yes, I think it's useful and I would use it | نعم، أعتقد أنها ستفيدني وسأستخدمها

91 responses

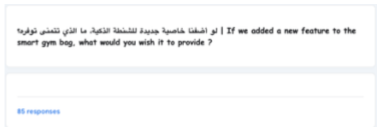
Maybe, if the design and price are suitable | ربما إذا كانت مناسبة من حيث الشكل والسعر

51 responses

No, I don't find it necessary for me at the moment | لا، لا أرى أنها ضرورية لي حالياً

13 responses

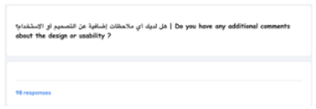
Figure 71 form 14



لو اضفنا خاصية جديدة للشنطة الذكية، ما الذي تتمنى توفره؟ | If we added a new feature to the smart gym bag, what would you wish it to provide ?

85 responses

Figure 72 form 15



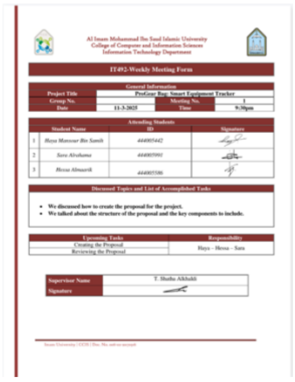
هل لديك اي ملاحظات إضافية عن التصميم او الإستخدام؟ | Do you have any additional comments about the design or usability ?

98 responses

Figure 73 form 16

D. Meetings

Weekly Meetings



Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

IT492-Weekly Meeting Form

General Information			
Project Title	ProGear Bag: Smart Equipment Tracker		
Group No.		Meeting No.	1
Date	11-3-2025	Time	9:30pm

Attending Students			
	Student Name	ID	Signature
1	Haya Mansour Bin Samih	444005442	[illegible]
2	Sara Alrahama	444005991	[illegible]
3	Hessa Almaarik	444005586	[illegible]

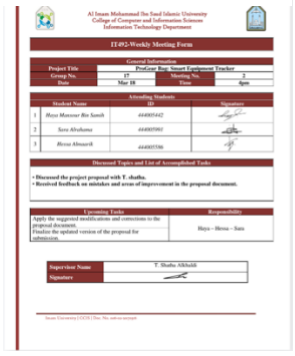
Discussed Topics and List of Accomplished Tasks

- We discussed how to create the proposal for the project.
- We talked about the structure of the proposal and the key components to include.

Upcoming Tasks	Responsibility
Creating the Proposal	Haya – Hessa – Sara
Reviewing the Proposal	

Supervisor Name	T. Shatha Alkhaldi
Signature	[illegible]

Figure 74 meeting 1



Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

IT492-Weekly Meeting Form

General Information			
Project Title	ProGear Bag: Smart Equipment Tracker		
Group No.	17	Meeting No.	2
Date	Mar 18	Time	4pm

Attending Students			
	Student Name	ID	Signature
1	Haya Mansour Bin Samih	444005442	[illegible]
2	Sara Alrahama	444005991	[illegible]
3	Hessa Almaarik	444005586	[illegible]

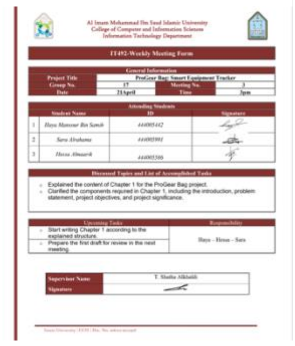
Discussed Topics and List of Accomplished Tasks

- Discussed the project proposal with T. shatha.
- Received feedback on mistakes and areas of improvement in the proposal document.

Upcoming Tasks	Responsibility
Apply the suggested modifications and corrections to the proposal document.	Haya – Hessa – Sara
Finalize the updated version of the proposal for submission.	

Supervisor Name	T. Shatha Alkhaldi
Signature	[illegible]

Figure 75 meeting 2



Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

IT492-Weekly Meeting Form

General Information			
Project Title	ProGear Bag: Smart Equipment Tracker		
Group No.	17	Meeting No.	3
Date	21April	Time	3pm

Attending Students			
	Student Name	ID	Signature
1	Haya Mansour Bin Samih	444005442	[illegible]
2	Sara Alrahama	444005991	[illegible]
3	Hessa Almaarik	444005586	[illegible]

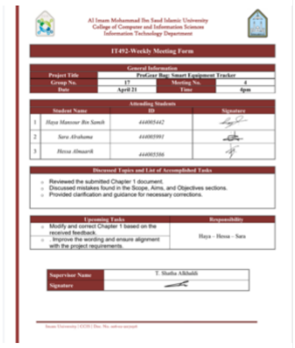
Discussed Topics and List of Accomplished Tasks

- Explained the content of Chapter 1 for the ProGear Bag project.
- Clarified the components required in Chapter 1, including the introduction, problem statement, project objectives, and project significance.

Upcoming Tasks	Responsibility
Start writing Chapter 1 according to the explained structure.	Haya – Hessa – Sara
Prepare the first draft for review in the next meeting.	

Supervisor Name	T. Shatha Alkhaldi
Signature	[illegible]

Figure 76 meeting 3



Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

IT492-Weekly Meeting Form

General Information			
Project Title	ProGear Bag: Smart Equipment Tracker		
Group No.	17	Meeting No.	4
Date	April 21	Time	4pm

Attending Students			
	Student Name	ID	Signature
1	Haya Mansour Bin Samih	444005442	[illegible]
2	Sara Alrahama	444005991	[illegible]
3	Hessa Almaarik	444005586	[illegible]

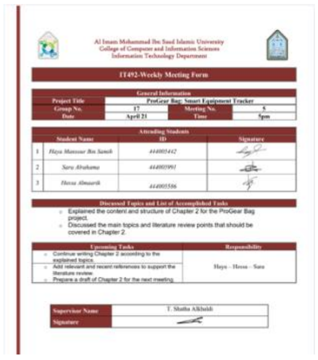
Discussed Topics and List of Accomplished Tasks

- Reviewed the submitted Chapter 1 document.
- Discussed mistakes found in the Scope, Aims, and Objectives sections.
- Provided clarification and guidance for necessary corrections.

Upcoming Tasks	Responsibility
Modify and correct Chapter 1 based on the received feedback.	Haya – Hessa – Sara
. Improve the wording and ensure alignment with the project requirements.	

Supervisor Name	T. Shatha Alkhaldi
Signature	[illegible]

Figure 77 meeting 4



Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

IT492-Weekly Meeting Form

General Information			
Project Title	ProGear Bag: Smart Equipment Tracker		
Group No.	17	Meeting No.	5
Date	April 21	Time	5pm

Attending Students			
	Student Name	ID	Signature
1	Haya Mansour Bin Samih	444005442	[illegible]
2	Sara Alrahama	444005991	[illegible]
3	Hessa Almaarik	444005586	[illegible]

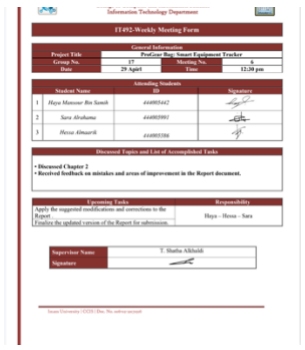
Discussed Topics and List of Accomplished Tasks

- Explained the content and structure of Chapter 2 for the ProGear Bag project.
- Discussed the main topics and literature review points that should be covered in Chapter 2.

Upcoming Tasks	Responsibility
Continue writing Chapter 2 according to the explained topics. Add relevant and recent references to support the literature review. Prepare a draft of Chapter 2 for the next meeting.	Haya – Hessa – Sara

Supervisor Name	T. Shatha Alkhaldi
Signature	[illegible]

Figure 78 meeting 5



Information Technology Department

IT492-Weekly Meeting Form

General Information			
Project Title	ProGear Bag: Smart Equipment Tracker		
Group No.	17	Meeting No.	6
Date	29 April	Time	12:30 pm

Attending Students			
	Student Name	ID	Signature
1	Haya Mansour Bin Samih	444005442	[illegible]
2	Sara Alrahama	444005991	[illegible]
3	Hessa Almaarik	444005586	[illegible]

Discussed Topics and List of Accomplished Tasks

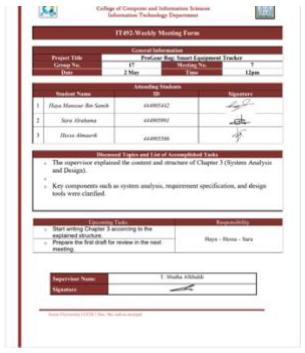
- **Discussed Chapter 2**
- **Received feedback on mistakes and areas of improvement in the Report document.**

Upcoming Tasks	Responsibility
Apply the suggested modifications and corrections to the Report . Finalize the updated version of the Report for submission.	Haya – Hessa – Sara

Supervisor Name	T. Shatha Alkhaldi
Signature	[illegible]

Imam University | CCIS | Doc. No. 006-01-201710316

Figure 79 meeting 6



College of Computer and Information Sciences
Information Technology Department

IT492-Weekly Meeting Form

General Information			
Project Title	ProGear Bag: Smart Equipment Tracker		
Group No.	17	Meeting No.	7
Date	2 May	Time	12pm

Attending Students			
	Student Name	ID	Signature
1	Haya Mansour Bin Samih	444005442	[illegible]
2	Sara Alrahama	444005991	[illegible]
3	Hessa Almaarik	444005586	[illegible]

Discussed Topics and List of Accomplished Tasks

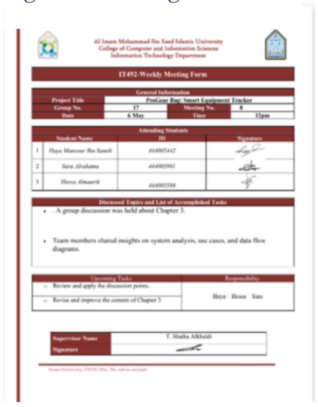
- The supervisor explained the content and structure of Chapter 3 (System Analysis and Design).
- Key components such as system analysis, requirement specification, and design tools were clarified.

Upcoming Tasks	Responsibility
Start writing Chapter 3 according to the explained structure. Prepare the first draft for review in the next meeting.	Haya – Hessa – Sara

Supervisor Name	T. Shatha Alkhaldi
Signature	[illegible]

Imam University | CCIS | Doc. No. 006-01-201710316

Figure 80 meeting 7



Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

IT492-Weekly Meeting Form

General Information			
Project Title	ProGear Bag: Smart Equipment Tracker		
Group No.	17	Meeting No.	8
Date	6 May	Time	12pm

Attending Students			
	Student Name	ID	Signature
1	Haya Mansour Bin Samih	444005442	[illegible]
2	Sara Alrahama	444005991	[illegible]
3	Hessa Almaarik	444005586	[illegible]

Discussed Topics and List of Accomplished Tasks

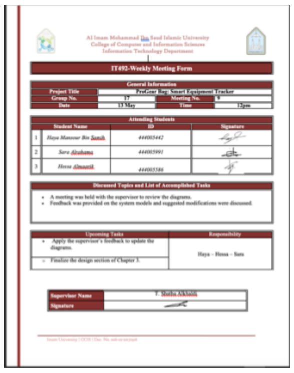
- . A group discussion was held about Chapter 3.
- Team members shared insights on system analysis, use cases, and data flow diagrams.

Upcoming Tasks	Responsibility
Review and apply the discussion points. Revise and improve the content of Chapter 3	Haya – Hessa – Sara

Supervisor Name	T. Shatha Alkhaldi
Signature	[illegible]

Imam University | CCIS | Doc. No. 006-01-201710316

Figure 81 meeting 8



Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

IT492-Weekly Meeting Form

General Information			
Project Title	ProGear Bag: Smart Equipment Tracker		
Group No.	17	Meeting No.	9
Date	13 May	Time	12pm

Attending Students			
	Student Name	ID	Signature
1	Haya Mansour Bin Samih	444005442	[illegible]
2	Sara Alrahama	444005991	[illegible]
3	Hessa Almaarik	444005586	[illegible]

Discussed Topics and List of Accomplished Tasks

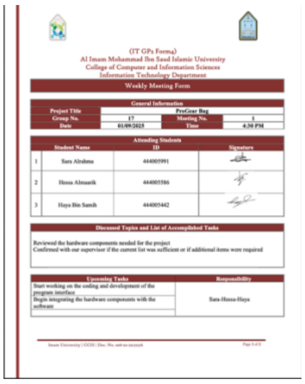
- A meeting was held with the supervisor to review the diagrams.
- Feedback was provided on the system models and suggested modifications were discussed.

Upcoming Tasks	Responsibility
• Apply the supervisor's feedback to update the diagrams.	Haya – Hessa – Sara
○ Finalize the design section of Chapter 3.	

Supervisor Name	T. Shatha Alkhaldi
Signature	[illegible]

Imam University | CCIS | Doc. No. [illegible]

Figure 82 meeting 9



(IT GP2 Form4)
Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

Weekly Meeting Form

General Information			
Project Title	ProGear Bag		
Group No.	17	Meeting No.	1
Date	01/09/2025	Time	4:30 PM

Attending Students			
	Student Name	ID	Signature
1	Sara Alrahma	444005991	[illegible]
2	Hessa Almaarik	444005586	[illegible]
3	Haya Bin Samih	444005442	[illegible]

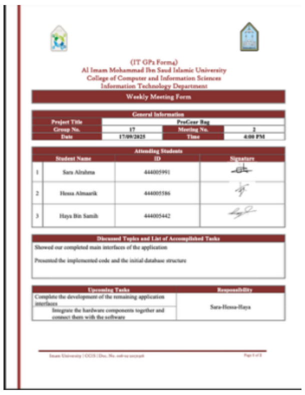
Discussed Topics and List of Accomplished Tasks

Reviewed the hardware components needed for the project
Confirmed with our supervisor if the current list was sufficient or if additional items were required

Upcoming Tasks	Responsibility
Start working on the coding and development of the program interface	Sara-Hessa-Haya
Begin integrating the hardware components with the software	

Imam University | CCIS | Doc. No. [illegible] Page 1 of 2

Figure 83 meeting 10



(IT GP2 Form4)
Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

Weekly Meeting Form

General Information			
Project Title	ProGear Bag		
Group No.	17	Meeting No.	2
Date	17/09/2025	Time	4:00 PM

Attending Students			
	Student Name	ID	Signature
1	Sara Alrahma	444005991	[illegible]
2	Hessa Almaarik	444005586	[illegible]
3	Haya Bin Samih	444005442	[illegible]

Discussed Topics and List of Accomplished Tasks

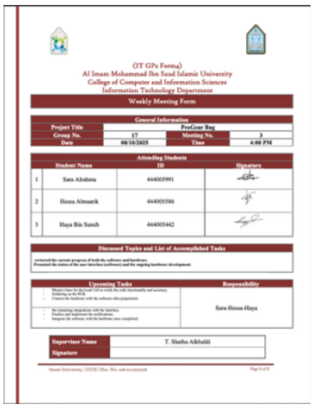
Showed our completed main interfaces of the application

Presented the implemented code and the initial database structure

Upcoming Tasks	Responsibility
Complete the development of the remaining application interfaces	Sara-Hessa-Haya
Integrate the hardware components together and connect them with the software	

Imam University | CCIS | Doc. No. [illegible] Page 1 of 2

Figure 84 meeting 11



(IT GP2 Form4)
Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

Weekly Meeting Form

General Information			
Project Title	ProGear Bag		
Group No.	17	Meeting No.	3
Date	08/10/2025	Time	4:00 PM

Attending Students			
	Student Name	ID	Signature
1	Sara Alrahma	444005991	[illegible]
2	Hessa Almaarik	444005586	[illegible]
3	Haya Bin Samih	444005442	[illegible]

Discussed Topics and List of Accomplished Tasks

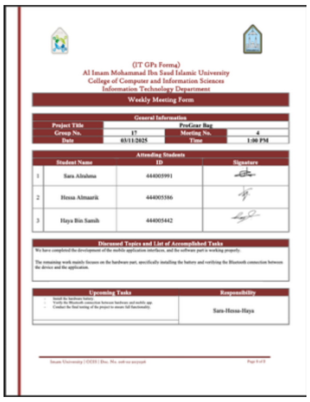
reviewed the current progress of both the software and hardware.
Presented the status of the user interface (software) and the ongoing hardware development.

Upcoming Tasks	Responsibility
- Mount a base for the Load Cell to verify the code functionality and accuracy. - Soldering on the PCB. - Connect the hardware with the software after preparation.	Sara-Hessa-Haya
- the remaining integrations with the interface. - Finalize and implement the notifications. - Integrate the software with the hardware once completed.	

Supervisor Name	T. Shatha Alkhaldi
Signature	

Imam University | CCIS | Doc. No. [illegible] Page 2 of 2

Figure 85 meeting 12



(IT GP2 Form4)
Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

Weekly Meeting Form

General Information			
Project Title	ProGear Bag		
Group No.	17	Meeting No.	4
Date	03/11/2025	Time	1:00 PM

Attending Students			
	Student Name	ID	Signature
1	Sara Alrahma	444005991	[illegible]
2	Hessa Almaarik	444005586	[illegible]
3	Haya Bin Samih	444005442	[illegible]

Discussed Topics and List of Accomplished Tasks

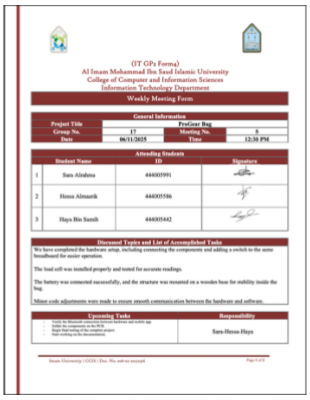
We have completed the development of the mobile application interfaces, and the software part is working properly.

The remaining work mainly focuses on the hardware part, specifically installing the battery and verifying the Bluetooth connection between the device and the application.

Upcoming Tasks	Responsibility
- Install the hardware battery. - Verify the Bluetooth connection between hardware and mobile app. - Conduct the final testing of the project to ensure full functionality.	Sara-Hessa-Haya

Imam University | CCIS | Doc. No. 006-01-20171036 Page 8 of 8

Figure 86 meeting 13



(IT GP2 Form4)
Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Information Technology Department

Weekly Meeting Form

General Information			
Project Title	ProGear Bag		
Group No.	17	Meeting No.	5
Date	06/11/2025	Time	12:30 PM

Attending Students			
	Student Name	ID	Signature
1	Sara Alrahma	444005991	[illegible]
2	Hessa Almaarik	444005586	[illegible]
3	Haya Bin Samih	444005442	[illegible]

Discussed Topics and List of Accomplished Tasks

We have completed the hardware setup, including connecting the components and adding a switch to the same breadboard for easier operation.

The load cell was installed properly and tested for accurate readings.

The battery was connected successfully, and the structure was mounted on a wooden base for stability inside the bag.

Minor code adjustments were made to ensure smooth communication between the hardware and software.

Upcoming Tasks	Responsibility
- Verify the Bluetooth connection between hardware and mobile app. - Solder the components on the PCB. - Begin final testing of the complete project. - Start working on the documentation.	Sara-Hessa-Haya

Imam University | CCIS | Doc. No. 006-01-20171036 Page 8 of 8

Figure 87 meeting 14

E. User Manual

progear

User Manual

Figure 88 UM1

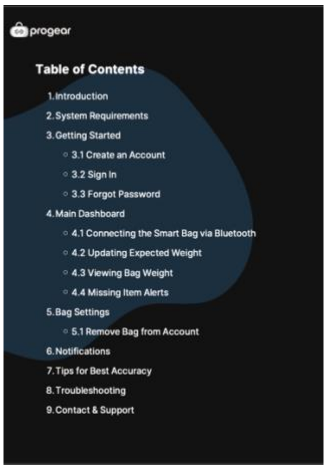
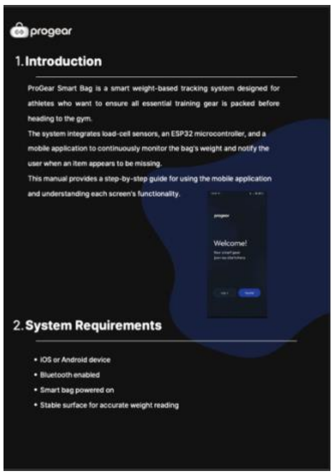

progear

Table of Contents

1. Introduction
2. System Requirements
3. Getting Started
   - 3.1 Create an Account
   - 3.2 Sign In
   - 3.3 Forgot Password
4. Main Dashboard
   - 4.1 Connecting the Smart Bag via Bluetooth
   - 4.2 Updating Expected Weight
   - 4.3 Viewing Bag Weight
   - 4.4 Missing Item Alerts
5. Bag Settings
   - 5.1 Remove Bag from Account
6. Notifications
7. Tips for Best Accuracy
8. Troubleshooting
9. Contact & Support

Figure 89 UM2


progear

1. Introduction

ProGear Smart Bag is a smart weight-based tracking system designed for athletes who want to ensure all essential training gear is packed before heading to the gym.

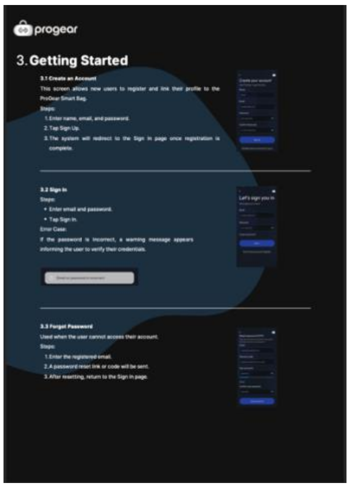
The system integrates load-cell sensors, an ESP32 microcontroller, and a mobile application to continuously monitor the bag's weight and notify the user when an item appears to be missing.

This manual provides a step-by-step guide for using the mobile application and understanding each screen's functionality.

2. System Requirements

- IOS or Android device
- Bluetooth enabled
- Smart bag powered on
- Stable surface for accurate weight reading

Figure 90 UM3



progear

3. **Getting Started**

3.1 Create an Account

This screen allows new users to register and link their profile to the ProGear Smart Bag.

Steps:

1. Enter name, email, and password.
2. Tap Sign Up.
3. The system will redirect to the Sign in page once registration is complete.

3.2 Sign In

Steps:

- Enter email and password.
- Tap Sign In.

Error Case:

If the password is incorrect, a warning message appears informing the user to verify their credentials.

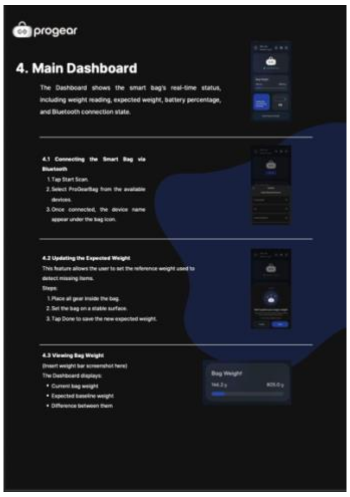
3.3 Forgot Password

Used when the user cannot access their account.

Steps:

1. Enter the registered email.
2. A password reset link or code will be sent.
3. After resetting, return to the Sign in page.

Figure 91 UM4



progear

4. Main Dashboard

The Dashboard shows the smart bag's real-time status, including weight reading, expected weight, battery percentage, and Bluetooth connection state.

4.1 Connecting the Smart Bag via Bluetooth

1. Tap Start Scan.
2. Select ProGearBag from the available devices.
3. Once connected, the device name appear under the bag icon.

4.2 Updating the Expected Weight

This feature allows the user to set the reference weight used to detect missing items.

Steps:

1. Place all gear inside the bag.
2. Set the bag on a stable surface.
3. Tap Done to save the new expected weight.

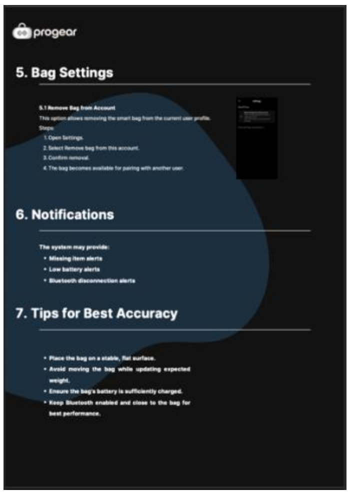
4.3 Viewing Bag Weight

(Insert weight bar screenshot here)

The Dashboard displays:

- Current bag weight
- Expected baseline weight
- Difference between them

Figure 92 UM5



progear

5. Bag Settings

5.1 Remove Bag from Account

This option allows removing the smart bag from the current user profile.

Steps:

1. Open Settings.
2. Select Remove bag from this account.
3. Confirm removal.
4. The bag becomes available for pairing with another user.

6. Notifications

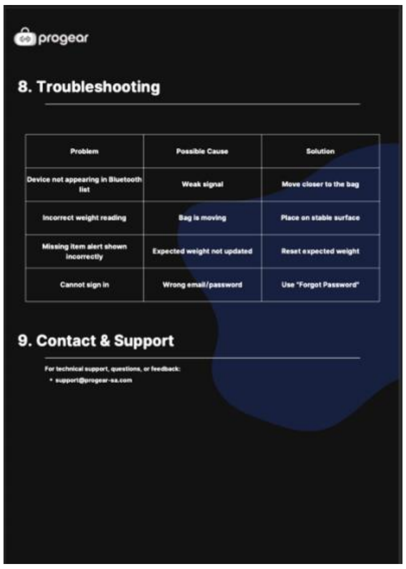
The system may provide:

- **Missing item alerts**
- **Low battery alerts**
- **Bluetooth disconnection alerts**

7. Tips for Best Accuracy

- **Place the bag on a stable, flat surface.**
- **Avoid moving the bag while updating expected weight.**
- **Ensure the bag's battery is sufficiently charged.**
- **Keep Bluetooth enabled and close to the bag for best performance.**

Figure 93 UM6



progear

8. Troubleshooting

Problem	Possible Cause	Solution
Device not appearing in Bluetooth list	Weak signal	Move closer to the bag
Incorrect weight reading	Bag is moving	Place on stable surface
Missing item alert shown incorrectly	Expected weight not updated	Reset expected weight
Cannot sign in	Wrong email/password	Use "Forgot Password"

9. Contact & Support

For technical support, questions, or feedback:

- support@progear-sa.com

Figure 94 UM7