Design and Development of Software Model for Therapeutic Medical Devices

By:

M Siqlain Hanif Khan

(Registration No.: MS-CSE-20-330423)

Supervisor

Dr. Muhammad Usman Akram

Co Supervisor

Dr. Sajid Gul Khawaja

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING,
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING,
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,
ISLAMABAD

July 10, 2024

Annex A

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by NS **Siqlain Hanif** Registration No. 00000330423, of College of E&ME has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the thesis.

Signature : [illegible]

Name of Supervisor: Dr Muhammad Usman Akrm

Date: 10-07-2024

Signature of HOD: [illegible]
(Dr Usman Qamar)
Date: 10-07-2024

Signature of Dean: Rashid
(Brig Dr Nasir Rashid)
Date: 10-07-2024

Design and Development of Software Model for Therapeutic Medical Devices

By:

M Siqlain Hanif Khan

(Registration No.: MS-CSE-20-330423)

A thesis submitted to National University of Sciences and Technology, Islamabad
in partial fulfillment of the requirements for the degree of
Master of Sciences in Software Engineering

Supervisor

Dr. Muhammad Usman Akram

Co Supervisor

Dr. Sajid Gul Khawaja

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING,
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING,
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,
ISLAMABAD

July 10, 2024

Dedicated to my exceptional parents whose tremendous support and cooperation led me to this accomplishment!

ACKNOWLEDGEMENTS

All praise and glory to Almighty Allah (the most glorified, the highest) who gave me the courage, patience, knowledge, and ability to carry out this work and to persevere and complete it satisfactorily. Undoubtedly, HE eased my way and without HIS blessings I can achieve nothing.

I would like to express my sincere gratitude to my advisor Dr. Muhammad Usman Akram for boosting my morale and for his continual assistance, motivation, dedication, and invaluable guidance in my quest for knowledge. I am blessed to have such a co-operative advisor and kind mentor for my research.

Along with my advisor, I would like to acknowledge my entire thesis committee: Assistant Professor Dr. Muhammad Yasin and Lecturer Anum Abdul Salam for their cooperation and prudent suggestions.

My acknowledgement would be incomplete without thanking the biggest source of my strength, my family. I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout every department of my life.

Finally, I would like to express my gratitude to all my friends and the individuals who have encouraged and supported me through this entire period.

ABSTRACT

Healthcare equipment include a vast range of goods, from basic equipment to complex software systems. Incorporating software into such equipment is difficult since it requires market permission and approval from regional regulatory organizations. As a consequence, medical enterprises must oversee both the creation of these equipment and compliance with regulatory norms and standards. While regulatory organizations do not specify a particular model, they do need careful consideration. Plan-driven techniques, on the other hand, tend to stifle software development and shifts, while agile techniques are frequently criticized for insufficient preparation and paperwork.

The goal of our investigations is to offer an adequate process framework for developing health care devices while taking regulatory constraints into account. We first introduced the Improved Agile the V-Model (EAV) after conducting a thorough examination of the previous research and McHugh's suggested framework. This model combines plan-driven and agile methodologies. We then linked the suggested solution with the MDEVSPICE architecture to verify that it meets the IEC62304 standard. Finally, we assessed the suggested model using an actual case study utilizing a wave therapy device for medicine.

The EAV strategy flexibility for both waterfall and agile processes makes it easier to incorporate novel demands into medical equipment, and the suggested systems design methodology helps in software and hardware interaction. Connecting the EAV paradigm to the MDEVSPICE infrastructure indicates total compliance with requirements. In addition, the proposed approach was statistically tested and effectively executed in our particular study. We assessed equipment usage along with effectiveness indicators, and most questions had a trust level of $P<0.05$.

The suggested design meets regulatory requirements and has already been used effectively in the production of a wave therapy equipment. However, its usefulness for smaller and simple medical goods need to be confirmed, which can be evaluated via the model.

Key Words: "Healthcare equipment creation, Software engineering either Subsystem-level break down, SDLC, Enhanced Agile V-Model (EAV)"

CONTENTS

ACKNOWLEDGEMENTS .... **II**

ABSTRACT .... **III**

LIST OF FIGURES .... **VI**

LIST OF TABLE .... **VIII**

CHAPTER 1 .... **1**

1.1 INTRODUCTION 1
1.2 MOTIVATION 1
1.3 PROBLEM STATEMENT 2
1.4 AIMS AND OBJECTIVES 3
1.5 STRUCTURE OF THESIS 3

CHAPTER 2 .... **4**

2.1 BACKGROUND 4
2.2 LITERATURE REVIEW 5
2.3 RESEARCH GAP 7

CHAPTER 3 .... **10**

3.1 PROPOSED METHODOLOGY 10
3.2 MDEVSPICE AND EAV-MODEL 13
3.2.1 MDevSPICE Processes of Medical Device System Life Cycle Processes 14
3.2.2 MDevSPICE Processes of Medical Device Software Life Cycle Processes 16
3.2.3 MDevSPICE Processes of Medical Device Support Processes 18

CHAPTER 4 .... **21**

4.1 INTRODUCTION 21
4.2 CHAPTER CONVENTIONS 22
4.3 PROJECT PERSPECTIVE 22
4.4 OVERALL DESCRIPTION 23
4.4.1 Product Description *23*
4.4.2 Product Features *23*
4.4.2.1 The Android Testing Controllers Application for Portable Devices 24
4.4.2.2 Completely Functional Smartphone App for Handheld Device 24
4.4.2.3 Embedded Software for clinic 24
4.4.2.4 Web Application as central data repository 25

4.4.3 User Class and Characteristics 25
4.4.4 Software Development Life Cycle Approach of the Product 25
4.4.5 Requirements Specification 28
4.4.5.1 Functional Requirements 30
4.4.5.2 Non-Functional Requirements 35
4.4.6 Use Case Diagram 36
4.4.6.1 **Comprehensive explanations of use cases** 39
4.5 ARCHITECTURE AND DESIGN 47
4.5.1 Logical Architecture 47
4.5.2 System Design 48
4.5.3 Class Diagram 48
4.5.4 Sequence Diagram 48
4.6 IMPLEMENTATION 54
4.6.1 Hardware 54
4.6.2 Software 55
4.7 TESTING 55
4.7.1 Test Items 55
4.7.2 Approach 58
4.7.3 Item passing or failing Criteria 59
4.7.4 Test Case Description 59
4.8 SUMMARY 59
CHAPTER 5 61
CONCLUSION AND FUTURE WORK 61
REFERENCES

LIST OF FIGURES

Figure 1.1: Plot depicting the evolution of computerized health care systems throughout the years. 2

Figure 3.1: Enhanced AV-Model blends agile approaches with the conventional plan-driven design method. 12

Figure 4.1: Depicts the indicated sense motor positions where the device will be attached. These sites transmit vibrating pulses that stimulate the neurons in the brain. 23

Figure 4.2: Figure 4.2: Medical Technology Life Cycle: The concept begins with basic research and evolves through applied studies. If the applied study receives funding, it is necessary to do an economic feasibility and risk evaluation in order to develop the end result. The creation of products entails merging several development factors, such as both hardware and software (either electrical or mechanical principles, depending on the demands). At last, after creation, the gadget must be approved by the appropriate nation-specific regulatory organization for entry into the market and acceptance. 26

Figure 4.3: System breakdown into subsystems allows for distinct processing of criteria for each subsystem. 28

Figure 4.4: Depicts the classifications of requirements for every subsystem 29

Figure 4.5: The numbering conventions for the different categories of specifications for simpler maintenance 30

Figure 4.6: A Case Study Representation of the System-User Communication for the Hospital Variant of Integrated Program. 37

Figure 4.7: Case Study Model of System-User Communication for the Handheld Variant of the Healthcare Device Test Controller Smartphone App 37

Figure 4.8: It depicts the relationship between the software and the user in the portable variant of the Healthcare Device Smartphone App. 38

Figure 4.9: It depicts the relationship involving the system and the end user of a website application 39

Figure 4.10: The device's logical design, which is an n-tier architecture with the physical technological layer indicated as the very last layer. 47

Figure 4.11: Sequence diagram of the "User Login" in the Subsystem SS3 Smartphone app, demonstrating connection with the Object Individual 49

Figure 4.12: Sequence Diagram of the usage scenario "Connect Device via Bluetooth connectivity" for the Subsystem-level SS3 Smartphone App, with the Device Entity reflecting essential hardware characteristics. 50

Figure 4.13: A Sequence Diagram of the usage scenario "Change Password" for the SS3 Smartphone app, with the Home User Object including Fundamental individual Information, Individual Details, and Register Details. 51

Figure 4.14: Sequence Diagram for the usage scenario "View Therapy Status," The main Home User Object contains fundamental personal data, whereas the Session Therapy Object contains comprehensive details on the individual's therapeutic sessions. 52

Figure 4.15: Sequence Diagram for the usage scenario "Manage Therapy Session," The Home User Property contains private data, the Session Therapy Property contains descriptions of therapy sessions, and the Nerve Point Property has data concerning every single nerve point, while the Device Property contains data about the hardware. 53

Figure 4.16: Deployment Diagram 54

LIST OF TABLE

Table 3-1: Limits in McHugh's Agile-V Approach for Healthcare Equipment Design 11
Table 4-1: Chapter conventions: Terms with important definitions 21
Table 4-2: Functional specifications for Subsystem 'SS3' - Completely Functioning Android Smartphone Application for Domestic Use of Handheld Wave Therapy Equipment 31
Table 4-3: Technical specifications for Subsystem 'SS4' - Fully Practical Test Controller Smartphone App for Handheld Wave Therapeutic Equipment for Domestic Usage. 32
Table 4-4: Technical specifications for the subsystem 'SS2' - Integrated Software 32
Table 4-5: Specifications for Subsystem-level 'SS1' and 'SS5' Equipment for Either Handheld or Hospital Editions of the Wave Therapeutic Medical Device. 33
Table 4-6: The functional specifications for Subsystem-level 'SS6' web-based app. 34
Table 4-7: The functional specifications for the subsystems-level 'SS6' web-based application. 35
Table 4-8: Comprehensive Overview of Use Case "Login" to SS3. 40
Table 4-9: A Comprehensive Overview of the Case for "Register/Signup" in SS3 40
Table 4-10: Comprehensive Overview of the usage scenario "Request Patient Profile" in SS3 41
Table 4-11: Comprehensive overview of the Case for "Change Password" in SS3 42
Table 4-12: Comprehensive Overview of the Case Study "Pending Patient Requests" in SS3 43
Table 4-13: Comprehensive Overview of the Case Study “View Patient Therapy Status “in SS3 ... 43
Table 4-14: Comprehensive Overview of the Case Study “Manage Patient Therapy Session" in SS3 44
Table 4-15: Comprehensive Overview of the Case Study “Connect Device via Bluetooth “in SS3 45
Table 4-16: Comprehensive Overview of the Case Study “Manage patient profile" in SS3 45
Table 4-17: Comprehensive Overview of the Case Study “Logout “in SS3 46
Table 4-18: “Login” Functionality Test Case 56
Table 4-19: “Sign up Request.” Functionality Test Case 56
Table 4-20: “Device Connection via Bluetooth.” Functionality Test Case 56
Table 4-21: “Start Therapy Session.” Functionality Test Case 57
Table 4-22: “Manage Therapy Session.” Functionality Test Case 57
Table 4-23: “Change Password” Functionality Test Case 58
Table 4-24: “View Recent Therapy Activities.” Functionality Test Case 58

CHAPTER 1

1.1 Introduction

In the past decade, the integration between health studies and technological developments has resulted in significant advances in the healthcare business. Figure 1.1 depicts the healthcare business's shift to the usage of digital technology and computerized healthcare systems in recent decades.. Among these, therapeutic medical devices have emerged as pivotal tools in delivering targeted treatments and improving patient outcomes. Concurrently, the integration of software solutions within these devices has opened new avenues for precision, adaptability, and real-time monitoring. This intersection of medical devices and software development presents a promising domain for research and innovation

But one of the major concerns for software products is, "how safe are they?", as defects in such software might lead to inaccurate diagnosis which could be fatal. So, to comply the safety and the standardization of such systems Multiple rules have been suggested in various locations, such as the Code of Federal rules (CFR) in US law & the Medical Devices Directives (MDD) throughout the European Union, that serve as guidelines for developing and evaluating computer-assisted healthcare equipment., and these documents provide information about the design, production, testing, and performance of a medical device. This research aims to provide a design and development of software model the aids the developers to rigid for the regulations and flexible enough incorporate changes in less time.

1.2 Motivation

Digitization of healthcare facilities is currently one of the top technological trends, and every vendor wants their products to be available in the market as soon as possible, and at the same time their products need to be safe enough for intended audience use. To ensure the safety of the product, it goes through a rigorous process of approval from regulation authority to comply with the safety and standardization requirements, for that every aspect of the product needs to fulfill the requirements of the regulation authority and needs to be properly documented and traceable for future reference.

These requirements from the regulation body are mandatory to meet so to ensure the public safety but adds a bit of difficulty in introducing the product to the market in time, which gets even harder when the product needs to implement a change in the middle of things. So, keeping all the aspects in mind we want to introduce a development process that can help the product owners to easily implement changes and be on top of the requirements from the regulation body.

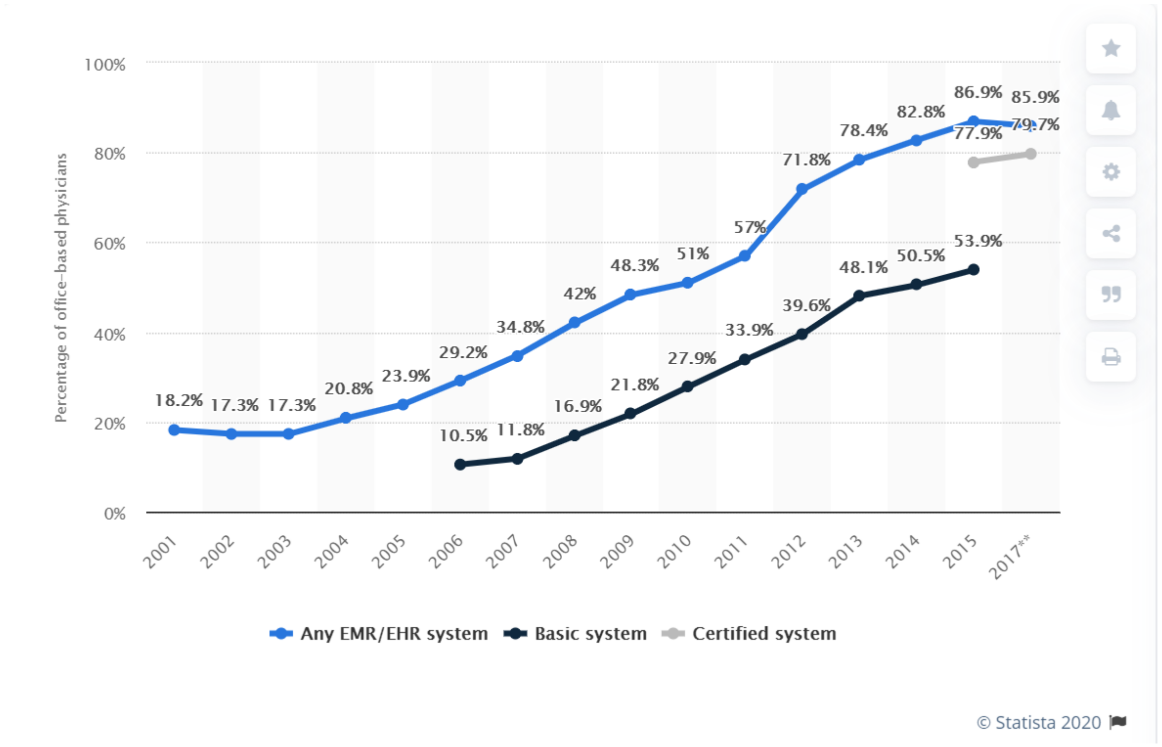


Figure 1.1: Plot depicting the evolution of computerized health care systems throughout the years.

1.3 Problem Statement

One of the major concerns in developing such a system is the choice of software development model as no regulation authority has mentioned any specific development model in designing or developing of medical based products.

The choice of software development model becomes more challenging as we need to the development process to be stable/formal enough to provides stability the regulation authority needs and adaptable/informal enough to cope up with the fast-growing industry.

In accordance with a survey undertaken by the Irish Healthcare Device Design Organization, fifty percent of healthcare products are produced utilizing the V-Model of manufacturing. Which is a very formal in nature and provides the stability for the regulation authority requirements but lacks the ability to adapt to changes in a fast manner.

The purpose of this research is to introduce a software construction model that will be stable enough as V-Model and adaptable enough as Agile model to help the medical product industry to develop the stable and safe product and deliver it in minimum time to the market.

1.4 Aims and Objectives

This research aims to address the challenges associated with designing and developing a robust software model for therapeutic medical devices. The primary objectives of this research are as follows:

- To analyze the existing regulatory frameworks and standards that govern the development and deployment of software-integrated therapeutic medical devices.
- To determine and recommend the most appropriate SDLC method for creating automated devices for medical use.
- To expand on complying with laws and regulations.

To illustrate its efficacy, an extensive case investigation will be conducted on the use of the suggested model in the creation of a wave therapy healthcare device.

1.5 Structure of Thesis

This structure of work as follows:

Chapter 2 covers the importance of software development lifecycle in the development of product.

Chapter 3 analyzes the available research and emphasizes major current studies on tuberculosis diagnostics.

Chapter 4 is an exhaustive description of the suggested approach.

Chapter 5 case study of proposed methodology

Chapter 6 wraps up the thesis and defines the subsequent scope of the study.

CHAPTER 2

2.1 Background

Software development life cycle: The Software Development Life Cycle (SDLC) is an organized process for producing excellent, affordable software in the lowest amount of time. Its primary purpose is to create solutions that meets and surpasses its customers' expectations and wants. SDLC offers a detailed plan organized into phases or steps, each of which has its own procedures and objectives. Compliance to the SDLC accelerates production while lowering the hazards and costs associated with alternate methods of manufacturing. While the stages or phases of SDLC may differ based on the technique or framework employed, the following is a popular and simplified portrayal of the SDLC:

- **Planning:** Planning is the phase in which project stakeholders determine the project's scope, objectives, needs, and limitations. It includes defining goals, selecting a budget, and developing a project timetable. Feasibility studies are undertaken to determine the project's technical, operational, and financial viability.
- **Analysis**: During this phase, the project team works with stakeholders to determine and document detailed software requirements. This step entails comprehending and specifying both the software's functionality (what it should accomplish) and non-functional characteristics (performance, security needs, etc.).
- **Design:** The design phase focuses on establishing the software's architecture and technical demands. This covers both high-level design, which defines the general structure plus components, and low-level design, which describes how particular components are implemented. User interface design is also an important part of this phase.
- **Implementation**: Throughout this stage, professionals implement the code itself according to the design criteria. This stage includes coding, unit testing (testing individual parts or code units), and finally integration (combining individual components to form a full system).
- **Testing**: Product testing assures proper functionality. It consists of multiple levels of testing, including unit testing, examination of integration (how various parts link), system testing (the comprehensive system), and user acceptance evaluation (end-user checking).

- **Deployment**: Once tested and approved, the program is operational. This process involves establishing up, implementing, and, if necessary, passing on the data.
- **Maintenance**: After installation, ongoing support is needed to address complications, improve the product, and adapt to changing needs. Maintenance actions include correcting (bug repairs), adapting (alterations in response to new demands or situations), faultless (functional upgrades), and preventative efforts.

The process of software development (SDLC) is an extensible paradigm that can be personalized to a wide range of project and strategies. Agile approaches, such as Kanban and Scrum, emphasize incremental and iterative development, whereas the Waterfall paradigm is more sequential and linear. The project life cycle (SDLC) framework has been modified to the project's specific needs and constraints. Prominent SDLC models include the following:

- **Waterfall**: One of the SDLC methodologies necessitate finishing each work step before moving on to the next one, just like the ways a waterfall flows.
- **Agile:** Agile is a modern software building approach that prioritizes efficiency and effectiveness. Project cycles are smaller, which allows for more rapid development and easier adjustments.
- **Lean** This style is similar to Agile but places more emphasis on efficiency and reducing waste. It is called "eliminating waste." When opposed to Agile or Waterfall initiatives, Lean projects usually have a shorter timeframe and fewer features.
- **Iterative**: This model goes through several iterations of development, each of which yields a partially finished product. Additional project needs are handled in subsequent cycles until the final product is produced.
- **Spiral**: Highlighting particular product risk patterns, the Spiral model gives the development team the ability to choose which additional process models to use.
- **V-Shaped:** This structure has a V-shaped cycle of concurrent validation and verification operations, with a testing phase for every development phase.

2.2 Literature Review

The final selection of a Software Development Model is critical in the development of computer software because it affects system performance, reliability, and stability. The choosing of an acceptable model is key to the product's success. Several software development models have been developed to meet the industry's changing needs, each adapted for a different type of software system.

Plan-driven models, such as the V-Model, are more commonly employed in the field of medical software than iterative approaches, such as agile development. The Association for the Advancement of Medical Instrumentation (AAMI) has published a study outlining how medical software companies can incorporate agile techniques while complying with FDA restrictions. This study also connects the principles of Agile to the development phases outlined by IEC 62304 Standard. Furthermore, G. Regan et al. emphasize the relevance of traceability in certifying software as "safe," and suggest its incorporation into the SDLC of medical devices. Their study compares various standards used in medical software design and assesses the consistency of two medical software designs using the MedTrace Framework. The findings highlight the need to include traceability in the software design stage to ensure secure and dependable software.

A 2013 comparative study examined software design methodologies employed in the the Software Development Life Cycle as published in a variety of scholarly papers. According to the research, including Agile techniques within the Software Development Life Cycle (SDLC) can help improve healthcare software development. As a result, to effectively tackle the requirements and layout for healthcare programs, an optimized SDLC combining plan-driven and Agile methodologies was presented.

In 2017, a hybrid evaluation strategy for healthcare software design enterprises was developed, incorporating agile methodologies to enhance their design abilities while using MDevSPICE as a starting point. Four medical businesses investigated this idea, which revealed a number of shortcomings in the development of complex healthcare supplies, such as insufficient tracking, limited consumer engagement, and security issues. The KATA standards was introduced to enhance the method of evaluation and address these issues, resulting in an extra reliable structure.

Another research project expanded on the above research by contrasting the MDevSPICE framework to agile techniques like Scrum, Extreme Programming and Dynamic System Programming. The findings demonstrated an important connection among these agile techniques and MDevSPICE, especially during the development and testing stages.

A 2019 study examined the suitability of different agile methodologies for healthcare software creation with regard to of compliance with regulations and standardization. The present research examined the Scrum methodology, Extreme Programming (EPA), and Feature-Driven Production (FDD) to identify weaknesses in agile methodologies that could create disruptions in the design and requirements stages because of repetitive approach to software development. The research revealed that Extreme Programming fails to meet regulation requirements because of inadequate advance requirement manufacturing, documentation, and tracking. In a comparable manner Scrum lacked adequate paperwork, while FDD constituted the agile method which came closest to meeting the established concepts and standards.

The study demonstrates that agile development methodologies are used in many different kinds of methods to create healthcare software. But fundamental components like traceability, upfront planning, and fixed demand management are frequently absent from pure Agile approaches. Furthermore, not all medical device production and testing standards fully adhere to Agile concepts. In addition, extensive testing of medical equipment designs is necessary to guarantee stability and improve reliability.

In this work, we introduce an adapted Agile-V model and address the SDLC methodology that was applied to the modeling of a therapeutic device, as well as the performance-enhancing test cases that were carried out. The value of the suggested paradigm is found in its two-pronged approach, which combines a systems engineering viewpoint with Agile and plan-driven windows for static and dynamic needs that are relevant to every subsystem.

2.3 Research Gap

1. **Integrating Systems Engineering into Medical Device Software Development**

   Gap: Current models, including the Agile-V model, do not include a comprehensive systems engineering approach that tackles both hardware and software needs.

   Need of Research: Develop a methodology for effectively incorporating systems engineering concepts into healthcare equipment SDLC.

2. **Software Classification Gap**

Gap: Current methodologies do not differentiate among Software as a Healthcare Device (SaHD) with program which enables healthcare equipment, disregarding the limitations of regulation.

Need of Research: Create unique development techniques that follow the particular regulatory requirements for various kinds of healthcare applications.

3. **Architectural Integrity in Agile Gap in development**

   Gap: Using agile system design might undermine hazard, particularly for complex healthcare equipment.

   Need of Research: Examine the benefits and drawbacks of plan-driven development of architecture using an agile methodology to ensure reliability and compatibility while remaining flexible.

4. **Agile methodologies and Traceability**

   Gap: Agile methodologies usually lack tracking tools, which are necessary for compliance with regulations for healthcare systems.

   Need of Research: Investigate approaches for improving tracing within agile methodology for the purpose to effectively satisfy the demands of regulators.

4. **Customer Participation in Plan-Driven Systems**

   Gap: Conventional plan-driven designs, including the one called the V-Model, frequently fail to incorporate consumer input, resulting in a mismatch between user desires and the requirements of regulators.

   Need of Research: Develop strategies to encourage stakeholder interaction and evaluation throughout the plan-driven design stages.

5. **Balancing Flexibility and Paperwork**

   Gap: Agile techniques prioritize adaptability over documentation, while regulatory agencies require extensive paperwork for healthcare equipment.

   Need of Research: Determine how to achieve a balance between the demand for agility and constant enhancement and the need for thorough documentation.

6. **Test Processes for Healthcare Software**

Gap: Traditional strategies for SDLC may not include medical-specific tests, thereby limiting safety and dependability.

Need for Research: Improve testing frameworks within the Software Development Life Cycle (SDLC) that fulfill the specialized testing specifications for healthcare applications in order to meet stringent safety and reliability requirements.

7. **Continuous tracking and adaptability**

   Gap: Existing approaches to development do not fully examine how to take advantage of continuous evaluation and adaptive aspects of healthcare software.

   Need of Research: Investigate how to effortlessly incorporate continuous tracking and flexibility into the design procedure in as to enhance the results for patients.

8. **Sustainability of Agile Techniques in Complicated Healthcare Equipment**

   Gap: The ability to scale might be difficult for agile methodologies when dealing with complex healthcare equipment that must fulfill stringent regulatory requirements. **Need of Research:** Look into whether agile techniques can be expanded successfully while getting rules-compliant and handling complication.

9. **Compliance with Regulations for Mixed Models**

   Gap: The adoption of mixed methodologies in the creation of healthcare devices that combine Agile and plan-driven approaches is still not well described in the context of compliance with regulations.

   Need of Research: Conducting an empirical investigation to confirm the regulatory compliance and efficacy of mixed devices, especially the EAV model, in addressing healthcare equipment rules.

CHAPTER 3

3.1 Proposed Methodology

The latest developments in the relationship between health technology and science have transformed the healthcare industry. Among these advancements, therapy medical equipment have come out as crucial tools for providing personalized therapy and enhancing the health of patients. The employment of modern software solutions in these devices has provided innovative choices for precision, adaptability, and monitoring in real time. The intersection with health care equipment and software development creates an ideal environment for study and development.

One major concern with software products, particularly in the healthcare sector, is safety. Mistakes in healthcare software can result in incorrect diagnoses, which can be lethal. As a result, strict standards exist to ensure the safety and reliability of such systems. Notable laws comprise the United States Code of Federal Regulations (CFR) and the European Medical Device Directives. These regulations define specific requirements for the development, production, testing, possibly and functioning of healthcare equipment. The goal of this study is to create a software model that helps programmers adhere to these criteria while being flexible enough to incorporate modifications fast.

The choice of software development platform has an enormous effect on the program's equilibrium, effectiveness, and overall effectiveness. This is particularly relevant in healthcare computer software, which requires stability and accessibility due to the potential hazards connected with software breakdowns. Conventional plan-driven approaches, like the Waterfall approach, provide plenty of documentation and tracking, allowing regulatory bodies to grant approval for projects. However, the rigid structure of these models makes them inappropriate for the changing and rapid characteristics of the healthcare field.

Regulatory agencies do not mandate the utilization of any particular software production approach, allowing creators to choose a suitable approach. Based to a research undertaken by an Irish healthcare device manufacturing company, 50% of healthcare equipment were built using the V-Model, 25% utilizing agile standards, and the remaining 25% employing iterative design methodologies such as Waterfall.

The the V-Model scenario, a plan-driven paradigm, is recognized in the healthcare application industry for its reliability and traceability. However, its rigid structure makes it difficult to adjust to changing needs. To address this, McHugh and others proposed the Agility-V (AV) Model, which combines Agile with plan-driven methodologies. This paradigm expands on the V-paradigm by using agile principles to improve scalability. Every phase of the design process are checked to ensure that they can be recursive or has to be accomplished in one attempt. While the AV model provides significant advantages, our research revealed many limitations. Table 3.1 shows these shortcomings and our proposed improvements to fix them.

Table 3-1: Limits in McHugh's Agile-V Approach for Healthcare Equipment Design

S.	**Agile V-Model Limitations**
1	The absence of a systems engineering strategy that addresses software/hardware specifications.
2	Ensure consistent management of all software products, whether classified as SaMD or supporting software.
3	Agile architecture can undermine safety and risk management in complicated medical devices.

By incorporating these additions, the EAV model not only maintains the agile components that allow for flexibility and adaptation, but it also imposes tight protocols for specific areas of the software development process to assure system stability and regulatory compliance. For instance, the audiovisual (AV) model permitted rapid modifications to the system's infrastructure that may have an impact on protection, however the EAV model mandates that architecture be created in an ordered, plan-driven manner.

This approach guarantees any structural changes are adequately managed and reported, preserving the health care device's validity and security.

Our medical equipment was developed using an EAV paradigm, which has been recommended as a universal foundation.

The Enhanced Agile-V (EAV) model includes the following major enhancements:

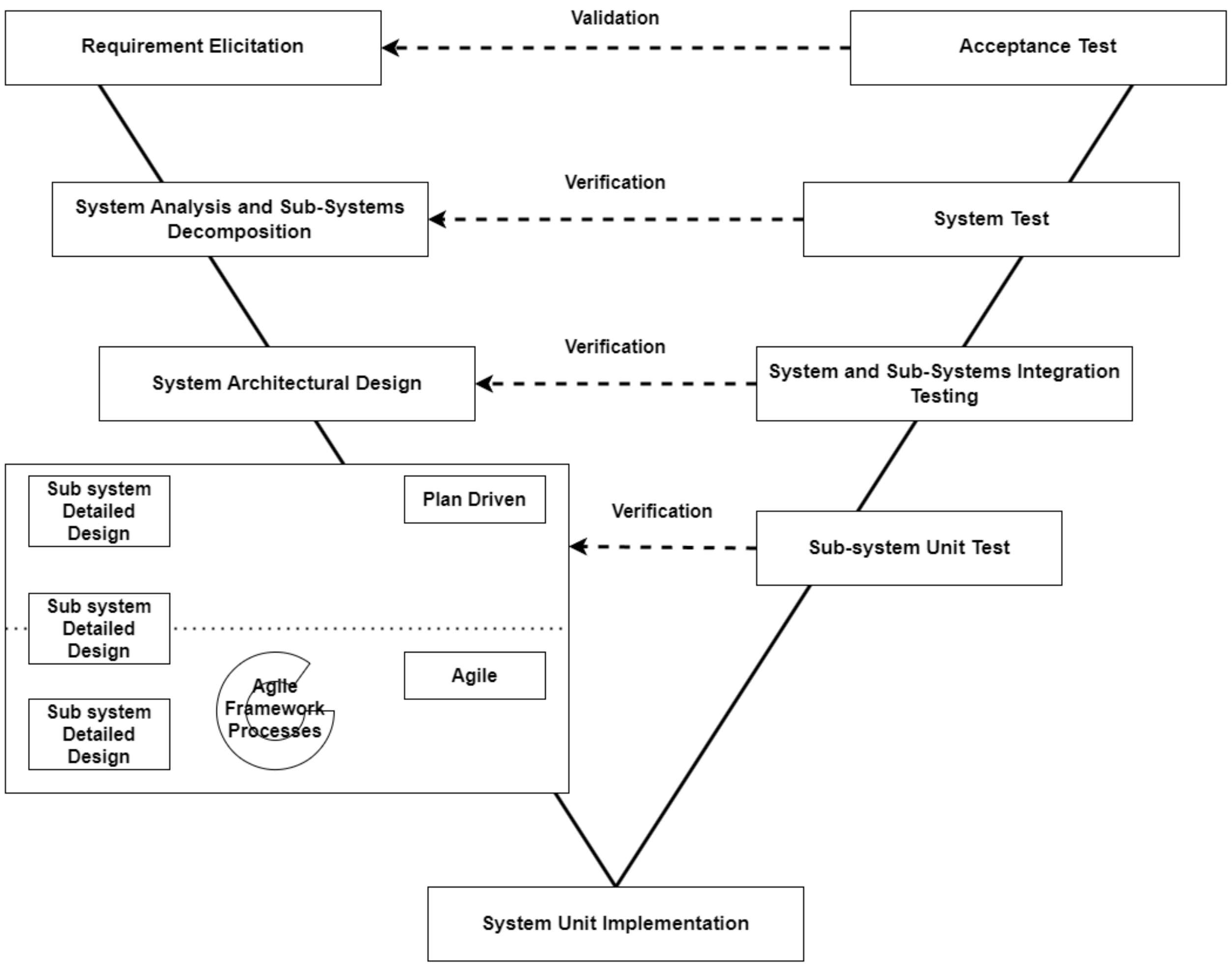

Validation
Requirement Elicitation
Acceptance Test
Verification
System Analysis and Sub-Systems Decomposition
System Test
Verification
System Architectural Design
System and Sub-Systems Integration Testing
Sub system Detailed Design
Plan Driven
Verification
Sub-system Unit Test
Sub system Detailed Design
Agile
Agile Framework Processes
Sub system Detailed Design
System Unit Implementation

Figure 3.1: Enhanced AV-Model blends agile approaches with the conventional plan-driven design method.

- Systems Engineering Integration: The methodology takes a systems engineering method to successfully solve both software/hardware issues. This is accomplished by breaking apart the system into discrete subsystems during the system analysis step.
- Subsystem Definition in High-Level Design: Each subsystem is specified at the high level of the design or architecture phase, ensuring clarity and structure throughout the design process.
- Classification of Requirement: Requirements are divided into fixed and variable sets. Each requirement is given a unique identification that indicates its type. These needs are managed via agile and plan-driven windows. For instance, critical to safety needs are handled during

plan-driven stages, while different needs can be addressed in agile window frames. This enables portions or entire subsystems to be controlled effectively according to their criticality.

- Maintaining V-Model Framework for Compliance: The core framework of the V-Model is maintained to meet regulatory body requirements, although the variable set of requirements is managed using agile approaches.
- Re-approval from regulatory agencies is only required for changes that affect safety or risk-related standards and entail changes to the system's architectural layout.

3.2 MDevSPICE and EAV-Model

To verify that the Enhanced Agile-V (EAV) model meets regulatory requirements, we linked it with the MDevSPICE® framework. MDevSPICE® contains requirements from a number of key norms, such as the FDA's IEC 62304, ISO/IEC 12207, ISO 14971, and ISO 13485. IEC 62304, is intended to guarantee the safety and efficacy of medical equipment by establishing specific procedures, activities, and tasks that ensure the device performs as intended while avoiding unacceptable hazards. MDevSPICE® does not impose a particular process model or method to device development. Instead, it provides as an assessment framework for medical device development, assisting firms in complying with IEC 62304 requirements. It is used by software suppliers to prepare for regulatory audits and by large medical device makers to assess potential software partners. Essentially, MDevSPICE® is a comprehensive set of regulatory requirements and processes for ensuring software satisfies market standards.

The MDevSPICE® architecture consists of three parts: the MDevSPICE® Process Reference Model, the MDevSPICE® Process Assessment Model, and the MDevSPICE® methods for evaluation. The figures 4 and 5 show how these representations are generated from IEC 62304 and describe the numerous processes in MDevSPICE®, such as healthcare systems life cycle methods, software development processes, and maintenance activities.
To ensure compliance, we matched our EAV model to the processes outlined in MDevSPICE®'s PRM. This mapping (described in Table 4.2) reveals that the EAV model completely complies to the MDevSPICE® PRM processes. As a result, the EAV model may be relied on as a process model for manufacturing medical devices while maintaining quality and regulatory compliance.

Table 4.2 describes the steps taken inside each MDevSPICE® procedure, as specified in the software development standard of information and the architecture itself. Every step is associated with its relevant component in the EAV paradigm. Because the EAV model includes all PRM processes, firms that use it can analyze specific base principles and work outputs utilizing the MDevSPICE® PAM. The PAM consists of basic principles and instructive procedures that direct the method of operation and help accomplish process results. These core actions create or use work products that are required for achieving specified process objectives.

3.2.1 MDevSPICE Processes of Medical Device System Life Cycle Processes

1. Project Planning

Project planning has several elements, including defining the project scope, analyzing feasibility, identifying required expertise, monitoring project interfaces, allocating resources and responsibilities, executing the plan for the project, and designing the project's life cycle model. This planning step occurs before and during the EAV model's initial elicitation phase and is led by the project or product owner. Given its importance in systems engineering, this phase entails thorough design for the complete product, including software, hardware, and integration components.

2. Risk Management

Risk management comprises identifying potential hazards associated with the product, developing mitigation solutions, and continuously reviewing effectiveness. It includes operational, marketing, and technology risk elements. In the EAV approach, risk management is an inherent component of the project planning phase, with a complete risk plan developed before the project's beginning to proactively handle anticipated obstacles and ensure smooth development.

3. Stakeholders Requirements Definition

The definition process includes identifying stakeholders, collecting requirements, defining restrictions, specifying user interactions, identifying essential demands, evaluating and agreeing on needs, and documenting them. Because the needs and the good itself are initially unclear, the first step is to comprehend the system. EAV employs Requirements elicitation methodologies and approaches during the Requirements elicitation and analysis stages. Furthermore, each subsystem needs to be assigned a unique identity that indicates whether it falls within a fixed or volatile timeframe. This detailed documentation improves configuration and change management over time.

4. Project Assessment and Control

Project Assessment and Control includes accumulating project activities, monitoring project characteristics and interfaces, reporting project progress, performing project reviews, addressing deviations, and carrying out project assessment and control. Project assessment and control are essential components of the EAV model, and are necessary not only during requirements creation and planning but also when modifications occur. Every outcome adds to the project's experience, allowing for approved adjustments based on that experience using agile methodologies.

5. System Requirements Analysis

It entails defining system requirements, improving project solutions, assessing system requirements, evaluating and updating system requirements, and conveying system requirements. System requirements analysis takes place during the EAV model's phases of requirements elicitation, system analysis, and subsystem decomposition. It includes key stakeholders such as requirements engineers, product owners, hardware designers, and software engineers. In addition, product owners and scrum masters undertake this phase on a daily basis to accommodate changing requirements.

6. System Architectural Design

It entails creating a theoretical representation of a the system's actions, structure, and additional features, as well as explaining the different parts of a healthcare device and their interactions via interfaces. Isolated software contains both the application itself and the system architecture. The objective is to recognize and handle health-related risks, speed up the inspection and clearance process, keep system designs brief, precise, and standard-compliant, and identify errors early on to minimize costly rework and delays. It defines both the system and software architectures. According to the suggested framework, the entire structure of the software and hardware must be decided before the actual comprehensive design phase begins, because any changes in architecture incur considerable costs.

7. System Integration

Integrating systems is a vital component that includes a variety of system engineering and management activities. It covers constructional, procedure, user interface and enterprise aspects to guarantee that both software and hardware solutions work together seamlessly. Systems and subsystems integration testing ensures that the product is compatible with the existing architecture. Both software and the hardware go through thorough testing to ensure that they function as a single unit.

8. System Qualification Testing

It assures that the system architecture fulfills its criteria by a set margin over the expected operational circumstances. These qualification margins are determined by systems engineers in consultation with project executives and customers and then incorporated into specifications. Testing takes place at different tiers, involving validation against functionality and paperwork inside the EAV framework. Throughout acceptance testing, the entire system is evaluated as a fully connected entity using already established standards, with comprehensive client oversight and agreement.

9. Software Installation

It identifying the necessary tasks, configuring hardware/software/network, assessing the physical environment's requirements, and ordering equipment, software, or services. Tasks are delegated tasks, and a comprehensive work schedule is established. The EAV model precisely outlines deployment requirements throughout the product design process, which are then contained within a thorough design model that includes software and hardware deployment details.

10. Software Acceptance Support

Software acceptance support is designed differently for agile and fixed sets of requirements.

Initially planned, extensive acceptance testing by customers and stakeholders takes place during the acceptance testing phase, when the system is ready for delivery. Later, for evolving requirements, agile support allows for adjustments and new requirements, ensuring system adaptability and evolution throughout time. Acceptance activities are repeated following the implementation of the modifications required.

3.2.2 MDevSPICE Processes of Medical Device Software Life Cycle Processes

1. Software Development Planning

This phase begins with a software engineering perspective, which includes purpose, scope, objectives, and constraints. Planning entails reviewing the architecture, determining required tasks, creating project documentation, allocating resources, and managing risks.Planning happens before and after the very first requirement gathering stage in the EAV framework, with agile preparation embracing the Scrum approach and handled mainly by the final item owner.

2. Software Requirements Analysis

This includes creating, prioritizing, assessing, and upgrading software specifications, as well as dealing with conflicts and communicating effectively. Participants in the EAV model include requirement specialists, consumers, and programmers. Product managers and scrum leaders help teams adapt to new requirements on a daily basis.

3. Software Architectural Design

This stage emphasizes the organization of system elements, as well as behavior and interaction. It aligns architectural choices with business objectives and selects architectural designs. In on EAV framework, both the system and software architectures are completely specified before comprehensive design, highlighting the relevance of architecture in avoiding expensive adjustments hereafter.

4. Software Detailed Design

Detailed requirements are created for coding and execution to ensure accurate implementation.

Figure 3.1 clearly shows the detailed design window in the EAV model. Initially, design takes a plan-driven approach before transitioning to agile approaches for new requirements. Software designers with technical skills play an important role in this phase, ensuring that system requirements are met.

5. Software Risk Management

Critical actions include risk identification, analysis, prioritization, and mitigation. Risk management in the EAV model begins with project planning, ensuring written risk plans are in place before project beginning, and addressing safety issues linked with medical.

6. Software Unit implementation and verification

Writing code, building databases, and carrying out the tasks required for design implementation are all part of this phase. To completely test each software unit, developers create test cases that include inputs, expected outputs, and assessment criteria. The test cases cover all facets of the complex design of the unit. Software that is customized to a software unit must be tested by developers.

There is a specific subsystem unit testing phase in the EAV model (see Figure 3.1). Subsystem units are implemented, and then each subsystem level is verified and tested to make sure it works correctly. This testing is done by software engineers. Developers create and run manual or automated test cases for both plan-driven and agile requirements. This stage involves extensive code testing to guarantee that every subsystem and its parts operate as intended.

7. Software Integration and Testing

Integration tests assess the interaction of several components or services. In the EAV framework (Figure 3.1), integration testing takes place during the system and subsystem integration phase to ensure that hardware and software are in sync with the intended architecture.

8. Software System Testing

This phase evaluates several portions, modules, or components of a software program concurrently.

In EAV model (Figure 3.1), system testing confirms that software requirements specifications are followed, guaranteeing that the system performs as intended.

3.2.3 MDevSPICE Processes of Medical Device Support Processes

1. Configuration Management(CM)

CM is the process of overseeing computer hardware and software systems to make sure they continue to work as intended over time. Planning, identification, change control, accounting for configuration state, audits, and reviews are all included. The EAV model integrates an agile strategy and plan for newly developed needs to efficiently handle configuration management and change management. This guarantees that requirements can be changed without difficulty, preserving the integrity and functionality of the system.

2. Software Release

Software Release: After significant improvements and problem corrections, a software release exposes a polished product to end users, marking the pinnacle of the software development process. The release process includes several critical stages, including preparing the release's development to ensure every aspect have been addressed, creating it to comply with construction and design guidelines, conducting accurate testing for user acceptance to confirm capabilities and effectiveness, preparing the release by ensuring all installation specifications are met, and ultimately deploying the throw to the production surroundings. The mechanism is carefully managed throughout the EAV concept. Before the project begins, basic planning is completed to provide a clear way forward. The building phase adheres to the stated architecture and design ideas, which ensures reliability and consistency. Acceptance testing occurs after development to ensure that the application meets all user requirements and standards. The last version is influenced by deployment methods established throughout the design stage, ensuring a smooth transition from development to operation.

3. *Software Problem Resolution*

Software creation and solving problems necessitates an organized method to identifying and resolving issues. The initial phase in this technique is to correctly recognize the problem and its symptoms. Following the identification and isolation of the underlying problem, the following step is to test various solutions. Once an approach is discovered, the problem is remedied, and the treatment is tested to ensure that it has been fully addressed. The EAV paradigm excels at dealing with software difficulties because it incorporates agile approaches. Agile techniques encourage progressive and repeated solving issues, which allows for quicker solutions. When an issue arises, the group develops the solution, examines the source of the problem, and makes any necessary changes to the plan and execution. These processes are helped by the Scrum methodology, an essential component to the EAV paradigm that operates numerous iteration cycles and ensures that changes have adequate documentation and issues are resolved swiftly.

4. *Software Change Management*

Change management is an essential part of constructing software, which includes recognizing and analyzing changes, understanding their ramifications, and planning their implementation. The EAV model blends agile and planned methodologies to easily integrate managing changes. Changes are initially identified and classified based on their nature and importance. A thorough review is conducted to understand the effects of the change on the framework in its entirety. A complete strategy is developed to guide the execution process, and choices on the implementation of the modification are made collaboratively. Because agile approaches are incorporated with the EAV approach, it is feasible to be adaptable and responsive, ensuring that changes are efficiently managed. This technique ensures that if demands and requirements for stakeholders alter, with the system gets used.to meet those needs.

5. Software Maintenance

Software maintenance covers an extensive variety of duties aimed at ensuring that the program continues to function effectively and adjusts to changing requirements. This includes planning system updates, organizing follow-up updates, managing change demands, conducting impact assessments, and implementing modifications into action. The EAV model combines software maintenance with other support operations, including software launch, management of changes, management of configurations, and solving problems. A complete impact study is undertaken to better understand the effects of each modification request on the system. Planning system releases ensures that updates are seamlessly integrated into the existing system. After the changes have been carefully applied, an updated copy of the platform is released. This iterative technique ensures that the software remains dependable, up to date, and capable of meeting the needs of users all through time. Refer to points 1–4 of the assistance methods, as maintenance of software involves handling changes, management of configurations, fixing problems, and system deployment. The EAV algorithm's agile procedures and methodical maintenance schedule ensure that the application can adapt effectively to novel possibilities and difficulties.

CHAPTER 4

CASE STUDY: PLAN AND DESIGN A MEDICAL DEVICE FOR WAVE THERAPEUTIC NEUROTRANSMISSION COGNITIVE THERAPY

The requirements, design, development, and testing procedures for the wave therapeutic medical device and its associated applications are covered in detail in this chapter.

4.1 Introduction

Specifications for the hardware device and related software were carefully recorded after significant discussions with developers, researchers, physicians, and patients, among other important parties. The IEEE template for software requirements specifications and the Software Engineering Book of Knowledge (SWBOK) served as sources of inspiration and modifications for the requirements, design, and evaluation of the documentation's structure and format. The specifications of both hardware device and supporting software are documented after detailed discussion with the relevant stakeholders i.e. doctor, patient, developers, and researchers.

The format of requirements, design and testing is inspired and adapted from SWBOK (Software Engineering Book of Knowledge) and IEEE template for software requirements specifications.

Table 4-1: Chapter conventions: Terms with important definitions

Term	**Definition**
DB (database)	A group of data that the system makes use of.
User	Any patient with legitimate credentials uses the system.
Android	A mobile operating system that Google Inc. created.
SRS (Software Requirements Specification)	A paper like this one, fully outlines every feature of a suggested system and also the limitations it must work within.
Vibro tactile gadget / Wave therapeutic device	A medical gadget with vibration wave transmission developed by RISETech and Biomisa Lab.
NCT (Neurotransmission Cognitive Therapy)	A treatment that describes how people with neurological problems can benefit from vibration waves as they help revive dead brain cells.

UC (Use case)	A single, fundamental system feature.
NFR(Non-Functional)	Describes the system's functioning or quality features.
FR (Functional Requirements)	Describes a certain system characteristic or feature.
MD (Medical Device)	A tool with medicinal applications.
UI (User Interface)	The point of access where users communicate with the system.
TC (Test Case)	An orderly process designed to confirm the system's operation.

4.2 Chapter Conventions

This chapter uses the conventions mentioned in Table 4.1

4.3 Project Perspective

The Wave Therapeutic Healthcare Equipment is available in two separate variants:

- Handheld/Portable (for domestic use)
- Clinic (for clinics or healthcare facilities)

To meet the unique requirements of both of these versions, several major product perspectives have been identified:

- Both the portable and clinic versions must have hardware that can send vibration vibrations.
- Every version ought to possess an interface that is simple to use adapted to its specific context. The portable variant ought to have a clear and user-friendly interface for domestic use.
- The health care variant should include a structure that allows healthcare practitioners to use it easily in clinical environments.
- Both of these variants have software assistance to improve functionality.
    - The clinic edition should have built-in software to permit smooth operation in healthcare facilities, designed for usage by Physicians and therapist.
    - An application for mobile devices needs to be compatible with the standalone variant, making it easier to utilize at homes.
- A web-based application will act as a central hub for gathering and analyzing information, available to scientists and physicians. This information can then be used for an information-driven approach to therapy suggestions.

Each variation and aspect described above has particular needs and demands that are elaborated on in the next sections of this chapter utilizing known software development approaches and methods.

4.4 Overall Description

4.4.1 Product Description

The suggested surgical instrument for NCT will be delivered to specific portions of the body of an individual using sensing motor points that correlate to core locations inside the brain. Figure 4.1 explains these concepts.

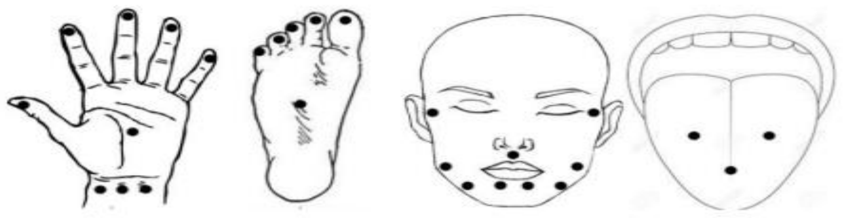


Figure 4.1: Depicts the indicated sense motor positions where the device will be attached. These sites transmit vibrating pulses that stimulate the neurons in the brain.

The tool consists of two major parts: a control board and an application tool. The metal headed transducer is intended for use when gently pressed to the designated sensory motor locations in order to transmit vibrating signals. The item comes in two versions: one for usage at home and one for clinical use. All of them come with software to make them easier to operate. In addition, an internet-based app is intended for research uses, gathering information from both of the versions. The data collected can then be utilized to provide computerized therapeutic suggestions based on the information gathered.

4.4.2 Product Features

The equipment being used includes two types

- Mobile
- Clinical

Both variants need to be able to transmit mechanically generated waves at the specified frequencies. The mobile and clinical variants of the wave therapy equipment should each have their own software with capabilities that allow users to control the device based on their location of choice and convenience. The suggested accompanying software platform for the mobile device consists of two versions:

- Test Controllers Mobile Software: A restricted feature Android application built primarily for testing the device's connection and working with the mobile software
- Fully Functioning Mobile Software: A robust smartphone application designed for use by patients and their families, with complete device capabilities

4.4.2.1 The Android Testing Controllers Application for Portable Devices

The Android app for the restricted capability test controllers allows patients to:

- Connect their device
- Adjust the attached device's frequency

4.4.2.2 Completely Functional Smartphone App for Handheld Device

The general features of the fully functional Android application will enable patients to:

- View therapy sessions
- Start or continue therapy
- Connect the device
- Edit their profile
- View daily therapy history
- View therapy statistics.

4.4.2.3 Embedded Software for clinic

The general features of the embedded software of clinic will enable the patients to:

- Start/continue therapy.
- View timer.
- Increase/Decrease the frequency of the vibrations.
- Touch supported screen.

4.4.2.4 Web Application as central data repository

The general features of the web application will enable doctors and researchers to:

- View statistics of devices in use by doctors.
- View statistics of devices in use by patients.
- View statistics in terms of age and gender separately.
- Has supported interface for therapy recommendation by doctors to the home users.
- Full data view support for researchers and doctors.

4.4.3 User Class and Characteristics

User is the one who will be using our system. Based on this definition we have categorized two types of users – doctor/therapist and patient/home users. Characteristics of each class are given below in detail.

1. Doctor/Therapist:
   a. Doctor/Therapist is the one who will use the device in clinics and their interaction is through embedded software.
   b. Doctors can view the details of their patients who are using the device at home through fully functional android applications.
   c. Doctor and therapist both have same roles for the systems and term can be used interchangeably.
2. Patient/ Home User:
   a. Patient/Home User is the one who will use the device at home and their interaction is through fully and limited functional android application.
   b. Patient and Home user both have same roles for the systems and term can be used interchangeably.

4.4.4 Software Development Life Cycle Approach of the Product

While regulatory organizations do not restrict the use of classic waterfall models for the creation of medical equipment, their clearance processes often demand documentation generated by these types of models. Ignoring the speed and volatility of criteria in the medical equipment development life cycle complicates managing these specifications, including tracking, instability, and verification. Using only agile strategies for the creation of medical goods can be troublesome since it ignores the organized documentation given by plan-driven software creation approaches. This lack of paperwork can lead to administrative rejection.

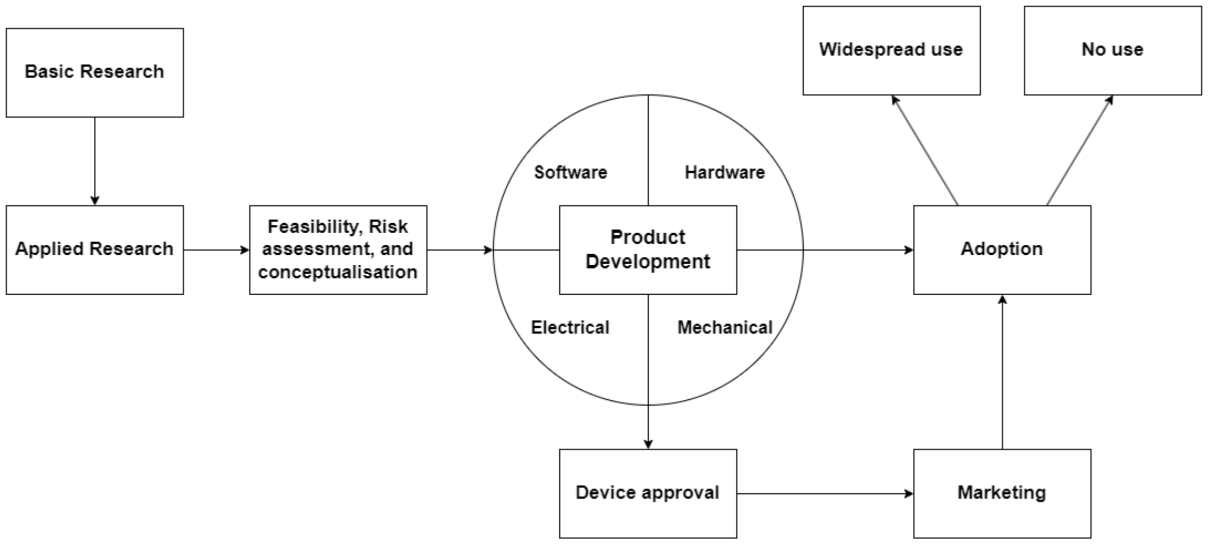


Figure 4.2: Figure 4.2: Medical Technology Life Cycle: The concept begins with basic research and evolves through applied studies. If the applied study receives funding, it is necessary to do an economic feasibility and risk evaluation in order to develop the end result. The creation of products entails merging several development factors, such as both hardware and software (either electrical or mechanical principles, depending on the demands). At last, after creation, the gadget must be approved by the appropriate nation-specific regulatory organization for entry into the market and acceptance.

Figure 4.2 depicts the normal lifespan of a medical device. To solve these problems, there is a rising preference for mixed approaches that combine the benefits each of plan-driven and agile strategies. Another example is the AV-Model [178], [187], which combines conventional V-model characteristics with agile methodologies.

We have suggested improvements to McHugh's AV model for the production of our therapeutically medical devices, as well as therapist medical equipment in particular. The following adjustments have been recommended to the model, and Figure 3.1 depicts the upgraded AV-model for the purpose of the medical system's development cycle.

- Incorporating a systems engineering technique into the modeling process to incorporate both software and hardware aspects by breaking them down into discrete subsystem throughout the entire system analysis stage.
- Creating such subsystems independently in the high-level design phase, also known as the system's architecture phase.
- Identify and handle the fixed and changeable sets of specifications for every subsystem independently. Two distinct techniques are identified: plan-driven and agile. These ways can be used to handle a subsystem, either completely or partially. As an example, if the software being used is a medical equipment, its dangerous nature demands that design be completed inside the plan-driven window. However, if the software supports usage of data or is merely for display reasons, it shall be managed in the agile window, which does not require substantial documentation.
- The core structure of the V-model will be maintained to meet regulatory standards while additionally promoting the agile model for a varied set of objectives.
- The regulatory body's permission will be needed solely for alterations which impact safety-sensitive requirements or entail alterations to the system's architectural design.

We chose the Enhanced AV Model (EAV Model) for our product development after completing a thorough study through brainstorming meetings with team members such as physicians, researchers, and engineers. The justification for choosing this model is its readiness to receive additional regulatory body permission, which necessitates paperwork generally supplied through conventional plan-driven systems such as the V-Model. However, the EAV Model removes the V-Model's rigidness and formality by introducing agile strategies to accept requirement variability.

The EAV Model complements a system engineering method for medical devices by combining the best aspects of both the AV Model and agile methodology. This approach enables the management of established and validated needs in a traditional plan-driven way. While continuously handling possibly changing needs, as seen in the agile approach. Utilizing the EAV Modeling, we extensively specified the requirements, allowing for successful management of requirements and the ability to adapt modifications as required.

4.4.5 Requirements Specification

Regarding requirement engineering, the item being considered can be completely divided into the categories that follow from both the development and product viewpoints:

- Creation of a vibratory wave healing therapy tool.
- Creation of a smartphone app allowing people at home to use the handheld version of the gadget.
- Created a test microcontroller for a smartphone app allowing people to debug a portable version at home.
- Create embedded programs with a simple interface for healthcare professionals to use in healthcare facilities.
- Create a web-based app that will act as a centralized information storage facility, providing medical practitioners and researchers with an accurate representation of patient data and therapy adjustments.

We first selected each subsystem after conducting comprehensive talks and research with the parties involved, as well as using the EAV model's system breakdown technique. We then divided the needs for each subsystem into functional and non-functional categories. Figure 4.3 depicts the subsystem breakup of our wave therapy apparatus.

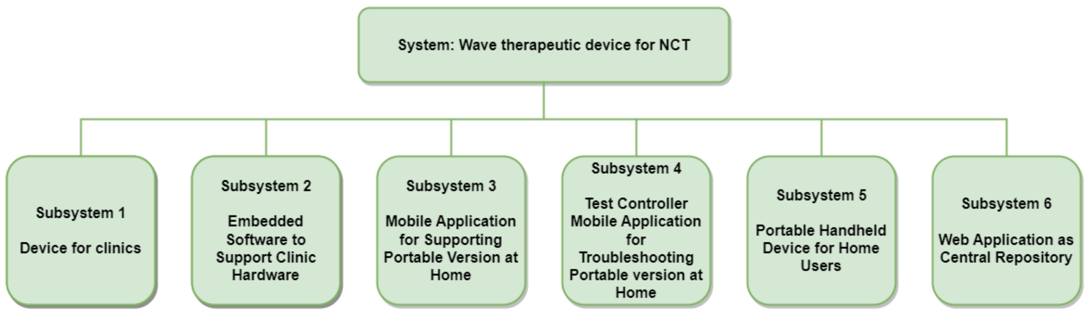

System: Wave therapeutic device for NCT
Subsystem 1
Device for clinics
Subsystem 2
Embedded Software to Support Clinic Hardware
Subsystem 3
Mobile Application for Supporting Portable Version at Home
Subsystem 4
Test Controller Mobile Application for Troubleshooting Portable version at Home
Subsystem 5
Portable Handheld Device for Home Users
Subsystem 6
Web Application as Central Repository

Figure 4.3: System breakdown into subsystems allows for distinct processing of criteria for each subsystem.

Following subsystem decomposition, the EAV-model categorizes constraints as functional or non-functional. The EAV-model requires an additional division of functional requirements into fixed and variable groups. A subsystem falls into one of two categories: fully plan-driven or fully agile.Fixed requirements, which comprise all safety-critical criteria, must be incorporated into the SRS report and assigned unique identifiers. Variable needs, such as design of interfaces and data viewpoints, might not be covered in the specification for SRS because they are subject to change following stakeholder meetings and lack official documentation.

For agile criteria that need to be recorded, fresh or evolving requirements are passed via scrum iterations. Variable needs, such as design of interfaces along with data viewpoints, might not appear in the SRS document because they are subject to change following meeting with stakeholders and lack official records. For agile criteria requiring to be written down, once new or evolving needs have been added, they go through scrum stages and validated after the following sprint's meeting, as illustrated in Figure 4.3. These approved needs, following execution, become fresh items or sub-items within the stated categories. Figure 4.4 depicts the distinct types of needs for each subsystem, providing easy traceability and independent administration for volatile and fixed set of criteria. Figure 4.5 defines the convention for numbering the necessary requirements for each subsystem.

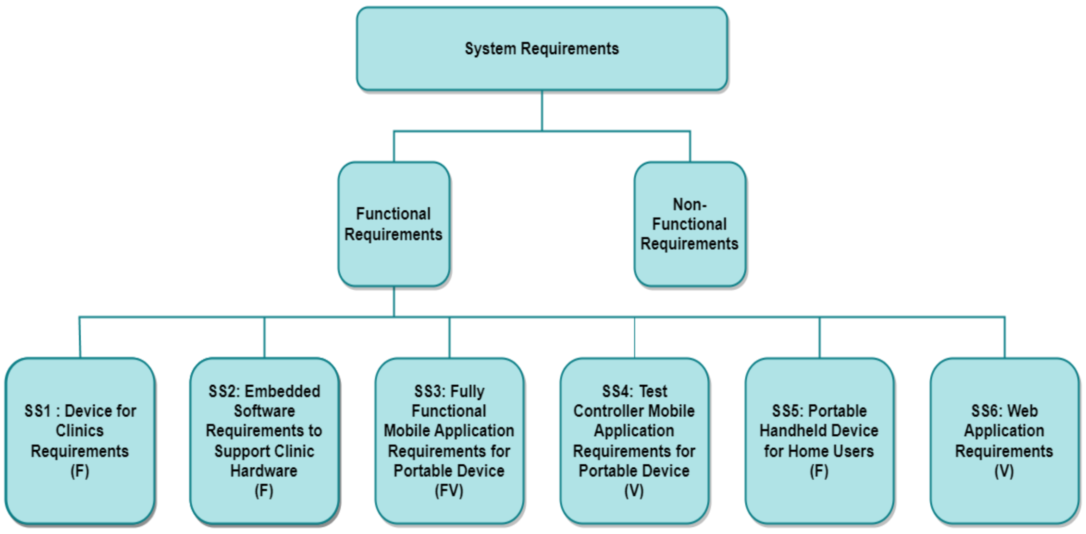

System Requirements
Functional Requirements
Non-Functional Requirements
SS1 : Device for Clinics Requirements (F)
SS2: Embedded Software Requirements to Support Clinic Hardware (F)
SS3: Fully Functional Mobile Application Requirements for Portable Device (FV)
SS4: Test Controller Mobile Application Requirements for Portable Device (V)
SS5: Portable Handheld Device for Home Users (F)
SS6: Web Application Requirements (V)

Figure 4.4: Depicts the classifications of requirements for every subsystem

FV: Subsystems with a mix of static and dynamic requirements, which are alternately handled using agile and plan-driven methodologies based on the type of the demand. V: Subsystems with dynamic requirements will use the agile approach. F: Subsystems with static requirements will use a plan-driven methodology.

4.4.5.1 Functional Requirements

Specifications for a fully functional smartphone application using subsystem 'SS3'.Figure 4.3 lists the specifications for the subsystem 'SS3' fully working mobile apps for handheld devices.

- "M" denotes SS3, "E" SS2, "H" symbolizes both SS1 and SS5, "W" SS6, and "T" SS4. Non-

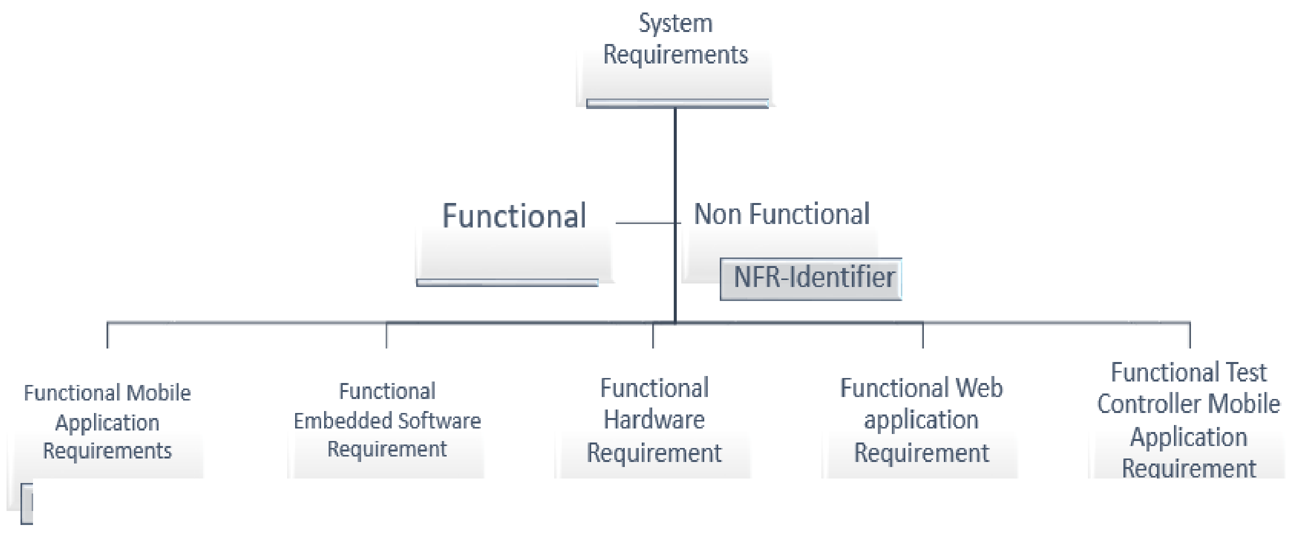

System Requirements
Functional
Non Functional
NFR-Identifier
Functional Mobile Application Requirements
Functional Embedded Software Requirement
Functional Hardware Requirement
Functional Web application Requirement
Functional Test Controller Mobile Application Requirement

Figure 4.5: The numbering conventions for the different categories of specifications for simpler maintenance

functional requirements (NFRs) are dealt with together for all systems (SS1-SS6) and lack a separate identity.

- Subgroups can have extra identifiers inserted at the end. As an example, consider FR-M-Identifier-identifier.
- The identifier "F," "V," or "F/V" is added at the end to indicate if the conditions have been fixed, variable, or a combination. "F" shows that the entire system has fixed needs, "V" shows that there are variable specifications across the subsystem, and "F/V" indicates that the specifications can be fixed or variable depending upon the characteristics of the specific demand.

Table 4-2: Functional specifications for Subsystem 'SS3' - Completely Functioning Android Smartphone Application for Domestic Use of Handheld Wave Therapy Equipment

Requirements Identity	Specifications for the Subsystems 'SS3' Fully Integrated Smartphone Application.
Number	Requirements for System Function.
FR-M-01-V	The handheld version of the equipment should come with an accessible smartphone application to help someone else operate it.
FR-M-02-V	FR-M-02-01: The procedure should start with the login by default FR-M-02-02: Following the login process, the session with therapy and account settings ought to be displayed.
FR-M-03-F	The app requires communication via Bluetooth in order to attach to the device.
FR-M-04-F	The software should have predetermined sessions for each sensory point, complete with graphics. When an individual needs to begin a session for a particular sensory point, they can determine the details using the image.
FR-M-05-V	The individual receiving treatment ought to have the option to stop or continue the therapeutic session.
FR-M-06-V	Following each session, the mobile application should show session data. The information provided ought to reflect both total duration and the total quantity of sensory points handled.
FR-M-07-V	The improvement of each session ought to be shown as a percentage
FR-M-08-V	The software should record the evolution of all session for future reference and assessment.
FR-M-09-V	The therapy's frequencies can be modified within 1 and 10 Hz, enabling individuals to choose any value inside this range of values.
FR-M-10-V	The user interface ought to be straightforward to use, with continuity across the program.
FR-M-11-V	FR-M-11-1-V: The user ought to be able to change their profile at any point. FR-M-11-V: When editing profiles, you should be able to modify the application's username and credentials.

4.4.5.1.1 Specifications for the 'SS4' Test Controller Smartphone App

Table 4-3: Technical specifications for Subsystem 'SS4' - Fully Practical Test Controller Smartphone App for Handheld Wave Therapeutic Equipment for Domestic Usage.

Requirements Identity	Specifications for Test Controller a smartphone app of Subsystem-level 'SS4'
Number	Requirements for System Function.
FR-T-01-V	It might be an excellent idea to have an independent application for verifying the hardware.
FR-T-02-V	This software will just verify connection to Bluetooth and different frequency channels.
FR-T-03-V	This application will also test the hardware's capabilities to stop and start again.
FR-T-04-V	Its objective is to make sure that the hardware works correctly.

4.4.5.1.2 Subsystem 'SS2' Embedded Software Requirements

Table 4.4 describes the specifications for the subsystem 'SS2', especially the software that is embedded, integrated in the clinic-designed variant of the equipment.

Table 4-4: Technical specifications for the subsystem 'SS2' - Integrated Software

Requirements Identity	Subsystem 'SS2' requirements for embedded software
Number	Requirements for System Function
FR-E-01-F	The physical device should be run by integrated software, enabling users to handle it not having to connect with complicated hardware parameters.
FR-E-02-F	The integrated program should include a panel for setting frequency between 1 to ten Hz.
FR-E-03-V	The program needs to be touch-sensitive and operate only with fingertips.
FR-E-04-V	Tapping your touchscreen will bring up the system's interface.

FR-E-05-V	The system ought to power off within two minutes of idleness.
FR-E-06-F	The switch located on the rear of the system should totally turn off the computer's components and the software.
FR-E-07-F	The monitor should include a clock to help consumers keep track of the length of the treatment session.
FR-E-08-F	The program should include choices to suspend and continue the therapeutic procedure.

4.4.5.1.3 Hardware specifications for subsystem 'SS1' and 'SS5'

Hardware needs are typically developed by hardware experts; nonetheless, a few particular essentials have been established defined following conversations with all stakeholders. These are included in Table 4.5.

Table 4-5: Specifications for Subsystem-level 'SS1' and 'SS5' Equipment for Either Handheld or Hospital Editions of the Wave Therapeutic Medical Device.

Requirements Identity	Subsystem ‘SS1’ and ‘SS5’ hardware Requirements
Number	Requirements for System Function.
FR-H-01-F	The equipment ought to be given in two versions: hospital and handheld for domestic use.
FR-H-02-F	The handheld variant should be controlled by a smartphone app that is linked via Bluetooth connectivity.
FR-H-03-F	The hospital or clinic variant ought to run by embedded programs developed directly for usage in healthcare facilities by experts. The prerequisites for the software being embedded are listed individually.
FR-H-04-F	The handheld variant should be fueled by batteries but also have an electrical supply.
FR-H-05-F	The hospital variant ought to be just charged by a power supply.
FR-H-06-F	The handheld variant should have an application device head for transferring waves and a device that is portable with accompanying components.

FR-H-07-F	The hospital version should have a control box with associated hardware and an application device head linked by a wire to convey sound waves.
FR-H-08-F	In the hospital version, attach the applicator to examine, which is subsequently linked to the application head.
FR-H-09-F	All equipment variants should be able to handle vibration waves that range between 1 and 10 Hz.
FR-H-10-F	The hospital's variant should feature an electrical button on the controller board that can turn the equipment on and off.
FR-H-11-F	The hospital version's controller board should include an interactive display to operate the instrument.

4.4.5.1.4 Subsystem-level 'SS6' Web-Based Applications. Requirements

Table 4.6 details the Subsystem-level 'SS6' Requirements, which refers to the web service that acts as an administrative hub for clinicians and researchers.

Table 4-6: The functional specifications for Subsystem-level 'SS6' web-based app.

Requirements Identity	Subsystem levels ‘SS1’ and ‘SS5’ hardware Requirements.
Number	Requirements for System Function.
FR-W-01-V	A web-based application ought to be created to act as one central repository for every record acquired from the treatment and therapeutic equipment.
FR-W-02-V	It must demonstrate data at different levels. FR-W-03-01-V: Data on every medical professional prescribing treatments. FR-W-03-02-V: Data on all patients receiving the course of treatment.
FR-W-03-V	Metrics should be provided on the main screen in a variety of graphs: FR-W-04-01-V: Pie chart showing the gender breakdown of therapeutic clients. FR-W-04-02-V: Pie chart showing the age breakdown of therapeutic clients. FR-W-04-03-V: Chart showing the regional geographic distribution of patients receiving the course of treatment.

	FR-W-04-04-V: A map depicting the regional geographic distribution of sufferers.
FR-W-04-V	The handheld variant ought to be battery-powered and also include an electrical supply.
FR-W-05-V	The main interface must instantly show the overall amount of patients, healthcare professionals, and continuing operations.
FR-W-06-V	FR-W-06-01-V: An administration dashboard should be provided for managing requests for entrance to the web-based program, which the administrator may get and accept.FR-W-06-02-V: The administrator dashboard ought to provide room for adding or removal of other illnesses served by the course of treatment.

4.4.5.2 Non-Functional Requirements

Nonfunctional requirements (NFRs) outline the system's key performance characteristics. **Satisfaction with these principles is necessary for developing a successful system or program. We** described the NFRs that were used for all of our subsystem levels in one table. Table 4.7 provides the system's operational and non-functional specifications.

Table 4-7: The functional specifications for the subsystems-level 'SS6' web-based application.

Requirements Identity	Software/Hardware Non-Functional Requirements
Number	System Non-Function Requirements
NFR-01	User-Friendly/Ease of Use NFR-01-01: The smartphone app needs to be user-friendly for everyone who can comprehend text and interact the display screen. NFR-01-02: Hospital hardware must be suited for table use, whereas residential hardware must be portable.
NFR-02	Fault Tolerance:To guarantee that both the hardware and the apps run well, the setup should generate suitable cautions and alarms.

NFR-03	Hardware Cleaning: Touch displays on equipment must be used for simple sterilization and cleaning as needed.
NFR-04	Mobility: The smartphone application should work with any model of Android phones.
NFR-05	Performance: The mobile application's reaction time shouldn't go past three seconds.
NFR-06	Maintainability: In the event that servicing is required, the infrastructure should allow for the substitution of parts. NFR-01-01: The vibration part of the application should be changeable. NFR-01-02: The energy source connector must be changeable.
NFR-07	Aesthetic and Consistency NFR-01-01: All both hardware and software should be designed with uniform color schemes: green/white for handheld devices, black/gray for devices. NFR-01-02: Alarms and cautions should be identified by color according to responsiveness, with yellow for cautionary and red for severe alert.

4.4.6 Use Case Diagram

Use cases are used to describe functional demands in a model once the requirements stage comes to an end. We depicted our subsystem in 4 separate scenarios for use, which are displayed in Figure 4.6, 4.7, 4.8, and 4.9, to make it easier to translate requirements during the design stage.

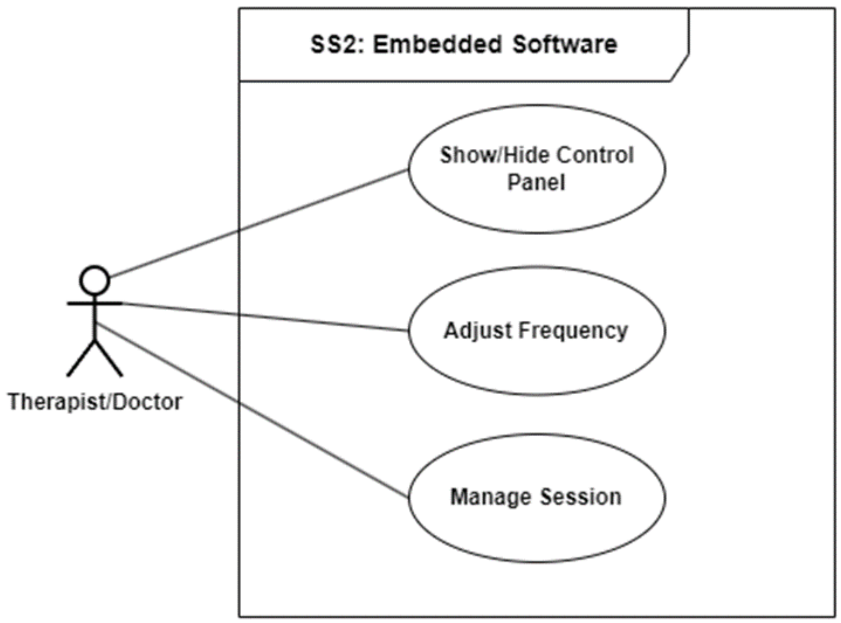


Figure 4.6: A Case Study Representation of the System-User Communication for the Hospital Variant of Integrated Program.

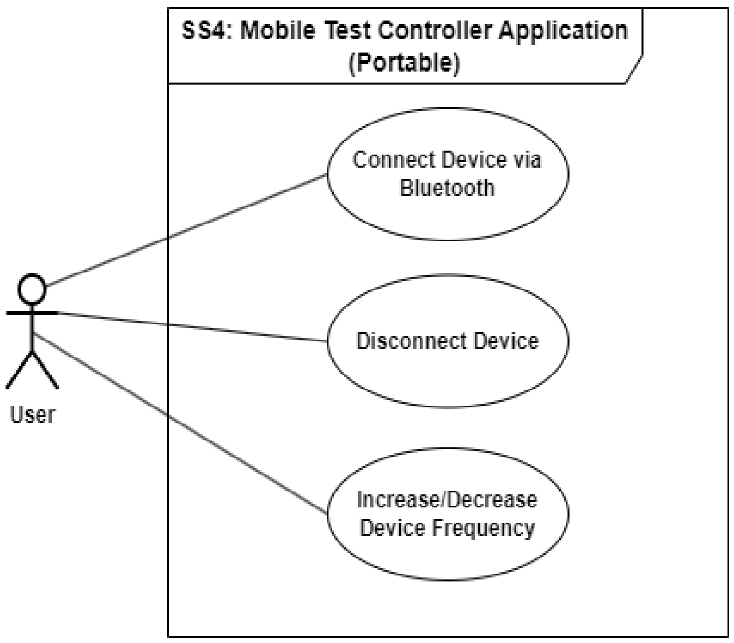

SS4: Mobile Test Controller Application (Portable)
Connect Device via Bluetooth
Disconnect Device
Increase/Decrease Device Frequency
User

Figure 4.7: Case Study Model of System-User Communication for the Handheld Variant of the Healthcare Device Test Controller Smartphone App

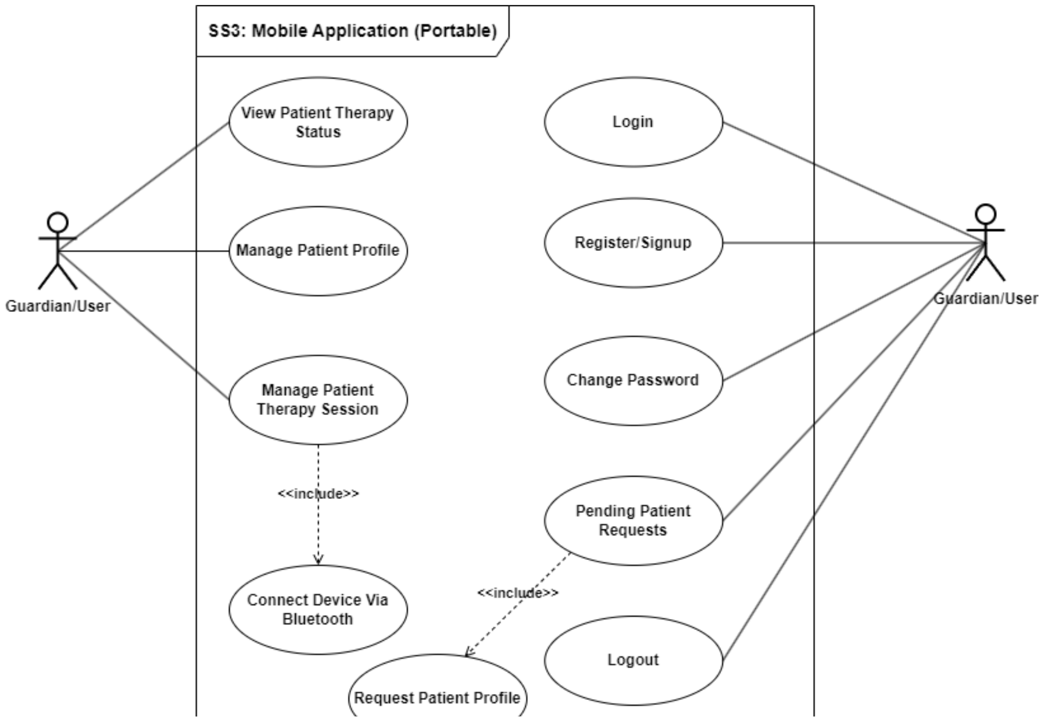


Figure 4.8: It depicts the relationship between the software and the user in the portable variant of the Healthcare Device Smartphone App.

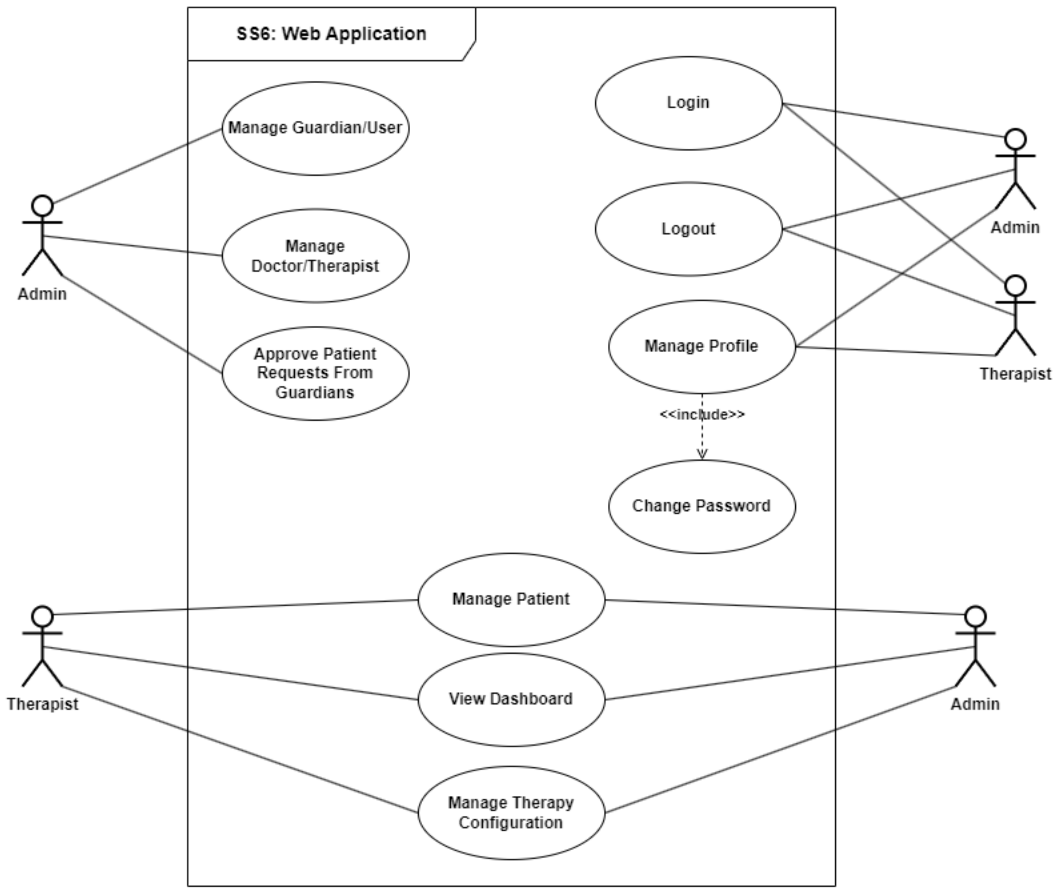


Figure 4.9: It depicts the relationship involving the system and the end user of a website application

4.4.6.1 Comprehensive explanations of use cases

A comprehensive overview of an ordinary use case is intended to thoroughly comprehend the objectives, characters that are involved, the order of events, and failure as well as achievement scenarios important to the use case. Tables 4.8–4.17 contain thorough explanations of difficult use cases for system SS3.

Table 4-8: Comprehensive Overview of Use Case "Login" to SS3.

Name of Use Case	Login
Identifier of Use Case	UC-SS3-01-V
Description of Use case	To utilize the program, the therapeutic client needs to be authorized by the software.
Scope	SS3: Portable, Perfectly Functioning Smartphone Applications
Primary Actors	Guardian/User
Precondition	The user must establish a registered account, and the app must be functioning with the login window displayed to them.
Normal Flow of Events	1. The user provides an acceptable identity. 2. The user provides a correct passcode. 3. The user clicks the login button.
Alternatives	1a: Username Invalid. 2a: Password Invalid.
Post Condition	The program checks the user's credentials before redirecting them to the primary program interface.

Table 4-9: A Comprehensive Overview of the Case for "Register/Signup" in SS3

Name of Use Case	Register/Signup
Identifier of Use Case	UC-SS3-02-V
Description of Use case	To use the software, the individual using it needs to have an operational account. A user may set up an account by completing the registration procedure.
Scope	SS3: Portable, Fully Functional Smartphone App
Primary Actors	Guardian/User
Precondition	The program operates and the user is shown the registration display.

Normal Flow of Events	1. The individual enters an accurate email address and passcode, then selects "Next Step." 2. The user enters a legitimate name, contact number, and place where they live before pressing "Confirm." 3. The program sets up the account, logs the user in, and displays the Processing request display with a few details.
Alternatives	1a. Email / password Invalid 2a. Name, phone-number, or city Invalid
Post Condition	Application validates the user and allows user to the Processing request display.

Table 4-10: Comprehensive Overview of the usage scenario "Request Patient Profile" in SS3

Name of Use Case	Patient Profile Request
Identifier of Use Case	UC-SS3-03-V
Description of Use case	To utilize the app's therapy user experience the person using it needs to have an operational account and be an accepted patient in the system's database. The user can obtain the patient profile through completing the inquiry for patient profile procedure.
Scope	SS3: Handheld complete functional smartphone app
Primary Actors	Guardian/User
Precondition	The application is started, and the user has signed in with no active patients. User is directed to the Pending Request Screen.
Normal Flow of Events	1. The individual clicks the "Add Patient" option. 2. The individual enters the patient's legitimate name, contact number, city of residence, address, gender, date of birth, and an approved prescription picture from the practitioner before clicking "Request Patient."

	3. The individual proceeds to the Pending Requests display that now shows a list item containing the patient's details and the current state of "pending."
Alternatives	2a. Invalid user name, contact number, city.
Post Condition	The software checks the patient details and leads the individual to the Pending Requests display, which shows an array of item with the desired patient's condition as "pending."

Table 4-11: Comprehensive overview of the Case for "Change Password" in SS3

Name of Use Case	Password Change
Identifier of Use Case	UC-SS3-04-V
Description of Use case	Individuals can update their login password.
Scope	SS3: Handheld completely functional smartphone app
Primary Actors	Guardian/User
Precondition	The program works properly and the user has logged in.
Normal Flow of Events	1. The individual presses the Settings button in the bottom navbar. 2. The individual selects the "Change Password" box. 3. The individual inputs their existing password, then the new password, retyping it to verify it, then clicking "Done."
Alternatives	a. If the password being used is wrong, the individual must be informed with the message "Your current password is incorrect” b. If the newly generated password and verification password fields do not match, the individual should see the message "Your new password and verification password did not match.” c. If a warning occurs when submitting the updated password form, show the following notification "Something went wrong.”
Post Condition	Changed Password Successfully

Table 4-12: Comprehensive Overview of the Case Study "Pending Patient Requests" in SS3

Name of Use Case	Requests of pending patient
Identifier of Use Case	UC-SS3-05-V
Description of Use case	Individual can see the patient requests current status.
Scope	SS3: Handheld completely functional smartphone app
Primary Actors	Guardian/User
Precondition	The program works properly and the user has logged in.
Normal Flow of Events	1. The individual taps the settings icon in the bottom navbar. 2. The individual selects "Pending Patient Requests." 3. The individual is provided with an array of patient requests awaiting clearance.
Alternatives	If there are not any pending inquiries, the app displays a blank screen with the message "Please enter your patient information to request a patient profile."
Post Condition	An individual can check the current state of patient requests.

Table 4-13: Comprehensive Overview of the Case Study "View Patient Therapy Status "in SS3

Name of Use Case	Status of View Patient Therapy
Identifier of Use Case	UC-SS3-06-V
Description of Use case	The program allows the user to view the quantity of prior therapy sessions that they have done.
Scope	SS3: Handheld completely functional mobile smartphone app
Primary Actors	Guardian/User
Precondition	The program works properly and the user has logged in.
Normal Flow of Events	Following login, the user is provided with the primary display, which displays latest activity statistical data: a. Total quantity of activities completed. b. A particularly recently task accomplished.

Post Condition	Stats are shown on the display as percentages and the state of completed actions.

Table 4-14: Comprehensive Overview of the Case Study “Manage Patient Therapy Session" in SS3

Name of Use Case	Patient Therapy Session Management
Identifier of Use Case	UC-SS3-07-V
Description	Users can initiate, stop and continue treatment for a patient, giving them authority over the doctor-prescribed therapy.
Scope	SS3: Handheld completely functional smartphone app
Primary Actors	Guardian/User
Precondition	The program works properly and the user has logged in.
Normal Flow of Events	1. The individual views the session page via the bottom navbar. 2. The individual is provided with the treatment points that were originally recommended to them. 3. The individual clicks "Begin Therapy." 4. The individual links the gadget using Bluetooth connectivity. 5. The individual begins the therapeutic session.
Alternatives	a. If the individual lacks any recommended treatment points, they will receive the notice "No therapy prescribed b. When an individual already began the treatment, a "Continue Therapy" option displays. c. When an individual previously finished the treatment, no button is displayed. d. Individuals may stop or continue their treatment session at any moment. e. If the wireless connection via Bluetooth gets disconnected, treatment ends and hidden control
Post Condition	The individual has effectively seen and completed the suggested treatment.

Table 4-15: Comprehensive Overview of the Case Study “Connect Device via Bluetooth “in SS3

Name of Use Case	Connect Device Via Bluetooth
Identifier of Use Case	UC-SS3-08-V
Description of Use case	The individual can attach the device and begin the therapeutic treatment.
Scope	SS3: Handheld completely functional smartphone app
Primary Actors	Guardian/User
Precondition	The program continues to run, the individual has logged in, and the current session page is visible.
Normal Flow of Events	1. The individual navigates to the Therapy page by pressing to the "Begin Therapy" option. 2. For connectivity to the equipment, press the Bluetooth option in the top right corner. 3. After pressing the button, the individual is provided with a bottom sheet that lists all possible devices in the vicinity. 4. The individual picks a device, and the program links to it.
Alternatives	a. If mobile devices Bluetooth connectivity is switched off, the individual gets asked to turn this on. b. If no equipment are discovered, the listing of accessible devices will be blank.
Post Condition	The individual has made an excellent connection to the physical device.

Table 4-16: Comprehensive Overview of the Case Study “Manage patient profile" in SS3

Name of Use Case	Patient profile Management
Identifier of Use Case	UC-SS3-09-V
Description of Use case	To add or edit patient account details.
Scope	SS3: Handheld completely functional smartphone app

Primary Actors	Guardian/User
Precondition	The program is functioning and the user has successfully signed in.
Normal Flow of Events	1. The individual presses the Profile button in the bottom navbar. 2. The individual enters the appropriate details and then presses "Update."
Alternatives	When a failure occurs during saving data from the profile, show the following notification "Something went wrong."
Post Condition	The details has been correctly up-to-date and the individual is routed to the home page.

Table 4-17: Comprehensive Overview of the Case Study "Logout "in SS3

Name of Use Case	Logging out
Identifier of Use Case	UC-SS3-10-V
Description of Use case	The individual can logout from the smartphone app.
Scope	SS3: Handheld completely functional smartphone app.
Primary Actors	Guardian/User
Precondition	The program is functioning and the user has successfully signed in.
Normal Flow of Events	1. The individual presses the Settings option under the bottom navbar. 2. The individual chooses "Logout."
Alternatives	If a failure happens while logging out, the program will display a notification saying "Something went wrong."
Post Condition	The individual gets forwarded to the application's welcome display.

4.5 Architecture and Design

4.5.1 Logical Architecture

Based to the EAV method, the more advanced design, or architecture, must be carefully developed because alterations at subsequent stages can be expensive and demand legislative permission. The system we use utilizes an architecture with layers, sometimes referred to as n-tier architecture. The following figure 4.10 depicts the logical architecture of our wave's therapy device for medicine, emphasizing the many subsystem.

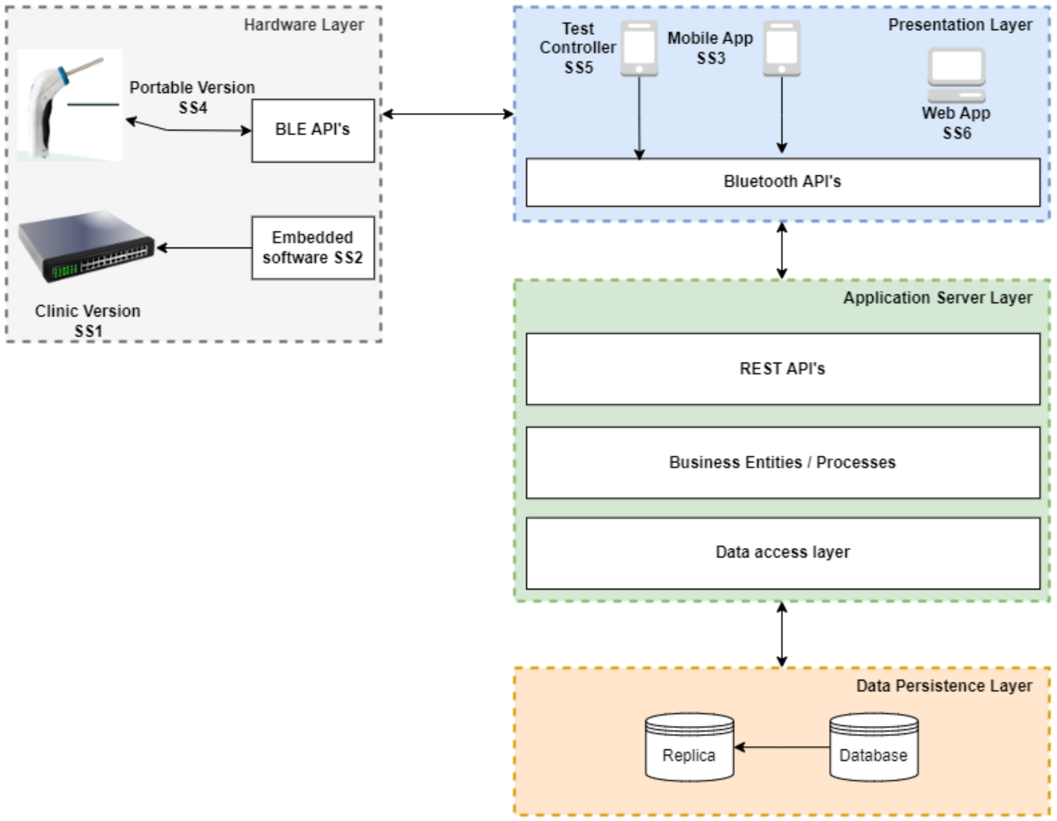


Figure 4.10: The device's logical design, which is an n-tier architecture with the physical technological layer indicated as the very last layer.

4.5.2 System Design

Following the effective gathering and documentation of specifications utilizing requirements techniques with the suggested EAV model, the following stage is to carefully convert those specifications into the design. Case studies serve as a link among requirements and design, and are developed following a thorough examination of the requirements. The use scenarios, in addition to the explanations and requirements paperwork, are used to inform the design of our system. We chose UML 2.0 and UML 2.5 for describing our system's complex design.

4.5.3 Class Diagram

A class diagram is an architectural representation that serves as the system's basic foundation block. It represents the system's basic architecture and displays the objects, their properties, and the capabilities that every item provides. Along with referring to the relevant objects, the class diagram depicts the connections among them. Figure 4.12 depicts the categories and relationships inside our subsystem SS3 smartphone app.

4.5.4 Sequence Diagram

The sequence diagram is an important aspect of the design phase because it reveals the particular features associated with complicated usage scenarios. It depicts how objects communicate as well as the order in which messages are passed. To handle the complex nature of particular scenarios, sequence diagrams depict the relationships of the individual using it and the system. Figures 4.11-4.15 depict comprehensive connections.

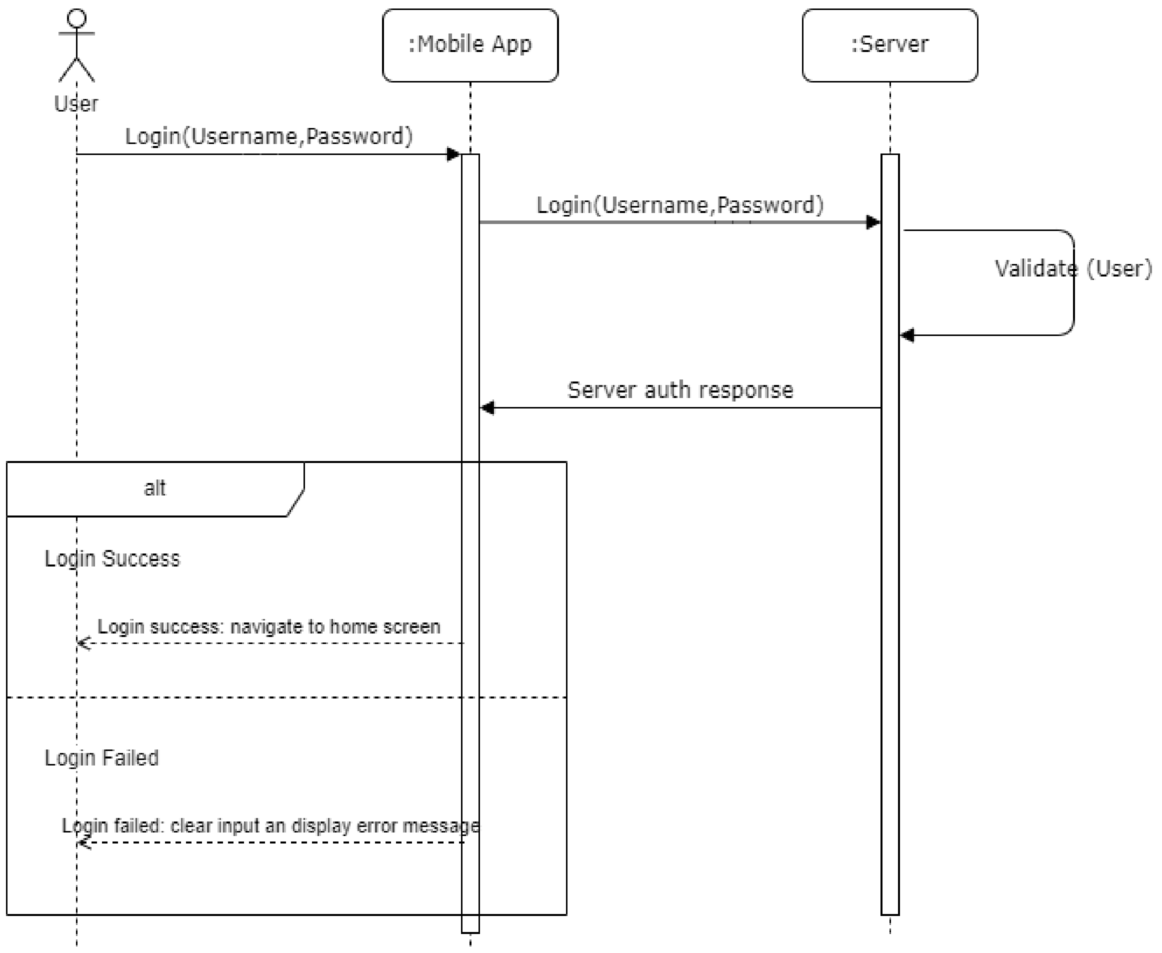
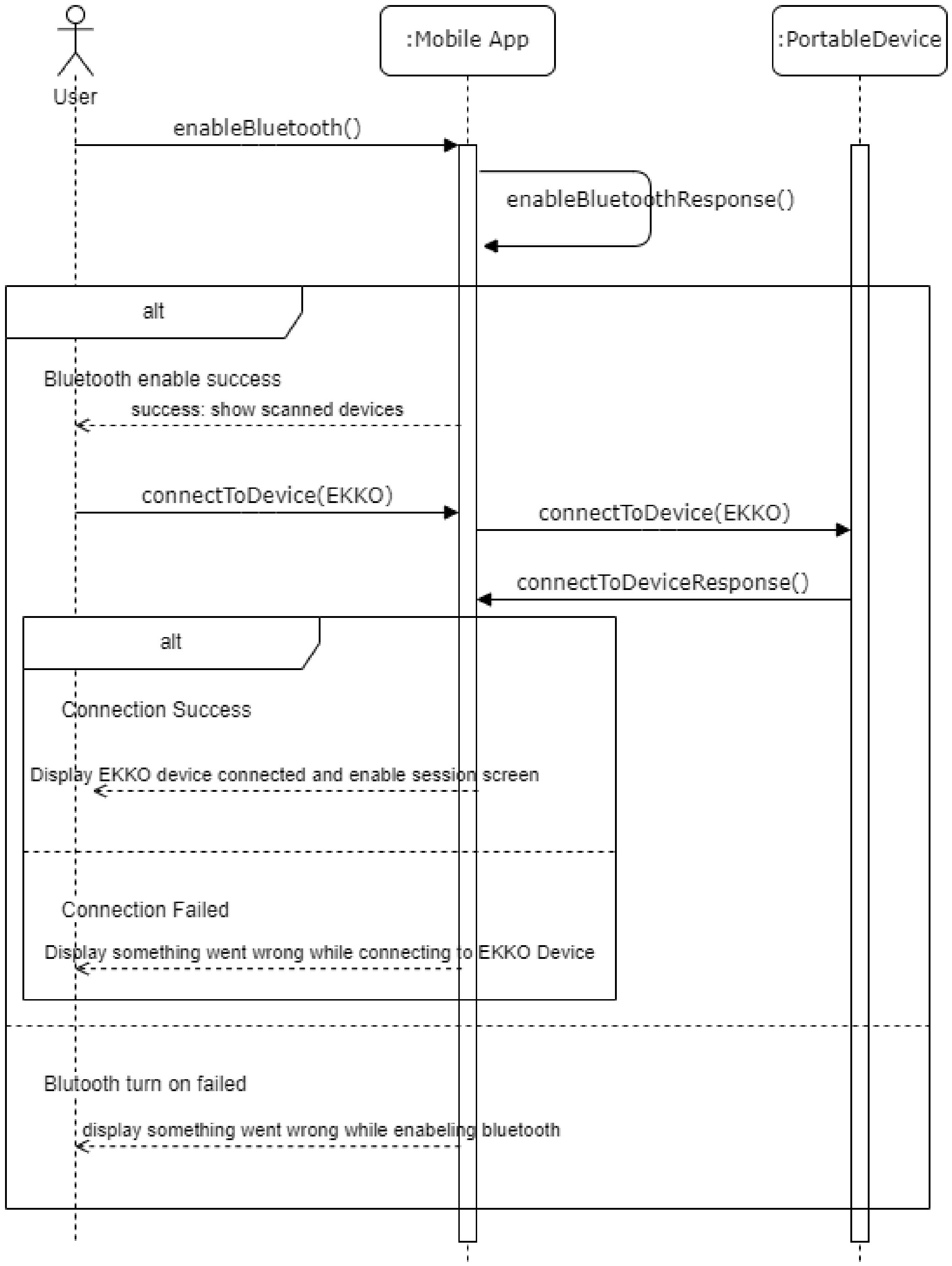


Figure 4.11: Sequence diagram of the "User Login" in the Subsystem SS3 Smartphone app, demonstrating connection with the Object Individual

.



User
:Mobile App
:PortableDevice
enableBluetooth()
enableBluetoothResponse()
alt
Bluetooth enable success
success: show scanned devices
connectToDevice(EKKO)
connectToDevice(EKKO)
connectToDeviceResponse()
alt
Connection Success
Display EKKO device connected and enable session screen
Connection Failed
Display something went wrong while connecting to EKKO Device
Blutooth turn on failed
display something went wrong while enabeling bluetooth

Figure 4.12: Sequence Diagram of the usage scenario "Connect Device via Bluetooth connectivity" for the Subsystem-level SS3 Smartphone App, with the Device Entity reflecting essential hardware characteristics.

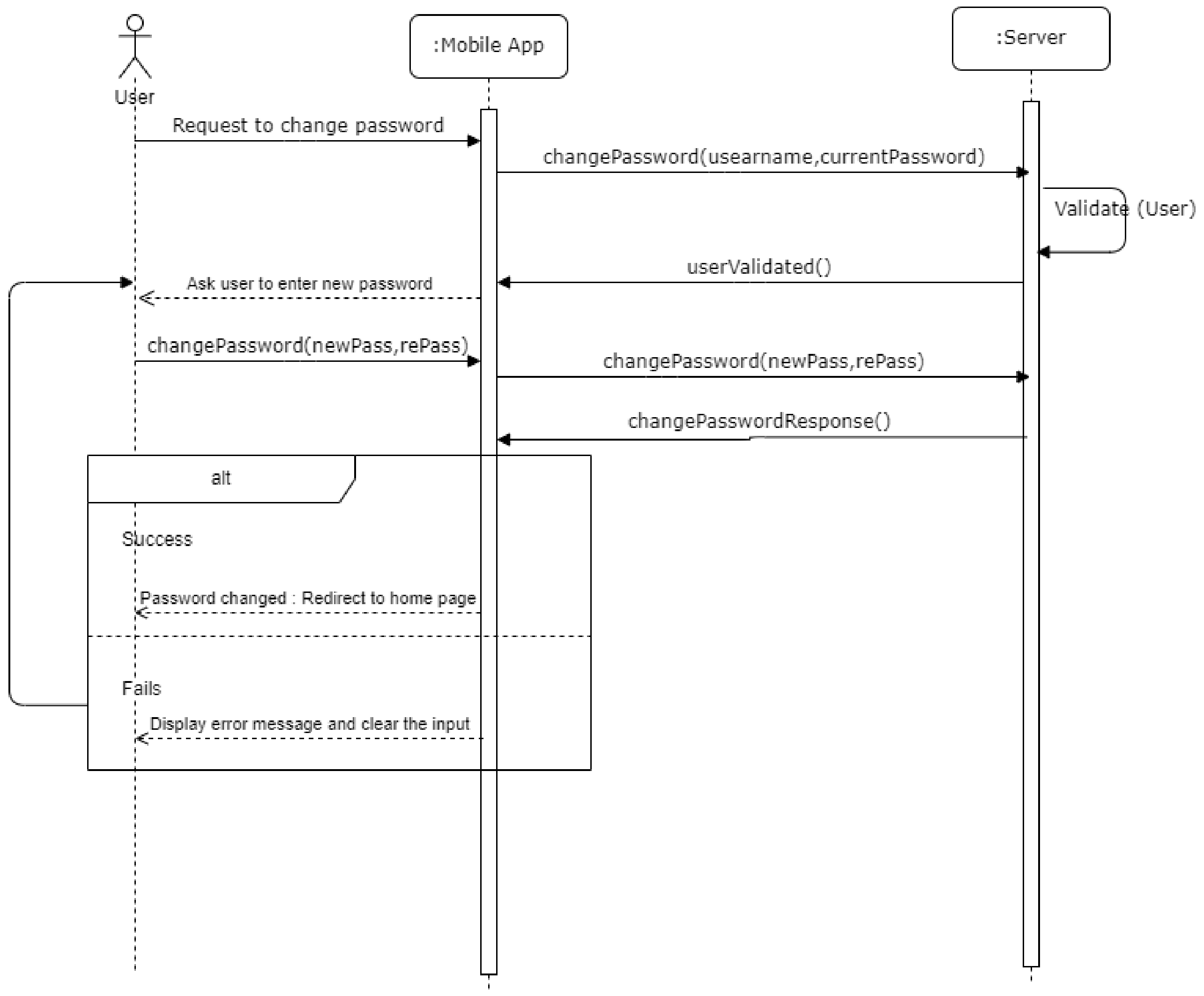

User
:Mobile App
:Server
Request to change password
changePassword(usearname,currentPassword)
Validate (User)
userValidated()
Ask user to enter new password
changePassword(newPass,rePass)
changePassword(newPass,rePass)
changePasswordResponse()
alt
Success
Password changed : Redirect to home page
Fails
Display error message and clear the input

Figure 4.13: A Sequence Diagram of the usage scenario "Change Password" for the SS3 Smartphone app, with the Home User Object including Fundamental individual Information, Individual Details, and Register Details.

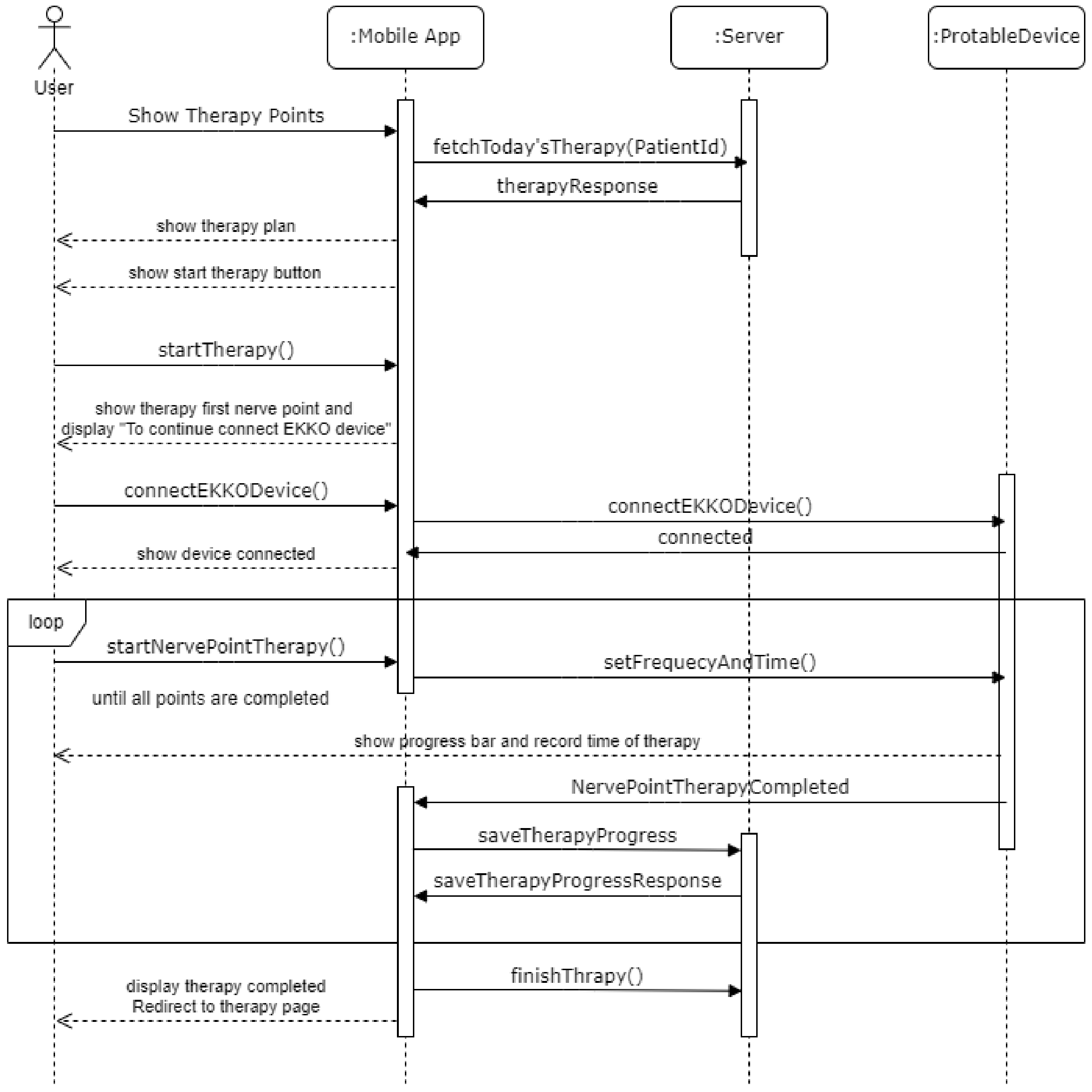

User
:Mobile App
:Server
:ProtableDevice
Show Therapy Points
fetchToday'sTherapy(PatientId)
therapyResponse
show therapy plan
show start therapy button
startTherapy()
show therapy first nerve point and display "To continue connect EKKO device"
connectEKKODevice()
connectEKKODevice()
connected
show device connected
loop
startNervePointTherapy()
setFrequecyAndTime()
until all points are completed
show progress bar and record time of therapy
NervePointTherapyCompleted
saveTherapyProgress
saveTherapyProgressResponse
finishThrapy()
display therapy completed Redirect to therapy page

Figure 4.14: Sequence Diagram for the usage scenario "View Therapy Status," The main Home User Object contains fundamental personal data, whereas the Session Therapy Object contains comprehensive details on the individual's therapeutic sessions.

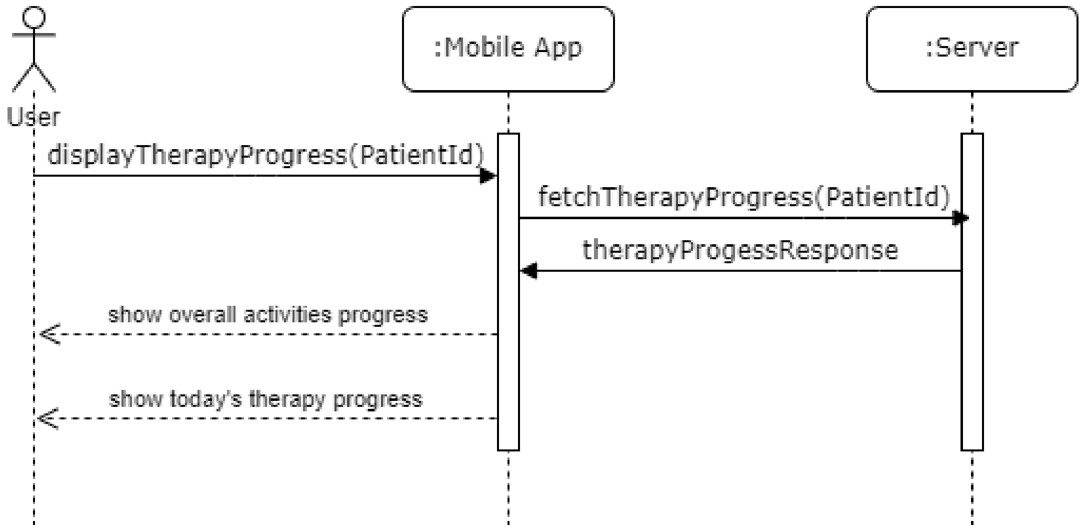

:Mobile App
:Server
User
displayTherapyProgress(PatientId)
fetchTherapyProgress(PatientId)
therapyProgessResponse
show overall activities progress
show today's therapy progress

Figure 4.15: Sequence Diagram for the usage scenario "Manage Therapy Session," The Home User Property contains private data, the Session Therapy Property contains descriptions of therapy sessions, and the Nerve Point Property has data concerning every single nerve point, while the Device Property contains data about the hardware.

Deployment Diagram

The deployment diagram depicts the tangible parts of the system, such as where hardware, software, and subsystem are deployed. Figure 5.15 depicts the deployment diagram, which includes all of our operational subsystems for the handheld and hospital variants. This design also provides connectivity data as needed, showing when our software systems are going to be installed.

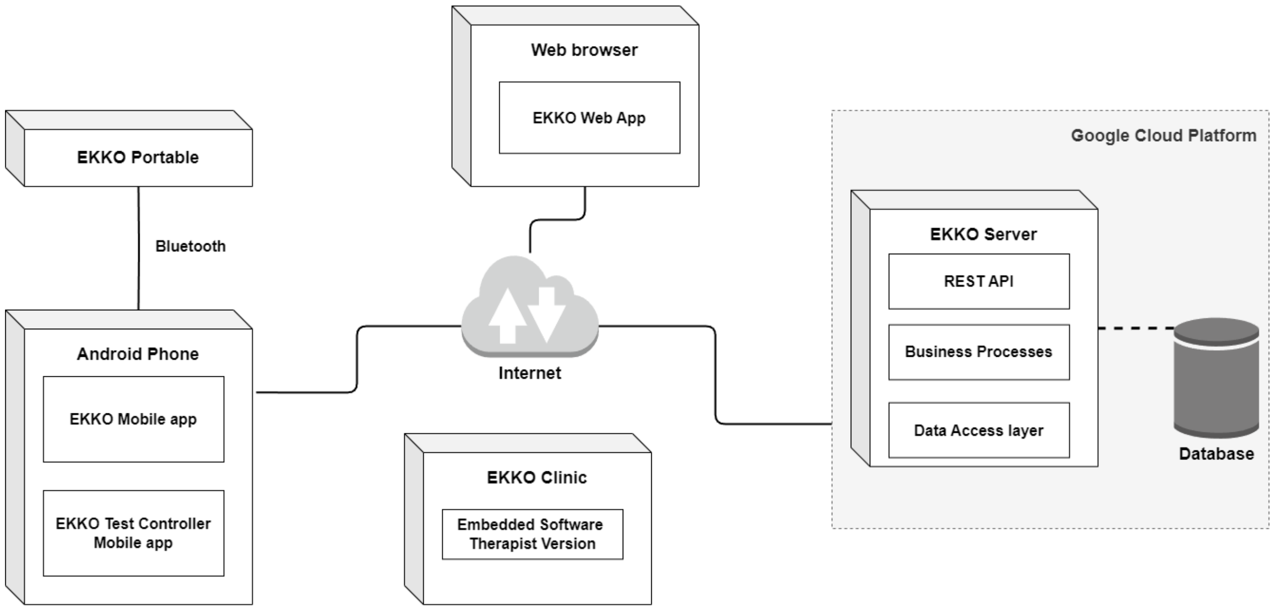


Figure 4.16: Deployment Diagram

4.6 Implementation

Many of the functional hardware along with software layouts are listed here for convenience.

4.6.1 Hardware

Figure 4.16 depicts the hardware of our healthcare equipment, which is available in two variants: hospital and handheld. The technical specifications of the hardware setup are as follows:

1. Software
    a. IDEs: Visual Studio Code and Platforms Input/Output.
    b. Framework: Expressif IDF (ESP 32)
    c. Language: C/C++
2. Electronics
    a) Processor: BLE enabled ESP32
    b) Circuit: Custom established on KiCAD having the corresponding characteristics: The battery driven and adapter-powered, regulation of voltage for a micro controller device, motor driver, cell powered / battery administration system.
3. Hardware

a. Design was created using SolidWorks.
b. Parts: LiPo-18650 Lithium battery, 12V vibration motor, 3D resin metal casing, applicator, and metallic manufactured heads.

4.6.2 Software

The technical aspects of software development.

a. IDE: Visual Studio Code
b. Operating System: Windows 10 and Linux (for deployment)
c. Technological stack includes the recently released Node, MongoDB, Angular, Express.js, and Docker.
d. Utility software: MongoDB Compass, Bitbucket etc

4.7 Testing

The fundamental goal of the testing strategy is to make sure whether the whole system, its subsystems, and each of the major components are working properly in accordance with the requirements provided by consumers and developers. Considering the fact that we utilize the AV and EAV models, testing is an essential component of every stage of the production phase. Our product includes four phases of testing.

1. Unit Testing
2. System/integration testing
3. Acceptance testing
4. Usability Testing

4.7.1 Test Items

Main things to be tested are given below.

1. Clinic hardware SS1 Subsystem
2. Embedded software SS2 Subsystem
3. Smartphone completely functional app SS3 Subsystem

SS4 Subsystem smartphone test controller app

4. Handheld Subsystem SS5
5. Web Application Subsystem SS6

Table 4-18: “Login” Functionality Test Case

Name of Test Case	Login
Identifier of Use Case	TC-01
Description of Test case	To get to the program, the user inputs a password and a username followed by clicking on the "Login" option.
Scope	SS3-Completely functional smartphone app
Expected Output	The program is functioning and the user has successfully signed in.

Table 4-19: “Sign up Request.” Functionality Test Case

Name of Test Case	Sign up
Identifier of Use Case	TC-02
Description of Test case	User requests to sign up to the smartphone app.
Scope	SS3-Completely functional smartphone app.
Expected Output	The query goes out and a notice appears saying "We will contact you shortly."

Table 4-20: “Device Connection via Bluetooth.” Functionality Test Case

Name of Test Case	Device connection through Bluetooth connectivity
Identifier of Use Case	TC-03
Description of Test case	For connection to the device, the individual simply presses the Bluetooth icon.
Scope	SS3-Completely functional smartphone app.
Expected Output	The device has been successfully linked together, and the graphical icon ought to adjust to reflect this.

Table 4-21: “Start Therapy Session.” Functionality Test Case

Name of Test Case	Start Therapy Session
Identifier of Use Case	TC-04
Description of Test case	The individual selects the session option from the menu list, then the session page appears, displaying all of the suggested nerve spots. The individual selects a nerve spot, changes the frequency, and finally presses "Start Session Therapy."
Scope	SS3-Completely functional smartphone app.
Expected Output	The therapeutic session needs to start at the desired frequency with the handheld variant of the equipment.

Table 4-22: “Manage Therapy Session.” Functionality Test Case

Name of Test Case	Therapy Session Management
Identifier of Use Case	TC-05
Description of Test case	a) The individual chooses the session option on the menu list, which opens the session page that includes every of the suggested nerve spots. The individual selects a nerve spot, changes the frequency, and finally presses "Start Session Therapy." b) Throughout the therapeutic session, you may interrupt or restart it.
Scope	SS3 - Completely functional smartphone app.
Expected Output	a) The therapy ought to begin at the desired frequency with the handheld variant of the equipment. A countdown timer ought to pop up on the display b) When the pause/resume option is pressed, the equipment and countdown ought to cease or continue properly.

Table 4-23: “Change Password” Functionality Test Case

Name of Test Case	Change Password
Identifier of Use Case	TC-06
Description of Test case	The individual demands that a password be changed from the profile area of the main menu. The individual inputs the updated password repeatedly.
Scope	SS3-Completely functional smartphone app
Expected Output	When the passwords don't match, a suitable error message ought to pop up. Once the passwords match up, the program will correctly change the login credentials and return the individual to the login screen.

Table 4-24: “View Recent Therapy Activities.” Functionality Test Case

Name of Test Case	View activities of recent therapy
Identifier of Use Case	TC-06
Description of Test case	After logging in, the individual is brought to the main page. On the other hand, an individual can go to the home page by pressing "Home" on the bottom menu bar.
Scope	SS3 - A completely functional smartphone app
Expected Output	The main display displays statistics for latest therapeutic actions as well as their past actions. Stats must include: a) Quantity of activities completed. b) A particularly recent task accomplished.

4.7.2 Approach

Developers as well as researchers do unit, system, and usability evaluations for all things. At first, comprehensive testing is performed to meet specified requirements. Following that, for any new and evolving demands, testing is carried out in an agile way, i.e., alongside development, based on the agile approach chosen. Clinicians work alongside developers and researchers to complete the items during testing for acceptance.

4.7.3 Item passing or failing Criteria

A test is regarded to have passed if all of the test case's characteristics have been effectively validated.

4.7.4 Test Case Description

An example test case specification details the processes used to ensure that the system functions properly. A test case consists of the test group, test procedures, test information, preconditions, and post conditions used to validate any specification. We developed testing scenarios for the solution based on the requirements specification, usage scenarios, and design representations. The test cases compare predicted and observed outcomes to assess whether our solution meets the requirements. Tables 4.19–4.25 provide examples of test cases. Depending on these examples, test scenarios for additional needs can be developed using the EAV paradigm.

4.8 Summary

In order to apply neural communication cognition therapy, a vibrotactile device capable of transmitting exceptionally low frequency vibrational analysis waves that activate neurons in the brain is necessary. This section discussed the system development of the vibrotactile device, which included software as well as hardware parts. Throughout determining requirements to testing, a modified variant of McHugh's AV paradigm was chosen and used in the development phase.

Specifications have been gathered and reported utilizing standard requirements techniques for engineering and the EAV model technique, which included everyone with an interest. These specifications were subsequently transformed into designs utilizing UML 2.0 and 2.5 annotations. Throughout the design stage, both the software and the hardware were implemented, with the system testing step serving as validation.

The AV model recommends combining conventional plan-driven and agile techniques for dealing with evolving demands. We improved the AV model by adopting a system engineering methodology that divided the entire structure into subsystems and managed variable and static requirements independently. This EAV-model technique was used during every stage of software development. The wave’s therapy medical device is available in a variety of hardware variants that can be powered by software across multiple platforms, including handheld devices, web, and integrated variants for hospital equipment. Handling such complicated structures with diverse criteria, particularly safety-sensitive and risk-oriented specifications, necessitates a comprehensive SDLC strategy. The EAV-model successfully controlled our product's lifecycle, although its adaptability to lighter and more straightforward healthcare products is uncertain and may be determined through additional usage to other medical products.

CHAPTER 5

CONCLUSION AND FUTURE WORK

To meet the safety critical needs of the medical devices and to meet the standards of regulatory authorities for medical devices, we have proposed an EAV model which is an improvement to McHugh's AV model that integrates agile and traditional approach to handle the requirements in a way where safety critical requirements are handled to have a high traceability and can easily be able meet the regulatory needs and other non-critical can be handled in agile window. In addition to this the strength of the model lies in the integration of system engineering approach through which the engineering of hardware and software can be managed.

So, we utilized the EAV model for the development of the neurotransmission cognitive theory/therapy-based gadget. The purpose of vibro-tactile gadget is to transmit ultra-low frequency vibrational waves for stimulation of the brain. The systems engineering of this gadget involves both hardware and software components. For product development, we proposed and selected an enhancement to McHugh's AV model, which we followed throughout the entire process, from requirements gathering to testing.

Before using the suggested process model, we chose the MDEVSPICE framework, which clearly defines IEC 62304 standards in several model forms. Our EAV model demonstrated full compliance with the PRM's procedures when we mapped it to the MDEVSPICE. After confirming that the model had every element required by the regulatory bodies, we started the development process by involving all relevant parties, gathering requirements, and documenting them using both traditional requirements engineering techniques and the EAV model approach. After that, the design is derived from these specifications using UML 2.0 and UML 2.5 notations.

Following system design, we implemented hardware and software, and finally tested the system. For extensive quality verification of the produced item, we calculated usability and reliability indicators using questionnaires and achieved positive findings.

Our wave therapeutic medical device comes in a variety of hardware and software configurations, including embedded, mobile, and online versions for use in clinics. To manage such complex systems with diverse types of criteria, such as safety-critical and risk-oriented needs, a whole system development life cycle approach is required. Based on the structure and complexity of our system, the EAV model performed well throughout the product’s life cycle however, its applicability to more compact and straightforward medical products is unknown and can be determined by using this model to analyze the performance of other medical products.

To demonstrate the compliance with regulatory authorities, we mapped the EAV model to the descriptive PRM of the MDEVSPICE Framework. In the future, we plan to map the EAV model to the PAM of MDEVSPICE, facilitating a more thorough alignment of the model with regulatory requirements.

REFERENCES

[1] Marshall, J.C., et al., What is an intensive care unit? A report of the task force of the World Federation of Societies of Intensive and Critical Care Medicine. Journal of critical care, 2017. **37**: p. 270-276.

[2] Loreto, M., T. Lisboa, and V.P. Moreira, *Early prediction of ICU readmissions using classification algorithms.* Computers in biology and medicine, 2020. **118**: p. 103636.

[3] Kramer, A.A., T.L. Higgins, and J.E. Zimmerman, *The association between ICU readmission rate and patient outcomes.* Critical care medicine, 2013. **41**(1): p. 24-33.

[4] Ponzoni, C.R., et al., Readmission to the intensive care unit: incidence, risk factors, resource use, and outcomes. A retrospective cohort study. Annals of the American Thoracic Society, 2017. **14**(8): p. 1312-1319.

[5] Desautels, T., et al., Prediction of early unplanned intensive care unit readmission in a UK tertiary care hospital: a cross-sectional machine learning approach. BMJ open, 2017. **7**(9): p. e017199.

[6] Chen, L.M., et al., Patients readmitted to the intensive care unit during the same hospitalization: clinical features and outcomes. Critical care medicine, 1998. **26**(11): p. 1834-1841.

[7] Singer, D.E., et al., Unexpected readmissions to the coronary-care unit during recovery from acute myocardial infarction. New England Journal of Medicine, 1981. **304**(11): p. 625-629.

[8] McIlvennan, C.K., Z.J. Eapen, and L.A. Allen, *Hospital readmissions reduction program.* Circulation, 2015. **131**(20): p. 1796-1803.

[9] FY2018-IPPS-Final-Rule-Data-Files. 2017; Available from: https://www.cms.gov/medicare/medicare-fee-for-service-payment/acuteinpatientpps/acute-inpatient-files-for-download-items/fy2018-final-rule-correction-notice-files.

[10] Wang, H.-j., et al., *Preventable readmission to intensive care unit in critically ill cancer patients.* World Journal of Emergency Medicine, 2018. **9**(3): p. 211.

[11] Baillie, C.A., et al., The readmission risk flag: using the electronic health record to automatically identify patients at risk for 30-day readmission. Journal of hospital medicine, 2013. **8**(12): p. 689-695.

[12] Kansagara, D., et al., Risk prediction models for hospital readmission: a systematic review. Jama, 2011. **306**(15): p. 1688-1698.

[13] Shams, I., S. Ajorlou, and K. Yang, A predictive analytics approach to reducing 30-day avoidable readmissions among patients with heart failure, acute myocardial infarction, pneumonia, or COPD. Health care management science, 2015. **18**: p. 19-34.

[14] Kessler, S., et al., Predicting readmission to the cardiovascular intensive care unit using recurrent neural networks. Digital Health, 2023. **9**: p. 20552076221149529.

[15] Moazemi, S., et al., Evaluating a recurrent neural network model for predicting readmission to cardiovascular ICUs based on clinical time series data. Engineering Proceedings, 2022. **18**(1): p. 1.

[16] Pishgar, M., et al., *Prediction of unplanned 30-day readmission for ICU patients with heart failure.* BMC Medical Informatics and Decision Making, 2022. **22**(1): p. 1-12.

[17] Kim, H., et al., *Scheduled and unscheduled hospital readmissions among diabetes patients.* The American journal of managed care, 2010. **16**(10): p. 760.

[18] Nijhawan, A.E., et al., Half of 30-day hospital readmissions among HIV-infected patients are potentially preventable. AIDS patient care and STDs, 2015. **29**(9): p. 465-473.

[19] McAdams-DeMarco, M.A., et al., *Frailty and early hospital readmission after kidney transplantation.* American journal of transplantation, 2013. **13**(8): p. 2091-2095.

[20] Curto, S., et al. Predicting ICU readmissions based on bedside medical text notes. in 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). 2016: IEEE.

[21] Harutyunyan, H., et al., Multitask learning and benchmarking with clinical time series data. Scientific data, 2019. **6**(1): p. 96.

[22] Choi, Y., C.Y.-I. Chiu, and D. Sontag, *Learning low-dimensional representations of medical concepts.* AMIA Summits on Translational Science Proceedings, 2016. **2016**: p. 41.

[23] Johnson, A.E., et al., *MIMIC-III, a freely accessible critical care database.* Scientific data, 2016. **3**(1): p. 1-9.

[24] Syed, M., et al. Application of machine learning in intensive care unit (ICU) settings using MIMIC dataset: systematic review. in Informatics. 2021: MDPI.

[25] Bennett, D. and J. Bion, *ABC of intensive care. Organisation of intensive care.* BMJ (Clinical research ed.), 1999. **318**: p. 1468-70.

[26] Messaoudi, Z., The 3 main steps of Machine Learning. 2020.

[27] Pandian, S., Time Series Analysis and Forecasting | Data-Driven Insights. 2023.

[28] Pakbin, A., et al. Prediction of ICU readmissions using data at patient discharge. in 2018 40th annual international conference of the IEEE engineering in medicine and biology society (EMBC). 2018: IEEE.

[29] Min, X., B. Yu, and F. Wang, Predictive modeling of the hospital readmission risk from patients' claims data using machine learning: a case study on COPD. Scientific reports, 2019. **9**(1): p. 2362.

[30] Rojas, J.C., et al., Predicting intensive care unit readmission with machine learning using electronic health record data. Annals of the American Thoracic Society, 2018. **15**(7): p. 846-853.

[31] McWilliams, C.J., et al., Towards a decision support tool for intensive care discharge: machine learning algorithm development using electronic healthcare data from MIMIC-III and Bristol, UK. BMJ open, 2019. **9**(3): p. e025925.

[32] Li, Z., et al., Early prediction of 30-day ICU re-admissions using natural language processing and machine learning. arXiv preprint arXiv:1910.02545, 2019.

[33] Assaf, R. and R. Jayousi. 30-day hospital readmission prediction using MIMIC data. in 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT). 2020: IEEE.

[34] Liu, W., et al., Predicting 30-day hospital readmissions using artificial neural networks with medical code embedding. PloS one, 2020. **15**(4): p. e0221606.

[35] Moerschbacher, A. and Z. He, Building prediction models for 30-day readmissions among icu patients using both structured and unstructured data in electronic health records. medRxiv, 2021: p. 2021.08. 10.21261858.

[36] Shi, K., et al. Predicting unplanned 7-day intensive care unit readmissions with machine learning models for improved discharge risk assessment. in AMIA Annual Symposium Proceedings. 2022: American Medical Informatics Association.

[37] Orangi-Fard, N., A. Akhbardeh, and H. Sagreiya. Predictive model for icu readmission based on discharge summaries using machine learning and natural language processing. in Informatics. 2022: MDPI.

[38] Rafi, P., A. Pakbin, and S.K. Pentyala, Interpretable deep learning framework for predicting all-cause 30-day ICU readmissions. Texas A&M University, 2018.

[39] Wang, H., et al., *Predicting hospital readmission via cost-sensitive deep learning.* IEEE/ACM transactions on computational biology and bioinformatics, 2018. **15**(6): p. 1968-1978.

[40] Lin, Y.-W., et al., Analysis and prediction of unplanned intensive care unit readmission using recurrent neural networks with long short-term memory. PloS one, 2019. **14**(7): p. e0218942.

[41] Ashfaq, A., et al., *Readmission prediction using deep learning on electronic health records.* Journal of biomedical informatics, 2019. **97**: p. 103256.

[42] Barbieri, S., et al., Benchmarking deep learning architectures for predicting readmission to the ICU and describing patients-at-risk. Scientific reports, 2020. **10**(1): p. 1111.

[43] Zhang, D., et al., Combining structured and unstructured data for predictive models: a deep learning approach. BMC Medical Informatics and Decision Making, 2020. **20**(1): p. 1-11.

[44] Sheetrit, E., M. Brief, and O. Elisha, Predicting Unplanned Readmissions in the Intensive Care Unit: A Multimodality Evaluation. arXiv preprint arXiv:2305.08139, 2023.

[45] Johnson, A.E.P., T.; Mark, R., MIMIC-III Clinical Database (version 1.4). PhysioNet. 2016.

[46] Brown, S.E., S.J. Ratcliffe, and S.D. Halpern, *An empirical derivation of the optimal time interval for defining ICU readmissions.* Medical care, 2013. **51**(8): p. 706.

[47] O'Shea, K. and R. Nash, *An introduction to convolutional neural networks.* arXiv preprint arXiv:1511.08458, 2015.

[48] Park, J., D. Yi, and S. Ji, *Analysis of recurrent neural network and predictions.* Symmetry, 2020. **12**(4): p. 615.

[49] Le, X.-H., et al., Application of long short-term memory (LSTM) neural network for flood forecasting. Water, 2019. **11**(7): p. 1387.

[50] ArunKumar, K., et al., Comparative analysis of Gated Recurrent Units (GRU), long Short-Term memory (LSTM) cells, autoregressive Integrated moving average (ARIMA), seasonal autoregressive Integrated moving average (SARIMA) for forecasting COVID-19 trends. Alexandria engineering journal, 2022. **61**(10): p. 7585-7603.