Navsangeet Kaur
CSCI 370
Software Engineering
Dr. Svadlenka

# Architecture and Design

## Introduction

The document provides a high level overview and explains the architecture of the Baby Heart system. It defines the goals of the architecture, the use case supported by the system, architectural styles, its major components that have been selected and the interaction among them. The major component is heart monitoring devices. The subsystems in the Baby Heart System are: optical LED light, LED light sensor, and a transducer. These subsystems have interaction among them such as the LED light sensors and a transducer communicate with each other to detect the fetal heartbeat per minute and displays the interactive outputs from both devices and decides whether the heartbeat is detected or not. Baby Heart system architecture fits the layered pattern because it separates skill sets by layers which makes it an easy way of writing well organized and testable applications.

Baby Heart needs an architecture that can help in separating all the skills and techniques so that all the requirements are considered and become priority when visible in each layer. It will help differentiate the requirements so they are fulfilled and not overlooked because all the requirements are necessary to develop satisfactory, acceptable, and exceptional system. The layered pattern will fulfill the non-functional requirements such as security, performance, data privacy, confidentiality which are the main characteristics of the system. These are supported the layered architecture pattern using Quality of Service, Smartness, and Security which ensures the quality of the data and analytic results, and visualization, ensures privacy and security. The only assumption that has been made in specifying the architecture is that the subsystems such as optical LED sensor and LED light, and the transducer will work together in coordination and won't conflict with each other's results. Also, that they won't clash with each other's systematic process.

## Architecture and Design Philosophy

Baby Heart architecture and design that I want to develop concerns the complexity of the major process of subsystems. I considered both the layered and pipe-filter patterns. I used the process of elimination that discarded the pipe-filter pattern because it's not a good choice for interactive systems and leads to loss of performance and increased complexity because of excessive parsing and unparsing. Therefore, I choose the layered pattern because it separates the user interface from business logic and business logic from the data access logic. So, the separation of these logic layers are achieved by the layered pattern architecture that allows to swap and reuse the components. The different components can

be independently updated, and maintained on different time schedules. It provides high testability independently. It makes the system possible to configure the different levels of security to different components on different layers.

**Architectural views**

The developed set of views enable the architecture to be communicated to, understood by all the stakeholders and verify that their concerns are addressed by the system. Thus, these views are a representation of a whole system from the perspective of all the concerns set for the system. "Baby Heart" presents the architecture as a series of following views; physical view, logical view, process view and deployment view.

- ***Logical view*** concerned with the functionality that the system provides to end-users. It specifies system decomposition into conceptual entities such as objects and connection between them. So, it helps to understand the interactions between the entities in the problem space domain of the application and their potential variation. It is comprised of 2 main packages:
  - *User Interface*: It contains classes for each of the forms that the actor uses to communicate with the System. Boundary classes exist to support login, maintaining personal info, viewing visual heartbeat records, and viewing audio heartbeat records.
  - *Business Services:* It contains control classes for interfacing with the controlling user registration of an account, and managing the user records.

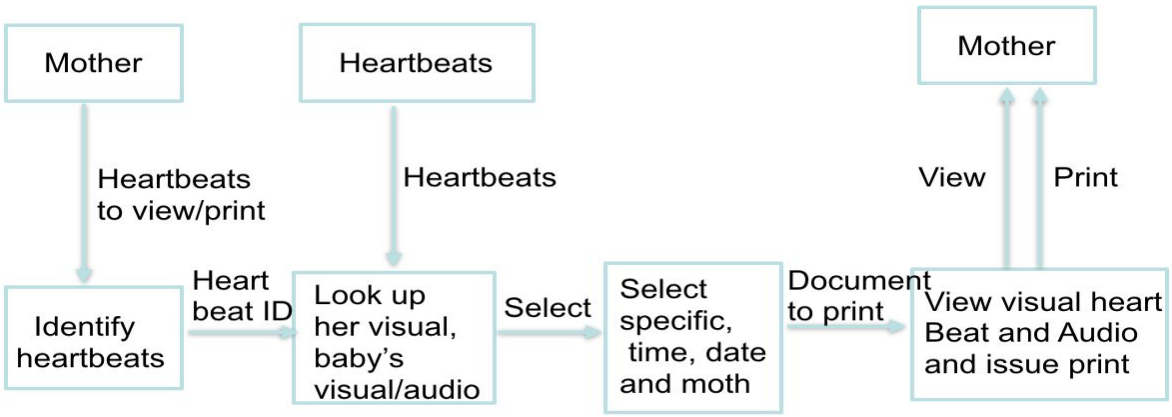


Figure: Logical view

- ***Process View*** describes the tasks involved in the system's execution, their interactions and configurations. It supports some non-functional requirements such as performance and availability of the system. In my system, it supports the performance of devices, and processes and 24/7 availability of the system to the users. It also illustrates the classes organized as executable processes that exist to support user registration and functions, and get records. It helps the system to decompose into many runtime execution units and organize them to provide communications between them.

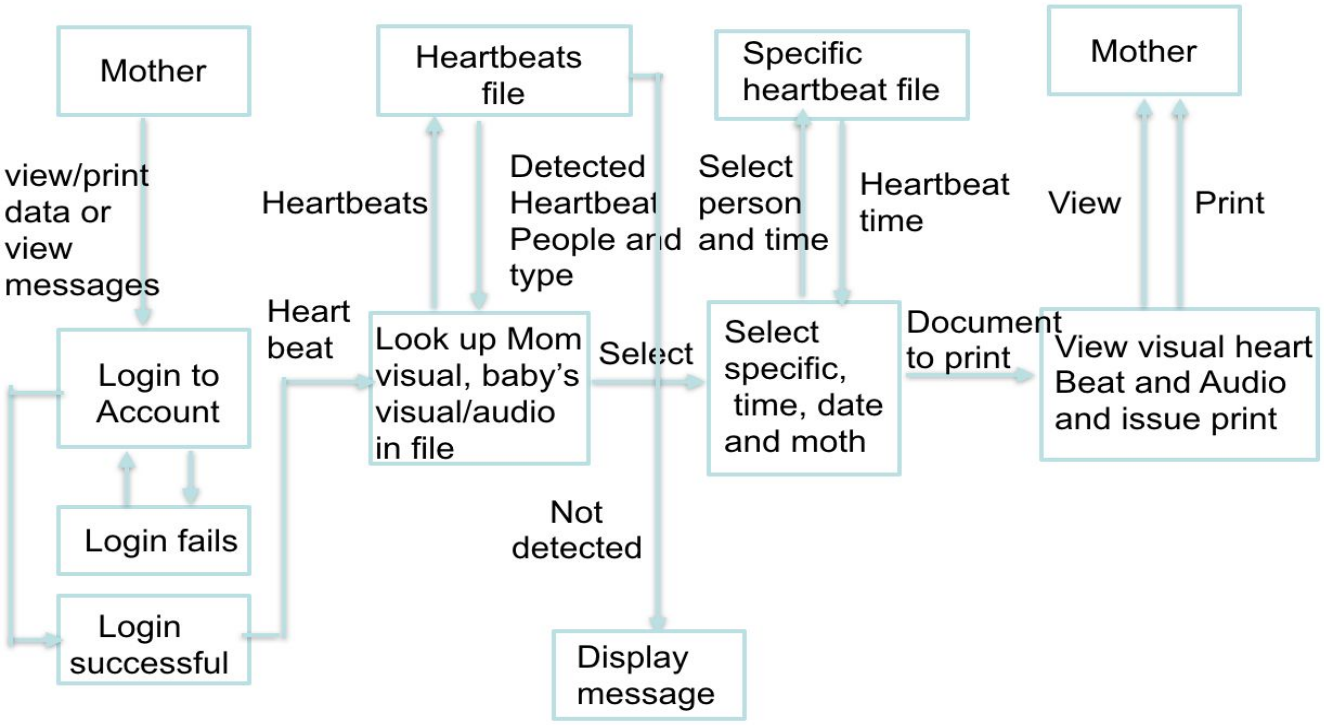


Figure: Process view

- ***Physical view*** supports several non-functional requirements such as scalability, performance, capacity, availability, and reliability. The software executes on processing nodes or network channels. It also considers network capacity, latency, and performance related considerations. The components are hardware entities and the links are communication pathways. For instance, it helps the system in providing access to the record database with no more than a 10 second latency and must be able to display heartbeats every second. Also, it confirms that the system works 24/7 with less than 4% downtime.

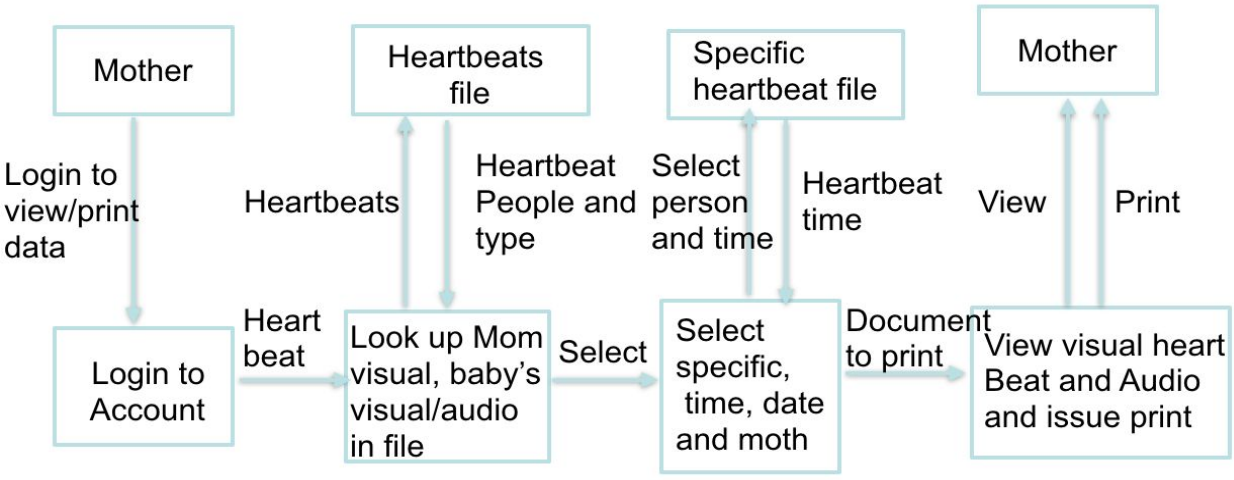


Figure: Physical view

- ***Development view*** It describes the various physical nodes for the most typical platform configurations. Also describes the allocation of tasks (from the Process View) to the physical nodes. It maps software component elements to actual physical directories, and files in the development environment. This includes Baby Heart node which is a central node and has threads to log devices, heartbeat display, audio display, network interface which is connected to the network server, record printer and heartbeat detectors which gives out processing and memory storage.

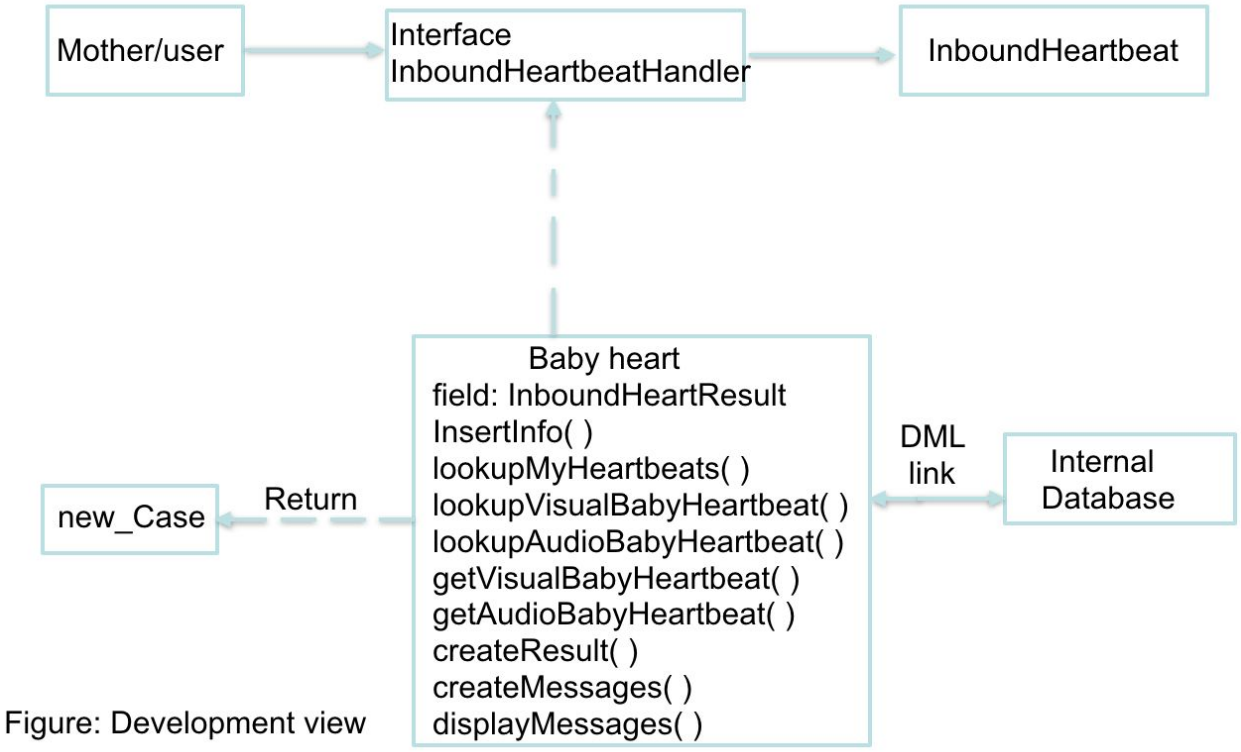


Figure: Development view

**Design Models**

- ***Context model:*** This model lets the developer to establish the system boundaries, which means what lies inside and outside the system. It helps in deciding what features are implemented in the system and what features are in other associated systems. The Baby Heart system uses an activity diagram, the system confirms if two heartbeats are detected, records it in the system and reports it to the system. Then the system decides between two option if the heartbeats are detected at two different rates and are differentiated, it will record the visual of the heartbeat as well as audio of the baby's heartbeat using LED device and transducer respectively, otherwise display that the other heartbeat is not detected and consult to the doctor. Thus, when the case goes out to the hospital that lies outside the system.

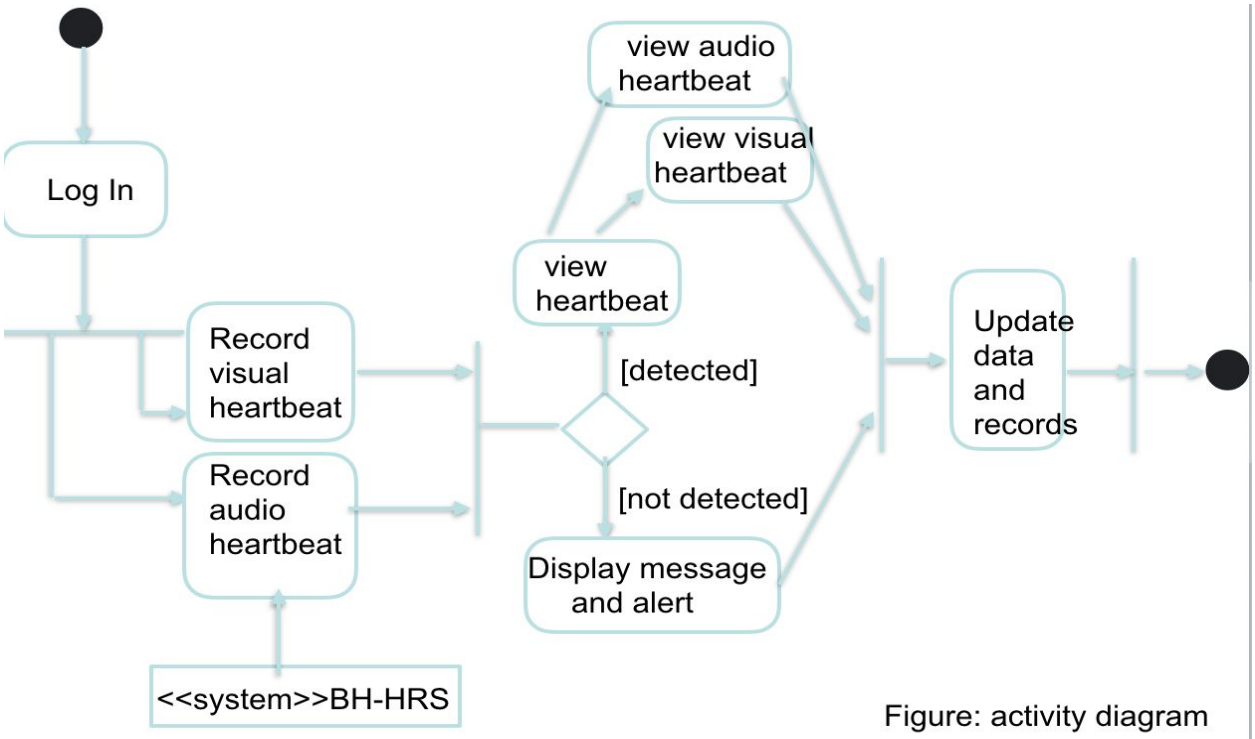


Figure: activity diagram

- ***Structural model:*** This model displays the organization of the system which is made of components and their relationships. It also discusses the flow and design of the architecture. It uses class diagrams to show the classes in a system and associations between them. All the classes either have 1 to 1, 1 to many, or many to many relationships between them. The Baby Heart system has 4 classes: mother, baby info, mother heart, baby heart, mother condition, doctor, and general practitioner. The mother class contains name, age, address, current pregnancy month, and delivery due date. The mother condition class includes mothers' health conditions and concerns such as allergies, etc. The general practitioner class has practice, phone number, email, and address. The doctor class has a doctor's name, hospital address, phone number, email, staff number, and pager number. The mother heart class has saved records of mothers' heartbeat during the whole pregnancy. The baby heart has a saved record of audio and visual of babys' heartbeat.

  Furthermore, mother class has many to 1 or many to many relationships with the baby's info depending if they are having one or more than one unborn baby. Additionally, the mother class also has a many to many relationship with condition class as well as many to 1 relationship with consultant. The mother class also has many to many relationships with mother heart as well as baby heart.

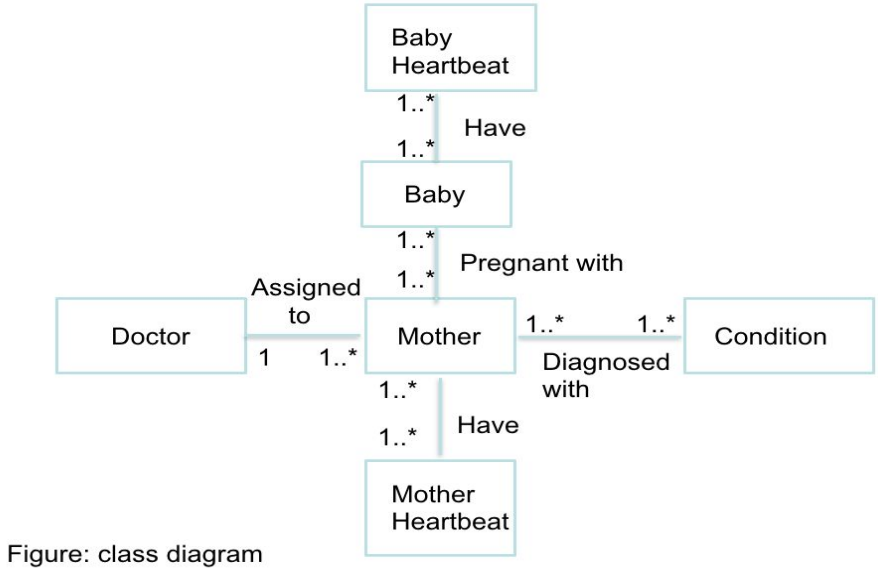


Figure: class diagram

- ***Behavioral model:*** This model conveys what happens and what is supposed to happen when the system responds to stimuli from its environment as the system is executing. The Baby Heart system uses a data-driven model that has to be processed by the system, and an event-driven model that triggers the system processing. The data-driven model in my system processes when the user tries to log in to the system, it first validates if the input is correct and acceptable, then validates. After validation, if it results "OK" then the user can access the data in the database to view or print, otherwise can attempt login again. As for the event-driven model, if the mother gets into unusual situations physically such as getting into an accident or fall, etc., the event triggers the system processing and reports it to the system to display the information to the user.

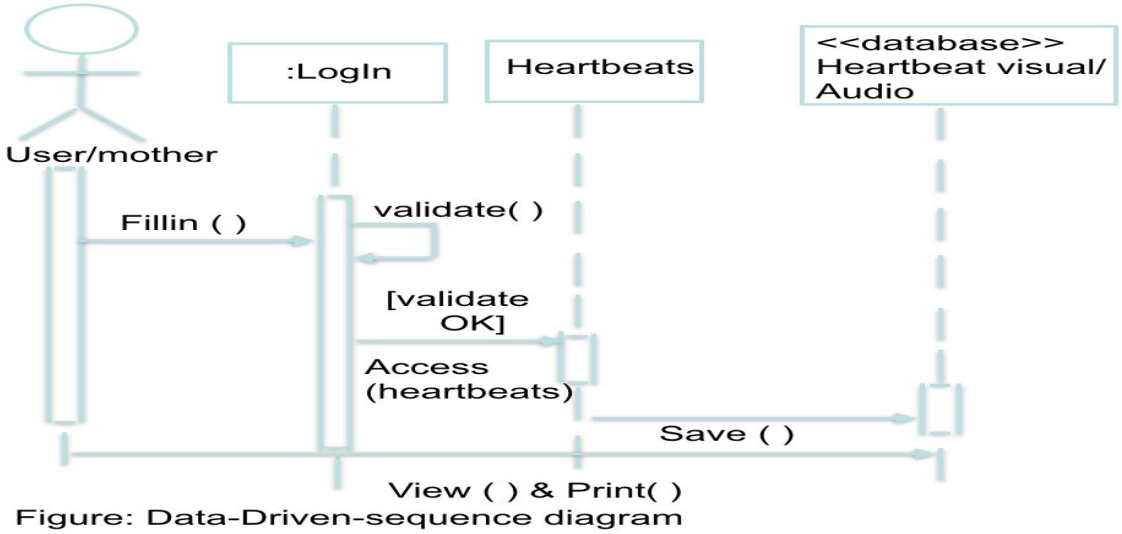


Figure: Data-Driven-sequence diagram

- ***Interaction model:*** This model reveals the user interaction, system-to-system interaction, and component interaction. Thus, a use case is chosen so the mother may access her and her baby's heartbeat that has account security. The heartbeat record may be either the

previous heartbeats or the updated heartbeats of a mother or her fetus. In the Baby Heart system, the actor is the mother, the precondition is the mother and the baby's heartbeat has been updated and the user must have an appropriate username and password to access the information and the results. The postcondition is the record has been updated and accessed by the user. The main success scenario is the system has the updated heartbeats in the general mother and baby heartbeat record databases, the user inputs their correct and accepted username and password and gets into the system and gains access to the information. The use case extends when the user inputs the incorrect username and/or password, then the system displays an error message and backs out of the use case.

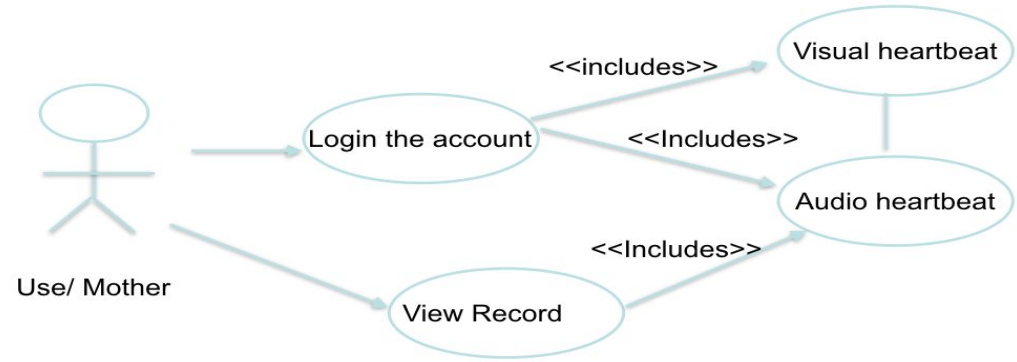


Figure: Use case

Furthermore, the algorithms and associated data structures that are critical to the success of Baby Heart system are: Hashing, and Search algorithm.

1. ***Hashing:*** Hashing is crucial for my system because it supports password storage which gives extra security to the user without compromising their confidentiality.. Instead of wasting time in storing the password repeatedly, it helps the system to match the user password to the given attempted password while providing protection to the users. The class hashmap is best to use in this system because it stores key value pairs and cannot contain duplicate keys that will prevent two different users from having the same password and it uses a division method that helps in generating the key using mathematical formula.
2. ***Search Algorithm***: A search algorithm uses binary search to search an element from the sorted dataset. It divides into half portions and narrows it down to select the desired element. In the Baby Heart system, search algorithms help in finding the heartbeat from specific time, date or month, or it also supports the system in finding specific waves where the heartbeat is abnormal. It also helps in debugging which is essential to prevent faults and errors which otherwise contribute to functionality issues. Additionally, the depth-first search is efficient because it follows one path deep into the root until it reaches the goal otherwise backtrack. This method is helpful because it will go through the whole baby's heartbeat and find any abnormalities and if it is not found then backtrack to find any.