**Chapter III**

**Design and Methodologies**

This chapter outlines and examines how the study's proponents obtained the essential data and information. It describes who the responses would be. This also illustrates the study method, data collection strategy, and software development tool employed.

**INPUT PROCESS OUTPUT**

**DSS App for De Roxas Pig farm**

**Agile Development Methodology**

* Planning
* Architecture
* Implementation
* Testing
* Deployment
* Maintenance

**Knowledge Requirement**

Related Literature

* DSS User Interface Design
* Types of DSS

**Hardware Requirements**

* Android Cellphone
* Laptop
* SSD

**Software**

**Requirements**

* Android Studio
* Flutter Framework
* Figma

**Fig. 3.1 IPO Diagram**

**Knowledge Requirement**

The proposed development of DSS for Pig Farm will follow a model that includes planning, architecture, implementation, testing, deployment, maintenance. This ensures that each module of the system is thoroughly tested and meets the necessary requirements before moving on to the next phase.

There are many user interfaces (UI) that can be used depending on the device and the user's requirements. UIs come in different forms such as graphical user interfaces (GUIs), command line interfaces (CLIs), menu-driven UIs, touch UIs, voice UIs (VUIs), form-based UIs, and natural language UIs.

According to an article by the Indeed Editorial Team with a title of “8 Decision Support System Examples to Guide Decision-Making” There are different types of Decision Support Systems (DSS) that can be used depending on the complexity of the decision problem. Some of the common types of DSS include data-driven, model-driven, knowledge-driven, communication-driven, document-driven, intelligent DSS, manual DSS and hybrid DSS. Document-driven DSS. The researchers are aiming to make a document driven DSS.

A document-driven DSS retrieves unstructured information from a variety of electronic sources. It searches webpages, documents in databases and other information based on a user's search terms to gather relevant information.

Developing a DSS for pig farming proper knowledge and skills in order to achieve the goal of creating a reliable and efficient system. The hardware requirement for the system should have at least the following specifications; 2GB of RAM; Internal memory of 8GB, the ideal is 16GB; Android system above 4.1. As for the software requirements the researchers will be using Android Studio as the Integrated Development Environment (IDE) and Flutter framework for the development of the app.

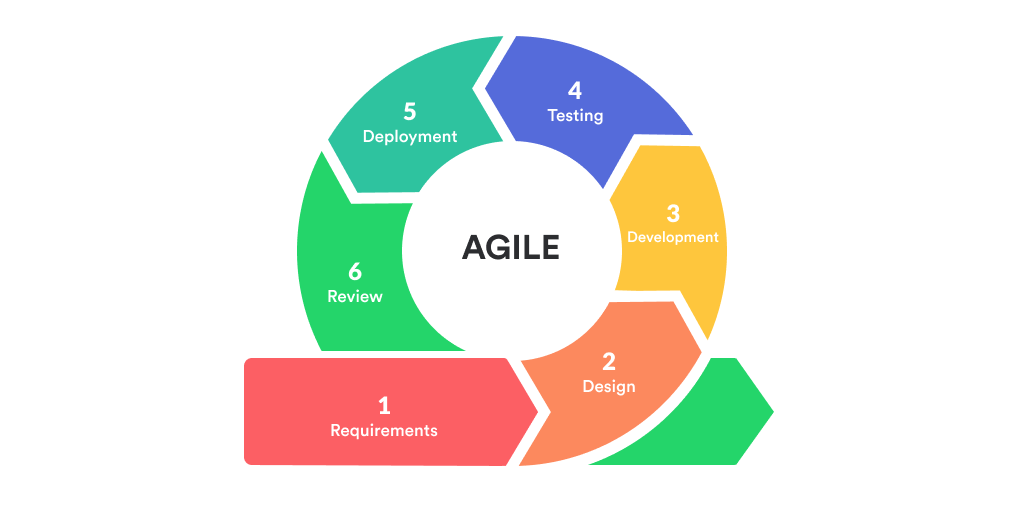
Flutter transforms the app development process. Build, test, and deploy beautiful mobile, web, desktop, and embedded apps from a single codebase, Dart programming language is used to create Flutter apps. Dart programming is similar to other programming languages such as Kotlin and Swift, and it may be trans-compiled into JavaScript code. Flutter is primarily designed to create 2D mobile apps that can operate on both the Android and iOS platforms. Dart is a programming language designed for client development, such as for the web and mobile apps. It is developed by Google and can also be used to build server and desktop applications. Dart is an object-oriented, class-based, garbage-collected language with C -style syntax. Dart can compile to either native code or JavaScript. In addition, the researchers will be using Figma to create prototypes of UI, Figma is a collaborative web platform for interface design, with desktop apps for macOS and Windows enabling additional offline functionalities. Figma's feature set focuses on user interface and user experience design, with a concentration on real-time collaboration and the usage of a variety of vector graphics editor and prototyping tools.

Hardware equipment is simple, researchers need laptop for Android Studio to run and Android cellphone to run the app iteration during the development process.

**Software Development Method**

The researchers employed a developmental research approach. The primary goal is to describe the data and characteristics of the subject under study.

The Agile Development Methodology was adopted by the researcher in their software development methodology. The Agile Development process is an advanced approach to creating software solutions that prioritize agility and rapidity. Agile development employs iterative and incremental methods to achieve error-free and timely delivery. Agile methodologies address this problem by allowing for backward tracking and working in increments, in which smaller components of a larger feature set are produced in time-limited cycles. This strategy taught the researchers a logical concept and guided them through the application process. The Agile Development Methodology provides an effective procedure in constructing the phases of the system.



**Fig. 3.2 Agile Development Methodology**

The phases illustrated in the diagram represent stages in the research model. Agile software development is based on the standard software development life cycle, which consists of planning, architecture, implementation, testing, deployment, and maintenance.

**Requirements**

The process of determining what will be built, how it will be built, and when it will be delivered is referred to as planning for the requirements. It entails the development team and stakeholders working together to define and prioritize features, set project goals, and design a plan of action for the development process.

**Design**

The process of developing and implementing the overall structure and framework of a software system is referred to as architecture. It entails identifying the system's components and subsystems, defining their duties, and specifying how they interact with one another in order to achieve the system's needs.

**Development**

Agile Development is an iterative and incremental process in which the development team consistently delivers working software features throughout the development cycle. The team can adjust to changing requirements and guarantee that the product fulfills the expectations of everyone involved by releasing software features in increments. This where the programmer does build the application and iteration.

**Testing**

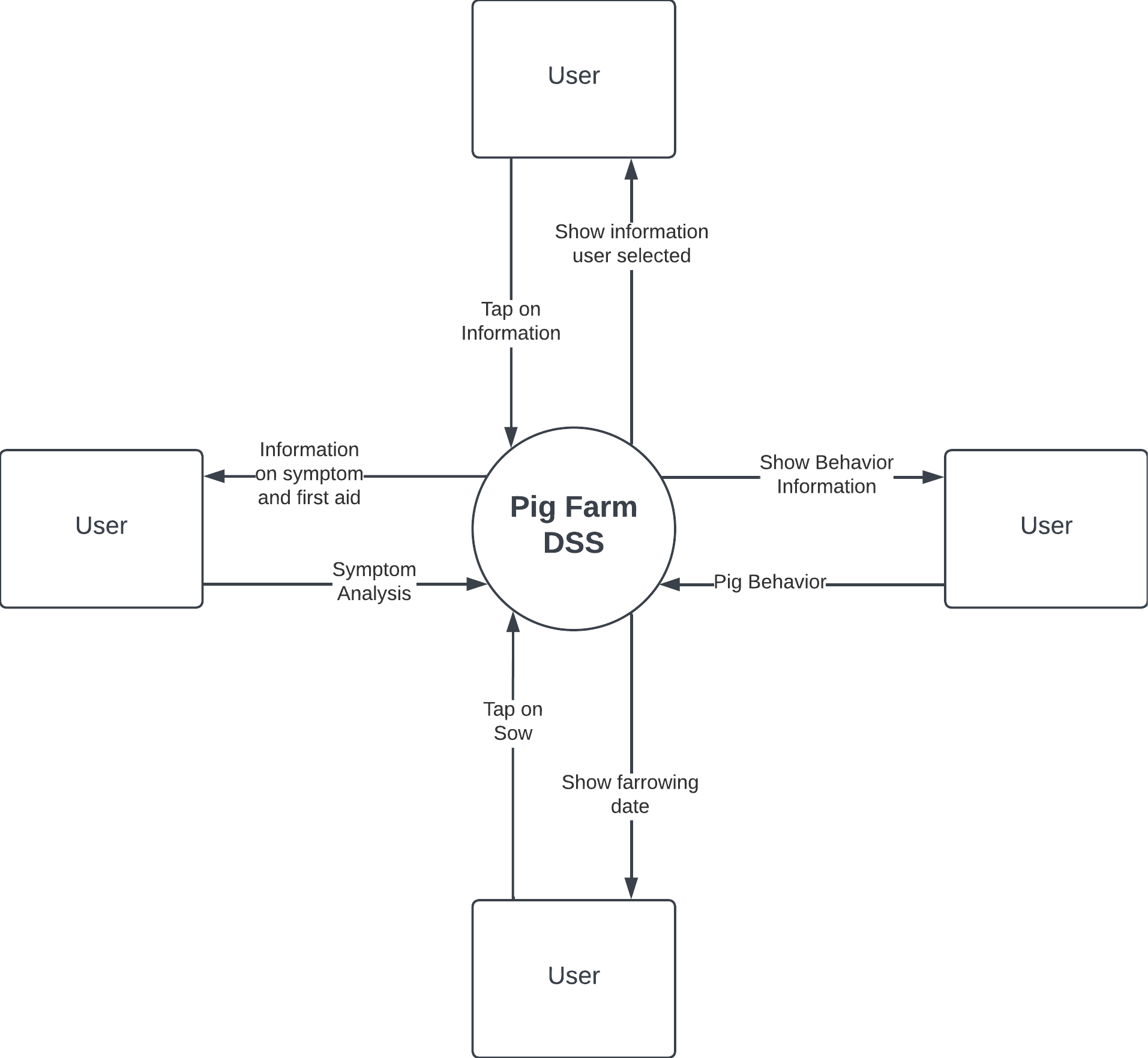
Testing is the process of reviewing software to determine whether or not it satisfies the requirements imposed by the parties involved and whether or not it functions appropriately. Testing is an essential component of Agile development since it assists in the detection of flaws and problems at an earlier stage in the development cycle, hence lowering the likelihood of more expensive rework and delays further down the line.

**Deployment**

The process of putting software into production and making it accessible to users at the end of the distribution chain is referred to as deployment. The deployment phase is an essential component of the agile software development process since it denotes the point at which the product is made usable by everyone involved in the project.

**Review**

The term "review" refers to the continual process of providing support for, as well as updates to, the software in order to guarantee that it continues to perform properly and satisfies the shifting requirements of the stakeholders. Maintenance is an important component of Agile Development since it helps to ensure that the product continues to give value over time. This is why maintenance is considered an integral part of Agile Development.

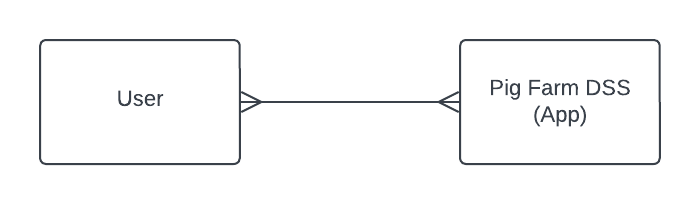


**Fig.3.3 Context Diagram**

The context diagram shows the basic user interaction using the system, the user gives an input and the system outputs the information.

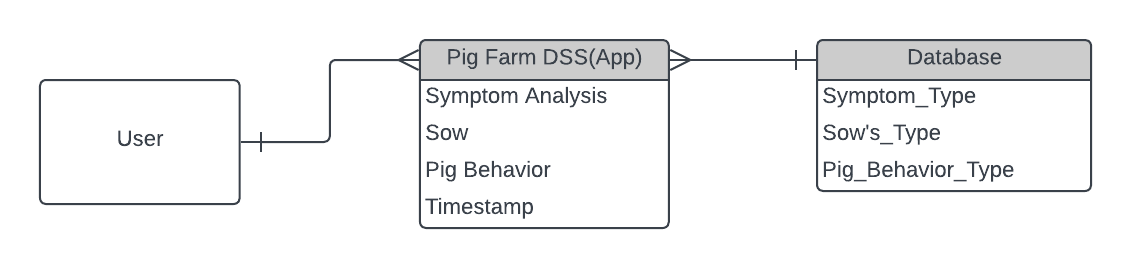
**Entity Relation Diagram**

An Entity Relationship Diagram (ERD) is a powerful tool that helps developers understand how data is organized and related to each other. It is a visual representation of the database structure that can help developers create and define clear relationships among the entities and attributes involved. ERDs are critical to the design and development of databases as they provide a logical structure of the database that can be easily understood by users. They serve as documentation tools and effectively communicate the logic of the database to users. ERDs are essential for any business that wants to gain a better understanding of the data contained in their database and connect the logical structure of the database to users.

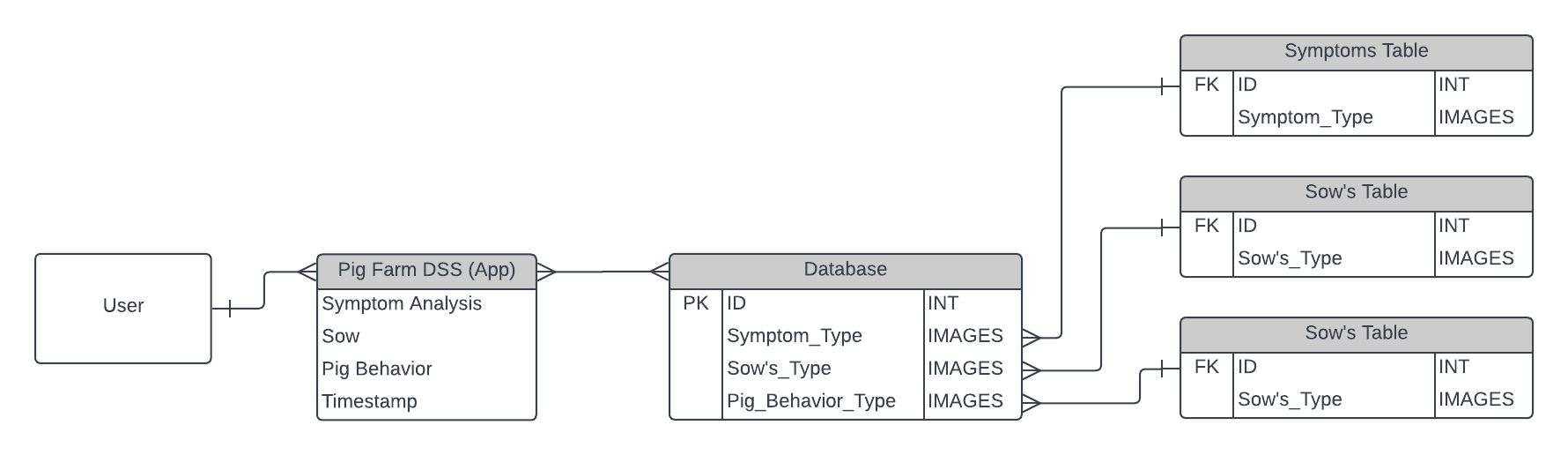


**Fig 3.4 Conceptual Data Model**

A high-level data model known as a conceptual data model (CDM) describes concepts and the connections between them. By identifying the business objects involved, a holistic picture of the system is presented. A CDM simulates the relationships between the business objects that should be present in a system. NOT which tables are defined, but which entities are.

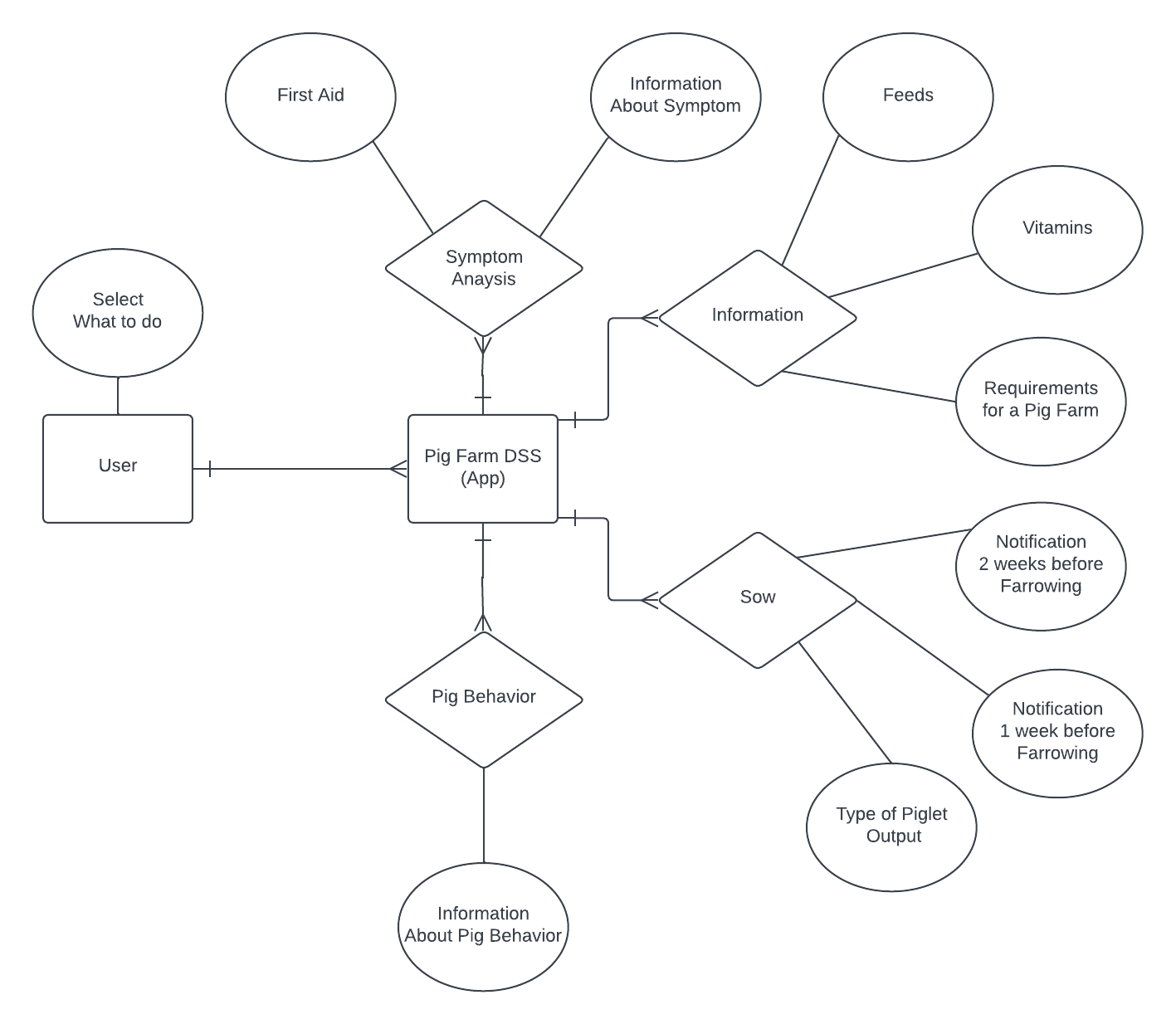


**Fig 3.5 Logical Data Model**

A conceptual model is made richer by introducing operational and transactional entities and clearly describing the columns in each entity in a logical data model (LDM), which is a detailed version of a conceptual ERD. An Entity Relationship Diagram (ERD) is a visual illustration of the data that a database can store. Entity relationship diagrams (ER Diagrams), which are representations of the data structures in a table for the app database, are used to document software systems.

**Fig 3.6 Physical Data Model**

The physical design of a relational database is represented by a Physical Data Model (PDM), which is a more thorough version of a Logical ERD. The logical data model is expanded upon by a physical data model, which gives each column a type, length, nullable, etc.



**Fig 3.7 Chen Notation Data Model**

In database and software design, Chen notation is a well-liked standard for Entity Relationship Diagrams (ERDs). Dr. Chen's proposed entity-relationship model is built on the basis of a natural understanding of how the real world is made up of entities and the relationships that exist with these defined entities.