**System Design**

**3.1 Introduction:**

System design is a primary phase of the software development. System design aims to identify the modules that should be in the system, specification of these modules and how they interact with each other the desired results. The goal of the design process is to produce a module or representation of a system which can be used later to build that system. It is a plan for the solution of the system. Design includes requirement specification and final solution for satisfying the requirements. The system design attention is given to what components can be implemented in the software is considered.

**3.2 Assumption and Constraints**

The system should be able to upload the files in the server. This web portal is user-friendly

and less cost. This is aimed to provide satisfactory result to the user by providing the required information. Administrator is created in the system manually. All classes of users needed to enter their login name and password for authentication during login.

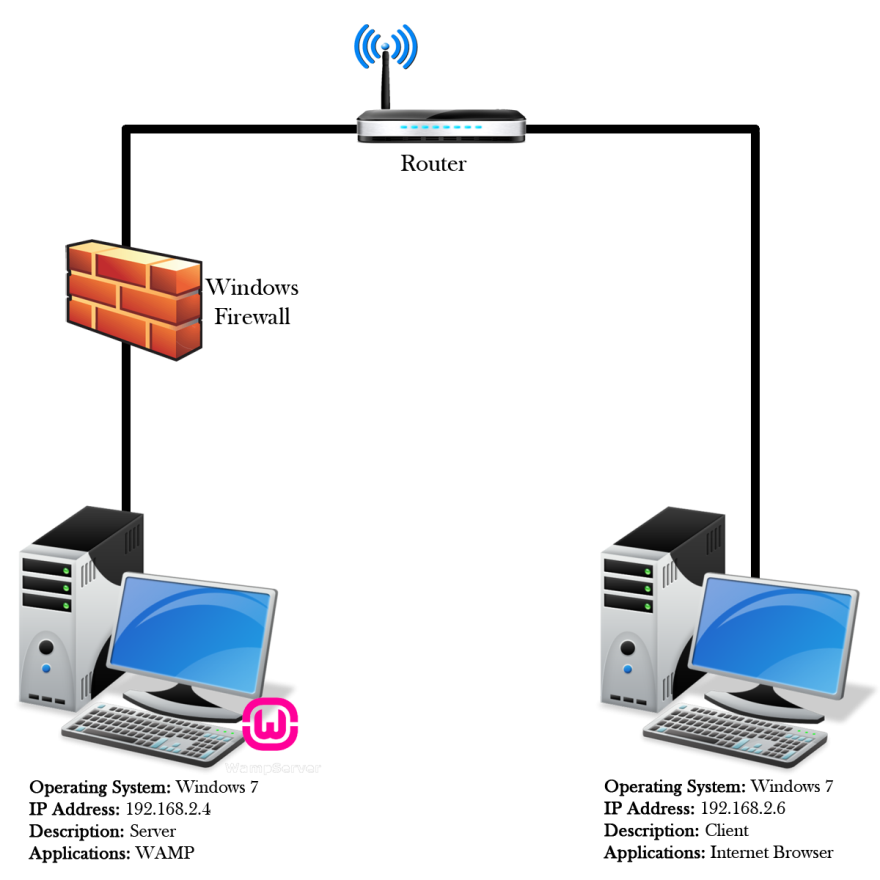
**System has following general constraints:**

* This system is provisioned to be built on the core PHP which is highly flexible.
* More space is required to keep all the record. Database should not be overloaded.

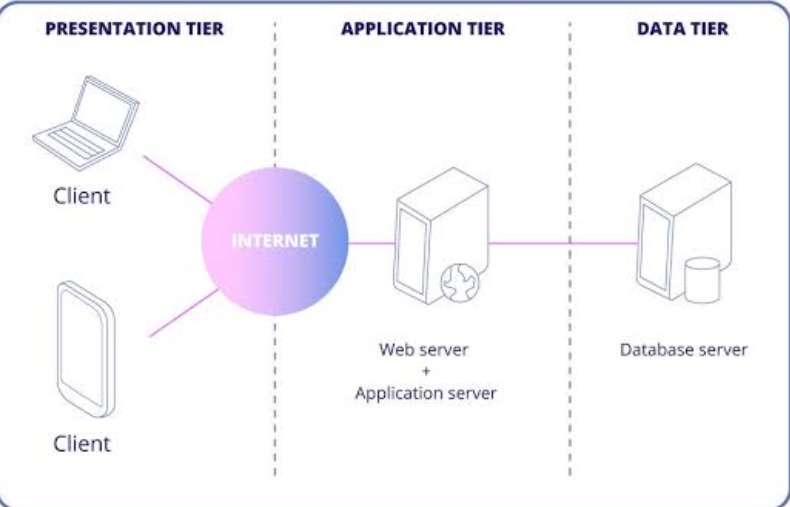
**3.3 Functional Decomposition:**

Functional decomposition is a technique used to break down a complex system into smaller, more manageable functions or modules. In the context of a hospital management system, functional decomposition helps identify the key functionalities or components that make up the system.

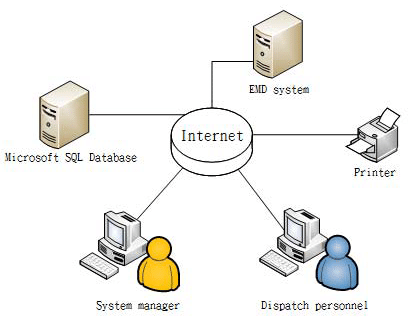
**3.3.1 System Software Architecture:**

****

**3.3.2 System Technical Architecture:**

****

**3.3.3 System Hardware Architecture:**

**3.3.4 External interfaces**

* **Name of the application:**

HOSPITO

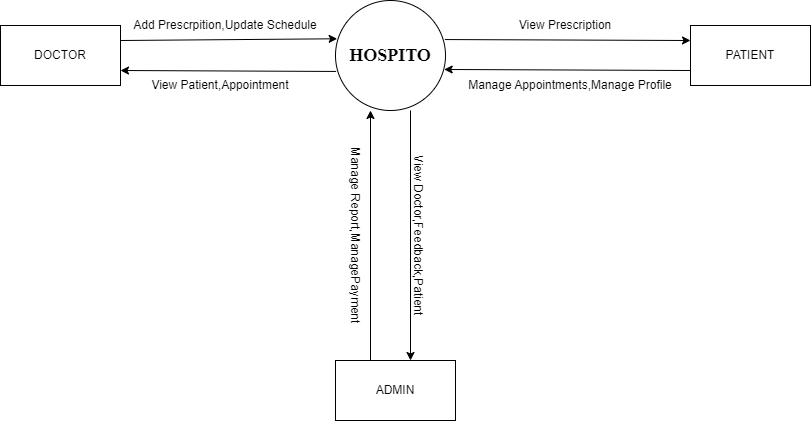
* **Type of Interface:**

Graphical User Interface, Menu Driven.

**3.4 Description of Programs**

**3.4.1 Context Flow Diagram (CFD)**

Context flow diagram is a top-level data flow diagram. It only contains one process node that generalizes the function of the entire system in relationship to external entities. In context diagram the entire system is treated as a single process and all its inputs, outputs, sinks and sources are identified and shown



**3.4.2 Data Flow Diagram (DFDs – Level 0, Level 1, Level 2)**

Data Flow Diagram is a graphical representation of a system or a portion of the system. It

consists of data flows, process, sources and sink and stores all the description through the use of easily understandable symbols. DFD is one of the most important modelling tools. It is used to model the system, components that interact with the system, uses the data and information flows in the system. DFD shows the information moves through the and how it is modified by a series of transformations. It is a graphical technique that depicts information moves from input or output. DFD is also knows as bubble chart or Data Flow Graphs. DFD may be used to represent the system at any level of abstraction. DFD’s may partition into a level that represents increasing information flows and functional details.

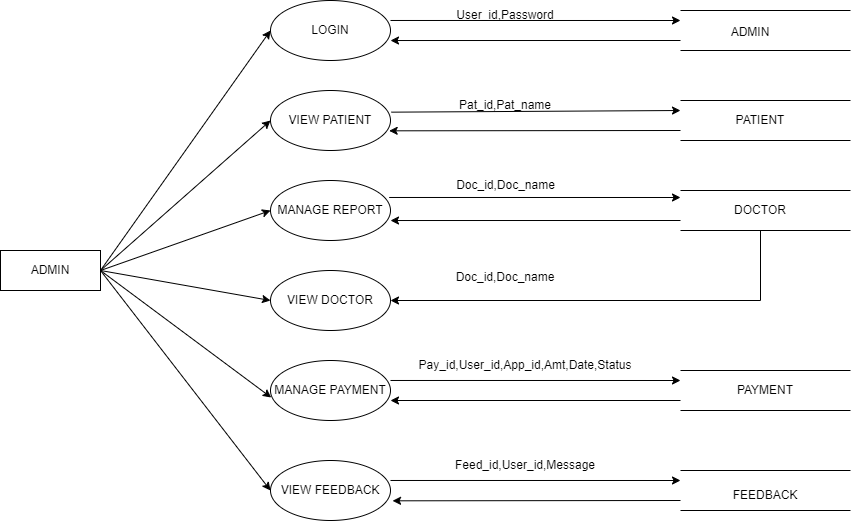
**Rules Regarding DFD Construction:**

* A process cannot have only outputs.
* A process cannot have only inputs.
* The inputs to a process must be sufficient to produce the outputs from the process.
* All data stores must be connected to at least one process.
* All data stores must be connected to a source or sink.
* A data flow can have only one direction of flow. Multiple data flows to and/or from the same process and data store must be shown by separate arrows.
* If the exact same data flows to two separate arrows, it should be represented by a forked arrow.
* Data cannot flow directly back into the process it has just left. All data flows

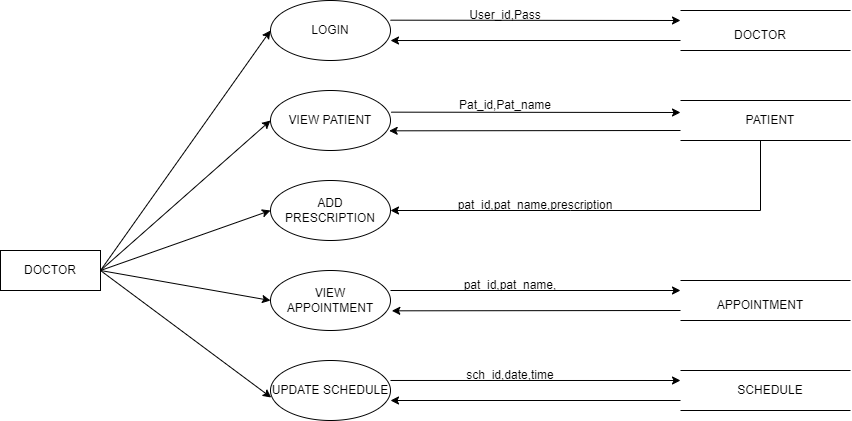
must be named using a noun phrase.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Name** | **Description** |
|  | **Start Process** | Circles are used to represent processes. Processes are actions taking place to transform inputs to outputs. In a context diagram, a system is represented by a single labeled circle. In a data flow diagram, multiple circles represent multiple processes within the system. |
|  | **Data Flow** | **Arrows represent data flows between processes, data stores and external entities. Data flows should be named to identify the piece of data.** |
|  | **External Entity** | **Boxes are used to represent external entities. These are any items, person or organization sitting outside the systems that provides data to the system or received data from the system.** |

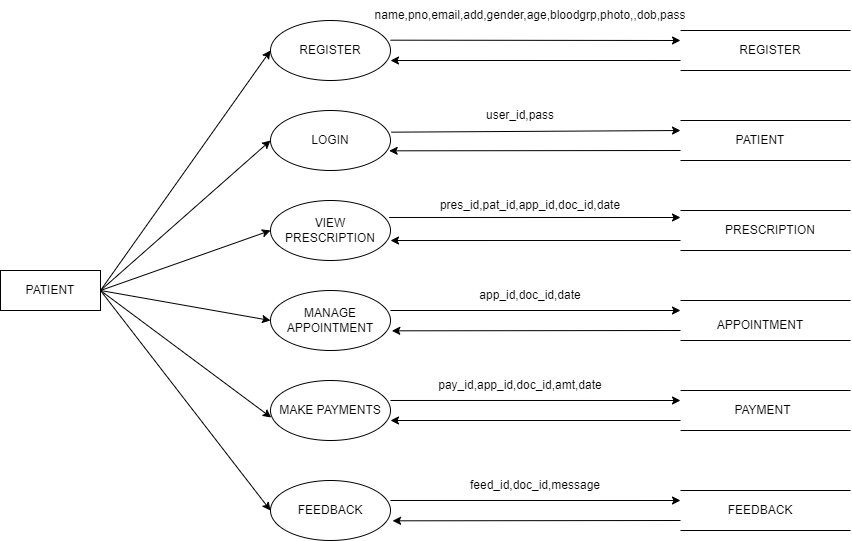
**DFD LEVEL 1(ADMIN):**

****

**DFD LEVEL 2(DOCTOR):**

****

**DFD LEVEL 3(PATIENT):**

****

**Description of Components :**

**Functional Component 1: Admin**

**Introduction:** This module is used to manage the details of Doctors and patients. Also, he can manage and modify the features of the application.

**Login:**

* Input: admin\_id, username, password.
* Process: Entered username and password will be checked for validity. If it is valid admin will be directed to homepage.
* Output: Admin views the homepage.

**View Patient:**

* Input: pat\_id, pat\_name
* Process: The details of the patients will be added to the database. Admin can view the details of all patients registered through the website.
* Output: Admin can view patient’s details.

**Manage Doctors:**

* Input: doc\_id, doc\_name
* Process: The details of the doctor will be added to the database. Admin can check for doctor’s update and performance registered through the website.
* Output: Admin can view doctor’s details.

**Manage Appointments:**

* Input: pat\_id,ap\_id
* Process: The details of the appointments will be managed in the database. Admin can check for doctor’s update and performance registered through the website.
* Output: Admin can accept or reject the appointment.

**View Feedback:**

* Input: feed\_id
* Process: The details of the feedback will be added to the database.
* Output: Admin can view patient’s feedback.

**Manage Payments:**

* Input: pat\_id, pay\_id
* Process: The details of the payments will be added to the database.
* Output: Admin can manage payment details.

**Functional Component 2: Doctor**

**Introduction:** In this module doctor can login by entering username and password. The doctors can view appointments, update schedule and add prescription details.

**Login:**

* Input: doc\_id, username,password
* Process: Entered username and password will be checked for validity. If it is valid admin will be directed to homepage.
* Output: Doctor views the homepage.

**View Patient:**

* Input: pat\_id, pat\_name
* Process: The details of the patients will be added to the database. Doctor can view the details of all patients registered through the website.
* Output: Doctor can view patient’s appointments and reports details.

**View Appointments:**

* Input: pat\_id,ap\_id
* Process: The details of the appointments will be managed in the database. Doctor can check for appointment updates registered through the website.
* Output: Doctor can View appointment.

**Add Prescription Details:**

* Input: pr\_id,pat\_id
* Process: The prescription details will be added to the database.
* Output: Doctor can add prescription details.

**Functional Component 3: Patient**

**Introduction:** In this module patient can register, login, take appointments, cancel appointment, view reports and prescription.

**Register:**

* Input: pat\_name, pat\_phone,pat\_email, pat\_password, pat\_date.
* Process: Entered details are stored in patient database. If all details are proper then he is directed to homepage.
* Output: Patient can view homepage.

**Login:**

* Input: pat\_id, pat\_name, pat\_password.
* Process: Entered details are stored in patient database. If it is valid then patient will be directed to homepage.
* Output: Patient can view the homepage.

**View Prescription:**

* Input: pat\_id, pat\_name, doc\_id.
* Process: Entered details will be verified. If it is valid patient will be directed to prescription page.
* Output: Patient views the prescription.

**Manage Appointments:**

* Input: pat\_id,pat\_name, doc\_id, age, phoneno, gender, ap\_date,ap\_time, problem, amount.
* Process: The details of the appointments will be stored in the database.
* Output: Patient can manage the appointment.

**Payment:**

* Input: pat\_id,pat\_name, ap\_id, amount.
* Process: Entered details will be verified. If it is valid patient will be directed to payment page.
* Output: Patient is directed to payment gateway.

**Send Feedback:**

* Input: feed\_id, pat\_id, doc\_id.
* Process: Entered details will be verified. If it is valid patient will be directed to feedback page.
* Output: Patient can add feedback.