**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.

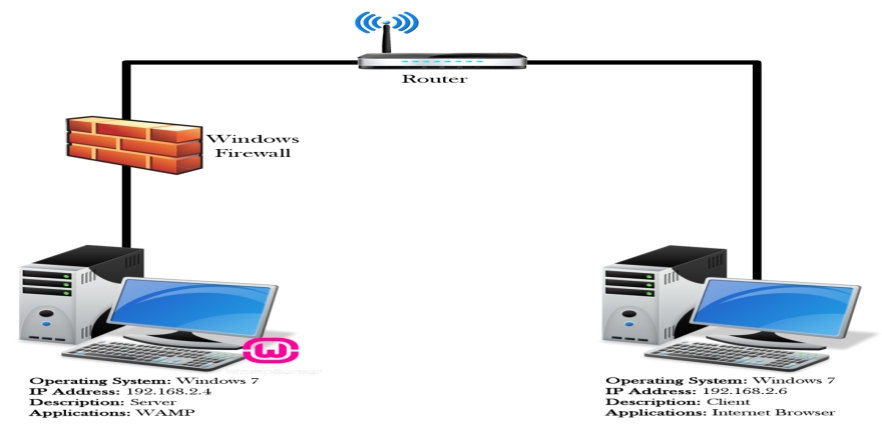
**System has following general constraints:**

* 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.

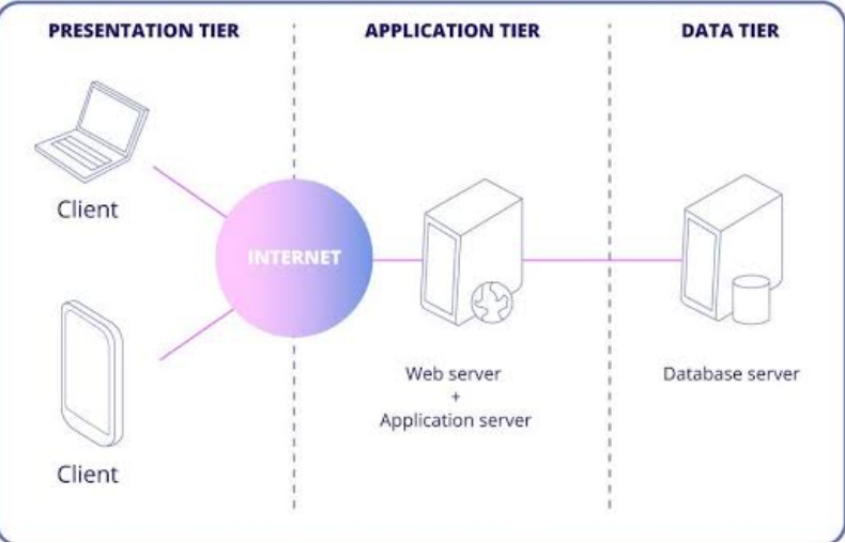
**3.3 Functional Decomposition:**

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.

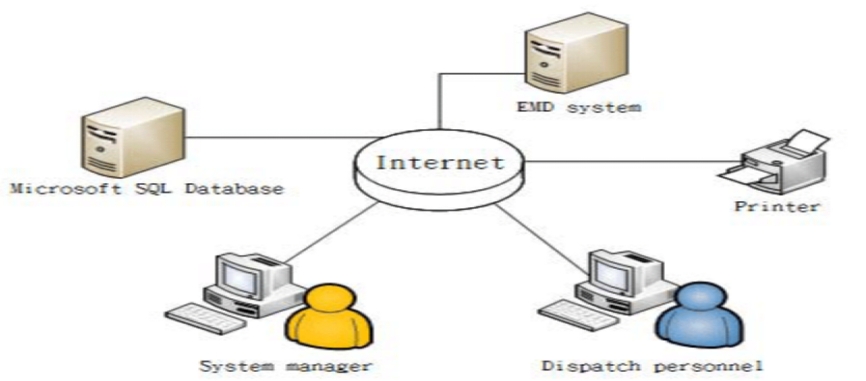
**3.3.1 System Software Architecture:**



**3.3.2 System Technical Architecture:**



**3.3.3 System Hardware Architecture:**

**3.3.4 External interfaces**

* **Name of the application:**

HOSPITO

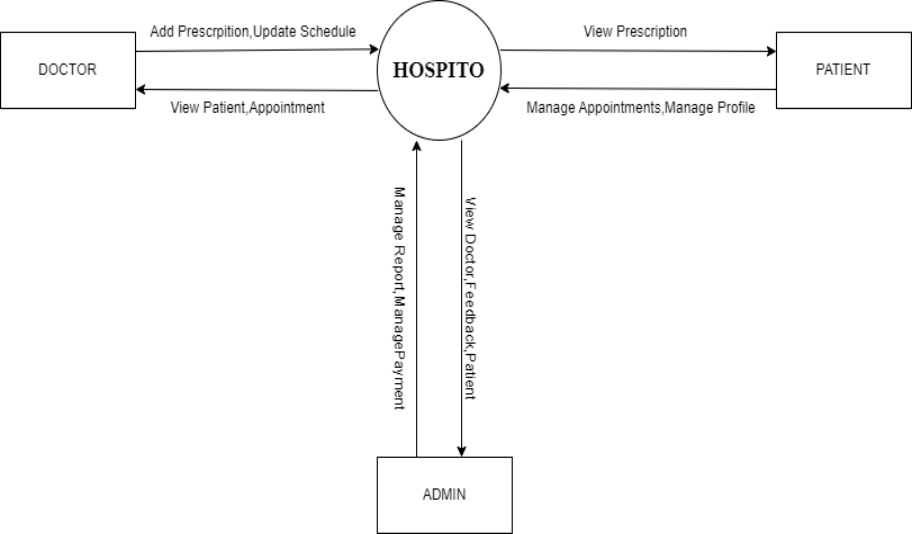
* **Type of Interface:**

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 modeling 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.

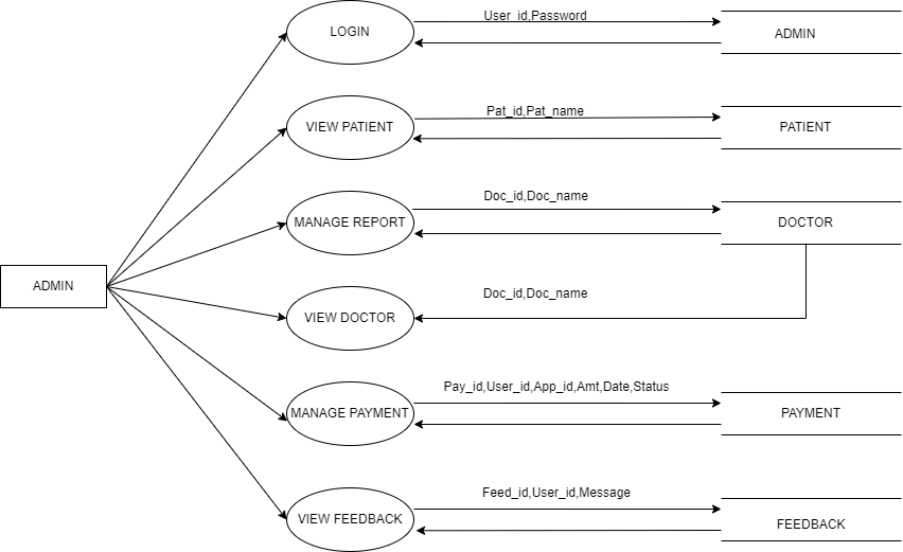
**Rules Regarding DFD Construction:**

* 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.

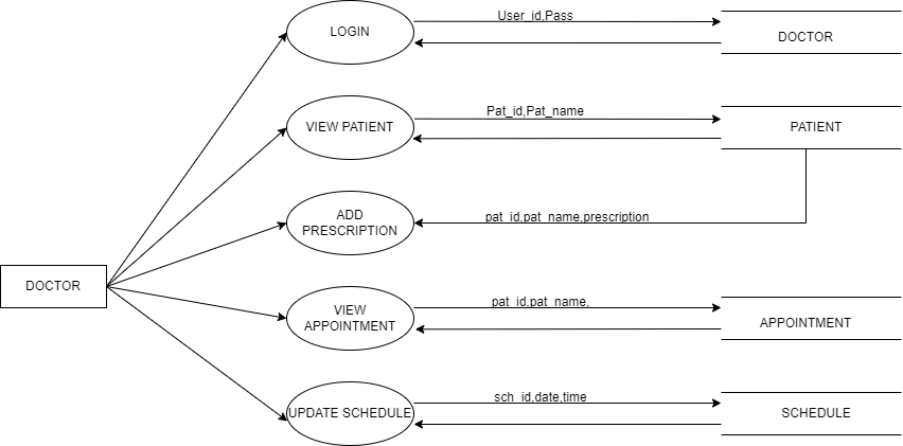
|  |  |  |
| --- | --- | --- |
| **Symbol** | **Name** | **Description** |
| **C:\Users\asus\AppData\Local\Temp\ksohtml12352\wps5.png** | **Start Process** | **Circles are used to represent processes. Processes are actions taking place to transform inputs to outputs. In a context diagram, a system is represented by a single labeled circle. In a data flow diagram, multiple circles represent multiple processes within the system.** |
| **C:\Users\asus\AppData\Local\Temp\ksohtml12352\wps6.png**  **C:\Users\asus\AppData\Local\Temp\ksohtml12352\wps7.png** | **Data Flow** | **Arrows represent data flows between processes, data stores and external entities. Data flows should be named to identify the piece of data.** |
| **C:\Users\asus\AppData\Local\Temp\ksohtml12352\wps8.png** | **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.** |

* Data cannot flow directly back into the process it has just left. All data flows must be named using a noun phrase.

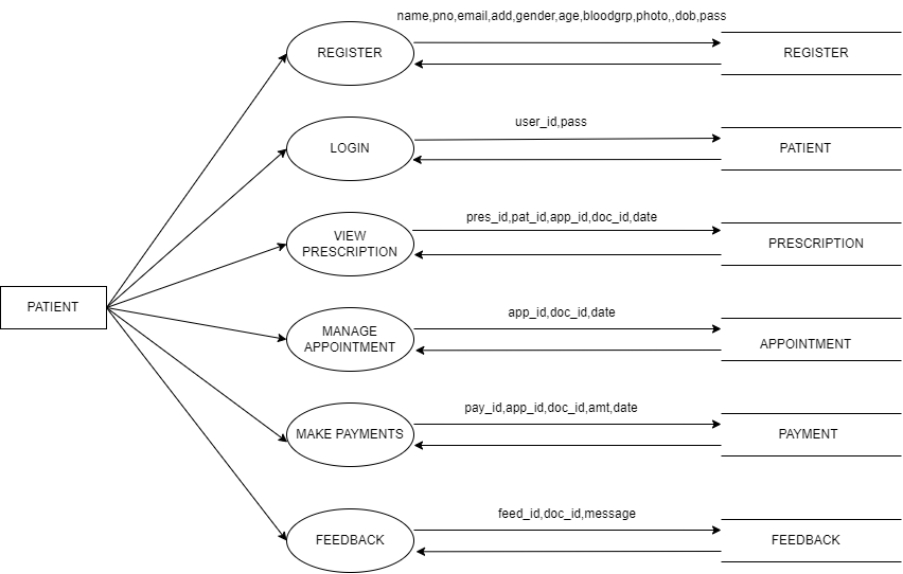
**DFD LEVEL 1(ADMIN):**



**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 Appointmets:**

* 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.

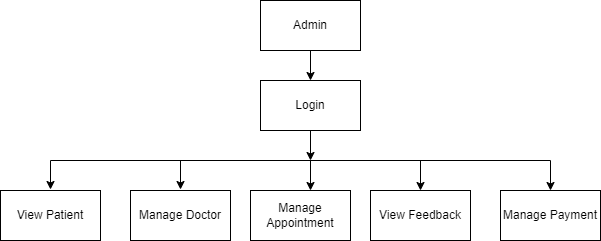
Detailed Design

**5.1: Introduction**

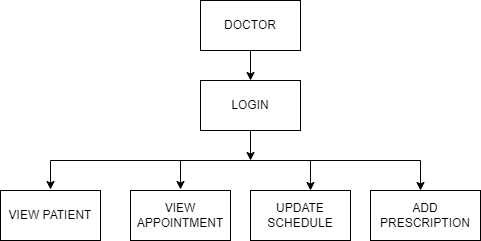
Detailed design, also known as low-level design, refers to the process of elaborating and specifying the components, modules, and interactions of a system or software solution. It involves translating the high-level design into a more detailed and implementation-oriented representation, providing instructions for developers to construct the system.

**5.2: Structure of Software Package**

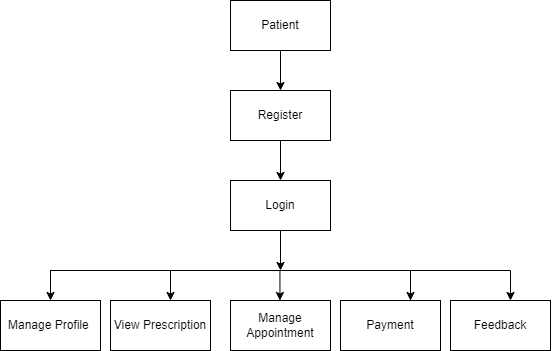
**• Structure of admin module**

****

**• Structure of doctor module**

****

**• Structure of patient module**

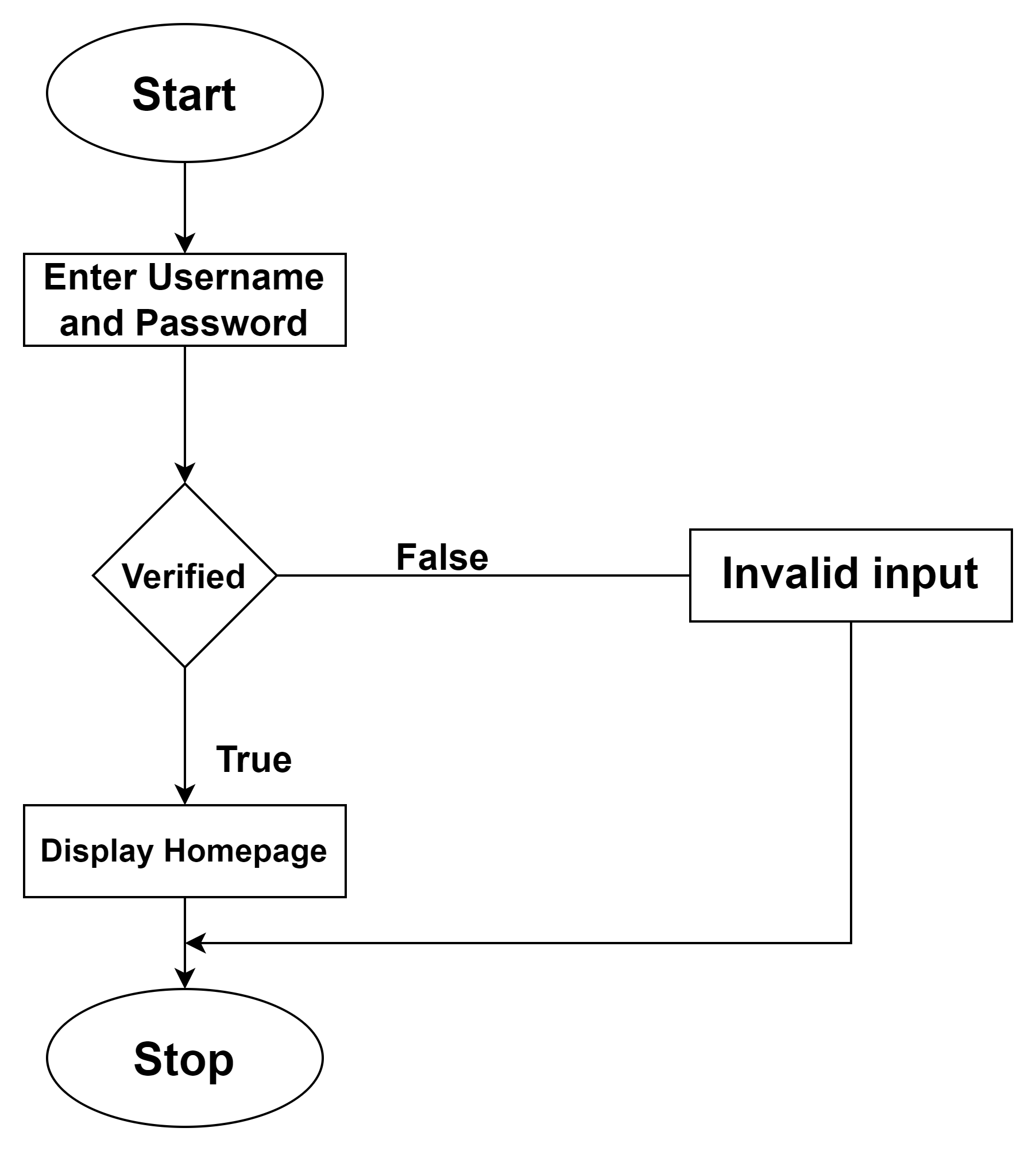
****

**5.3: Modular decomposition of the System**

**5.3.1:Module 1:Admin**

**Login**

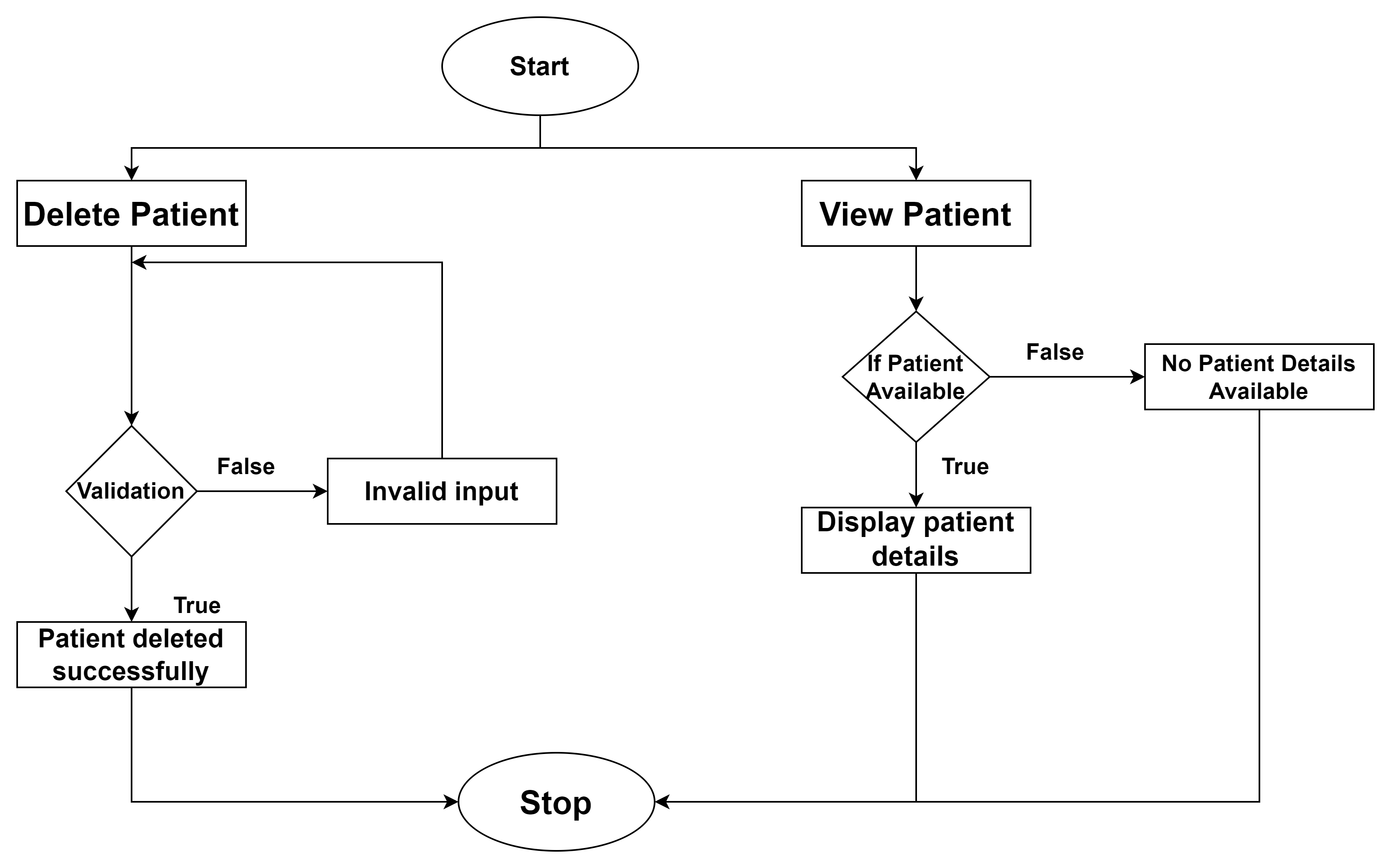
* **Input:username,password**
* **Procedural details:**



* **File I/O interfaces :**
* **Output:**

**View patient**

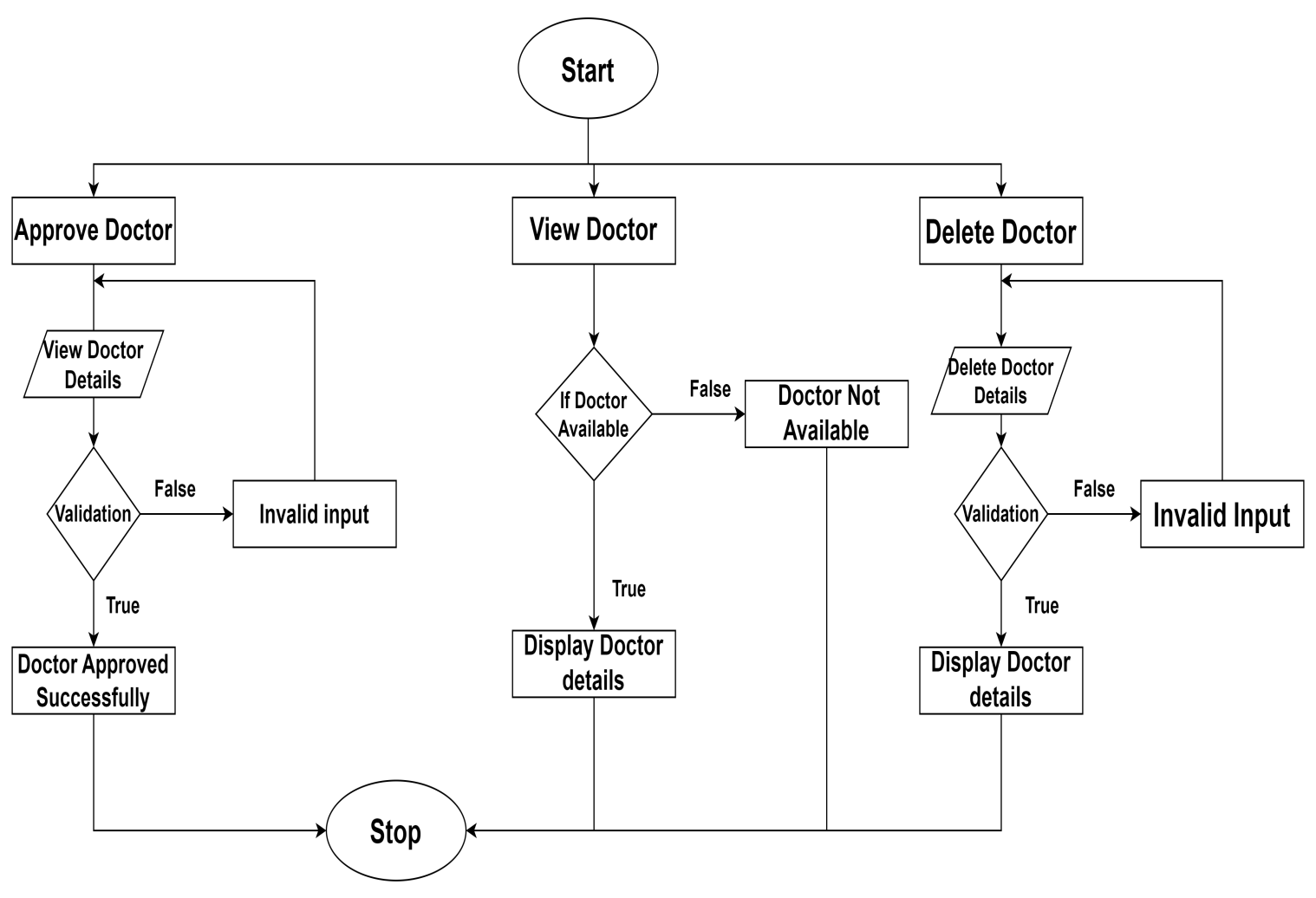
* **Input:**
* **Procedural details:**



* **File I/O interfaces :**
* **Output:**

**Manage Doctors**

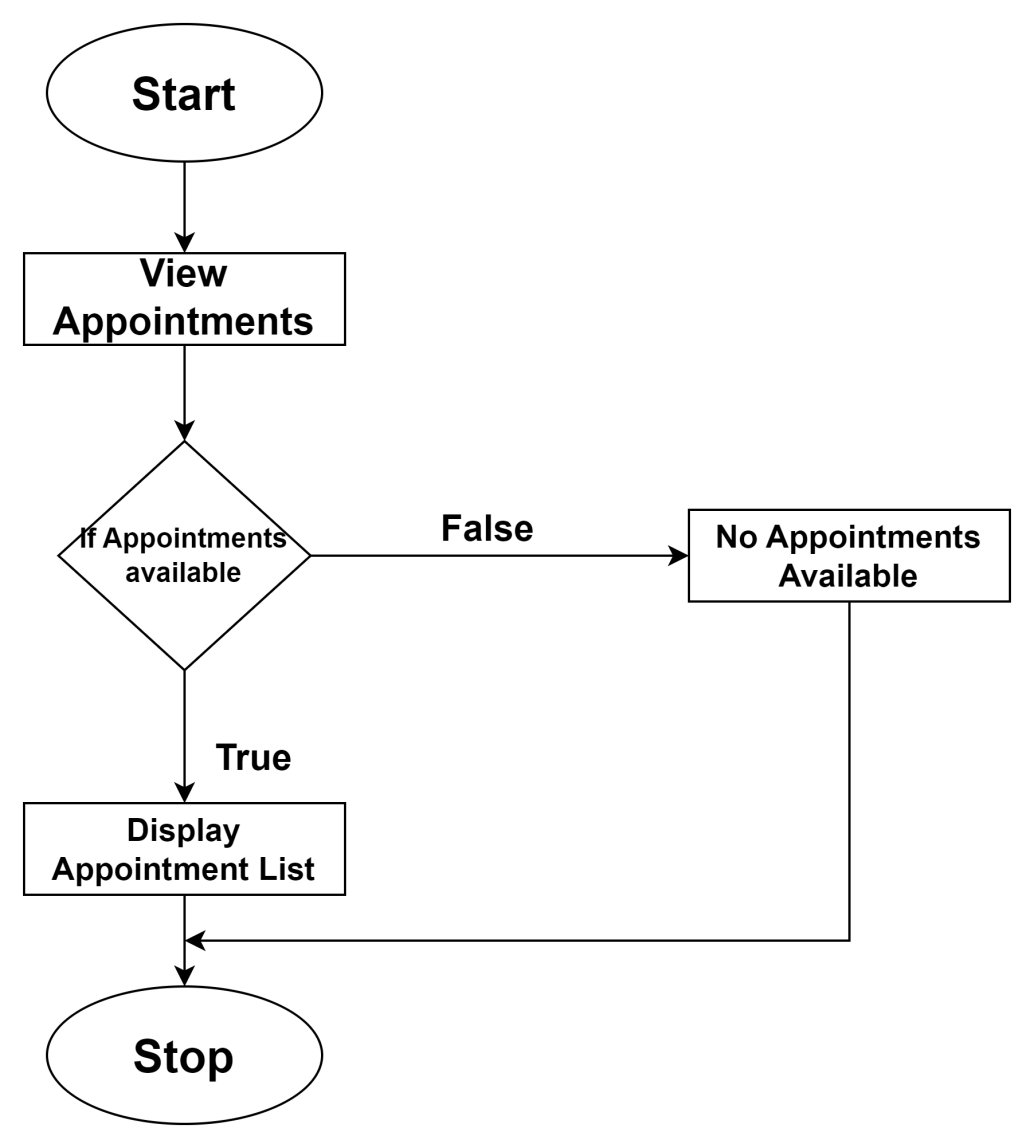
* **Input:**
* **Procedural details:**



* **File I/O interfaces :**
* **Output:**

**View Appointment**

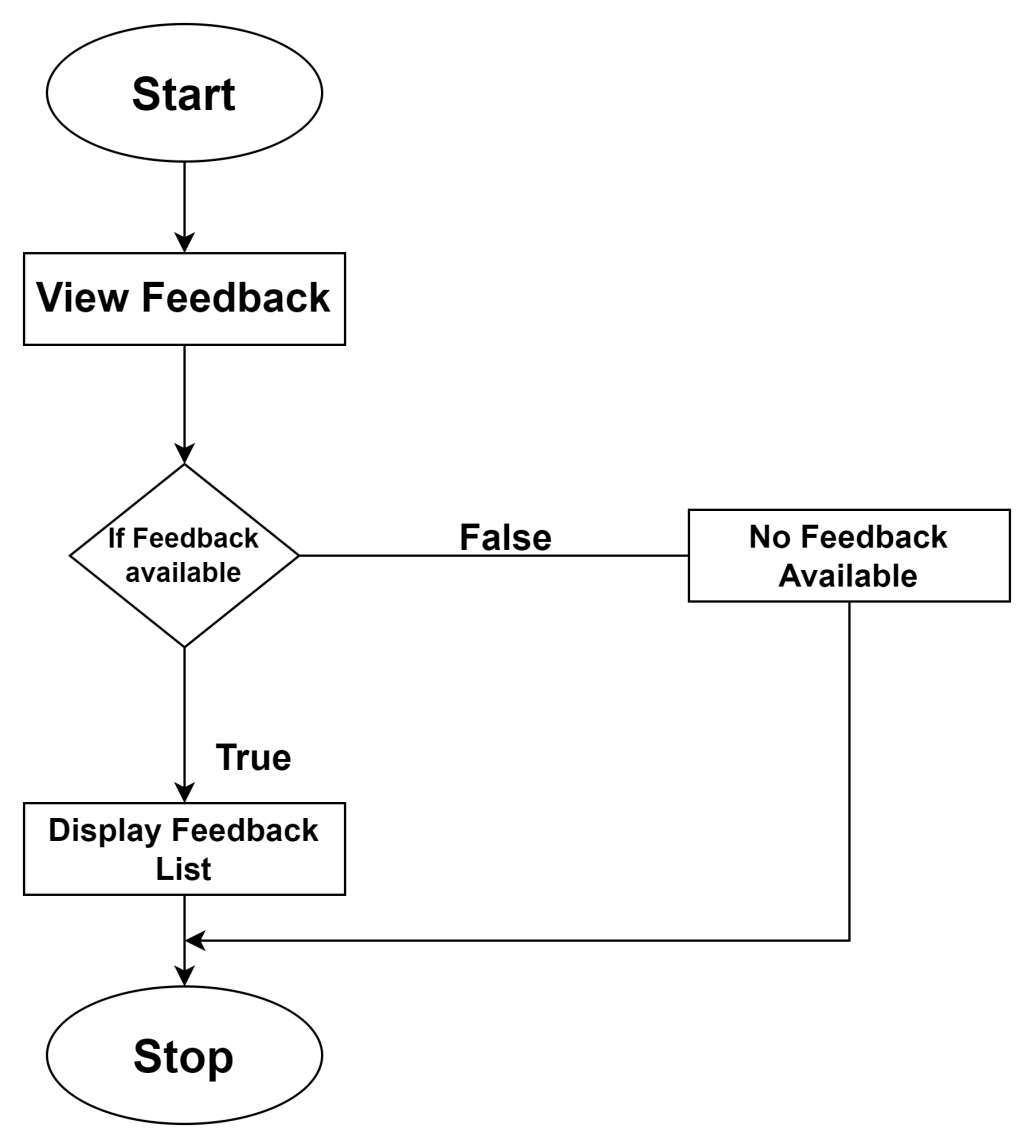
* **Input:**
* **Procedural details:**



* **File I/O interfaces :**
* **Output:**

**View Feedback**

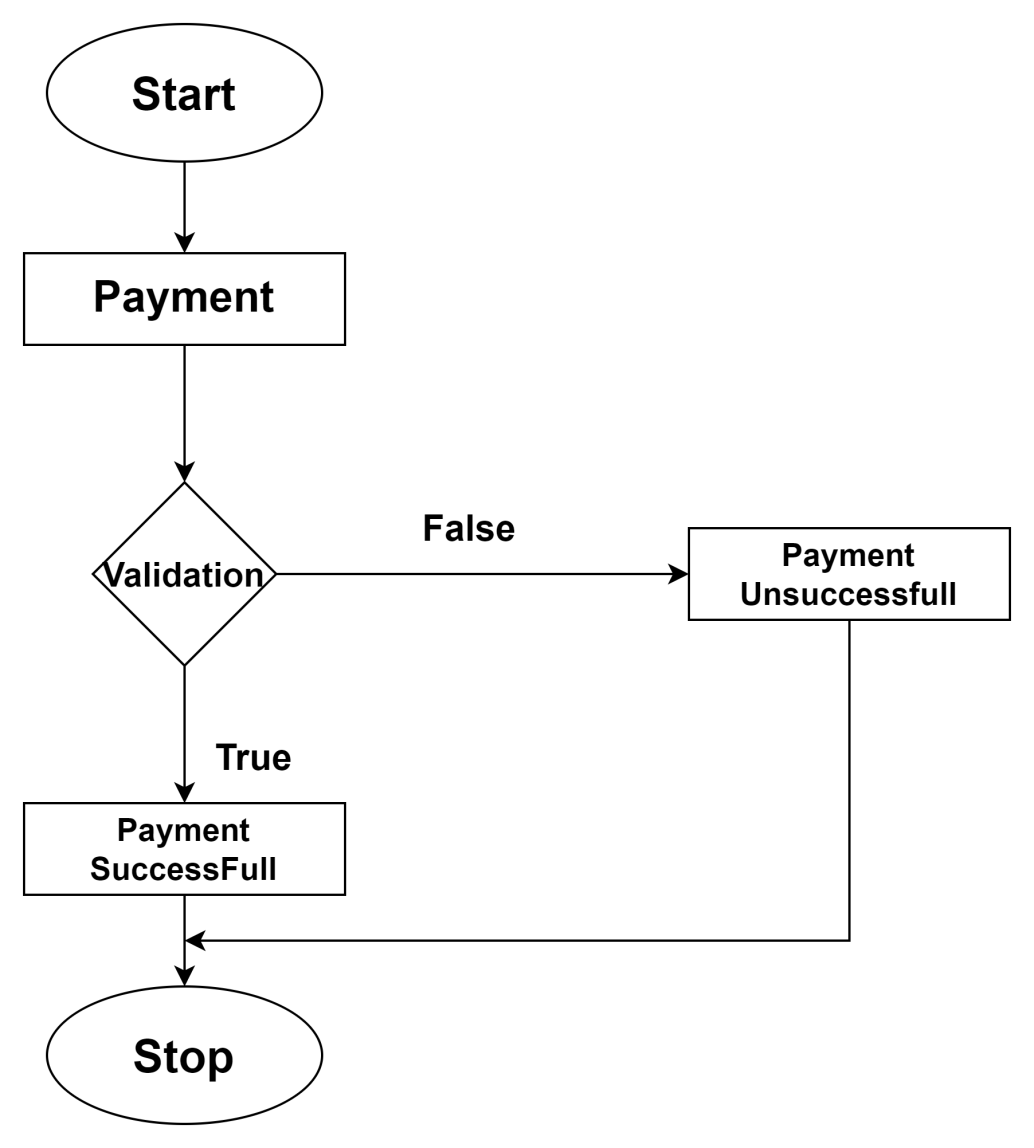
* **Input:**
* **Procedural details:**



* **File I/O interfaces :**
* **Output:**

**Manage payment:**

* **Input:**
* **Procedural details:**

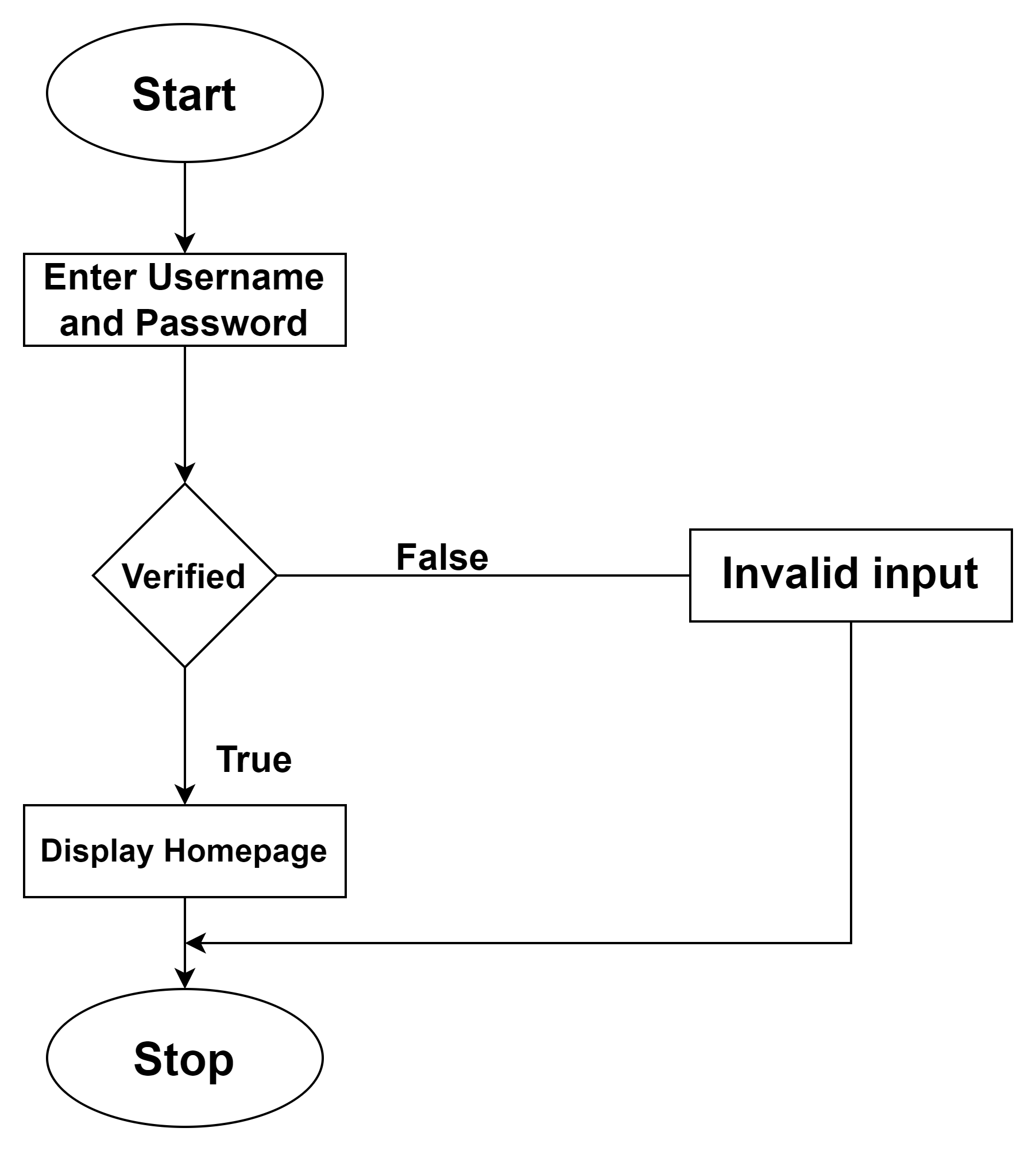


* **File I/O interfaces :**
* **Output:**

**5.3.2: Module 2 :Doctor**

**Login**

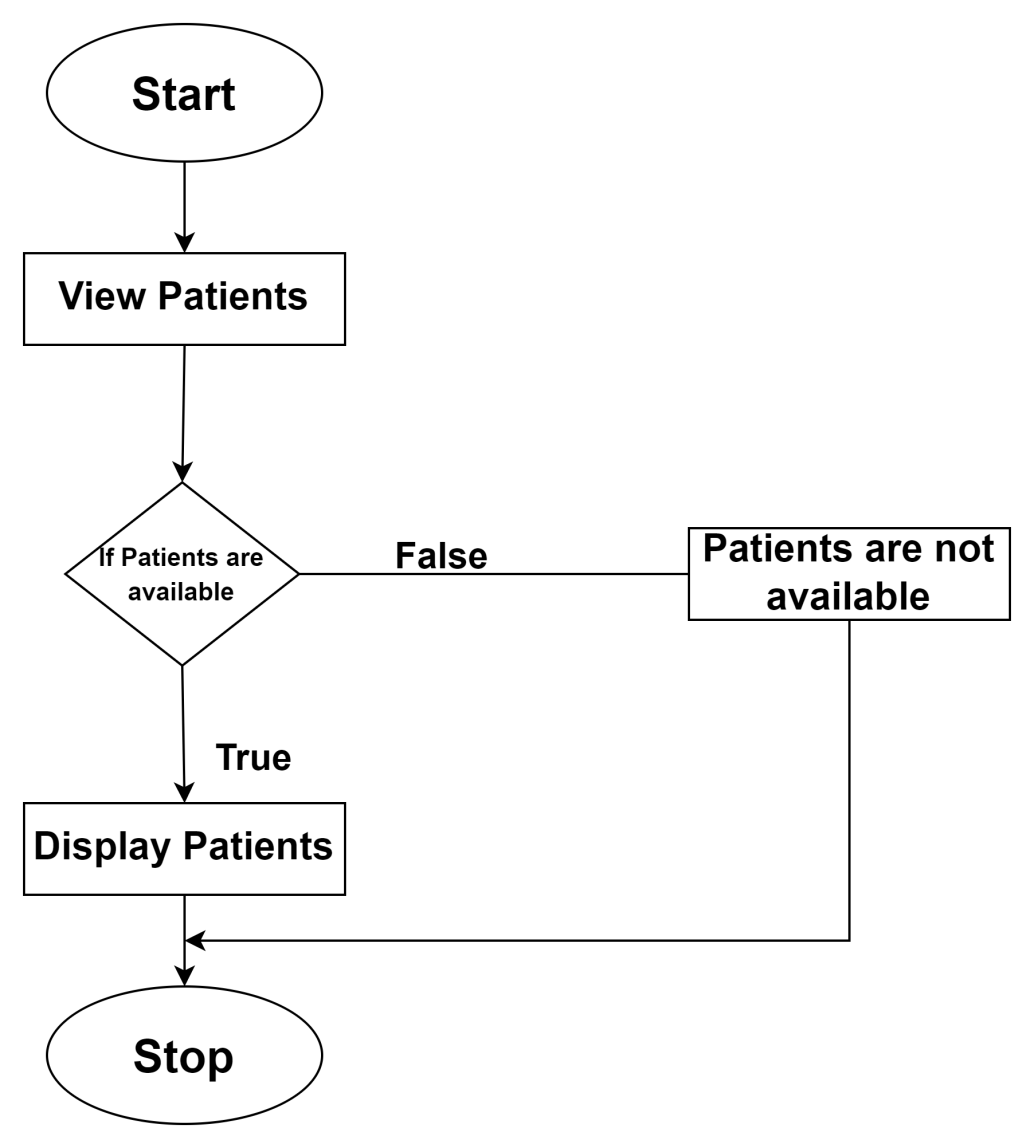
* **Inputs**
* **Procedural Details**



* **File I/O interfaces :**
* **Output:**

**View Patients**

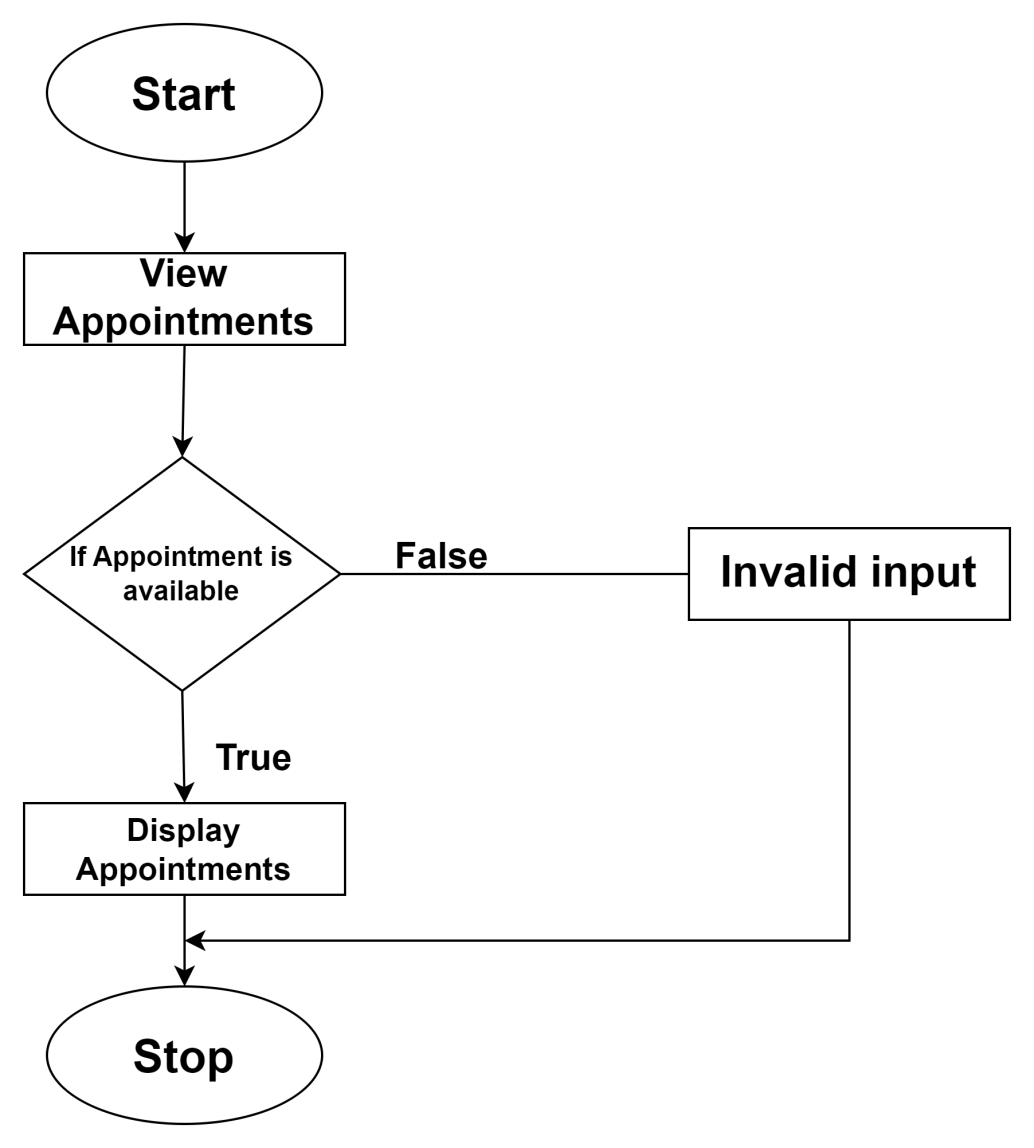
* **Inputs**
* **Procedural Details**



* **File I/O interfaces :**
* **Output:**

**View Appointments:**

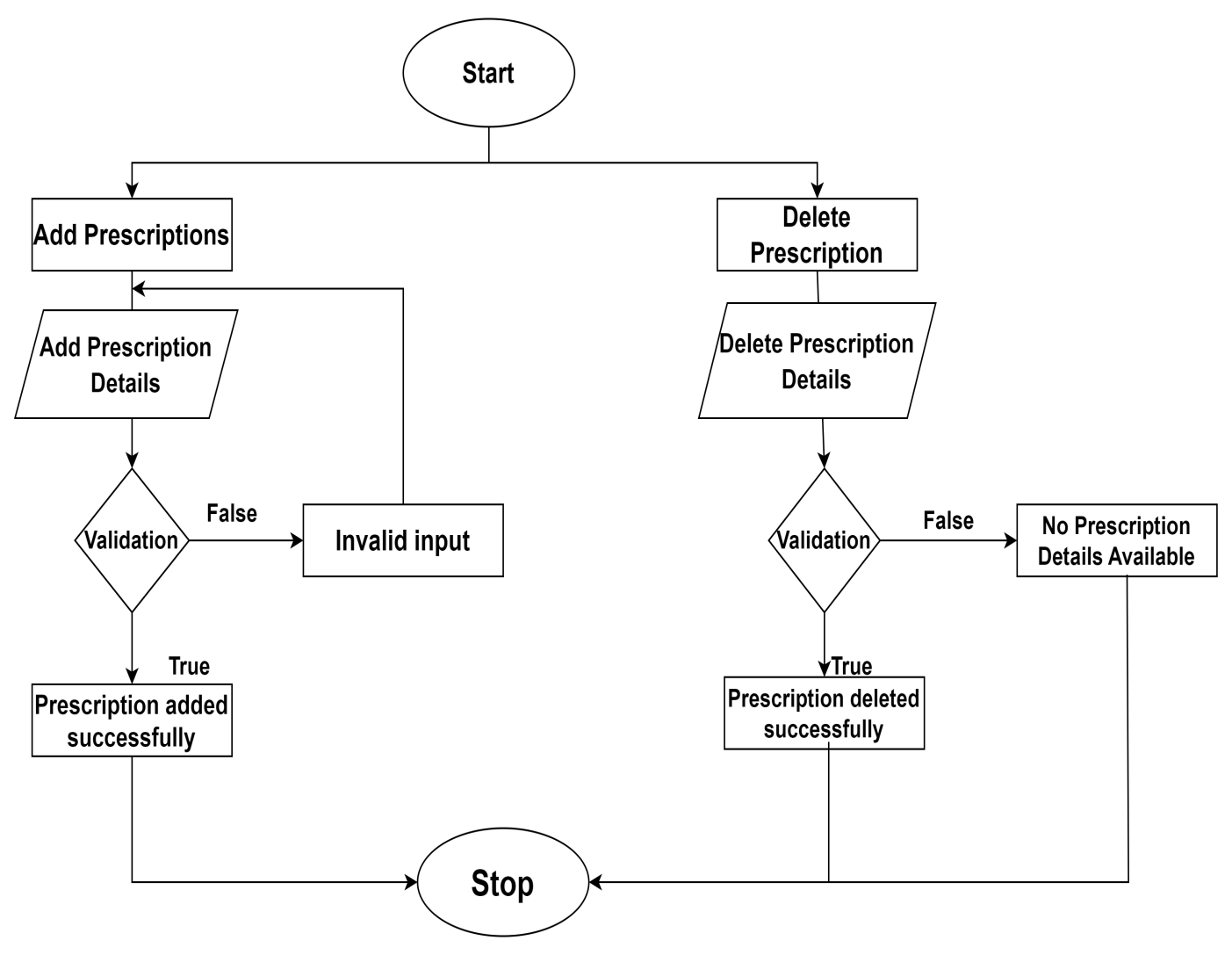
* **Inputs**
* **Procedural Details**



* **File I/O interfaces :**
* **Output:**

**Add Prescription Details**

* **Inputs**
* **Procedural Details**

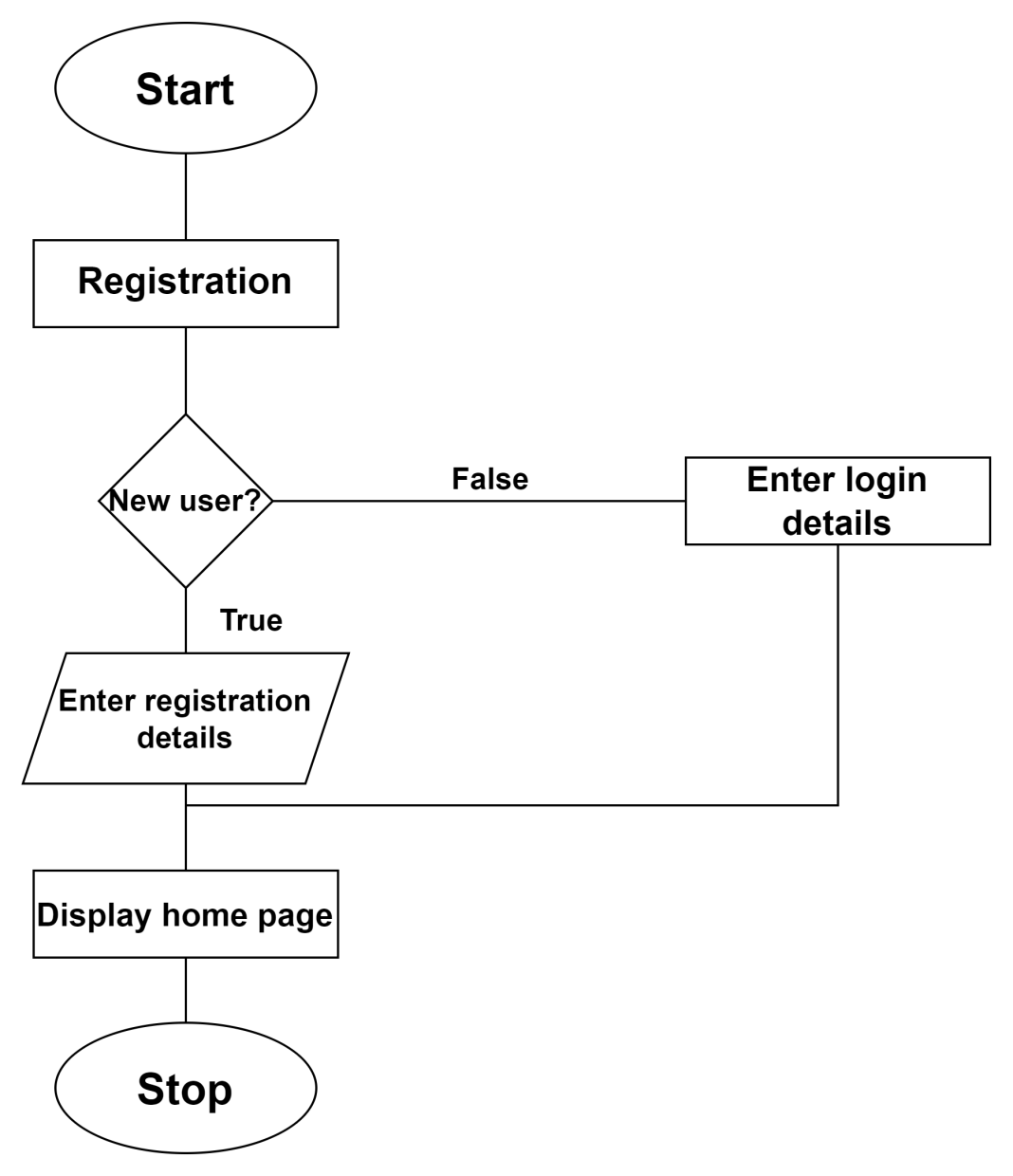


* **File I/O interfaces :**
* **Output:**

**5.3.3: Module 3:Patient**

**Registration**

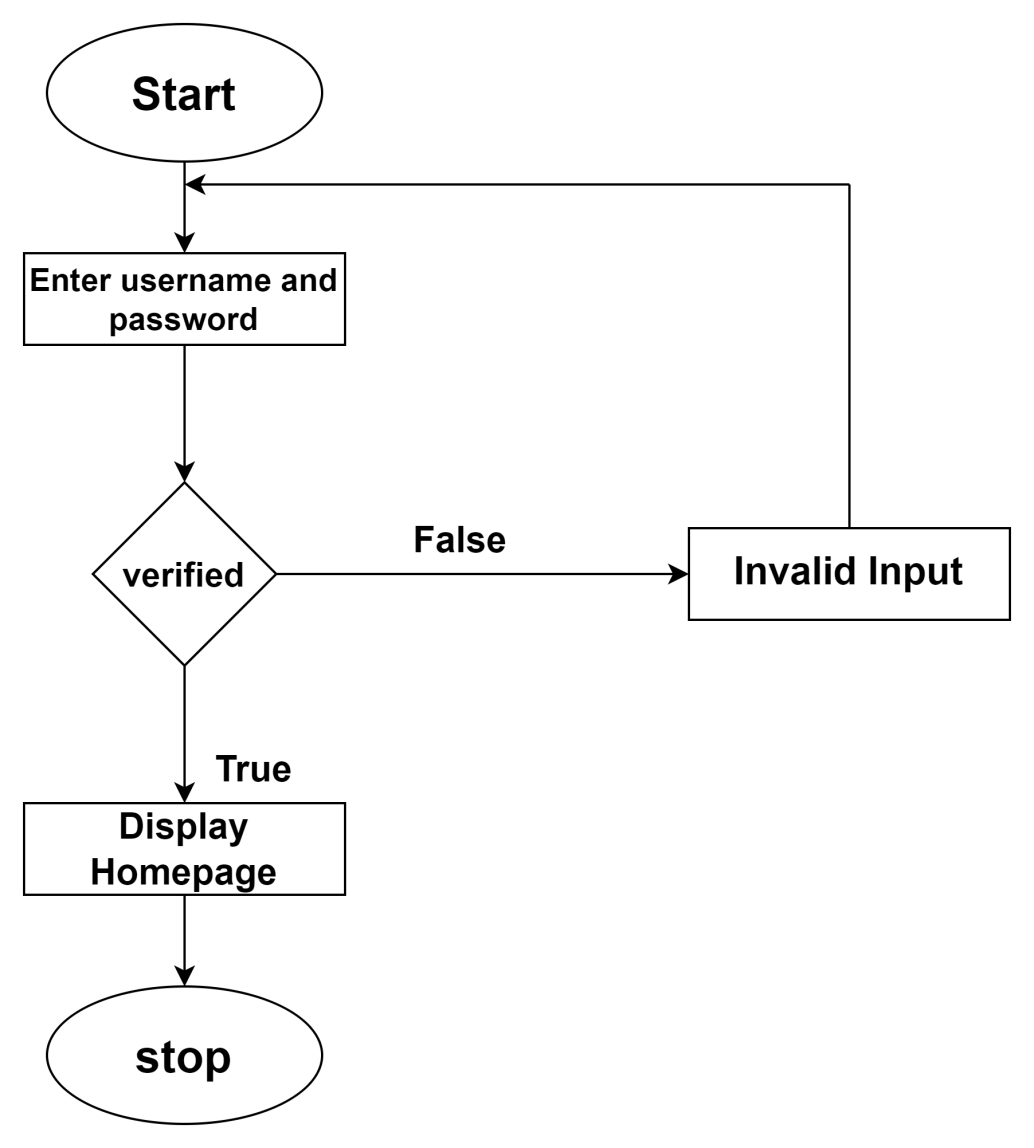
* **Inputs**
* **Procedural Details**



* **File I/O interfaces :**
* **Output:**

**Login**

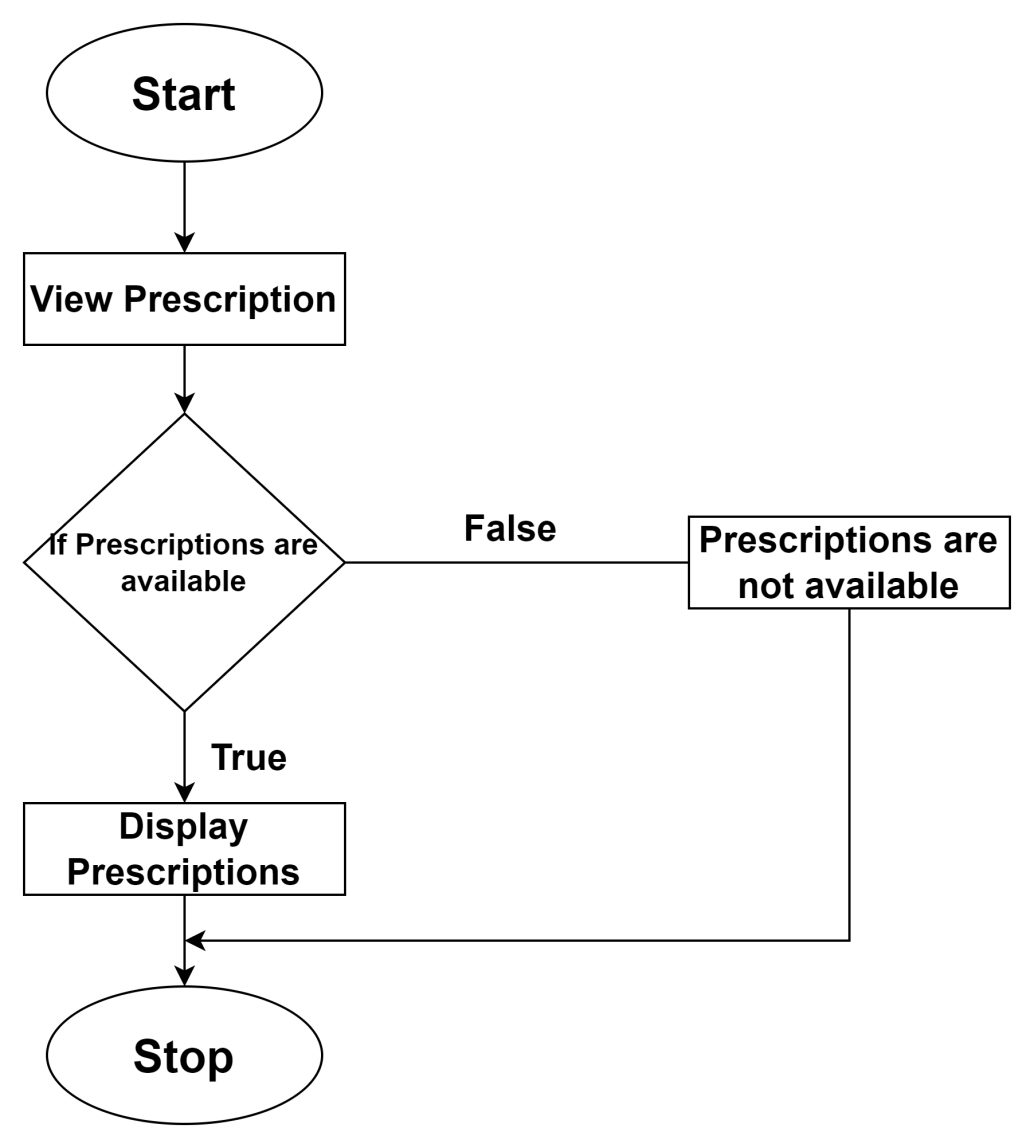
* **Inputs :**
* **Procedural Details :**



* **File I/O interfaces :**
* **Output:**

**View Prescription**

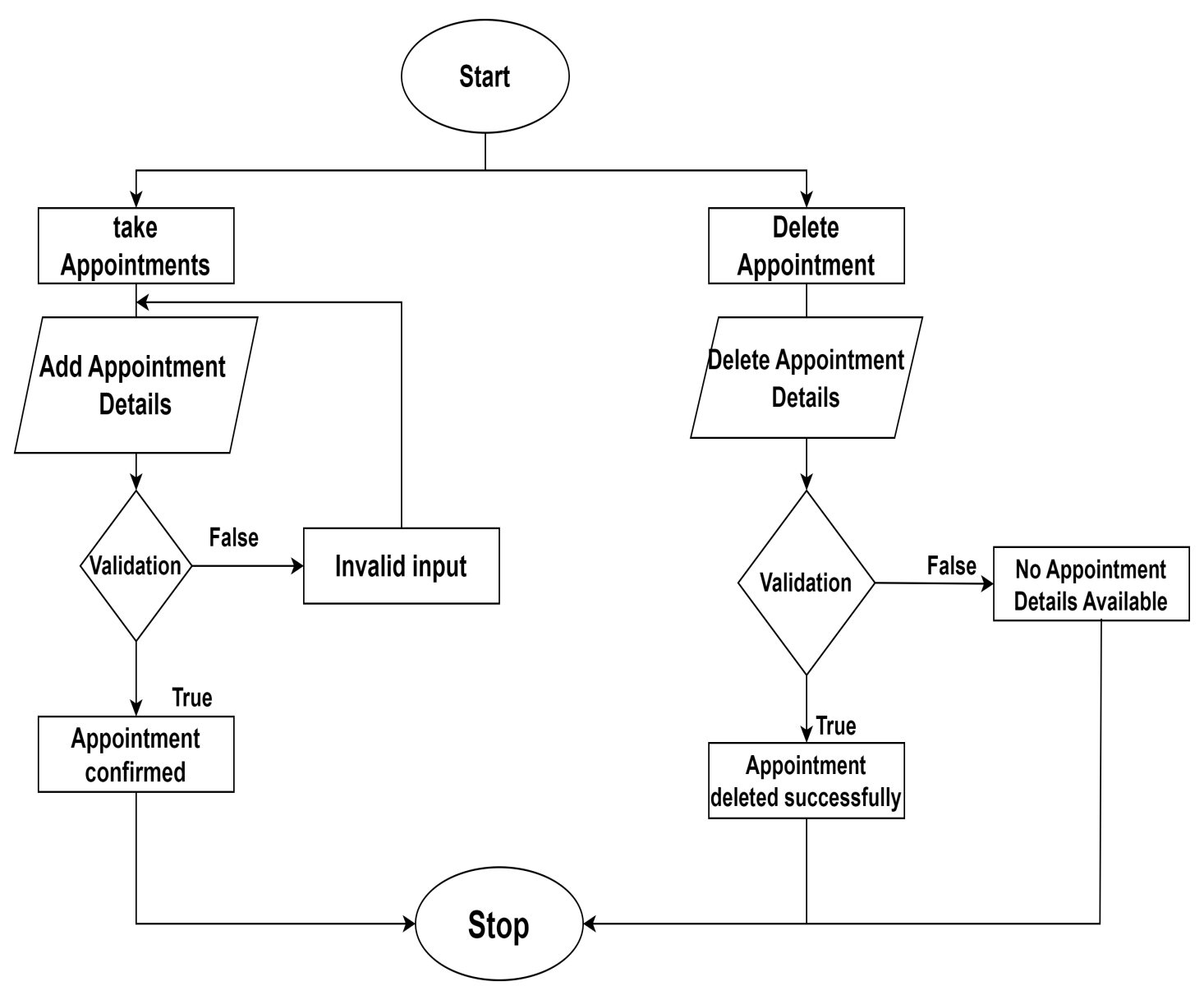
* **Inputs :**
* **Procedural Details :**



* **File I/O interfaces :**
* **Output:**

**Manage Appointments**

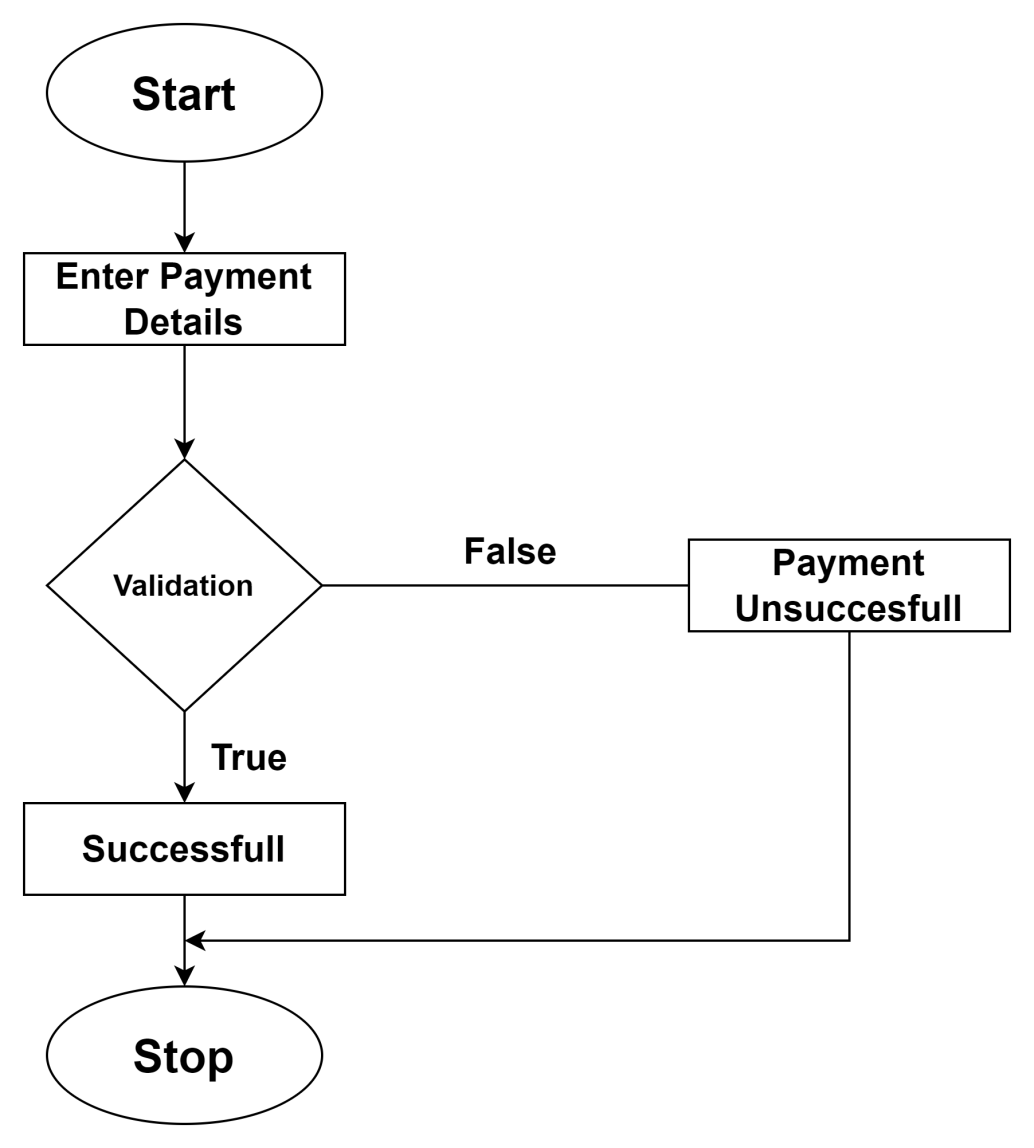
* **Inputs :**
* **Procedural Details :**



* **File I/O interfaces :**
* **Output:**

**Payments**

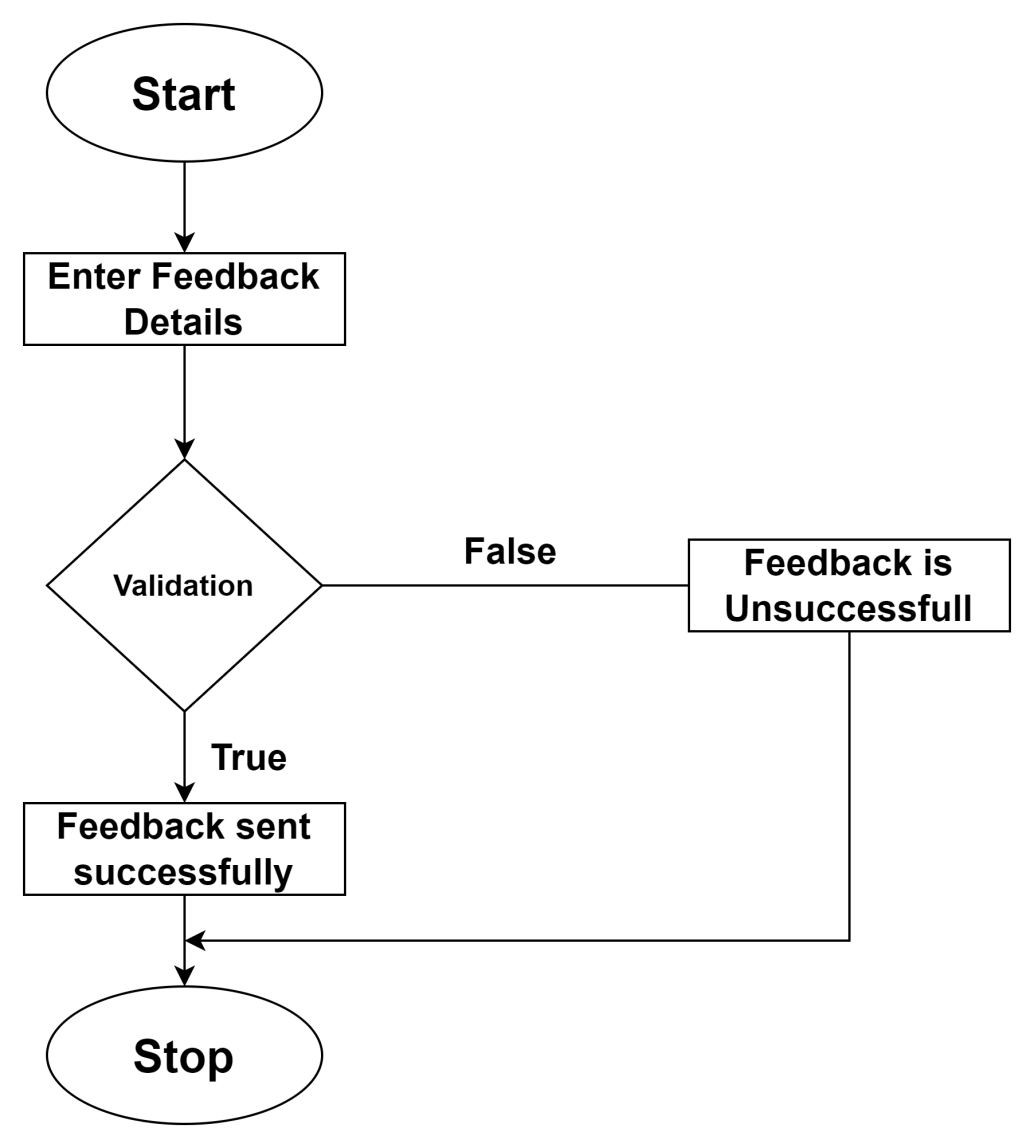
* **Inputs :**
* **Procedural Details :**



* **File I/O interfaces :**
* **Output:**

**Send Feedback**

* **Inputs :**
* **Procedural Details :**



* **File I/O interfaces :**
* **Output:**

DataBase Design

**4.1: Introduction**

Database design is used to manage large bodies of information. The management of data involves both the definition of structure of storage and provision for the manipulation of information. In addition, the database system must provide the safety of the information solved despite system crashes or due to the attempts at unauthorized access. We have to fulfil certain conditions such as:

* Control Redundancy.
* Easy to use.
* Data Independence.
* Accuracy and Integrity.
* Recovery and Security.
* Performance.

**4.2: Purpose and Scope**

The purpose of database design is to create a structured and efficient system for organizing and storing data. It involves defining the logical and physical structure of a database, determining the relationships between data elements, and establishing rules to ensure data integrity and security. The ultimate goal is to provide a solid foundation for data management, retrieval, and manipulation.

**4.3: DataBase Identification**

Database identification refers to the process of determining the need for a database in a particular context or scenario. It involves recognizing the data requirements, the purpose of data storage, and the potential benefits of using a database management system (DBMS) to organize and manage that data.

**4.4: Schema Information**

Schema information refers to the details and specifications of a database schema. A database schema defines the structure, organization, and relationships of the data within a database. It includes information about tables, columns, data types, constraints, relationships, and other elements that define the logical and physical structure of the database.

**4.5: Table Definition**

Tbl\_admin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ad\_id | ad\_name | ad\_email | ad\_password | ad\_date |

Tbl\_doctor

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| doc\_id | doc\_  name | doc\_  img | doc\_  about | doc\_  phone | doc  \_email | doc\_exp | doc\_  educ | doc\_  password | doc\_  status |

Tbl\_patient

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| pat\_id | pat\_name | pat\_phone | pat\_email | pat\_password | pat\_date |

Tbl\_appointment

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ap\_id | Doc\_id | Pat\_  id | age | phoneno | gender | Ap\_  date | Ap\_  time | problem | Amount | Ap\_  status | App\_  date |

Tbl\_feedback

|  |  |  |  |
| --- | --- | --- | --- |
| feed\_id | pat\_id | feed | feed\_date |

Tbl\_payment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pay\_id | ap\_id | pay\_method | amount | trans\_id | pay\_status | pay\_date |

Tbl\_schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| sch\_id | doc\_id | days | time | sch\_status | sch\_date |

Tbl\_prescription

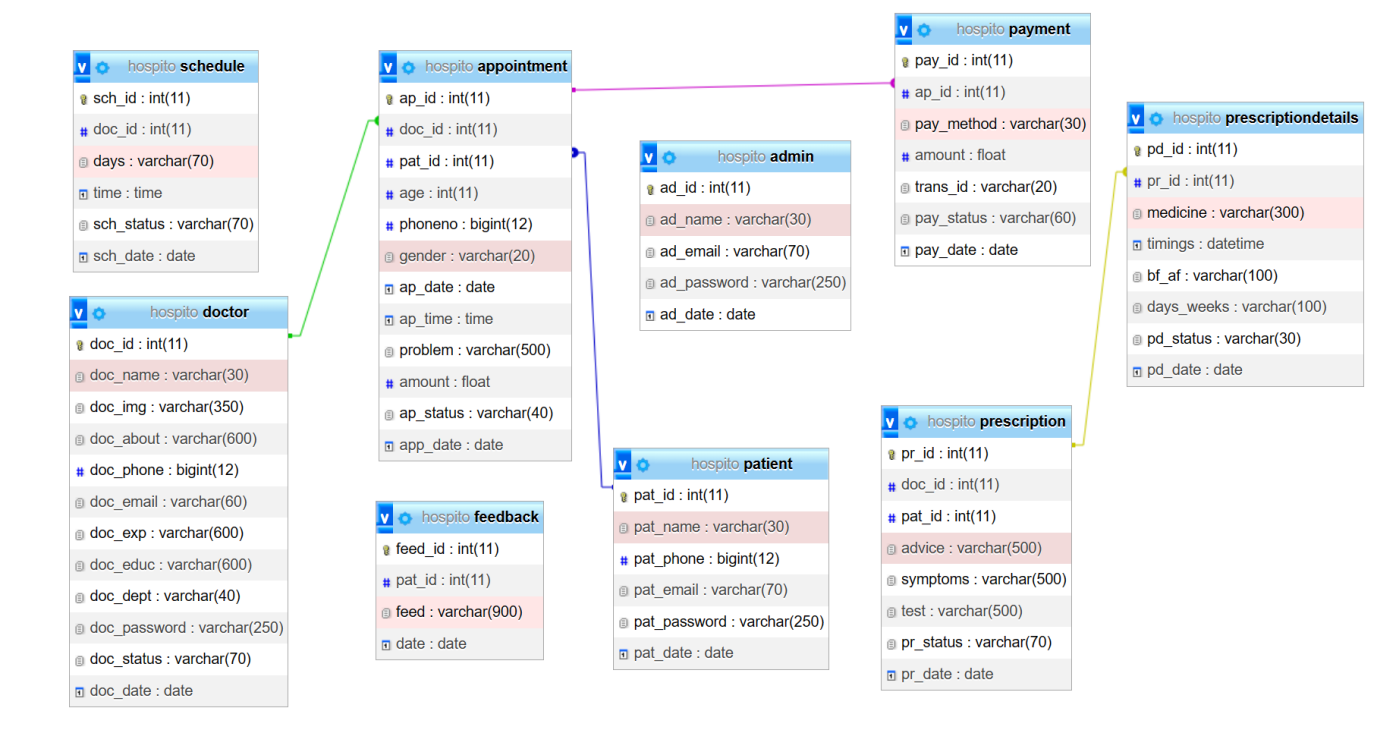
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| pr\_id | doc\_id | pat\_id | advice | symptoms | test | pr\_status | pr\_date |

Tbl\_prescription details

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| pd\_id | pr\_id | medicine | timings | bf\_af | days\_weeks | pd\_status | pd\_date |

**4.6: Physical Design**

Physical design refers to the process of determining the physical implementation details of a database schema in a specific database management system (DBMS). It involves making decisions about how the data will be stored, organized, and accessed on the underlying storage devices. The physical design is responsible for optimizing performance, storage efficiency, and scalability of the database.



**4.7: Data Dictionary**

* Admin table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| ad\_id | int | 11 | Primary key | Primary key of the table |
| ad\_name | varchar | 50 | Not null | To store admin’s id |
| ad\_email | varchar | 50 | Not null | To store admin’s email |
| ad\_password | varchar | 250 | Not null | To store admin’s password |
| ad\_date | date | 6 | Not null | To store date |

* Doctor table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| doc\_id | int | 11 | primary key | primary key of the table |
| doc\_name | varchar | 50 | not null | to store doctor’s name |
| doc\_img | varchar | 350 | not null | to store doctor’s image |
| doc\_about | varchar | 250 | not null | to store doctor’s about |
| doc\_phone | bigint | 12 | not null | to store doctor’s phone number |
| doc\_email | varchar | 50 | not null | To store doctor’s email |
| doc\_exp | varchar | 200 | not null | To store doctor’s experience |
| doc\_educ | varchar | 200 | not null | To store doctor’s education |
| doc\_password | varchar | 250 | not null | To store doctor’s password |
| doc\_status | varchar | 100 | not null | To store doctor’s status |

* patient table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| p\_id | int | 11 | Primary key | Primary key of the table |
| p\_name | varchar | 50 | Not null | To store patient’s name |
| p\_phone | bigint | 12 | Not null | To store patient’s phone |
| p\_email | varchar | 50 | Not null | To store patient’s email |
| p\_password | varchar | 250 | Not null | To store patient’s password |
| p\_date | date | 6 | Not null | To store date |

* Appointment table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| ap\_id | int | 11 | primary key | primary key of the table |
| doc\_id | int | 50 | Foreign key | Foreign key of the table |
| p\_id | int | 350 | Foreign key | Foreign key of the table |
| Age | varchar | 250 | not null | to store age of the patient |
| Phone number | bigint | 12 | not null | to store patient’s phone number |
| gender | varchar | 50 | not null | To store patient’s gender |
| ap\_date | date | 200 | not null | To store appointment date |
| ap\_time | time | 200 | not null | To store appointment time |
| problem | varchar | 250 | not null | To store patient’s problem |
| amount | Varc har | 100 | not null | To store appointment amount |
| ap\_status | varchar | 100 | Not null | To store appointment status |
| app\_date | Date | 6 | Not null | To store date |

* Feedback table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| feed\_id | int | 11 | Primary key | Primary key of the table |
| p\_id | Int | 11 | Foreign key | Foreign key of the table |
| Feed | varchar | 50 | Not null | To store feedback |
| feed\_date | date | 6 | Not null | To store feedback date |

* Feedback table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| feed\_id | int | 11 | Primary key | Primary key of the table |
| p\_id | Int | 11 | Foreign key | Foreign key of the table |
| Feed | varchar | 50 | Not null | To store feedback |
| feed\_date | date | 6 | Not null | To store feedback date |

* payment table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| pay\_id | int | 11 | Primary key | Primary key of the table |
| ap\_id | int | 11 | Foreign key | Foreign key of the table |
| pay\_method | varchar | 12 | Not null | To store payment method |
| amount | varchar | 50 | Not null | To store payment amount |
| trans\_id | varchar | 20 | Not null | To store transaction id |
| pay\_status | varchar | 100 | Not null | To store the status |
| pay\_date | date | 6 | Not null | To store payment date |

* Schedule table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| sch\_id | int | 11 | Primary key | Primary key of the table |
| doc\_id | varchar | 50 | Not null | Foreign key of the table |
| Days | varchar | 50 | Not null | To store the schedule days |
| Time | varchar | 100 | Not null | To store time |
| Sch\_status | varchar | 10 | Not null | To store status |
| Sch\_date | date | 6 | Not null | To store schedule date |

* Prescription table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| pr\_id | int | 11 | Primary key | Primary key of the table |
| doc\_id | int | 11 | Foreign key | Foreign key of the table |
| p\_id | int | 11 | Foreign key | Foreign key of the table |
| advice | varchar | 50 | Not null | To store prescription advice |
| symptoms | varchar | 20 | Not null | To store prescription symptoms |
| Test | varchar | 100 | Not null | To store the prescription test details |
| pr\_status | varchar | 6 | Not null | To store the status |
| pr\_date | date | 6 | Not null | To store prescription date |

* Prescription details table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | length | Constraint type | comment |
| pd\_id | int | 11 | Primary key | Primary key of the table |
| pr\_id | int | 11 | Foreign key | Foreign key of the table |
| medicine | varchar | 50 | Not null | to store prescription |
| timings | varchar | 50 | Not null | To store timings |
| bf\_af | varchar | 20 | Not null | To store prescription before and after meals |
| days\_weeks | varchar | 100 | Not null | To store the days |
| pd\_status | varchar | 6 | Not null | To store the status |
| pd\_date | date | 6 | Not null | To store prescription date |

An ER (Entity-Relationship) diagram is a graphical representation of the entities (objects or concepts) within a system or domain, their attributes (properties or characteristics), and the relationships between entities. It is a visual tool used in database design to illustrate the logical structure of a database.

|  |  |  |
| --- | --- | --- |
| name | Notation | discription |
| entity | C:\Users\asus\AppData\Local\Temp\ksohtml13108\wps1.png | The entity is represented by a box within the ERD. Entities are abstract concepts, each representing one or more instances of the concept in question. An entity might be considered a container that holds all of the instances of a particular thing in a system. |
| Relationship | C:\Users\asus\AppData\Local\Temp\ksohtml13108\wps2.png | Relationships are represented by  Diamonds. A relationship is a named collection or association between entities or used to relate  to two or more entities with some common attributes or meaningful interaction between the objects. |
| Link | C:\Users\asus\AppData\Local\Temp\ksohtml13108\wps3.png | Lines link attributes to entity sets or entity sets to relationship sets |
| attribute | C:\Users\asus\AppData\Local\Temp\ksohtml13108\wps4.png | Attributes are represented by  Oval. An attribute is a single data item related to a database object. The database scheme  associates with each database entity. |
| Derived attribute | C:\Users\asus\AppData\Local\Temp\ksohtml13108\wps5.jpg | Dashed ellipse denotes derived attributes. |
| Key Attribute or Single  Valued Attribute | C:\Users\asus\AppData\Local\Temp\ksohtml13108\wps6.pngC:\Users\asus\AppData\Local\Temp\ksohtml13108\wps7.png | As entity type usually has an attribute whose values are distinct for each individual entry in the entity set. It is represented by an underline word in ellipse. |
| Multivalued Attribute | C:\Users\asus\AppData\Local\Temp\ksohtml13108\wps8.pngC:\Users\asus\AppData\Local\Temp\ksohtml13108\wps9.png | Attributes that have different numbers of values for a particular attribute. It is represented by a double ellipse. |
| Cardinality Ratio | 1:1  1:M  M:1  M:M | It specifies the maximum  number of relationships  instances that an entity can participate in. There are four cardinality ratios. |
|  |  |  |

**4.9: Data Base Administration**

**4.9.1: System Information**

* **Server:** localhost via TCP/IP
* **Server type**: MySQL
* **Server version:** 5.6.12-log - MySQL Community Server (GPL)
* **Protocol version:** 10
* **User:** root@localhost
* **Server charset:** UTF-8 Unicode (utf8)

**4.9.2: DBMS Configuration**

* **XAMPP Version:** 2.4
* **MySQL version:** 5.6.12

**4.9.3: Support Software Required**

The system installs MYSQL server while installing WAMP software. All the backup content stores in MYSQL data folder.

**4.9.4: Storage Requirement**

The storage engine represents the heart of a MySQL Server.

* Recovering the database from system failure
* Management of files and database pages used to store data
* Manage data buffers and system IO to the physical data pages
  + Manage locking and concurrency issues

**4.9.5: Backup and Recovery**

Database recovery is the process of restoring the databases to a correct state following a failure. The failure may be the result of a system crash due to hardware of software errors, a media failure, such as a head crash, or a software error in the application, such as a logical error in the program that is accessing the database. It is the responsibility of DBMS to ensure that the database is reliable and remains in a consistent state in the presence of failure. In general, backup and recovery refer to the various strategies and procedures involved in protecting the database against data loss and reconstructing the data such as that no data is lost after failure.