

**JAGAN INSTITUTE OF MANAGEMENT STUDIES**

**SECTOR – 5, ROHINI, NEW DELHI**



**(Affiliated to)**

**GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY**

**SECTOR – 16 C, DWARKA, NEW DELHI**



**Submitted To: Mr. Sanjive Saxena**

**Submitted By: Saurab Sharma**

**Enrollment No: 03014004423**

**MCA 1st Year (Section - A)**

**1st Semester**

## **ACKNOWLEDGEMENT**

I take this opportunity to present my votes of thanks to my faculty who really acted as pillars to help my way throughout the execution of lab exercises that has led to successful and satisfactory completion of the Project.

I feel great sense of gratitude for Mr. Sanjive Saxena under whose guidance and motivation this work has been performed.

I would also like to express my thanks to all lab assistants for giving me opportunity to work under their esteemed guidance. This project would not have completed without their guidance and coordination.

The inspiration of the faculty members of the Information Technology Department of JIMS Rohini (Sec-5) enabled me to make a thorough study of these subjects.

**Student Name: Saurab Sharma**

**Enrollment No: 03014004423**

## **CERTIFICATE**

I, Saurab (**03014004423**) certify that the Project Report entitled Hospital System is done by me and it is an authentic work carried out by me at JIMS. The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

**Mr. Sanjive Saxena**  
**Professor (IT)**  
**JIMS, Rohini**

# CONTENTS

S. No	Topic	Page No
1.	Certificate	-
2.	Acknowledgements	-
3.	Synopsis	-
4.	<b>CHAPTER-1 PROBLEM FORMULATION</b> <ul style="list-style-type: none"> <li>➤ Introduction</li> <li>➤ present state of the art</li> <li>➤ Need of computerization</li> <li>➤ Tools and Platforms</li> <li>➤ Software Specifications</li> <li>➤ Importance of the work</li> </ul>	1-7
5.	<b>CHAPTER-2 SYSTEM ANALYSIS</b> <ul style="list-style-type: none"> <li>➤ Feasibility study</li> <li>➤ Technical Feasibility</li> <li>➤ Economic Feasibility</li> <li>➤ Operational Feasibility</li> <li>➤ Other Feasibility Dimensions</li> <li>➤ Analysis Methodology</li> <li>➤ Choice of the platform</li> </ul>	8-10
6.	<b>CHAPTER-3 SYSTEM DESIGN</b> <ul style="list-style-type: none"> <li>➤ Design Methodology</li> <li>➤ Database Design</li> <li>➤ ERD</li> <li>➤ DFD</li> <li>➤ Input Design</li> <li>➤ Output Design</li> <li>➤ Code Design &amp; Development</li> </ul>	11 -31
7.	<b>CHAPTER-4: TESTING AND IMPLEMENTATION</b> <ul style="list-style-type: none"> <li>➤ Testing Methodology</li> <li>➤ Unit Testing</li> <li>➤ Module Testing</li> <li>➤ Integration Testing</li> <li>➤ System Testing</li> <li>➤ Acceptance Testing</li> <li>➤ Test Data &amp; Test Cases</li> <li>➤ Test Reports and debugging</li> <li>➤ Implementation Manual</li> <li>➤ Implementation</li> <li>➤ Users Training</li> <li>➤ Post Implementation Maintenance</li> </ul>	32-42
8.	<b>CHAPTER-5: CONCLUSION &amp; REFERENCES</b>	43-47
9.	<b>CHAPTER-6 (ANNEXURES) SCREEN SNAPSHOTS</b>	48-51

## **LIST OF FIGURES**

<b>Fig No.</b>	<b>Description</b>	<b>Page No.</b>
Fig 0.1	Sdlc model	7
Fig 1.1	Use case model	27
Fig 1.2	Erd model	28
Fig 1.3	Level 0 dfd	29
Fig 1.4	Level 1 dfd	30
Fig 1.5	Level 2 dfd	31
Fig 1.6	Input design	31
Fig 1.7	Output design	32
Fig 1.8	Log in page	32
Fig 1.9	Doctor dashboard	33
Fig 2.0	Add patient details	33
Fig 2.1	Payrolls	33
Fig 2.2	Admin dashboard	34
Fig 2.3	Add pharm details	34
Fig 2.4	Admin details	35
Fig 2.5	Different eqp code	35
Fig 2.6	Doctors log in	35
Fig 2.7	Payrolls	35
Fig 2.8	Prescription details	35
Fig 2.9	Vital details	35
Fig 3.0	Test cases 1	54
Fig 3.1	Test results	54
Fig 3.2	Test data	54
Fig 3.3	Use case model	63
Fig 3.4	Class diagram	64
Fig 3.5	Context of hm	64
Fig 3.6	Log in page	65
Fig 3.7	Vendor details	65
Fig 3.8	Patient details	66
Fig 3.9	Create parametrical	66

## **LIST OF TABLES**

<b>Table No.</b>	<b>Title</b>	<b>Page No.</b>
Fig 0.1	ER SYMBOLS	28

# Synopsis

## Title of project:

Hospital system

## Problems with Existing system:

Lack of integration: Hospital management systems often involve multiple modules and functionalities, such as patient registration, scheduling, billing, and inventory management. If these modules are not properly integrated, it can lead to inefficiencies, data inconsistencies, and increased workload for staff.

- **Lack of integration:** Hospital management systems often involve multiple modules and functionalities, such as patient registration, scheduling, billing, and inventory management. If these modules are not properly integrated, it can lead to inefficiencies, data inconsistencies, and increased workload for staff.
- **Limited scalability:** Some hospital management systems may not be designed to handle the growing needs of a healthcare facility. As the hospital expands or introduces new departments, the system may struggle to accommodate the increased volume of data, leading to performance issues and slowdowns.
- **User interface and usability:** Complex and unintuitive user interfaces can make it challenging for staff to navigate and use the system effectively. This can result in errors, delays, and frustration among users.
- **Data security and privacy concerns:** Hospital management systems store sensitive patient data, including medical records and personal information. Inadequate security measures or vulnerabilities in the system can put this data at risk of unauthorized access or breaches, compromising patient privacy and confidentiality.
- **Inefficient workflows and processes:** If the hospital management system does not streamline and automate workflows, it can lead to manual and time-consuming processes. This can decrease productivity, increase administrative burden, and hinder effective patient care.
- **Lack of interoperability:** Hospitals often need to share patient information and collaborate with external healthcare providers, laboratories, and insurance companies. If the existing system does not support interoperability standards, it can be challenging to exchange data accurately and efficiently, leading to communication gaps and delays in decision-making.
- **Limited reporting and analytics capabilities:** Effective decision-making requires access to accurate and timely data. If the hospital management system lacks comprehensive

## Reason For Choosing the Topic:

There are several reasons why someone might choose a hospital management system.

- **Efficient Workflow:** Hospitals deal with a vast amount of data and operations daily. Implementing an HMS can streamline workflows by automating processes such as patient registration, appointment scheduling, billing, and inventory management. This can save time, reduce errors, and improve overall efficiency.
- **Improved Patient Care:** An HMS can enhance the quality of patient care by facilitating accurate and timely access to medical records, test results, and treatment information. It enables healthcare professionals to make informed decisions, track patient progress, and ensure coordinated care across different departments.
- **Enhanced Data Management:** Hospitals generate and handle large volumes of data. An HMS provides a centralized system for storing, managing, and securing electronic medical records, thereby improving data accessibility, integrity, and privacy. It also enables advanced analytics for better decision-making and research purposes.
- **Resource Optimization:** Effective management of hospital resources is crucial for providing efficient healthcare services. An HMS can assist in optimizing resource allocation, including beds, equipment, and staff, based on real-time data and demand. This leads to better utilization of resources and cost savings.
- **Financial Management:** Hospitals need to manage billing, insurance claims, and financial transactions effectively. An HMS can automate billing processes, generate accurate invoices, track payments, and manage insurance information, ensuring transparency and reducing financial discrepancies.
- **Regulatory Compliance:** Healthcare institutions must comply with various regulations and standards. An HMS can help hospitals maintain compliance by ensuring proper documentation, data security measures, and adherence to privacy regulations like HIPAA (Health Insurance Portability and Accountability Act).
- **Scalability and Future Readiness:** Hospitals aim for long-term growth and sustainability. By implementing an HMS, hospitals can scale their operations, accommodate increasing patient volumes, and adapt to evolving healthcare technologies and practices.

## 1) Objective & Scope of Project:

The objective of a Hospital Management System (HMS) is to streamline and automate the administrative, operational, and clinical processes within a healthcare organization. It aims to improve the efficiency, accuracy, and quality of healthcare services while enhancing patient care and safety. The scope of a Hospital Management System typically includes the following areas:

- **Patient Management:** The system handles patient registration, appointment scheduling, admission, discharge, and transfer processes. It maintains patient records, including demographics, medical history, diagnosis, and treatment details.
- **Staff Management:** The HMS manages information related to healthcare professionals, including doctors, nurses, and other support staff. It facilitates staff scheduling, attendance tracking, and payroll management.
- **Inventory and Pharmacy Management:** The system helps in managing the stock of medicines, medical supplies, and equipment. It tracks inventory levels, generates purchase orders, and manages the distribution of medications within the hospital.
- **Billing and Finance:** The HMS handles billing and invoicing processes, including insurance claims and patient payments. It generates financial reports, tracks expenses, and manages accounts receivable and payable.
- **Laboratory and Radiology Management:** The system integrates with laboratory and radiology departments, allowing for the management of diagnostic tests, results recording, and report generation.
- **Electronic Health Records (EHR):** A critical component of an HMS is the maintenance and management of electronic health records. It enables authorized healthcare providers to access and update patient information securely, ensuring continuity of care.
- **Decision Support and Analytics:** The system may provide tools for data analysis, reporting, and decision support. It helps in identifying trends, optimizing resource allocation, and improving clinical outcomes.
- **Integration and Interoperability:** The HMS may integrate with other healthcare systems, such as electronic prescribing, billing systems, and health information exchanges, to facilitate seamless information exchange and interoperability.

Overall, the objective and scope of a Hospital Management System revolve around improving the efficiency and effectiveness of healthcare operations, enhancing patient care, and ensuring the smooth functioning of the hospital's administrative and clinical processes.



## 2) Analysis Methodology

STAGES OF ITERATIVE MODEL: Use Iterative SDLC (Software Development Life Cycle)

### Phases of Life Cycle Model

- **Requirement gathering & analysis:** In this phase, requirements are gathered from customers and checked by an analyst whether requirements will fulfil or not.
- **Design:** In the design phase, the team design the software by different diagrams which include a Data Flow diagram, activity diagram, class diagram, state transition diagram, etc.
- **Implementation:** In the implementation, requirements are written in the coding language and transformed into a computer program which is called Software.
- **Testing:** After completing the coding phase, software testing starts using different test methods. There are many test methods, but the most common are white box, black box, and grey box test methods.
- **Review:** If there is any error found then the process starts again from the requirement gathering.
- **Deployment:** After completing all the phases, the software is deployed to its work environment.
- **Maintenance:** In the maintenance phase, after deployment of the software in the working environment there may be some bugs, some errors or new updates are required.

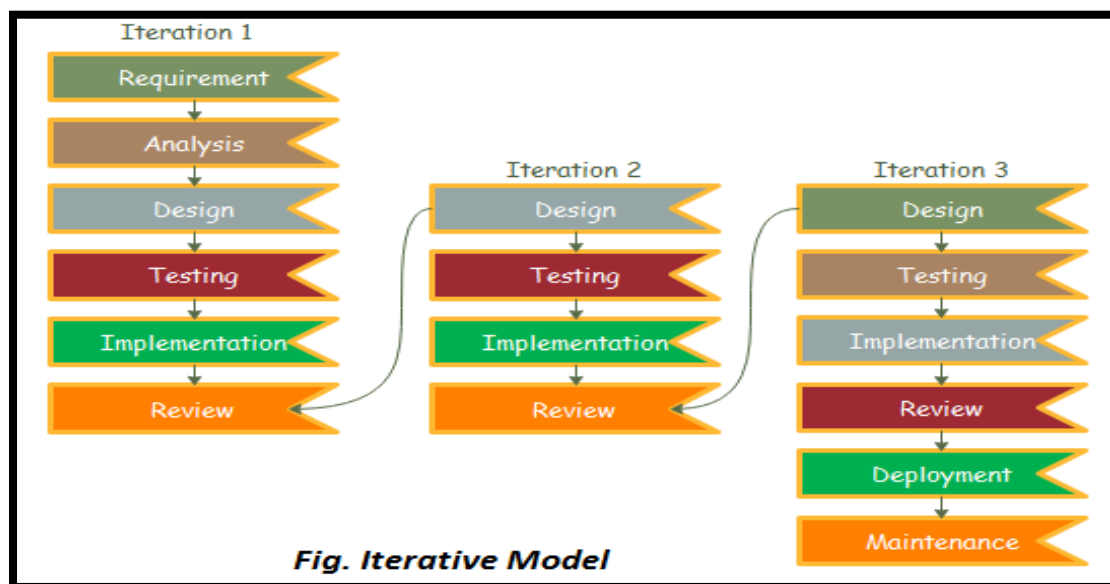


Fig 0.1 (SDLC MODEL)

➤ **Reason for choosing the Iterative model:**

- Requirements of the complete system are clearly defined and understood.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill sets are not available and are planned to be used on a contract basis for specific iterations.
- Some functionalities or requested enhancements may evolve with time.

➤ **Advantages of the model:**

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- It works well for smaller projects where requirements are very well understood.

**3) Tools and Platform:**

➤ **Hardware Components**

- Hard Disk: - 1000 GB
- Processor: INTEL CORE i5
- RAM: - 8 GB
- Front end: - HTML, CSS, JavaScript
- Back end: - PHP version 3.0, MySQL

➤ **Software Specifications:**

- **Operating System:** Window/ Linux operating system
- **(HTML)** elements are the building blocks of HTML pages. With HTML constructs, images, and other objects such as interactive forms may be embedded into the rendered page.
- **(CSS)** enable the separation of content and presentation, including layout, colors, fonts, etc. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.
- **(JavaScript)** is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles.

- **(Bootstrap)** is an HTML, CSS and JS library that focuses on simplifying the primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font, and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements.
- **(Backend)**- PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. It allows web developers to create dynamic content that interacts with databases like MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

#### 4) **Testing Methods Used:**

The testing methodology plays a crucial role in an Hospital Management system project as it ensures the quality, reliability, and security of the system. Here are some key roles that the testing methodology plays in this project:

- **Quality Assurance:** The testing methodology helps in identifying and resolving defects, bugs, and errors in the system. It ensures that the system functions as intended and meets the specified requirements and quality standards.
- **Risk Mitigation:** Testing helps in mitigating risks associated with the Hospital Management system, such as security vulnerabilities, data breaches, and financial risks. By performing thorough testing, potential issues can be identified and addressed before the system goes live.
- **Functional Verification:** The testing methodology verifies the functionality of the Hospital Management system, ensuring that all features and functionalities work as expected. It involves testing different scenarios and use cases to validate that the system functions accurately and reliably.
- **Usability Testing:** Usability testing is an essential part of the testing methodology for an Hospital Management system. It focuses on assessing the system's user-friendliness, intuitive interface, and ease of navigation. Usability testing helps in identifying any usability issues and provides insights to enhance the user experience.
- **Security Testing:** Security is of utmost importance in an Hospital Management system. The testing methodology includes security testing techniques to identify vulnerabilities, validate access controls, test encryption mechanisms, and ensure compliance with industry security standards. This helps in safeguarding customer data and preventing unauthorized access.
- **Performance Testing:** Hospital Management systems must handle a large number of concurrent users and process transactions efficiently. The testing methodology includes performance testing to assess the system's responsiveness, scalability, and stability under various load conditions. It helps in identifying performance bottlenecks and optimizing the system's performance.

## ➤ **Unit Testing:**

It is a testing level that focuses on testing individual units or components of the system in isolation. Here are some key roles that unit testing plays in an Hospital Management system project:

- **Code Verification:** Unit testing helps in verifying the correctness of individual units of code within the system. It ensures that each unit functions as intended and produces the expected output based on different input scenarios. This level of testing helps identify coding errors, logic flaws, and functional defects early in the development process.
- **Isolation and Debugging:** Unit testing allows developers to isolate specific units of code and test them independently of the rest of the system. This isolation makes it easier to identify and debug issues within a specific unit, as the focus is solely on that unit. It helps in pinpointing the root cause of any defects or failures and facilitates faster troubleshooting.
- **Code Maintainability and Refactoring:** Unit testing encourages developers to write modular and loosely coupled code. By breaking down the system into smaller units, it becomes easier to maintain and update the codebase. Unit tests act as a safety net when refactoring or modifying code, ensuring that changes in one unit do not inadvertently affect the behavior of other units.
- **Regression Testing:** Unit tests serve as a form of regression testing, verifying that existing functionalities continue to work correctly as new code is added or modified. Whenever changes are made to the system, unit tests can be run to ensure that previously working units are not adversely affected.

By conducting thorough unit testing, developers can identify and rectify defects at an early stage, leading to a more stable, reliable, and maintainable Hospital Management system.

## ➤ **Module Testing:**

Module testing, also known as unit testing, is a crucial level of testing in the development of an Hospital Management system. It focuses on testing individual modules or components of the system in isolation to ensure their proper functioning. Here are the key roles of module testing in an Hospital Management system project:

- **Isolating and Testing Individual Modules:** Module testing allows for the isolation of specific modules or components within the Hospital Management system. Each module is tested independently to verify its functionality, adherence to design specifications, and correct integration with other modules.
- **Detecting and Correcting Defects:** By conducting module testing, defects or bugs within individual modules can be identified and resolved at an early stage. It helps in improving the quality of the code and minimizing the risk of errors propagating to other parts of the system.

- **Verification of Module Functionality:** Module testing verifies the functionality of each module as per its intended purpose. It ensures that the module performs the expected operations, processes data correctly, and produces accurate outputs.
- **Supporting Maintenance and Refactoring:** Module testing provides a solid foundation for system maintenance and future enhancements. By having comprehensive tests for each module, developers can refactor or modify the code with confidence, knowing that the intended functionality remains intact.
- **Enabling Parallel Development:** Module testing enables parallel development within a project. Different teams or developers can work on separate modules simultaneously, allowing for faster development cycles and improved overall productivity.

In summary, module testing plays a vital role in an Hospital Management system project by ensuring the quality, functionality, and reliability of individual modules. It helps in identifying and fixing defects, verifying module interactions, and facilitating efficient development and maintenance processes.

### ➤ **Integration Testing:**

The testing methodology plays a crucial role in an Hospital Management system project as it ensures the quality, reliability, and security of the system. Here are some key roles that the testing methodology plays in such a project:

- **Quality Assurance:** The testing methodology helps in identifying and resolving defects, bugs, and errors in the system. It ensures that the system functions as intended and meets the specified requirements and quality standards.
- **Risk Mitigation:** Testing helps in mitigating risks associated with the Hospital Management system, such as security vulnerabilities, data breaches, and financial risks. By performing thorough testing, potential issues can be identified and addressed before the system goes live.
- **Functional Verification:** The testing methodology verifies the functionality of the Hospital Management system, ensuring that all features and functionalities work as expected. It involves testing different scenarios and use cases to validate that the system functions accurately and reliably.
- **Usability Testing:** Usability testing is an essential part of the testing methodology for an Hospital Management system. It focuses on assessing the system's user-friendliness, intuitive interface, and ease of navigation. Usability testing helps in identifying any usability issues and provides insights to enhance the user experience.
- **Security Testing:** Security is of utmost importance in an Hospital Management system. The testing methodology includes security testing techniques to identify vulnerabilities, validate access controls, test encryption mechanisms, and ensure compliance with industry security standards. This helps in safeguarding customer data and preventing unauthorized access.

## ➤ **System Testing:**

System testing, which is a higher-level testing phase, plays a significant role in the development and implementation of an Hospital Management system. It focuses on testing the system, integrating different modules and components, and evaluating its behavior and performance in a real-world environment.

- **Integration Testing:** System testing involves the integration of various modules and components of the Hospital Management system. It verifies that all the individual components work together seamlessly and exchange data correctly. Integration testing ensures that the system functions as a cohesive unit, and any issues arising from the interaction between different modules can be identified and addressed.
- **End-to-End Testing:** System testing includes end-to-end testing, which evaluates the entire flow of operations in the Hospital Management system. It involves testing various scenarios, starting from user login, account management, transaction processing, and other core functionalities. End-to-end testing verifies that all the interconnected components and systems work together correctly to provide a smooth and comprehensive user experience.
- **Functional Testing:** System testing verifies the functional requirements of the Hospital Management system. It ensures that all the expected functionalities, such as balance inquiries, fund transfers, bill payments, and statement generation, are working accurately. Function testing involves designing test cases that cover all the system's functionalities and validating their proper implementation.
- **Performance Testing:** System testing assesses the performance of the Hospital Management system under realistic conditions. It measures the system's response time, throughput, and scalability when handling a high volume of concurrent users and transactions. Performance testing helps identify performance bottlenecks, optimize system resources, and ensure that the system can handle the expected load without degradation.
- **Security Testing:** Security is a critical aspect of an Hospital Management system. System testing includes security testing to identify vulnerabilities, ensure data encryption, validate access controls, and protect against unauthorized access or malicious activities. Security testing helps in maintaining the confidentiality, integrity, and availability of customer data and financial transactions.
- **Usability Testing:** Usability testing evaluates the user-friendliness and intuitiveness of the Hospital Management system. It ensures that the system's interface is easy to navigate, transactions can be performed efficiently, and customers can access the required information without confusion or frustration. Usability testing helps in identifying and addressing any usability issues, improving the overall user experience.

Overall, system testing in an Hospital Management system project plays a vital role in ensuring that the integrated system meets the functional requirements, performs well under expected loads, and maintains the highest level of security. It aims to deliver a reliable, user-friendly, and secure Hospital experience to customers.

## ➤ **Whitebox/Blackbox Testing:**

White-box and black-box testing are two different levels of testing that play important roles in ensuring the quality and reliability of an Hospital Management system.

### ▪ **White-box Testing:**

- Role: White-box testing is performed at the code or system level and focuses on the internal structure and logic of the system. It is typically conducted by developers or testers with access to the system's source code.

- Purpose: The main purpose of white-box testing is to validate the correctness of the system's internal workings, including algorithms, data structures, and program flows.

- Techniques: White-box testing techniques involve examining the code and designing test cases based on the knowledge of the system's internal structure. Techniques such as statement coverage, branch coverage, and path coverage are used to ensure that all possible code paths are tested.

- Role in Hospital Management System: White-box testing helps identify defects and vulnerabilities in the system's code, ensuring that the logic and algorithms are implemented correctly. It helps in detecting coding errors, security vulnerabilities, and other potential issues that may impact the system's reliability and security.

### ▪ **Black-box Testing:**

- Role: Black-box testing is performed at the functional or system level, focusing on the external behavior of the system without considering its internal implementation. It is typically conducted by independent testers who do not have knowledge of the system's internal workings.

- Purpose: The main purpose of black-box testing is to evaluate the system's functionality, usability, and adherence to specified requirements.

- Techniques: Black-box testing techniques involve designing test cases based on the system's requirements and expected behavior. Techniques such as equivalence partitioning, boundary value analysis, and use case testing are used to ensure that the system functions correctly from a user's perspective.

- Role in Hospital Management System: Black-box testing helps assess the Hospital Management system's functionality, usability, and performance from the user's standpoint. It ensures that all user interactions, such as account management, transaction processing, and security features, work as intended and meet the specified requirements.

In the context of an Hospital Management system, both white-box and black-box testing are crucial. White-box testing helps validate the internal workings of the system, ensuring code correctness and identifying security vulnerabilities. Black-box testing, on the other hand, focuses on the system's external behavior, verifying that it meets functional and usability requirements. By combining both testing levels, the Hospital Management system can be thoroughly validated, ensuring its reliability, security, and adherence to user expectations.

## ➤ **Acceptance Testing:**

Acceptance testing plays a significant role in the development of an Hospital Management system. It is a crucial testing level that focuses on evaluating whether the system meets the requirements and expectations of its intended users, typically the bank's customers. Here are the roles and benefits of acceptance testing in an Hospital Management system project:

- **Customer Validation:** Acceptance testing allows the bank's customers to validate and verify that the Hospital Management system meets their needs and expectations. It ensures that the system functions as intended and provides the desired features and functionalities to the end-users.
- **Requirement Verification:** Acceptance testing helps in validating that the system meets all the specified requirements. By conducting this testing level, any gaps or discrepancies between the system's actual behavior and the documented requirements can be identified and addressed.
- **User Experience Assessment:** Acceptance testing focuses on assessing the user experience of the Hospital Management system. It ensures that the system is user-friendly, intuitive, and easy to navigate for customers. Feedback from users during acceptance testing helps in improving the system's interface and overall user experience.
- **Bug Identification:** Acceptance testing helps in identifying any defects, bugs, or issues that were not caught during earlier testing phases. By involving real users in the testing process, different scenarios and use cases can be explored, leading to the discovery of potential issues that might affect the system's functionality or performance.
- **Risk Mitigation:** Acceptance testing assists in mitigating risks associated with the Hospital Management system. By involving end-users in the testing process, potential security vulnerabilities or data breaches can be identified and addressed before the system is deployed.
- **Business Validation:** Acceptance testing ensures that the Hospital Management system aligns with the bank's business objectives and processes. It verifies that the system supports the required financial transactions, account management, and other Hospital operations accurately and reliably.
- **Customer Satisfaction:** Ultimately, acceptance testing helps in ensuring customer satisfaction. By involving end-users in the testing process and addressing any issues or concerns raised during this phase, the bank can deliver a high-quality Hospital Management system that meets customer expectations and fosters trust and confidence in the service.

In summary, acceptance testing at the user level plays a crucial role in validating the Hospital Management system's compliance with customer requirements, enhancing user experience, identifying bugs, and mitigating risks. It helps in delivering a reliable and customer-centric Hospital Management experience.



## 5) **Contribution Project Makes:**

Here are some potential contributions that an Hospital system can offer:

- **Streamlined Patient Registration:** The HMS can automate the patient registration process, allowing for quick and accurate data entry, reducing paperwork, and ensuring efficient management of patient records.
- **Efficient Appointment Scheduling:** The system can facilitate appointment scheduling for patients, doctors, and other medical staff. It can optimize appointment slots, minimize scheduling conflicts, and send automated reminders to patients.
- **Seamless Electronic Medical Records (EMR):** The HMS enables the digital storage and management of patient medical records. It ensures easy access, updates, and sharing of medical information among authorized healthcare professionals, leading to improved patient care and coordination.
- **Effective Billing and Insurance Management:** The system can handle billing processes, generate invoices, and manage insurance claims. It helps streamline financial transactions, reduces errors, and enhances transparency in the billing and payment cycle.
- **Inventory and Pharmacy Management:** The HMS can assist in managing hospital inventory, including medical supplies, medications, and equipment. It helps track stock levels, automate reordering, and ensure proper inventory management to avoid shortages or excesses.
- **Enhanced Staff and Resource Management:** The system can aid in scheduling and managing healthcare staff, assigning duties, and tracking their attendance. It also facilitates efficient allocation and utilization of hospital resources, such as rooms, equipment, and facilities.
- **Data Analytics and Reporting:** The HMS can generate comprehensive reports and analytics regarding patient demographics, treatment outcomes, resource utilization, and financial performance. This data-driven insight can support informed decision-making and improve overall operational efficiency.
- **Enhanced Communication and Collaboration:** The system enables better communication and collaboration among healthcare professionals, departments, and stakeholders. It can facilitate secure messaging, sharing of test results, and interdisciplinary coordination, leading to improved patient care.

These are just a few potential contributions of a Hospital Management System. Implementing an effective HMS can result in improved patient experience, streamlined operations.

# Chapter-1

## Problem Formulation

### 1.1) Introduction:

The Hospital Management System (HMS) is a comprehensive software solution designed to streamline and automate the administrative and clinical processes within a healthcare facility. It serves as a centralized platform that facilitates the efficient management of various aspects of hospital operations, including patient registration, appointment scheduling, medical records management, billing and invoicing, inventory management, and more.

The primary objective of a Hospital Management System is to enhance the overall efficiency and effectiveness of healthcare service delivery. By digitizing and integrating various processes, it eliminates manual paperwork, reduces human errors, and enables healthcare providers to focus more on patient care.

#### **Key Features of Hospital Management System:**

- **Patient Registration:** The HMS allows quick and accurate registration of patients, capturing essential details such as personal information, medical history, and contact details.
- **Appointment Scheduling:** This feature enables efficient management of patient appointments, including online booking, rescheduling, and cancellations. It helps optimize doctor-patient coordination and reduces waiting times.
- **Electronic Health Records (EHR):** The system stores and manages patients' electronic health records, including medical history, diagnoses, lab reports, prescription details, and treatment plans. It ensures secure and centralized access to patient information for authorized healthcare professionals.
- **Billing and Invoicing:** The HMS automates the billing process by generating invoices, tracking payments, and managing insurance claims. It helps in accurate billing, reduces errors, and ensures timely payments.
- **Pharmacy and Inventory Management:** This module assists in managing hospital inventory, including medicines, medical supplies, and equipment. It tracks stock levels, automates reordering, and optimizes inventory management to avoid shortages or excesses.
- **Laboratory and Radiology Information System:** The system integrates laboratory and radiology departments, enabling efficient test orders, sample tracking, result recording, and report generation. It streamlines the diagnostic process and enhances communication

- **Staff Management:** The HMS facilitates managing healthcare staff, including doctors, nurses, technicians, and administrative personnel. It assists in rostering, scheduling shifts, leave management, and performance tracking.
- **Reporting and Analytics:** The system generates comprehensive reports and analytics on various hospital parameters, such as patient inflow, revenue generation, resource utilization, and clinical outcomes. These insights aid in strategic decision-making and process improvement.

## 1.2) **Present state of art:**

The present state of the art in hospital management systems (HMS) encompassed several key features and trends.

- **Electronic Health Records (EHR):** EHR systems have become the backbone of hospital management, allowing healthcare providers to maintain comprehensive and digitized patient records. Advanced EHR systems incorporate features such as clinical decision support, interoperability, and data analytics to enhance patient care and streamline administrative processes.
- **Integration and Interoperability:** HMS now emphasize interoperability to facilitate the seamless exchange of patient data between different healthcare systems, departments, and facilities. This allows for a more holistic view of the patient's medical history and enhances coordination among healthcare providers.
- **Mobile Applications:** The integration of mobile applications into hospital management systems has grown significantly. Mobile apps provide easy access to patient information, appointment scheduling, medication reminders, and remote consultations. They also support real-time communication between healthcare providers, patients, and caregivers.
- **Artificial Intelligence (AI) and Machine Learning (ML):** AI and ML technologies are being applied to various aspects of hospital management. These include predictive analytics for resource optimization, automated diagnosis and decision support systems, image analysis, natural language processing for data extraction, and chatbots for patient engagement and triage.
- **Telehealth and Remote Monitoring:** Hospital management systems now integrate telehealth capabilities, allowing healthcare providers to conduct virtual consultations and monitor patients remotely. This helps in improving access to care, reducing hospital visits, and managing chronic conditions effectively.

- **Patient Engagement:** HMS focus on patient engagement by providing portals or mobile apps that allow patients to access their medical records, schedule appointments, receive reminders, and communicate with healthcare providers. Patient satisfaction surveys and feedback mechanisms are also integrated into these systems.
- **Analytics and Reporting:** Advanced data analytics and reporting functionalities enable hospitals to gain valuable insights into their operations, patient outcomes, resource utilization, and financial performance. These insights can drive evidence-based decision-making, quality improvement initiatives, and cost reduction efforts.
- **Security and Privacy:** With the increasing digitization of patient data, robust security measures are critical. HMS incorporate advanced security features like encryption, access controls, and audit trails to protect patient information from unauthorized access or breaches.
- **Internet of Things (IoT):** IoT devices are being used to monitor patients, track assets, and optimize workflow within hospitals. These devices can collect real-time data, such as vital signs or environmental conditions, and integrate it with the HMS for enhanced patient care and operational efficiency.
- **Cloud Computing:** Cloud-based HMS solutions have gained popularity due to their scalability, flexibility, and cost-effectiveness. Cloud computing enables seamless data sharing, remote accessibility, and centralized management of hospital systems.

### 1.3) Tools & Platforms

#### ➤ Hardware Components:

- Hard Disk: - 500 GB
- Processor: INTEL CORE i3
- RAM: - 2 GB
- Front end: - HTML, CSS, JavaScript
- Back end: - PHP version 3.0, MySQL

#### ▪ **Development Environment:**

- **XAMPP:** A popular cross-platform software bundle that includes Apache web server, MySQL database, and PHP, allowing you to set up a local development environment.
- **WAMP:** Like XAMPP, but designed for Windows operating system.

#### ▪ **Code Editors:**

- Visual Studio Code: A lightweight and versatile code editor with support for PHP development and various extensions for enhancing productivity.

- **Version Control:**
  - Git: A distributed version control system widely used for managing source code. Platforms like GitHub, GitLab, or Bitbucket can host your repository and enable collaboration with others.
- **Database Management:**
  - phpMyAdmin: A web-based tool for managing MySQL databases. It provides an intuitive interface for executing SQL queries, managing tables, and interacting with the database.
- **Testing and Debugging:**
  - **PHPUnit:** A unit testing framework for PHP that helps ensure the correctness of your code by running automated tests.
  - **Xdebug:** A PHP extension that facilitates debugging and profiling by providing features like breakpoints, stack traces, and variable inspection.

#### **1.4) Need of Computerization:**

Computerization in hospital management systems offers numerous benefits and plays a crucial role in improving overall healthcare delivery.

- **Enhanced Efficiency:** Computerizing hospital management systems streamlines administrative processes, such as patient registration, appointment scheduling, billing, and inventory management. Automation reduces manual errors, improves accuracy, and enables faster processing, leading to increased efficiency and productivity.
- **Improved Patient Care:** Computerized systems provide healthcare professionals with instant access to comprehensive patient records, including medical history, test results, prescriptions, and treatment plans. This quick and accurate information facilitates better decision-making, improves diagnosis, and ensures appropriate and timely treatment for patients.
- **Enhanced Communication and Collaboration:** Computerization enables seamless communication and collaboration among different departments and healthcare professionals within a hospital. Digital systems allow for instant sharing of patient information, test results, and medical reports, fostering coordinated care and enabling prompt consultations among doctors, nurses, and specialists.
- **Efficient Resource Management:** Hospital management systems help optimize the allocation and utilization of resources such as staff, equipment, and supplies. Computerization enables better planning and scheduling of appointments, surgeries, and treatments, ensuring optimal utilization of resources and minimizing waiting times for patients.

- **Accurate Billing and Financial Management:** Automated billing systems eliminate manual errors, ensure accurate invoicing, and facilitate efficient financial management. Computerized systems enable seamless integration with insurance providers, reducing paperwork and processing times, and improving revenue cycle management.
- **Data Management and Analysis:** Computerized hospital management systems enable the collection, storage, and analysis of vast amounts of patient data. This data can be utilized for research, quality improvement initiatives, and clinical decision support systems. Analyzing trends and patterns can help identify potential health risks, improve treatment outcomes, and support evidence-based medical practices.
- **Regulatory Compliance:** Computerization helps hospitals comply with regulatory requirements and standards, such as electronic health record (EHR) mandates and data privacy regulations. Digital systems ensure data security, privacy, and confidentiality of patient information, with appropriate access controls and audit trails.
- **Telemedicine and Remote Care:** In recent times, computerized hospital management systems have become even more crucial due to the rise of telemedicine and remote care. Digital platforms enable virtual consultations, remote patient monitoring, and the exchange of medical information, expanding access to healthcare services and improving patient convenience.

### 1.5) Tools & Platforms

- **Hardware Components:**
  - Hard Disk: - 1000 GB
  - Processor: INTEL CORE i5
  - RAM: - 8 GB
  - Front end: - HTML, CSS, JavaScript
  - Back end: - PHP version 3.0, MySQL
- **Development Environment:**
  - XAMPP: A popular cross-platform software bundle that includes Apache web server, MySQL database, and PHP, allowing you to set up a local development environment.
  - WAMP: Like XAMPP, but designed for Windows operating system.
- **Code Editors:**
  - Visual Studio Code: A lightweight and versatile code editor with support for PHP development and various extensions for enhancing productivity.

- **Version Control:**
  - **Git:** A distributed version control system widely used for managing source code. Platforms like GitHub, GitLab, or Bitbucket can host your repository and enable collaboration with others.
- **Database Management:**
  - **phpMyAdmin:** A web-based tool for managing MySQL databases. It provides an intuitive interface for executing SQL queries, managing tables, and interacting with the database.
- **Testing and Debugging:**
  - **PHPUnit:** A unit testing framework for PHP that helps ensure the correctness of your code by running automated tests.
  - **Xdebug:** A PHP extension that facilitates debugging and profiling by providing features like breakpoints, stack traces, and variable inspection.

### 1.6) Importance of the work:

Here are some key aspects highlighting the significance of work in a hospital management system:

- **Efficient Patient Care:** A hospital management system streamlines various processes and workflows within a hospital, enabling efficient patient care. It facilitates tasks such as patient registration, appointment scheduling, electronic medical records management, billing, and inventory management. By automating these processes, healthcare professionals can focus more on providing quality care to patients.
- **Enhanced Workflow:** Hospital management systems help in optimizing workflows and improving overall operational efficiency. They provide a centralized platform for managing appointments, allocating resources, coordinating between departments, and tracking patient progress. This ensures smooth communication and coordination among staff members, leading to better patient outcomes.
- **Improved Decision Making:** Hospital management systems provide real-time access to patient data, medical history, diagnostic reports, and treatment plans. This enables healthcare providers to make informed decisions quickly. Having access to accurate and up-to-date information allows physicians to diagnose patients accurately, prescribe appropriate treatments, and monitor their progress effectively.
- **Enhanced Patient Safety:** Patient safety is a top priority in healthcare. A hospital management system helps in minimizing medical errors by maintaining accurate and complete electronic health records (EHRs).

- **Efficient Resource Management:** Hospital management systems aid in optimizing the utilization of resources within a healthcare facility. They provide tools for managing inventories, tracking medical supplies, and monitoring equipment maintenance. By efficiently managing resources, hospitals can reduce wastage, control costs, and ensure that necessary supplies are available when needed.
- **Data Analytics and Reporting:** Hospital management systems generate comprehensive reports and analytics based on the data collected. These insights help administrators and healthcare professionals identify trends, monitor performance metrics, and make data-driven decisions. Data analytics can be utilized to improve operational efficiency, allocate resources effectively, and enhance patient outcomes.
- **Compliance and Regulatory Requirements:** Hospitals need to comply with various regulatory requirements, such as patient data privacy (e.g., HIPAA in the United States). A hospital management system helps in ensuring compliance by securely storing patient information, restricting access to authorized personnel, and maintaining audit trails. It simplifies the process of generating reports required for regulatory audits and assessments.



## Chapter -2

### System Analysis

#### 2.1) Feasibility Study:

- **Improved Patient Care:** Implementing an efficient Hospital Management System (HMS) can lead to enhanced patient care. It enables streamlined processes for patient registration, appointment scheduling, and medical record management. This ensures that patients receive timely and accurate care, resulting in better health outcomes.
- **Enhanced Operational Efficiency:** An effective HMS automates various administrative tasks, such as billing, inventory management, and staff scheduling. This reduces manual errors, optimizes resource utilization, and improves overall operational efficiency within the hospital.
- **Streamlined Communication:** HMS facilitates seamless communication and collaboration among healthcare providers, departments, and patients. It allows for easy sharing of medical information, test results, and treatment plans, leading to improved coordination and continuity of care.
- **Effective Data Management:** Hospitals generate and handle large amounts of data on a daily basis. An HMS helps in organizing and managing this data, including electronic health records, diagnostic reports, and patient demographics. It ensures data security, accessibility, and supports data-driven decision-making.
- **Financial Management:** HMS plays a crucial role in managing the financial aspects of a hospital. It enables accurate billing, insurance claims processing, and revenue management. By automating financial processes, the system helps to reduce billing errors, track payments, and ensure financial stability.
- **Regulatory Compliance:** Healthcare organizations need to comply with various regulations and standards. An HMS helps in maintaining compliance with legal and regulatory requirements, such as data privacy (e.g., HIPAA), quality standards, and reporting obligations.
- **Scalability and Future Growth:** Hospitals need systems that can scale and adapt to their evolving needs. An HMS provides the flexibility to accommodate growing patient volumes, new services, and technological advancements. It allows for future growth and expansion of the hospital's operations.

## **2.2) Analysis Methodology**

STAGES OF ITERATIVE MODEL: Use Waterfall SDLC (Software Development Life Cycle)

### **Phases of Life Cycle Model**

- **Requirement Analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. It helps in specifying hardware and system requirements and helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment:** This will evaluate the outcome of the processing whether the application is running successfully or not.
- **Maintenance:** Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

### ➤ **Reason for choosing iterative model:**

- As we are new to this it is very difficult to analyze all requirements at once. Building the whole application at once is not possible for us.
- So, by dividing the Project into small parts and implementing them makes easy for us to make this application.

### ➤ **Advantages of the model:**

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- It works well for smaller projects where requirements are very well understood.

### 2.3) **Choice of Platform:**

#### ➤ **Hardware Components**

- Hard Disk: - 1000 GB
- Processor: INTEL CORE i5
- RAM: - 8 GB
- Front end: - HTML, CSS, JavaScript
- Back end: - PHP version 3.0, MySQL

#### ➤ **Software Specifications:**

- **Operating System:** Window/ Linux operating system
- **(HTML)** elements are the building blocks of HTML pages. With HTML constructs, images, and other objects such as interactive forms may be embedded into the rendered page.
- **(CSS)** enable the separation of content and presentation, including layout, colors, fonts, etc. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.
- **(JavaScript)** is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event- driven, functional, and imperative programming styles.
- **(Bootstrap)** is an HTML, CSS and JS library that focuses on simplifying the primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font, and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements.
- **(Backend)-** PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. It allows web developers to create dynamic content that interacts with databases like MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

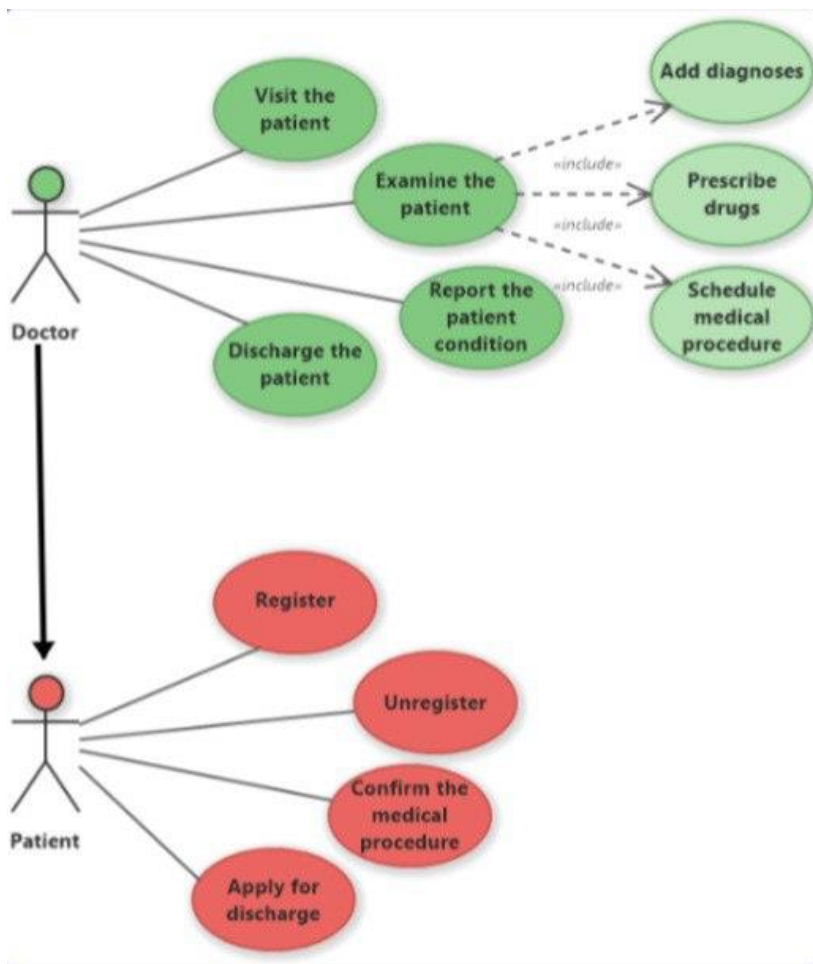
## Chapter-3

### System Design

- **Requirements Gathering:** Begin by understanding the requirements of the hospital and stakeholders. Conduct interviews and meetings with hospital administrators, doctors, nurses, and other staff members to identify their needs..
- **System Analysis:** Perform a comprehensive analysis of the existing hospital processes and workflows. Identify areas where the HMS can bring improvements and streamline operations. Analyze data flow, user roles and permissions, integration requirements, and any specific customization needs.
- **System Design:** Based on the gathered requirements and analysis, create a high-level system design. Define the overall architecture of the HMS, including modules, subsystems, and their interconnections. Identify the necessary data models, user interfaces, and integration points with other systems.
- **Database Design:** Design the database structure and schema to efficiently store and manage data. Define the tables, relationships, and data attributes required to support the functionalities of the HMS. Ensure proper normalization, data integrity, and security measures.
- **User Interface Design:** Create a user-friendly and intuitive interface for different user roles within the hospital. Consider the specific needs of doctors, nurses, administrators, and patients. Design screens, forms, and menus that provide easy navigation, data entry, and information retrieval.
- **System Prototyping:** Develop prototypes or mock-ups of the HMS to validate the design with stakeholders. Use prototyping tools or development frameworks to create interactive representations of the system's functionalities. Gather feedback and make necessary adjustments before proceeding to the development phase.
- **Deployment and Training:** Deploy the HMS in the hospital's environment and infrastructure. Ensure proper installation, configuration, and data migration. Provide comprehensive training to the hospital staff on how to use the system effectively and efficiently.

### 3.2) Database Design:

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data into the database model. A database management system manages the data accordingly.



**Fig 1.1 (Use case Design)**

### ▪ **ER D Diagram:**

The Entity Relationship Diagram is used to represent the relationship between entities in the table. The following symbols are used in ER-diagram:





Symbols	Purpose
	Represent Entity Set
	Represent Attributes
	Represent Relationship Sets
	Represent Line flow

Table 0.1 (ER Symbols)

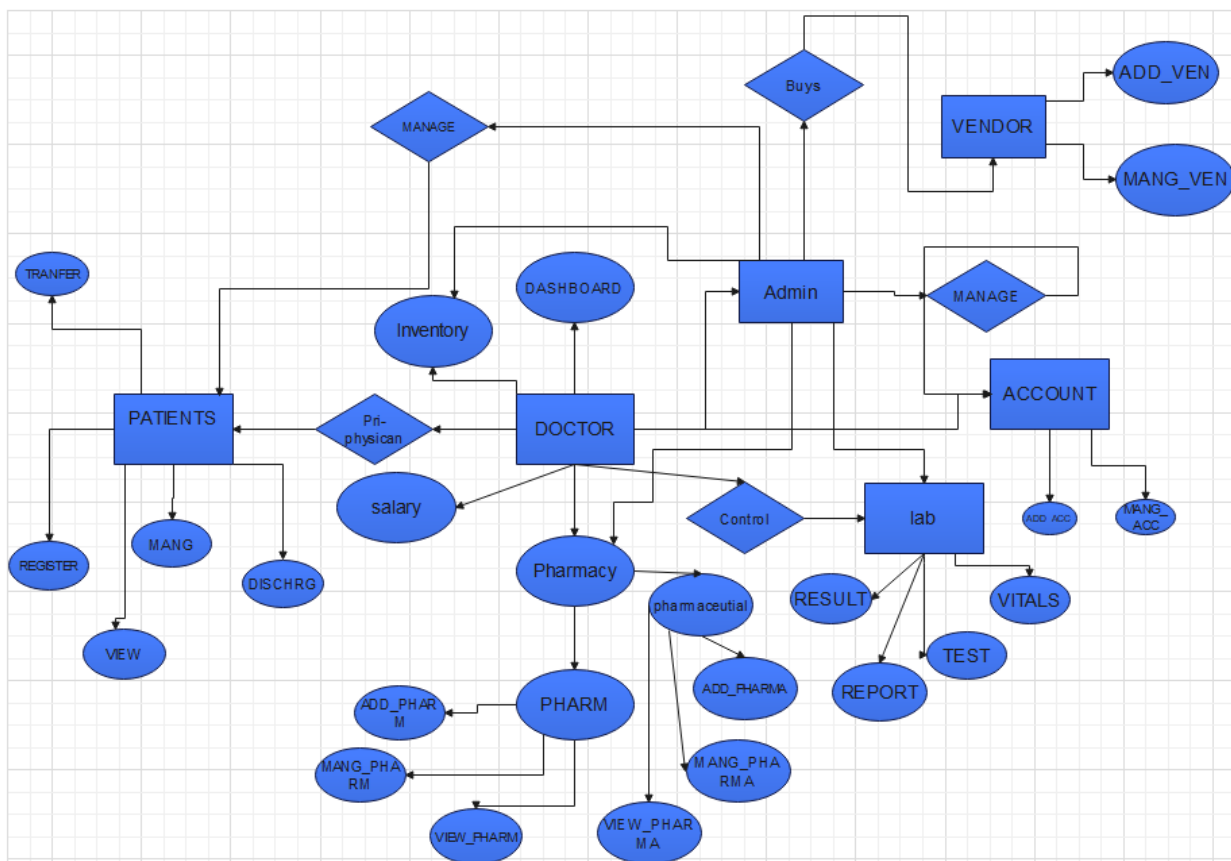
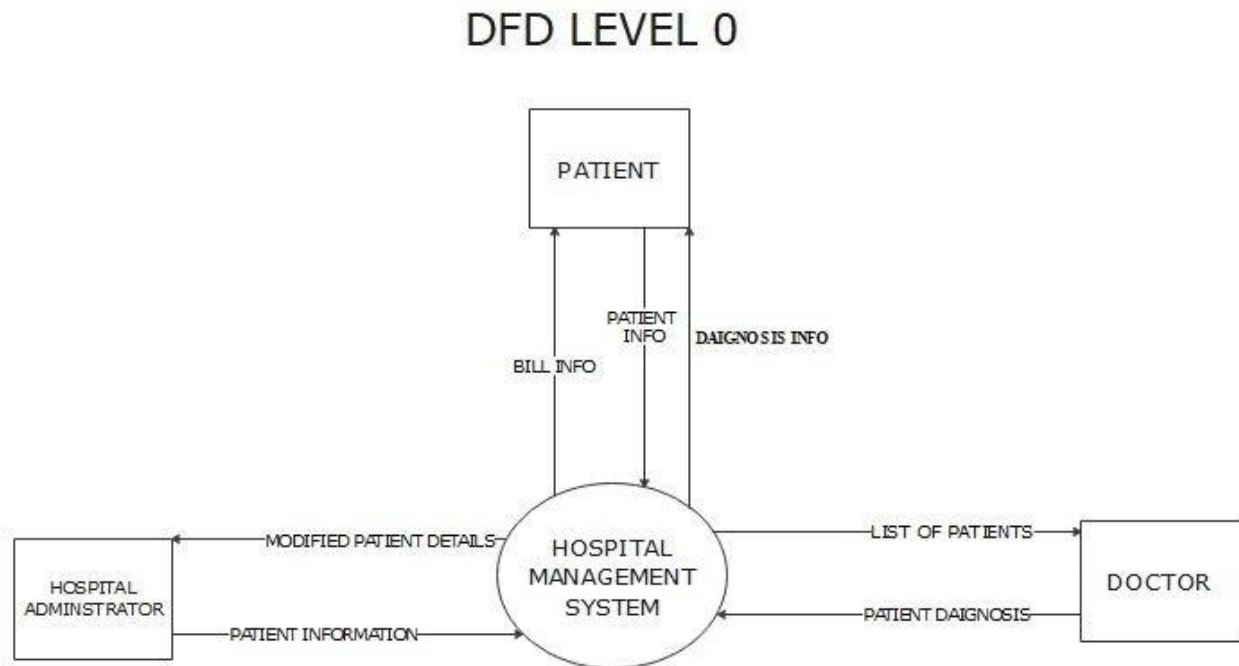


Fig 1.2 (ERD Diagram)

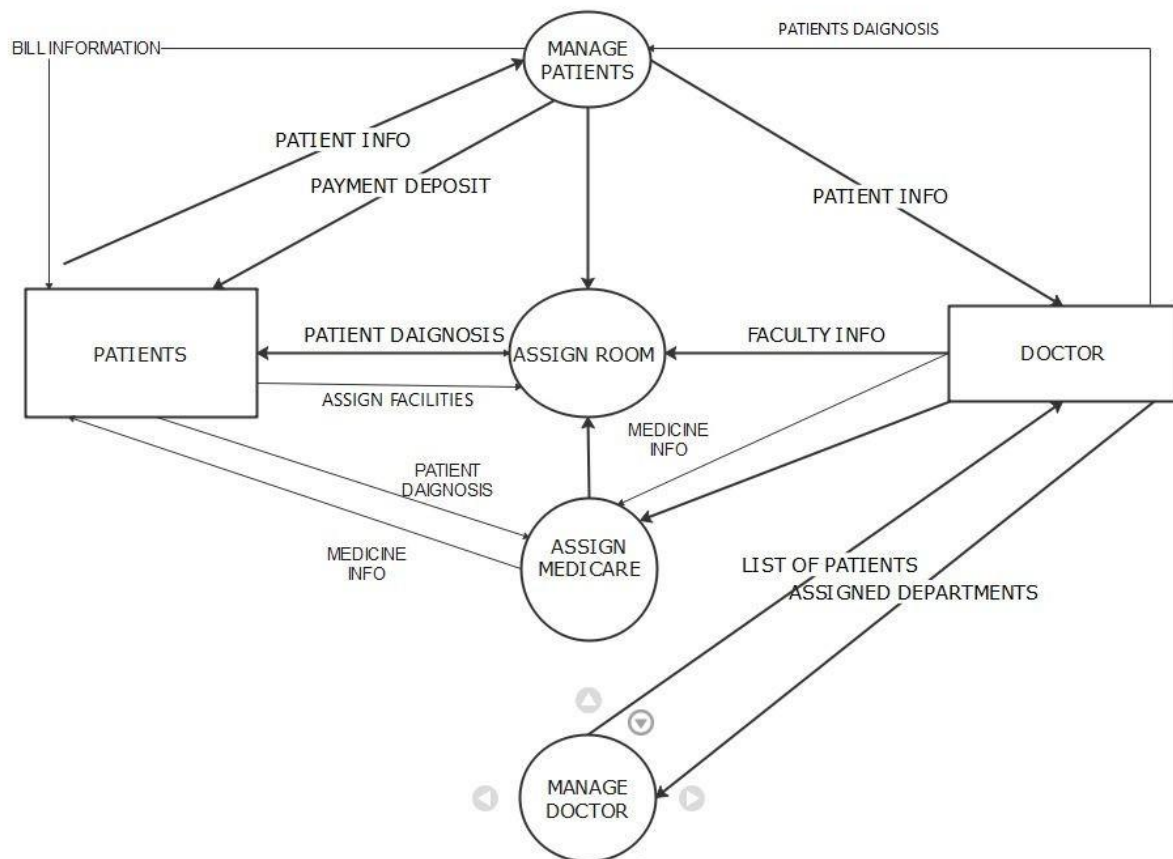
- **DFD:**

A data flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow – there are no decision rules and no loops.



**Fig 1.3 (Level-0 DFD)**

The illustration presents the main process in a single node to introduce the project context. This context explains how the project works in just one look. The user feeds data into the system and then receives the output from it.



**Fig 1.4 (Level-1 DFD)**

The designed diagram portrays four different scenarios: customer information management, bank records management, transaction monitoring and management, and customer status management.

Firstly, the flow of data starts from the bank personnel and members/client. Then the system caters to the transaction. This idea was based on bank management processes or transactions. You can also see the data store used or the database. The database is also used in storing users' data inputs. Then it serves as the source of outputs.



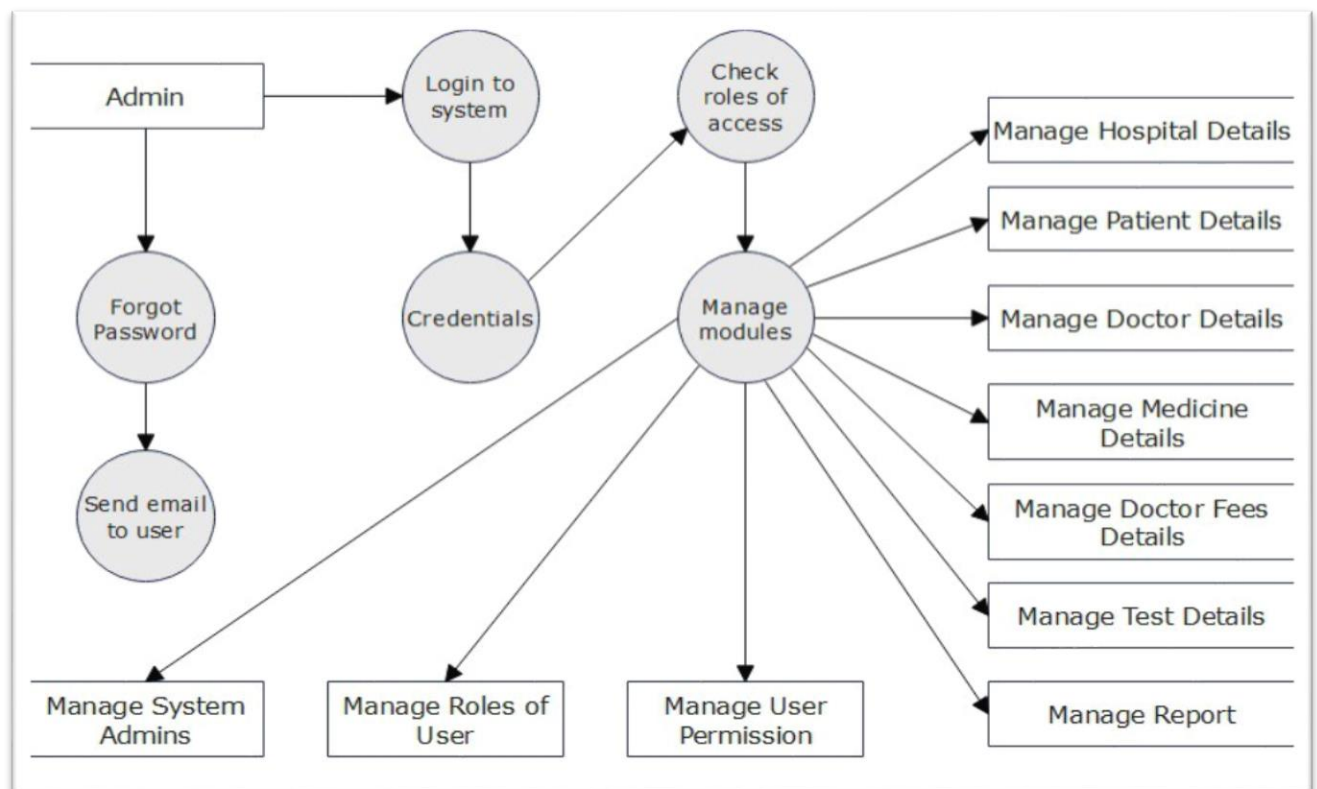


Fig 1.5 (Level -2 DFD)

### 3.3) Input Design:

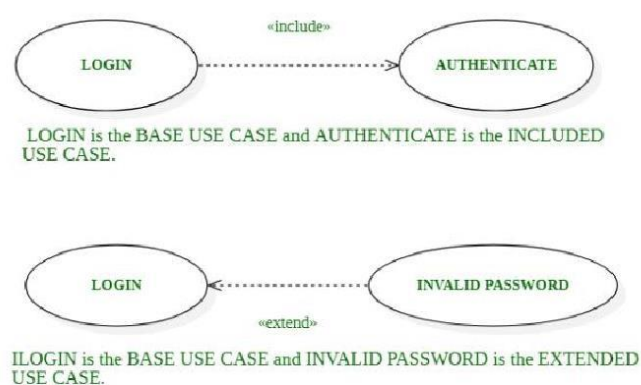


Fig 1.6 (Input design)

### 3.4) Output Design:

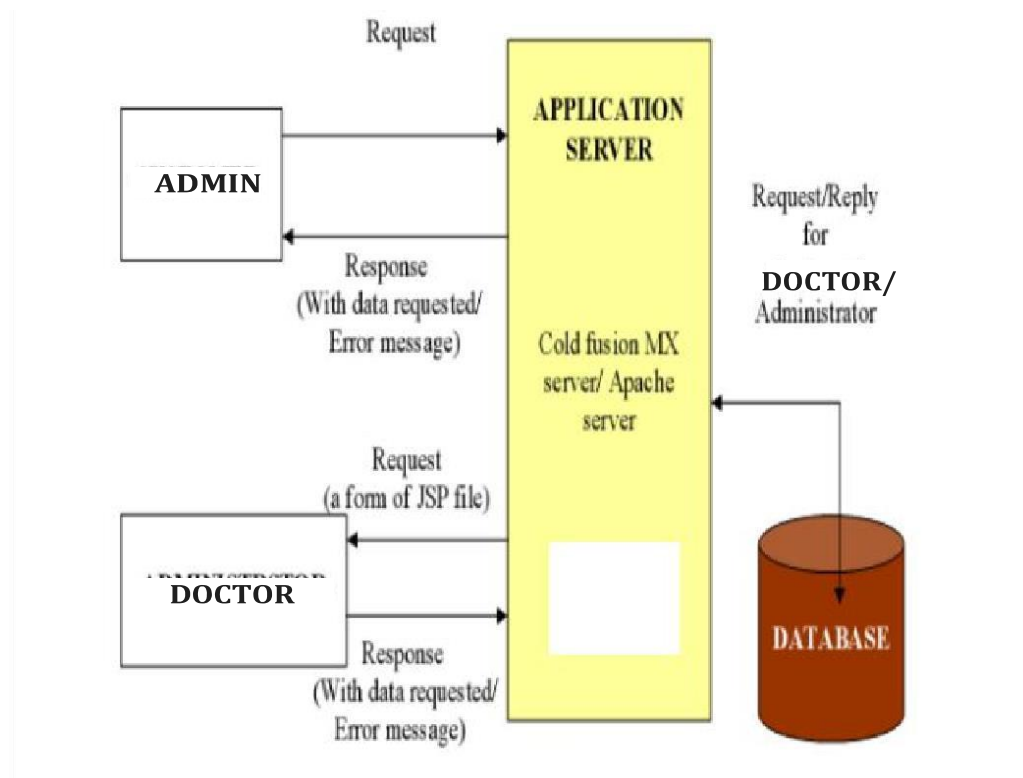


Fig 1.7 (output design)

### 3.5) Code Design & Development:

- **Input & Output Interferences:**

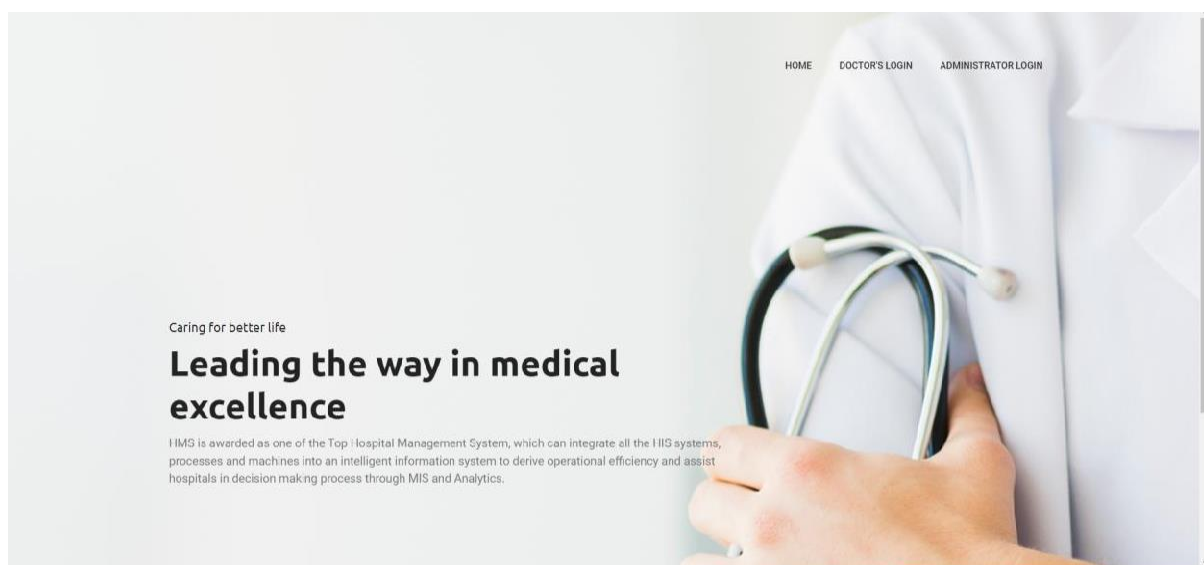


Fig 1.8 (Log in page)

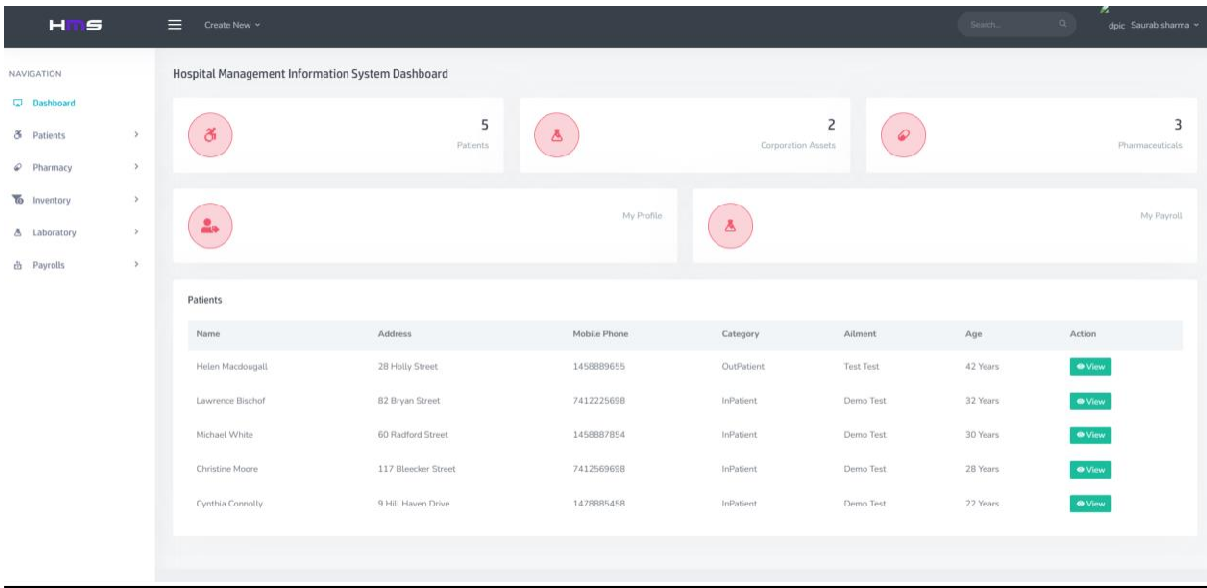


Fig 1.9 (Doctor Dashboard)

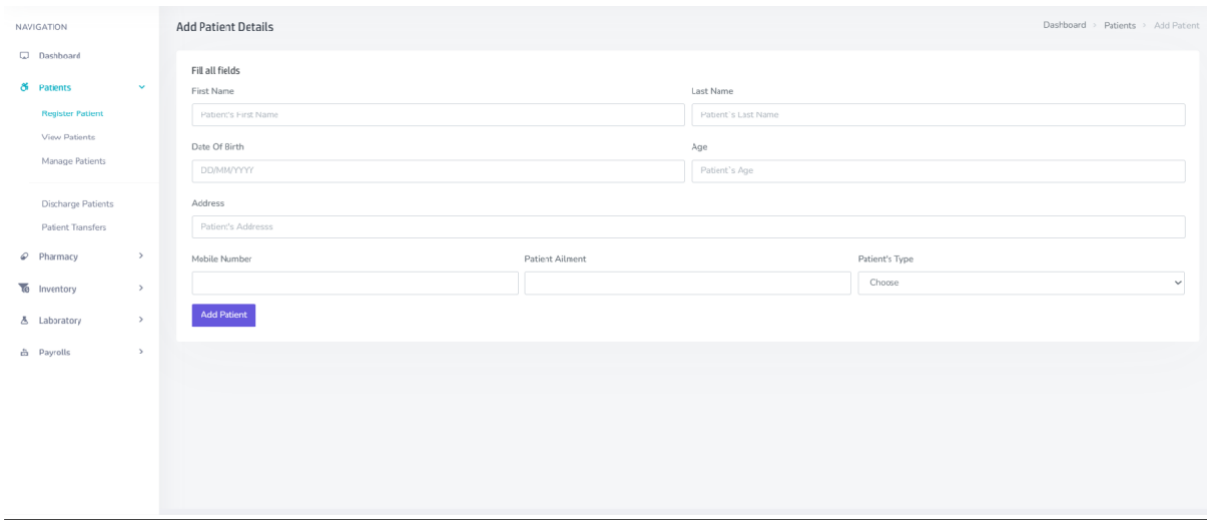


Fig 2.0 (Adding patient details)

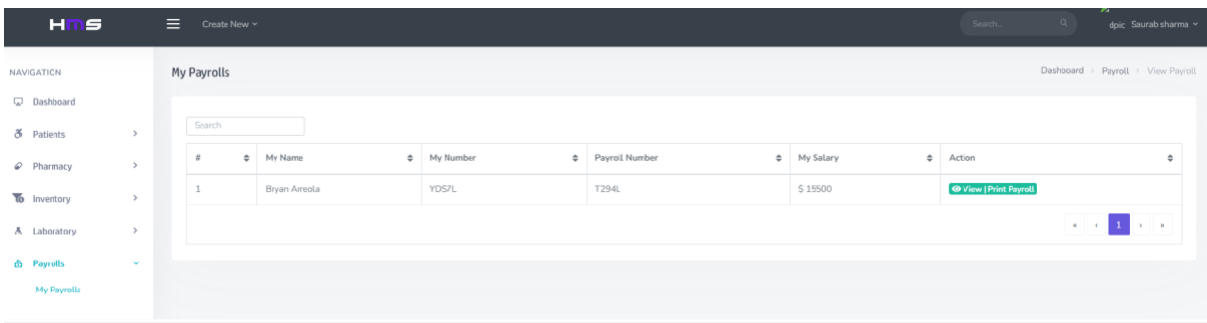


Fig 2.1 (payrolls)

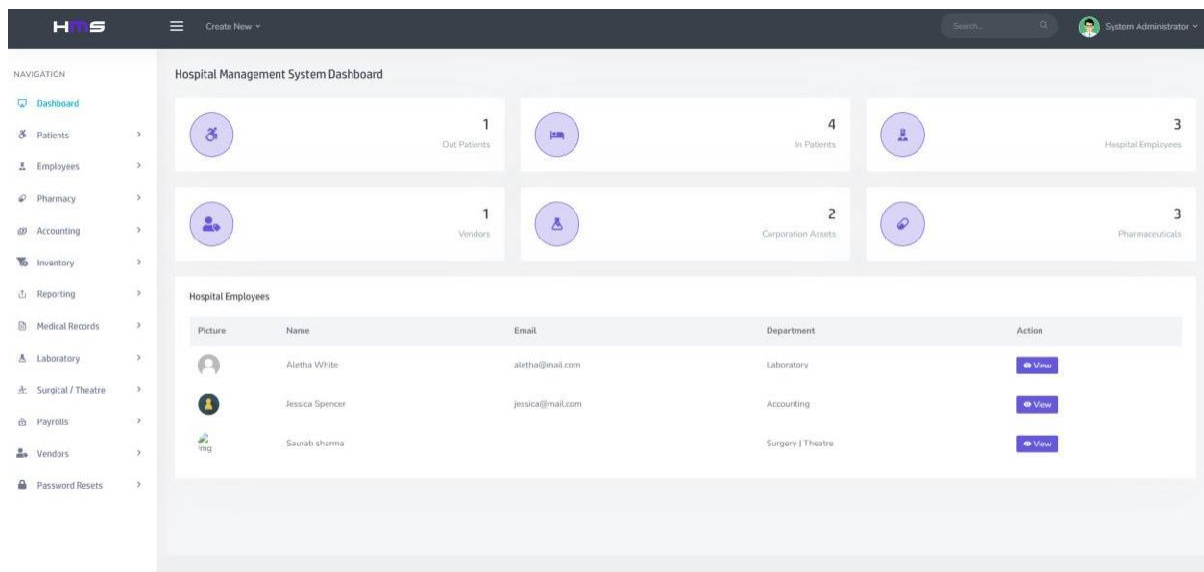


Fig 2.2 (Admin Dashboard)

The screenshot shows the 'Create A Pharmaceutical Category' form. The top navigation bar is identical to the previous figure. The left sidebar highlights the 'Pharmacy' section, with sub-items: Add Pharm Category, View Pharm Category, and Manage Pharm Category. The main content area is titled 'Create A Pharmaceutical Category' and includes a breadcrumb trail: Dashboard > Pharmaceuticals > Add Pharmaceutical Category. The form contains the following fields:

- Fill all fields:**
  - Pharmaceutical Category Name:
  - Pharmaceutical Vendor:
- Pharmaceutical Category Description:** A rich text editor with a toolbar (Bold, Italic, Underline, Bulleted List, Numbered List, Link, Unlink, Undo, Redo) and a large text area.
- Add Category:** A green button at the bottom left of the form.

Fig 2.3 (creating pharmaceutical category)

## Database Interferences

	ad_id	ad_fname	ad_lname	ad_email	ad_pwd	ad_dpvc
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	System	Administrator	saurab@162	4c7f5919e957f354d57243d37f223cf31e9e7181	doc-icon.png
<input type="checkbox"/> Check all	With selected: <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete <input type="checkbox"/> Export					

Fig 2.4(Admin Details)

	eqp_id	eqp_code	eqp_name	eqp_vendor	eqp_desc	eqp_dept	eqp_status	eqp_qty
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	178640239	TestTubes	Casico	<p>Testtubes are used to perform lab tests--</p>	Laboratory	Functioning	700000
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	052367981	Surgical Robot	Nexus	<p>Surgical Rcbots aid in surgery process.</p>	Surgical   Theatre	Functioning	100
<input type="checkbox"/> Check all	With selected: <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete <input type="checkbox"/> Export							

Fig 2.5(Different ean code)

	doc_id	doc_fname	doc_lname	doc_email	doc_pwd	doc_dept	doc_number	doc_dpvc
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	Alcitha	White	alcitha@mail.com	dce0b27ba675df41e9cc07af80cc59c475810824	Laboratory	BKTWQ	defaultimg.jpg
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	Saurab	sharma		53c3b5386c486feb662a0785f340938f518c547f	Surgery   Theatre	YDS7L	
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	12	Jessica	Spencer	jessica@mail.com	dce0b27ba675df41e9cc07af80cc59c475810824	Accounting	SVIFT	usrpic.png
<input type="checkbox"/> Check all	With selected: <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete <input type="checkbox"/> Export							

Fig 2.6(Doctors log in details)

	pay_id	pay_number	pay_doc_name	pay_doc_number	pay_doc_email	pay_emp_salary	pay_date_generated	pay_status	pay_descr
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	HJT13	Henry Doc	N8TD	hcnryd@hms.org	7555	2022-10-20 22:44:18.3708	Paid	<p> Lorem ipsum dolor sit amet, consectetur adipisicing...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	T29ML	Bryan Arreola	YDS7L	bryan@mail.com	15500	2022-10-20 22:44:50.5583	NULL	<p>demo demo demo demo demo</p>
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	3UOXY	Jessica Spencer	SVIFT	jessica@mail.com	4150	2022-10-22 16:34:36.9626	NULL	<p> This is a demo payroll description for test!! </p>
<input type="checkbox"/> Check all	With selected: <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete <input type="checkbox"/> Export								
<input type="checkbox"/> Show all	Number of rows: 25 Filter rows: Search this table Sort by key: None								

Fig 2.7(Payrolls details)

	pres_id	pres_patname	pres_pat_age	pres_pat_number	pres_number	pres_pat_addr	pres_pat_type	pres_date	pres_pat_ailment	pres_ins
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	Mart Developers	23	8F8-U	J9DCE	127001 LocalHost	InPatient	2020-01-11 18:02:39.6953	Fever	<ul><li><a href="https://www.medicinenewstoday.com/...</li></ul>
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	John Doe	30	RAV6C	HZQ8J	12 900 NVE	OutPatient	2020-01-11 18:38:46.7358	Malaria	<ul><li>Combination of atovaquone and proguanil [M...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	Lorem Ipsum	10	7EWOL	HQC3D	12 9001 Machakos	OutPatient	2020-01-13 17:19:30.3702	Flu	<ul><li><a href="https://www.google.com/search?cl...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	Christine Moore	28	11LGL	U5Y2s	11/ Blacker Street	InPatient	2022-10-22 16:27:10.7496	Demo Test	<ul><li>This is a demo prescription.</li></ul> This is...
<input type="checkbox"/> Check all	With selected: <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete <input type="checkbox"/> Export									
<input type="checkbox"/> Show all	Number of rows: 25 Filter rows: Search this table Sort by key: None									

Fig 2.8 (Prescriptions details)

	vit_id	vit_number	vit_pat_number	vit_bodytemp	vit_heartpulse	vit_resprate	vit_bloodpress	vit_daterec
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	1KB9V	3Z14K	38	77	12	90/60	2022-10-18 22:40:16.904915
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	ELYOM	BKTWQ	38	88	12	110/80	2022-10-18 07:19:55.814783
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	ALQJ8	YDS7L	36	72	15	90/60	2022-10-18 23:12:17.500662
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	MS2OJ	4TLG0	37	70	15	120/80	2022-10-22 16:31:52.148658
<input type="checkbox"/> Check all	With selected: <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete <input type="checkbox"/> Export							
<input type="checkbox"/> Show all	Number of rows: 25 Filter rows: Search this table Sort by key: None							

Fig 2.9 (vitals details)

### 3.1) Code Design & Development:

#### ▪ His admin view.php

```
<?php
    session_start();
    include('assets/inc/config.php');
    include('assets/inc/checklogin.php');
    check_login();
    $aid=$_SESSION['ad_id'];
?>

<!DOCTYPE html>
<html lang="en">

<?php include('assets/inc/head.php');?>

    <body>

        <!-- Begin page -->
        <div id="wrapper">

            <!-- Topbar Start -->
            <?php include('assets/inc/nav.php');?>
            <!-- end Topbar -->

            <!-- ===== Left Sidebar Start ===== -->
            <?php include("assets/inc/sidebar.php");?>
            <!-- Left Sidebar End -->

            <!-- ===== -->
            <!-- Start Page Content here -->
            <!-- ===== -->

            <div class="content-page">
                <div class="content">

                    <!-- Start Content-->
                    <div class="container-fluid">

                        <!-- start page title -->
                        <div class="row">
                            <div class="col-12">
                                <div class="page-title-box">
                                    <div class="page-title-right">
                                        <ol class="breadcrumb m-0">
                                            <li class="breadcrumb-item"><a
href="javascript: void(0);">Dashboard</a></li>
                                            <li class="breadcrumb-item"><a
href="javascript: void(0);">Patients</a></li>
                                            <li class="breadcrumb-item active">View
Patients</li>
                                        </ol>
                                    </div>
                                    <h4 class="page-title">Patient Details</h4>
                                </div>
                            </div>
                        </div>
                    </div>
                    <!-- end page title -->
                </div>
            </div>
        </div>
    </body>
</html>
```

```

<div class="row">
  <div class="col-12">
    <div class="card-box">
      <h4 class="header-title"></h4>
      <div class="mb-2">
        <div class="row">
          <div class="col-12 text-sm-center form-
inline" >
            <div class="form-group mr-2"
style="display:none">
              <select id="demo-foo-filter-
status" class="custom-select custom-select-sm">
                <option value="">Show
all</option>
                <option
value="Discharged">Discharged</option>
                <option
value="OutPatients">OutPatients</option>
                <option
value="InPatients">InPatients</option>
              </select>
            </div>
            <div class="form-group">
              <input id="demo-foo-search"
type="text" placeholder="Search" class="form-control form-control-sm"
autocomplete="on">
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
  <div class="table-responsive">
    <table id="demo-foo-filtering" class="table
table-bordered toggle-circle mb-0" data-page-size="7">
      <thead>
        <tr>
          <th>#</th>
          <th data-toggle="true">Name</th>
          <th data-hide="phone">Number</th>
          <th data-hide="phone">Address</th>
          <th data-hide="phone">Phone</th>
          <th data-hide="phone">Age</th>
          <th data-hide="phone">Category</th>
          <th data-hide="phone">Action</th>
        </tr>
      </thead>
    </table>
  </div>
</div>
<?php
/*
 *get details of allpatients
 *
 */
$ret="SELECT * FROM his_patients
ORDER BY RAND() ";
docs randomly
//sql code to get to ten

$stmt= $mysqli->prepare($ret) ;
$stmt->execute() ;//ok
$res=$stmt->get_result();
$cnt=1;

```

```
while($row=$re  
s-  
>fetch_object(  
))
```



```

        {
            ?>
            <tbody>
            <tr>
                <td><?php echo $cnt;?></td>
                <td><?php echo $row-
>pat_fname;?> <?php echo $row->pat_lname;?></td>
                <td><?php echo $row-
>pat_number;?></td>
                <td><?php echo $row-
>pat_addr;?></td>
                <td><?php echo $row-
>pat_phone;?></td>
                <td><?php echo $row->pat_age;?>
Years</td>
                <td><?php echo $row-
>pat_type;?></td>
                <td><a
href="his_admin_view_single_patient.php?pat_id=<?php echo $row-
>pat_id;?>&&pat_number=<?php echo $row->pat_number;?>" class="badge badge-
success"><i class="mdi mdi-eye"></i> View</a></td>
            </tr>
            </tbody>
            <?php $cnt = $cnt +1 ; }?>
            <tfoot>
            <tr class="active">
                <td colspan="8">
                    <div class="text-right">
                        <ul class="pagination
pagination-rounded justify-content-end footable-pagination m-t-10 mb-0"></ul>
                    </div>
                </td>
            </tr>
            </tfoot>
            </table>
            </div> <!-- end .table-responsive-->
        </div> <!-- end card-box -->
    </div> <!-- end col -->
</div>
<!-- end row -->

</div> <!-- container -->

</div> <!-- content -->

<!-- Footer Start -->
<!-- end Footer -->

</div>

<!-- ===== -->
<!-- End Page content -->
<!-- ===== -->

</div>
<!-- END wrapper -->

<!-- Right bar overlay-->
<div class="rightbar-overlay"></div>

```

```

        <!-- Vendor js -->
        <script src="assets/js/vendor.min.js"></script>

        <!-- Footable js -->
        <script src="assets/libs/footable/footable.all.min.js"></script>

        <!-- Init js -->
        <script src="assets/js/pages/foo-tables.init.js"></script>

        <!-- App js -->
        <script src="assets/js/app.min.js"></script>

    </body>

</html>

```

### ▪ His Admin Accounting records .Php

```

<?php
    session_start();
    include('assets/inc/config.php');
    include('assets/inc/checklogin.php');
    check_login();
    $aid=$_SESSION['ad_id'];

?>

<!DOCTYPE html>
<html lang="en">

<?php include('assets/inc/head.php');?>

    <body>

        <!-- Begin page -->
        <div id="wrapper">

            <!-- Topbar Start -->
            <?php include('assets/inc/nav.php');?>
            <!-- end Topbar -->

            <!-- ===== Left Sidebar Start ===== -->
            <?php include("assets/inc/sidebar.php");?>
            <!-- Left Sidebar End -->

            <!-- ===== -->
            <!-- Start Page Content here -->
            <!-- ===== -->

            <div class="content-page">
                <div class="content">

                    <!-- Start Content-->
                    <div class="container-fluid">

                        <!-- start page title -->
                        <div class="row">

```

```

        <div class="col-12">
            <div class="page-title-box">
                <div class="page-title-right">
                    <ol class="breadcrumb m-0">
                        <li class="breadcrumb-item"><a
href="javascript: void(0);">Dashboard</a></li>
                        <li class="breadcrumb-item"><a
href="javascript: void(0);">Reporting</a></li>
                        <li class="breadcrumb-item
active">Accounts</li>
                    </ol>
                </div>
            </div>
            <div class="page-title">Accounts Records</div>
        </div>
    </div>
</div>
<!-- end page title -->

<div class="row">
    <div class="col-12">
        <div class="card-box">
            <h4 class="header-title"></h4>
            <div class="mb-2">
                <div class="row">
                    <div class="col-12 text-sm-center form-
inline" >
                        <div class="form-group mr-2"
style="display:none">
                            <select id="demo-foo-filter-
status" class="custom-select custom-select-sm">
                                <option value="">Show
                                <option
                                <option
                                <option
                                <option
                                </select>
                            </div>
                        <div class="form-group">
                            <input id="demo-foo-search"
type="text" placeholder="Search" class="form-control form-control-sm"
autocomplete="on">
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="table-responsive">
            <table id="demo-foo-filtering" class="table
table-bordered toggle-circle mb-0" data-page-size="7">
                <thead>
                    <tr>
                        <th>Name</th>
                        <th>Number</th>
                        <th>Amount</th>
                    </tr>
                </thead>
            </table>
        </div>
    </div>
</div>

```

```
<th>#</th>
<th
data-
toggle
="true
">Acco
unt
```

```
<th
data-
hide=
"phon
e">Ac
count
```

```
<th
data-
hide=
"phon
e">Ac
count
```

```

Type</th>
<th data-hide="phone">Account
<th data-hide="phone">Action</th>
</tr>
</thead>
<?php
$ret="SELECT * FROM his_accounts ORDER
BY RAND() ";

$stmt= $mysqli->prepare($ret) ;
$stmt->execute() ;//ok
$res=$stmt->get_result();
$cnt=1;
while($row=$res->fetch_object())
{
?>

<tbody>
<tr>
<td><?php echo $cnt;?></td>
<td><?php echo $row->acc_name;?></td>
<td><?php echo $row-
>acc_number;?></td>
<td>$ <?php echo $row-
>acc_amount;?></td>
<td><?php echo $row->acc_type;?></td>
<td>
<a
href="his_admin_view_single_account.php?acc_number=<?php echo $row->acc_number;?>"
class="badge badge-success"><i class="fas fa-eye "></i> View</a>
</td>
</tr>
</tbody>
<?php $cnt = $cnt +1 ; }?>
<tfoot>
<tr class="active">
<td colspan="8">
<div class="text-right">
<ul class="pagination
pagination-rounded justify-content-end footable-pagination m-t-10 mb-0"></ul>
</div>
</td>
</tr>
</tfoot>
</table>
</div> <!-- end .table-responsive-->
</div> <!-- end card-box -->
</div> <!-- end col -->
</div>
<!-- end row -->

</div> <!-- container -->

</div> <!-- content -->

<!-- Footer Start -->
<?php include('assets/inc/footer.php');?>
<!-- end Footer -->

</div>

```

```

        <!-- ===== -->
        <!-- End Page content -->
        <!-- ===== -->

    </div>
    <!-- END wrapper -->

    <!-- Right bar overlay-->
    <div class="rightbar-overlay"></div>

    <!-- Vendor js -->
    <script src="assets/js/vendor.min.js"></script>

    <!-- Footable js -->
    <script src="assets/libs/footable/footable.all.min.js"></script>

    <!-- Init js -->
    <script src="assets/js/pages/foo-tables.init.js"></script>

    <!-- App js -->
    <script src="assets/js/app.min.js"></script>

</body>

</html>

```

## ▪ Configure.php:

```

<?php
    $dbuser="root";
    $dbpass="";
    $host="localhost";
    $db="Hoaspital management ";
    $mysqli=new mysqli($host,$dbuser, $dbpass, $db);

```

## Chapter - 4

### Test & Implementation

#### 4.1) Testing Methodology:

- **Test Planning:**
  - Define testing objectives, scope, and deliverables.
  - Identify test resources, including test environments, tools, and personnel.
  - Determine the testing timeline and prioritize testing activities.
- **Test Requirements:**
  - Review the system requirements and design documents.
  - Define test scenarios and test cases based on functional and non-functional requirements.
  - Ensure that the test cases cover all the critical functionalities and edge cases.
- **Test Environment Setup:**
  - Set up the necessary test environments, including hardware, software, and network configurations.
  - Prepare test data that represents real-world scenarios and includes both valid and invalid inputs.
- **Test Execution:**
  - Execute the test cases based on the defined test scenarios.
  - Record the actual results and compare them against the expected results.
  - Identify and report any defects or discrepancies found during testing.
- **Types of Testing:**
  - Functional Testing: Verify that the HMS functions correctly according to the specified requirements.
  - Integration Testing: Validate the interaction between different modules and subsystems of the HMS.
  - Performance Testing: Assess the system's performance under different load conditions and stress levels.
  - Security Testing: Evaluate the system's security measures, including data encryption, access controls, and vulnerability scanning.
  - Usability Testing: Ensure that the HMS is user-friendly, intuitive, and meets the needs of different user roles.
  - Regression Testing: Re-test previously tested functionalities after changes or fixes to ensure that no new issues are introduced.
- **Test Reporting and Defect Management:**
  - Document and track all test activities, including test execution results, identified defects, and their status.
  - Prioritize and assign severity levels to the reported defects.
  - Collaborate with the development team to resolve the defects and retest them.

- **Test Completion and Sign-off:**
  - Evaluate the overall test coverage and results.
  - Assess whether the HMS meets the defined acceptance criteria and quality standards.
  - Obtain necessary approvals and sign-off from stakeholders before proceeding to the deployment phase.
  
- **Unit Testing:**
  
- **Testable Units:** Identify the units or components of the HMS that can be tested in isolation. This can include individual functions, methods, classes, or modules.
- **Test Framework and Tools:** Select a suitable unit testing framework and tools that are compatible with the programming language and technologies used in the HMS development. Common unit testing frameworks include JUnit for Java, NUnit for .NET, and pytest for Python.
- **Test Cases:** Define test cases for each unit, covering different scenarios, inputs, and boundary conditions. Test cases should verify the expected behavior and outputs of the unit.
- **Test Data:** Prepare relevant and representative test data to be used during unit testing. This includes both valid and invalid inputs, edge cases, and corner cases that the unit should handle correctly.
- **Test Execution:** Execute the unit tests for each unit in isolation, ensuring that all relevant test cases are run. Monitor the test execution and record the results for further analysis.
- **Assertions:** Use assertions or assertions libraries provided by the testing framework to compare the actual results of the unit with the expected results defined in the test cases. Assertions help validate that the unit is functioning as intended.
- **Test Coverage:** Aim for high test coverage by ensuring that a significant portion of the codebase is covered by unit tests. This helps identify potential issues and provides confidence in the stability and reliability of the units.
- **Test Automation:** Automate the execution of unit tests to facilitate frequent testing, especially during development iterations. Continuous Integration (CI) tools can be used to automatically run unit tests whenever code changes are made.
- **Test Maintenance:** Update and maintain the unit tests as the HMS evolves. Whenever changes are made to units or new features are added, ensure that the corresponding unit tests are updated accordingly.
- By conducting thorough unit testing, you can identify and fix defects early in the development process, enhance the quality and reliability of the units, and support the overall stability of the Hospital Management System.



➤ **Module Testing:**

Module testing, also known as component testing or integration testing, focuses on testing the interaction and integration between different modules or components of a Hospital Management System (HMS). It verifies that the modules function correctly when combined and communicate with each other as expected. Here are the key aspects of module testing:

- **Identify Modules:** Determine the modules or components of the HMS that need to be tested together. This typically involves identifying the interconnected modules that interact to perform specific functionalities.
- **Test Scenarios:** Define test scenarios that cover various integration scenarios. Consider the inputs, outputs, and data flow between the modules. Test cases should include positive and negative scenarios, boundary conditions, and exception handling.
- **Test Data:** Prepare test data that represents real-world scenarios and covers a wide range of possibilities. This includes valid and invalid inputs, as well as edge cases to ensure the modules handle different data combinations correctly.
- **Test Environment:** Set up a test environment that closely resembles the production environment. This includes configuring databases, external services, and simulated dependencies that the modules rely on during integration.
- **Stubbing/Mocking:** If certain modules are not yet available or depend on external systems that are not easily accessible during testing, use stubs or mocks to simulate their behavior and provide controlled responses.
- **Test Execution:** Execute the module tests by combining the relevant modules and simulating their interactions. Monitor the test execution and record the results for analysis.
- **Integration Points:** Pay attention to the integration points between the modules, such as data exchange, message passing, API calls, or shared resources. Verify that the modules correctly handle data synchronization, error handling, and proper communication protocols.
- **Error Handling and Exception Handling:** Validate that the modules handle errors and exceptions gracefully. Test scenarios that simulate error conditions or unexpected inputs to ensure that the modules respond appropriately and provide meaningful error messages.
- **Test Coverage:** Aim for comprehensive test coverage by ensuring that all critical integration paths and scenarios are covered. This helps identify potential integration issues and increases confidence in the overall functionality of the HMS.
- **Test Reporting:** Document the test results, including any defects or issues found during module testing. Provide clear and detailed reports to facilitate troubleshooting and bug fixing.

➤ **Integration Testing:**

- **Identify Integration Points:** Identify the points where modules or components interact with each other. This includes data exchange, API calls, message passing, shared resources, or any other forms of communication.
- **Test Strategy:** Define the integration testing strategy, which can be either a top-down or bottom-up approach. In a top-down approach, higher-level modules are tested first, with stubs or simulated lower-level modules. In a bottom-up approach, lower-level modules are tested first, with drivers or simulated higher-level modules.
- **Test Cases:** Define test cases that cover the integration scenarios. These test cases should verify the correct flow of data and the interaction between modules. Include both positive and negative test cases, boundary cases, and exception handling scenarios.
- **Integration Environment:** Set up an integration testing environment that replicates the production environment as closely as possible. This includes configuring servers, databases, network connections, and any necessary external systems or services.
- **Stubbing/Mocking:** If certain modules or components are not yet available or depend on external systems that are not easily accessible during integration testing, use stubs or mocks to simulate their behavior and provide controlled responses.
- **Data Flow Testing:** Test the flow of data between different modules or subsystems. Verify that the data is correctly passed and transformed between components, and that the integrity of the data is maintained throughout the integration process.
- **Interface Testing:** Validate the interfaces between modules or components. This includes testing the input and output parameters, communication protocols, data formats, and any required data transformations.
- **Error Handling and Exception Testing:** Test error handling and exception scenarios during integration. Validate that the system handles errors and exceptions gracefully and provides meaningful error messages or appropriate actions.
- **Integration Order:** Determine the order in which the modules or components should be integrated and tested. This can be based on dependencies, critical functionalities, or any other logical sequence.
- **Compatibility Testing:** Ensure that the integrated system is compatible with the required hardware, software, and third-party systems or services. Test the integration across different platforms, browsers, operating systems, or versions to identify and address any compatibility issues.
- **Regression Testing:** Perform regression testing after integrating new modules or components to ensure that the existing functionalities are not affected and continue to work as expected.

➤ **System Testing:**

- **Test Environment:** Set up a dedicated test environment that closely resembles the production environment where the HMS will be deployed. This includes configuring servers, databases, network connections, and other necessary components.
- **Test Scenarios:** Define a set of test scenarios that cover the entire system functionality and various usage scenarios. Test cases should include end-to-end workflows, data flow, user interactions, and system behavior under normal and exceptional conditions.
- **Functional Testing:** Verify that the HMS functions correctly as a whole, including all the modules, subsystems, and integrations. Test the end-to-end processes, such as patient registration, appointment scheduling, billing, medication management, and reporting.
- **Performance Testing:** Evaluate the system's performance and scalability by simulating realistic workloads. This includes testing response times, throughput, resource utilization, and system stability under different load conditions.
- **Security Testing:** Assess the HMS's security measures to protect sensitive data and ensure compliance with privacy regulations. Test access controls, authentication mechanisms, data encryption, and vulnerability scanning to identify and address security vulnerabilities.
- **Usability Testing:** Evaluate the user-friendliness and ease of use of the HMS. Gather feedback from end-users, such as doctors, nurses, administrators, and patients, to assess the system's intuitiveness, efficiency, and overall user experience.
- **Compatibility Testing:** Ensure that the HMS is compatible with different browsers, operating systems, and devices commonly used by end-users. Test the system on different platforms to identify any compatibility issues and ensure consistent behavior across environments.
- **Error Handling and Recovery:** Test the system's error handling and recovery mechanisms. Validate that error messages are meaningful, exceptions are properly handled, and the system gracefully recovers from failures without data loss or corruption.
- **Regression Testing:** Perform regression testing by retesting previously tested functionalities after changes or fixes to ensure that no new issues are introduced. This helps ensure that the system remains stable and functional throughout the development and maintenance phases.
- **Test Reporting and Documentation:** Document and report the test results, including any defects or issues found during system testing. Provide clear and detailed reports to facilitate the resolution of identified issues.

### ➤ Whitebox/Blackbox Testing:

White-box and black-box testing are two different approaches to software testing, including the testing of a Hospital Management System (HMS). Here's a brief explanation of each:

#### ▪ **White-box Testing:**

White-box testing, also known as clear-box testing or structural testing, involves examining the internal structure, logic, and implementation details of the software being tested. Testers have access to the source code and use their knowledge of the system's internal workings to design test cases. The focus is on ensuring that all paths, statements, conditions, and branches of the code are tested thoroughly. Key aspects of white-box testing include:

- **Code Coverage:** The goal is to achieve high code coverage, which means exercising as much of the code as possible through the test cases.
- **Path Testing:** Testers analyze the different paths that can be taken within the code and design test cases to cover these paths.
- **Unit Testing:** White-box testing is commonly used at the unit testing level to test individual functions, methods, or classes.

#### ▪ **Black-box Testing:**

Black-box testing, also known as functional testing, focuses on testing the functionality of the system without considering its internal structure or implementation details. Testers have no knowledge of the internal workings of the system and treat it as a black box, testing only the inputs, outputs, and expected behavior. Key aspects of black-box testing include:

- **Test Scenarios:** Testers design test scenarios based on the system's requirements, specifications, or user stories, without any knowledge of the internal code.
- **Test Data:** Testers prepare test data representing different scenarios and inputs to verify the system's behavior.
- **Functional Validation:** Testers focus on validating that the system functions correctly and produces the expected outputs for different inputs and conditions.
- **User Perspective:** Black-box testing aims to ensure that the system meets user expectations and requirements without considering the internal implementation details.

Both white-box and black-box testing are important and complementary approaches to ensure the quality and reliability of software systems. White-box testing is more detailed and targets specific areas of the code, while black-box testing focuses on overall functionality and user experience. A combination of both approaches is often used to achieve comprehensive testing coverage for the HMS.

➤ **Acceptance Testing:**

Acceptance testing is a level of testing that aims to determine whether a Hospital Management System (HMS) meets the specified requirements and is ready for deployment and use in a real-world environment. It involves validating the system's functionality, usability, and compliance with the stakeholders' needs. Acceptance testing can be performed by end-users, clients, or other stakeholders who will be using the system. Here are the key aspects of acceptance testing:

- **User Acceptance Testing (UAT):** UAT focuses on validating the system from the end-users' perspective. It involves real users performing test scenarios and providing feedback on the system's usability, functionality, and fitness for their specific needs.
- **Test Environment:** Set up a test environment that closely resembles the production environment where the HMS will be deployed. This includes configuring servers, databases, network connections, and any necessary external systems or services.
- **Acceptance Criteria:** Define clear acceptance criteria based on the system requirements, user stories, or specifications. These criteria serve as the basis for determining whether the system meets the stakeholders' expectations.
- **Test Scenarios:** Design test scenarios that cover a wide range of typical and critical use cases. These scenarios should represent the day-to-day operations of the HMS and the workflows of different user roles, such as doctors, nurses, administrators, and patients.
- **Usability Testing:** Evaluate the system's usability and user experience. Testers should assess the system's intuitiveness, ease of use, efficiency, and overall satisfaction for the end-users.
- **Functional Validation:** Validate that the system functions correctly and meets the specified requirements. Testers should verify that the HMS performs the intended tasks accurately and produces the expected outputs.
- **Performance Testing:** Assess the system's performance and response times under expected workloads. Testers should verify that the system can handle the anticipated user load without significant performance degradation.
- **Security Testing:** Verify that the system complies with security requirements and protects sensitive data. Testers should assess access controls, authentication mechanisms, data encryption, and other security features.
- **Compliance Testing:** Ensure that the HMS meets regulatory, legal, and industry-specific requirements, such as privacy regulations or healthcare standards.
- **Test Reporting:** Document the test results, including any issues or defects found during acceptance testing. Provide clear and detailed reports to facilitate communication and decision-making for system readiness.

**4.2) Test Data & Test Cases:**

QA Tester's Log		Review comments from Bill incorporate in version 2.1							
Tester's Name		saurab	Date Tested		June 12, 2023	Test Case (Pass/Fail/Not		Pass	
S #	Prerequisites:				S #	Test Data			
1	Access to Chrome Browser				1	Userid = mg12345			
2					2	Pass = df12@434c			
3					3				
4					4				

**Fig 3.0 (Test Case 1 on log id)**

<b>Test Scenario</b>		Verify on entering valid userid and password, the customer can login					
<b>Step #</b>	<b>Step Details</b>	<b>Expected Results</b>	<b>Actual Results</b>		<b>Pass / Fail / Not executed / Suspended</b>		
1	Navigate to localhost/internet banking-php	Site should open	As Expected		Pass		
2	Enter Userid & Password	Credential can be entered	As Expected		Pass		
3	Click Submit	Cutomer is logged in	As Expected		Pass		

**Fig 3.1 (Test results or scenario of expected results and final results)**

		<b>ad_id</b>	<b>ad_fname</b>	<b>ad_lname</b>	<b>ad_email</b>	<b>ad_pwd</b>	<b>ad_dpvc</b>
<input type="checkbox"/>	Edit	Copy	Delete	1 System	Administrator saarab@162	4c7f5919e957f354d57243d37f223cf31e9e7181	doc-icon.png
	<input type="checkbox"/> Check all	With selected:		Edit	Copy	Delete	Export

**Fig 3.2 (Test Data for log in the customers to fetch data)**

### 4.3) Debugging:

- **Issue Identification:** Identify and reproduce the issue or bug within the HMS. This can be done by observing the system's behavior, gathering error messages or logs, and analyzing user-reported problems.
- **Issue Reproduction:** Try to reproduce the issue consistently by following specific steps or providing certain inputs. Reproducing the issue helps in understanding its cause and allows for more effective debugging.
- **Root Cause Analysis:** Analyze the code, configurations, and data flow associated with the issue. Identify the root cause of the problem, which can include coding errors, configuration issues, data inconsistencies, or external dependencies.
- **Debugging Techniques:** Employ various debugging techniques to isolate and diagnose the issue. This can involve using integrated development environments (IDEs) with debugging capabilities, adding log statements to trace the flow of execution, or utilizing debugging tools and utilities.
- **Observing Variables and State:** Monitor and observe the values of variables, data structures, and system state during the execution of the HMS. This helps identify any discrepancies or unexpected behavior that may be causing the issue.
- **Step-by-Step Execution:** Execute the problematic section of the code step-by-step, examining the behavior at each step. This allows for pinpointing the exact location or operation that triggers the issue.
- **Code Review:** Conduct a thorough review of the relevant code, looking for logical errors, incorrect assumptions, or improper usage of functions, classes, or libraries. Ensure compliance with coding standards, best practices, and HMS design principles.
- **Data Validation:** Verify the integrity and consistency of the data used by the HMS. Check for data validation errors, incorrect data handling, or improper data transformations that may contribute to the issue.
- **Fixing the Issue:** Once the root cause is identified, implement the necessary fixes in the code, configuration, or data. This may involve making changes to the code logic, adjusting configurations, patching the HMS, or resolving data inconsistencies.
- **Retesting and Validation:** After applying the fix, retest the HMS to ensure that the issue is resolved and that it does not introduce new problems. Validate the system's functionality, performance, and stability to ensure that the debugging efforts were successful.
- **Documentation:** Document the debugging process, including the identified issue, root cause, steps taken to reproduce and resolve the problem, and any related findings or observations. This documentation serves as a reference for future debugging efforts and helps maintain the HMS.

Debugging is an iterative process that requires patience, attention to detail, and analytical thinking. It is crucial for ensuring the smooth operation of the HMS and delivering a reliable and error-free system to healthcare professionals and patients.

#### **4.4) Implementation Manual:**

- **Introduction:**
  - Overview: Provide a high-level introduction to the HMS, its purpose, and its benefits for the healthcare organization.
  - Document Purpose: Clearly state the purpose of the implementation manual and its intended audience.
- **System Requirements:**
  - Hardware Requirements: Specify the hardware infrastructure needed to support the HMS, including server specifications, network requirements, and any specialized equipment.
  - Software Requirements: List the required operating systems, databases, web servers, and other software components necessary to run the HMS.
  - Third-Party Integrations: Identify any external systems or services that need to be integrated with the HMS and provide instructions on how to set up and configure those integrations.
- **Installation and Configuration:**
  - Installation Instructions: Provide step-by-step instructions for installing the HMS software on the designated servers or workstations. Include prerequisites, installation options, and any specific configurations required.
  - Database Setup: Explain how to set up the database or data storage for the HMS, including creating the necessary tables, configuring access rights, and ensuring data security.
  - System Configuration: Detail the configuration settings for the HMS, such as system parameters, user roles and permissions, default settings, and any customization options.
  - User Management: Explain how to create user accounts, assign roles, and manage user authentication and access control within the HMS.
- **Data Migration and Integration:**
  - Data Migration Guidelines: Provide instructions on migrating existing data from legacy systems or other sources to the HMS. Include data cleansing, formatting, and validation procedures.
  - Integration Setup: Explain the process of integrating the HMS with external systems, such as laboratory systems, billing systems, or electronic health record (EHR) systems. Detail the required configurations and communication protocols.
- **System Customization:**
  - Customization Options: Explain how to customize the HMS to meet specific organizational requirements, such as adding custom fields, modifying workflows, or creating new reports.
  - Templates and Forms: Provide guidance on creating or modifying templates and forms within the HMS, including patient registration forms, medical record templates, and reporting templates.
- **Training and User Support:**
  - Training Guidelines: Outline the training approach and materials required to train end-users on using the HMS effectively. Include training schedules, training sessions, and any training resources or documentation.
  - User Support: Provide information on the support channels available to users, such as help desk contacts, online resources, FAQs, and user forums. Include instructions on how to report issues or seek assistance.



- **System Maintenance and Upgrades:**
  - Maintenance Procedures: Explain the routine maintenance tasks required to keep the HMS running smoothly, such as database backups, system backups, and log management.
  - Upgrade Guidelines: Provide instructions on upgrading the HMS to newer versions or applying software patches. Include any specific considerations or requirements for each upgrade.
- **Troubleshooting and FAQs:**
  - Troubleshooting Guide: Include a comprehensive troubleshooting section with common issues, error messages, and their resolutions. Provide step-by-step instructions for diagnosing and resolving problems.
  - Frequently Asked Questions (FAQs): Compile a list of frequently asked questions and their answers related to the implementation, configuration, and usage of the HMS.
- **Glossary and References:**
  - Glossary: Include a glossary of terms and acronyms specific to the HMS to aid understanding.
  - References: List any external references, documentation, or resources used in creating the implementation manual.

## **Chapter-5**

### **Conclusion And Reference**

#### **5.1) Conclusion:**

In conclusion, a hospital management system is a crucial tool that plays a significant role in streamlining and improving the overall efficiency of healthcare facilities. It encompasses various modules and functionalities designed to automate and integrate different aspects of hospital operations, including patient registration, appointment scheduling, medical records management, billing and invoicing, inventory management, and more.

By implementing a hospital management system, healthcare organizations can benefit from several advantages. Firstly, it enhances the accuracy and accessibility of patient data, leading to improved diagnosis and treatment outcomes. The system allows healthcare providers to access patient records in real-time, ensuring that they have up-to-date and comprehensive information to make informed decisions.

Additionally, a hospital management system facilitates efficient scheduling and management of appointments, reducing waiting times for patients and optimizing the utilization of healthcare resources. It enables seamless coordination among various departments, healthcare professionals, and administrative staff, promoting effective communication and collaboration.

The system also helps in automating administrative tasks such as billing and invoicing, eliminating manual paperwork and reducing errors. This leads to streamlined financial operations and improved revenue management for the hospital.

Furthermore, a hospital management system provides robust security measures to protect sensitive patient information, ensuring compliance with privacy regulations and safeguarding against unauthorized access.

Overall, the adoption of a hospital management system brings significant benefits to healthcare organizations, including improved patient care, enhanced operational efficiency, increased cost-effectiveness, and better decision-making. It empowers healthcare professionals to focus on their core responsibilities while reducing administrative burden, ultimately contributing to the overall growth and success of the hospital.

## **5.2) System Specifications:**

### **➤ Hardware Requirements:**

- Servers: You will require powerful servers to handle the system's database and processing needs.
- Network Infrastructure: A robust and secure network infrastructure to connect servers, client devices, and other components.
- Firewalls and Security Appliances: To protect the system from unauthorized access and cyber threats.

### **➤ Software Requirements:**

- Operating System: Choose a secure and reliable server operating system such as Linux or Windows Server.
- Web Server: Select a web server software like Apache or Nginx to handle HTTP requests and serve web pages.
- Database Management System: Use a database management system (DBMS) like MySQL, Oracle, or PostgreSQL to store and manage customer data, transactions, and other related information.
- Programming Languages: Depending on your preferences and the technologies you choose, you may use languages such as Java, C#, PHP, or Python to develop the system.
- Security Software: Implement encryption protocols (such as SSL/TLS) to secure data transmission, and employ security measures like firewalls and intrusion detection systems to protect against unauthorized access.
- Development Tools: IDEs (Integrated Development Environments) like Eclipse, Visual Studio, or IntelliJ IDEA to write, debug, and test code.

## **5.3) Limitations of The System:**

While a hospital management system offers many benefits, there are some drawbacks that organizations should be aware of. Here are a few potential drawbacks or challenges associated with hospital management systems:

- **Implementation and Training:** Implementing a hospital management system can be a complex and time-consuming process. It requires careful planning, data migration, and training for staff members. The initial setup can be challenging, and there may be a learning curve for employees who need to adapt to the new system.
- **Cost:** Hospital management systems can be expensive to implement and maintain. They often require significant upfront costs for software licenses, hardware infrastructure, and ongoing technical support. Additionally, there may be recurring costs for system updates, maintenance, and training.
- **Customization and Flexibility:** Hospital management systems typically come with a standard set of features and modules. While they can be customized to some extent, it may not always meet the specific needs and workflows of every healthcare organization. Customizations may require additional development efforts or may not be possible at all, limiting flexibility.
- **Technical Challenges:** Like any software system, hospital management systems may face technical issues such as software bugs, system crashes, or data integration problems. These issues can disrupt daily operations and require prompt technical support to resolve.

- **Staff Resistance and Training:** Introducing a new hospital management system can sometimes face resistance from staff members who may be accustomed to existing workflows and processes. It may take time and effort to convince and train employees to embrace the new system, which can impact productivity during the transition period.
- **Security and Privacy Concerns:** Hospital management systems store and handle sensitive patient data, making them potential targets for cybersecurity threats. It's crucial to have robust security measures in place to protect patient information and comply with relevant privacy regulations.
- **Dependency on Technology:** While technology offers numerous advantages, it also introduces a level of dependency. If the hospital management system encounters a significant failure or downtime, it can disrupt operations and impact patient care until the system is restored.

It's essential to evaluate these drawbacks and potential challenges alongside the benefits when considering the implementation of a hospital management system. Engaging stakeholders, conducting thorough research, and consulting with experienced professionals can help mitigate these drawbacks and ensure a successful implementation and utilization of the system.

#### **5.4) Future Scope for Modification:**

The future scope for modification in a hospital management system is vast, and advancements in technology continue to provide opportunities for improvement. Here are some potential areas of development:

- **Integration of Artificial Intelligence (AI) and Machine Learning (ML):** Implementing AI and ML algorithms can enhance various aspects of the hospital management system. For example, AI-powered chatbots can provide instant responses to patients' inquiries and help with appointment scheduling. ML algorithms can analyze patient data to identify patterns, predict disease progression, and optimize treatment plans.
- **Internet of Things (IoT) Integration:** IoT devices can be utilized to collect real-time data from patients and medical equipment. This data can be integrated into the hospital management system to monitor patient vitals, track inventory levels, and ensure equipment maintenance. IoT-enabled wearables can also assist in remote patient monitoring, allowing healthcare providers to deliver personalized care outside the hospital setting.
- **Blockchain for Data Security:** Blockchain technology offers secure and tamper-proof storage of sensitive patient data. Implementing blockchain in the hospital management system can enhance data privacy, streamline data sharing between healthcare providers, and facilitate interoperability between different systems.
- **Telemedicine and Remote Patient Management:** The COVID-19 pandemic has accelerated the adoption of telemedicine, and its integration into hospital management systems is likely to continue. Enhancements can be made to facilitate virtual consultations, remote patient monitoring, electronic prescriptions, and seamless integration of telemedicine platforms with existing hospital workflows.
- **Data Analytics and Business Intelligence:** Hospital management systems generate vast amounts of data. Implementing robust data analytics and business intelligence tools can help derive insights, identify trends, and make informed decisions. Predictive analytics can

optimize resource allocation, improve operational efficiency, and enable proactive patient care management.

- **Mobile Applications and Patient Portals:** Developing intuitive mobile applications and patient portals can empower patients to manage their healthcare better. Features like appointment scheduling, access to medical records, medication reminders, and secure communication with healthcare providers can be incorporated into mobile apps and portals, enhancing patient engagement and satisfaction.
- **Interoperability and Standardization:** Seamless exchange of information between different healthcare systems and organizations is crucial. Future modifications should focus on improving interoperability standards, such as HL7 and FHIR, to ensure efficient data sharing, eliminate duplicative data entry, and enhance care coordination.
- **Enhanced Security and Privacy Measures:** With the increasing digitalization of healthcare, ensuring robust security and privacy measures is critical. Future modifications should prioritize the implementation of advanced cybersecurity protocols, data encryption, and access controls to protect patient information from unauthorized access and cyber threats.
- These are just a few potential areas for future modifications in a hospital management system. As technology continues to advance, there will likely be more opportunities to improve efficiency, patient care, and overall healthcare outcomes through innovative modifications.

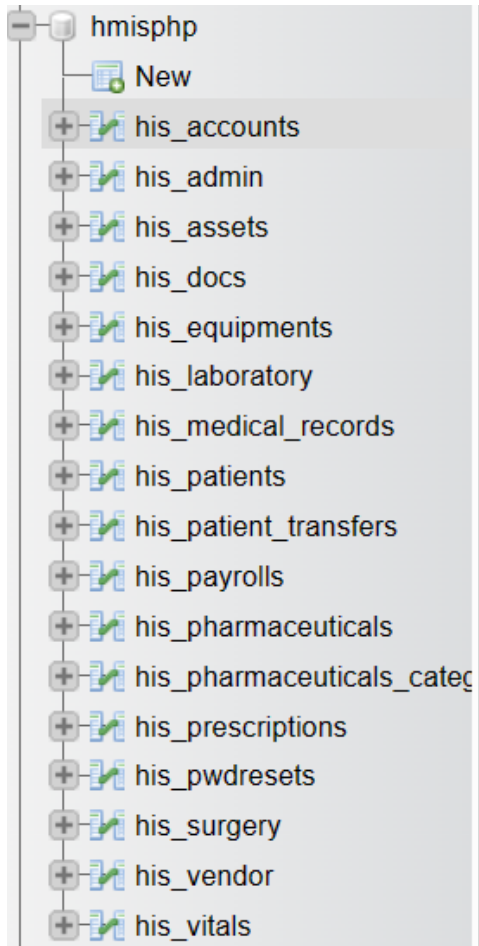
### 5.5) **References/Bibliography:**

- [hospital management : Features, Advantages, Registration, Login, Fund Transfer \(paisabazaar.com\)](#)
- [ChatGPT \(openai.com\)](#)
- Prof. Assankutty, Jojo Mon. NA. (2007). “Financial service”, Calicut University, Calicut:Kerala
- Reserve Bank of India (November, 2013), ATMs and Plastic money report
- Kurnia, S., Peng, F., & Liu, Y. R. (2010). Understanding the adoption of in China. In *43rd Hawaii International Conference on System Sciences*, Honolulu, Hawaii, USA, pp. 1–10.
- Vrîncianu, M., & Popa, L. A. (2010). Considerations regarding the security and protection of e services consumers’ interests. *The Amfiteatru Economic Journal*, 12(28), 388–403.
- Peotta, L., Holtz, M. D., David, B. M., Deus, F. G., & Timoteo de Sousa, R. (2011). A formal classification of internet and vulnerabilities. *International Journal of Computer Science and Information Technology*, 3(1), 186–197.
- Drig, I., & Isac, C. (2014– Features, challenges and benefits. 10.
- Singhal, D., & Padhmanabhan, V. (2009). A study on customer perception towards major contributing factors. *Journal of Nepalese Business Studies*, 5(1), 101–111.
- Bahl, D. S. (2012). E: Challenges and policy implications. *International Journal of Computing & Business Research*, 229–6166.
- Omariba, Z. B., & Masese, N. B. (2012). Security and privacy of electronic *International Journal of Computer Science Issues (IJCSI)*, 9(4), 432–446.

## Chapter-6

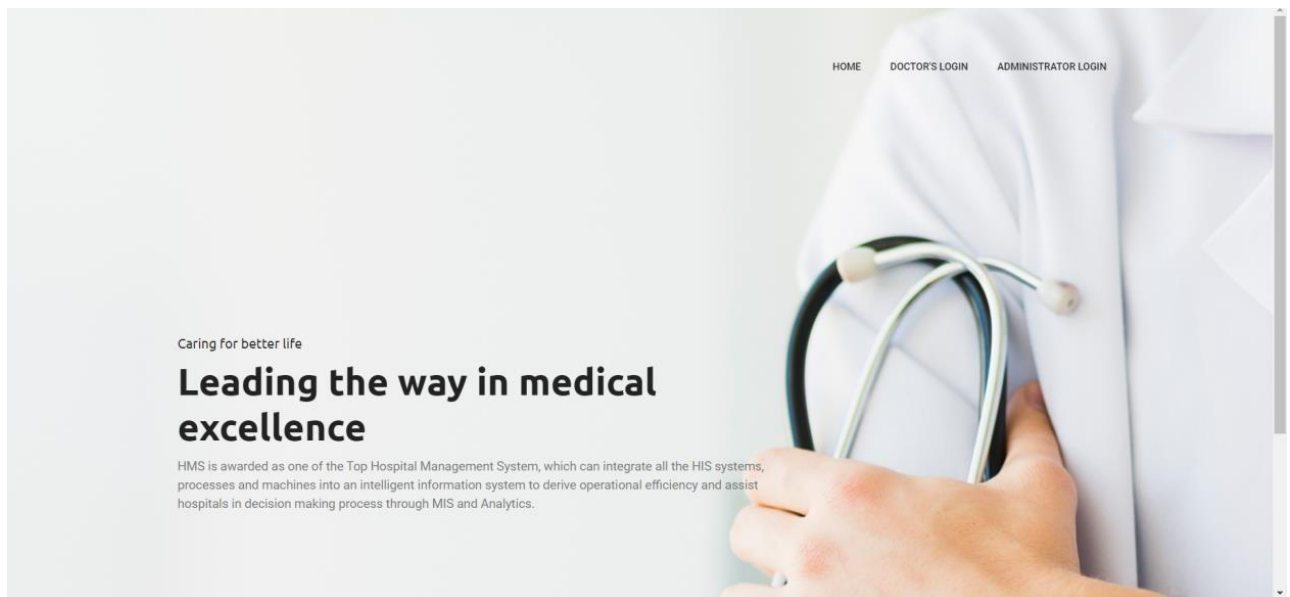
### ANNEXURS

#### 6.1) Structured Chart:



**Fig 3.3 (Simple Class Diagram)**

## 6.7) Sample Inputs/Outputs:



**Fig 3.6 (Log in page)**

The image shows the 'Add Vendor Details' form within the HMS interface. The top header bar includes the HMS logo, a 'Create New' button, a search bar, and a user profile for 'System Administrator'. A left sidebar lists various navigation options: Dashboard, Patients, Employees, Pharmacy, Accounting, Inventory, Reporting, Medical Records, Laboratory, Surgical / Theatre, Payrolls, and Vendors (which is currently selected). The main content area is titled 'Add Vendor Details' and includes a breadcrumb trail: 'Dashboard > Vendor > Add Vendor'. The form itself has a section 'Fill all fields' with three input fields for 'Vendor Name', 'Vendor Phone Number', and 'Vendor Address', followed by a 'Vendor Email' field. Below these is a 'Vendor Details' section with a rich text editor containing formatting icons (bold, italic, link, etc.) and a large text area. At the bottom of the form is a green 'Add Vendor' button.

**Fig 3.7 (vendor details)**



The screenshot shows the 'Add Patient Details' form in the HMS system. The top navigation bar includes the HMS logo, a 'Create New' dropdown, a search bar, and a user profile for 'System Administrator'. The left sidebar lists navigation options: Dashboard, Patients (selected), Employees, Pharmacy, Accounting, Inventory, and Reporting. The 'Patients' menu is expanded, showing 'Register Patient', 'View Patients', 'Manage Patients', 'Discharge Patients', and 'Patient Transfers'. The main form area is titled 'Add Patient Details' and contains the following fields: 'First Name' (with placeholder 'Patient's First Name'), 'Last Name' (with placeholder 'Patient's Last Name'), 'Date Of Birth' (with placeholder 'DD/MM/YYYY'), 'Age' (with placeholder 'Patient's Age'), 'Address' (with placeholder 'Patient's Address'), 'Mobile Number', 'Patient Ailment', and 'Patient's Type' (a dropdown menu with 'Choose' selected). A blue 'Add Patient' button is located at the bottom of the form.

**Fig 3.8 (Add Patient Details)**

The screenshot shows the 'Create A Pharmaceutical' form in the HMS system. The top navigation bar is identical to the previous screenshot. The left sidebar lists navigation options: Dashboard, Patients, Employees, Pharmacy (selected), Accounting, Inventory, and Reporting. The 'Pharmacy' menu is expanded, showing 'Add Pharm Category', 'View Pharm Category', 'Manage Pharm Category', 'Add Pharmaceuticals' (selected), 'View Pharmaceuticals', 'Manage Pharmaceuticals', 'Add Prescriptions', 'View Prescriptions', and 'Manage Prescriptions'. The main form area is titled 'Create A Pharmaceutical' and contains the following fields: 'Pharmaceutical Name', 'Pharmaceutical Quantity(Cartons)', 'Pharmaceutical Category' (a dropdown menu with 'Analgesics' selected), 'Pharmaceutical Vendor' (a dropdown menu with 'Cosmos Pharmaceutical Limited' selected), 'Pharmaceutical Barcode(EAN-B)' (with placeholder '246197580'), and 'Pharmaceutical Description' (a rich text editor with a toolbar containing bold, italic, link, unlink, bulleted list, numbered list, indent, outdent, and help icons). A blue 'Add Pharmaceutical' button is located at the bottom of the form.

**Fig 3.9 (Creating Parametrical)**