Hospital Management System with Scheduling and Billing

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology/Master of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

D.JATHIN (AP23110010726)

SIRI.K (AP23110010777)

Under the Guidance of

Dr. C H MARY

SRM University-AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[NOV, 2024]

Certificate

Date: 16-Nov-24

This is to certify that the work present in this Project entitled “Hospital Management System with Scheduling and Billing” has been carried out by [Rithwik, Vignesh, Taneesh] under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/ Master of Technology in School of Engineering and Sciences.

Supervisor

Prof. / Dr. [C H MARY
]
Designation,
Affiliation.

Co-supervisor

(Signature)
Prof. / Dr. [Name]
Designation,
Affiliation.

Acknowledgements

I would like to thank all those who contributed to the successful completion of this Hospital Management System project. I am thankful to Ms.Poonam Yadav, for the invaluable guidance and support during the development process. Their expertise and constructive feedback helped shape the project.

I am grateful to my colleagues and peers who cooperated with me and gave me the confidence to proceed with the task. I thank my family for the support and tolerance during all the stages of this project.

I would not have completed this project if it were not for the availability of resources and literature in the field of software development, which inspired me to pursue this project with a good approach.

Table of Contents

Abstract

This code implements a basic Hospital Management System. Users can login with a password and then choose from various functionalities:

- Adding a new patient record with details like name, address, contact information, blood group, age, any previous diagnosis, etc.
- Appending diagnosis information to a patient's existing file, including symptoms, diagnosis, medication, and ward allocation (if necessary).
- Displaying the full medical history of a patient from their file.
- Managing appointments: adding new appointments with patient ID, doctor's name, date, and time, and viewing existing appointments.
- Billing system: Calculating and generating bills based on services rendered, medications prescribed, and room charges.
- Exiting the program.

The system utilizes text files to store patient information and appointments.

Abbreviations

- HMS: Hospital Management System – refers to the entire program.

- ID: Identifier – a unique string or number used to identify each patient.
- pat_file: Patient File – the file object for storing patient data.
- billingFile: Billing File – the file object for storing billing information.
- appointmentFile: Appointment File – the file object for storing appointment data.
- fname: File Name – a variable used to hold the name of a patient file.
- AppointmentInfo: Abbreviated as ApptInfo – stores details about an appointment.
- BillingInfo: Abbreviated as BillInfo – stores billing-related details.
- patient_info: Abbreviated as PatInfo – stores general patient information.
- app: Abbreviated as AppInfo – stores information about a patient’s diagnosis within their file.

List of Tables

Table -1(class and description)

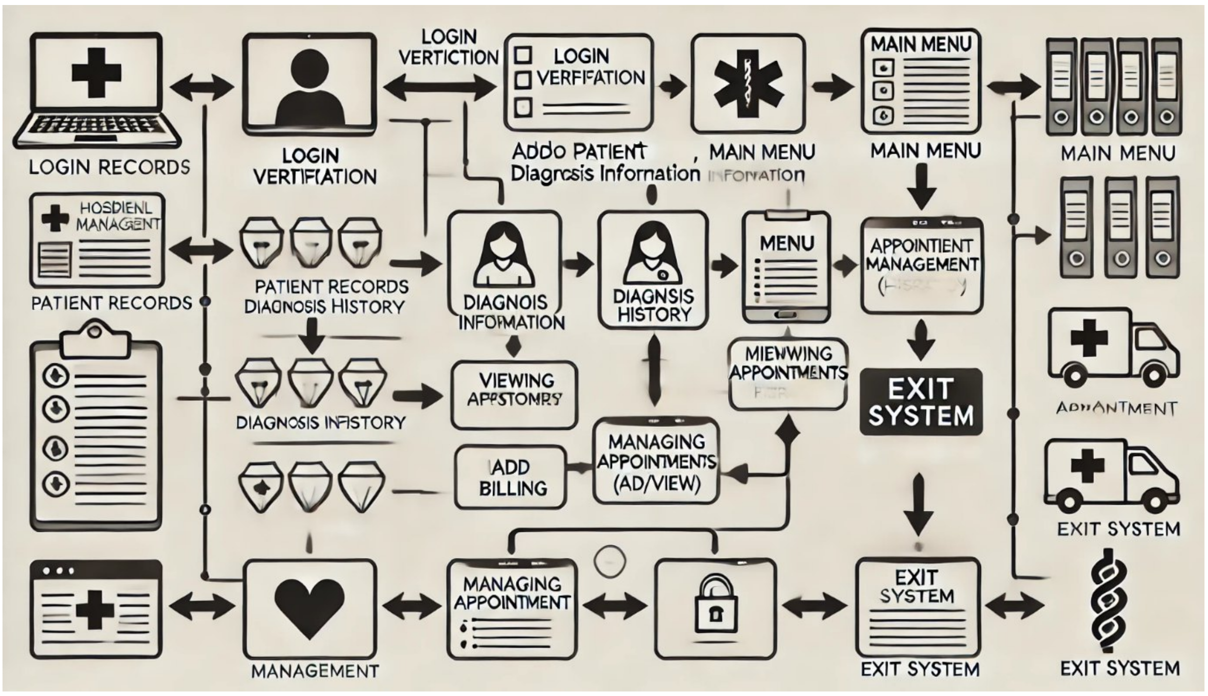
Class	description
Billinginfo	This class stores the information a patient billing
Members	**Description**
Patient id	This stores the id of the patient
amount	This stores the amount billed to the patient
Description	Stores a description of the services rendered

Class	Description
Appointmentinfo	This class stores information about a patient's appointment
Members	**Description**
Doctor name	This stores name of the doctor name
Appointment date	This holds the appointment date in DD/ MM/YYYY format
Appointment time	This hold the appointment time in HH:MM format

Table -2(method and description)

Method	Description
Main()	This is the programs entry point. This displays welcome message and then prompts the user for login
Login()	This function handles the login process. It prompts the user to enter a password and hides the characters as they are typer using _getch
AddPatientRecord()	This function helps to adds new patient information to file.
viewPatientHistory()	This function retrieves and displays a patient's medical history information stored in a file
addAppointment()	This function allows adding new appointment details to a file.
viewAppointment()	This function retrieves and displays existing appointment information stored in the "appointments.txt" file
addBilling()	This function allows adding new billing information to a file
viewBilling()	This function retrieves and displays existing billing information stored in the "billing.txt" file

List of Figures


LOGIN RECORDS
LOGIN VERTIFIATION
LOGIN VERTICTION
LOGIN VERFIFATION
ADDO PATIENT Diagrcsis Information
MAIN MENU INFONATION
MAIN MENU
MAIN MENU
MAIN MENU
HOSDIENL MANAGENT
PATIENT RECORDS
PATIENT RECORDS DIAGNOSIS HISTORY
DIAGNOSIS INFOMATION
DIAGNSIS HISTORY
MENU
APPOINTIENT MANAGEMENT
DIAGNOSIS INFISTORY
VIEWING APPSTONRS
MIEWWING APPOINTMENTS
EXIT SYSTEM
ADVANTMENT
ADD BILLING
MANAGING APPOINTMENTS (AD/VIEW)
EXIT SYSTEM
MANAGEMENT
MANAGING APPOINTMENT
EXIT SYSTEM
EXIT SYSTEM
EXIT SYSTEM

Flowchart for Hospital Management System (C++)

1. Login Process
   - Start   Enter Password
   - If Password Correct   Proceed to Main Menu
   - If Password Incorrect   Retry Login
2. Main Menu Options (Choose from options 1–6)
   - Option 1: Add Patient Record
     - Input: Patient Info (Name, Address, Contact, Age, etc.)
     - Action: Save to File
   - Option 2: Add Diagnosis Information
     - Input: Patient File, Symptoms, Diagnosis, Medication
     - Action: Append to Patient File
   - Option 3: View Full History
     - Input: Patient File Name
     - Action: Read and Display File Content
   - Option 4: Appointments
     - Sub-options:
       - Add Appointment: Input Patient ID, Doctor, Date, Time   Save to File
       - View Appointments: Open file and display appointment list
   - Option 5: Billing Information
     - Sub-options:
       - Add Billing Info: Input Patient ID, Amount, Description   Save to File
   - Option 6: Exit Program

Introduction

The provided code implements a basic Hospital Management

System (HMS) that offers the following functionalities:

1. Patient Management:
   - Adding New Patients: Users can input patient details like name, address, contact information, and medical history. This information is stored in separate files for each patient.
   - Adding Diagnosis Information: Users can append diagnosis details, such as symptoms, diagnosis, medication, and ward allocation, to existing patient files.
   - Viewing Full Medical History: Users can access and view the complete medical history of a patient from their respective files.
2. Appointment Scheduling:
   - Adding Appointments: Users can schedule appointments by specifying patient ID, doctor's name, date, and time.
   - Viewing Appointments: Users can view existing appointments.
3. Billing:
   - Billing system: users can get the bill of the patient foe their treatments and services including consultation fee.

Technical Implementation:

- File-Based Storage: The system utilizes text files to store patient records and appointments. Each patient has a dedicated file to store their medical history.
- User Interface: The user interface is primarily text-based, providing a console-based menu for user interaction.
- Security: The system employs a simple password-based login mechanism to restrict access. However, it lacks advanced security features like password hashing or encryption.

Objective

The primary objective of the Hospital Management System (HMS) is to streamline and automate various hospital operations, leading to improved efficiency, accuracy, and patient care.

- Patient record management
- Appointment scheduling
- Diagnose and treatment management
- Billing management

2. Methodology

Methodology for Hospital Management System (HMS) The design of the Hospital Management System (HMS) has a well-structured methodology so that the system will be efficient, user-friendly, and satisfies the requirements of the hospital. Here is the simple methodology used in this project.

1. System Design-

- Data structures:-

Struct patient_info:- Stores patient details like name, address, contact, age, sex, blood group, past diseases, and ID.

Struct appointmentinfo:- Stores appointment details like patient ID, doctor's name, appointment date, and time.

- Security:-

Simple password protection using a hardcoded password ("pass") for login.

2. Implementation-

- Code structure:-

Includes necessary header files for input/output operations, time handling, string manipulation, and console control.

Defines functions for login, adding patient records, managing appointments, and viewing patient history.

The main function serves as the program entry point, handling user interaction and calling other functions.

- Login Function (login):
- Prompts user for a password.
- Hides password input using _getch function.
- Grants access if the password matches the hardcoded string ("pass").
- Managing Appointments (addAppointment, viewAppointments):
- addAppointment:
    - Prompts user for appointment details (patient ID, doctor name, date, time).
    - Appends the appointment information to the "appointments.txt" file.
- Viewing Patient History (viewHistory):
- Prompts user for the patient ID.
- Opens the corresponding patient record file and displays its content.

3. Discussion

The Hospital Management System implemented in this code effectively showcases how object-oriented programming principles can be used to structure and model real-world systems like a hospital. By breaking down the system into distinct classes (Person, Patient, Doctor, Billing, and Appointment), the code achieves high modularity, making it easier to manage and expand.

1. User Experience

- The current system relies heavily on text-based interactions, which can be less intuitive and engaging for users. A graphical user interface (GUI) would significantly improve the user experience. Additionally, features like user authentication, data validation, and error handling should be implemented to ensure data integrity and security. By addressing these areas, the HMS can provide a more efficient, user-friendly, and secure experience for healthcare professionals.

2. Data Structure

- Data Structure: The system employs simple classes for modeling patients, doctors, and billing details, but it could be expanded by adding additional features like multiple doctors per patient, handling patient history over multiple appointments, or integrating a database to store patient information persistently.

3. Error Handing

- The custom HMS Exception class ensures that any errors in the system are caught and appropriately handled. However, more specific exceptions could be added to handle different types of errors, such as invalid input types or date conflicts

4.

Concluding Remarks

The Hospital Management System (HMS) project is designed to manage a hospital's basic operations, such as managing patient records, storing and retrieving billing information, handling appointments, and providing a secure login for authorized access. The project is structured with modular functions, each handling a specific task to maintain code organization and readability.

Modular Functions: Each function (e.g., addAppointment, viewAppointments, addBilling, viewBilling) is designed to handle a specific operation, which enhances readability and maintainability of the code.

Input Validation: Ensures that user inputs are within the expected range or format (e.g., menu choices, patient IDs).

User-Friendly Interface: Menus, prompts, and feedback messages guide users effectively through the system.

Use of goto Statements: goto statements are used to handle repetitive tasks and invalid input cases but could be replaced with loops to improve readability and avoid potential errors.

5. Future Work

1. Database Integration
    - Transition from File Storage to Database Management: Replace text files with a database (e.g., MySQL, SQLite) to handle larger data sets more efficiently and enable complex queries, like searching for records by date, doctor, or treatment type.
    - Improved Data Integrity and Security: A database offers built-in data validation, indexing for fast retrieval, and access control, enhancing the overall security and reliability of the system.
2. User Management and Access Control
    - Role-Based Access: Implement a user management system that differentiates between roles, such as doctors, nurses, receptionists, and administrators, with different access levels to sensitive information.
    - Enhanced Login Security: Use encryption for password storage and implement multi-factor authentication (MFA) to strengthen access security.

3. Mobile Application
    - Patient and Staff Mobile App: Develop mobile apps for patients and hospital staff, enabling them to manage appointments, view records, and perform essential functions on the go.
    -

6. References

<u>Websites</u>

https://www.geeksforgeeks.org/

https://www.w3schools.com/cpp/

Books

C++ Primer" by Stanley B. Lippman, Josée Lajoie, and Barbara E. Moo.