**1.Login**

@startuml

skinparam activity {

BackgroundColor<<success>> LightGreen

BorderColor<<success>> Green

BackgroundColor<<error>> LightSalmon

BorderColor<<error>> DarkRed

}

start

:User opens the application and selects "Login";

:Display the login screen;

:User enters login information;

if (Is information correct?) then (yes)

:System verifies the information;

:Navigate to the main page of the system;

stop

else (no)

:Display error message;

:Request to re-enter login information;

stop

@enduml

**2. Sign up**

@startuml

skinparam activity {

BackgroundColor<<success>> LightGreen

BorderColor<<success>> Green

BackgroundColor<<error>> LightSalmon

BorderColor<<error>> DarkRed

}

start

:Prospective User opens the application;

:Prospective User selects "Sign Up";

:Display the sign-up page;

if (User provides valid information?) then (yes)

:Present registration form;

:Prospective User fills out the form with valid information;

if (Information is valid?) then (yes)

:System validates the information;

:System creates a new user account;

:System logs in the new user;

:Redirect user to the main page;

stop

else (no)

:Display error message;

:Request user to correct information;

stop

endif

else (no)

:Display error message;

:Request user to provide valid information;

stop

endif

@enduml

**3. mã hóa dữ liệu**

@startuml

!define ENCODE\_DATA <<(A,blue) Encode Data>>

start

:Prepare data to be encoded;

:Select encryption method;

:Hệ thống mã hóa dữ liệu;

:Encoded data is generated;

stop

@enduml

**4. forgot password**

@startuml

skinparam activity {

BackgroundColor<<success>> LightGreen

BorderColor<<success>> Green

BackgroundColor<<error>> LightSalmon

BorderColor<<error>> DarkRed

}

start

:User selects "Forgot Password";

:Display the Forgot Password page;

if (User provides valid email address?) then (yes)

:System prompts user to enter their registered email address;

:User provides their email address and submits the request;

if (Email address is valid?) then (yes)

:System sends a recovery email to the provided address;

:User receives instructions on how to reset their password;

stop

else (no)

:Display error message;

:Request user to provide a valid email address;

stop

endif

else (no)

:Display error message;

:Request user to provide a valid email address;

stop

endif

@enduml

**5. Send Messages**

@startuml

skinparam activity {

BackgroundColor<<message>> LightSkyBlue

BorderColor<<message>> RoyalBlue

}

start

:Employee logs into Messaging System;

:Employee initiates a new conversation or selects an existing one;

:Employee enters message content;

:Employee sends message to User;

:User receives message and reads it;

:User responds to the message;

:Messaging System records conversation history;

stop

@enduml

**6. Calculate Salary**

@startuml

skinparam activity {

BackgroundColor<<calculate>> LightYellow

BorderColor<<calculate>> DarkSlateGray

}

start

:Retrieve Employee Information;

:Calculate Gross Salary;

if (Overtime Calculation Needed?) then (yes)

:Calculate Overtime Pay;

:Add Overtime Pay to Gross Salary;

endif

if (Bonus Calculation Needed?) then (yes)

:Calculate Bonus;

:Add Bonus to Gross Salary;

endif

:Apply Deductions;

:Calculate Net Salary;

:Update Payroll Records;

stop

@enduml

**7. Terminate Employee**

@startuml

skinparam activity {

BackgroundColor<<terminate>> LightCoral

BorderColor<<terminate>> Crimson

}

start

:HR Manager logs in;

:Navigate to Employee Management;

:Select Employee to be Terminated;

:Provide Termination Reason;

:Update Employee Status to "Terminated";

:Revoke Access Privileges;

stop

@enduml

**8. CRUD Patient Info**

@startuml

skinparam activity {

BackgroundColor<<crud>> LightGoldenRodYellow

BorderColor<<crud>> GoldenRod

}

start

:Healthcare Staff logs in;

if (Authentication and Permissions are valid?) then (yes)

:Healthcare Staff selects CRUD operation;

if (Operation is Create) then (yes)

:Healthcare Staff enters new patient details;

:System stores new patient record;

else (no)

if (Operation is Read) then (yes)

:Healthcare Staff initiates search or request for patient information;

:System retrieves and displays patient information;

else (no)

if (Operation is Update) then (yes)

:Healthcare Staff searches for existing patient record;

:Healthcare Staff updates patient information;

:System stores updated patient information;

else (no)

if (Operation is Delete) then (yes)

:Healthcare Staff searches for existing patient record;

:System permanently removes patient record;

else (no)

:Error: Invalid Operation;

endif

endif

endif

endif

else (no)

:Error: Unauthorized Access;

endif

stop

@enduml

**9. Pay Medical Fees**

@startuml

skinparam activity {

BackgroundColor<<payment>> LightSalmon

BorderColor<<payment>> DarkRed

}

start

:Patient initiates payment;

:Provide payment information;

if (Payment Information is valid?) then (yes)

:Verify payment information;

:Process payment;

:Update billing system;

:Generate payment receipt;

:Provide receipt to Patient;

else (no)

:Error: Invalid Payment Information;

stop

endif

stop

@enduml

**10. Generate Invoice**

@startuml

skinparam activity {

BackgroundColor<<invoice>> LightCyan

BorderColor<<invoice>> DarkSlateGray

}

start

:Initiate Invoice Generation;

:Retrieve Patient's Medical Charges;

if (Charges Verification Required?) then (yes)

:Healthcare Staff Verifies Charges;

if (Verification Successful?) then (yes)

:Adjust Invoice if needed;

else (no)

:Error: Verification Failed;

stop

endif

endif

:Generate Invoice with Detailed Charges;

:Store Invoice in Billing System;

:Notify Patient of Generated Invoice;

stop

@enduml

**11. CRUD Appointment**

@startuml

skinparam activity {

BackgroundColor<<crud>> LightYellow

BorderColor<<crud>> DarkOrange

}

start

:Healthcare Staff logs in;

if (Authentication and Authorization are valid?) then (yes)

:Healthcare Staff selects CRUD operation;

if (Operation is Create) then (yes)

:Healthcare Staff enters new appointment details;

:System records new appointment;

else (no)

if (Operation is Read) then (yes)

:Healthcare Staff views existing appointments;

:Appointments are displayed;

else (no)

if (Operation is Update) then (yes)

:Healthcare Staff selects appointment for modification;

:Healthcare Staff updates appointment details;

:System records updated appointment;

else (no)

if (Operation is Delete) then (yes)

:Healthcare Staff selects appointment for cancellation;

:System removes appointment from schedule;

else (no)

:Error: Invalid Operation;

endif

endif

endif

endif

else (no)

:Error: Unauthorized Access;

endif

stop

@enduml

**12. View Medical History**

@startuml

skinparam activity {

BackgroundColor<<view>> LightBlue

BorderColor<<view>> DarkBlue

}

start

:Actor (Patient or Healthcare Staff) initiates request to view medical visit history;

if (Authentication and Authorization are valid?) then (yes)

:System verifies credentials;

if (Request by Patient?) then (yes)

:Patient logs into patient portal or contacts healthcare staff;

:System retrieves and displays medical visit history;

:Patient reviews medical visit history;

else (no)

:Healthcare Staff accesses medical visit history through EHR system;

:System retrieves and displays medical visit history;

:Healthcare Staff reviews medical visit history;

endif

else (no)

:Error: Unauthorized Access;

endif

stop

@enduml

**13. CRUD Employee Info**

@startuml

skinparam activity {

BackgroundColor<<crud>> LightPink

BorderColor<<crud>> Crimson

}

start

:Actor (Admin Staff, HR Personnel, or Sys Admin) initiates CRUD operation;

if (Authentication and Authorization are valid?) then (yes)

:System verifies credentials;

if (Operation is Create) then (yes)

:User enters new staff information;

:System records new staff information;

else (no)

if (Operation is Read) then (yes)

:User accesses staff information system;

:Existing staff information is displayed;

else (no)

if (Operation is Update) then (yes)

:User selects staff information for modification;

:User updates staff information;

:System records updated staff information;

else (no)

if (Operation is Delete) then (yes)

:User selects staff information for deletion;

:System removes staff information;

else (no)

:Error: Invalid Operation;

endif

endif

endif

endif

else (no)

:Error: Unauthorized Access;

endif

stop

@enduml

**14. Manage Work Schedule**

@startuml

skinparam activity {

BackgroundColor<<crud>> LightCyan

BorderColor<<crud>> DarkSlateGray

}

start

:Actor (HR Personnel or Admin Staff) initiates schedule management;

if (Authentication and Authorization are valid?) then (yes)

:System verifies credentials;

if (Operation is Create) then (yes)

:User enters new work schedule details;

:System records new work schedule;

else (no)

if (Operation is Read) then (yes)

:User accesses schedule management system;

:Existing work schedules are displayed;

else (no)

if (Operation is Update) then (yes)

:User selects work schedule for modification;

:User updates work schedule details;

:System records updated work schedule;

else (no)

if (Operation is Delete) then (yes)

:User selects work schedule for deletion;

:System removes work schedule;

else (no)

:Error: Invalid Operation;

endif

endif

endif

endif

else (no)

:Error: Unauthorized Access;

endif

stop

@enduml

**15. Track Revenue and Expenses**

@startuml

skinparam activity {

BackgroundColor<<monitor>> LightYellow

BorderColor<<monitor>> DarkGoldenRod

}

start

:Financial Analyst or Business Manager initiates monitoring;

if (Authentication and Authorization are valid?) then (yes)

:System verifies credentials;

:User accesses financial monitoring system;

:User specifies monitoring parameters;

:System retrieves fee-related transaction data;

:User analyzes fee revenue data;

if (Reports needed?) then (yes)

:User generates reports;

endif

else (no)

:Error: Unauthorized Access;

endif

stop

@enduml

16. Create Financial Report

@startuml

!define CREATE\_FINANCIAL\_REPORT <<(A,blue) Create Financial Report>>

start

:User selects the "Create Financial Report" option;

:Retrieve financial data from the financial database;

:Generate a financial report based on the retrieved data;

:Display the financial report on the user interface;

stop

@enduml

17. Track Inventory and Supplies

@startuml

!define TRACK\_INVENTORY\_SUPPLIES <<(A,green) Track Inventory and Supplies>>

start

:User selects the "Track Inventory and Supplies" option;

:Retrieve inventory and supply data from the database;

:Display the inventory and supply data on the user interface;

stop

@enduml

18. Create Inventory Receipt

@startuml

!define CREATE\_INVENTORY\_RECEIPT <<(A,blue) Create Inventory Receipt>>

start

:User selects the "Create Inventory Receipt" option (In or Out);

if (Receipt type is In) then (yes)

:User enters details of items received and quantities;

:System updates inventory with incoming items;

:Generate an incoming inventory receipt;

else (Receipt type is Out)

:User selects items to be taken out and enters quantities;

:System updates inventory with outgoing items;

:Generate an outgoing inventory receipt;

endif

:Display the generated receipt on the user interface;

stop

@enduml

19. Send Appointment Reminder

@startuml

!define SEND\_APPOINTMENT\_REMINDER <<(A,green) Send Appointment Reminder>>

start

:User (employee or doctor) selects an appointment to send a reminder;

:Retrieve appointment details and contact information for the patient;

:Compose a reminder message;

:Send the appointment reminder to the patient;

:Record the sent reminder in the appointment history;

stop

@enduml

20. CRUD Prescription

@startuml

!define CRUD\_PRESCRIPTION <<(A,green) CRUD Prescription>>

start

:Doctor selects operation (Create, Read, Update, Delete);

if (Operation is Create) then (yes)

:Doctor creates a new prescription for a patient;

:Prescription details are added to the patient's medical record;

:New prescription is saved;

else (Operation is not Create)

if (Operation is Read) then (yes)

:Doctor searches for a patient's prescription;

:Prescription details are retrieved and displayed;

else (Operation is not Read)

if (Operation is Update) then (yes)

:Doctor selects a patient's prescription for modification;

:Doctor makes necessary changes to the prescription details;

:Updated prescription is saved;

else (Operation is Delete)

:Doctor selects a patient's prescription for deletion;

:Prescription details are removed from the patient's medical record;

:Deleted prescription is saved;

endif

endif

endif

stop

@enduml

21. Manage Medical Records

@startuml

!define MANAGE\_MEDICAL\_RECORDS <<(A,blue) Manage Medical Records>>

start

:User (doctor or medical staff) selects a patient's medical record;

:Retrieve and display the patient's medical record details;

if (User wants to make changes?) then (yes)

:User makes necessary updates to the medical record;

:Updated medical record is saved in the system;

else (no)

:User reviews the medical record;

:No changes are made to the medical record;

endif

stop

@enduml

22. Print Medical Records

@startuml

!define PRINT\_MEDICAL\_RECORDS <<(A,green) Print Medical Records>>

start

:User (doctor or medical staff) selects a patient's medical record to print;

:Retrieve and display the patient's medical record details;

:User initiates the printing process for the medical record;

:System generates a printed copy of the medical record;

:User receives the printed medical record copy;

stop

@enduml

23. Monitor Equipment Status

@startuml

!define MONITOR\_EQUIPMENT\_STATUS <<(A,blue) Monitor Equipment Status>>

start

:Manager selects equipment for monitoring;

:Retrieve and display equipment status and maintenance information;

:Monitor equipment status in real-time;

if (Equipment status changes?) then (yes)

:Notify manager about equipment status change;

else (no)

:No changes in equipment status;

endif

stop

@enduml

24. Create Equipment Maintenance Plan

@startuml

!define CREATE\_MAINTENANCE\_PLAN <<(A,green) Create Equipment Maintenance Plan>>

start

:Manager selects equipment to create a maintenance plan;

:Retrieve equipment details and history of maintenance activities;

:Manager defines a maintenance schedule and tasks for the equipment;

:Equipment maintenance plan is created and saved in the system;

stop

@enduml

25. CRUD Advertising

@startuml

!define CRUD\_ADVERTISING <<(A,green) CRUD Advertising>>

start

:Manager selects operation (Create, Read, Update, Delete);

if (Operation is Create) then (yes)

:Manager creates a new advertising campaign;

:Advertising campaign details are added to the system;

:New advertising campaign is saved;

else (Operation is not Create)

if (Operation is Read) then (yes)

:Manager searches for an existing advertising campaign;

:Advertising campaign details are retrieved and displayed;

else (Operation is not Read)

if (Operation is Update) then (yes)

:Manager selects an advertising campaign for modification;

:Manager makes necessary changes to the campaign details;

:Updated advertising campaign is saved;

else (Operation is Delete)

:Manager selects an advertising campaign for deletion;

:Advertising campaign details are removed from the system;

:Deleted advertising campaign is saved;

endif

endif

endif

stop

@enduml

26. CRUD Voucher

@startuml

!define CRUD\_VOUCHER <<(A,green) CRUD Voucher>>

start

:Manager selects operation (Create, Read, Update, Delete);

if (Operation is Create) then (yes)

:Manager creates a new voucher;

:Voucher details are added to the system;

:New voucher is saved;

else (Operation is not Create)

if (Operation is Read) then (yes)

:Manager searches for an existing voucher;

:Voucher details are retrieved and displayed;

else (Operation is not Read)

if (Operation is Update) then (yes)

:Manager selects a voucher for modification;

:Manager makes necessary changes to the voucher details;

:Updated voucher is saved;

else (Operation is Delete)

:Manager selects a voucher for deletion;

:Voucher details are removed from the system;

:Deleted voucher is saved;

endif

endif

endif

stop

@enduml

27. Provide Service Feedback

@startuml

!define PROVIDE\_FEEDBACK <<(A,green) Provide Service Feedback>>

start

:Patient selects the service to provide feedback;

:Retrieve and display service details;

:Patient provides feedback and rating for the service;

:Feedback and rating are submitted to the system;

:Feedback is recorded in the system for service improvement;

stop

@enduml

28. Create Treatment Plan for Patient

@startuml

!define CREATE\_TREATMENT\_PLAN <<(A,green) Create Treatment Plan for Patient>>

start

:Doctor selects a patient for treatment planning;

:Retrieve patient's medical history and examination results;

:Doctor creates a treatment plan based on the patient's condition;

:Treatment plan is reviewed and approved;

:Treatment plan details are saved in the system;

stop

@enduml

29. Track Treatment Plan Progress

@startuml

!define TRACK\_TREATMENT\_PROGRESS <<(A,green) Track Treatment Plan Progress>>

start

:Doctor or Patient selects the treatment plan to track;

:Retrieve and display the treatment plan details;

if (User is Doctor) then (yes)

:Doctor reviews the progress and updates the plan;

:Updated plan details are saved in the system;

else (User is Patient)

:Patient reviews the progress and provides feedback;

:Feedback is recorded in the system for adjustments;

endif

stop

@enduml

30. CRUD Service Contracts

@startuml

!define CRUD\_SERVICE\_CONTRACTS <<(A,green) CRUD Service Contracts>>

start

:User (Admin) selects a service contract for management;

if (User chooses to Create) then (yes)

:User creates a new service contract and enters details;

:New service contract is saved in the system;

else (User chooses to Read, Update, or Delete) then (no)

:User views, updates, or deletes the selected service contract;

if (User chooses to Update) then (yes)

:Updated service contract details are saved in the system;

else (User chooses to Delete) then (no)

:Selected service contract is removed from the system;

endif

endif

stop

@enduml

31. CRUD Supply Contracts

@startuml

!define CRUD\_SUPPLY\_CONTRACTS <<(A,green) CRUD Supply Contracts>>

start

:User (Admin) selects a supply contract for management;

if (User chooses to Create) then (yes)

:User creates a new supply contract and enters details;

:New supply contract is saved in the system;

else (User chooses to Read, Update, or Delete) then (no)

:User views, updates, or deletes the selected supply contract;

if (User chooses to Update) then (yes)

:Updated supply contract details are saved in the system;

else (User chooses to Delete) then (no)

:Selected supply contract is removed from the system;

endif

endif

stop

@enduml

32. CRUD Potential Customers (Businesses)

@startuml

!define CRUD\_POTENTIAL\_CUSTOMERS <<(A,green) CRUD Potential Customers (Businesses)>>

start

:User (Admin) selects a potential customer (business) for management;

if (User chooses to Create) then (yes)

:User creates a new potential customer (business) and enters details;

:New potential customer (business) is saved in the system;

else (User chooses to Read, Update, or Delete) then (no)

:User views, updates, or deletes the selected potential customer (business);

if (User chooses to Update) then (yes)

:Updated potential customer (business) details are saved in the system;

else (User chooses to Delete) then (no)

:Selected potential customer (business) is removed from the system;

endif

endif

stop

@enduml

33. CRUD Information of Healthcare Facilities

@startuml

!define CRUD\_HEALTHCARE\_FACILITIES <<(A,green) CRUD Information of Healthcare Facilities>>

start

:User (Admin) selects a healthcare facility for management;

if (User chooses to Create) then (yes)

:User creates a new healthcare facility and enters details;

:New healthcare facility is saved in the system;

else (User chooses to Read, Update, or Delete) then (no)

:User views, updates, or deletes the selected healthcare facility;

if (User chooses to Update) then (yes)

:Updated healthcare facility details are saved in the system;

else (User chooses to Delete) then (no)

:Selected healthcare facility is removed from the system;

endif

endif

stop

@enduml