APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 1 (2025) 463-485
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Hannani Clinic Appointment System with D-OTP (HCAS)

Dheshini A/P Nagalingam[1], Khairul Amin Mohamad Sukri[1*]

[1] *Faculty of Science Computer and Information Technology,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: khairulm@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.026

Article Info

Received: 21 July 2024
Accepted: 19 june 2025
Available online: 30 June 2025

Keywords

Appointment system, Unique email one-time passcode (OTP), Account Lockout, Email Verification, Email Validation,Session Timeout,reCaptcha,

Abstract

The Hannani Clinic Appointment System with D-OTP (HCAS) is a comprehensive web-based platform designed to enhance appointment scheduling for patients at Clinic Dr Hannani, while ensuring robust data security. Utilizing an object-oriented approach, the system employs HTML, CSS, and JavaScript for development and styling, and MySQL with SQL queries for database management, guided by an Entity-Relationship Diagram (ERD). The primary objectives of this application are to design an email-based One Time Password (D-OTP) integration, develop the HCAS as a web-based platform, and thoroughly test its functionality. This outlines the implementation and testing of a security model within HCAS, incorporating features such as session timeout, encryption, password strength validation, username and email validation, account lockout, one-time passcode (OTP), and reCaptcha. The methodology involved coding, integrating these security features, and meticulously testing each module to ensure functionality and effectiveness. Testing results confirm that HCAS operates seamlessly, providing a user-friendly interface with advanced security measures, ensuring that all components function as intended and offer robust protection for user data. In conclusion, HCAS successfully integrates efficient scheduling capabilities with comprehensive security features, offering a reliable, secure, and user-friendly appointment system for Clinic Dr Hannani.

1. Introduction

Clinic Dr Hannani provides exceptional healthcare services to its patients. Established with a focus on quality care and patient satisfaction, the clinic offers a range of medical services designed to meet the needs of the community. The existing appointment scheduling system utilized by Clinic Dr Hannani via rumahsihatkita.com lacks proactive elements, encountering a scenario known as "phone tag," where two parties contact each other using telephone calls or messages. Despite Dr. Hannani Clinic's implementation of an online booking system, the current system lacks proper security mechanisms to authenticate users. This lack of security can lead to potential data breaches, exposing patient's personal information to attackers. Additionally, the absence of authentication measures allows for fake appointment requests, wasting doctor's time and preventing legitimate bookings. According to a study, robust user authentication is essential in healthcare systems to protect sensitive patient data with a reference to a journal article [1]. Therefore, to overcome these limitations, the Hannani Clinic Appointment System (HCAS) was developed. HCAS is a web-based system designed to simplify appointment scheduling for patients at Clinic Dr. Hannani. The system's goals include creating an object-oriented clinic appointment system, developing a web-based algorithm, and testing the Hannani Clinic Appointment System's user interface. Key features of HCAS encompass session timeout, encryption, password strength validation, reCAPTCHA, username and email validation, account lockout, one-time passcode (OTP), and email verification.

The security model within HCAS aims to ensure data integrity and user authentication, thereby providing a seamless experience for both patients and healthcare providers. As we embark on this journey to enhance the appointment scheduling experience, we delve into the intricacies of HCAS, where innovation converges with security to redefine healthcare accessibility and efficiency.

2. Related Work

This section discusses authentication features such as session timeout, encryption, password strength validation, username and email validation, account lockout, one-time passcode (OTP), and reCaptcha.

2.1 Authentication

The process of determining a user's identity or entity prior to interacting with the software or system is known as authentication. The three most crucial components of cybersecurity are secrecy, security, and authentication [2]. Authentication protocols are the basis of security in many distributed systems, and it is therefore essential to ensure that these protocols function correctly [3]. The authors provide a further discussion on password entropy, and the feasibility of such attacks on passwords.

2.1.1 Unique email OTP authentication

Unique email OTP authentication is a method used to verify the identity of users during the login process. It involves generating a one-time passcode (OTP) and sending it to the user's registered email address. The user then must enter OTP to complete the authentication process. This method adds an extra layer of security by ensuring that only the rightful user has access to their account.

2.1.2 reCaptcha

A reCAPTCHA is implemented to prevent automated software from accessing the system. This feature helps protect against spam bots and other automated attacks by verifying that the user is human.

2.1.3 Session Timeout

Session timeout is a security feature that allows users to remotely end each active session of the application. This functionality is crucial in case a user misplaces their mobile device or suspects their account has been compromised. Vulnerabilities in web servers and operating system that black hats could easily exploit these infrastructural elements.[8]

2.1.4 Username and Password Strength Validation

The code implements validation checks for usernames and passwords to enhance security. For usernames, it ensures that they meet specific criteria, such as containing at least one letter, symbol, and number, and being at least 6 characters long. For passwords, it enforces strong password policies, including a minimum length of 8 characters, at least one uppercase letter, one lowercase letter, one number, and one special character.

2.1.5 Email Validation

Email validation ensures that the email entered by the user meets certain criteria before proceeding with the registration process. It checks if the email already exists in the database and displays an error message if it does. This helps prevent duplicate accounts and ensures the integrity of user data.

2.1.6 Email Verification

The code sends a verification email to the user's registered email address, containing a email verification to login.This helps verify the user's email address and prevent email spoofing.

2.1.7 Account Lockout

When an account is locked out due to multiple failed login 3 times attempts, the system generates an email notification to inform the user. The email notification includes a countdown timer, indicating the remaining time before the account is locked out for a specified duration. This feature ensures that users are aware of the lockout and can take necessary actions to regain access to their account.

2.2 Comparison Between Existing System and Proposed System

This section provides an overview of three existing healthcare appointment booking systems: Thomson Hospital Kota Damansara, Klinik Dr Najwa, Gleneagles Hospital Medini Johor.

2.2.1 Thomson Hospital Kota Damansara

Thomson Hospital Kota Damansara's online system seems to offer basic appointment booking services through their website. It appears that the website primarily serves as an informational platform about the hospital's services, specialties, and facilities rather than facilitating online appointment scheduling directly through the site.

2.2.2 Klinik Dr Najwa

Similar to Thomson Hospital Kota Damansara, the website serves more as a source of general information about the clinic rather than offering a comprehensive online appointment booking system directly on the site. Patients may need to contact the clinic via phone or visit in person to schedule appointments.

2.2.3 Gleneagles Hospital Medini Johor

While direct appointment booking capabilities are not explicitly mentioned on the provided link, Gleneagles Hospital's website typically offers online appointment scheduling features. Patients can access information about medical specialties, doctors, facilities, and other services offered by the hospital. They may also be able to schedule appointments online through the hospital's official website or patient portal. However, without direct access to the appointment booking system, it's challenging to provide detailed information about its functionality.

Table 1 *Comparison between existing system and proposed system*

Features	Thomson Hospital Kota Damansara	Klinik Dr Najwa	Gleneagles Hospital Medini Johor	Proposed System
Login	X	X	X	✓
Registration	X	X	X	✓
Appointment Booking	✓	✓	✓	✓
Reminder	X	X	X	✓
Notification Authentication method	X	X	X	✓
Email verification	X	X	✓	✓
Unique email OTP authentication	X	X	X	✓
Account Lockout	X	X	X	✓
Email Validation	X	X	✓	✓
Username and Password Strength Validation	X	X	X	✓
Session Timeout	X	X	X	✓
reCaptcha	X	X	✓	✓

Based on the comparison, Thomson Hospital Kota Damansara and Klinik Dr Najwa offer basic appointment booking functionalities without advanced features such as login, registration, reminders, notifications, or advanced authentication methods. Gleneagles Hospital Medini Johor provides a more comprehensive online healthcare experience, including user accounts, appointment booking, reminders, notifications, email verification, and reCaptcha. The proposed system aims to bridge the gap by offering similar advanced features, including login, registration, reminders, notifications, authentication methods, and security measures like reCaptcha, to enhance the overall appointment scheduling experience for patients.

3. Methodology

Object-oriented system development model is used for the methodology of the proposed system. There are five phases in this methodology. There are requirement analysis, design, implementation, testing, and maintenance. Object Oriented System Development (OOSD) considers and object active if it displays independent motive power[4]. To break down problem definitions into discrete entities,Object Oriented Design (OOD) requires programmers to establish principles. They also need to ascertain the relationships between the entities, which ultimately lead to the problem's solution. The Object-Oriented Analysis (OOA) is basically collected works of concurring or cascading system modelling, incorporate various requirements and pre and post analysis methodology for software system [5].

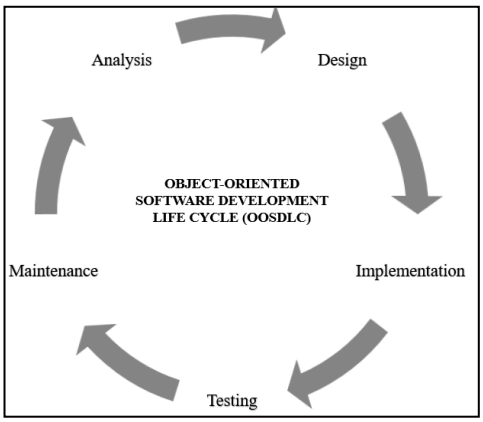

Fig. 1 *Object-oriented Software Development Model (OOM)*

3.1 Object-Oriented Analysis

In the analysis phase of the Hannani Clinic Appointment System (HCAS) development process, obtaining an in-depth knowledge of the issue domain and gathering requirements are the main priorities. This is accomplished by conducting interviews and gathering data in an organized manner.

3.1.1 Software and Hardware requirement

Table 2 *The software and hardware requirement*

Tools /Techniques	Description
Software	(i) XAMPP version 3.3.0 - which stands for Cross-Platform, Apache, MySQL, PHP, Perl
	(ii) Programming languages : HTML,CSS,JAVASCRIPT,PHP,
	(iii) Operating system : Windows 11 Home Single Language
Hardware	(iv) Processor : Intel(R) Celeron(R) N4500 @ 1.10GHz 1.11 GHz
	(v) RAM : 3,81 GB
	(vi) Hard Disk space : 173 GB
	(vii) Monitor : 1920 x 1080

3.2 Object-Oriented Design

In this Object-oriented design, design refines the model produced in the requirements analysis phase into a software architecture and defines a language-specific representation of that architecture [6]. This involves identifying essential elements and outlining the system's architecture into a detailed design context for instance, Unified Modeling Language (UML), Entity Relationship Diagram (ERD) and flowchart.

3.3 Object-Oriented Implementation

From ERD to guarantee that the database implementation complies with the given constraints and relationship, hence promoting data integrity. Use the ERD, UML, and flowcharts as a reference once more. Next, use the HTML programming language during implementation, making use of the UML diagram to guarantee that the components are represented correctly. The HTML element is then styled by CSS to provide a unified and aesthetically pleasing user interface. Moreover, JavaScript implements client-side functionalities by using flowcharts and UML diagrams to describe dynamic behaviors. The database schema will be implemented using MySQL and SQL queries based on the ERD will be run.

3.4 Object-Oriented Testing

Once the system is implemented, it needs to undergo various testing activities to ensure that it meets the specified requirements, functions correctly, and is free of defects. The outline of the testing plan after the implementation phase is unit testing. Ensure that the system's user interface (UI) meets user expectations and provides positive user experience. The feedback collection on the visual design, navigation and overall usability. Next, the user acceptance testing (UAT). Verify that the system meets the business requirements and is ready for deployment. Verify the accuracy and integrity of data processed by the system. Lastly, functionality testing which will access the functionality of the entire system, ensuring that each feature works as intended[10].

3.5 Object-Oriented Maintenance

The software development process moves into the maintenance phase following the testing phase. Maintenance includes both adaptive maintenance, which focuses on modifying the system and corrective maintenance, which addresses flaws and problems found during testing or reported by users. This continuous cycle of observation, evaluation, and feedback gathering adds to the system's durability and long-term capacity to satisfy user needs.

4. System Analysis and Design

This section explains the design of the system which consists of functional and non-functional requirements, system analysis and system design.

4.1 Functional Requirement

Functional requirements define the data that must be kept track of to accomplish the user tasks. The Hannani Clinic Appointment System (HCAS) needs to meet the following functional requirements in Table 3,

Table 3 *Functional Requirement*

Module	Functionalities
User Registration	Patients must register to use the clinic's facilities, providing necessary personal information.
Login System	Implement a secure login system with two-factor authentication, reCaptcha and unique email OTP verification with password policy
Appointment Module	Enable users to review, reschedule, and cancel appointments easily
Admin Module	Allow system administrators to configure back-end settings, manage user access, and set security parameters including password reset period, inactive account lockout, inactive session time out
Notification Module	Implement a notification system to remind users of upcoming appointments.

4.2 Non-Functional Requirement

Non-functional requirements specify the criteria that are essential for the system's success but are not related to specific functionalities. Non-functional requirements describe a variety of system characteristics: operational, performance and security. For the HCAS, non-functional requirements in Table 4,

Table 4 *Non-Functional Requirements*

Requirements	Description
Security	Ensure the confidentiality and privacy of user information through robust security measures.
Usability	Design an intuitive and user-friendly interface for easy navigation and interaction.
Performance	The system should handle a reasonable number of simultaneous users without significant performance degradation.
Reliability	Ensure the system is available and reliable, minimizing downtime and errors

4.3 Unified Modelling Language (UML)

Unified Modeling Language (UML) is a standardized visual modeling language widely utilized in software engineering to depict and communicate complex systems throughout the software development life cycle. UML includes a set of graphic notation techniques to create visual models of object-oriented software system [7].

4.3.1 Use Case Diagram

Fig. 2 shows patients interact with the system by first registering and logging with verified unique email OTP. Once registered, the patient can book appointments, schedule and cancel the appointments. Patients also can view appointment details and the result, and lastly sign-out.

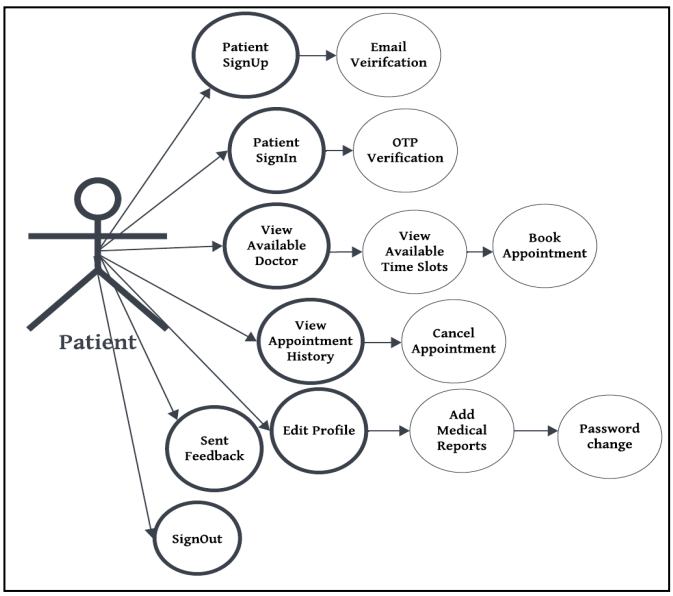

Fig. 2 *Use case diagram for patient*

Fig. 3 shows Doctor interacts with the system by first signing in. Doctors can manage patient details which can list the patients. Next, doctor can manage patient appointments which is add appointment and update appointment. Then, doctor also can generate patient reports and lastly sign out.

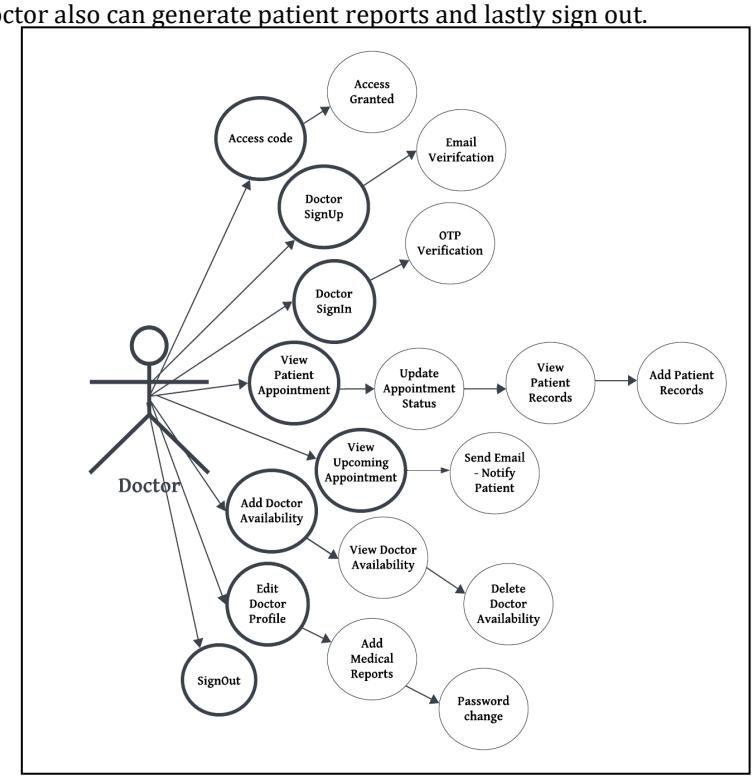

Fig. 3 *Use case diagram for doctor*

Fig. 4 shows Admin interacts with the system by signing in. Firstly, admin manage clinic management which is admin can add,edit,update clinic information. Next patient management which admin can list patient

details, delete patient and add patient. Then, for doctor management which admin can list,add,delete,update the doctors.

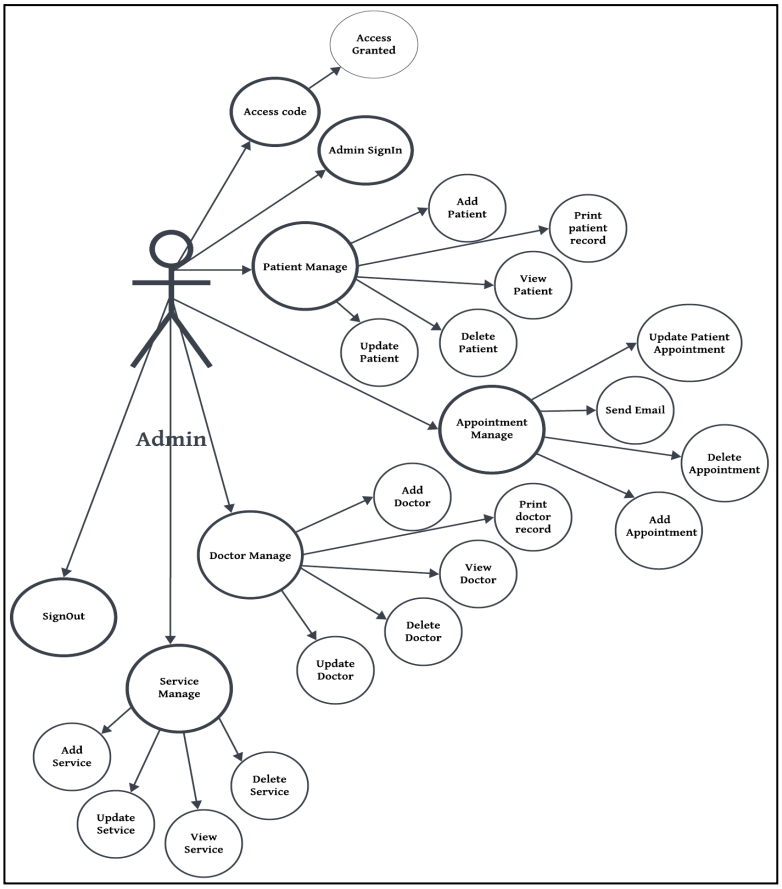

Fig. 4 *Use case diagram for admin*

4.3.2 Sequence Diagram

Fig. 5 shows that the process begins with the patient register with verifying email. After successful registration, the patient can log in using their credentials and generates verifying unique email OTP. The patient interacts with the system to book appointments and view their booking details, highlighting a streamlined process for managing healthcare appointments securely.

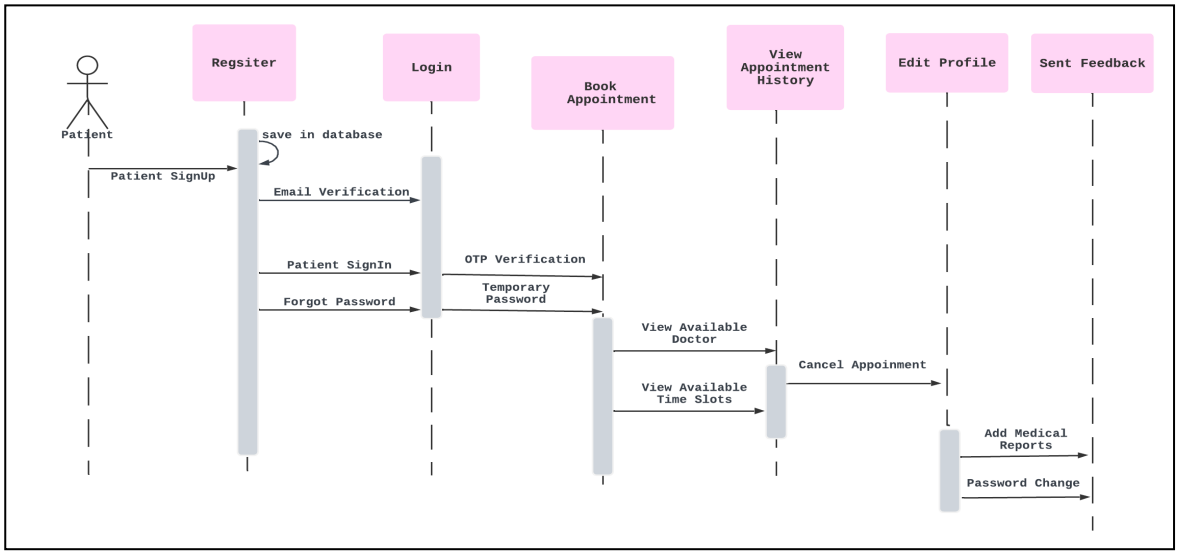

Fig. 5 *Sequence diagram of patient*

Fig. 6 shows that the login process initiates as the admin logs into the system. Once authenticated, the admin engages with clinic management to oversee and update clinic details, followed by interactions with patient management and doctor management to manage patient records and doctor information. Finally, the admin utilizes the appointment management module to oversee and organize appointments.

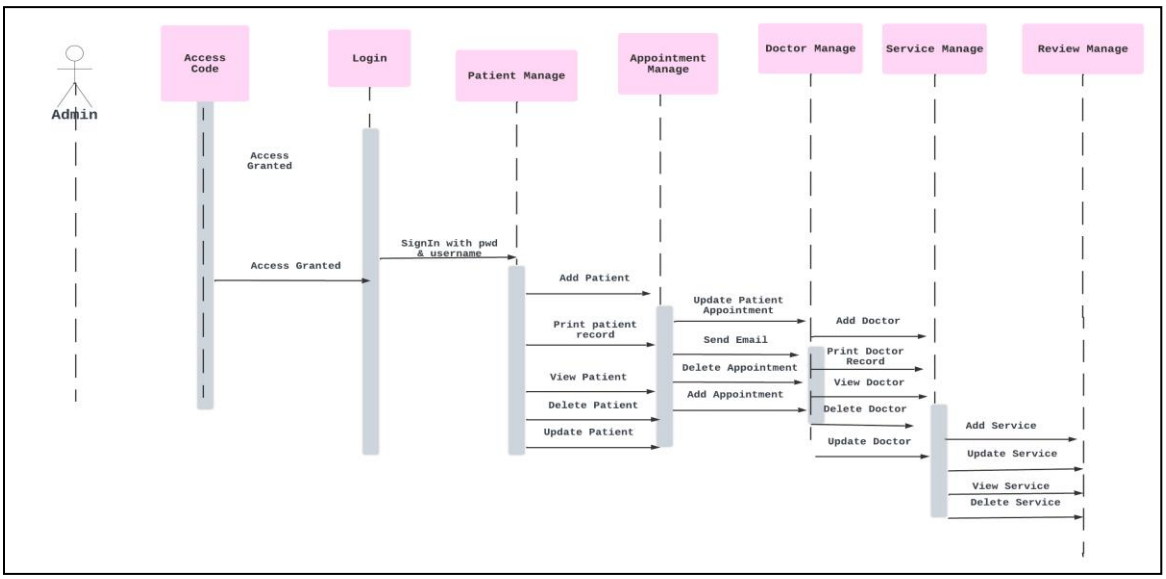

Fig. 6 *Sequence diagram of admin*

Fig. 7 shows that the process begins with the doctor logging into the system. After successful authentication, the doctor interacts with the patient management system to access and manage patient records. Subsequently, the doctor engages with the appointment management module to view and manage appointments, and finally, utilizes the report management system to generate and manage medical reports.

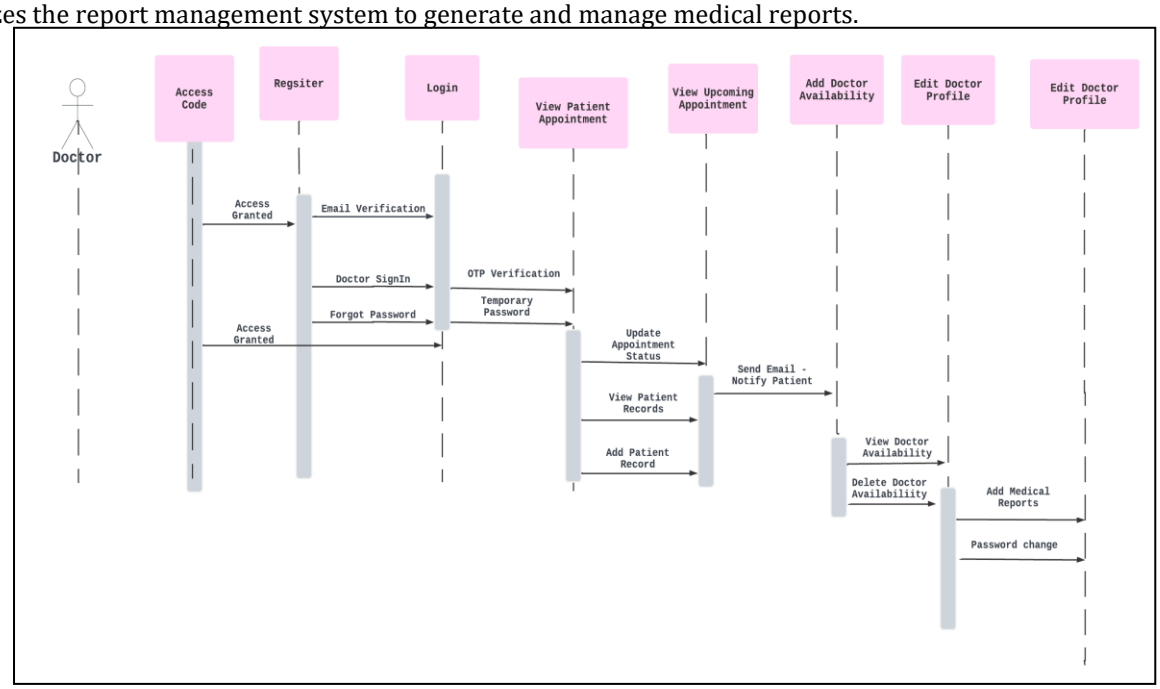

Fig. 7 *Sequence diagram of doctor*

4.3.3 Activity Diagram

Fig. 8 activity diagrams show patient in a healthcare system outlines the sequential flow of actions that a patient can perform.Activities can be linked to one another by transition lines or decision points [8]. The activity diagram shows in Fig. 9 is for a doctor in a healthcare system illustrates the sequence of activities from initial access to various functionalities and lastly,the activity diagram in Fig. 10 is for an admin in a healthcare system outlines the workflow starting from entering an access code.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 463-485

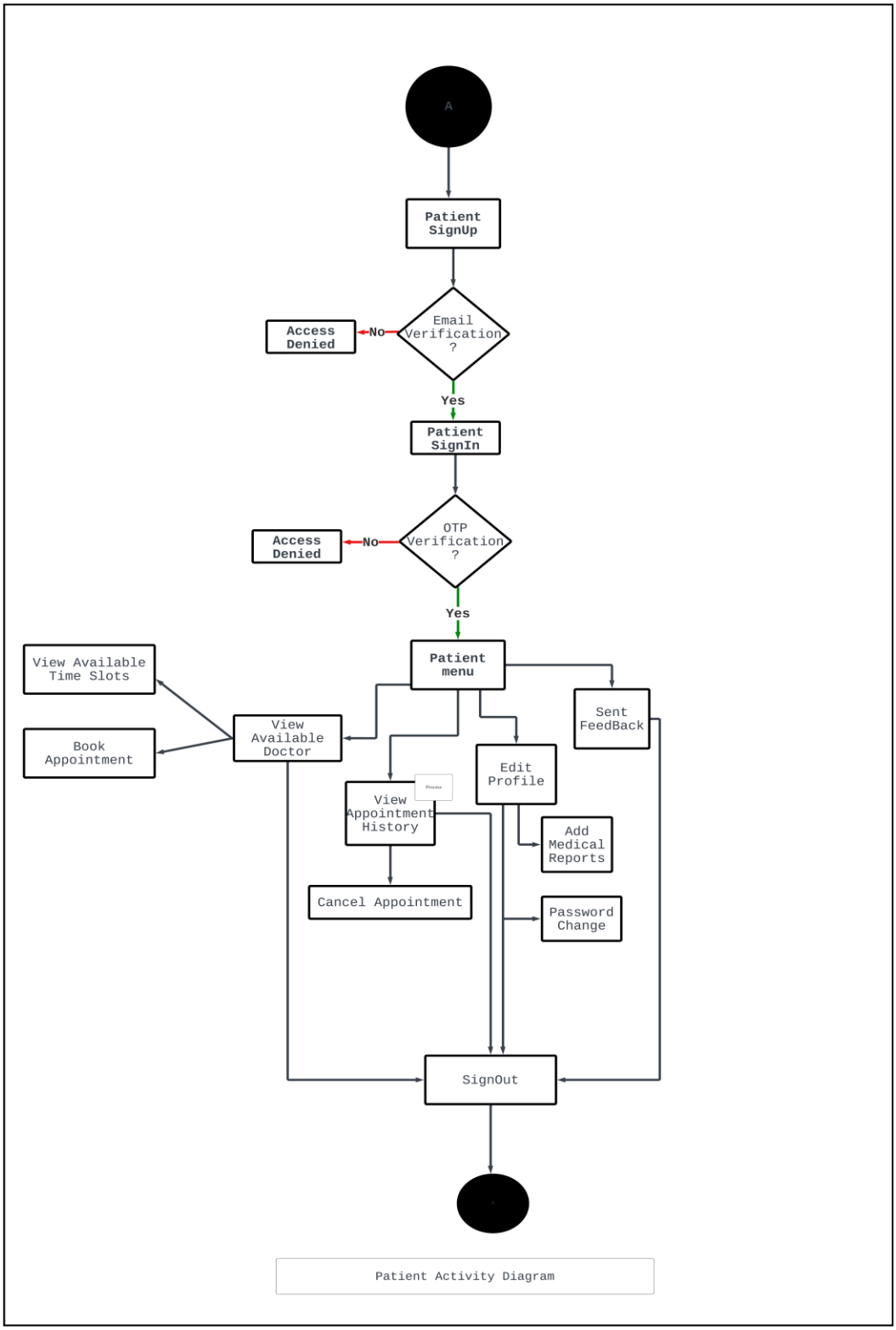

Fig. 8 *Activity diagram for the patient*

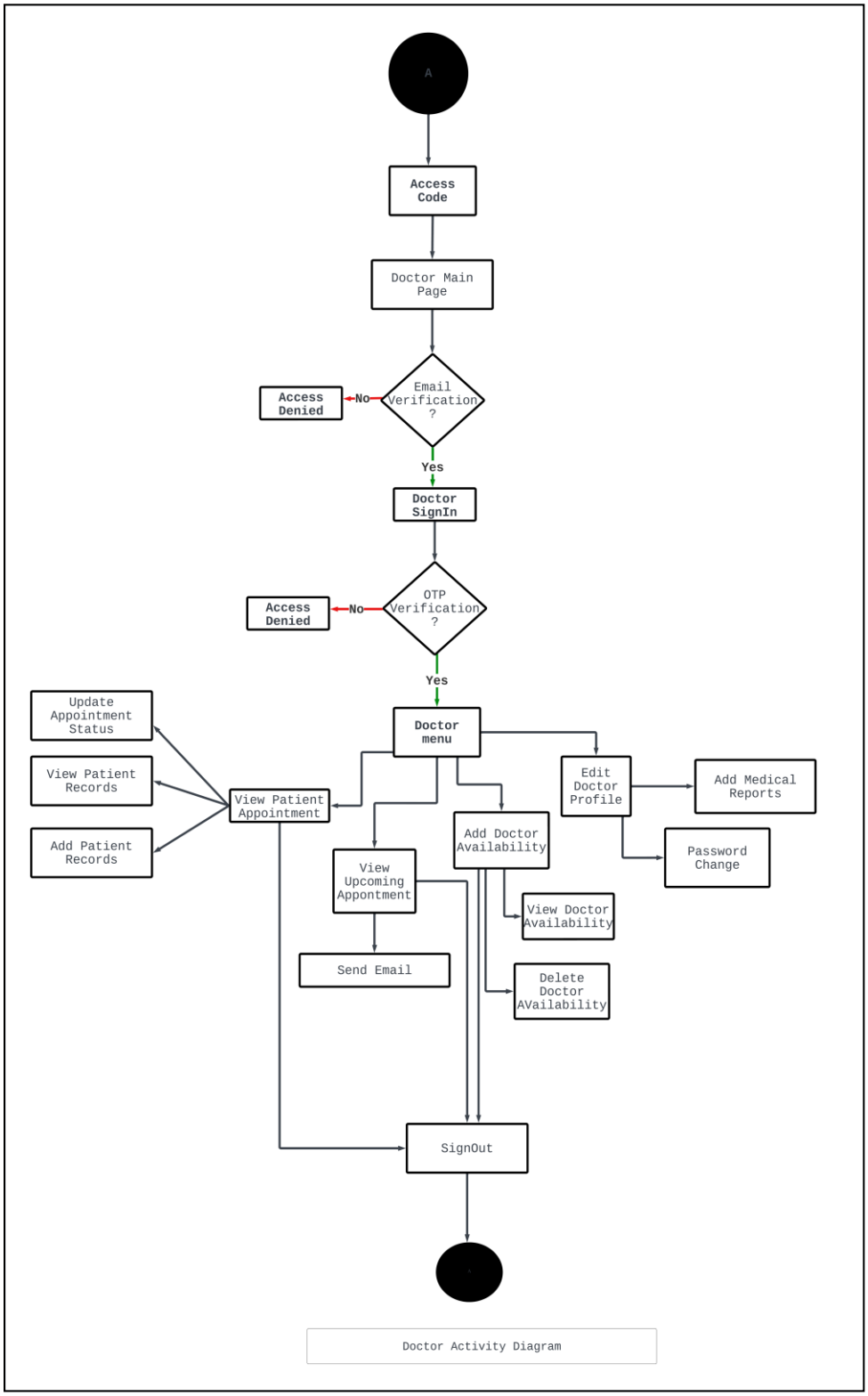

Fig. 9 *Activity diagram for the doctor*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 463-485

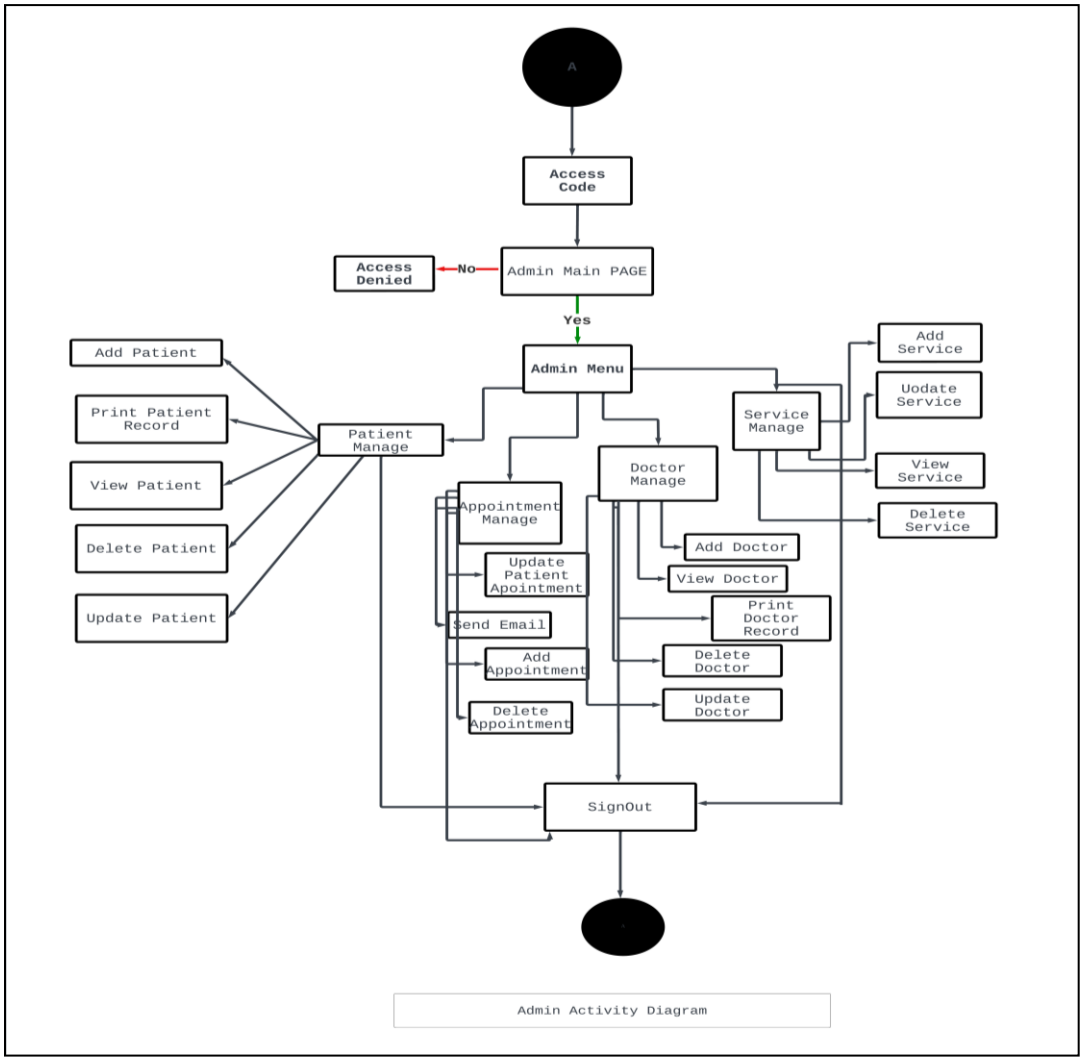

Fig. 10 *Activity diagram for the admin*

4.3.4 Class Diagram

Fig. 11 shows that class diagram illustrates the structure and relationships between key entities in the Hannani Clinic Appointment System, including Admin, Patient, Doctor, Appointment,Schedule, Database, and OTP Service.

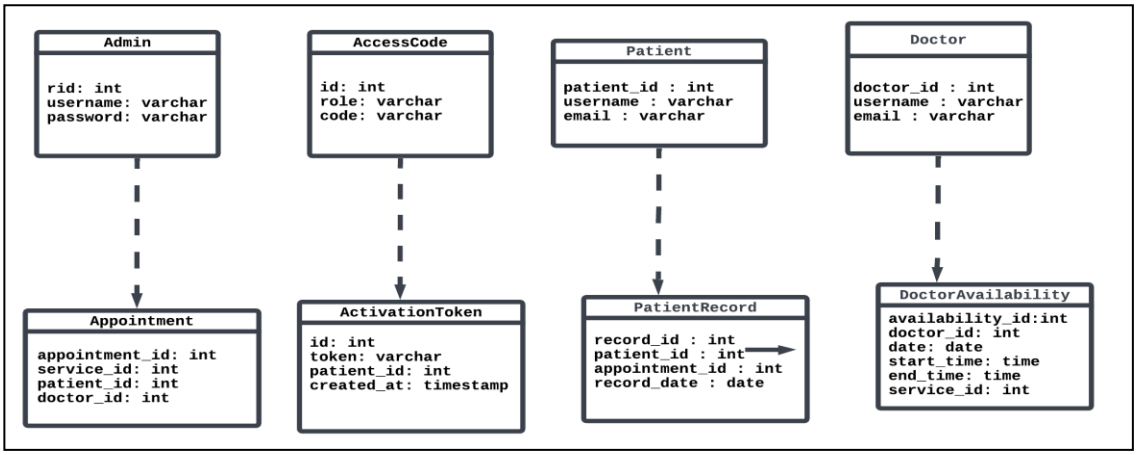

Fig. 11 *Class diagram for the proposed system*

4.4 Database Design

Database design is part of the database development process that involves analysis of a problem definition (specifications and requirements) and provides all necessary findings for building a logical structure of data [8]. The goal of database design is to produce a database schema that represents the logical structure of the database system and meets the requirements of the users and the application.

4.4.1 Entity Relationship Diagram (ERD)

Fig. 12 shows that the Entity Relationship Diagrams (ERDs) and most of the contemporary database design tools can transform the diagrams into metadata [8]. ERDs are commonly used during the database design phase to communicate the structure of a database in a clear and concise manner.

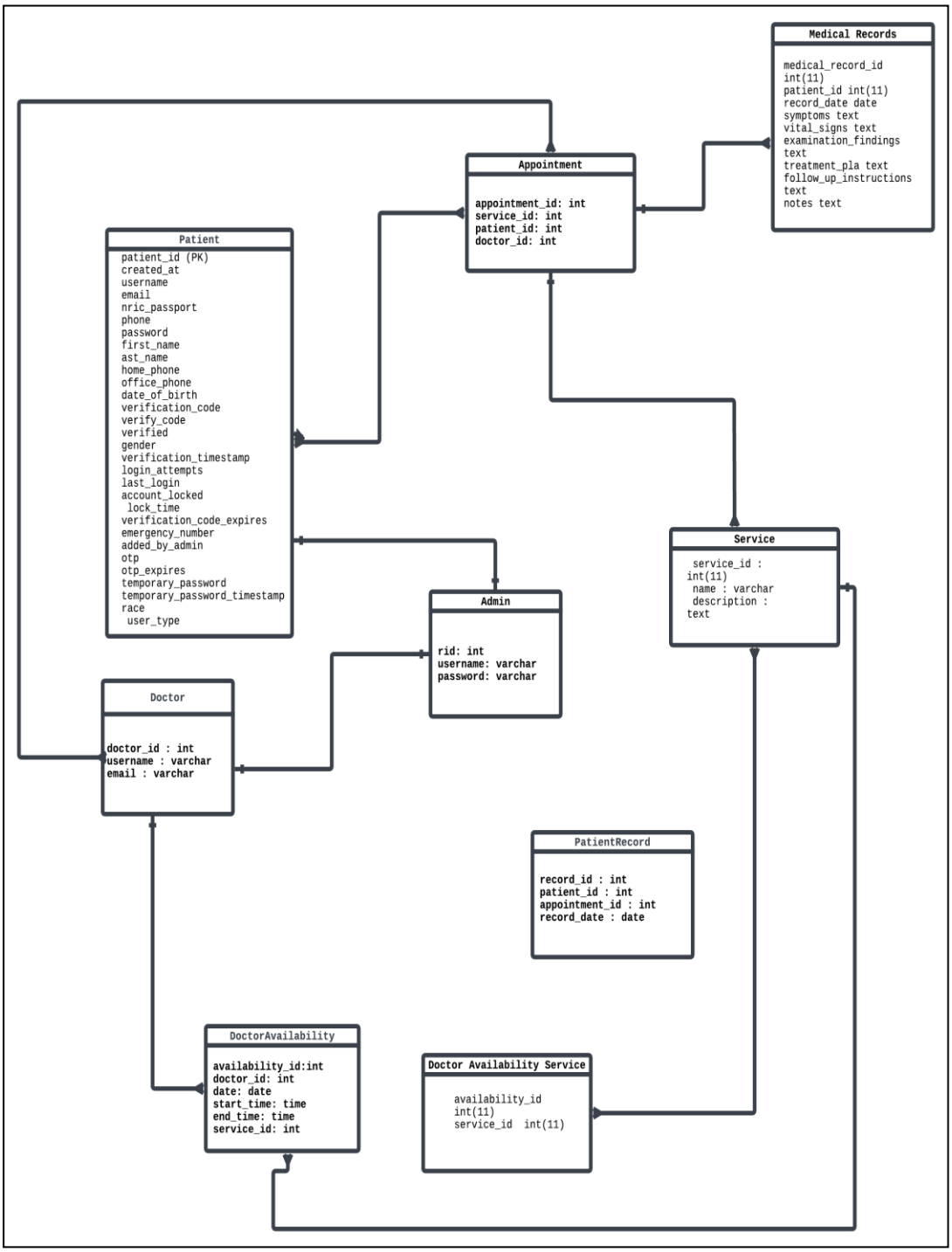

Fig. 12 *The ERD diagram for proposed system*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 463-485

4.5 Interface Design

The wireframe diagram for the Hannani Clinic Appointment System (HCAS) is used to design and communicate the structure of the user interface. It provides a clear, visual representation of the various components and their arrangement on the screen. It enables a user to perform all defined tasks in a manner that meets every usability goal defined for the system [10]. The wireframe of the HCAS is presented in Fig. 13, Fig.14 and Fig.15.

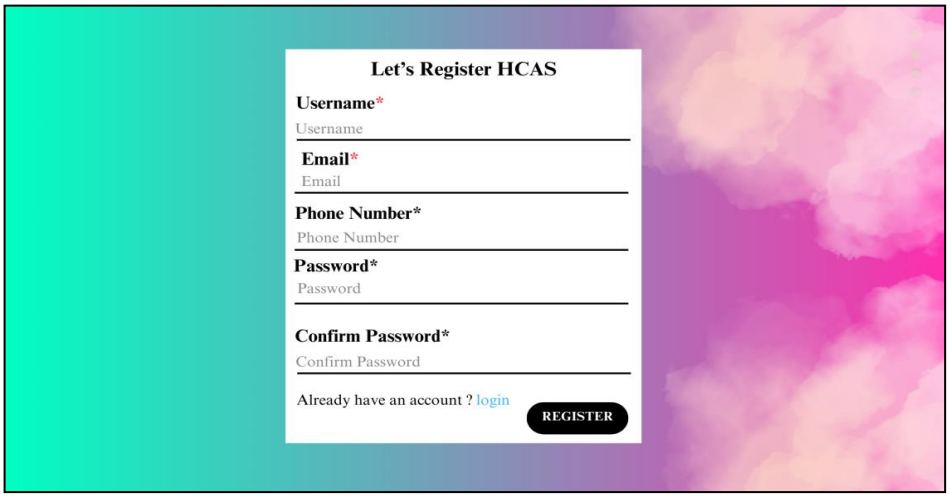

Fig. 13 *The wireframe of registration page*

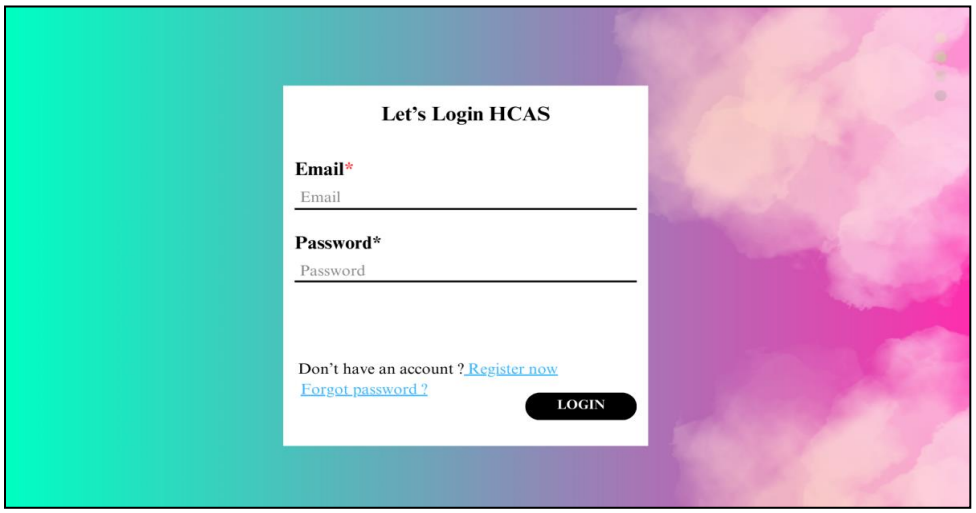

Fig. 14 *The wireframe of login page*

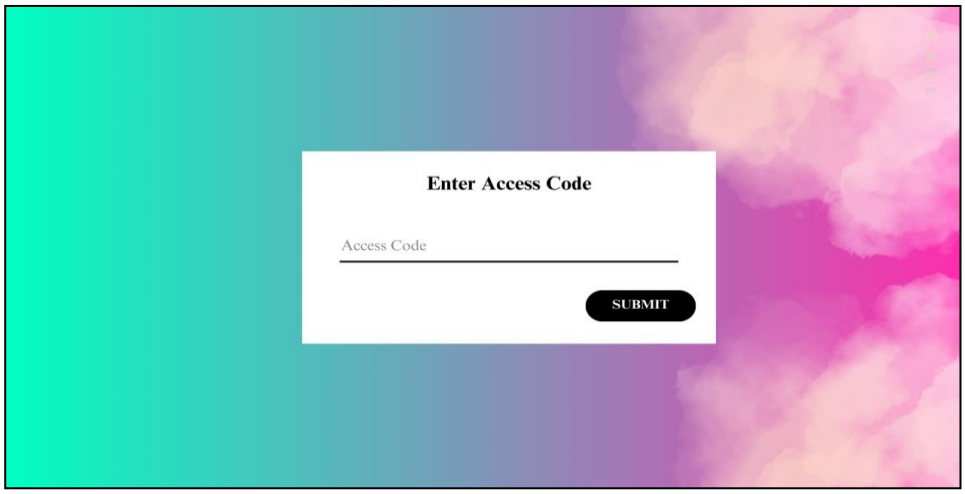

Fig. 15 *The wireframe of account setting page*

5. System Implementation and Testing

This phase involves converting the design into a functional system followed by developing testing to ensure the system meets all requirements and performs reliably. MySQL is chosen for its robustness, ease of integration with PHP, and ability to handle complex queries and large volumes of data efficiently.

5.1 Implementation of Security Module

One critical aspect of the implementation phase is the implementation of a security model to ensure the protection of user data and privacy.

5.1.1 Implementation of Session Timeout

A security feature called "session timeout" enables the user to remotely end each and every active session of the application. In the event that a user misplaces their mobile device or believes their account has been hijacked, this functionality is crucial. According to an Open Web Application Security Project (OWASP) Top Ten study, “session management is critical to ensuring the security of web applications, and session timeout is an essential component of session management"[9].

session.php

```php
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

// Set session timeout duration in seconds (2 minutes = 120 seconds)
$timeout_duration = 300;

if (isset($_SESSION['LAST_ACTIVITY'])) {
    if (time() - $_SESSION['LAST_ACTIVITY'] > $timeout_duration) {
        session_unset();
        session_destroy();
        header("Location: logout.php?timeout=1");
        exit();
    }
}

$_SESSION['LAST_ACTIVITY'] = time();
?>
```

Fig. 16 *Partial code for session timeout*

Fig. 17 *The alert message box for session timeout*

5.1.2 Implementation of Encryption

The code uses password_hash() to store passwords securely, ensuring that passwords are not stored in plaintext. Hashing Password, If there are no errors, it uses the password_hash() function to securely hash the user's password. PASSWORD_DEFAULT, a constant that represents the default algorithm, which is currently bcrypt. Using PASSWORD_DEFAULT ensures that your code will automatically use the best available algorithm in future PHP versions.

patient_register.php

```php
// If there are no errors, proceed with registration
if (empty($username_err) && empty($email_err) && empty($phone_err) && empty($password_err) && empty($cpassword_err)) {
    // Hash password
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);

```

Fig. 18 *Partial code for password hashing*

	doctor_id	username	email	phone	password
elete	72	Teddy00!	teddyboy2910@gmail.com	5689784589	$2y$10$FR5O.Kni11d6.N.IfTp/pO3c0.L0MPdsXW3tBMJJjf2...
elete	73	Persheela22!	persha1473@gmail.com	0103843391	$2y$10$pFs/CfPrE.qc/yyrnCDkxOgkAP3ZWx3e85gITR8wn4Z...

Fig 19 *The hashed* **password** *in database*

5.1.3 Implementation of Username and Password Strength Validation

The code initializes variables for username, email, phone, password, and confirmation password, along with error variables for validation messages. For username validation, it ensures that the username meets specific criteria to enhance security and prevent users from choosing weak usernames. For password validation, it enforces strong password policies to mitigate the risk of unauthorized access.

patient_register.php

```php
// Validate username
if (empty(trim($_POST['username']))) {
    $username_err = 'Please enter a username.';
} else {
    $username = trim($_POST['username']);
    // Check if username contains at least one letter, one symbol, and one number
    if (!preg_match('/^(?=.*[A-Za-z])(?=.*\d)(?=.*[^A-Za-z\d\s])\S{6,}$/', $username)) {
        $username_err = '<span class="error">Username should contain at least one letter, one symbol, one number, and be at least 6 characters long.</span>';
    } else {
        // Check if username already exists
        $sql_username_check = "SELECT username FROM patients WHERE username = ?";
        $stmt_username_check = mysqli_prepare($conn, $sql_username_check);
        mysqli_stmt_bind_param($stmt_username_check, "s", $username);
        mysqli_stmt_execute($stmt_username_check);
        mysqli_stmt_store_result($stmt_username_check);
        if (mysqli_stmt_num_rows($stmt_username_check) > 0) {
            $username_err = 'This username is already taken. Please use a different username.';
        }
        mysqli_stmt_close($stmt_username_check);
    }
    // Check if username is the same as the password
    if ($username === $password) {
        $username_err = '<span class="error">Username should not be the same as the password.</span>';
    }
}
```

patient_register.php

```php
// Validate password
if (empty(trim($_POST['password']))) {
    $password_err = '<span style="color: red;">Please enter a password.</span>';
} elseif (strlen(trim($_POST['password'])) < 6) {
    $password_err = '<span style="color: red;">Password must be at least 6 characters long.</span>';
} else {
    $password = trim($_POST['password']);
    // Password strength validation
    if (!preg_match('/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%^&*()_+{":;\'?\/><.,|`~])(?=.*[^\w\d\s]).{8,}$/', $password)) {
        $password_err = '<span style="color: red;">Password must contain at least 8
        characters with at least one uppercase letter, one lowercase letter,
        one number, and one special character.</span>';
    }
```

Fig. 20 *Partial code for username & password strength validation*

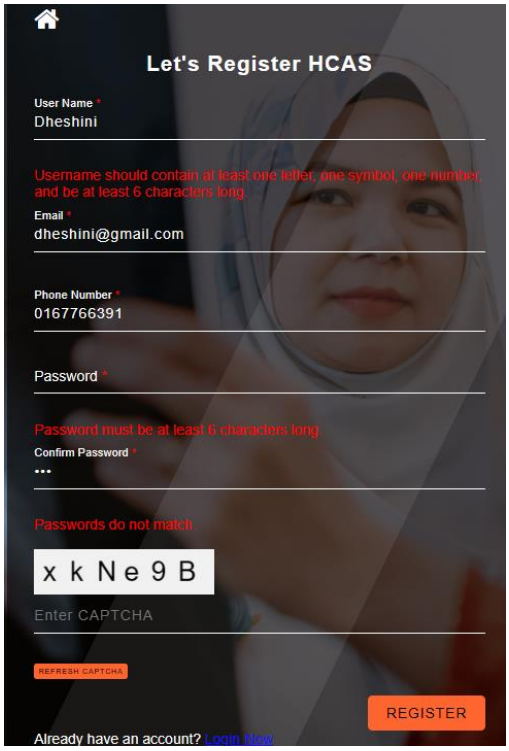

Fig. 21 *The interface for the register and the username and password strength validation*

5.1.4 Implementation of reCaptcha

The code implements a reCAPTCHA to prevent automated software from accessing the system.

patient_register.php

```
// Verify reCAPTCHA
$recaptcha_secret_key = "6LcU6vQpAAAAAFQj3OcxaeCcWuRrvXx3YZU6Jttu"; // Replace wit
$recaptcha_response = $_POST['g-recaptcha-response'];

// Make POST request to Google reCAPTCHA API
$recaptcha_url = 'https://www.google.com/recaptcha/api/siteverify';
$recaptcha_data = [
    'secret' => $recaptcha_secret_key,
    'response' => $recaptcha_response,
];

$recaptcha_options = [
    'http' => [
        'method' => 'POST',
        'content' => http_build_query($recaptcha_data)
    ]
];

$recaptcha_context = stream_context_create($recaptcha_options);
$recaptcha_result = file_get_contents($recaptcha_url, false, $recaptcha_context);
$recaptcha_response_data = json_decode($recaptcha_result);

if (!$recaptcha_response_data->success) {
    // CAPTCHA verification failed
    echo "<script>alert('Please complete the CAPTCHA verification.');</script>";
} else {
    // CAPTCHA verification successful
    // Proceed with other form processing
}
```

Fig. 22 *Partial code for the reCaptcha*

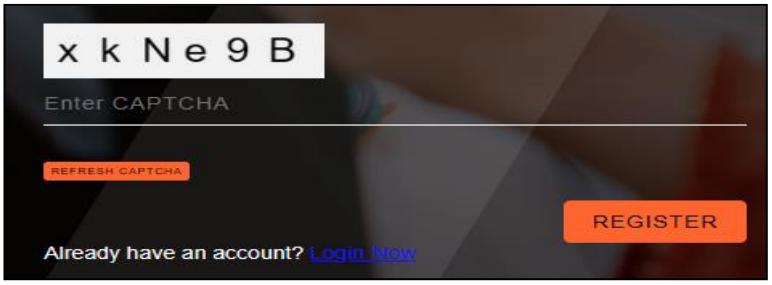

Fig. 23 *The interface for the reCaptcha*

5.1.5 Implementation of Email Validation

The email validation in the provided PHP code ensures that the email entered by the user meets certain criteria before proceeding with the registration process. After trimming the email input, the code checks if the entered email already exists in the database.

patient_register.php

```
// Validate email
if (empty(trim($_POST['email']))) {
    $email_err = 'Please enter your email.';
} else {
    $email = trim($_POST['email']);
    // Check if email already exists
    $sql_email_check = "SELECT email FROM patients WHERE email = ?";
    $stmt_email_check = mysqli_prepare($conn, $sql_email_check);
    mysqli_stmt_bind_param($stmt_email_check, "s", $email);
    mysqli_stmt_execute($stmt_email_check);
    mysqli_stmt_store_result($stmt_email_check);
    if (mysqli_stmt_num_rows($stmt_email_check) > 0) {
        $email_err = 'This email is already registered.';
    }
    mysqli_stmt_close($stmt_email_check);
}
```

Fig. 24 *Partial code for email validation*

Fig. 25 *The interface for email validation*

5.1.6 Implementation of Account Lockout

When an account is locked out, the system generates an email notification to inform the user of the lockout. The email notification also provides a countdown timer, indicating that the user has 5 minutes to login before the account is locked out for the full 20 minutes.

patient_login.php

```
109             // Check if the account is locked
110             if ($user['account_locked'] == 1) {
111                 // Check if lock duration has expired (20 minutes)
112                 $lockTime = strtotime($user['lock_time']);
113                 $currentTime = time();
114                 $lockDuration = 20 * 60; // 20 minutes in seconds
115                 if ($currentTime - $lockTime >= $lockDuration) {
116                     // Account lock duration expired, unlock the account
117                     unlockAccount($email);
118                 } else {
119                     // Account still locked, inform the user
120                     $remainingTime = $lockDuration - ($currentTime - $lockTime);
121                     $remainingMinutes = ceil($remainingTime / 60);
122 echo "<script>
123     alert('Your account is locked. Please try again after $remainingMinutes minutes.');
124     window.location.href = 'patient_login.php';
125 </script>";
126                     exit();
127                 }
128             }
```

Fig. 26 *The partial code for the account lockout*

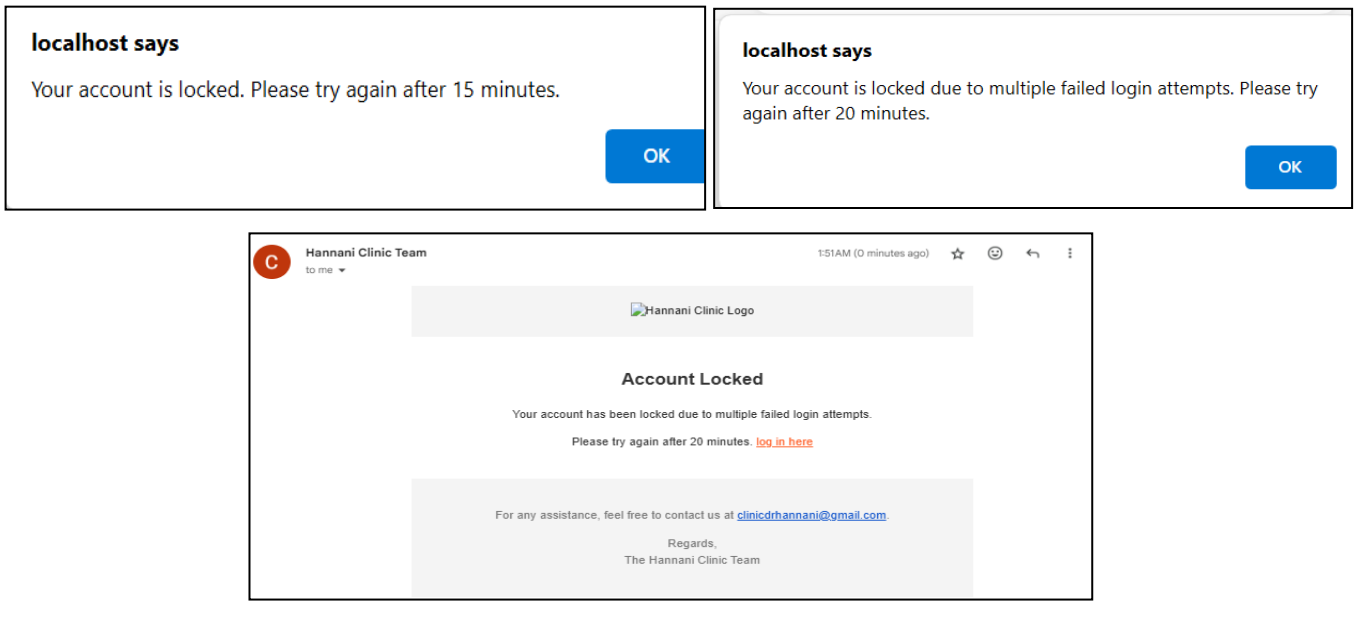

Fig. 27 *The interface for alert box message and the email notification*

5.1.7 Implementation of One Time Passcode (OTP)

The code generates a unique One-Time Passcode (OTP) for each user and doctor, which is sent to their registered email address. This adds an extra layer of security to the login process. When a user or doctor attempts to log in, a unique email OTP is generated and sent to their registered email address. This OTP is required to complete the login process, ensuring that only the intended user can access their account.

patient_login.php

```php
// Function to generate OTP
function generateOTP() {
    $characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*()';
    $characters_length = strlen($characters);
    $otp_length = 6;
    $otp = '';
    $used_indexes = [];

    // Generate an OTP with unique characters
    for ($i = 0; $i < $otp_length; $i++) {
        $random_index = rand(0, $characters_length - 1);

        // Ensure the character at $random_index is not already used
        while (in_array($random_index, $used_indexes)) {
            $random_index = rand(0, $characters_length - 1);
        }

        // Add the character to the OTP
        $otp .= $characters[$random_index];
        $used_indexes[] = $random_index;
    }

    return $otp;
}
```

Fig. 28 *The partial code for the OTP Generation*

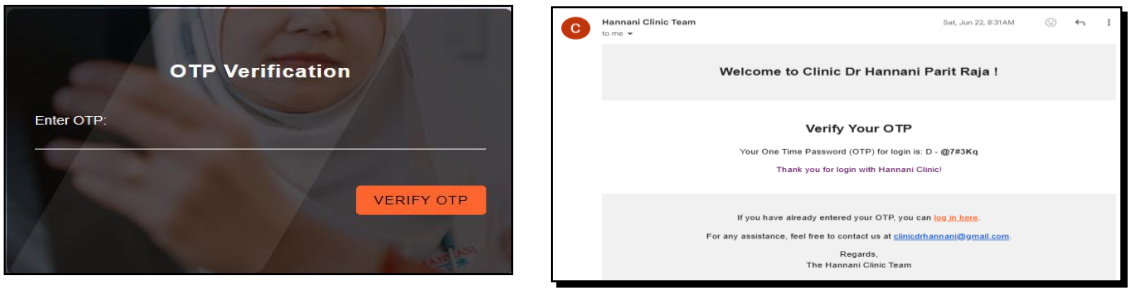

Fig. 29 *The interface for the OTP generates*

5.2 Implementation of Module

This section covers the implementation of various modules in the Healthcare Appointment Scheduling (HCAS) application. Each module is designed to perform specific functions and improve the overall user experience.

5.2.1 Implementation of Register Module

For the register module, the function for doctor and patient is same is validates the input data for the username, email, phone, password, and confirm password fields.The validation checks for empty fields, minimum length, and password strength.

patient_register.php

```php
<h2>Let's Register HCAS</h2>
    <div class="input-group">
    <input type="text" name="username" value="<?php echo $username; ?>" required />
    <label for="name">User Name <span style="color: red;">*</span></label> <!-- Add asterisk symbol -->
        <span class="error" style="color: red"><?php echo $username_err; ?></span>
    </div>

    <br>
   <div class="input-group">
    <input type="email" name="email" value="<?php echo $email; ?>" required />
    <label for="email">Email <span style="color: red;">*</span></label> <!-- Add asterisk symbol -->
    <!-- Your error message handling for email -->
    <?php if (!empty($email_err)) echo "<span style='color: red' class='error'>$email_err</span>"; ?>
    </div>
    <br>

    <div class="input-group">
        <input type="text" name="phone" value="<?php echo $phone; ?>" />
        <label for="phone">Phone Number <span style="color: red;">*</span></label> <!-- Add asterisk symbol -->
        <!-- Your error message handling for phone -->
        <?php if (!empty($phone_err)) echo "<span class='error'>$phone_err</span>"; ?>
    </div>

    <div class="input-group">
        <input type="password" name="password" value="<?php echo $password; ?>" required onkeyup="checkPasswordStrength(this.value)" />
        <label for="phone">Password <span style="color: red;">*</span></label> <!-- Add asterisk symbol -->
        <!-- Your error message handling for password -->
        <?php if (!empty($password_err)) echo "<span class='error'>$password_err</span>"; ?>
    </div>
    <div id="password-strength-indicator"></div> <!-- Password strength indicator -->
    <br>
    <div class="input-group">
        <input type="password" name="cpassword" value="<?php echo $cpassword; ?>" required />
        <label for="phone">Confirm Password <span style="color: red;">*</span></label> <!-- Add asterisk symbol -->
        <!-- Your error message handling for confirm password -->
        <?php if (!empty($cpassword_err)) echo "<span class='error'>$cpassword_err</span>"; ?>
    </div>
```

Fig. 30 *The interface for the register module*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 463-485

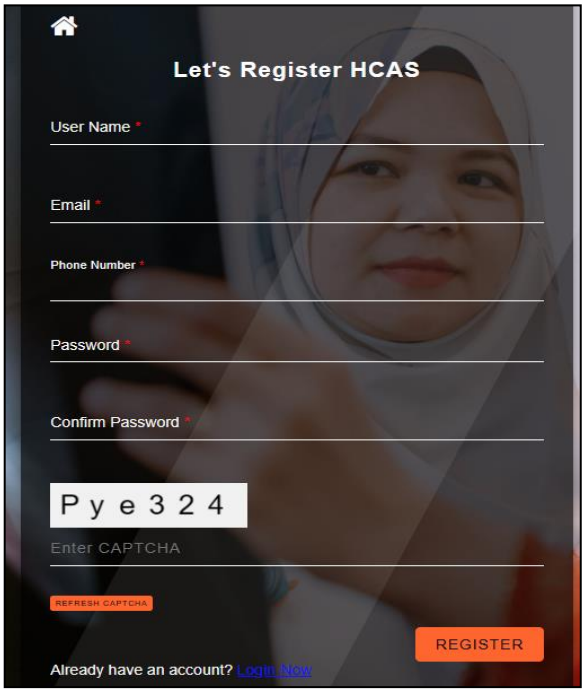

Fig. 31 *The partial code for the register module*

5.2.2 Implementation of Login Module

The login module, start by validates the input data for the email and password fields. Then, after the code match generates an unique email OTP and sends it via email using the sentOTP( ) function. If the OTP is sent successfully, the code redirects the user to the OTP verification page.

Fig. 32 *The partial code for the login module*

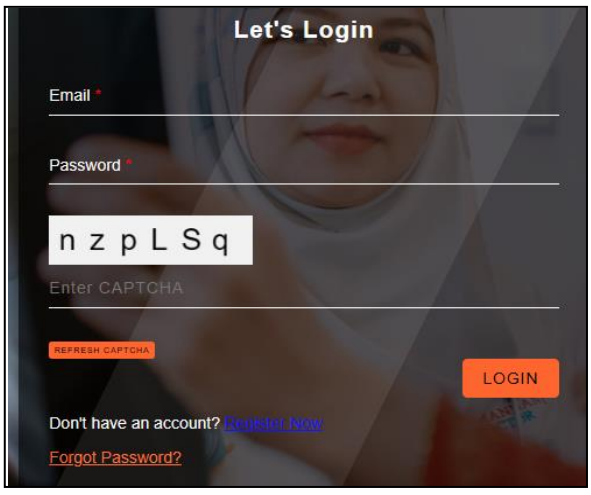

Fig. 33 *The interface for the login module*

5.2.3 Implementation of Book Appointment Module

For the booking appointment, first of all, the users should fetch available doctors for the given date and service along with their available times, and prepares an array to store available doctors. If the number of appointments does not exceed the maximum allowed, the doctor is added to the list of available doctors. Then, the code calculates the next available time slot with a 15-minute gap after the last booking and handles the form submission to insert the appointment data into the database.

book_appointment.php

```
<div class="date-selection">
    <h1>Available Services and Doctors</h1>

    <!-- Form to select date to view available services -->
    <form method="get" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
        <label for="date">Select Date:</label>
        <input type="date" id="date" name="date" required min="<?php echo $current_date; ?>" max="<?php echo $max_date; ?>">
        <button type="submit">View Available Services</button>
    </form>
</div>

<!-- Display available services -->
<div class="service-list">
    <?php if (isset($result_available_services) && $result_available_services->num_rows > 0): ?>
        <h2>Available Services on <?php echo $date; ?>:</h2>
        <ul>
            <?php while ($row_service = $result_available_services->fetch_assoc()): ?>

                <li>
                    <strong><?php echo $row_service["name"]; ?></strong>
                    <p> - <?php echo $row_service["description"]; ?></p> <!-- Display service description -->
                    <ul>
                        <!-- Retrieve available doctors for the service on the selected date -->
                        <?php
                        $service_id = $row_service["service_id"];
                        $sql_available_doctors = "SELECT d.doctor_id, d.username, da.start_time, da.end_time
                                                  FROM doctor_availability da
                                                  INNER JOIN doctors d ON da.doctor_id = d.doctor_id
                                                  WHERE da.service_id = ? AND da.date = ?";
                        $stmt_available_doctors = $conn->prepare($sql_available_doctors);
                        if (!$stmt_available_doctors) {
                            die("Error preparing statement: " . $conn->error);
                        }
                        $stmt_available_doctors->bind_param("is", $service_id, $date);
                        $stmt_available_doctors->execute();
                        $result_available_doctors = $stmt_available_doctors->get_result();
                        ?>
                        <?php while ($row_doctor = $result_available_doctors->fetch_assoc()): ?>
                            <li>
                                Doctor: <?php echo $row_doctor["username"]; ?>
                                ( Time: <?php echo $row_doctor["start_time"]; ?> - <?php echo $row_doctor["end_time"]; ?> )
                                <form method="post" action="booking.php">
                                    <input type="hidden" name="doctor" value="<?php echo $row_doctor["doctor_id"]; ?>">
                                    <input type="hidden" name="service" value="<?php echo $service_id; ?>">
                                    <input type="hidden" name="date" value="<?php echo $date; ?>">
                                    <br><input type="submit" value="Book" class="book-btn">
                                </form>
```

Fig. 34 *The partial code for the book appointment module*

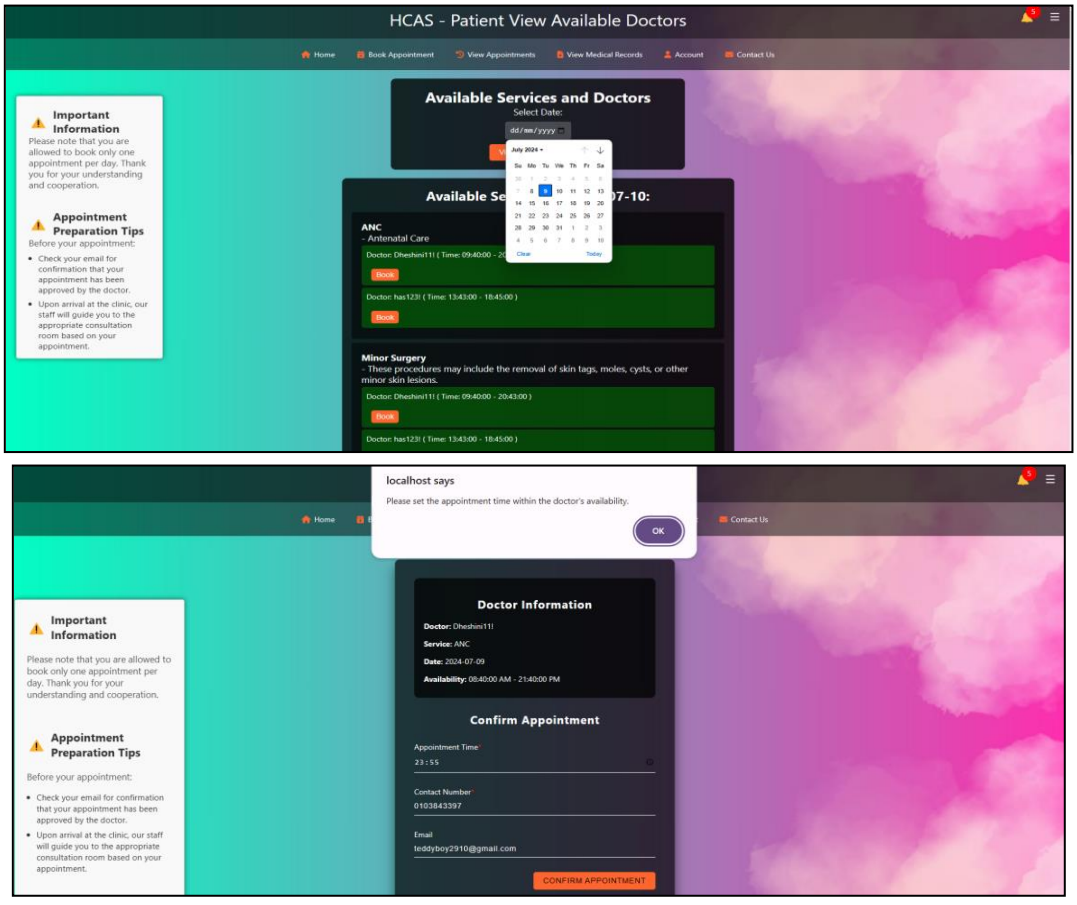

Fig. 35 *The interface for the book appointment module*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 463-485

5.3 Testing

Testing is an integral aspect of the software development life cycle (SDLC) that verifies the program fulfil the user's needs and necessary specifications. The testing process comprises executing the software with the purpose of identifying faults and confirming that the software performs as planned.

5.3.1 Test Plan Result

Table 5 shows that the test plan results indicate the HCAS application successfully passes all tests without any major issues. The application functions as expected, and the user interface is intuitive and easy to use.

Table 5 *The test plan for patients*

Test Plan	Expected	Result
Register	Patient can register into the system with successful email verification	Success
Login	Patient can login into the system with the reCaptcha, and successful unique email OTP generation	Success
Book Appointment	Patients can book appointments with available doctors and services	Success
Appointment History	Patient can view their appointment history and they cancel the appointment as well	Success
Edit Profile	Patient can edit their profile information	Success
View Available Doctors	Patient can view available doctors for a specific service and date	Success
View Available Time Slots	Patient can view available time slots for a specific doctor and service	Success
View Doctor Profile	Patient can view doctor profiles, including their availability and services	Success

Table 6 *The test plan for doctor*

Test Plan	Expected	Result
Register	Doctor can register into the system with successful email verification	Success
Login	Doctor can login into the system with the reCaptcha, and successful unique email OTP generation	Success
Add Availability	Doctor can add their availability for specific dates and times	Success
View Appointments	Doctor can view all appointments of patients under their care	Success
Update Appointment Status	Doctor can update the status of appointments (e.g. pending, approved, cancelled)	Success
Edit Profile	Doctor can edit their profile information	Success

Table 7 *The* test *plan for admin*

Test Plan	Expected	Result
Login	Admin can login into the system with data in the admin table store the username and password.	Success
Add Patient	Admin can add new patients to the system	Success
Add Doctor	Admin can add new doctors to the system	Success
Add Service	Admin can add new services to the system	Success
Add Appointment	Admin can add appointments for patients	Success
View Patient List	Admin can view a list of all patients in the system	Success
View Doctor List	Admin can view a list of all doctors in the system	Success
View Appointment List	Admin can view a list of all appointments in the system	Success

5.3.2 User Acceptance Tests Results

The user acceptance testing (UAT) results show that the HCAS application meets the user's expectations and requirements. The chart represents user satisfaction ratings for various aspects of the Hannani Clinic

Appointment System (HCAS) based on a survey. First question is ease of scheduling, most respondents found scheduling either very easy or moderately easy. Second question, wait time satisfaction are rates varied, with a mix of satisfaction and dissatisfaction.Third question, online portal navigation similar to scheduling, the majority found the online portal easy to navigate. Forth question, website information availability mixed responses, with a trend towards finding the information adequate but room for improvement.Lastly, likelihood to recommend varied opinions, indicating some users would recommend the system while others might not. The majority of users rated the ease of booking an appointment stated in Fig. 36 as "1" (very difficult), indicating significant challenges in navigating the system. A notable number of users in the 18-24 age group particularly found it challenging. Only a few users rated it higher, suggesting that improvements in user interface and experience are necessary to make the booking process smoother. Ratings for the system design state in Fig. 37 were diverse, but many respondents also gave low scores, particularly "1" and "2". This reflects dissatisfaction with the overall design and usability of the appointment system. Some users, however, rated the design as "3" or higher, which shows that there are aspects of the design that are functional and appreciated, but enhancements are needed to ensure a consistently positive experience. Fig. 38 is user ratings for the security of the Hanna Clinic Appointment System varied, with a significant portion rating it as "1" (least secure). This suggests concerns about the system's security measures. Interestingly, among those who used the One-Time Password (D-OTP) feature, many rated its effectiveness highly, indicating that while general security perception is low, specific security features like D-OTP are viewed positively when used.

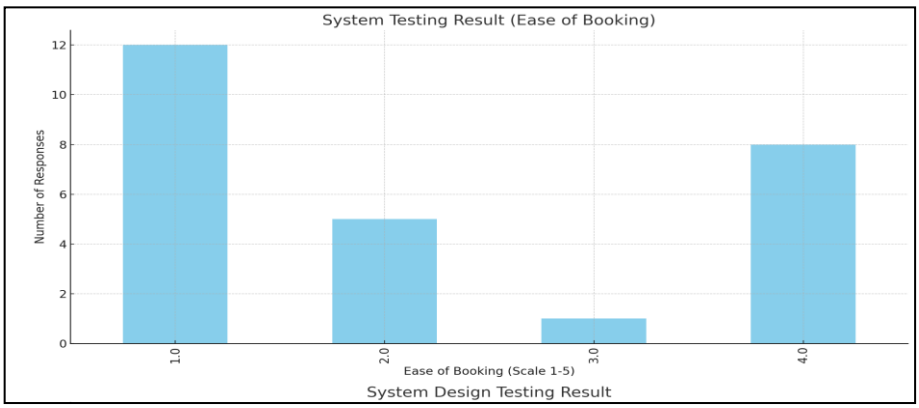

Fig. 36 *System Testing Result (Ease of Booking)*

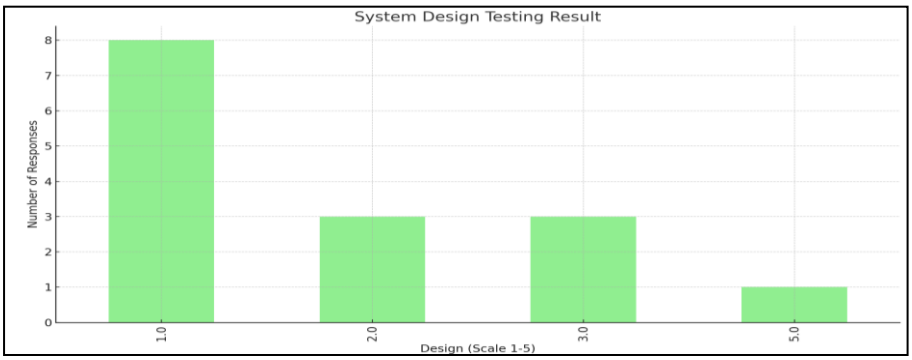

Fig. 37 *System Design Testing Result*

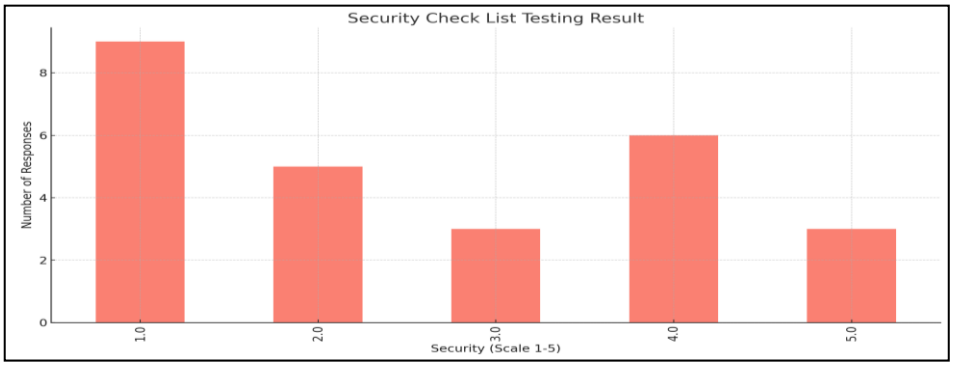

Fig. 38 *Security Check List Testing Result*

6. Conclusion

The user acceptance testing (UAT) results show that the HCAS application meets the user's expectations and requirements. The chart represents user satisfaction ratings for various aspects of the Hannani Clinic Appointment System (HCAS) based on a survey. The first question is ease of scheduling, most respondents found scheduling either very easy or moderately easy. Second question, wait time satisfaction are rates varied, with a mix of satisfaction and dissatisfaction.Third question, online portal navigation similar to scheduling, the majority found the online portal easy to navigate. Fourth question, website information availability mixed responses, with a trend towards finding the information adequate but room for improvement.Lastly, likelihood to recommend varied opinions, indicating some users would recommend the system while others might not. Most users rated the ease of booking an appointment as "1" (very difficult), indicating significant challenges in navigating the system. A notable number of users in the 18-24 age group particularly found it challenging. Only a few users rated it higher, suggesting that improvements in user interface and experience are necessary to make the booking process smoother. Ratings for the system design were diverse, but many respondents also gave low scores, particularly "1" and "2". This reflects dissatisfaction with the overall design and usability of the appointment system. Some users, however, rated the design as "3" or higher, which shows that there are aspects of the design that are functional and appreciated, but enhancements are needed to ensure a consistently positive experience. User ratings for the security of the Hanna Clinic Appointment System varied, with a significant portion rating as "1" (least secure). This suggests concerns about the system's security measures. Interestingly, among those who used the One-Time Password (D-OTP) feature, many rated its effectiveness highly, indicating that while general security perception is low, specific security features like D-OTP are viewed positively when used.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: **study conception and design:** *D. Nagalingam, K. A. Mohamad Sukri*; **data collection:** *D. Nagalingam, K. A. Mohamad Sukri*; **analysis and interpretation of results:** *D. Nagalingam, K. A. Mohamad Sukri*; **draft manuscript preparation** *D. Nagalingam, K. A. Mohamad Sukri*. All authors reviewed the results and approved the final version of the manuscript.

References

[1] F. I. Hassan, M. A., Pavel, M. I., Muhtasim, D. A., Shahi, S. M. K. H., & Rumpa, *Enhanced Security of User Authentication on Doctor E-Appointment System. In Lecture notes on data engineering and communications technologies*. 2022.

[2] W. Stallings, *Computer Security Princuplew and Practices*. 2014.

[3] R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Commun. ACM*, vol. 21, no. 12, pp. 993–999, 1978, doi: 10.1145/359657.359659.

[4] E. Colbert, "The object-oriented software development method: A practical approach to object-oriented development," *Proc. Conf. Tri-Ada 1989 Ada Technol. Context Appl. Dev. Deployment, TRI-Ada 1989*, pp. 400–415, 1989, doi: 10.1145/74261.74291.

[5] S. Roy, "Object-Oriented Analysis and Design," vol. 1, no. 1, pp. 18–24, 2016.

[6] B. Padmanabhan, "Unified Modeling Language (UML) Overview," *Princ. Softw. Eng.*, pp. 1–20, 2012.

[7] D. Bell, "UML basics : An introduction to the Unified Modeling Language A little background," *Ration. Softw.*, pp. 1–11, 2003, [Online]. Available: http://www.therationaledge.com/content/jun_03/f_umlintro_db.jsp

[8] J. Letwoski, "Doing database design with MySQL," *J. Technol. Res.*, vol. 6, no. February, pp. 1–15, 2014, [Online]. Available: https://www.researchgate.net/publication/271910489_Doing_database_design_with_MySQL

[9] ENISA, "ENISA Threat Landscape 2020 Web Application Attacks," *Eur. Union Agency Cybersecurity*, no. 10, p. 18, 2020.

[10] I. Sommerville, *Software Engineering (9th ed.; Boston, Ed.). Massachusetts: Pearson Education.* 2011.