**INTI International College Penang School of Computing**

**3+0 Bachelor of Science (Hons) in Computer Science, in collaboration with Coventry University, UK**

**3+0 Bachelor of Science (Hons) in Computing, in collaboration with Coventry University, UK**

**Coursework cover sheet**

**Section A - To be completed by the student**

|  |  |
| --- | --- |
| Full Name: **HARVIND NAIR A/L SELVAM** | |
| CU Student ID Number: 15095721 | |
| Semester: APR2024 | |
| Lecturer: Mr. Shahriman Mohd Said | |
| Module Code and Title: 5001CEM Software Engineering | |
| Assignment No. / Title: Portfolio | % Of Module Mark: 10% |
| Hand out date: 25 April 2024 | Due date: 25 June 2024 |
| Penalties: No late work will be accepted. If you are unable to submit coursework on time due to extenuating circumstances, you may be eligible for an extension. Please consult the lecturer. | |
| Declaration: I/we the undersigned confirm that I/we have read and agree to abide by the University regulations on plagiarism and cheating and Faculty coursework policies and procedures. I/we confirm that this piece of work is my/our own. I/we consent to appropriate storage of our work for plagiarism checking. | |

**Section B - To be completed by the module leader**

|  |
| --- |
| Learning Outcomes  LO4 -Select, evaluate and use tools and techniques to successfully manage a large scale software  project, including configuration management and version control.  LO5 - Use a range of appropriate tools to contribute to the development of a solution to a real-world  problem.  LO6 - Select, evaluate and apply standards, tools and techniques for assuring software quality. |
| Lecturer’s Feedback |
| Internal Moderator’s Feedback |

Table of Contents

[**1.0** **Introduction** 4](#_Toc139392744)

[**1.1** **Methodology Used for this project.** 4](#_Toc139392745)

[**1.2** **Importance of the system in real world** 5](#_Toc139392746)

[**2.0** **Progress Cycle** 6](#_Toc139392747)

[**CYCLE 1** 6](#_Toc139392748)

[**CYCLE 2** 7](#_Toc139392749)

[**CYCLE 3** 8](#_Toc139392750)

[**Prototype Link:** 8](#_Toc139392751)

[**CYCLE 4** 9](#_Toc139392752)

[**CYCLE 5** 10](#_Toc139392753)

[**ERD Diagram** 11](#_Toc139392754)

[**Database** 12](#_Toc139392755)

[**CYCLE 6** 13](#_Toc139392756)

[**2.1** **Testing Process** 14](#_Toc139392757)

[**HR Home Page** 14](#_Toc139392758)

[**HR List of Staff Page** 14](#_Toc139392759)

[**HR Training Info Page** 15](#_Toc139392760)

[**Staff Training List Page** 15](#_Toc139392761)

[**Staff Enrollment Page** 16](#_Toc139392762)

[**3.0** **GitHub** 17](#_Toc139392763)

[**4.0** **References** 18](#_Toc139392764)

# **Introduction**

The "Call a Doctor" (CaD) system, developed and utilized by Sage Enterprise, is a ground-breaking tool intended to revolutionize the way individuals obtain healthcare. This method closes the gap between conventional healthcare and contemporary convenience by allowing patients to consult doctors from the comfort of their homes. Patients can easily locate and request home visits from clinics that have registered with CaD, which improves the efficiency and accessibility of healthcare services.

# **Methodology Used for this project.**

For this project, the Agile Methodology is indispensable. Agile project management places a strong emphasis on cross-functional cooperation and ongoing development. It guides teams through cycles of planning, implementation, and assessment and divides large projects into smaller, more manageable chunks.

The following are the main guiding principles for project management using the Agile methodology:

1. Evolution is welcomed at any stage of the process.
2. Products or services are delivered more frequently.
3. To achieve project success and optimal outcomes, all stakeholders and team members must remain motivated. Teams have all the necessary resources and support to complete the project.
4. In-person meetings are considered the most effective and efficient means of communication for project success.
5. The ultimate measure of success is a functioning final product.
6. Agile methods allow stakeholders and development teams to work at a steady and sustainable pace, fostering continuous progress.
7. A constant focus on technical excellence and solid planning will enhance agility.
8. Simplicity must be a priority at every stage of the project.
9. Self-organizing teams are more likely to generate the best ideas and projects while meeting the requirements.
10. Teams adjust their behaviour to boost productivity and efficiency.

# **Importance of the system in real world**

The Call a Doctor (CaD) system is extremely important. By providing patients with a centralized platform to request doctor visits at home, it streamlines and organizes the process of accessing medical care. The system automates tedious tasks such as managing appointment requests and verifying clinic registrations. This increases overall efficiency, reduces administrative burdens, and saves both patients and healthcare providers time and money.

The Call a Doctor (CaD) system holds great significance. Accessing medical treatment is made easier and more organized by giving patients a single platform to schedule house visits with doctors. Difficult processes like tracking down appointments and confirming clinic registrations are automated by the technology. This saves time and money for patients as well as healthcare professionals and improves overall efficiency while lowering administrative costs.

Collaboration and communication amongst the many parties involved in the healthcare process are improved by the CaD system. When there are any changes to the appointment schedule or the status of a patient's doctor request, they are promptly and clearly informed. Clinic managers, physicians, and patients can collaborate and communicate efficiently thanks to the system. This guarantees that all parties are in agreement, promotes cooperation, and boosts transparency.

The CaD system also aids in keeping precise records of doctor appointments and medicines. Clinic managers can quickly handle patient requests, monitor physician status, and make sure resources are allocated correctly thanks to technology. To make the most use of the medical resources that are available, the system can also help with scheduling changes or doctor reallocations based on demand. The solution lets administrator’s backup and safely store patient records to guarantee data integrity and accessibility for later use.   
All things considered, maintaining correct data, enhancing communication, and managing healthcare services are all made possible by the Call a Doctor system. In the end, it benefits patients as well as healthcare providers by improving the effectiveness, productivity, and cost-effectiveness of medical care delivery.

# **Progress Cycle**

### **CYCLE 1**

**Meeting 1**

**Date:** 10/04/2024

**Method:** (Face to Face)

**Meeting Description:** Discussion of the intended purpose

**Attendance:** Harvind, Linkesh

A close-up of a paper

Description automatically generatedA spiral notebook with a pen and diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

We discussed adding the feature to the system and use case diagram during this meeting depending on the assignment requirement. We also added kanban board and backlog for this meeting.

### **CYCLE 2**

**Meeting 2**

**Date:** 01/05/2024

**Method:** (Face to face)

**Meeting Description:** Discussion of the system prototype

**Attendance:** Harvind, Linkesh

A close-up of a login page

Description automatically generated

We discussed the color schemes to employ as well as the design of the prototype.

### **CYCLE 3**

**Meeting 3**

**Date:** 18/05/2024

**Method:** (Face to Face)

**Meeting Description:** Discussion and development of a system prototype

**Attendance:** Harvind, Linkesh

For each function, pages were created at the meeting, and each page was assigned to the person in charge of developing the prototypes. I had the task of creating prototypes for the following pages: Main page, Register page, Forget password page, Clinic register page,

Patient register page, patient home page, Book appointment page.

The evidence of some of the prototypes I designed is below.

A screenshot of a login page

Description automatically generated

Diagram 1.0 Main Page

A screenshot of a computer

Description automatically generated

Diagram 1.1 Register Page

A blue form with white text

Description automatically generated

Diagram 1.2 Clinic Register Page

A screenshot of a registration form

Description automatically generated

Diagram 1.3 Patient Register Page

**Prototype Link:** <https://www.figma.com/design/7xrhQRkcFjIAyDGaHRXet9/Sage-mode?node-id=0-1&t=SFWPJs8yHJBS5yu6-0>

### **CYCLE 4**

**Meeting 4**

**Date:** 20/05/2024

**Method:** (Face to Face)

**Meeting Description:** Discussion and development of a system prototype

**Attendance:** Harvind, Linkesh

A few more pages were needed to finish our prototypes. I was tasked with creating prototypes for the following pages: Patient profile page, Patient edit profile page, Patient appointment summary page. We've finished our prototypes using this. Below is evidence of one of my designed prototypes.

A screenshot of a patient profile

Description automatically generated

Diagram 1.4 Patient Register Page

A close-up of a paper

Description automatically generated

Diagram 1.5 Parts of pages that we must develop.

A screenshot of a computer

Description automatically generated

We have also created a kanban board with backlog included before moving on to do the database.

### **CYCLE 5**

**Meeting 5**

**Date:** 01/06/2024

**Method:** (Online)

**Meeting Description:** Database Creation

**Attendance:** Harvind, Linkesh

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

Diagram 1.0 Creating a Database Table

The task of building the last three database table in MySQL was assigned to me. I made the patient, prescription, and users database table to store the data used by the system, The main part of the database was constructed by Linkesh.

### **CYCLE 6**

**Meeting 6**

**Date:** 12/06/2024

**Method:** (Face to face)

**Meeting Description:** ERD Making

**Attendance:** Harvind, Linkesh

A notebook with writing on it

Description automatically generated

My main contribution to the development of the ERD included working with Linkesh, who built the MySQL database. We discussed many ideas at the meeting. To store the data used by the system, a database was constructed, and I made the ERD using the database.

### **ERD Diagram**

A screenshot of a computer

Description automatically generated

### **Database**

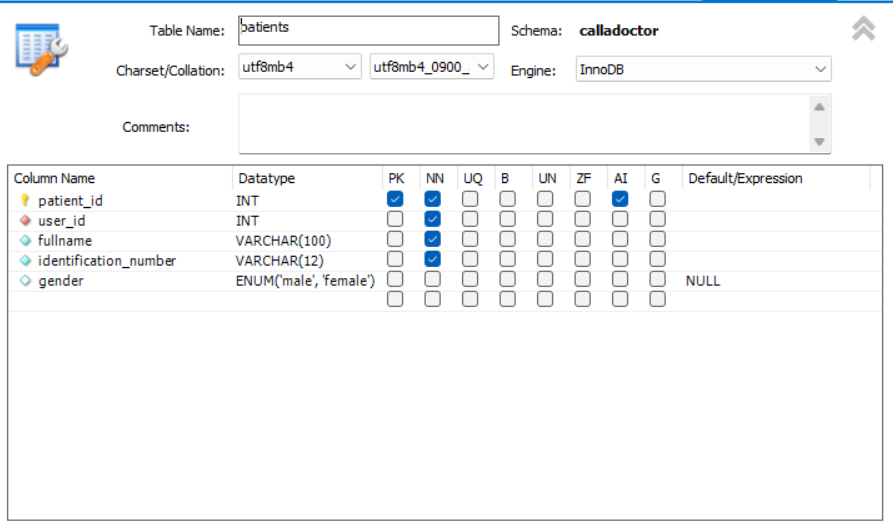


Diagram 1.0 Patient Table

A screenshot of a computer

Description automatically generated

Diagram 1.1 Prescription Table

A screenshot of a computer

Description automatically generated

Diagram 1.2 Users Table

### **CYCLE 7**

**Meeting 7**

**Date:** 18/06/2024

**Method:** (Face to face)

**Meeting Description:** Progression Checking and Problem Solvement

**Attendance:** Harvind, Linkesh

This meeting's goal was to assess the processes and move toward making minor adjustments. After completing most of the tasks and fixing a few small issues, I was now retrieving data from the database. I regularly update GitHub with the corrected code to ensure that my group members are aware of my work.

A person using a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

By this time, most of our features were functioning, and the interface designs were complete. All that was left for the project was testing and documentation.

# **Testing Process**

## **Main Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Status** |
| When the user inputs their respective username and password the respective home page will be opened. (Doctor home page and patient home page) | Respective username and password. | The respective home page will open when the user input their respective username and password. | PASS |
| The forgot password page will be open when the forgot password hyperlink is clicked. | None | The forgot password page opened. | PASS |
| The register page will be open when the “click here” hyperlink is clicked. | None | The register page opened. | PASS |
| A validation will be shown if user didn’t input username or password. | None | A validation message “Please fill out all fields” shown. | PASS |
| A validation will be shown if user didn’t input username or password. | Gfbfg | A validation message “Invalid username or password” is shown. | Pass |

## **Register Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Status** |
| The main page will be open when the “click here” hyperlink is clicked. | None | The main page is opened. | PASS |
| The patient register page will be open when the patient image button is clicked. | None | The patient register page is opened. | PASS |
| The clinic register page will be open when the clinic image button is clicked. | None | The clinic register page is opened. | PASS |

## **Forget Password Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Status** |
| A validation will be shown when user didn’t fill out any fields. | None | A validation message “Please fill out all fields” is shown. | PASS |
| When a new password and confirm password is different it should show a validation message. | Different Passwords | A validation message “Password do not match” is shown. | Pass |
| After giving the correct username, same new password and confirm password is input it should show a validation message and update the new password in the database. | Same Password | A validation message “Password reset successfully” is shown and the new password is updated in the database. | Pass |

## **Clinic Register Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Status** |
| When all field is filled out it should show a validation message and goes back to login page. | All fields | A validation message “Request for clinic registration page has been sent! Wait for approval.” is shown and it updated in database. | PASS |
| When some field or any field is not filled out it should show a validation message. | Some fields. | A validation message “All fields must be field out.” is shown. | PASS |
| When choose file button is clicked it should open file directory. | None | Opened file directory. | PASS |
| When back button is clicked it should open registration page. | None | Registration page is opened when back button is clicked. | PASS |

## **Patient Register Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Status** |
| A validation will be shown when user didn’t fill out any fields. | Any fields | A validation message “Please fill all the fields” is shown. | PASS |
| A validation will be shown when user input different passwords in the field. | Different Password | A validation message “Password do not match” is shown. | Pass |
| A validation will be shown when user fill out all the fields. | All fields | A validation message “Registration successful” is shown. | Pass |

## **Patient Home Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Status** |
| Display the approved of reject appointments and pending appointment for the specific patient that login from the database. | None | A table displayed containing all approved of reject appointments and pending appointment for the specific patient. | PASS |
| When the patient clicks the delete button in the pending appointment table is shows a validation message. | None | A validation message “Do you want to delete this appointment?” is shown. | PASS |
| Navigate to request appointment page when click the book appointment button. | None | Opened request appointment page when click the book appointment button. | PASS |
| Navigate to profile page when click the profile button. | None | Opened profile page when click the profile button. | PASS |
| Navigate to appointment summary page when click the appointment summary button. | None | Opened appointment summary page when click the appointment summary button. | PASS |
| Navigate to login page when click the logout button. | None | Opened login page when click the logout button. | PASS |

## **Book Appointment Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Status** |
| A validation will be shown when user didn’t choose any fields. | None | A validation message “Please select a clinic” is shown. | PASS |
| A validation will be shown when user didn’t fill the doctor field. | None | A validation message “Please select a doctor” is shown. | PASS |
| A validation will be shown when user didn’t fill the reason field. | None | A validation message “Please provide a reason” is shown. | PASS |
| A validation will be shown when user didn’t fill the time field. | None | A validation message “Please select a time for an appointment” is shown. | PASS |
| A validation will be shown when user fill out all the field. | None | A validation message “Appointment request sent successfully” is shown. | PASS |
| When the back button is clicked, it should navigate to patient home page. | None | Opened patient home page when click the back button. | PASS |

## **Patient Profile Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Status** |
| Display the patient’s information that logged in. | None | The system will display the patient’s profile that logged in. | PASS |
| When patient click the “edit profile” button, the system should direct to patient edit profile page. | None | The system will close the patient profile page and open the patient edit profile page. | PASS |
| When patient click the “back” button, the system should direct back to patient home page. | None | The system will close the patient profile page and open back the patient home page. | PASS |

## **Patient Edit Profile Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Status** |
| Display the patient’s information that logged in. Patient should be able to edit their address, phone number and email only. | Patient’s email, phone number and address. | The system will let the patient to edit their address, phone number and email only. | PASS |
| When patient click the “confirm” button, the system should save the current info and go back to the patient profile page. | None | The system will save the patient’s info and open back the patient profile page. | PASS |
| When the patient clicks the “back” button, the system should direct back to patient profile page. | None | The system will close the patient edit profile page and open back the patient profile page. | PASS |
| When patient click the “confirm” button without changing info, the system should still save the changes and go back to patient profile page. | None | The system will still save the changes and close the patient edit profile page. Then, open back the patient profile page. | PASS |

## **Patient Appointment Summary Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Status** |
| Display the past appointments and upcoming appointment for the specific patient that login from the database. | None | A table displayed containing all past appointments and upcoming appointment for the specific patient. | PASS |
| When patient click the “back” button, the system should direct to patient home page. | None | The system will close the patient appointment summary page and open the patient home page. | PASS |

# **Automated Testing**

* + 1. Test for send\_appointment\_request in Patient Request Appointment Page

One essential tool for expediting the process of setting up patient appointments is the **“send\_appointment\_request”** button on the Patient Request Appointment page. A series of actions are started when a patient clicks this button to make sure the appointment request is correctly processed and entered the system. Maintaining an effective and user-friendly appointment booking system is essential for improving patient happiness and the overall operational efficacy of healthcare services.  
  
As soon as the **“send\_appointment\_request”** button is clicked, the patient's input data must be gathered and verified. This contains the desired time, date, and clinic as well as the reason for the appointment. The system makes sure that every field is filled out accurately and tells the patient right away if there is any missing or incorrect information. This step is crucial to preventing inaccurate or missing data from entering the system, which could result in misunderstandings or conflicts with scheduling. The necessity of user instruction and data quality in digital forms is highlighted by the validation process.  
  
The button initiates a function that handles the appointment request processing after the data has been verified. The function starts by changing the chosen date's format from **'mm/dd/yy'** to the **'yyyy-mm-dd'** format that the database requires. To guarantee consistency and compatibility with the database schema, this conversion is essential. The function then uses specified credentials to create a connection to the MySQL database. Using this secure connection, the system may communicate with the database and run SQL queries to effectively handle data.

The creation and execution of a SQL query to enter the appointment details into the database forms the basis of the **“send\_appointment\_request”** function. The initial status of the query is **"pending,"** and it contains the following information: clinic ID, doctor ID, patient ID, appointment date, appointment time, and reason for appointment. This insertion makes sure that the patient's request is available for the clinic administration to review and approve in addition to recording it. The integration of user actions with backend operations is demonstrated in this step, which guarantees that appointment requests are processed and stored in an organized manner.  
  
The system notifies the patient that their request has been received once the appointment data has been successfully entered. An essential part of the user experience is this confirmation, which completes the interaction loop and provides reassurance. To further reinforce a smooth and simple user experience, the interface is made to automatically reroute the patient back to the main page. This reroutes aids in preserving the application's flow and gets the patient ready for more exchanges or to check their scheduled and confirmed appointments.  
  
All things considered, the **“send\_appointment\_request”** button encompasses a vital piece of functionality that combines data processing, user input validation, database interaction, and user feedback. It guarantees that requests for appointments are handled effectively, which improves patient satisfaction and helps the healthcare system run smoothly. This feature's careful planning and execution highlight how crucial it is to combine solid backend procedures with frontend interactions to produce a dependable and efficient digital healthcare solution.

Code

|  |
| --- |
| **def** send\_appointment\_request():      """Sends the appointment request to the database."""      clinic\_name = clinic\_var.get()  *# Get selected clinic name*      clinic\_id = clinics.get(clinic\_name)  *# Get clinic\_id for selected clinic\_name from clinics dictionary*      doctor\_name = doctor\_var.get()  *# Get selected doctor name*      doctor\_id = doctor\_dict.get(doctor\_name)  *# Get doctor\_id for selected doctor\_name from doctor\_dict dictionary*      reason = reason\_entry.get("1.0", tk.END).strip()  *# Get reason for appointment*      date = cal.get\_date()  *# Get selected date*      hour = hour\_var.get()  *# Get selected hour*      minute = minute\_var.get()  *# Get selected minute*      time = **f**"{hour}:{minute}"  *# Format selected time as "hour:minute"*  *# Validation*      if not clinic\_id:          messagebox.showerror("Error", "Please select a clinic.")  *# Show error message if clinic not selected*          return      if not doctor\_id:          messagebox.showerror("Error", "Please select a doctor.")  *# Show error message if doctor not selected*          return      if not reason:          messagebox.showerror("Error", "Please provide a reason for the appointment.")  *# Show error message if reason not provided*          return      if not date:          messagebox.showerror("Error", "Please select a date for the appointment.")  *# Show error message if date not selected*          return      if not hour or not minute:          messagebox.showerror("Error", "Please select a time for the appointment.")  *# Show error message if time not selected*          return  *# Convert date format from 'mm/dd/yy' to 'yyyy-mm-dd'*      date\_obj = datetime.strptime(date, '%m/%d/%y')  *# Parse date string to datetime object*      formatted\_date = date\_obj.strftime('%Y-%m-%d')  *# Format datetime object as string 'yyyy-mm-dd'*      try:          connection = mysql.connector.connect(\*\*db\_config)  *# Connect to the database using db\_config*          cursor = connection.cursor()  *# Create a cursor object using the connection*          cursor.execute("""              INSERT INTO appointments (clinic\_id, doctor\_id, patient\_id, appointment\_date, appointment\_time, reason, appointment\_request\_status)              VALUES (%s, %s, %s, %s, %s, %s, 'pending')          """, (clinic\_id, doctor\_id, patient\_id, formatted\_date, time, reason))  *# Execute SQL query with parameters*          connection.commit()  *# Commit transaction*          cursor.close()  *# Close the cursor*          connection.close()  *# Close the database connection*          messagebox.showinfo("Success", "Appointment request sent successfully!")  *# Show success message*          go\_back\_to\_patient\_home()  *# Go back to patient home after sending the request*      except Error as e:          messagebox.showerror("Error", **f**"Failed to send appointment request: {e}")  *# Show error message on exception* |

* + 1. Test for fetch\_appointments in Patient Appointment Summary Page

A patient's past and future appointments can be retrieved from a database using the **“fetch\_appointments”** function in the appointment summary module. It communicates with the database via a MySQL connection, filtering pertinent records using the patient ID that has been supplied. Using the **mysql.connector.connect** method, the function first creates a connection to the database; the connection details are defined in **“db\_config”.** It generates a cursor object to run SQL queries after it is connected.  
  
The function runs a SQL query that links the appointments, doctors, clinics, and prescriptions columns in order to retrieve previous appointments. For appointments when the treatment status is marked as "done," this query chooses the appointment date, time, reason, doctor's name, clinic name, and medical report (if available). The results are arranged according to the date of the appointments, decreasing.  
  
A second query is run, linking the appointments, doctors, and clinics tables, for future appointments. Similar information (apart from prescriptions) is retrieved by this query for appointments where the appointment request status is **“accepted”**, and the treatment status is **“pending”**.The results are arranged according to the dates of appointments, descending.  
  
The function runs both queries, closes the database connection and cursor, and then produces two lists, one with previous appointments and the other with ones that are still to come. During this process, any database issues that arise are detected, printed, and empty lists are sent back. The ability to provide the patient's appointment history and future schedule in the user interface depends on this feature.

Code

|  |
| --- |
| **def** fetch\_appointments(patient\_id):      try:          connection = mysql.connector.connect(\*\*db\_config)  *# Establishing connection to MySQL database*          cursor = connection.cursor()  *# Creating a cursor object*  *# Fetch past appointments*          cursor.execute("""              SELECT a.appointment\_date, a.appointment\_time, a.reason, d.fullname AS doctor\_name, c.clinic\_name, IFNULL(pr.medical\_report, 'N/A')              FROM appointments a              JOIN doctors d ON a.doctor\_id = d.doctor\_id              JOIN clinics c ON a.clinic\_id = c.clinic\_id              LEFT JOIN prescriptions pr ON a.appointment\_id = pr.appointment\_id              WHERE a.patient\_id = %s AND a.treatment\_status = 'done'              ORDER BY a.appointment\_date DESC          """, (patient\_id,))          past\_appointments = cursor.fetchall()  *# Fetching all past appointments*  *# Fetch upcoming appointments*          cursor.execute("""              SELECT a.appointment\_date, a.appointment\_time, a.reason, d.fullname AS doctor\_name, c.clinic\_name, 'N/A' AS prescriptions              FROM appointments a              JOIN doctors d ON a.doctor\_id = d.doctor\_id              JOIN clinics c ON a.clinic\_id = c.clinic\_id              WHERE a.patient\_id = %s AND a.treatment\_status = 'pending' AND a.appointment\_request\_status = 'accepted'              ORDER BY a.appointment\_date ASC          """, (patient\_id,))          upcoming\_appointments = cursor.fetchall()  *# Fetching all upcoming appointments*          cursor.close()  *# Closing the cursor*          connection.close()  *# Closing the database connection*          return past\_appointments, upcoming\_appointments  *# Returning fetched appointments*      except mysql.connector.Error as e:          print(**f**"The error '{e}' occurred")  *# Printing error message*          return [], []  *# Returning empty lists if error occurs* |

* + 1. Test for confirm\_delete\_appointment in Patient Profile Page

One of the most important features for efficiently handling appointment requests is the delete option located on the patient profile page. When a patient chooses to cancel an appointment, pressing this button starts a sequence of thoughtfully planned actions that guarantee a dependable and easy-to-use process. A confirmation dialog box asks the user to confirm that they really want to delete the appointment once they click the delete button. This confirmation phase is essential because it prevents unintentional deletions and gives the user a chance to change their mind before it's too late. By guaranteeing that only intentional deletions are handled and eliminating accidental data loss, this step improves the user experience.  
  
Upon receiving confirmation from the user, the **“confirm\_delete\_appointment”** method is triggered. To cancel the appointment, certain backend processes must be managed by this function. To make sure the right appointment is targeted, it starts by recording the appointment date of the request to be erased. Upon establishing a secure connection with the database, the function modifies the appointment's status to 'cancelled'. The database's consistency and integrity are guaranteed by the SQL query used to carry out this change. By designating the appointment as cancelled, the system makes sure that all appointments are appropriately reflected in its database, avoiding any inconsistencies that might result from inconsistent data.  
  
The database update is just the beginning of the delete button's capabilities. Upon effectively cancelling the appointment, the user receives instant response from the system via a message box verifying that the cancellation was successful. This feedback loop lets the patient know that their request was handled appropriately, which is crucial for preserving openness and confidence. The UI is also modified to consider the modifications. The cancelled appointment is taken out of the pending requests part of the list of appointment requests, which is refreshed. The patient will have a smooth and simple experience thanks to this real-time update, which guarantees that the user interface is accurate and up to date.

Moreover, the robustness and efficiency of the program are highlighted by the delete button's connection with the entire system architecture. It ensures that everything runs smoothly and effectively by bridging the gap between the user interface and backend activities. The delete button makes sure that the system is dependable and easy to use by managing the removal procedure thoroughly. It gives patients convenience in scheduling their visits and a feeling of control and flexibility over their relationships with healthcare providers.  
  
In summary, the patient profile page's delete button is a thoughtfully crafted feature that is essential to appointment scheduling. Appointment cancellations are handled accurately and efficiently thanks to its confirmation dialog, secure backend processes, quick user feedback, and real-time interface updates. This feature is essential to the patient appointment scheduling process since it not only improves the user experience but also upholds the integrity and dependability of the system.

Code

|  |
| --- |
| **def** confirm\_delete\_appointment(appointment\_id):      """Confirms and deletes the appointment from the database."""      response = messagebox.askyesno("Confirm Deletion", "Are you sure you want to delete this appointment?")  *# Ask for confirmation*      if response:  *# If the user confirms*          try:              connection = mysql.connector.connect(                  host='localhost',                  user='root',                  password='calladoctor1234',                  database='calladoctor'              )  *# Establishing connection to MySQL database*              cursor = connection.cursor()  *# Creating a cursor object*  *# Delete the appointment from the database*              cursor.execute("DELETE FROM appointments WHERE appointment\_id = %s", (appointment\_id,))              connection.commit()  *# Commit the transaction*              cursor.close()  *# Closing the cursor*              connection.close()  *# Closing the database connection*              messagebox.showinfo("Success", "Appointment deleted successfully!")  *# Show success message*  *# Refresh the appointments list or redirect as needed*  *# Example: refresh\_appointments\_list()*          except Error as e:              messagebox.showerror("Error", **f**"Failed to delete appointment: {e}")  *# Show error message on exception* |

* + 1. Test for create\_forgot\_password\_window in Main Page

For users who need to reset their passwords because they forgot their credentials, the main page's "forgot password" button is an essential feature. This button is intended to start a simple and safe procedure that walks the user through changing their password, making sure they can get back into their account without any further hassles.  
  
The **“open\_forgot\_password\_page”** function is initiated by the system when the lost password button is clicked. The user gets switched from the main login screen to the forgotten password window via this function. To make the switch, close the window that is now open during login and open a new one that is meant to be used for password recovery. This division of functions into distinct windows lowers the possibility of confusion and improves the user experience overall by maintaining a clear and concise user interface.  
  
The **“create\_forgot\_password\_window”** function generates the lost password window, which offers customers an easy-to-use interface to input their username and new password information. It has distinct labels for the username, new password, and password confirmation sections to help the user navigate. Two more buttons are included in the window: one to reset the password and another to return to the main login screen. These components guarantee that users have an easy-to-follow path and options to go back to the login page when necessary.  
  
The **“reset\_password”** function has all the essential features of the forgot password button. The system first verifies that all fields are filled in when a user selects the reset password button after completing the required forms. An error message alerting the user to the need to complete all fields is displayed if any are left unfilled. To avoid incomplete submissions and guarantee a smooth procedure, this validation phase is crucial.  
  
The function verifies that the new password and the confirmation password match once the input has been validated. If they don't match, the user is prompted to re-enter their passwords with matching ones and is displayed an error notice explaining the difference. Ensuring sure the user sets the password they intended and preserving data integrity depend on this step.  
  
In the event that the passwords match, the function talks to the database. Using pre-configured credentials, it creates a secure connection to the MySQL database and queries it to confirm that the supplied username indeed exists. The function modifies the user's password in the database with the new password entered if the username is located. The user receives a success message after the update is successful, verifying that the password reset procedure was finished. After that, the user is taken back to the main login page where they can enter their new login information and shut the forgotten password box.  
  
An error message notifying the user that the specified username does not exist is presented if the username cannot be in the database. Users may swiftly recognize and fix their errors with the aid of this instant feedback, making for a more seamless and effective user experience.  
  
The system ensures that the application stays stable and dependable throughout this process by gently handling any possible database failures and informing the user with the relevant error messages. This thorough approach to password reset management not only improves system security but also guarantees that users who need to reset their passwords will have a good and helpful experience. As a result, the forgot password button is essential to preserving user access and pleasure and enhancing the application's overall security and usability.

Code

|  |
| --- |
| def create\_forgot\_password\_window():      global forgot\_password\_root, email\_entry        ctk.set\_appearance\_mode("light")  *# Set the appearance mode*      ctk.set\_default\_color\_theme("blue")  *# Set the default color theme*      forgot\_password\_root = ctk.CTk()  *# Create the main window*      forgot\_password\_root.title("Forgot Password")  *# Set the window title*      forgot\_password\_root.geometry("400x300")  *# Set the window size*      top\_frame = ctk.CTkFrame(forgot\_password\_root, fg\_color="#ADD8E6", width=400, height=300)      top\_frame.pack(fill="both", expand=True)  *# Create and pack the top frame*      email\_label = ctk.CTkLabel(top\_frame, text="Enter your email", font=("Arial", 16), fg\_color="#ADD8E6")      email\_label.place(relx=0.5, rely=0.3, anchor="center")  *# Create and place the email label*      email\_entry = ctk.CTkEntry(top\_frame, font=("Arial", 16), fg\_color="white", text\_color='black', width=300, height=30)      email\_entry.place(relx=0.5, rely=0.4, anchor="center")  *# Create and place the email entry*      reset\_button = ctk.CTkButton(top\_frame, text="Reset Password", font=("Arial", 16), command=reset\_password, fg\_color="#4682B4", hover\_color="#5A9BD4", text\_color="white")      reset\_button.place(relx=0.5, rely=0.6, anchor="center")  *# Create and place the reset button*      back\_button = ctk.CTkButton(top\_frame, text="Back", font=("Arial", 12), command=forgot\_password\_root.destroy, fg\_color="#4682B4", hover\_color="#5A9BD4", text\_color="white")      back\_button.place(relx=0.5, rely=0.8, anchor="center")  *# Create and place the back button*      forgot\_password\_root.mainloop()  *# Start the main loop* |

* + 1. Test for confirm\_changes in Patient Edit Profile Page

The system begins a sequence of actions to validate and save changes made by the user when they alter their profile information, particularly the Address, Email, or Tel fields, and click the **"Confirm"** button. Initially, the user is presented with a confirmation dialog box by the **“confirm\_changes”** function. This prompt asks if they are certain they want to move forward with the modifications. Should the user click **"Yes,"** the function **“save\_changes”** is invoked.  
  
The **“save\_changes”** function checks that the phone number only consists of digits by retrieving the modified values from the editable fields. An error message appears, and the function stops without changing if the phone number is invalid. The function updates the patient's information in the database if the phone number is legitimate.  
  
Based on the patient ID, the **“update\_patient\_details”** function in the database runs a SQL UPDATE statement to change the user's address, email address, and phone number. This entails using the mysql.connector.connect method to connect to the MySQL database, making a cursor object, then running the SQL statement. The connection is disconnected once the database change has been completed successfully.  
  
The user is notified that their profile has been changed with a success message that appears after the database update has been completed successfully. After that, the current window is closed, and the patient profile window is reopened with the new information displayed. The user's modifications are safely preserved, and the profile interface appropriately shows the most recent data, thanks to the complete procedure.

* + 1. Code

|  |
| --- |
| **def** confirm\_changes():      """Function to confirm changes made to the profile."""      response = messagebox.askyesno("Confirm Changes", "Are you sure you want to make the changes?")  *# Confirm dialog*      if response:          save\_changes()  *# Save changes if confirmed*  **def** save\_changes():      """Function to save changes made to the profile."""      address = entries[4].get()      email = entries[6].get()      phone\_number = entries[7].get()      if not validate\_phone\_number(phone\_number):  *# Validate phone number*          return      update\_patient\_details(patient\_id, address, email, phone\_number)  *# Update patient details*      messagebox.showinfo("Success", "Profile updated successfully")  *# Show success message*      root.destroy()  *# Destroy current window*      patientprofile.create\_patient\_profile\_window(patient\_id, patient\_fullname)  *# Create patient profile window* |

# **GitHub**

Contribution:

A graph with numbers and a line

Description automatically generated

# **References**

*What is Agile Methodology in project management?*. Versatile & Robust Project Management Software. (n.d.). <https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/>

Staff, A. T. (2021, November 16). *Agile methodology: Advantages and disadvantages of an innovative method*. Twproject. <https://twproject.com/blog/agile-methodology-advantages-disadvantages-innovative-method/>

*What is software project management?*. Versatile & Robust Project Management Software. (n.d.-b). <https://www.wrike.com/project-management-guide/faq/what-is-software-project-management/>

*Agile model (software engineering) - javatpoint*. www.javatpoint.com. (n.d.). <https://www.javatpoint.com/software-engineering-agile-model>

*A guide to the Entity Relationship Diagram(ERD) - databasestar*. www.databasestar.com.. <https://www.databasestar.com/entity-relationship-diagram/>

*What is an appointment scheduling system? - Qmatic*. www.qmatic.com.. [https://www.qmatic.com/blog/appointment-scheduling-solution-what-how-and-why#:~:text=An%20appointment%20scheduling%20system%2C%20also,Mobile%20app%20booking](https://www.qmatic.com/blog/appointment-scheduling-solution-what-how-and-why%23:~:text=An%20appointment%20scheduling%20system%2C%20also,Mobile%20app%20booking)

*What is the purpose of an appointment system? - Engageware*. www.engageware.com.. [https://engageware.com/blog/purpose-of-appointment-system/#:~:text=At%20their%20core%2C%20appointment%20scheduling,a%20robust%20customer%20engagement%20strategy.](https://engageware.com/blog/purpose-of-appointment-system/%23:~:text=At%20their%20core%2C%20appointment%20scheduling,a%20robust%20customer%20engagement%20strategy.)