NYU Abu Dhabi
CS-UH 2012: Software Engineering

Assignment 4: Construction and Testing

by

Sohaila Mohammed (sm10688)

Faizan Raza (mr5985)

Khadija Khalid (kk4597)

Ajla Šačić (as15471)

Prepared for

Professor Mohamad Kassab

Instructor Dena Ahmed

Section 1: Introduction

1.1 Test Project Name:

The name of the test project is MedSync Use Case Testing.

1.2 Summary of the Rest of the Test Plan:

This test plan for MedSync is structured to ensure thorough examination and validation of all functionalities encompassed by the system's use cases. Following this introduction, the report will detail each major component as follows:

- Section 2 - Feature Description: This section provides detailed descriptions of the final use cases that are being tested. It includes all user interactions and system responses tied to booking appointments, issuing prescriptions, uploading medical records, requesting diagnostic support, and initiating video calls.
- Section 3 - Assumptions: This section lists the prerequisites for the testing phase, including any test case exclusions and the testing tools employed, which will set the groundwork for the testing procedures.
- Section 4 - Test Approach: This section outlines the strategies we employed in testing, including justifications for the chosen approaches and the tools used for testing each component of the system.
- Section 5 - Test Cases: This section lists all test cases for each use case, categorized as feature or security related. A traceability matrix will also be included to demonstrate the correlation between use cases and the respective test cases.
- Section 6 - Testing Results: This section summarizes the outcomes of the testing phase, highlighting the successes, failures, and unresolved issues encountered during the test execution.
- Section 7 - Recommendations on Software Quality: Based on the test results, this section will offer actionable recommendations to enhance the overall quality and performance of the MedSync system.

The document concludes with appendices that provide supplementary information to support the comprehensiveness of the testing report.

Section 2: Feature Description

1. Use case title: Book an Appointment

Main success scenario:

1. The patient navigates to the appointments section of their profile.
2. The system presents to the patient the booked appointments (if any) and provides the option to schedule a new appointment.
3. The patient selects the option to schedule a new appointment.
4. The system presents the patient with an interface to provide the referral code of the doctor to book an appointment with, which the patient fills and then selects the retrieved doctor accordingly.
5. The system presents to the patient the selected doctor's available appointment slots. From which the patient selects an available time slot.
6. The system prompts the patient to choose the preferred mode of consultation (physical meeting or video call) and to provide details of why they are scheduling an appointment, and the patient provides both accordingly.
7. The patient confirms the appointment details, and the system confirms the appointment booking success.

Extensions:

4a. The patient does not have a referral code for a doctor on the platform.

1. The system will not display any doctor options on the doctor selection page, essentially disallowing the patient from proceeding.

5a. The doctor has no available time slots.

1. The system will not display any time slots for the selected doctor, essentially disallowing the patient from proceeding.

5b. The user decides he/she does not want to complete the booking anymore.

1. The user can navigate all the way back to the main page without interruptions.
2. The system disregards the user's actions during the appointment booking and does not confirm the appointment.

6a. The patient does not choose a mode of consultation.

1. The system highlights to the user that they have to choose a mode of consultation before proceeding.
2. Use case continues at step 6.

6b. The patient does not provide the appointment reason.

1. The system highlights to the patient that the appointment cannot be booked without providing the needed details.
2. Use case continues at step 6.

2. Use case name: Issue Medical Prescription

Main Success Scenario:

1. The doctor navigates to the medical prescription issuing section of their profile.
2. The system presents to the doctor the patients with which he/she has booked appointments.
3. The doctor selects the patient they want to issue a prescription for.
4. The system redirects the doctor to a page where they fill out the prescription details for the patient.
5. The doctor fills in the necessary details.
6. The doctor confirms the prescription.
7. The system confirms successful upload of the prescription to the patient's profile.

Extensions:

2a. The doctor does not have any booked appointments with patients on the platform.

1. The system will not display any patient options on the medical prescription issuing page.

5a. The doctor does not provide all the prescription details.

1. The system displays an error message, highlighting the missing details and prompting the doctor to provide them.
2. Use case continues at step 5

3. Use case title: Upload a Medical Record

Main Success Scenario:

1. The doctor navigates to the medical record uploading section of their profile.
2. The system presents to the doctor the patients with which he/she has booked appointments along with the option to upload a medical record for each.
3. The doctor chooses to upload a medical record for the intended patient, provides the record, and confirms their uploading.
4. The system confirms successful upload of the medical record to the patient's profile.

Extensions:

2a. The doctor does not have any booked appointments with patients on the platform.

2. The system will not display any patient options on the medical prescription issuing page.

3a. The doctor attempts to confirm the uploading without having uploaded a file.

1. The system displays an error message, informing the doctor that they need to provide a file to upload a medical record for the patient.
2. Use case continues at step 3.

3b. The doctor attempts to upload a file format that is not supported.

1. The system informs the doctor of the error and prompts them to upload the medical record in one of the website's accepted file formats.
2. Use case continues at step 3.

3c. The doctor uploaded a file that is too large.

1. The system informs the doctor of the error and prompts them to upload a file within the size limit.
2. Use case continues at step 3.

4. Use case title: Request Intelligent Diagnostic Support

Main success scenario:

1. The doctor navigates to the symptom checker feature available on their profile.
2. The system redirects the doctor to the interaction interface and prompts the doctor to enter the relevant symptoms to diagnose, and the relevant gender and date of birth.
3. The doctor enters the required details.
4. The system processes the input through the ApiMedic API.
5. ApiMedic returns to the system the potential diagnoses based on the inputted information.
6. The system displays the ApiMedic diagnosis results for the doctor to review and consider based on his/her own professional judgment.

Extensions:

3a. The doctor does not provide all the required details.

1. The system highlights to the doctor the missing details that must be provided to receive a diagnosis.
2. The system re-prompts the doctor to enter the relevant details.

3b. The doctor provides more symptoms than the system accepts.

1. The system informs the doctor that they have inputted too many symptoms.
2. The system re-prompts the doctor to enter the symptoms to diagnose.

5. Use case title: Initiate Video Call Appointment

Main Success Scenario:

1. The patient navigates to the appointments section of their profile.
2. The system presents to the patient his/her booked appointments, and for the appointments booked for virtual consultation, provides the option to join a video call.
3. The patient chooses to join the intended virtual appointment over a video call.
4. The system initiates the video call session with the selected doctor through ZegoCloud and presents the patient the option to enter the video call.
5. The patient enters the video call, and the consultation is conducted virtually.

Extensions:

2a. The patient does not have any scheduled appointments on their profile.

1. The system will not display any appointment options on the patient's page.

2b. The patient does not have any appointments scheduled for virtual consultation on their profile.

1. The system will not display the option to join a video call for the non-virtual appointments.

3a. The patient attempts to join a video call not scheduled for the current date.

1. The system will display an inactivated option for joining the video call, disabling the patient from joining a video call not scheduled for the current date.

4a. The patient decides not to enter the video call.

1. The system allows the patient to navigate back to their profile dashboard.

5a. The doctor is not available at the video call appointment.

1. Through ZegoCloud, the patient can wait for the doctor or exit the video call.

Section 3: Assumptions

3.1 Test Case Exclusions:

In the development of our testing strategy for the MedSync platform, certain test cases and scenarios have been deliberately excluded to focus resources and attention on the most critical aspects of the system. The exclusions are as follows:

- Performance Testing: Given the scale and scope of MedSync, which mainly supports five key features, extensive performance testing such as load and stress testing will not be included. The system's current scale and usage do not justify the in-depth recording of response times and other performance metrics, which are typically critical in larger, more robust systems.
- Usability Testing: Due to the absence of a broad user base and the constraints of time, comprehensive usability testing involving user feedback on the platform's ease of use is not feasible. Practical testing of usability would require interactions with an end-user group to gather meaningful insights, which are not available within the current project constraints.
- Security Testing: Although some of our test cases may focus on common vulnerabilities relevant to the core features of the system, more in-depth security testing such as penetration testing or full security audits are not included at this stage due to the limited scope of the project and the early stage of the platform's deployment.

3.2 Test Tool:

The majority of our testing has been conducted manually, which naturally limits our reliance on sophisticated testing tools. For specific techniques like pairwise testing, we have utilized a simple, online-based tool known as pairwiseTool[1] to help design and organize test cases efficiently.

[1] https://pairwise.teremokgames.com/

Section 4: Test Approach

4.1 Test Strategy:

The primary testing techniques employed in evaluating the different system components of MedSync are black box testing and exploratory testing.

- Black Box Testing: Using this method, we treat the software as a "black box," focusing solely on the inputs and outputs without regard to the internal workings of the system. Our goal is to validate whether the system behaves as expected under various conditions, effectively covering all possible scenarios that an end-user might encounter. This approach is particularly beneficial for us to test user interfaces and user interaction flows, ensuring that the system meets the specified requirements as stated in the documentation (Gao, 2003).
- Exploratory Testing: Exploratory testing is particularly useful as it is highly adaptive and enables us (as the testers) to leverage our creativity and insights gained during the testing process itself. We can dynamically devise and execute tests to explore the software's functionalities, learning about the system's behavior and simultaneously designing and running relevant tests. It also allows us testers to quickly identify and respond to issues and anomalies as they arise (Bach, 2003).

These techniques complement each other, with black box testing providing a structured approach to requirement validation and exploratory testing offering a more flexible and reactive strategy to uncover subtle, unforeseen issues.

4.2 Rationale

Our website uses React for the frontend and Express.js for the backend, which are both frameworks used widely across the industry.

- Limitations on White Box Testing: Although tools like Mocha, Chai, and Jest are available for unit testing in Express.js, and Enzyme can be used for React component testing alongside, the time constraints of this project restricted our ability to develop comprehensive white box testing strategies. White box testing requires a detailed examination of internal code and logic, which is time-intensive and was not feasible within our project timeline. These tools, while powerful, demand an extensive setup with the development environment, but the time frame

within which this project lies did not allow for sophisticated testing that would involve utilizing these tools.

- Focus on Black Box Testing: Given the project constraints, we prioritized black box testing. This approach does not rely on the internal workings of the application, allowing us to focus on testing the software from the user's perspective. Black box testing is quicker to implement and effectively addresses the broader functional requirements of the application, ensuring that all major features perform as expected.

- Exploratory Testing: We also incorporated exploratory testing, albeit to a lesser extent than black box testing. This method is highly effective in uncovering unexpected behaviors and use cases by simulating real-user interactions. It also helps in understanding the general flow of the program and the ways in which a user might be led towards a successful and intuitive completion of wanted tasks. Due to the aforementioned reasoning, the testing was done manually in order to simply test the overall features of a system made in a short time frame.

- Tool Utilization: The only external tool we utilized was pairwiseTool as mentioned previously. It was primarily used to ease the process of test case formation but also because it is a tool which helps decrease the amount of unnecessary tests made when different techniques are used (such as permutations, for example).

In summary, the decision to focus on black box and exploratory testing was driven by the need to balance thorough testing with the practical limitations imposed by project timelines and resource availability. This approach ensured that we could deliver a functionally reliable system without the complexities and time requirements of in-depth white box testing.

Section 5: Test Cases

5.1 Test Cases: List all test cases. It is preferable if you group your test cases following any categorization scheme you may choose (E.g. test cases related to features, security, performance, etc.)

Please note that we are categorizing the test cases based on the system functionalities or use cases they concern.

We are also adding test cases that do not concern valuable use cases such as Login and Register due to the necessity of both registration and login for further app functionalities and use cases.

1. Book an Appointment

Test Case 1

Test Case Title: Adding a Time Slot in the Past

Use Case Tested by the Test Case: Book an Appointment (Doctor Prerequisite)

Technique Used: Equivalence Partitioning. We use this technique to divide the input into classes based on the expected response of the system. Here, in the context of a doctor adding a timeslot to be available for appointment booking, the classes are as follows:

Equivalence Partitioning for Time Slot Test Case	
Invalid	**Valid**
Time before today's current time	Time on or after today's current time

Table 1: Equivalence Partitioning for Time Slot Test Case Table

Preconditions:

1. The tester is logged into a test doctor account.
2. The system is on and available (not in downtime).

Input:

- Date: inputted through a calendar interface in the form yyyy/mm/dd
- Time: inputted through a drop down menu in the form of hh:mm

Steps to Execute:

1. Navigate to the Manage Slots section of the doctor profile.
2. Set the date to be the current date.
3. Select the time to be a time before the current time.
4. Attempt to add a new slot.

Expected Results: The system should display an alert for invalid time slot.

Test Case 2

Test Case Title: Verify Mode of Consultation Selection for Appointment Booking

Use Case Tested by the Test Case: Book an Appointment

Technique Used to Generate Test Case: Equivalence Class Partitioning. We use this technique to divide the input into classes based on the expected response of the system. Here, in the context of choosing a mode of consultation when booking an appointment, the classes are "One Mode Selected" and "No Mode Selected," determined by whether the user has selected a preferred mode of consultation to proceed with the booking.

Preconditions:

- The tester is logged into a test patient account.
- The patient account is connected to at least one doctor account with available appointment slots on the platform.

Input:

- For class 1 (One Mode Selected): a selection from a dropdown for the chosen mode of consultation.
- For class 2 (No Mode Selected): no selection from the mode of consultation dropdown.

Steps to Execute:

1. Navigate to the appointments section of the patient profile.
2. Select schedule new appointment and choose a doctor that has available appointment slots from the list.
3. Choose a time slot from the available appointment slots.

- For Class 1:

    4. Choose a mode of consultation for the appointment from the respective dropdown (in-person or video consultation).

- For Class 2:

    4. Leave the mode of consultation dropdown empty.

5. Enter details for the reason for the appointment booking in the respective field.
6. Attempt to confirm the booking.

Expected Results:

- For Class 1 (One Mode Selected): The system should successfully confirm the appointment, updating the doctor’s schedule and the patient’s schedule accordingly.
- For Class 2 (No Mode Selected): The system should not allow the booking to proceed. A message should be displayed highlighting that the mode of consultation has not been selected.

Test Case 3

Test Case Title: Verify Reason for Appointment Input in Appointment Booking

Use Case Tested by the Test Case: Book an Appointment

Technique Used to Generate Test Case: Equivalence Class Partitioning. I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of providing a reason for the appointment when booking it, the classes are "Reason Provided" and "No Reason Provided," determined by whether the user has inputted a reason for the appointment to proceed with the booking.

Preconditions:

- The tester is logged into a test patient account.
- The patient account is connected to at least one doctor account with available appointment slots on the platform.

Input:

- For class 1 (Reason Provided): a string (assumed to explain the reason for booking the appointment).
- For class 2 (No Reason Provided): no input in the respective field.

Steps to Execute:

1. Navigate to the appointment booking section of the patient profile.
2. Select a doctor that has available appointment slots from the list.
3. Choose a time slot from the available appointment slots and click “continue”.
4. Choose a mode of consultation for the appointment from the respective dropdown (inperson or video consultation).

- For Class 1:
    5. Type in a statement in the respective field for reason for appointment.
- For Class 2:
    5. Leave the reason for appointment field empty.

6. Attempt to confirm the booking.

Expected Results:

- For Class 1 (Reason Provided): The system should successfully confirm the appointment, updating the doctor’s schedule and the patient’s schedule accordingly.

For Class 2 (No Reason Provided): The system should not allow the booking to proceed. A message should be displayed highlighting that the reason for appointment field has not been filled.

Test Case 4

Test Case Title: Truth Table Testing for Appointment Booking

Use Case Tested by the Test Case: Book an Appointment

Technique Used: Truth Table Testing.

	Rules			
Conditions	**1**	**2**	**3**	**4**
Consultation method selected	T	F	T	F
Booking Appointment Reason filled in	T	T	F	F
System confirms successful appointment booking	✓			
System displays an error message, notifying the user to select a consultation method.		✓		✓
System displays an error message, notifying the user to enter a reason for the booking			✓	

Table 2: Truth Table for Testing Appointment Booking.

Preconditions

1. The tester is logged into a test patient account.
2. The system is on and available (Not in downtime).
3. The patient is connected to a doctor.
4. There is an available time slot for the doctor on the chosen day.

Inputs: Truth table

1. Test 1:
    a. Preferred method of Consultation: Video Conferencing or In-person
    b. Reason for Booking Appointment: Any reason
2. Test 2:

    a. Preferred method of Consultation: *empty*
    b. Reason for Booking Appointment: Any reason
3. Test 3:
    a. Preferred method of Consultation: Video Conferencing or In-person
    b. Reason for Booking Appointment: *empty*
4. Test 4:
    a. Preferred method of Consultation: *empty*
    b. Reason for Booking Appointment: *empty*

Steps to Execute:

Steps to Execute 1:

1. Navigate to the Appointments section of the patient profile.
2. Select to schedule a new appointment.
3. Select a doctor with an available time slot.
4. Select one of the doctor's available appointment slots.
5. Select Video Consultation or In-person as preferred method of Consultation.
6. Input any reason as Reason for Booking Appointment.

Steps to Execute 2:

1. Navigate to the Appointments section of the patient profile.
2. Select to schedule a new appointment.
3. Select a doctor with an available time slot.
4. Select one of the doctor's available appointment slots.
5. Leave the Preferred Method of Consultation empty.
6. Input any reason as Reason for Booking Appointment.

Steps to Execute 3:

1. Navigate to the Appointments section of the patient profile.
2. Select to schedule a new appointment.
3. Select a doctor with an available time slot.
4. Select one of the doctor's available appointment slots.
5. Select Video Consultation or In-person as preferred method of Consultation.
6. Leave Reason for Booking Appointment empty.

Steps to Execute 4:

1. Navigate to the Appointments section of the patient profile.
2. Select to schedule a new appointment.
3. Select a doctor with an available time slot.
4. Select one of the doctor's available appointment slots.
5. Leave the Preferred Method of Consultation empty.
6. Leave Reason for Booking Appointment empty.

Expected Results:

Test 1: Successful Booking Confirmation Displayed by the system

Test 2: System displays error message notifying the user to select a valid method of consultation.

Test 3: System displays error message notifying the user to input a reason for booking.

Test 4: System displays error message notifying the user to select a valid method of consultation.

Test Case 5

Test Case Title: Confirm Real-Time Update of Doctor Availability

Use Case Tested by the Test Case: Book an Appointment

Technique Used: Exploratory Testing.

This test involves real-time interaction with the system's dynamic content—doctor availability schedules. Exploratory testing is chosen to verify the system's responsiveness and accuracy in reflecting changes made to availability data immediately.

Preconditions

1. The tester has access to a test doctor account and a test patient account connected to that doctor.
2. The doctor account has listed at least one available time slot
3. There is an established method for doctors to update their availability (i.e., deleting a time slot).

Inputs: Doctor action: Deletion of a previously available time slot.

Steps to Execute:

1. Log in to the test doctor account.
2. Navigate to the Manage Slots section of the doctor's profile.
3. Identify a time slot that is currently listed as available.
4. Delete the identified time slot.
5. Immediately after updating, log in as a patient.
6. Navigate to the appointments section and try to book an appointment with this doctor, specifically looking for the recently updated time slot.
7. Observe whether the deleted or updated time slot is still listed as available.

Expected Results: The system should not display the deleted time slot to the patient attempting to book an appointment.

Test Case 6

Test Case Title: Add Doctor with Referral Code

Use Case Tested by the Test Case: Book an Appointment

Technique Used: Equivalence Partitioning

The input space is partitioned into valid and invalid referral codes. Testing with a valid referral code checks if the system correctly adds the doctor associated with the code, confirming functionality for correct inputs.

Preconditions:

1. The tester is logged into a test patient account.
2. For valid referral codes, the patient must have acquired a referral code for a doctor on the platform.

Input:

1. Correct Referral Code (e.g., "DR1234")
2. Incorrect Referral Code (e.g., "INVALID")

Steps to Execute:

1. Navigate to the appointments section of the patient profile.
2. Select schedule new appointment.
3. In the "Enter Referral Code" input field in the schedule appointment page, enter a correct referral code into and submit.
4. Observe and note the system's response.
5. Clear the referral code input field.
6. Enter an incorrect referral code into the input field and submit.
7. Observe and note the system's response.

Expected Results:

1. For the correct referral code:
    - The system successfully adds the doctor to the patient's list of available doctors.
2. For the incorrect referral code:
    - The system does not add any doctor to the list.
    - An error message is displayed, asking the user to enter a valid referral code.

Test Case 7

Test Case Title: Check Appointment Booking Initiation System Display

Use Case Tested by the Test Case: Book an Appointment

Technique Used to Generate Test Case: Decision Table Testing.

We use this technique to determine how different combinations of conditions will affect the start of an appointment booking, associating specific sets of conditions with specific actions. Below is the decision table created:

	Rules		
Conditions	1	2	3
Patient has at least one connected doctor	T	T	F
Selected doctor has at least one available appointment slot	T	F	N/A
Actions			
Display connected doctor(s) account(s)	✓	✓	
Display the option for the patient to add doctors via referral codes	✓	✓	✓
Display selected doctor's available appointment slots	✓		
Display unavailability message for the time slots of the selected doctor		✓	

So we essentially have 3 scenarios we're testing:

1. Scenario 1:
   - The patient attempts to start an appointment booking with a connected doctor that has available appointment slots.
2. Scenario 2:
   - The patient attempts to start an appointment booking with a connected doctor that does not have available appointment slots.
3. Scenario 3:
   - The patient attempts to start an appointment booking when they are not connected to any doctors.

And the system is expected to behave differently according to each.

Preconditions:

- The tester has access to two types of test patient accounts, one is connected to test doctor accounts (some who have available appointment slots and some who don't) and the other is not connected to any doctor accounts.

Input: an initiation of an appointment booking request (button click) and (if applicable) the selection of a connected doctor.

Steps to Execute:

Scenario 1:

1. Log into the system with a test patient account that is connected to doctors.
2. Select a connected doctor with available slots.

Scenario 2:

1. Log into the system with a test patient account that is connected to doctors.
2. Select a connected doctor without available slots.

Scenario 3:

1. Log into the system with a test patient account that is not connected to doctors.
2. Navigate to the appointments booking section of the patient's profile.

Expected Results:

1. **Scenario 1:** The system displays the connected doctors' accounts (and the option to add other doctors via referral codes) then the available appointment slots of the selected doctor, allowing the patient to proceed with booking.
2. **Scenario 2:** The system displays the connected doctors' accounts (and the option to add other doctors via referral codes) then a message indicating that there are no available time slots for each day from the selected doctor's time slots.
3. **Scenario 3:** The system only displays the option to add other doctors via referral codes.

Test Case 8

Test Case Title: Validate Blocking of Booking Appointments in Past Time Slots

Use Case Tested by the Test Case: Schedule Appointment

Technique Used: Truth Table

Proof: This approach helps to determine the logical outcomes based on different conditions of time slot selection (past, present, and future).

- **Current Time Before Slot** (T if the current time is before the slot time, F otherwise)
- **Current Time After Slot** (T if the current time is after the slot time, F otherwise)

The outcomes will be simplified into:

- **Allow Booking** (True if allowed, False if blocked)

Here's how the Truth Table could look using boolean values (True/False):

Current Time Before Slot	Current Time After Slot	Allow Booking
T	**F**	**T**
F	**T**	**F**
F	**F**	**T**

Preconditions:

1. The tester is logged into a test patient account that is connected to at least one doctor account.
2. The system presents available time slots for doctors connected to the patient's account.

Input: Time Slot Selection: Selecting a slot that has already passed.

Steps to Execute:

1. Log into the patient's account and navigate to the booking section.
2. Select a doctor and view the list of upcoming and past appointments (the past appointments are typically not shown but are included here for testing the system's rejection capabilities).

3. Attempt to select a time slot that has a timestamp earlier than the current system time.
4. Confirm and attempt to book the appointment.

5. Observe and record the system's response to the attempt.

Expected Results:

The system should automatically filter out past time slots from the selectable options, but if a past slot is somehow selected (for testing purposes):

- The system should not allow the booking to proceed.
- An error message should be displayed to the user stating that selecting past time slots is not permitted.
- The booking process should be terminated or redirected back to valid slot selections.

Test Case 9

Test Case Title: Ensure Appointment Slot Unavailability Post Booking

Use Case Tested by the Test Case: Book an Appointment

Technique Used: Concurrent User Testing

This technique tests the system's ability to handle and display real-time data changes (slot availability) across different user sessions. It ensures that once a slot is booked by one user, it immediately reflects as unavailable in another user's session who might be booking concurrently.

Preconditions

- The system must have at least two patients registered and able to book appointments.
- A doctor has a set of available appointment slots displayed to patients.

Input:

- Patient A books an available slot.
- Patient B attempts to book an appointment simultaneously or immediately after Patient A.

Steps to Execute:

- **Patient A Books a Slot**:
  - Patient A logs into their account.
  - Navigates to the doctor's appointment booking page.
  - Selects an available time slot.
  - Completes the booking process and receives confirmation that the slot is booked.
- **Patient B Attempts to Book the Same Slot**:
  - Patient B logs into their account simultaneously or immediately after Patient A's booking.
  - Navigates to the same doctor's appointment booking page.
  - Looks for the same slot that Patient A booked.
- **Verify Slot Availability for Patient B**:
  - Check if the slot booked by Patient A is visible and selectable to Patient B.
  - Attempt to select the same slot if it appears.

    - Observe any notifications or error messages regarding slot availability.

Expected Results:

- Patient A should successfully book the selected slot without issues.
- For Patient B, the slot booked by Patient A should either not appear in the list of available slots or should be marked clearly as unavailable.
- If Patient B tries to select the same slot, the system should display a message that the slot is no longer available and prevent further booking actions.

2. Issue Medical Prescription

Test Case 1

Test Case Title: Verify Medicine Name Provision in Prescription

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used to Generate Test Case: Equivalence Class Partitioning.

I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of providing the medicine name when issuing a prescription, the classes are "Medicine Name Provided" and "No Medicine Name Provided," determined by whether the doctor has provided a medicine name when filling out a prescription.

Preconditions:

1. The tester is logged into a test doctor account.
2. The doctor account has at least one appointment with a patient account (the doctor can only issue prescriptions to the patients he/she has appointments with).

Input:

1. Class 1 (Medicine Name Provided): a string for the medicine name is filled in the respective field.
2. Class 2 (No Medicine Name Provided): no input in the respective field.

Steps to Execute:

1. Navigate to the prescription issuing section of the doctor's profile.
2. Select a patient to issue a prescription for.

- For Class 1:

    3. Type in an input in the respective field for medicine name.

- For Class 2:

    3. Leave the medicine name field empty.

4. In the dosage section, input a positive numerical value indicating the amount of the medication and select the unit of measurement from the accompanying dropdown menu (mg, ml, pills, droplets).
5. In the frequency section, input a positive numerical value indicating how often the medication should be taken and select the interval at which the medication

should be taken from the accompanying dropdown menu (minutes, hours, days, weeks, months, years).

6. In the duration of treatment section, input a positive numerical value defining the period over which the medication should be administered and select the unit of time from the accompanying dropdown menu (days, weeks, months, years).
7. Attempt to confirm the prescription.

Expected Results:

- Class 1: The system confirms the prescription, adding it to the patient's profile accordingly.
- Class 2: The system should not allow the prescription to be issued. A message should be displayed highlighting that the medicine name has not been filled.

Test Case 2

Test Case Title: Verify Dosage Provision in Prescription

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used to Generate Test Case: Equivalence Class Partitioning.

I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of providing the dosage numeric input when issuing a prescription, the classes are "Dosage Amount Provided" and "No Dosage Amount Provided," determined by whether the doctor has provided a dosage numeric input when filling out a prescription.

Preconditions:

1. The tester is logged into a test doctor account.
2. The doctor account has at least one appointment with a patient account (the doctor can only issue prescriptions to the patients he/she has appointments with).

Input:

1. Class 1 (Dosage Amount Provided): A positive number to indicate a specific dosage amount.
2. Class 2 (No Dosage Amount Provided): No input in the respective field.

Steps to Execute:

1. Navigate to the prescription issuing section of the doctor's profile.
2. Select a patient to issue a prescription for.
3. Type in an input in the respective field for medicine name.

- For Class 1:

    4. In the dosage section, input a positive numerical value indicating the amount of the medication and select the unit of measurement from the accompanying dropdown menu (mg, ml, pills, droplets).

- For Class 2:

    4. In the dosage section, leave the field for numerical input empty and select the unit of measurement from the accompanying dropdown menu (mg, ml, pills, droplets).

5. In the frequency section, input a positive numerical value indicating how often the medication should be taken and select the interval at which the medication

should be taken from the accompanying dropdown menu (minutes, hours, days, weeks, months, years).

6. In the duration of treatment section, input a positive numerical value defining the period over which the medication should be administered and select the unit of time from the accompanying dropdown menu (days, weeks, months, years).
7. Attempt to confirm the prescription.

Expected Results:

1. Class 1: The system confirms the prescription, adding it to the patient's profile accordingly.
2. Class 2: The system should not allow the prescription to be issued. A message should be displayed highlighting that the dosage numerical input field has not been filled.

Test Case 3

Test Case Title: Verify Frequency Interval Selection in Prescription

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used to Generate Test Case: Equivalence Class Partitioning. I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of providing the interval for the frequency at which the medication should be taken, the classes are "Frequency Interval Selected" and "No Frequency Interval Selected," determined by whether the doctor has selected an interval for medication frequency when filling out a prescription.

Preconditions:

1. The tester is logged into a test doctor account.
2. The doctor account has at least one appointment with a patient account (the doctor can only issue prescriptions to the patients he/she has appointments with).

Input:

- For class 1 (Frequency Interval Selected): a selection from a dropdown for the interval for the frequency at which the medication should be taken.

- For class 2 (No Frequency Interval Selected): no selection from the frequency interval dropdown.

Steps to Execute:

1. Navigate to the prescription issuing section of the doctor's profile.
2. Select a patient to issue a prescription for.
3. Type in an input in the respective field for medicine name.
4. In the dosage section, input a positive numerical value indicating the amount of the medication and select the unit of measurement from the accompanying dropdown menu (mg, ml, pills, droplets).

- For Class 1:

    5. In the frequency section, input a positive numerical value indicating how often the medication should be taken and select the interval at which the medication should be taken from the accompanying dropdown menu (minutes, hours, days, weeks, months, years).

- For Class 2:

5. In the frequency section, input a positive numerical value indicating how often the medication should be taken but do not select the interval at which the medication should be taken from the accompanying dropdown menu.

6. In the duration of treatment section, input a positive numerical value defining the period over which the medication should be administered and select the unit of time from the accompanying dropdown menu (days, weeks, months, years).

7. Attempt to confirm the prescription.

Expected Results:

1. Class 1: The system confirms the prescription, adding it to the patient's profile accordingly.
2. Class 2: The system should not allow the prescription to be issued. A message should be displayed highlighting that no selection was made in the frequency interval dropdown menu.

Test Case 4

Test Case Title: Verify Duration of Treatment Provision in Prescription

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used to Generate Test Case: Equivalence Class Partitioning.

I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of providing the duration of treatment numerical input when issuing a prescription, the classes are "Duration of Treatment Provided" and "No Duration of Treatment Provided," determined by whether the doctor has provided a duration of treatment numeric input when filling out a prescription.

Preconditions:

1. The tester is logged into a test doctor account.
2. The doctor account has at least one appointment with a patient account (the doctor can only issue prescriptions to the patients he/she has appointments with).

Input:

1. Class 1 (Duration of Treatment Provided): A positive number to indicate for how long a medicine should be taken.
2. Class 2 (No Duration of Treatment Provided): No input.

Steps to Execute:

1. Navigate to the prescription issuing section of the doctor's profile.
2. Select a patient to issue a prescription for.
3. Type in an input in the respective field for medicine name.
4. In the dosage section, input a positive numerical value indicating the amount of the medication and select the unit of measurement from the accompanying dropdown menu (mg, ml, pills, droplets).
5. In the frequency section, input a positive numerical value indicating how often the medication should be taken and select the interval at which the medication should be taken from the accompanying dropdown menu (minutes, hours, days, weeks, months, years).

- For Class 1:

    6. In the duration of treatment section, input a positive numerical value defining the period over which the medication should be administered and select the unit of time from the accompanying dropdown menu (days, weeks, months, years).

- For Class 2:

    6. In the duration of treatment section, leave the field for numerical input empty and select the unit of time from the accompanying dropdown menu (days, weeks, months, years).

7. Attempt to confirm the prescription.

Expected Results:

1. Class 1: The system confirms the prescription, adding it to the patient's profile accordingly.
2. Class 2: The system should not allow the prescription to be issued. A message should be displayed highlighting that the duration of treatment numerical input field has not been filled.

Test Case 5

Test Case Title: Verify Dosage Numerical Input Range in Prescription

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used to Generate Test Case: Equivalence Class Partitioning.

I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of providing a valid numerical input for the dosage when issuing a prescription, the classes are "Positive Dosage Numerical Input" and " Non-Positive Dosage Numerical Input," determined by whether the doctor has provided a positive number for the dosage when filling out a prescription.

Preconditions:

1. The tester is logged into a test doctor account.
2. The doctor account has at least one appointment with a patient account (the doctor can only issue prescriptions to the patients he/she has appointments with).

Input:

1. Class 1 (Positive Dosage Numerical Input): A positive number to indicate a specific dosage amount.
2. Class 2 (Non-Positive Dosage Numerical Input): a non-positive number in the respective field.

Steps to Execute:

1. Navigate to the prescription issuing section of the doctor's profile.
2. Select a patient to issue a prescription for.
3. Type in an input in the respective field for medicine name.

- For Class 1:

    4. In the dosage section, input a positive numerical value indicating the amount of the medication and select the unit of measurement from the accompanying dropdown menu (mg, ml, pills, droplets).

- For Class 2:

    4. In the dosage section, input a non-positive number in the field for numerical input and select the unit of measurement from the accompanying dropdown menu (mg, ml, pills, droplets).

5. In the frequency section, input a positive numerical value indicating how often the medication should be taken and select the interval at which the medication

should be taken from the accompanying dropdown menu (minutes, hours, days, weeks, months, years).

6. In the duration of treatment section, input a positive numerical value defining the period over which the medication should be administered and select the unit of time from the accompanying dropdown menu (days, weeks, months, years).
7. Attempt to confirm the prescription.

Expected Results:

1. Class 1: The system confirms the prescription, adding it to the patient's profile accordingly.
2. Class 2: The system should not allow the prescription to be issued. A message should be displayed highlighting that an invalid input has been provided in the dosage numerical input field.

Test Case 6

Test Case Title: Verify Frequency Numerical Input Range in Prescription

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used to Generate Test Case: Equivalence Class Partitioning. I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of providing a valid numerical input for the frequency of the medication when issuing a prescription, the classes are "Positive Frequency Numerical Input" and " Non-Positive Frequency Numerical Input," determined by the whether the doctor has provided a positive number for the frequency of medication when filling out a prescription.

Preconditions:

1. The tester is logged into a test doctor account.
2. The doctor account has at least one appointment with a patient account (the doctor can only issue prescriptions to the patients he/she has appointments with).

Input:

1. For class 1 (Positive Frequency Numerical Input): A positive number to indicate the frequency of taking a medication.
2. For class 2 (NonPositive Frequency Numerical Input): a non-positive number in the respective field.

Steps to Execute:

1. Navigate to the prescription issuing section of the doctor's profile.
2. Select a patient to issue a prescription for.
3. Type in an input in the respective field for medicine name.
4. In the dosage section, input a positive numerical value indicating the amount of the medication and select the unit of measurement from the accompanying dropdown menu (mg, ml, pills, droplets).

- For Class 1:

  5. In the frequency section, input a positive numerical value indicating how often the medication should be taken and select the interval at which the medication should be taken from the accompanying dropdown menu (minutes, hours, days, weeks, months, years).

- For Class 2:

5. In the frequency section, input a non-positive numerical value indicating how often the medication should be taken and select the interval at which the medication should be taken from the accompanying dropdown menu (minutes, hours, days, weeks, months, years).

6. In the duration of treatment section, input a positive numerical value defining the period over which the medication should be administered and select the unit of time from the accompanying dropdown menu (days, weeks, months, years).

7. Attempt to confirm the prescription.

Expected Results:

1. Class 1: The system confirms the prescription, adding it to the patient's profile accordingly.
2. Class 2: The system should not allow the prescription to be issued. A message should be displayed highlighting that an invalid input has been provided in the medication frequency numerical input field.

Test Case 7

Test Case Title: Verify Duration of Treatment Numerical Input Range in Prescription

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used to Generate Test Case: Equivalence Class Partitioning.

I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of providing a valid numeric input for duration of treatment when issuing a prescription, the classes are "Positive Duration of Treatment Numerical Input" and "Non-Positive Duration of Treatment Numerical Input," determined by the whether the doctor has provided a positive duration of treatment numeric input when filling out a prescription.

Preconditions:

1. The tester is logged into a test doctor account.
2. The doctor account has at least one appointment with a patient account (the doctor can only issue prescriptions to the patients he/she has appointments with).

Input:

1. Class 1 (Positive Duration of Treatment Numerical Input): A positive number to indicate for how long a medicine should be taken.
2. Class 2 (Non-Positive Duration of Treatment Numerical Input): a non-positive number in the respective field.

Steps to Execute:

1. Navigate to the prescription issuing section of the doctor's profile.
2. Select a patient to issue a prescription for.
3. Type in an input in the respective field for medicine name.
4. In the dosage section, input a positive numerical value indicating the amount of the medication and select the unit of measurement from the accompanying dropdown menu (mg, ml, pills, droplets).
5. In the frequency section, input a positive numerical value indicating how often the medication should be taken and select the interval at which the medication should be taken from the accompanying dropdown menu (minutes, hours, days, weeks, months, years).

- For Class 1:

    6. In the duration of treatment section, input a positive numerical value defining the period over which the medication should be administered and

select the unit of time from the accompanying dropdown menu (days, weeks, months, years).

- For Class 2:

  6. In the duration of treatment section, enter a non-positive number for the numerical input and select the unit of time from the accompanying dropdown menu (days, weeks, months, years).

7. Attempt to confirm the prescription.

Expected Results:

1. **Class 1:** The system confirms the prescription, adding it to the patient's profile accordingly.
2. **Class 2:** The system should not allow the prescription to be issued. A message should be displayed highlighting that an invalid input has been provided for the duration of treatment numerical input field.

Test Case 8

Test Case Title: Truth Table Testing for Prescription Issuing

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used: Truth Table Testing

	Rules			
Conditions	**1**	**2**	**3**	**4**
Doctor is Logged in	T	T	T	F
User exists (Appointment exists)	T	T	F	T
All details filled in	T	F	F	F
System confirms successful prescription addition	□			
System displays an error message, notifying the doctor to fill in all prescription details.		□		
System displays a notification that the user does not exist..			□	
Doctor is redirected to login				□

Table 7: Truth Table Testing Test Case Table

Note: some of the conditions cannot happen without others. For example, adding all details given that the patient does not exist cannot occur, so it is omitted from the table due to logic issues.

Preconditions

Condition 1:

1. The tester is logged into a test patient account.
2. The system is on and available (Not in downtime).
3. The patient is connected to a doctor.

Condition 2:

1. The tester is logged into a test patient account.
2. The system is on and available (Not in downtime).
3. The patient is connected to a doctor.

Condition 3:

1. The tester is logged into a test patient account.
2. The system is on and available (Not in downtime).

Condition 4:

1. The system is on and available (Not in downtime).

Input:

Condition 1:

1. Valid user.
2. Completeness of prescription details: medicine name, dosage and duration.

Condition 2:

1. Valid user.
2. Completeness of prescription details: medicine name, dosage and duration.

Condition 3:

1. No input conditions

Condition 4:

1. No input conditions

Steps to Execute:

Condition 1:

1. Navigate to the Prescription Issuing Section.
2. Select a patient from the list on their profile.
3. Click on the option to issue a prescription.
4. Fill in the medication name.
5. Fill in negative dosage with a unit of measurement.
6. Fill in duration with a unit of time.
7. Confirm the prescription.

Condition 2:

1. Navigate to the Prescription Issuing Section.
2. Select a patient from the list on their profile.
3. Click on the option to issue a prescription.

4. Fill in the medication name.
5. Fill in negative dosage with a unit of measurement.
6. Fill in duration with a unit of time.
7. Attempt to confirm the prescription.

Condition 3:

1. Navigate to the Prescription Issuing Section.
2. Attempt to select a patient from the list on the profile.

Condition 4:

1. Do not log in.

Expected Results:

Condition 1: System confirms successful upload of the prescription.

Condition 2: System notifies the user (doctor) to input all necessary fields: medication name, dosage with unit of measurement, and duration with unit of time. of the prescription, no system updates are done. .

Condition 3: The system does not display a user, no system updates are done.

Condition 4: The doctor is redirected to login, no system updates are done.

Test Case 9

Test Case Title: Boundary Value Testing for Dosage in milligrams

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used: Boundary Value Testing

Boundary Value Testing for Dosage in Milligrams Test Case		
Invalid Test Case Min - 1	**Valid Test Case Min, Mid, Max**	**Invalid Test Case Max + 1**
0	1, 1000, 2000	2001

Table 8: Boundary Value Testing for Dosage in Milligrams Test Case Table

Preconditions

1. The tester is logged into a test doctor account.
2. The system is on and available (Not in downtime).
3. The doctor account has at least one appointment with a patient account (the doctor can only upload medical records to the patients he/she has appointments with).

Input: Dosage input of 0, 1, 1000, 2000, and 2001 mg

Steps to Execute:

1. Navigate to the Prescription Issuing Section.
2. Select a patient from the list on their profile.
3. Click on the option to issue a prescription.
4. Fill in the medication name.
5. Fill in the dosage as 0.
6. Fill in the remaining required sections correctly.
7. Confirm the prescription and observe.
8. Repeat the steps for each of the other mg dosage inputs.

Expected Results: The system notifies the user that a positive dosage input must be made.

Test Case 10

Test Case Title: Validate Dosage Field for Completeness and Correct Input

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used: Technique Used to Generate Test Case with Proof: Truth Table

Proof: By defining logical conditions and expected outcomes in a Truth Table, we provide a clear framework for testing the system's response to different dosage input scenarios. This ensures that all potential input cases are covered and the system's reactions are predictable and as designed.

Dosage Input Provided	Dosage Format Valid	Allow Submission
F	**F**	**F**
T	**F**	**F**
T	**T**	**T**

Preconditions

1. The doctor is logged into their account.
2. The doctor has selected a patient for whom a prescription is to be issued.

Input:

1. **Condition 1:** No Input in Dosage Field
2. **Condition 2:** Invalid Input in Dosage Field
3. **Condition 3:** Valid Input in Dosage Field

Steps to Execute:

Condition 1:

1. Access the prescription form.
2. Leave the dosage field blank.
3. Attempt to submit the prescription.
4. Observe and record the system's response.

Condition 2:

1. Access the prescription form.
2. Enter an invalid dosage (e.g., "1.34 mg") in the dosage field.
3. Attempt to submit the prescription.
4. Observe and record the system's response.

Condition 3:

1. Access the prescription form.
2. Enter a valid and realistic dosage (e.g., "500 mg") in the dosage field.
3. Attempt to submit the prescription.
4. Observe and record the system's response.

Expected Results:

Condition 1: The system should reject the submission and display an error message indicating that the dosage field is required.
Condition 2: The system should reject the input, showing an error message indicating that the entered dosage is invalid.
Condition 3: The prescription should be accepted and processed without any errors, confirming that the input validation rules are correctly implemented.

Test Case 11

Test Case Title: Verify Medicine Name Provision in Prescription

Use Case Tested by the Test Case: Issue Medical Prescription

Technique Used to Generate Test Case: All Pairs Testing.

With this technique, we aim to uncover if any errors arise based on the pair combination of the dosage, frequency, and duration dropdown menu choices. Below is the all-pairs table we generated:

Pairwise Testing Test Cases		
Dosage	**Frequency**	**Duration**
mg	hour	days
mg	day	weeks
mg	week	months
mg	month	years
pill	day	months
pill	week	years
pill	month	days
pill	hour	weeks
ml	week	days
ml	month	weeks
ml	hour	months
ml	day	years
droplet	month	months
droplet	hour	years
droplet	day	days

droplet	week	weeks

Preconditions:

1. The tester is logged into a test doctor account.
2. The doctor account has at least one appointment with a patient account (the doctor can only issue prescriptions to the patients he/she has appointments with).

Input: Selections from the dropdown menus for the dosage, frequency, and duration of medicine being prescribed.

Steps to Execute:

1. Navigate to the prescription issuing section of the doctor's profile.
2. Select a patient to issue a prescription for.
3. Type in an input in the respective field for medicine name.
4. In the dosage, frequency, and duration of treatment sections, input valid positive integers in the respective fields.
5. Select one row from the all-pairs table and fill the dosage, frequency, and duration dropdowns according to the combination in the row.
6. Attempt to confirm the prescription.
7. Repeat the steps for all rows in the table.

Expected Results: The system should always successfully confirm the prescription, adding it to the patient's profile accordingly.

3. Upload a Medical Record

Test Case 1

Test Case Title: Uploading File Without Proceeding

Use Case Tested by the Test Case: Upload a Medical Record

Technique Used: Exploratory Testing. Here, we are navigating to another section without completing the upload to check if the system commits data that wasn't fully submitted. This verifies that incomplete processes do not lead to unwanted outcomes.

Precondition:

1. The tester is logged into a test doctor account.
2. The system is on and available (Not in downtime).
3. The doctor is connected to a patient through an appointment.
4. There exists a file to be uploaded.

Input: One PDF file of size at most 16 MB is selected for upload.

Steps to Execute:

1. Navigate to the medical records uploading section of the doctor’s profile.
2. Select the patient to upload a medical record for.
3. Select a PDF file whose size is at most 16 MB to upload.
4. Navigate to another section without uploading.

Expected Results: No medical record is uploaded to the database under the selected user’s entry.

Test Case 2

Test Case Title: Verify File Provision in Medical Record Uploading

Use Case Tested by the Test Case: Upload a Medical Record

Technique Used to Generate Test Case: Equivalence Class Partitioning. I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of providing a file to upload for a patient's medical records, the classes are "File Provided" and "No File Provided," determined by whether the doctor has provided a file when uploading a medical record.

Preconditions:

- The tester is logged into a test doctor account.
- The doctor account has at least one appointment with a patient account (the doctor can only upload medical records to the patients he/she has appointments with).

Input:

- Class 1 (File Provided): One pdf file of the accepted size (at most 16MB) is selected for upload.
- Class 2 (No File Provided): No file is selected for upload.

Steps to Execute:

1. Navigate to the medical records uploading section of the doctor's profile.
2. Select the patient to upload a medical record for.

- For Class 1:

  3. Select a PDF file of size at most 16 MB to upload.

- For Class 2:

  3. Do not select a file to upload.

4. Attempt to confirm the medical record uploading.

Expected Results:

1. For Class 1: The system should successfully confirm the medical record uploading, adding the record to the patient's profile accordingly.
2. For Class 2: The system should not allow the medical record uploading to proceed. An error message should be displayed, indicating that no file has been selected to be uploaded as a medical record.

Test Case 3

Test Case Title: Verify File Type in Medical Record Uploading

Use Case Tested by the Test Case: Upload a Medical Record

Technique Used to Generate Test Case: Equivalence Class Partitioning. I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of uploading a valid file type to a patient's medical records, the classes are "PDF Files" and "Non-PDF Files," determined by whether the doctor has attempted to upload a PDF file or another type for a patient's medical record.

Preconditions:

1. The tester is logged into a test doctor account.
2. The doctor account has at least one appointment with a patient account (the doctor can only upload medical records to the patients he/she has appointments with).

Input:

1. Class 1 (PDF Files): One PDF file of size at most 16MB is selected for upload.
2. Class 2 (Non-PDF Files): a non-PDF file is selected for upload.

Steps to Execute:

1. Navigate to the medical records uploading section of the doctor's profile.
2. Select the patient to upload a medical record for.

- For Class 1:

    3. Select a PDF file of size at most 16 MB to upload.

- For Class 2:

    3. Select a non-PDF file (a file whose type is not PDF) to upload.

4. Attempt to confirm the medical record uploading.

Expected Results:

1. For Class 1: The system should successfully confirm the medical record uploading, adding the record to the patient's profile accordingly.
2. For Class 2: The system should not allow the medical record uploading to proceed. An error message should be displayed, indicating that the file type provided for the medical record is invalid (not acceptable by the system).

Test Case 4

Test Case Title: Verify File Size in Medical Record Uploading

Use Case Tested by the Test Case: Upload a Medical Record

Technique Used to Generate Test Case: Equivalence Class Partitioning.

I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of uploading a file of valid size to a patient's medical records, the classes are "File of size at most 16MB" and "File of size greater than 16MB," determined by the whether the doctor has attempted to upload a PDF file of the acceptable size for a patient's medical record.

Preconditions:

1. The tester is logged into a test doctor account.
2. The doctor account has at least one appointment with a patient account (the doctor can only upload medical records to the patients he/she has appointments with).

Input:

1. Class 1 (File of size at most 16MB): One PDF file of size at most 16 MB is selected for upload.
2. Class 2 (File of size greater than 16MB): One PDF file of size greater than 16 MB is selected for upload.

Steps to Execute:

1. Navigate to the medical records uploading section of the doctor's profile.
2. Select the patient to upload a medical record for.

- For Class 1:

  3. Select a PDF file whose size is at most 16 MB to upload.

- For Class 2:

  3. Select a PDF file whose size is greater than 16 MB to upload.

4. Attempt to confirm the medical record uploading.

Expected Results:

1. For Class 1: The system should successfully confirm the medical record uploading, adding the record to the patient's profile accordingly.
2. For Class 2: The system should not allow the medical record uploading to proceed. An error message should be displayed, indicating that the size of the file provided for the medical record is too large (not acceptable by the system).

Test Case 5

Test Case Title: Test Upload Limits with Maximum File Size

Use Case Tested by the Test Case: Upload a Medical Record

Technique Used to Generate Test Case with Proof: Boundary Value Testing

Proof: The technique involves testing at the edges of input limits. Here, the limits are the maximum file size allowed for uploads. The test will include scenarios where the file size is just below the maximum, exactly at the maximum, and slightly above the maximum limit to observe how the system handles these cases.

Preconditions

- The doctor is logged into their system.
- The doctor has selected a patient to upload a medical record for.

Input:

1. Condition 1: File slightly below the maximum size limit, one at the maximum limit, and one slightly above the maximum limit .
2. Condition 2: File at the maximum limit,
3. Condition 3: File slightly above the maximum limit

(Example: since the limit is 16MB, we can test with files of sizes 15.95MB, 16MB, and 16.05MB)

Steps to Execute:

Condition 1: File Slightly Below the Maximum Size Limit

1. Log into the doctor's account.
2. Navigate to the medical record upload section for a selected patient.
3. Attempt to upload a file slightly below the maximum size limit (e.g., 15.95MB).
4. Observe and record any error messages or success confirmations.
5. Check if the upload process completes successfully.

Condition 2: File at the Maximum Size Limit

1. Log into the doctor's account.
2. Navigate to the medical record upload section for a selected patient.
3. Attempt to upload a file exactly at the maximum size limit (e.g., 16MB).

4. Observe and record any error messages or success confirmations.
5. Check if the upload process completes successfully.

Condition 3: File Slightly Above the Maximum Size Limit

1. Log into the doctor's account.
2. Navigate to the medical record upload section for a selected patient.
3. Attempt to upload a file slightly above the maximum size limit (e.g., 16.05MB).
4. Observe and record any error messages or success confirmations.
5. Check if the upload process fails due to exceeding the file size limit.

Expected Results:

1. **Condition 1:** The system should successfully upload the file that is just below the maximum limit without any issues.
2. **Condition 2**: The system should also handle the file exactly at the limit, assuming proper file handling is coded for edge cases.
3. **Condition 3:** The system should reject the upload, displaying an appropriate error message about the file size exceeding the maximum allowed limit.

Test Case 6

Test Case Title: Confirm Accessibility of Medical Records Post-Upload

Use Case Tested by the Test Case: Upload a Medical Record

Technique Used to Generate Test Case with Proof: Exploratory Testing. With this approach, we test a sequence of actions reflecting real-world usage—from uploading a medical record to checking its availability in the patient's profile. This ensures the system functions collectively and correctly across multiple functionalities.

Preconditions

- The doctor is logged into their system.
- The doctor has selected a patient to whom the medical record will be uploaded.

Input:

- Medical Record File: One PDF file of size at most 16 MB.

Steps to Execute:

- **Log into the Doctor's Account**:
  - Navigate to the system and enter login credentials.
- **Select a Patient**:
  - Go to the patient list.
  - Choose a patient for whom the medical record is intended.
- **Upload the Medical Record**:
  - Navigate to the 'Upload Medical Records' section within the selected patient’s profile.
  - Choose a medical record file to upload.
  - Confirm the upload and observe any notifications or messages confirming the successful upload.
- **Verify the Upload**:
  - After uploading, remain on the patient's profile.
  - Navigate to the section where uploaded records are displayed.
  - Look for the newly uploaded document in the list or through a specific interface designed for accessing medical records.
- **Access the Uploaded Record**:
  - Attempt to open or view the uploaded record.
  - Check if the document is correctly rendered and accessible.

Expected Results:

- The upload process should complete without errors, and a confirmation message should be displayed.
- The medical record should appear in the list or designated area for accessing patient records within the profile.
- The file should be accessible, openable, and viewable without any issues, indicating it is stored correctly and linked appropriately to the patient’s profile.

4. Initiate Video Call Appointment

Test Case 1

Test Case Title: No scheduled appointments

Use Case Tested by the Test Case: Initiate Video Call Appointment

Technique Used to generate test case: Exploratory Testing. In this case, we are guessing that a common issue users might encounter is attempting to view or join scheduled appointments when none exist. This situation could lead to confusion or errors if a video call appointment is displayed given that there is no online appointment scheduled.

Preconditions

1. The tester is logged into a test doctor account.
2. The system is on and available (Not in downtime).
3. No scheduled appointments.

Input: View appointments on the patient's profile.

Steps to Execute:

1. Navigate to the booked appointments section of the patient's profile.

Expected Results: The system should not display any appointment options, and show a message indicating there are no scheduled appointments.

Test Case 2

Test Case Title: In-person Appointment Mistakenly Selected for Video Call

Use Case Tested by the Test Case: Initiate Video Call Appointment

Technique Used to generate test case: Exploratory Testing

Preconditions

1. The tester is logged into a test doctor account.

2. The system is on and available (Not in downtime).
3. Appointment scheduled in-person.

Patient is logged in, selects an appointment scheduled as in-person.

Input: Patient attempts to initiate a video call for an in-person appointment.

Steps to Execute:

1. Navigate to the booked appointments section of the profile.

Expected Results: The system should not provide the capability to join a video call appointment for appointments booked for in-person.

Test Case 3

Test Case Title: Ensure Successful Connection to Scheduled Video Call

Use Case Tested by the Test Case: Initiate Video Call Appointment

Technique Used to generate test case: Exploratory Testing, to test a specific action within the system—joining a video call—to ensure it functions as intended and provides a reliable, user-friendly experience.

Preconditions

- The patient has previously scheduled a video call appointment for the current day with a doctor.
- The patient is logged into their account and on the appropriate interface to join the video call.

Input:

- Video Button Click: Provided as part of the appointment details on the patient's appointments section.

Steps to Execute:

- **Navigate to the Scheduled Appointments Section**:
    - Click the appointments section in the side nav bar.
- **Select Initiate the Video Call for the Scheduled Video Call Appointment**:
    - Click the button to join the video call at the scheduled time.
    - Verify that any necessary permissions (e.g., camera, microphone) are granted if prompted.
- **Observe the Connection Process**:
    - Watch for any indicators of the connection being established, such as loading screens, confirmation messages, or direct entry into the video interface.
- **Verify Successful Joining of the Call**:
    - Confirm that the video and audio functionalities are working.
    - Ensure that both the patient's and the doctor's video feeds are visible and that communication is possible.

Expected Results:

- Upon clicking the button, the system should successfully connect the patient to the scheduled video call without errors.
- The patient should see themselves and the doctor on the video feed, and both parties should be able to communicate clearly.

Test Case 4

Test Case Title: Verify Virtual Consultation Access

Use Case Tested by the Test Case: Initiate Video Call Appointment

Technique Used to Generate Test Case: Decision Table Testing. I use this technique to determine how different combinations of conditions will affect whether a patient can join a booked appointment virtually, associating specific sets of conditions with specific actions. Below is the decision table I created:

	Rules		
Conditions	1	2	3
The appointment the patient seeks to join was booked for virtual consultation	T	T	F
The date of the virtual appointment the patient seeks to join matches the current date	T	F	
Actions			
Display an activated button to join a video call beside the appointment	✓		
Display an inactivated button to join a video call beside the appointment		✓	
Display a note beside the appointment highlighting that this appointment was booked for in-person consultation			✓

So we essentially have 3 scenarios we're testing

1. Scenario 1: The patient attempts to virtually join a booked appointment that was indeed set to be conducted virtually on the current day.
2. Scenario 2: The patient attempts to virtually join a booked appointment that was indeed set to be conducted virtually but not on the current day.
3. Scenario 3: The patient attempts to virtually join a booked appointment that was not set to be conducted virtually but rather in-person.

And the system is expected to behave differently according to each.

Preconditions:

- The tester is logged in to a test patient account that has at least one appointment booked for virtual consultation on the current day, at least one appointment booked for virtual consultation on another day, and at least one appointment booked for in-person consultation.

Input: a request to join a video call, made through a button click (if applicable).

Steps to Execute:

1. Navigate to the booked appointments section of the patient's profile.

- For Scenario 1:

  2. Choose a virtual-mode appointment booked for the current date and click on the button to join the video call.

- For Scenario 2:

  2. Choose a virtual-mode appointment booked for another date and click on the button to join the video call.

- For Scenario 3:

  2. Choose any in-person-mode booked appointment.

Expected Results:

1. Scenario 1: The system successfully redirects the patient to join the video call.
2. Scenario 2: The system displays a message indicating to the user that the virtual consultation is not set for the current date.
3. Scenario 3: The system displays a note beside the appointment highlighting that this appointment was booked for in-person consultation, essentially not allowing the patient to attempt to join this appointment virtually.

5. Request Intelligent Diagnostic Support

Test Case 1

Test Case Title: Test Extreme Boundary for Year of Birth Input

Use Case Tested by the Test Case: Request Intelligent Diagnostic Support

Technique Used: Boundary Value Analysis

Boundary Value for Year of Birth Input Test Case		
Invalid Test Case Min - 1	**Valid Test Case Min, Mid, Max**	**Invalid Test Case Max + 1**
1906	1907, 1965, 2024	2025

Table 1: Boundary Value for Year of Birth Test Case Table

Preconditions

1. The tester is logged into a test doctor account.
2. The system is on and available (Not in downtime).

Input:

Symptoms: Any symptoms
Gender: "Male"
Year of Birth: "1907" (assuming 1907 is the earliest valid year as that is the year the older person living today was born[2]).

Steps to Execute:

1. Enter symptoms in the symptoms field
2. Select "Male" from the gender dropdown.
3. Enter "1907" in the year of birth.
4. Submit the form.

Expected Results: The system should accept the input without errors and fetch diagnostics based on the provided data.

[2]According to Gerontology Research Group: https://www.grg-supercentenarians.org/world-supercentenarian-rankings-list/

Test Case 2

Test Case Title: Test Empty Symptom Selection

Use Case Tested by the Test Case: Request Intelligent Diagnostic Support

Technique Used: Equivalence Partitioning

Equivalence Partitioning Test Case	
Invalid	**Valid**
No Symptom Selection	Any symptom

Table 2: Equivalence Partitioning for Symptom Selection Test Case Table

Preconditions

1. The tester is logged into a test doctor account.
2. The system is on and available (Not in downtime).

Input:

Condition 1:

Symptoms: "" ,
Gender: "Male",
Year of Birth: "1973".

Condition 2:

Symptoms: "Headache",
Gender: "Male",
Year of Birth: "1973".

Steps to Execute:

Condition 1:

1. Navigate to the Diagnostic section.
2. Enter "" in the symptoms field.
3. Enter “Male” for gender from the gender dropdown.
4. Submit the form.

Condition 2:

1. Navigate to the Diagnostic section.

2. Enter symptom in the symptoms field.
3. Enter “Male” for gender from the gender dropdown.
4. Submit the form.

Expected Results:

Condition 1: The system should notify the user to select at least one symptom.
Condition 2: The system should accept the input without errors and fetch diagnostics based on the provided data.

Test Case 3

Test Case Title: Test Valid Number of Symptom Selections

Use Case Tested by the Test Case: Request Intelligent Diagnostic Support

Technique Used: Equivalence Partitioning

Equivalence Partitioning Test Case	
Invalid	**Valid**
More than 5 symptoms	Less than 5 symptoms, at least one.

Table 2: Equivalence Partitioning for Valid Number of Symptom Selections Selection Test Case Table

Preconditions

1. The tester is logged into a test doctor account.
2. The system is on and available (Not in downtime).

Input:

Condition 1:

Symptoms: Headache, Abdominal Pain, Heartburn, Cough, Runny Nose, Tiredness

Gender: "Male",

Year of Birth: "1973".

Condition 2:

Symptoms: Any symptoms

Gender: "Male",

Year of Birth: "1973".

Steps to Execute:

Condition 1:

1. Navigate to the Diagnostic section.
2. Enter six symptoms in the symptoms field.
3. Enter “Male” for gender from the gender dropdown.
4. Submit the form.

Condition 2:

1. Navigate to the Diagnostic section.

2. Enter 1-5 symptoms in the symptoms field.
3. Enter “Male” for gender from the gender dropdown.
4. Submit the form.

Expected Results:

Condition 1: The system does not allow the user to select Tiredness at all.

Condition 2: The system should accept the input without errors and fetch diagnostics based on the provided data.

Test Case 4

Test Case Title: Validate Symptom Input Handling for Intelligent Diagnostic Support Request

Use Case Tested by the Test Case: Request Intelligent Diagnostic Support

Technique Used to Generate Test Case: Equivalence Class Partitioning.

I use equivalence class partitioning to divide the input data into partitions that represent different test conditions, which here are based on the number of symptoms sent to the ApiMedic API. Since a doctor needs to provide at least 1 symptom and at most 5 at a time, the classes are based on the number of symptoms inputted—specifically, 1, 2, 3, 4, or 5 symptoms for valid inputs, and I'll test 0 and 6 inputs for invalid inputs.

Preconditions: The tester is logged into a test doctor account.

Input: symptoms, which will be passed by choosing them from a dropdown.

Steps to Execute:

1. Navigate to the Intelligent Diagnostic Support section of the doctor's profile.

- For Class 1:

  2. Input one symptom by selecting one from the dropdown.

- For Class 2:

  2. Input two symptoms by selecting two from the dropdown.

- For Class 3:

  2. Input three symptoms by selecting three from the dropdown.

- For Class 4:

  2. Input four symptoms by selecting four from the dropdown.

- For Class 5:

  2. Input five symptoms by selecting five from the dropdown.

- For Class 6:

  2. Do not input any symptoms in the respective field.

- For Class 7:

  2. Input six symptoms by attempting to select six from the dropdown.

3. Select a gender from the respective dropdown (“Male” or “Female”).
4. Attempt to submit the intelligent diagnostic support request.

Expected Results:

1. **For Classes 1 through 5**: The system should successfully process the input and provide diagnosis suggestions as per the ApiMedic API response.
2. **For Class 6:** The system should not allow the intelligent diagnostic support request to go through. An error message should be displayed, indicating that an invalid number of symptoms has been passed in the symptoms field.
3. **For Class 7**: The system should not allow the selection of more than 5 symptoms; the doctor has to deselect at least one of the five already selected.

Test Case 5

Test Case Title: Verify Gender Selection for Intelligent Diagnostic Support Request

Use Case Tested by the Test Case: Request Intelligent Diagnostic Support

Technique Used to Generate Test Case: Equivalence Class Partitioning.

I use this technique to divide the input into classes based on the expected response of the system. Here, in the context of selecting the relevant gender when requesting intelligent diagnostic support from the system, the classes are "Gender Selected" and "No Gender Selected," determined by whether the doctor has selected a gender to proceed with the intelligent diagnostic support request.

Preconditions:

- The tester is logged into a test doctor account.

Input:

- For class 1 (Gender Selected): a selection from the dropdown for the gender relevant for the diagnosis request.
- For class 2 (No Gender Selected): no selection from the gender dropdown.

Steps to Execute:

1. Navigate to the Intelligent Diagnostic Support section of the doctor's profile.
2. Input one to five symptoms by selecting them from the dropdown.
- For Class 1:

  3. Select a gender from the respective dropdown ("Male" or "Female").

- For Class 2:

  3. Leave the gender dropdown empty.

4. Attempt to submit the intelligent diagnostic support request.

Expected Results:

- Class 1: The system should successfully process the input and provide diagnosis suggestions as per the ApiMedic API response.
- Class 2: The system should not allow the intelligent diagnostic support request to go through. An error message should be displayed, indicating that no selection was made in the gender dropdown menu.

6. Login

Test Case 1

Test Case Title: Truth Table Test for Login Functionality

Use Case Tested by the Test Case: Login into Account (not a use case, but a functionality)

Technique Used to generate test case: Truth Table Testing

	Rules		
Conditions	**1**	**2**	**3**
Existing email provided	T	T	F
Correct password provided	T	F	F
User is successfully logged in	□		
System displays an error message, notifying the user that their password is incorrect		□	
System displays an error message, notifying the email does not exist within it			□

Preconditions

Condition 1:

The user has an account in the MedSync system

Condition 2:

The user has an account in the MedSync system

Condition 3:

The user does not have an account in the system

Input:

Condition 1:

1. Email that exists in the system
2. Correct password for given email

Condition 2:

1. Email that exists in the system
2. Incorrect password for given email

Condition 3:

1. Email that does not exists in the system
2. Any password

Steps to Execute:

Condition 1:

1. Go to the Login page
2. Input existing email.
3. Input correct password for the email.
4. Press the Login button.

Condition 2:

1. Go to the Login page
2. Input existing account email.
3. Input incorrect password for the email.
4. Press the Login button.

Condition 3:

1. Go to the Login page
2. Input an email not registered on MedSync.
3. Input any password for the email.
4. Press the Login button.

Expected Results: .

Condition 1: The user is successfully logged in to the system and is met with their dashboard, depending on which account they possess, doctor or patient.

Condition 2: The system displays an error message notifying the user that an incorrect password was provided.

Condition 3: The system displays an error message notifying the user that a non existing email was provided.

Test Case 2

Test Case Title: Exploratory Testing for Cancellation of Login

Use Case Tested by the Test Case: Login into Account (not a Requirement, but a functionality)

Technique Used to generate test case: Exploratory Testing

The proof of using exploratory testing here is grounded in its ability to provide an examination of the procedure behind canceling a login attempt which can happen due to a multitude of factors for any given user.

Preconditions

The user is able to access the webpage (The system is not in downtime)

Input:

Clicking the back button on the webpage.

Steps to Execute:

1. The user clicks on the Login button on the Home page.
2. The user clicks on the back button at the top left corner.

Expected Results: .

The user is successfully returned to the Home page and no Login has occurred.

Test Case 3

Test Case Title: Exploratory Testing for Switching From Login to Register

Use Case Tested by the Test Case: Login into Account (not a Requirement, but a functionality)

Technique Used to generate test case: Exploratory Testing

The proof of using exploratory testing here is grounded in its ability to provide an examination of the procedure behind switching to registration after realizing that the user does not have an account in the system.

Preconditions

The user is able to access the webpage (The system is not in downtime)

Input:

Clicking the “Don’t have an Account? Register!” button.

Steps to Execute:

1. The user clicks on the Login button on the Home page.
2. The user clicks on the “Don’t have an Account? Register!” button.

Expected Results: .

The user is successfully redirected to the Register page and no Login has occurred.

7. Register

Test Case 1

Test Case Title: Exploratory Testing for Switching From Register to Login

Use Case Tested by the Test Case: Register an Account (not a Requirement, but a functionality)

Technique Used to generate test case: Exploratory Testing

The proof of using exploratory testing here is grounded in its ability to provide an examination of the procedure behind switching to login after realizing that the user does have an account in the system.

Preconditions

The user is able to access the webpage (The system is not in downtime)

Input: Clicking the “Already have an Account? Login!” button.

Steps to Execute:

1. The user clicks on the Register button on the Home page.
2. The user clicks on the “Already have an Account? Login!” button.

Expected Results: The user is successfully redirected to the Login page and no Registration has occurred.

Test Case 2

Test Case Title: Truth Table Test for Patient Register Functionality

Use Case Tested by the Test Case: Register a Patient Account (not a Requirement, but a functionality)

Technique Used to generate test case: Truth Table

	Rules					
Conditions	**1**	**2**	**3**	**4**	**5**	**6**
First name provided	T	F	T	T	T	T
Second name provided	T	T	F	T	T	T
Email provided	T	T	T	F	T	T
Password provided	T	T	T	T	F	T
Phone Number provided	T	T	T	T	T	F
User is successfully logged in	□					

System displays an error message, notifying the user that they need to fill in a required field.		□	□	□	□	□

Note: Not filling in multiple fields just points back to the first field that is not filled in.

Preconditions

The user is able to access the webpage (The system is not in downtime)

Input:

Clicking the “Already have an Account? Login!” button.

Steps to Execute:

Condition 1:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an Email.
5. Input a Password.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Condition 2:

1. Go to the Register page
2. Do not input a First Name
3. Input a Last Name
4. Input an Email.
5. Input a Password.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Condition 3:

1. Go to the Register page
2. Input a First Name
3. Do not input a Last Name
4. Input an Email.
5. Input a Password.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Condition 4:

1. Go to the Register page
2. Input a First Name

3. Input a Last Name
4. Do not input an Email.
5. Input a Password.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Condition 5:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email.
5. Do not input a password.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Condition 6:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email.
5. Input a password.
6. Do not input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Expected Results: .

Condition 1: The user is successfully redirected to the Login page and Registration has occurred.

Condition 2: The user is prompted to fill in the missing information.

Condition 3: The user is prompted to fill in the missing information.

Condition 4: The user is prompted to fill in the missing information.

Condition 5: The user is prompted to fill in the missing information.
Condition 6: The user is prompted to fill in the missing information.

Test Case 3

Test Case Title: Truth Table Test for Doctor Register Functionality

Use Case Tested by the Test Case: Register a Patient Account (not a Requirement, but a functionality)

Technique Used to generate test case: Truth Table

	Rules							
Conditions	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
First name provided	T	F	T	T	T	T	T	T
Second name provided	T	T	F	T	T	T	T	T
Email provided	T	T	T	F	T	T	T	T
Password provided	T	T	T	T	F	T	T	T
Phone Number provided	T	T	T	T	T	F	T	T
Specialization provided	T	T	T	T	T	T	F	T
Education provided	T	T	T	T	T	T	T	F
User is successfully logged in	□							
System displays an error message, notifying the user that they need to fill in a required field.		□	□	□	□	□	□	□

Note: Not filling in multiple fields just points back to the first field that is not filled in.

Preconditions The user is able to access the webpage (The system is not in downtime)

Input: Clicking the “Already have an Account? Login!” button.

Steps to Execute:

Condition 1:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an Email.
5. Input a Password.
6. Input a Phone Number
7. Set Role to Doctor.
8. Input Specialization
9. Input Education
10. Press the Register button.

Condition 2:

1. Go to the Register page
2. Do not input a First Name
3. Input a Last Name
4. Input an Email.
5. Input a Password.
6. Input a Phone Number
7. Set Role to Doctor.
8. Input Specialization
9. Input Education

Condition 3:

1. Go to the Register page
2. Input a First Name
3. Do not input a Last Name
4. Input an Email.
5. Input a Password.
6. Input a Phone Number
7. Set Role to Doctor.
8. Input Specialization
9. Input Education

Condition 4:

1. Go to the Register page

2. Input a First Name
3. Input a Last Name
4. Do not input an Email.
5. Input a Password.
6. Input a Phone Number
7. Set Role to Doctor.
8. Input Specialization
9. Input Education

Condition 5:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email.
5. Do not input a password.
6. Input a Phone Number
7. Set Role to Doctor.
8. Input Specialization
9. Input Education

Condition 6:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email.
5. Input a password.
6. Do not input a Phone Number
7. Set Role to Doctor.
8. Input Specialization
9. Input Education

Condition 7:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email.

5. Input a password.
6. Do not input a Phone Number
7. Set Role to Doctor.
8. Do not input Specialization
9. Input Education

Condition 8:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email.
5. Input a password.
6. Do not input a Phone Number
7. Set Role to Doctor.
8. Input Specialization
9. Do not input Education

Expected Results: .

Condition 1: The user is successfully redirected to the Login page and Registration has occurred.
Condition 2: The user is prompted to fill in the missing information.
Condition 3: The user is prompted to fill in the missing information.
Condition 4: The user is prompted to fill in the missing information.
Condition 5: The user is prompted to fill in the missing information.
Condition 6: The user is prompted to fill in the missing information.
Condition 7: The user is prompted to fill in the missing information.
Condition 8: The user is prompted to fill in the missing information.

Test Case 4

Test Case Title: Equivalence Partitioning for Password Length at Registration

Use Case Tested by the Test Case: Register an Account (not a Requirement, but a functionality)

Technique Used to generate test case: Equivalence Partitioning Testing

Boundary Value for Password Length at Registration Test Case		
Invalid Test Case Min - 1	**Valid Test Case Min, Mid, Max**	**Invalid Test Case Max + 1**
Password value of 7 characters	Any value of 8, 12 and 16 characters	Password value of 17 characters

Table 1: Boundary Value for Year of Birth Test Case Table

Preconditions The user is able to access the webpage (The system is not in downtime)

Input:

Condition 1:

1. Password value of 7 characters
2. Valid values for First Name, Last Name, Email and Phone Number (i.e., inputs filled in)

Condition 2:

1. Password value of 8, 12 and 16 characters
2. Valid values for First Name, Last Name, Email and Phone Number (i.e., inputs filled in)

Condition 3:

1. Password value of 17 characters

2. Valid values for First Name, Last Name, Email and Phone Number (i.e., inputs filled in)

Steps to Execute:

Condition 1:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email.
5. Input a password 7 characters long.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Condition 2:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email.
5. Input a password 8 characters long.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Condition 3:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email.
5. Input a password 12 characters long.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Condition 4:

1. Go to the Register page

2. Input a First Name
3. Input a Last Name
4. Input an email.
5. Input a password 16 characters long.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Condition 5:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email.
5. Input a password 17 characters long.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Expected Results:

Condition 1: The user is prompted to make the password between 8 and 16 characters inclusive.
Condition 2: The user is successfully redirected to the Login page and Registration has occurred.
Condition 3: The user is successfully redirected to the Login page and Registration has occurred.
Condition 4: The user is successfully redirected to the Login page and Registration has occurred.
Condition 5: The user is prompted to make the password between 8 and 16 characters inclusive.

Test Case 5

Test Case Title: Equivalence Partitioning for Valid Email Input at Registration

Use Case Tested by the Test Case: Register an Account (not a Requirement, but a functionality)

Technique Used to generate test case: Equivalence Partitioning

Equivalence Partitioning for Valid Email Input at Registration Test Case	
Invalid	**Valid**
Email already present in System at time of registration	Email not already present in System at time of registration

Table 1: Equivalence Partitioning for Valid Email Input at Registration Table

Preconditions The user is able to access the webpage (The system is not in downtime)

Input:

Condition 1 (invalid):

1. Email which is already present in the system at the time of registration.
2. Valid values for First Name, Last Name, Password and Phone Number (i.e., inputs filled in)

Condition 2 (valid):

1. Email which is not already present in the system at the time of registration.
2. Valid values for First Name, Last Name, Password and Phone Number (i.e., inputs filled in)

Steps to Execute:

Condition 1:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name

4. Input an email which is already registered.
5. Input a password.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Condition 2:

1. Go to the Register page
2. Input a First Name
3. Input a Last Name
4. Input an email which is not already registered.
5. Input a password.
6. Input a Phone Number
7. Set Role to Patient.
8. Press the Register button.

Expected Results:

Condition 1: The user is prompted that the email already exists.
Condition 2: The user is successfully redirected to the Login page and Registration has occurred.

5.2 Traceability Matrix:

Please find the traceability matrix in the spreadsheet linked below:
https://docs.google.com/spreadsheets/d/1X3R9unRCsU7FuqHv0DBCrsJYhE2IoSLqJ78Br_v0N-Q/edit?usp=sharing

To see why some test cases are correlated to more than one use case or feature, consider for example how the preconditions to many test cases is a successful login (therefore the success of a login test case is relevant to many other use cases).

Section 6: Testing Results

Offer a summary of the testing outcomes, highlighting what test cases passed, failed, and unresolved issues.

Overall Testing Statistics:

- **Total Number of Test Cases:** 43
- **Total Passed:** 38
- **Total Failed:** 5

Failed Test Cases: The test cases that failed during this testing phase were:

- Test Case 1.6 (Add Doctor with Referral Code)
- Test Case 3.1 (Uploading File Without Proceeding)
- Test Case 3.4 (Verify File Size in Medical Record Uploading)
- Test Case 3.5 (Test Upload Limits with Maximum File Size)
- Test Case 4.4 (Verify Virtual Consultation Access)

These failures highlighted specific areas where the website did not perform exactly as expected.

Analysis of Failures:

Upon reviewing the failed test cases, it was observed that the discrepancies between expected and actual outcomes were primarily due to limitations in handling specific input scenarios or accurately reflecting system errors to the user.

Resolution Status:

As of the conclusion of this testing phase, there are no unresolved issues. All identified problems have been documented and resolved. We have identified the causes of these errors and rectified the system response to ensure it performs as expected.

Section 7: Recommendations on Software Quality

Based on the insights gathered from the recent testing phase, we have identified several recommendations to enhance the software's quality and user experience of the MedSync platform.

1. Enhanced Error Handling and User Feedback:
   - Reflecting Errors to Users: a lot of the times we found out that the system correctly handles an error from the backend but does not reflect doing so to the user on the frontend. Therefore, it is crucial that the system not only handle errors effectively on the backend but also communicate them clearly to the users. When an operation fails, the user interface should display understandable error messages that guide users on how to proceed and how to rectify the error.

2. Diagnostic Support Enhancements: Testing also revealed areas of improvement for our platform even if not related to errors per se. For example, in cases where the diagnostic support system fails to make a diagnosis due to unrelated symptoms, we recognize that there should be explicit feedback to the doctor that no valid diagnosis could be made, and not just provide no output.

3. Feature Extensions for Improved Usability: Testing the features also revealed more room for our website to grow to become more usable and robust. For example,

   - Appointment Management: We can introduce functionality for canceling and rescheduling appointments. This flexibility is important for enhancing user satisfaction and accommodating the dynamic nature of medical needs.

   - Medical Record and Prescription Management: Similarly, we can add features to allow users to delete medical records and prescriptions when necessary. This capability is important for maintaining user autonomy over their data and ensuring the system adapts to real-world use cases where changes are common.

4. Reliability through Internal API Development: Through testing, we also recognized that relying on external APIs can introduce risks related to availability and changes in third-party service policies. It could therefore improve the software quality to actually implement our own API for intelligent diagnosis, for instance.

In the future, possibly implementing these changes to MedSync can ensure a more robust, intuitive, and reliable platform that better serves its users' needs.

References

Bach, J. (2003). *Exploratory Testing Explained* (Version 1.3) [PDF file]. Satisfice, Inc. Retrieved from https://satisfice.us/articles/et-article.pdf

Gao, J., Tsai, H., & Wu, Y. (2003). *Testing and quality assurance for component-based software*. Artech House.