Ahmed Ghosn, Amal Al Hawry, Dema Ben Jabr, Julio Costa Sanabria

Phase Three

**Use Cases:** Appointment booking, Admin Dashboard, Patient Registration

## 2. Testing strategies

Because our application is intended to be used in safety-critical healthcare settings (clinics, hospitals, etc.), the standards for testing must be considerably high and rigorous enough to eliminate any risk of defects. The testing strategy must be robust and thorough in order to ensure the application is reliable, secure, and meets user requirements

As such, our overall strategy will focus heavily on the system requirements to develop test cases (requirement-based testing). White box testing will combine automated and manual tests to ensure the correctness of the code and internal logic, as well as compliance with safety standards. Functional and non-functional testing techniques will be carried out to verify that each component of the system meets the appropriate standards on its own (as a component), as an integrated part of the system, as an evolving system. These include a combination of unit testing, integration testing, performance testing, security testing, usability testing, regression testing, acceptance testing, exploratory testing, and continuous testing.

## 3. Test cases based on functional & non-functional goals

Each test case: TYPE + PURPOSE+DESC/METHOD+INPUT+EXP OUTPUT

a. *Appointment booking:*
    1. User Authentication/verification (functionality) - to ensure patients are able to successfully register into the system
       Method: attempt to register a new patient with valid information.
       Input Data: Patient name, contact information, medical history.
       Expected Output: Successful registration, generation of unique patient ID
    2. Appointment scheduling (functionality) - verify that the system allows registered patients to schedule appointments
       Method: schedule an appointment with a specific healthcare provider for a specific, registered patient
       Input Data: Patient ID, preferred date and time, provider.
       Expected Output: Successful scheduling, confirmation of the appointment (message on screen).
    3. Patient data confidentiality (security) - Ensure that patient data is confidential and only accessible by authorized personnel.
       Method: Attempt to access patient records without proper authorization.
       Input Data: Unauthorized user credentials.
       Expected Output: Access denied with appropriate security notification.

4. User interface navigation (usability) - Assess the ease of use of the appointment booking interface.
   Method: Navigate through the appointment booking process and assess the intuitiveness (this test case could include potential/target clients)..
   Input Data: Patient login credentials.
   Expected Output: Smooth navigation with clear instructions and minimal user friction.
5. Rescheduling an appointment (functionality) - verify that patients can successfully reschedule appointments when necessary.
   Method: Attempt to reschedule an existing appointment.
   Input Data: Patient ID, current appointment details, new date and time.
   Expected Output: Successful rescheduling of the appointment.
6. Canceling an appointment (functionality) - verify patients can cancel existing appointments.
   Method: Attempt to cancel an existing appointment for a specific patient.
   Input Data: Patient ID, appointment details.
   Expected Output: Successful cancellation of the appointment.
7. End user access control (security) - Ensure patients are only authorized to access their own account details/information..
   Method: attempt to access patient #1 appointment records from an unauthorized account (patient #2 account).
   Input Data: Patient #2 ID, patient #1 appointment details
   Expected Output: reject request, access to patient #1 details limited to patient #1 and relevant providers and administrators.
8. Concurrent appointment bookings (performance) - Assess system performance under concurrent appointment booking scenarios.
   Method: Simulate multiple users attempting to book appointments simultaneously.
   Input Data: Multiple patient requests for appointment booking.
   Expected Output: Stable system performance without significant delays or errors.
9. Screen reader compatibility (accessibility) - Ensure that the appointment booking system is accessible to users with visual impairments.
   Method: Use a screen reader to navigate and interact with the system.
   Input Data: Screen reader input commands.
   Expected Output: Screen reader provides accurate and meaningful information
10. Duplicate appointment prevention (security) - Prevent the creation of duplicate appointments for the same patient.
    Method: Attempt to book an appointment for a patient who already has a scheduled appointment at the same time.
    Input Data: Patient ID, existing appointment details.

Expected Output: System prevents the creation of duplicate appointments.

11. Appointment notification reminders (functionality) - Verify that patients receive reminders for upcoming appointments.
    Method: Schedule an appointment and wait for the reminder notification.
    Input Data: Patient ID, appointment details.
    Expected Output: Patient receives a timely reminder before the scheduled appointment
12. Clear error messages (usability) - Assess the clarity of error messages presented to users.
    Method: Intentionally enter incorrect information during the appointment booking process.
    Input Data: Invalid patient details or conflicting appointment data.
    Expected Output: Clear and informative error messages guide the user on correcting input issues.
13. System response time (performance) - Measure the system's response time during appointment booking.
    Method: Record the time taken to complete the appointment booking process.
    Input Data: Patient ID, appointment details.
    Expected Output: System responds within an acceptable time frame
14. Emergency appointment booking (functionality) - Verify that the system supports the booking of emergency appointments.
    Method: Attempt to book an appointment marked as an emergency.
    Input Data: emergency flag, patient details.
    Expected Output: system prioritizes and processes emergency appointments appropriately
15. Session timeout handling (security) - Ensure that the system handles session timeouts securely.
    Method: allow user session to timeout and attempt to resume an activity.
    Input Data: expired session.
    Expected Output: System prompts for re-authentication and prevents unauthorized access.

b. *Administrator's dashboard:*
   1. User creation (functionality) - verify that administrators can create new user accounts (mainly patients and providers).
      a. Method: Attempt to create a new user with valid information.
      b. Input Data: User details (username, password, role).
      c. Expected Output: Successful creation of a new user account.
   2. User deactivation (functionality) - ensure that administrators can deactivate user accounts.
      a. Method: Deactivate an existing user account.

    b. Input Data: User ID of the account to be deactivated.
    c. Expected Output: Successful deactivation of the user account.
3. <u>Configuration of system settings (functionality)</u> - verify that administrators can configure system settings.
    a. Method: Adjust system settings such as appointment duration or notification preferences.
    b. Input Data: Modified system settings.
    c. Expected Output: System reflects the updated configuration.
4. <u>Restrict user access control (security)</u> - confirm that users can only access functions relevant to their role.
    a. Method: Attempt to access functions outside the scope of the administrator's role.
    b. Input Data: Administrator login credentials.
    c. Expected Output: Access denied for unauthorized functions.
5. <u>Ease of dashboard navigation (usability)</u> - assess the ease of navigation within the administrator's dashboard.
    a. Method: Navigate through different sections of the dashboard.
    b. Input Data: Administrator login credentials.
    c. Expected Output: Smooth navigation with clear menu structures.
6. <u>User creation speed (performance)</u> - measure the time it takes to create a new user account.
    a. Method: Record the time taken to complete the user creation process.
    b. Input Data: User details for a new account.
    c. Expected Output: User account created within an acceptable time frame.
7. <u>User management at scale (scalability)</u> - evaluate system performance with a large number of user accounts.
    a. Method: Create and manage a significant number of user accounts.
    b. Input Data: Multiple user account details.
    c. Expected Output: System maintains responsiveness and functionality with a scalable number of users.
8. <u>Enforce password policy (security)</u> - ensure that the system enforces password policies.
    a. Method: Attempt to set a password that violates the defined policy.
    b. Input Data: Non-compliant password.
    c. Expected Output: System rejects non-compliant passwords and enforces policy.
9. <u>Dashboard loading time (performance)</u> - measure the time it takes for the administrator's dashboard to load
    a. Method: Access the dashboard and record the loading time.
    b. Input Data: Administrator login credentials.

    c. Expected Output: Dashboard loads within an acceptable timeframe.

10. <u>Concurrent user access (scalability)</u> - assess system performance with multiple administrators accessing the dashboard simultaneously.
    a. Method: Simulate concurrent access to the dashboard by multiple administrators.
    b. Input Data: Multiple administrator login credentials.
    c. Expected Output: System maintains responsiveness under concurrent user access.
11. <u>Session timeout handling (security)</u> - ensure that the system handles session timeouts securely.
    a. Method: Allow an administrator session to timeout and attempt to resume an activity.
    b. Input Data: Expired session.
    c. Expected Output: System prompts for re authentication and prevents unauthorized access.
12. <u>Role-based access control (security)</u> - ensure administrators can assign roles with specific access permissions.
    a. Method: Assign different roles to user accounts and verify access privileges.
    b. Input Data: User IDs and assigned roles.
    c. Expected Output: Users have access only to functions aligned with their assigned roles.
13. <u>System settings modification (usability)</u> - evaluate the ease of modifying system settings.
    a. Method: Modify various system settings and assess the process.
    b. Input Data: Adjusted system settings.
    c. Expected Output: Smooth modification of system settings with clear feedback.
14. <u>Logging and auditing (security)</u> - verify that the system logs and audits administrator activities.
    a. Method: Perform various administrative actions and review system logs.
    b. Input Data: Administrator actions.
    c. Expected Output: Comprehensive logs capturing admin activities.
15. <u>Resource utilization monitoring (performance)</u> - monitor system resource utilization during peak usage times.
    a. Method: Simulate peak usage by performing various tasks simultaneously.
    b. Input Data: Concurrent administrator activities.
    c. Expected Output: System maintains stable performance without resource exhaustion.

c. *Patient registration:*

1. <u>User registration (functionality)</u> - verify that patients can successfully register in the system.
    a. Method: Complete the user registration form with valid information.
    b. Input Data: Patient name, contact details, medical history.
    c. Expected Output: Successful registration and account creation.
2. <u>Prevention of duplicate registrations (functionality)</u> - ensure that duplicate accounts cannot be created for the same patient.
    a. Method: Attempt to register a patient with existing details.
    b. Input Data: Patient details already in the system.
    c. Expected Output: System prevents the creation of duplicate accounts.
3. <u>Password encryption (security)</u> - confirm that patient passwords are securely encrypted.
    a. Method: Register a new patient and examine the stored password.
    b. Input Data: New patient registration.
    c. Expected Output: Password stored securely with encryption.
4. <u>User friendly registration form (usability)</u> - evaluate the usability of the patient registration form.
    a. Method: Assess the clarity and intuitiveness of the registration form.
    b. Input Data: Patient registration details.
    c. Expected Output: User-friendly registration form with clear instructions.
5. <u>Registration form response time (performance)</u> - measure the time it takes for the registration form to load and submit.
    a. Method: Record the time taken to complete the registration process.
    b. Input Data: Patient registration details.
    c. Expected Output: Form loads and processes within an acceptable time frame (ideally two seconds).
6. <u>Account verification process (security)</u> - verify the effectiveness of the account verification process.
    a. Method: Register a new patient and undergo the account verification process.
    b. Input Data: Verification link or code.
    c. Expected Output: Successful verification and activation of the patient account.
7. <u>Account verification communication clarity (usability)</u> - assess the clarity and informativeness of account verification communication.
    a. Method: Verify the communication received after registration for clarity and instructions.
    b. Input Data: Verification email or message.

    c. Expected Output: Clear instructions and confirmation of account verification.
8. Account verification time (performance) - measure the time it takes for the account verification process.
    a. Method: Record the time taken from receiving verification communication to completing the process.
    b. Input Data: Verification link or code.
    c. Expected Output: Verification process completes within 5 seconds.
9. Consent form confidentiality (security) - ensure the confidentiality of patient consent forms.
    a. Method: Complete and submit a consent form and review system logs.
    b. Input Data: Consent form details.
    c. Expected Output: Consent form information stored securely with restricted access.
10. Consent form submission (functionality) - confirm that patients can submit consent forms successfully.
    a. Method: Complete and submit a consent form.
    b. Input Data: Patient consent details.
    c. Expected Output: Successful submission and storage of the consent form.
11. Consent form clarity (usability) - evaluate the clarity and readability of the consent form.
    a. Method: Assess the content and layout of the consent form.
    b. Input Data: Patient reviewing the consent form.
    c. Expected Output: Clear and easily understandable consent form.
12. Consent form submission time (performance) - measure the time it takes for a patient to submit a consent form.
    a. Method: Record the time taken to complete and submit the consent form.
    b. Input Data: Patient consent details.
    c. Expected Output: Consent form submission completes within an acceptable timeframe.
13. Session timeout handling (security)- ensure that the system handles session timeouts securely during the registration process.
    a. Method: Allow a registration session to timeout and attempt to resume the process.
    b. Input Data: Expired registration session.
    c. Expected Output: System prompts for re-authentication and prevents unauthorized access.
14. Error handling during registration (usability) - assess the clarity of error messages during the registration process.

   a. Method: Intentionally enter incorrect information and review error messages.
   b. Input Data: Invalid patient registration details.
   c. Expected Output: Clear and informative error messages guide the user in correcting input issues.

15. System responsiveness during peak registration (performance) - evaluate system responsiveness during peak times of patient registration.
    a. Method: Simulate a high volume of simultaneous registration attempts.
    b. Input Data: Multiple patient registration attempts.
    c. Expected Output: System maintains stability and responsiveness under peak registration loads.

**4. Tools for testing**

Key considerations when selecting these tools include compatibility with the technology stack, ease of integration, community support, prioritization of test automation for critical scenarios, seamless integration into the CI/CD pipeline, scalability to accommodate growth, and alignment with budget constraints. Careful consideration and use of the ideal tools, will give us a testing process that is robust, thorough, and effective.

For unit testing, we'll use JUnit to verify each unit of code. Integration testing, crucial for validating component interactions, can be efficiently performed using Postman for API testing. Functional testing can be addressed with Selenium WebDriver for web applications and Appium for mobile applications, depending on whether the application is expanded to the web. Performance testing, vital for evaluating responsiveness and scalability, will be done using Apache JMeter. Since this is a safety critical healthcare system, security testing will be important and for this we can use OWASP ZAP. Usability testing is facilitated by user testing platforms like UsabilityHub, and regression testing can be efficiently conducted with Selenium WebDriver as well. Cucumber can be used for acceptance testing. Continuous testing, integrated into the CI/CD pipeline, can be managed using Jenkins. Database testing can use DbUnit for ensuring data integrity. API testing can also be conducted using Postman. Accessibility testing can be facilitated by tools like Axe. For mobile application testing, we will use Appium. Collaboration and communication tools like Discord, Google workspace, and Github will also be used to allow us to communicate as a team.