K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Batch: D-2	**Roll No.:** 16010122151
Experiment / assignment / tutorial No :- 07	

Title: Designing test plan document for Mini Project

Aim: To learn and understand the way of developing the software by classical methods of software engineering. Planning and monitoring, testing, validating of the project using tools and prepares a document for the same by using the concept of software engineering

CO:

Books/ Journals/ Websites referred:

1. Roger Pressman, Software Engineering: A practitioners Approach, McGraq Hill, 2010 ,6th edition
2. Ian Somerville , Software Engineering , Addison Wesley,2011,9th edition
3 http://en.wikipedia.org/wiki/Software_requirements_specification

Test Plan Template:

HealthCare Management System

Prepared by:

Nikhil Patil [16010122136]

Hyder Presswala [16010122151]

Date:- 31-10-2024

TABLE OF CONTENTS

17.0 Approvals

1.0 INTRODUCTION

The HealthCare Management System (HCMS) is designed to improve healthcare accessibility for underserved populations, including the mentally challenged and the elderly. This system aims to streamline healthcare management by offering a user-friendly platform that facilitates medication tracking, appointment management, health insurance access, and medication purchasing. The core functions of HealthCare Management System include:-

High-Level Functions

1. **Track Medication Usage**
    - Users can monitor their medication schedules, including reminders for when to take medications and dosage amounts.
2. **Manage Appointments**
    - The system allows users to keep track of upcoming doctor's appointments and schedule new ones with healthcare providers.
3. **Access Health Insurance**
    - Users can view and manage their health insurance information, including policy details and coverage options.
4. **Book Appointments**
    - The application enables users to schedule and confirm appointments with healthcare providers directly.
5. **Purchase Medications**
    - Users can buy medications through integrated partnerships with pharmaceutical companies and medical suppliers, facilitating easy access to necessary drugs.

.

2.0 OBJECTIVES AND TASKS

2.1 Objectives

Objectives of the Master Test Plan

1. **Define Testing Scope and Approach**
    - Clearly outline the testing scope, methodologies, and strategies to be used for validating the functionality of the HealthCare Management System.
2. **Assign Tasks and Responsibilities**
    - Establish roles and responsibilities for all team members involved in the testing process, ensuring accountability and clear ownership of tasks.
3. **Facilitate Communication**

   - Serve as a communication vehicle among stakeholders, including developers, testers, project managers, and end-users, to ensure alignment and transparency throughout the testing phases.
4. **Set Quality Standards**
   - Define the quality criteria and acceptance standards that the HealthCare Management System must meet, providing a benchmark for success.
5. **Track Progress and Issues**
   - Provide a framework for tracking testing progress, managing issues, and reporting on the status of testing activities, ensuring that potential risks are identified and addressed promptly.
6. **Document Test Cases and Results**
   - Serve as a formal document to outline test cases, expected outcomes, and actual results, facilitating thorough documentation for review and compliance.
7. **Support Service Level Agreements (SLAs)**
   - Act as a reference document for SLAs, detailing the testing deliverables, timelines, and quality expectations that must be met by the development and testing teams.
8. **Enhance User Satisfaction**
   - Ensure that the system meets user needs and requirements by validating that all functionalities perform as expected, ultimately contributing to improved user satisfaction and accessibility.
9. **Continuous Improvement**
   - Provide insights and feedback for future enhancements, helping to refine testing processes and improve the overall quality of the HealthCare Management System in subsequent iterations.

2.2 Tasks

Test Planning

- Develop a comprehensive test strategy and outline testing objectives, scope, and timelines.

Test Case Development

- Design and document detailed test cases for all system functionalities, including medication tracking, appointment management, and insurance access.

Test Environment Setup

- Prepare and configure the testing environment, including necessary hardware, software, and test data.

Functional Testing

- Execute test cases to validate the functionality of the HealthCare Management System against specified requirements.

 Usability Testing

- Assess the user interface and overall user experience to ensure the system is intuitive and accessible for target populations.

 Integration Testing

- Verify that all components of the system work together seamlessly, including integrations with third-party services for medication purchasing and appointment scheduling.

 Performance Testing

- Conduct load and stress testing to evaluate the system's performance under various conditions and ensure it can handle expected user volumes.

 Security Testing

- Assess the system for vulnerabilities, ensuring that user data is protected and compliance with relevant regulations is maintained.

 User Acceptance Testing (UAT)

- Facilitate testing by actual users from the target populations to validate that the system meets their needs and expectations.

 Bug Reporting and Tracking

- Document and report any identified defects, categorizing them based on severity and impact, and track their resolution.

 Regression Testing

- Perform regression tests after any bug fixes or updates to ensure that existing functionalities remain unaffected.

 Post-Testing Review

- Conduct a review session to evaluate the testing process, document lessons learned, and gather feedback for future improvements.

 Documentation and Reporting

- Compile and distribute test reports summarizing testing activities, results, and any outstanding issues to stakeholders.

 Training and Support

- Provide training materials and support for end-users to help them navigate the system effectively once it is deployed

3.0 SCOPE

General

The scope of the testing for the HealthCare Management System encompasses all functionalities and features designed to enhance healthcare accessibility for underserved populations, including the mentally challenged and the elderly. The following components will be included in the testing process:

1. **Medication Tracking**
   - Verification of features that allow users to monitor medication schedules, including reminders for dosage times and amounts.
2. **Appointment Management**
   - Testing the functionality for managing upcoming doctor's appointments, scheduling new appointments, and sending notifications.
3. **Health Insurance Access**
   - Assessment of the system's ability to display and manage health insurance information, including policy details and coverage options.
4. **Appointment Booking**
   - Evaluation of the booking process for scheduling and confirming appointments with healthcare providers directly through the application.
5. **Medication Purchasing**
   - Testing the integration with pharmaceutical companies and medical suppliers for the purchasing process, including transaction handling and inventory management.
6. **User Interface (UI) and User Experience (UX)**
   - Review of the overall design and usability of the application to ensure it is user-friendly, particularly for the target populations.
7. **Integration Testing**
   - Validation of the integration between the HealthCare Management System and third-party services, ensuring seamless communication and functionality.
8. **Security and Compliance**
   - Assessment of security measures in place to protect user data and ensure compliance with relevant healthcare regulations and standards.
9. **Performance and Load Testing**
   - Evaluation of the system's performance under various loads to ensure it can handle expected user volumes and maintain responsiveness.

10. **Accessibility Testing**
    - Ensuring that the application is accessible to all users, including those with disabilities, and complies with accessibility standards.

1. **Tactics**

   Medication Tracking

    - **Procedure**: Develop and execute specific test cases that simulate various user scenarios for tracking medication schedules. Involve key stakeholders to provide feedback on reminders and notifications. Schedule dedicated sessions to review test results and make necessary adjustments.
2. **Appointment Management**
    - **Procedure**: Collaborate with stakeholders to create test scenarios for managing appointments. Utilize a shared calendar tool to coordinate time slots for testing, ensuring that all necessary personnel are available to observe and validate functionality.
3. **Health Insurance Access**
    - **Procedure**: Work with the health insurance team to access real policy data for testing. Set up workshops to demonstrate the functionality and gather insights, ensuring that key representatives are available for feedback and clarification.
4. **Appointment Booking**
    - **Procedure**: Conduct user acceptance testing (UAT) sessions with actual users from target populations. Schedule these sessions well in advance, allowing participants to provide feedback on the booking process. Document findings for any necessary system improvements.
5. **Medication Purchasing**
    - **Procedure**: Establish connections with pharmaceutical partners to validate the purchasing process. Create a test environment that simulates real transactions and involve representatives from the partner organizations during testing to confirm data accuracy.
6. **User Interface (UI) and User Experience (UX)**
    - **Procedure**: Perform usability testing sessions with target users to gather qualitative feedback. Utilize surveys and direct observation methods. Schedule testing sessions that accommodate user availability and preferences.
7. **Integration Testing**
    - **Procedure**: Develop a detailed integration testing plan that outlines all interfaces between the HealthCare Management System and third-party services. Notify integration partners well in advance, allocating specific times for collaborative testing efforts.
8. **Security and Compliance**
    - **Procedure**: Conduct security assessments using tools and methodologies that simulate potential threats. Involve compliance

officers and security specialists in the testing phase to validate that all necessary regulations are being met.

9. **Performance and Load Testing**
    - **Procedure**: Utilize automated performance testing tools to simulate high-load scenarios. Schedule testing during off-peak hours to minimize disruption, and involve the development team to monitor system behavior and address issues in real time.
10. **Accessibility Testing**
    - **Procedure**: Engage users with disabilities to perform accessibility testing on the application. Use accessibility assessment tools to identify potential issues and schedule follow-up sessions to discuss findings and potential improvements.

Notification and Scheduling

- **Stakeholder Engagement**: Use email and project management tools to notify key stakeholders about upcoming testing sessions. Clearly outline the purpose of each session and its importance to the overall project.
- **Time Allocation**: Create a shared calendar or scheduling tool to allow stakeholders to block off time for their participation. Send reminders leading up to the sessions to ensure availability and preparedness.
- **Documentation and Feedback**: After each testing session, document findings and share them with stakeholders for transparency and further collaboration, fostering a culture of continuous improvement.

4.0 TESTING STRATEGY

Testing Strategy for HealthCare Management System

The testing strategy for the HealthCare Management System focuses on a comprehensive approach to ensure all major features are thoroughly validated. This includes functional, usability, integration, performance, security, and accessibility testing. The strategy is organized around key feature groups, detailing the testing approach, major activities, techniques, and tools to be employed.

1. Medication Tracking

Approach:

- Functional testing to verify that users can track their medication schedules effectively.

Major Activities:

- Develop test cases covering various scenarios (e.g., adding, editing, deleting medications).
- Execute test cases and log results.

Techniques:

- Black-box testing to assess input/output without needing knowledge of the internal code.

Tools:

- Test management software (e.g., TestRail) for tracking test cases and results.
- Automation tools (e.g., Selenium) for regression testing.

Estimated Time: 2 weeks

---

2. Appointment Management

Approach:

- Comprehensive functional and usability testing to ensure users can manage appointments seamlessly.

Major Activities:

- Create test cases for scheduling, modifying, and canceling appointments.
- Conduct usability testing sessions with actual users.

Techniques:

- Exploratory testing to uncover any usability issues during real-world scenarios.

Tools:

- Usability testing tools (e.g., UserTesting) for gathering user feedback.
- Scheduling tools for managing test sessions.

Estimated Time: 3 weeks

---

3. Health Insurance Access

Approach:

- Functional and integration testing to validate the display and management of health insurance information.

Major Activities:

- Develop test cases for viewing and updating insurance details.
- Integrate with insurance databases for end-to-end testing.

Techniques:

- API testing to verify data retrieval from external systems.

Tools:

- Postman for API testing.
- Integration testing tools (e.g., SoapUI).

Estimated Time: 2 weeks

---

4. Appointment Booking

Approach:

- User acceptance testing (UAT) to validate the booking functionality from the end-user perspective.

Major Activities:

- Organize UAT sessions with target users.
- Document feedback and issues encountered during testing.

Techniques:

- Scenario-based testing to reflect real user workflows.

Tools:

- Collaboration tools (e.g., Zoom) for conducting remote UAT sessions.
- Feedback collection tools (e.g., Google Forms).

Estimated Time: 2 weeks

---

5. Medication Purchasing

Approach:

- Functional and integration testing to ensure seamless purchasing from pharmaceutical partners.

Major Activities:

- Create test cases for the purchasing process, including payment processing.
- Verify integration with third-party systems.

Techniques:

- End-to-end testing to ensure all components work together.

Tools:

- Payment gateway simulators for testing transactions.
- Integration testing frameworks (e.g., JUnit).

Estimated Time: 3 weeks

---

6. User Interface (UI) and User Experience (UX)

Approach:

- Usability and accessibility testing to ensure the system is user-friendly and compliant.

Major Activities:

- Conduct usability testing with target users and gather feedback.
- Perform accessibility audits against standards (e.g., WCAG).

Techniques:

- Heuristic evaluation to identify usability issues.

Tools:

- Accessibility testing tools (e.g., Axe).
- User feedback tools (e.g., UsabilityHub).

Estimated Time: 3 weeks

---

7. Integration Testing

Approach:

- Comprehensive testing of integrations with third-party services and systems.

Major Activities:

- Develop integration test cases covering all interfaces.
- Execute tests to ensure data flows correctly between systems.

Techniques:

- Top-down and bottom-up integration testing.

Tools:

- Integration testing tools (e.g., Postman, SoapUI).

Estimated Time: 2 weeks

---

8. Performance Testing

Approach:

- Load and stress testing to evaluate system performance under various conditions.

Major Activities:

- Create performance test scenarios to simulate real-world load.
- Monitor system behavior during testing.

Techniques:

- Load testing to assess how the system handles concurrent users.

Tools:

- Performance testing tools (e.g., JMeter, LoadRunner).

Estimated Time: 2 weeks

---

9. Security Testing

Approach:

- Security assessments to identify vulnerabilities and ensure data protection.

Major Activities:

- Conduct penetration testing to uncover potential security issues.
- Review compliance with regulations.

Techniques:

- Static and dynamic analysis for code security.

Tools:

- Security testing tools (e.g., OWASP ZAP, Burp Suite).

Estimated Time: 2 weeks

---

10. Accessibility Testing

Approach:

- Testing to ensure the application is usable by individuals with disabilities.

Major Activities:

- Conduct tests with assistive technologies (e.g., screen readers).
- Evaluate compliance with accessibility standards.

Techniques:

- Manual and automated accessibility testing.

Tools:

- Accessibility assessment tools (e.g., WAVE).

Estimated Time: 1 week

Summary of Estimated Time

- **Total Estimated Time**: Approximately 20 weeks

4.1 Unit Testing

Definition:- Unit testing validates individual code units within the HealthCare Management System, ensuring they perform their intended tasks correctly. This process aims for comprehensive coverage of each module, such as medication tracking and appointment management, minimizing errors and ensuring reliability. Additionally, it provides thorough tracing of requirements to confirm that all specifications are met, ultimately enhancing the quality of the system before moving on to integration and system testing.

Participants:
Nikhil Patil
Hyder Presswala

Methodology:

- Develop test scripts with input data and expected outcomes.
- Execute unit tests systematically for each unit.
- Identify and document defects, assess their severity.
- Resolve defects, retest units for validation.
- Analyze code coverage for thorough testing.
- Maintain requirement traceability.

4.2 System and Integration Testing

Definition

System and Integration Testing is a critical phase in the software testing lifecycle where the entire HealthCare Management System is evaluated as a complete and integrated solution. This testing ensures that all components and modules work together as intended and that the system meets its specified requirements. System testing focuses on validating the end-to-end functionality, performance, and compliance of the entire system, while integration testing assesses the interactions between individual modules and external systems.

Participants

Nikhil Patil
Hyder Presswala

Methodology

- Define scope and objectives for system and integration testing.
- Create detailed test cases covering functional and non-functional requirements.
- Set up a testing environment that mirrors production.
- Execute test cases for individual components and their interactions.
- Document defects encountered, categorizing by severity and impact.
- Collaborate with the development team to resolve defects.
- Retest affected areas and perform regression testing.
- Generate reports summarizing test results and metrics.
- Conduct a final review to ensure all objectives are met.

4.3 Performance and Stress Testing

Definition:
Performance Testing evaluates the responsiveness, speed, scalability, and stability of the HealthCare Management System under various conditions. **Stress Testing**, a subset of performance testing, specifically assesses how the system behaves under extreme conditions, such as high user loads or resource depletion. The goal is to identify the system's breaking point, understand its behavior under stress, and ensure it can recover gracefully from failure.

Participants:

Nikhil Patil
Hyder Presswala

- **Methodology:**
  ⍰ Define performance and stress testing objectives, outlining specific metrics to evaluate, such as response time, throughput, and resource utilization.
- Identify key scenarios and user workflows to simulate, including high-traffic events and peak usage patterns that the system is likely to encounter.
- Set up the testing environment to closely replicate production conditions, ensuring that hardware, software, and network configurations match the live environment.

- Use performance testing tools (e.g., JMeter, LoadRunner) to create and configure test scripts that mimic real user behavior and load patterns.
- Execute performance tests to measure key metrics, such as response times under normal and peak conditions, and assess system behavior under varying loads.
- Conduct stress tests by gradually increasing the user load until the system reaches its breaking point, identifying the maximum capacity and failure points.
- Monitor system behavior during testing, capturing metrics like CPU usage, memory consumption, and network bandwidth to analyze system performance in real-time.
- Document results thoroughly, noting any bottlenecks, errors, or failures encountered during testing, and categorize them by severity and impact.
- Analyze findings to identify areas for optimization, such as code inefficiencies, database query performance, or server configuration issues.
- Provide a comprehensive summary report detailing the test results, metrics, identified issues, and recommendations for enhancing system performance and stability under load.

4.4 User Acceptance Testing

Definition:
User Acceptance Testing (UAT) is the final phase of testing conducted to ensure that the HealthCare Management System meets the business requirements and is ready for operational use. During UAT, end-users (customers) assess the system against its initial requirements, verifying that it performs as expected in real-world scenarios. This testing aims to validate usability, functionality, and overall satisfaction before deployment.

Participants:
Nikhil Patil
Hyder Presswala

- **Methodology:**
  ◻ Define UAT objectives to ensure the system meets business requirements and user needs.

- Identify key stakeholders and end-users for participation in the testing process.
- Develop UAT test cases based on user requirements and real-world scenarios.
- Set up a testing environment that mirrors production for realistic user interaction.
- Provide training and support materials to ensure users understand the system and testing process.
- Execute UAT with users performing defined test cases while capturing feedback.
- Document any issues encountered, categorizing by severity and impact.
- Collaborate with the development team to resolve identified issues.
- Analyze user feedback for patterns and assess overall satisfaction.
- Compile a final UAT report summarizing results, feedback, and recommended improvements.

4.5 Batch Testing

Definition

Batch Testing refers to the process of testing a group of functionalities or transactions within the HealthCare Management System as a single unit. This method is particularly useful for validating scenarios that involve bulk operations, such as processing multiple medication orders, managing appointments in bulk, or updating records en masse. The goal of batch testing is to ensure that the system can handle large volumes of data efficiently and accurately while maintaining performance and functionality.

Participants:

Nikhil Patil
Hyder Presswala

Methodology

- Define the objectives of Batch Testing, focusing on validating the processing of multiple transactions or data sets in one execution.
- Identify the specific batches of data or transactions to be tested, ensuring they represent real-world scenarios.

- Develop test cases that cover various batch processing scenarios, including edge cases and error conditions.
- Set up the testing environment to replicate production settings, allowing for accurate execution of batch processes.
- Execute Batch Testing by running the defined test cases, monitoring system performance and behavior during processing.
- Document any issues or anomalies encountered, categorizing them by severity and impact on batch processing.
- Collaborate with the development team to address identified defects and retest the affected batches.
- Analyze the results to evaluate the efficiency and accuracy of batch processing.
- Compile a report summarizing test outcomes, identified issues, and recommendations for optimization.
- Ensure that batch processing meets performance benchmarks and business requirements before going live.

4.6 Automated Regression Testing

Definition:
Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still works as specified in the requirements.

Participants:

Nikhil Patil
Hyder Presswala

Methodology:

- Define the objectives of Automated Regression Testing, focusing on verifying that new code changes do not adversely affect existing functionalities.
- Identify the test cases that need to be automated, prioritizing critical functionalities and frequently used features.
- Select appropriate automation tools (e.g., Selenium, TestNG) to implement the automated tests.
- Develop and script the automated test cases, ensuring they cover various scenarios, including edge cases.
- Set up a continuous integration/continuous deployment (CI/CD) pipeline to run automated regression tests regularly.
- Execute the automated regression tests whenever new code is integrated or significant changes are made.

- Monitor test results and document any failures or discrepancies, categorizing issues by severity.
- Collaborate with the development team to investigate and resolve identified defects.
- Analyze test results to ensure that the application remains stable and performs as expected after changes.
- Compile a report summarizing the outcomes of the automated regression tests, highlighting any critical issues and overall system stability.

4.7 Beta Testing

Definition:

Beta testing for the HealthCare Management System is a phase in which a pre-release version of the application is made available to a select group of end-users, including healthcare providers and patients. The primary objective is to gather real-world feedback on the system's functionalities, usability, and performance in a healthcare context. This testing phase aims to identify any remaining issues or areas for improvement, ensuring that the final product meets user expectations and effectively addresses the needs of its target audience before the official launch.

Participants:

Nikhil Patil
Hyder Presswala

Methodology:

- Define the objectives of Beta Testing, focusing on gathering user feedback and identifying issues before the official release.
- Select a group of end-users who represent the target audience for testing the system in real-world scenarios.
- Provide participants with access to the beta version of the system, along with guidelines on how to report feedback and issues.
- Monitor user interactions and collect feedback on usability, functionality, and overall satisfaction.
- Document any issues encountered by beta testers, categorizing them by severity and impact on user experience.
- Collaborate with the development team to prioritize and address critical issues raised during beta testing.
- Analyze feedback for trends and common concerns to inform potential improvements or adjustments.
- Conduct follow-up sessions with beta testers to clarify feedback and gather additional insights.

- Compile a final report summarizing test results, user feedback, identified issues, and recommendations for the official release.
- Ensure that all critical issues are resolved and necessary changes are implemented before the system goes live.

5.0 HARDWARE REQUIREMENTS

Hardware	Configuration	No. of units
Computers	Intel Core 5, 8GB RAM, 50GB Hard disk.	1

6.0 ENVIRONMENT REQUIREMENTS

A suitable browser with active internet connection

6.1 Main Frame

Specify both the necessary and desired properties of the test environment. The specification should contain the physical characteristics of the facilities, including the hardware, the communications and system software, the mode of usage (for example, stand-alone), and any other software or supplies needed to support the test. Also specify the level of security which must be provided for the test facility, system software, and proprietary components such as software, data, and hardware.

Identify special test tools needed. Identify any other testing needs (for example, publications or office space). Identify the source of all needs which are not currently available to your group.

6.2 Workstation

7.0 TEST SCHEDULE

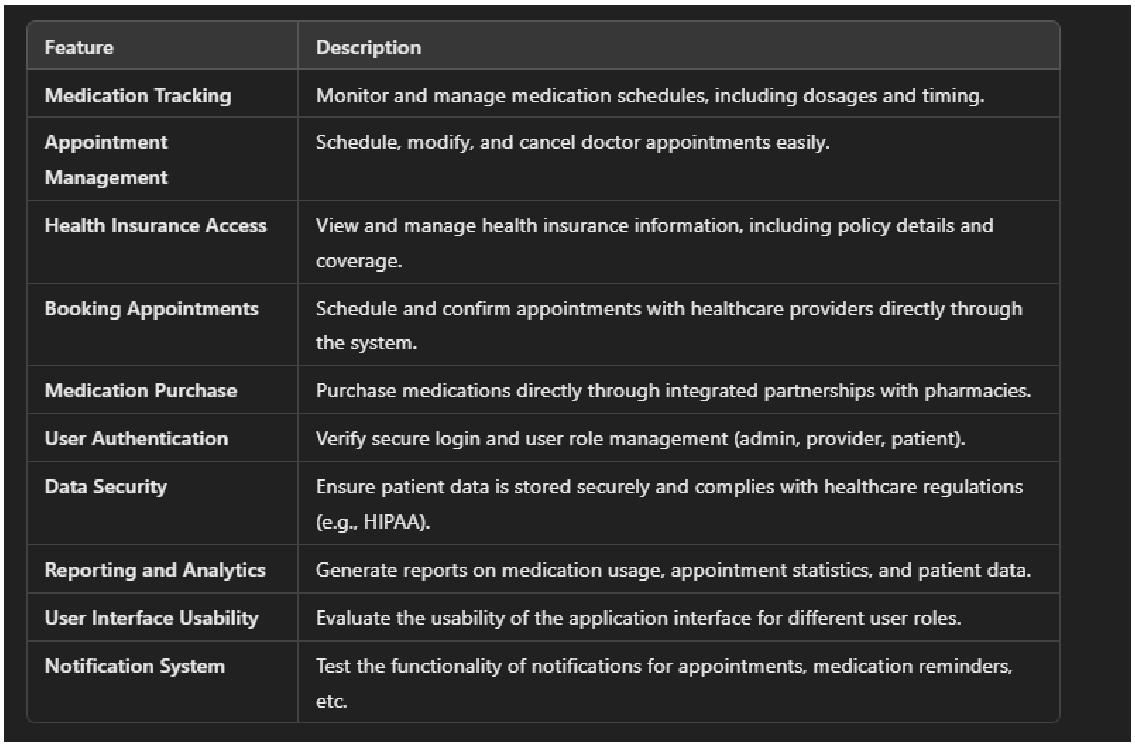
Duration	Milestone	Output
1st week	Unit testing	Documentation and feedback.
2nd week	System integration testing	Developer’s comments, automated tool reports.
3rd week	Performance testing and User Acceptance testing	User’s feedback, system performance report
4th week	Beta testing	User’s feedback

8.0 CONTROL PROCEDURES

Problem Reporting
If any problem or error has occurred during the testing, the tester can directly report the error to the developer, also the developer makes note of errors to be corrected

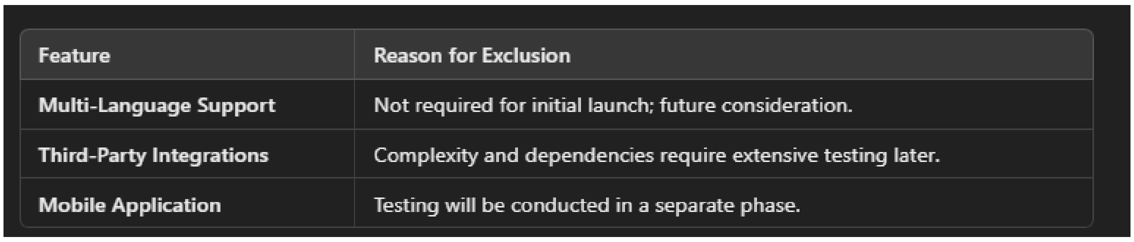
Change Requests
If any error is found by the tester, and if it can be resolved and 70% of developer teams agree to make the changes, then the changes are to be adapted in necessary modules

9.0 FEATURES TO BE TESTED



Feature	Description
Medication Tracking	Monitor and manage medication schedules, including dosages and timing.
Appointment Management	Schedule, modify, and cancel doctor appointments easily.
Health Insurance Access	View and manage health insurance information, including policy details and coverage.
Booking Appointments	Schedule and confirm appointments with healthcare providers directly through the system.
Medication Purchase	Purchase medications directly through integrated partnerships with pharmacies.
User Authentication	Verify secure login and user role management (admin, provider, patient).
Data Security	Ensure patient data is stored securely and complies with healthcare regulations (e.g., HIPAA).
Reporting and Analytics	Generate reports on medication usage, appointment statistics, and patient data.
User Interface Usability	Evaluate the usability of the application interface for different user roles.
Notification System	Test the functionality of notifications for appointments, medication reminders, etc.

10.0 FEATURES NOT TO BE TESTED



Feature	Reason for Exclusion
Multi-Language Support	Not required for initial launch; future consideration.
Third-Party Integrations	Complexity and dependencies require extensive testing later.
Mobile Application	Testing will be conducted in a separate phase.

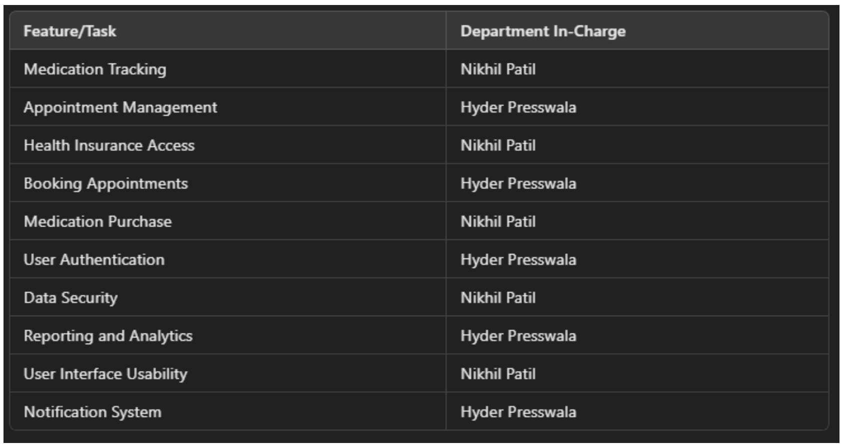
11.0 RESOURCES/ROLES & RESPONSIBILITIES

Nikhil Patil:- Preparation of Slides, resolving test cases, Managing, designing test cases.
Hyder Presswala:- Preparing, executing and resolving test cases.

12.0 SCHEDULES

Major Deliverables
Identify the deliverable documents. You can list the following documents:
– Test Plan
– Test Cases
– Test Incident Reports
– Test Summary Reports

13.0 SIGNIFICANTLY IMPACTED DEPARTMENTS (SIDs)



Feature/Task	Department In-Charge
Medication Tracking	Nikhil Patil
Appointment Management	Hyder Presswala
Health Insurance Access	Nikhil Patil
Booking Appointments	Hyder Presswala
Medication Purchase	Nikhil Patil
User Authentication	Hyder Presswala
Data Security	Nikhil Patil
Reporting and Analytics	Hyder Presswala
User Interface Usability	Nikhil Patil
Notification System	Hyder Presswala

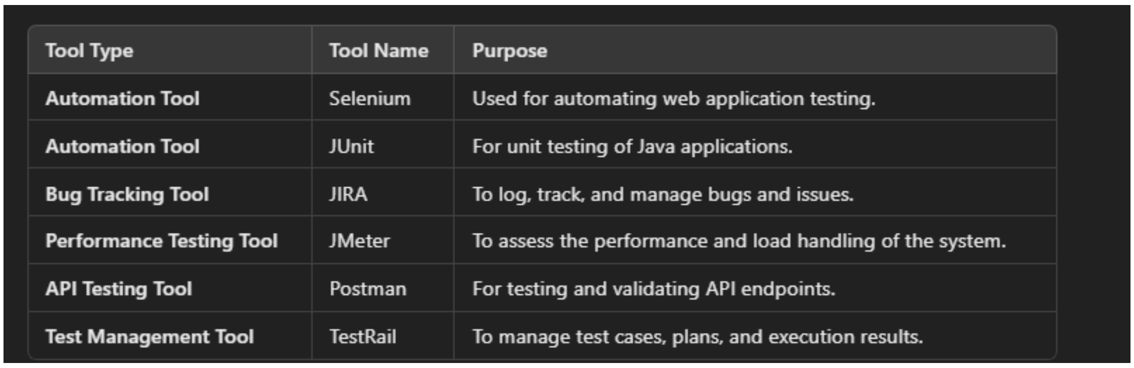
14.0 DEPENDENCIES

1. **Database Management System**: The HealthCare Management System relies on a stable database for securely storing and retrieving patient data, which is essential for all functionalities.
2. **Third-Party API Integrations**: Successful integration with external APIs for medication purchases and health insurance verification is critical, necessitating their availability and reliability for the system to function effectively.

15.0 RISKS/ASSUMPTIONS

- **Data Security Breaches**: There is a risk of unauthorized access to sensitive patient data, which could lead to data breaches and compliance violations (e.g., HIPAA). Ensuring robust security measures is essential to mitigate this risk.
- **Integration Failures**: The system's reliance on third-party APIs for medication purchases and health insurance verification poses a risk of integration failures or downtime, potentially disrupting critical functionalities and affecting user experience.
- **Insurance Data Mismanagement**: There is a risk that insurance information may be incorrectly accessed or exchanged between users, leading to potential mix-ups in patient coverage. Implementing strict access controls and verification processes is crucial to prevent such occurrences.

16.0 TOOLS



Tool Type	Tool Name	Purpose
Automation Tool	Selenium	Used for automating web application testing.
Automation Tool	JUnit	For unit testing of Java applications.
Bug Tracking Tool	JIRA	To log, track, and manage bugs and issues.
Performance Testing Tool	JMeter	To assess the performance and load handling of the system.
API Testing Tool	Postman	For testing and validating API endpoints.
Test Management Tool	TestRail	To manage test cases, plans, and execution results.

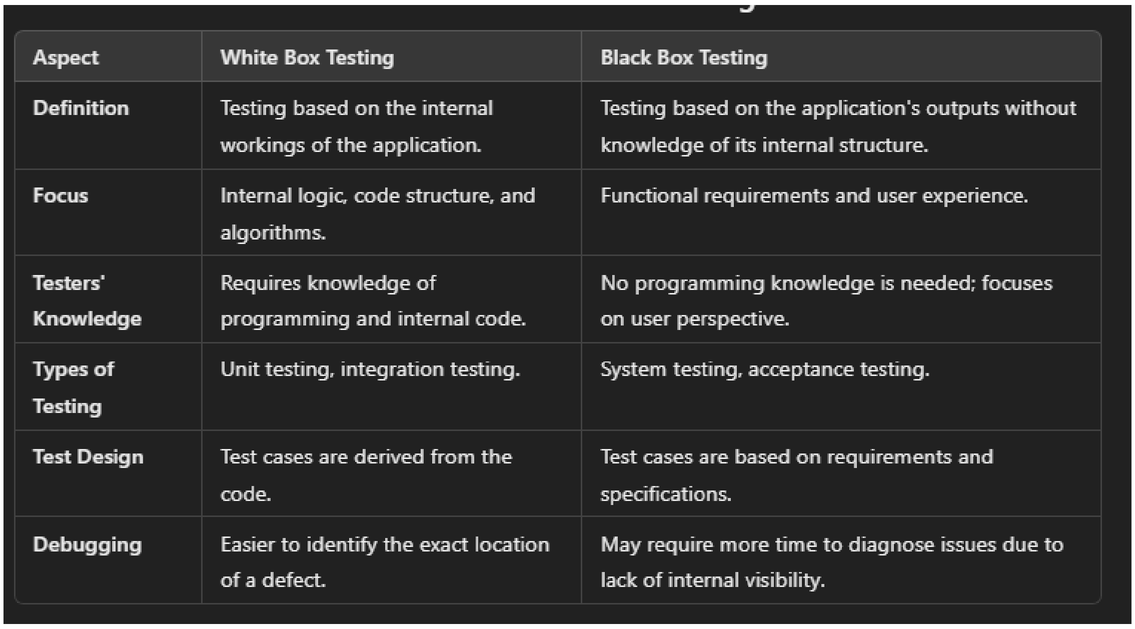
17.0 APPROVALS

Specify the names and titles of all persons who must approve this plan. Provide space for the signatures and dates.

Name (In Capital Letters) Signature Dat

Post Lab Descriptive Questions:

1. **Distinguish between Black Box and White Box Testing**



Aspect	White Box Testing	Black Box Testing
Definition	Testing based on the internal workings of the application.	Testing based on the application's outputs without knowledge of its internal structure.
Focus	Internal logic, code structure, and algorithms.	Functional requirements and user experience.
Testers' Knowledge	Requires knowledge of programming and internal code.	No programming knowledge is needed; focuses on user perspective.
Types of Testing	Unit testing, integration testing.	System testing, acceptance testing.
Test Design	Test cases are derived from the code.	Test cases are based on requirements and specifications.
Debugging	Easier to identify the exact location of a defect.	May require more time to diagnose issues due to lack of internal visibility.

2. Consider following scenario: An institute is interested in developing a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

   The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (e.g. passwords) is stored in plain text.

Assume following test suite is used

A test suite to verify the "User Login" feature	
#	**TS1**
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC1	Verify that user already registered with the LIS is able to login with correct user ID and password		Employee ID *149405* is a registered user of LIS; user's password is *this_is_password*	User is logged in	1. Type in employee ID as *149405* 2. Type in password *this_is_password* 3. Click on the 'Login' button	"Home" page for the user is displayed
TC2	Verify that an unregistered user of LIS is unable to login		Employee ID *149405xx* is not a registered user of LIS	User is not logged in	1. Type in employee ID as *149405xx* 2. Type in password *whatever* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message
TC3	Verify that user already registered with the LIS is unable to login with incorrect password		Employee ID *149405* is a registered user of LIS; user's password is *this_is_password*	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message

A test suite to verify the "User Login" feature	
#	**TS1**
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC4	Verify that user already registered with the LIS is unable to login with incorrect password given twice consecutively	TC3	This test case is executed after execution of TC3 before executing any other test case	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever2* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message
TC5	Verify that user already registered with the LIS is unable to login with incorrect password given thrice consecutively	TC4	This test case is executed after execution of TC4 before executing any other test case	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever3* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message; the security question and input box for the answer are displayed
TC6	Verify that a registered user can login after three consecutive failures by correctly answering the security question	TC5	This test case is executed after execution of TC6 before executing any other test case. Answer to the security question is *my_answer*.	Email sent containing new password. The email is expected to be received within 2 minute.	1. Type in the answer as *my_answer* 2. Click on the 'Email Password' button	Login dialog is displayed; an email containing the new password is received

A test suite to verify the "User Login" feature

#	TS1
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC7	Verify that a registered user's account is blocked after three consecutive failures and answering the security question incorrectly		Execute the test cases TC3, TC4, and TC5 once again (in order) before executing this test case	User account has been blocked	1. Type in the answer as *not_my_answer* 2. Click on the 'Email Password' button	The message *"Your account has been blocked! Please contact the administrator."* appears

create a Requirements Traceability Matrix (RTM) showing a mapping from individual requirement to test case(s).

Table 1: A simplified mapping from requirements to test cases

Requirement #	Test Case #
R1	
R2	
R3	
R4	

Consider requirements are summarized in the table below

#	Requirement
R1	New user registration
R2	User Login
R3	Search book
R4	Issue book
R5	Return book
R6	Reissue book