**User Acceptance Testing (UAT) Template**

|  |  |
| --- | --- |
| Date | 25-05-2025 |
| Team ID | LTVIP2025TMID55946 |
| Project Name | Docspot |
| Maximum Marks |  |

**Project Overview:**

**Project Name: Docspot**: Seamless Appointment Booking For Health

**Project Description**:

The **Seamless Healthcare Appointment Booking System** is a digital platform designed to streamline the entire healthcare appointment scheduling process. It provides an easy, efficient, and user-friendly interface for **patients**, **doctors**, and **administrators**. The system integrates with various functionalities such as real-time availability, calendar synchronization, patient health record management, and payment processing, ensuring a smooth and automated experience for all stakeholders.

**Testing Period**: 25-05-2025 to 30-06-2025

**Testing Scope:**

The Testing Scope defines the boundaries and areas of focus for testing throughout the development lifecycle of the Seamless Healthcare Appointment Booking System. This ensures that all components, from user interfaces to backend processes, are validated and verified to meet requirements before release.

**1 .Types of Testing**

They will undergo the following types of testing:

1. **Functional Testing**: Ensures that all features are working as expected (e.g., booking an appointment, login functionality).
2. **Non-Functional Testing**: Validates performance, security, and usability.
3. **Integration Testing**: Verifies that different modules (e.g., payment gateway, EHR integration) work seamlessly together.

**2. Testing Areas (Scope)**

**A. User Interface (UI) Testing**

* **Login/Registration**: Testing for input validation, error messages, password complexity requirements, and secure authentication.
* **Doctor Search & Filter**: Ensures the search function works for various parameters (location, specialty, availability) and results are displayed correctly.
* **Appointment Booking Flow**: Test booking, rescheduling, and canceling appointments. Check for error handling (e.g., doctor availability updates)

**Testing Environment:**

The testing environment is a controlled setup that mimics the production environment where the system is validated before going live. It helps ensure:

* Accurate testing of features and performance.
* Proper simulation of real-world scenarios (patients, doctors, admins).
* Data isolation and integrity during tests.

**Components:**

|  |  |
| --- | --- |
| **Component** | **Description** |
| Frontend Environment | Deployed UI for web and mobile platforms (React.js or React Native). |
| Backend API Server | RESTful API (Node.js, Python) handling all business logic, database access, and third-party calls. |
| Database | Test instance of PostgreSQL or MongoDB with mock patient/doctor/appointment data. |
| Authentication Server | Auth0 or OAuth 2.0 server to simulate login and session handling. |
| EHR Integration | Mock EHR interface using HL7/FHIR API simulators or sandbox APIs from real EHR providers. |
| Insurance API | Mock insurance verification API or test gateway from real providers (e.g., Availity, Change). |

**Notes:**

* Ensure that all test cases cover both positive and negative scenarios.
* Encourage testers to provide detailed feedback, including any suggestions for improvement.
* Bug tracking should include details such as severity, status, and steps to reproduce.
* Obtain sign-off from both the project manager and product owner before proceeding with deployment.