TestInsure - Hospital ERP & Insurance Management System

---

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for **TestInsure**, a comprehensive Hospital Enterprise Resource Planning (ERP) system. This document serves as a guideline for developers, testers, and stakeholders to understand the system's architecture, data flow, and core business logic.

1.2 Scope

TestInsure is a unified platform designed to bridge the gap between **Clinical Operations** (Test/Slot Management) and **Financial Operations** (Insurance Verification). The system simulates a "Cashless" healthcare environment where patient bookings are verified against real-time insurance policy limits to ensure revenue assurance for the hospital.

1.3 Definitions and Acronyms

- **ERP:** Enterprise Resource Planning.
- **TPA:** Third-Party Administrator (Insurance Desk).
- **Virtual Lock:** A mechanism to calculate effective balance by subtracting pending approvals from the total limit.
- **JWT:** JSON Web Token (used for secure authentication).

---

2. Overall Description

2.1 Product Perspective

TestInsure operates as a web-based application using a Client-Server architecture.

- **Frontend:** React.js (Vite) with Bootstrap 5.
- **Backend:** Spring Boot (Java 17) with Hibernate/JPA.
- **Database:** MySQL 8.0.

2.2 User Classes and Characteristics

1. **Patient:** A user seeking diagnostic services. They require a user-friendly interface to book tests, manage their insurance wallet, and view medical reports.

2. **Admin (Hospital Operations Manager):** A high-level user responsible for managing hospital capacity (slots/tests) and acting as the TPA desk to verify and approve insurance claims.

2.3 Operating Environment

- **Client:** Any modern web browser (Chrome, Edge, Firefox).
- **Server:** Tomcat (embedded in Spring Boot).
- **OS:** Platform Independent (Windows/Linux/MacOS).

---

3. Functional Requirements

3.1 Module: Authentication & Authorization

- **FR-01:** The system shall allow users to register as Patients using email and password.
- **FR-02:** The system shall authenticate users via **JWT (JSON Web Tokens)**.
- **FR-03:** The system shall restrict access to Admin pages to users with the role `ROLE_ADMIN`.

3.2 Module: Patient Portal

- **FR-04 (Wallet):** Patients shall be able to link an Insurance Policy by providing Provider Name, Policy Number, and Expiry Date.
- **FR-05 (Booking - Virtual Lock):** When a patient attempts to book a test via insurance:
  - The system must calculate `Pending Deductions` (Sum of all PENDING booking costs).
  - The system must validate if `(Policy Limit - Pending Deductions) >= Test Cost`.
  - If validation fails, the system must reject the booking with an "Insufficient Funds" error.
- **FR-06 (Dashboard):** Patients shall view a list of appointments with status badges (PENDING, CONFIRMED, COMPLETED, CANCELLED).
- **FR-07 (Downloads):** Patients shall be able to download payment receipts (PDF) and diagnostic reports (PDF) once generated.
- **FR-08 (Cancellation):** Patients can cancel a booking only if the status is `CONFIRMED` and the result has not yet been uploaded (`COMPLETED`).

3.3 Module: Admin Clinical Operations

- **FR-09 (Test Management):** Admin shall be able to Add, Update, or Delete diagnostic tests (Name, Cost, Prep Instructions).
- **FR-10 (Slot Management):** Admin shall create time slots for specific dates and define machine capacity (e.g., 10 slots for MRI).

- **FR-11 (Inventory Check):** The system must automatically decrease slot capacity by 1 upon a successful booking request.

3.4 Module: Admin Financial Operations (TPA Desk)

- **FR-12 (Claim Verification):** Admin shall view all PENDING insurance bookings.
- **FR-13 (Atomic Approval):** When Admin clicks "Verify Claim":
    - The system must perform a **Hard Deduction** from the Patient's Policy balance.
    - The system must update Booking Status to CONFIRMED.
    - Both actions must occur in a single Database Transaction (@Transactional).
- **FR-14 (Report Upload):** Admin shall be able to upload a PDF result for a specific booking.
- **FR-15 (Auto-Completion):** Upon report upload, the system must automatically update the booking status to COMPLETED.

---

4. Data Description (Database Schema)

The system manages the following data entities:

4.1 Users Table (users)

Field	Type	Description
user_id	BIGINT (PK)	Unique identifier
email	VARCHAR	User login email (Unique)
password	VARCHAR	BCrypt encoded password
role	VARCHAR	'PATIENT' or 'ADMIN'

4.2 Insurance Policy Table (insurance_policies)

Field	Type	Description
policy_id	BIGINT (PK)	Unique identifier
provider_name	VARCHAR	E.g., Aetna, BlueCross
coverage_amount	DOUBLE	**Current** available balance
total_limit	DOUBLE	Original max limit
user_id	FK	Links to Users table

4.3 Diagnostic Tests Table (laboratory_tests)

Field	Type	Description
test_id	BIGINT (PK)	Unique identifier
name	VARCHAR	E.g., MRI Scan, Blood Test
cost	DOUBLE	Price of the test
description	TEXT	Details and prep instructions

4.4 Bookings Table (bookings)

Field	Type	Description
booking_id	BIGINT (PK)	Unique identifier
status	ENUM	PENDING, CONFIRMED, COMPLETED, CANCELLED
payment_status	VARCHAR	PAID, PENDING, PAY_AT_COUNTER
is_insurance	BOOLEAN	True if insurance was used
user_id	FK	The patient
test_id	FK	The test booked
slot_id	FK	The time slot reserved

---

5. Non-Functional Requirements

5.1 Reliability (Data Integrity)

- **Atomic Transactions:** All financial operations (Booking deduction, Refund) must follow ACID properties to ensure no data is lost during a system failure.

5.2 Security

- **Password Storage:** All passwords must be hashed using BCrypt.
- **API Security:** All API endpoints (except Login/Register) must be protected via JWT Filters.
- **CORS:** The backend must only accept requests from the trusted Frontend origin (e.g., `localhost:5173`).

5.3 Performance

- **Response Time:** API responses should typically be under 200ms.
- **Concurrency:** The system must handle multiple users booking the same slot simultaneously without overbooking (handled via Database Locking/Transactional logic).

---

6. Interface Requirements

6.1 User Interface

- **Theme:** The system uses a "Pale Blue Enterprise" theme to convey professionalism and medical hygiene.
- **Dashboard Layout:**
    - **Patient:** Card-based layout for Wallet and Table layout for Appointments.
    - **Admin:** Split-pane layout separating "Clinical Resource Mgmt" and "Financial & Records".

6.2 Software Interfaces

- **REST API:** Communication between Frontend and Backend uses JSON over HTTP.
- **File System:** Medical reports are stored in a local secure directory (`uploads/`) and served via API streams.