Lab Inventory Tracker Project Report

1. Abstract

The Lab Inventory Tracker is a comprehensive web application developed using Oracle APEX and PL/SQL to streamline the management of laboratory items such as equipment and chemicals. It enables users to add, update, issue, return, and delete inventory items while maintaining data integrity and traceability. The system features secure role-based user authentication, audit logging for every action performed, and automated alerts for low stock via email triggers. It is designed for deployment in academic and research institutions and can be extended for mobile or enterprise use.

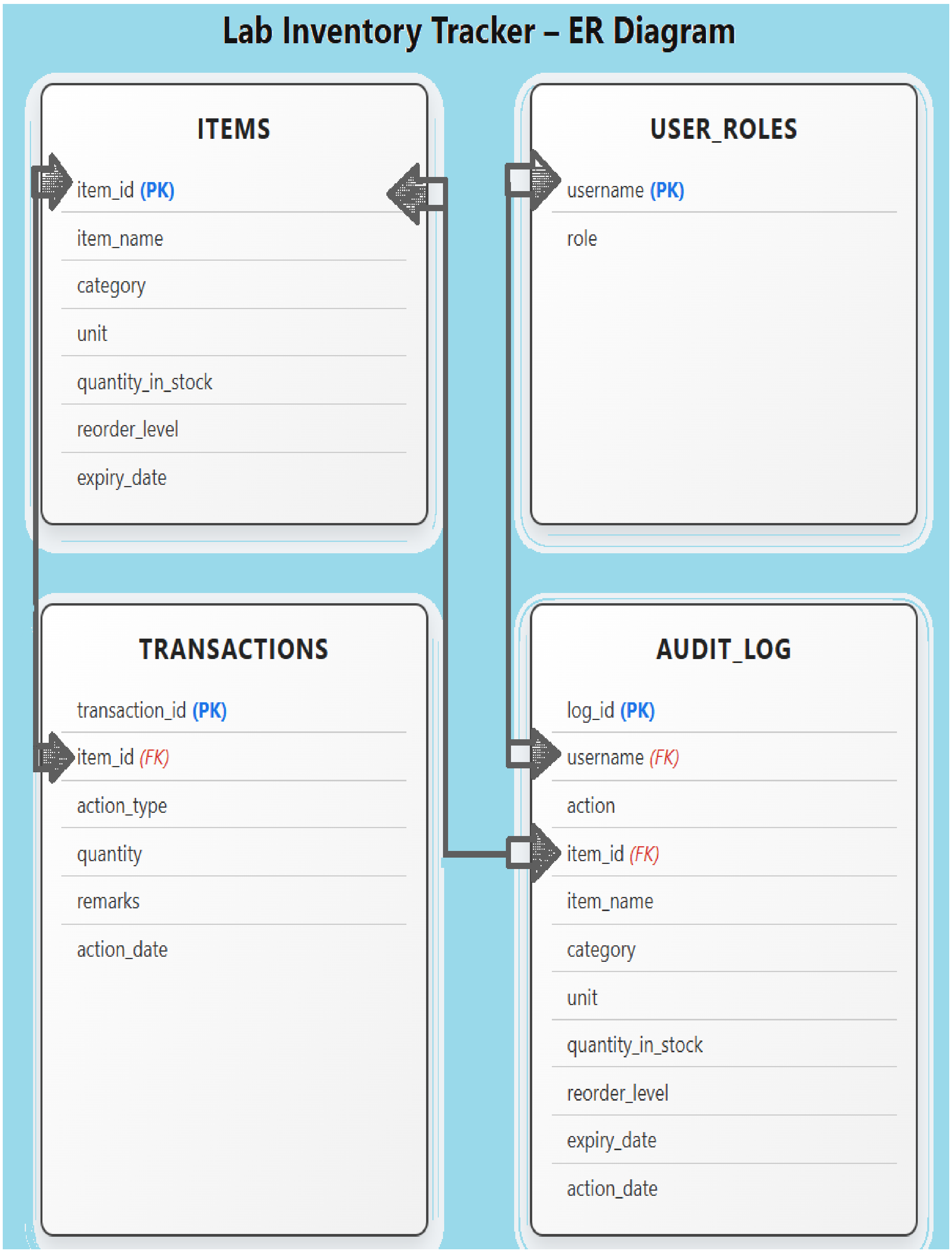
Access the Application URL: Click Here

Dummy Login Credentials Username/Password: user / 123456789

2. Modules Overview

Module	Description
Inventory Module	Manages all CRUD operations related to lab items, including tracking quantity and expiry.
Transaction Module	Logs every ISSUE and RETURN action with timestamps, item references, and remarks.
User Module	Provides login functionality and assigns roles (admin/user) to enforce access control.
Audit Log Module	Captures and records all insert, update, and delete actions performed by users.
Alerts Module	Sends email notifications when stock levels fall below defined reorder thresholds using database triggers.

3. ER Diagram Explanation



Entities and Attributes

1. items
    - item_id (Primary Key)
    - item_name
    - category
    - unit
    - quantity_in_stock
    - reorder_level
    - expiry_date
2. transactions
    - transaction_id (Primary Key)
    - item_id (Foreign Key to items)
    - action_type (ISSUE or RETURN)
    - quantity
    - remarks
    - action_date
3. user_roles
    - username (Primary Key)
    - role (admin or user)
4. audit_log
    - log_id (Primary Key)
    - username
    - action (INSERT, UPDATE, DELETE)
    - table_name

    - row_id
    - action_date

Relationships

- One item can have multiple transactions.
- Audit logs are associated with user actions across all tracked tables.
- Role-based access determines what actions users can perform within the application.

4. Technologies and Tools Used

Component	Technology or Tool
Backend Logic :	PL/SQL (procedures, functions, triggers)
Frontend Interface :	Oracle APEX (Application Builder)
Database :	Oracle 21c XE
Deployment Platform :	Oracle APEX Cloud or on-premises APEX environment
Email Notification :	APEX_MAIL or UTL_MAIL package
Security :	Oracle APEX authentication and authorization

5. Core Functionalities

Inventory Management

- Create, update, and delete items with stock quantity and expiry tracking.
- Prevent duplicate entries by checking existing item names.

Transactions

- Allow users to issue or return inventory items.
- Automatically adjust stock levels based on the action type.

- Log each transaction with user input and timestamp.

User Management

- Secure login system with admin and standard user roles.
- Admins have full access to item and user management; users are restricted to transaction activities.

Audit Logging

- Triggers capture all insert, update, and delete operations.
- Audit data includes user identity, table affected, and timestamp.
- Only administrators can view the audit log.

Email Alerts

- Stock update trigger checks if quantity falls below the reorder level.
- Automatically sends email alerts to designated users when low stock is detected.

6.Table Creation Scripts

```
-- ==================================
-- LAB INVENTORY TRACKER SQL SCRIPT
-- ==================================

-- 1. ITEMS TABLE
CREATE TABLE items (
   item_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
   item_name VARCHAR2(100),
   category VARCHAR2(50),
   unit VARCHAR2(20),
   quantity_in_stock NUMBER,
   reorder_level NUMBER,
   expiry_date DATE,
);
```

```
-- 2. TRANSACTIONS TABLE
CREATE TABLE transactions (
   transaction_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY
KEY,
   item_id NUMBER REFERENCES items(item_id),
   action_type VARCHAR2(10),
   quantity NUMBER,
   remarks VARCHAR2(200),
   action_date DATE DEFAULT SYSDATE
);

-- 3. USER ROLES TABLE
CREATE TABLE user_roles (
   username VARCHAR2(100) PRIMARY KEY,
   role VARCHAR2(10)
);

-- 4. AUDIT LOG TABLE
CREATE TABLE audit_log (
   log_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
   username VARCHAR2(100),
   action VARCHAR2(10),
   table_name VARCHAR2(50),
   row_id NUMBER,
   action_date DATE DEFAULT SYSDATE
);


-- ========================
-- TRIGGERS & PROCEDURES
-- ========================

-- Trigger: AUDIT ITEMS
CREATE OR REPLACE TRIGGER trg_audit_items
AFTER INSERT OR UPDATE OR DELETE ON items
FOR EACH ROW
BEGIN
  INSERT INTO audit_log (username, action, table_name, row_id)
  VALUES (
    NVL(V('APP_USER'), USER),
    CASE
      WHEN INSERTING THEN 'INSERT'
      WHEN UPDATING THEN 'UPDATE'
      WHEN DELETING THEN 'DELETE'
    END,
    'items',
    CASE
      WHEN INSERTING THEN :NEW.item_id
      WHEN UPDATING THEN :NEW.item_id
      WHEN DELETING THEN :OLD.item_id
    END
  );
END;
/

-- Trigger: STOCK ALERT EMAIL
CREATE OR REPLACE TRIGGER trg_stock_alert
AFTER UPDATE ON items
FOR EACH ROW
WHEN (NEW.quantity_in_stock <= NEW.reorder_level)
BEGIN
  APEX_MAIL.SEND(
    p_to        => 'anirudhsingh3019@gmail.com',
    p_from      => 'anirudhsingh3019@gmail.com',
    p_subj      => '⚠ Stock Low: ' || :NEW.item_name,
    p_body      => 'Stock for "' || :NEW.item_name || '" is now ' ||
:NEW.quantity_in_stock ||
                   '. Reorder level is ' || :NEW.reorder_level || '.'
  );
  APEX_MAIL.PUSH_QUEUE;
END;
/

-- Procedure: ADD OR UPDATE ITEM
CREATE OR REPLACE PROCEDURE add_or_update_item (
    p_item_name     IN VARCHAR2,
    p_category      IN VARCHAR2,
    p_unit          IN VARCHAR2,
    p_quantity      IN NUMBER,
    p_reorder       IN NUMBER,
    p_expiry_date   IN DATE
) AS
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM items
    WHERE LOWER(TRIM(item_name)) = LOWER(TRIM(p_item_name));

    IF v_count > 0 THEN
       UPDATE items
       SET quantity_in_stock = quantity_in_stock + p_quantity
       WHERE LOWER(TRIM(item_name)) = LOWER(TRIM(p_item_name));
    ELSE
       INSERT INTO items (item_name, category, unit, quantity_in_stock,
reorder_level, expiry_date)
       VALUES (p_item_name, p_category, p_unit, p_quantity, p_reorder,
p_expiry_date);
    END IF;
END;
/

-- Admin Authorization Scheme (for APEX)
-- Use as PL/SQL Function Returning Boolean
-- Return TRUE if user is Admin
DECLARE
    v_role user_roles.role%TYPE;
BEGIN
    SELECT role INTO v_role FROM user_roles WHERE username = :APP_USER;
    RETURN v_role = 'Admin';
EXCEPTION
    WHEN NO_DATA_FOUND THEN
       RETURN FALSE;
END;
```