



Industrial Training Project Report
On

INVENTORY MANAGEMENT SYSTEM

Submitted to

**HINDALCO Industries Limited,
Muri, Jharkhand - 835101**

BY

RISHAV PRASAD

Roll No. - 2105818

Branch - Computer Science and Engineering
College - Kalinga Institute of Industrial Technology
Bhubaneswar, Odisha - 751024

Training Period - 6st May, 2024 - 5th June, 2024



KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

CERTIFICATE

This is certify that the project entitled "**Inventory Management System**" submitted by **Rishav Prasad** of KIIT University, Bhubaneswar is a record of bonafide work carried out by him, in the partial fulfillment of the requirement for the award of Industrial Training Completion Certificate at HINDALCO INDUSTRIES LIMITED, Muri Works. This work is done during 6th May, 2024 - 5th June, 2024, under my guidance.

Under the Guidance of

MAHTAB ALAM
SENIOR OFFICER, SYSTEM
HINDALCO INDUSTRIES LTD.
PROJECT INCHARGE

ATUL KUMAR
HEAD, IT Department
HINDALCO INDUSTRIES LTD.

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all the individuals who played a crucial role in the successful completion of the Inventory Management System project during my industrial training. Their guidance, support, and contributions were invaluable throughout the entire development process.

First and foremost, I would like to extend my deepest appreciation to **Mr. Atul Kumar**, my mentor, whose expertise and guidance were instrumental in shaping the project. His insightful feedback, constant encouragement, and extensive knowledge of inventory management greatly contributed to the system's development and implementation. His dedication and support have been truly invaluable.

I would also like to extend my heartfelt thanks to **Mr. Mahtab Alam, Mr. Kumar Sudhanshu, Mr. Timothy Murmu** and the entire staff of Hindalco Industries Ltd's IT department for their continuous assistance, technical insights, and valuable suggestions throughout the project. Their expertise and collaboration significantly enhanced the functionality and efficiency of the Inventory Management System. Their contributions and willingness to share their knowledge have been instrumental in overcoming challenges and achieving the project's objectives.

I would also like to express my gratitude to **Mrs. Angela D'Vaz** for giving me this wonderful opportunity. Their prompt responses to queries and their efforts in organizing the necessary resources and logistics were greatly appreciated.

Lastly, I would like to thank my college and the training coordinators for providing me with the opportunity to undertake this industrial training and gain valuable hands-on experience in software development.

I am truly grateful to all these individuals for their unwavering support, guidance, and encouragement throughout this journey. Their contributions have not only enriched my technical skills but have also shaped me both personally and professionally.

Rishav Prasad
2105818
CSE
KIIT University

ABSTRACT

The Inventory Management System is a web-based application developed for the IT department of Hindalco Industries Ltd, Muri Works. The system aims to streamline and optimize the management of two systems, namely P&B and CapEx. By leveraging technologies such as HTML, CSS, JavaScript, Oracle Application Express, PL/SQL and MySQL, the system provides authorized users with read and write access, authenticated users with read access only, and restricts unauthenticated users from accessing the system.

This project report presents an in-depth analysis of the development and implementation of the Inventory Management System. It outlines the project objectives, provides an overview of the technical stack used, and highlights the key features of the system, including the login feature with differentiated access levels. The report also discusses the system's architecture, implementation details, and the evaluation of the project's outcomes.

Through this project, the IT department at Hindalco Industries Ltd aims to enhance inventory management processes, improve efficiency, and ensure accurate tracking and control of inventory across the P&B and CapEx systems. The project's successful implementation will contribute to streamlined operations, reduced manual efforts, and improved decision-making capabilities.

Keywords: Inventory Management System, P&B, CapEx, HTML, CSS, JavaScript, Oracle Application Express, PL/SQL, MySQL.

CONTENTS

1	Introduction	1
2	Software Requirement Specification (SRS)	2
3	Diagrams	4
	3.1 Use Case Diagram	
	3.2 Entity Relationship (ER) Diagram	
	3.3 Control Flow Diagram	
4	Source Code	6
5	Conclusion	71
6	Future Scope	72
7	References	73

INTRODUCTION

This section provides an overview of the project and sets the context for the development of Inventory Management System for the IT department at Hindalco Industries Ltd. This section also outlines the purpose of the project, the scope of work, and the significance of effective inventory management in the IT department.

1.1 Project Overview: The Inventory Management System project aims to develop a web-based application to streamline the management of two systems within Hindalco Industries Ltd's IT department: P&B (Planning & Budgeting) and CapEx (Capital Expenditure). The system leverages a technology stack consisting of HTML, CSS, JavaScript, PL/SQL, and MySQL to provide efficient tracking, control, and reporting of inventory-related activities.

1.2 Purpose: The purpose of this project is to address the challenges faced by the IT department at Hindalco Industries Ltd in managing inventory across various systems. The Inventory Management System will provide authorized users with read and write access, authenticated users with read access only, and restrict unauthenticated users from accessing the system. By implementing this system, the IT department aims to enhance inventory control, reduce manual efforts, and ensure accurate and efficient inventory management.

1.3 Scope: The scope of the project is focused on the development of a web application specifically tailored for the IT department's inventory management needs. The system will allow users to perform tasks such as adding and updating inventory items, generating reports, and tracking inventory movement. The project encompasses the design, development, testing, and implementation of the Inventory Management System.

1.4 Significance: Effective inventory management is crucial for the smooth functioning of any organization, and it holds particular importance in the IT department. Proper inventory control ensures the availability of necessary equipment, reduces operational costs, minimizes downtime, and enables efficient planning and budgeting. The Inventory Management System will enable the IT department at Hindalco Industries Ltd to optimize their inventory management processes, improve productivity, and make informed decisions based on accurate and up-to-date inventory data.

Software Requirement Specification (SRS)

Introduction

Purpose: The purpose of this Software Requirement Specification (SRS) document is to outline the functional and non-functional requirements for the development of the Inventory Management System. The SRS serves as a reference for the development team and stakeholders to understand the system's capabilities, constraints, and desired behavior.

Scope: The scope of the Inventory Management System includes the design, development, testing, and deployment of a web-based application for the IT department at Hindalco Industries Ltd. The system aims to manage inventory across two systems: P&B (Planning & Budgeting) and CapEx (Capital Expenditure).

Definitions, Acronyms, and Abbreviations:

P&B: Planning & Budgeting

CapEx: Capital Expenditure

SRS: Software Requirement Specification

System Overview

Functional Requirements:

The Inventory Management System shall provide the following functionalities:

- User authentication and authorization for different access levels (admin, authenticated users, unauthenticated users)
- Adding and updating inventory items
- Generating reports for inventory status, transactions, and usage
- Tracking inventory movement and location
- Integration with external systems for data synchronization (if applicable)

Non-functional Requirements:

The non-functional requirements for the Inventory Management System include:

- Usability: The system shall have an intuitive and user-friendly interface.
- Performance: The system shall be responsive and handle a significant number of concurrent users without significant delays.
- Security: The system shall implement appropriate security measures to protect user data and prevent unauthorized access.
- Reliability: The system shall be stable and provide accurate inventory information.

- Scalability: The system should be capable of scaling to accommodate future growth and increasing inventory data.
- Compatibility: The system should be compatible with modern web browsers and mobile devices.

System Architecture

High-Level Architecture: The client-side of the application will be implemented using HTML, CSS, and JavaScript. This includes the web interface for the users to interact with the inventory system, while the server-side will be developed using PL/SQL for business logic and MySQL for data storage and retrieval..

Database Design: The database will include tables to store information such as inventory items, users and transactions. The design will ensure proper normalization and enforce data integrity constraints.

User Interfaces

Login Interface: The login interface shall allow users to enter their credentials and authenticate themselves. Different user roles (admin, authenticated users) will have different levels of access within the system.

Inventory Management Interface: The inventory management interface shall provide functionality for adding and updating inventory items. It should allow users to search for items, view details, and perform bulk actions.

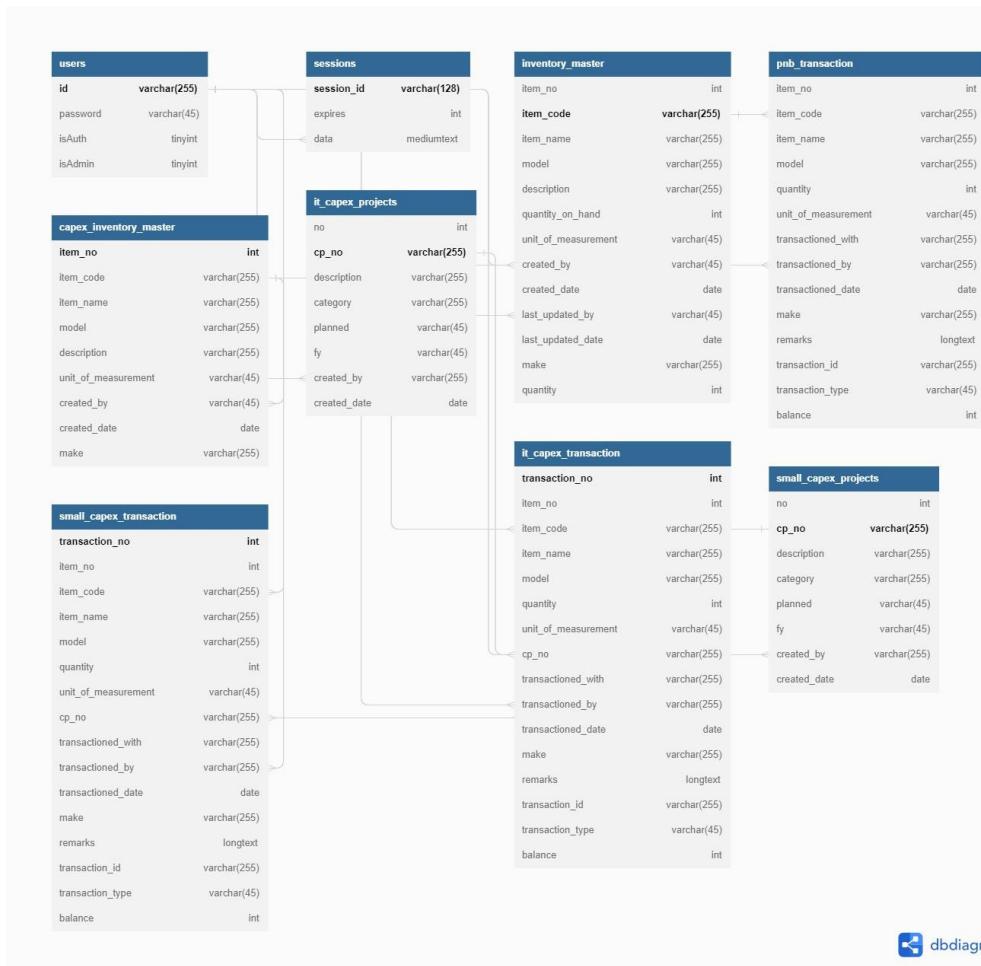
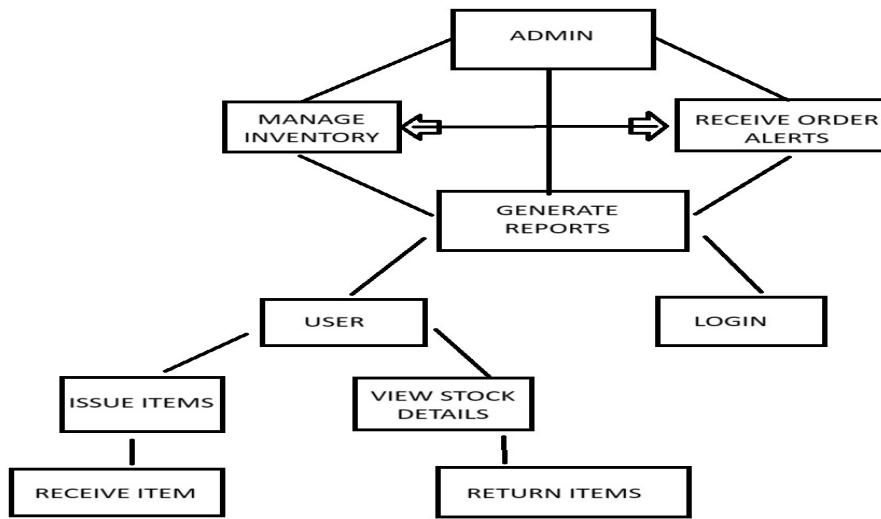
System Constraints

Technology Constraints: The Inventory Management System shall be developed using the following technologies:

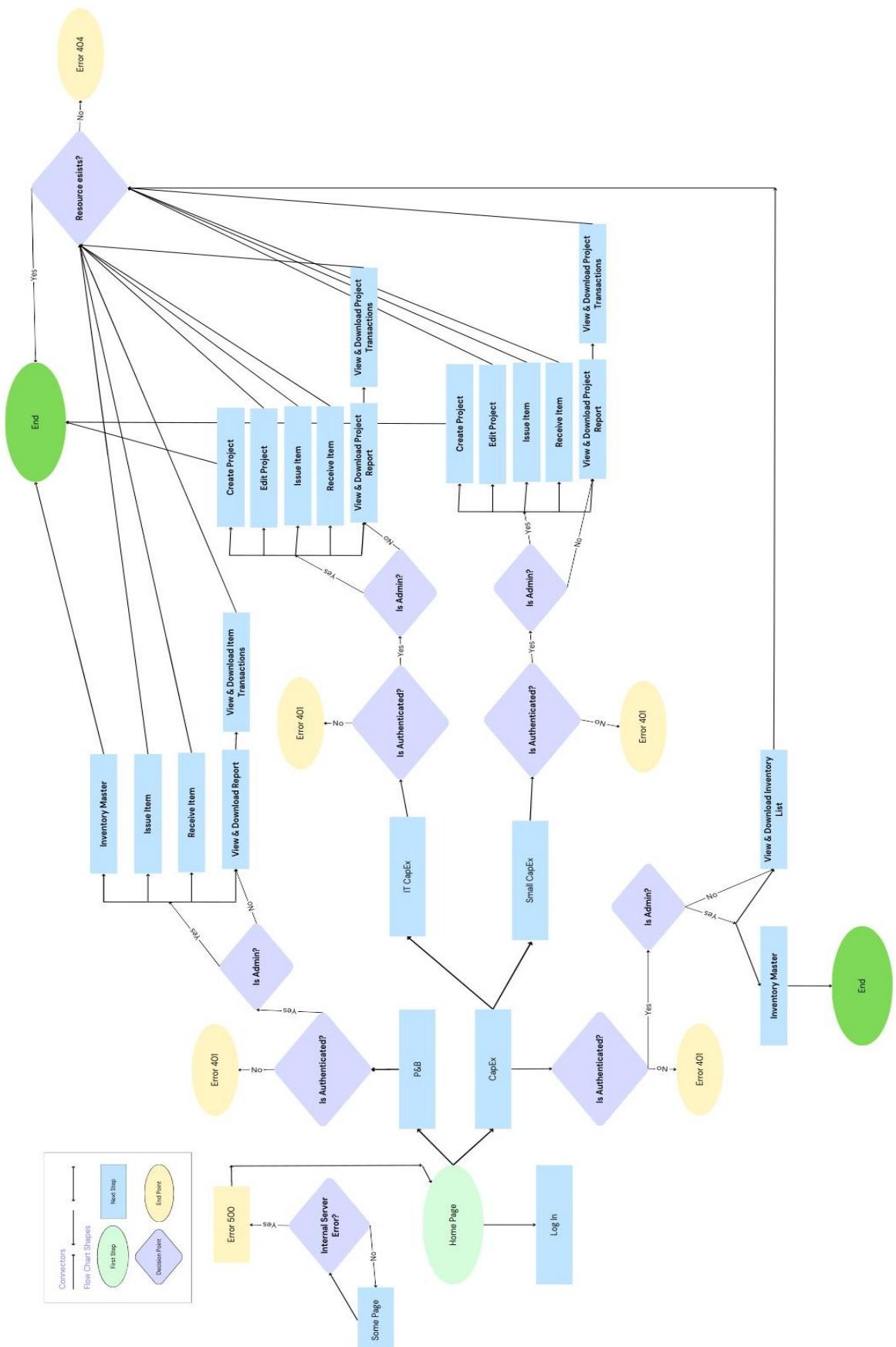
- Front-end Technologies: The system must use HTML, CSS, and JavaScript for the client-side.
- Back-end Technologies: The system must use PL/SQL for server-side processing.
- Database: The system must use Oracle Database for data storage and retrieval.

Security Constraints: The system shall implement appropriate security measures, including data encryption, secure user authentication, and access control mechanisms, to ensure the confidentiality and integrity of the system.

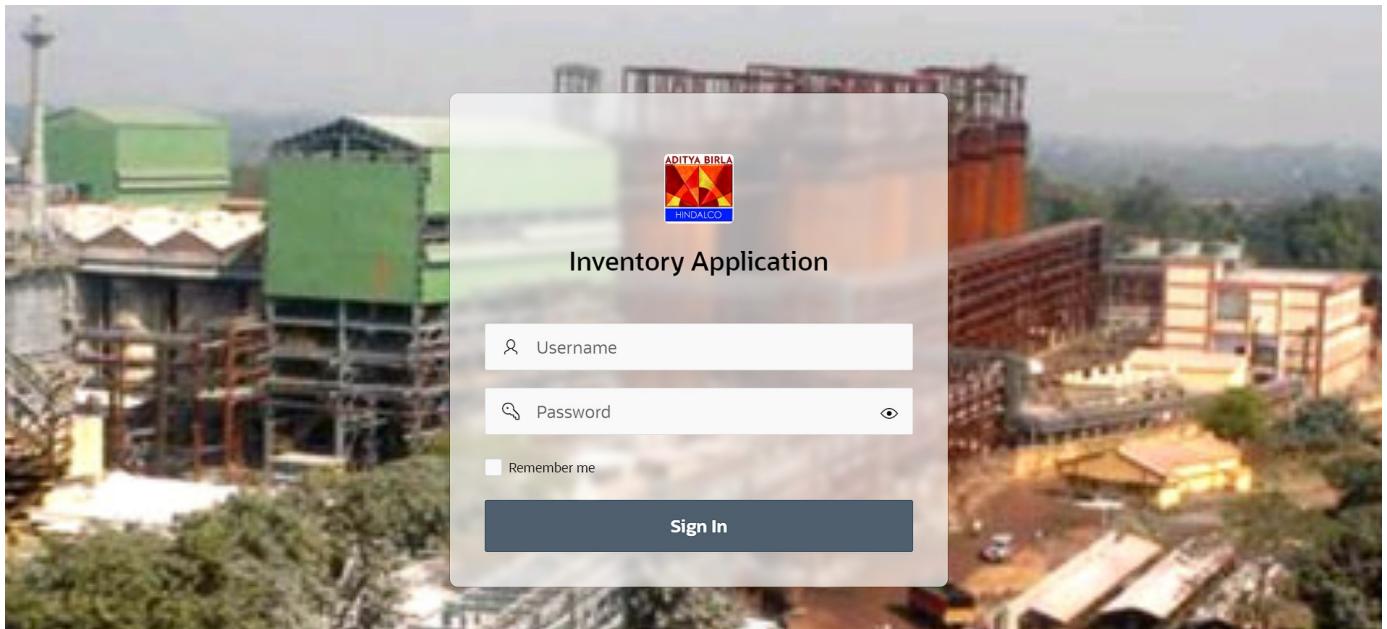
Use Case Diagram & Entity Relationship



CONTROL FLOW DIAGRAM



Login Page



Login id structure

MY_USERS

Columns Data Indexes Constraints Grants Statistics Triggers Dependencies DDL Sample Queries

Download Refresh

CREATE TABLE "MY_USERS"
("ID" NUMBER,
 "UNAME" VARCHAR2(60),
 "LOG_IN" VARCHAR2(50),
 "LOG_PASS" VARCHAR2(60),
 "STATUS" VARCHAR2(1),
 "USER_ROLL" VARCHAR2(20),
 "PASSWORD_HASH" VARCHAR2(100),
 PRIMARY KEY ("ID")
 USING INDEX ENABLE,
 UNIQUE ("LOG_IN")
 USING INDEX ENABLE,
 UNIQUE ("LOG_PASS")
 USING INDEX ENABLE
) ;

Custom Authentication for login Details

The screenshot shows the 'Authentication Schemes' page in Oracle APEX. The page title is 'Application 217661 \ Shared Components \ Authentication Schemes'. There are tabs for 'Authentication Schemes', 'Subscriptions', and 'History'. A search bar and action buttons ('Go', 'Actions', 'Reset', 'Create') are at the top. The main table has columns: Name, Is Current, Scheme Type, Subscribed From, Subscribers, and Updated. Two rows are listed:

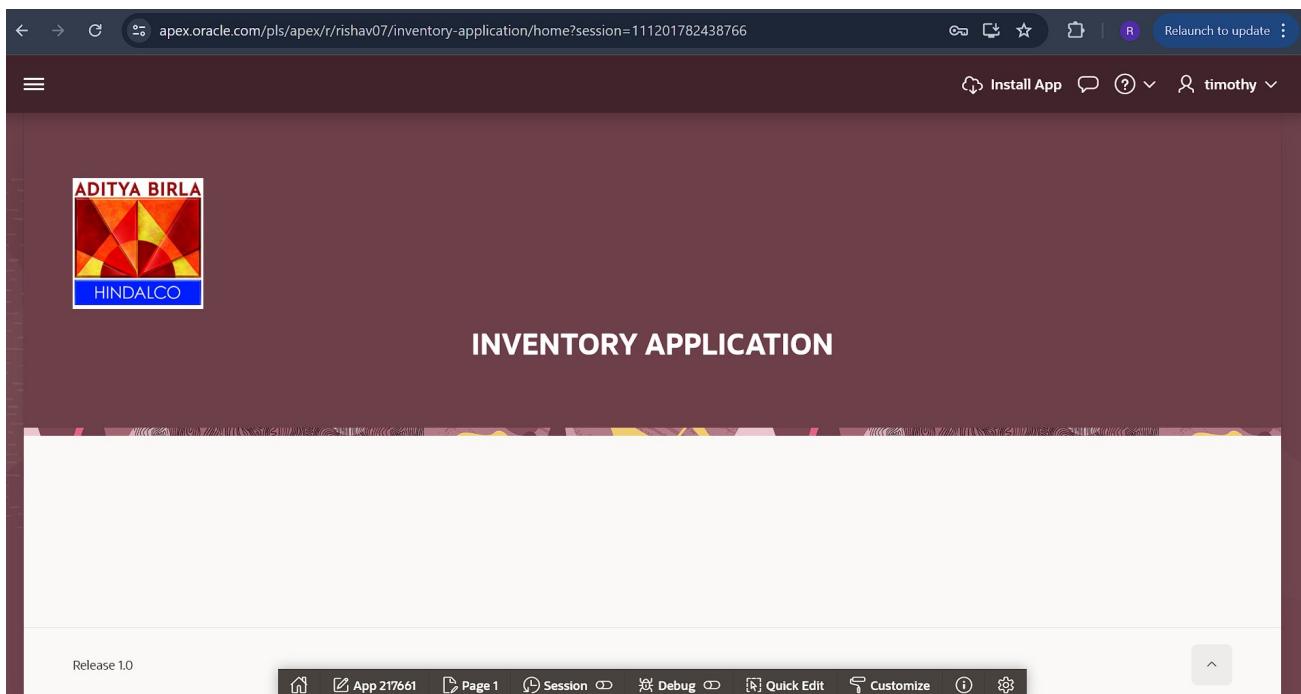
Name	Is Current	Scheme Type	Subscribed From	Subscribers	Updated
INVENTORY_SECURITY	Yes	Custom			2 weeks ago
Oracle APEX Accounts	No	Oracle APEX Accounts			2 weeks ago

A sidebar on the right is titled 'About' and contains information about authentication schemes.

Custom Authentication PL/SQL Code

```
FUNCTION user_aut      (
    p_username IN VARCHAR2,
    p_password VARCHAR2
)
RETURN BOOLEAN
AS
  lc_pwd_exit VARCHAR2 (1);
BEGIN
  SELECT 'Y'
  INTO lc_pwd_exit
  FROM MY_USERS
  WHERE upper(LOG_IN) = UPPER (p_username) AND LOG_PASS = p_password and
status='Y'
;
  RETURN TRUE;
EXCEPTION
  WHEN NO_DATA_FOUND
  THEN
    RETURN FALSE;
END user_aut;
```

INVENTORY MANAGEMENT SYSTEM HOME PAGE



P&B Master Table

The screenshot shows a detailed view of the 'P&B Master Details' section. The left sidebar has a navigation menu with items such as Home, P&B ALERT LOG, P&B MASTER DETAILS, P&B ISSUE ITEM, P&B ISSUE REPORT, P&B RECEIVE ITEM, P&B RECEIVE REPORT, P&B BALANCE REPORT, P&B ONHAND ITEMS, P&B RETURN ITEMS, P&B RETURN ITEMS REP..., and P&B IWYC. The main content area is a table titled 'P&B Master Table' with a 'Create' button. The table has columns for Item Code, Serial No, Item Name, Make, Model, Category, Stock, Uom, Location, Remarks, Userid, Username, and Last Upd By. There are six rows of data:

Item Code	Serial No	Item Name	Make	Model	Category	Stock	Uom	Location	Remarks	Userid	Username	Last Upd By
P001	1	8 PORT SWITCH	ARUBA	1930	CAPEX	4	NO					
P002	2	24 PORT SWITCH	ARUBA	1930	CAPEX	5	NO					
P003	3	24 PORT SWITCH	ARUBA	6000	CAPEX	4	NO					
P004	4	CASING CAPPING 1 IN	NA	1 IN	CAPEX	400	NO					
P005	5	PVC CONDUIT PIPE 3M	NA	NA	CAPEX	300	MTR					
P006	6	CASING CAPPING IN	NA	50X50 MM	CAPEX	40	MTR					

P&B Master Table Structure

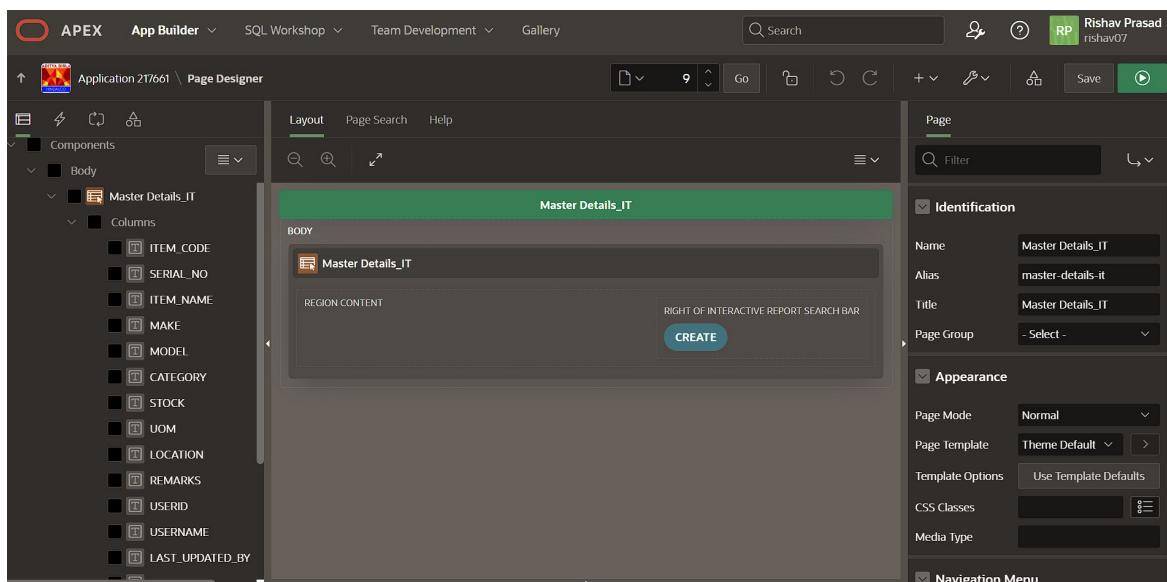
```

CREATE TABLE "IT_ITEMS"
(
    "SERIAL_NO" VARCHAR2(15),
    "ITEM_CODE" VARCHAR2(6),
    "ITEM_NAME" VARCHAR2(50),
    "MAKE" VARCHAR2(15),
    "MODEL" VARCHAR2(30),
    "CATEGORY" VARCHAR2(30),
    "STOCK" NUMBER,
    "UOM" VARCHAR2(6),
    "LOCATION" VARCHAR2(15),
    "REMARKS" VARCHAR2(100),
    "USERID" VARCHAR2(10),
    "USERNAME" VARCHAR2(30),
    "LAST_UPDATED_BY" VARCHAR2(30),
    "LAST_UPDATE_DT" DATE,
    "REORDER_LEVEL" NUMBER DEFAULT 1,
    CONSTRAINT "PK_IT_ITEMS" PRIMARY KEY ("ITEM_CODE")
    USING INDEX ENABLE
);

CREATE OR REPLACE EDITIONABLE TRIGGER "TR_IT_ITEMS_INSERT_SERIAL_NO"
BEFORE INSERT ON IT_ITEMS
FOR EACH ROW
BEGIN
    :NEW.SERIAL_NO := SEQ_IT_ITEMS_SERIAL_NO.NEXTVAL;
END;
/
ALTER TRIGGER "TR_IT_ITEMS_INSERT_SERIAL_NO" ENABLE;

```

P&B Master Table Page



P&B Issue Item Form Page

The screenshot shows the 'ISSUE ITEM FORM' page. The form includes fields for Item Code, Item Name, Make, Model, Category, Uom, ONHAND_STOCK, Issue Qty, Issue Dt, Cust Name, Cust Dept Name, Remarks, and Username (set to TIMOTHY). There are 'Cancel' and 'Create' buttons at the bottom.

P&B Issue Item Table Structure

```
CREATE TABLE "ISSUE_ITEMS"
(
    "ITEM_CODE" VARCHAR2(20),
    "ITEM_NAME" VARCHAR2(50),
    "MAKE" VARCHAR2(50),
    "MODEL" VARCHAR2(30),
    "CATEGORY" VARCHAR2(30),
    "UOM" VARCHAR2(50),
    "ISSUE_QTY" NUMBER,
    "ISSUE_DT" DATE,
    "USERID" VARCHAR2(10),
    "USERNAME" VARCHAR2(30),
    "CUST_CODE" VARCHAR2(10),
    "CUST_NAME" VARCHAR2(30),
    "CUST_DEPT_CODE" VARCHAR2(10),
    "CUST_DEPT_NAME" VARCHAR2(30),
    "REMARKS" VARCHAR2(250)
) ;
```

P&B Issue Item Page View

The screenshot shows the Oracle APEX App Builder interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The search bar has a placeholder 'Search'. The top right corner shows the user 'Rishav Prasad' (rp_rishav07). The main area is titled 'Page Designer' for Application 217661 \ Page Designer. The left sidebar lists components under 'Body' for the 'ISSUE ITEM FORM', specifically the 'Region Body' which contains the 'P22_ITEM_NAME' field. The central workspace displays the 'ISSUE ITEM FORM' with various input fields. The right sidebar is titled 'Page Item' and shows the configuration for the 'P22_ITEM_NAME' field, including a 'List of Values' section with an SQL query:

```
SELECT DISTINCT ITEM_NAME AS DISPLAY_VALUE, ITEM_NAME AS RETURN_VALUE
FROM IT_ITEMS
ORDER BY ITEM_NAME
```

P&B Issue Item Dynamic actions Code For fetch details from master table

ITEM NAME SQL QUERY FOR FETCH DETAILS

```
SELECT DISTINCT ITEM_NAME AS DISPLAY_VALUE, ITEM_NAME AS RETURN_VALUE
FROM IT_ITEMS
ORDER BY ITEM_NAME
```

ITEM MODEL SQL QUERY FOR FETCH MODEL DETAILS

```
SELECT MODEL AS DISPLAY_VALUE, MODEL AS RETURN_VALUE
FROM IT_ITEMS
WHERE ITEM_NAME = :P22_ITEM_NAME
ORDER BY MODEL
```

BY FOLLOWING THIS STEP IT CAN SHOW ITEM CODE, MAKE, CATEGORY, UOM

P&B Issue Item Report Page

The screenshot shows a web-based application interface for managing inventory. On the left, a dark sidebar menu lists various options: Home, P&B ALERT LOG, P&B MASTER DETAILS, P&B ISSUE ITEM, P&B ISSUE REPORT (which is selected), P&B RECEIVE ITEM, P&B RECEIVE REPORT, P&B BALANCE REPORT, P&B ONHAND ITEMS, P&B RETURN ITEMS, P&B RETURN ITEMS REP..., and P&B IWYC. The main content area features a search bar at the top with a 'Go' button and an 'Actions' dropdown. Below is a table with the following columns: Item Code, Item Name, Make, Model, Category, Uom, Issue Qty, Issue Dt, Userid, Username, Cust Code, Cust Name, Cust Dept Code, and Cust Dept. Five rows of data are displayed:

Item Code	Item Name	Make	Model	Category	Uom	Issue Qty	Issue Dt	Userid	Username	Cust Code	Cust Name	Cust Dept Code	Cust Dept
P003	24 PORT SWITCH	ARUBA	6000	CAPEX	NO	3	6/7/2024		TIMOTHY		rohit		ELECTRICAL
P015	6U RACK	DLINK	NWR -5545-GR-CKD	CAPEX	NO	2	6/7/2024		TIMOTHY		rishav		IT DEPARTMENT
P014	9U RACK	DLINK	NWR 5545 GR CKD	CAPEX	NO	1	6/7/2024		TIMOTHY		rishav		IT DEPARTMENT
P001	8 PORT SWITCH	ARUBA	1930	CAPEX	NO	3	6/7/2024		TIMOTHY		RISHAV		LABRADORE
P008	PATCH PANEL 24 PORT	DLINK	NPP-C61BLK241	CAPEX	NO	5	6/7/2024		TIMOTHY		RISHAV		LABRADORE

P&B Receive Item Form Page

The screenshot shows a web-based application interface for managing inventory. On the left, a dark sidebar menu lists various options: Home, P&B ALERT LOG, P&B MASTER DETAILS, P&B ISSUE ITEM, P&B ISSUE REPORT (which is selected), P&B RECEIVE ITEM (highlighted in blue), P&B RECEIVE REPORT, P&B BALANCE REPORT, P&B ONHAND ITEMS, P&B RETURN ITEMS, P&B RETURN ITEMS REP..., and P&B IWYC. The main content area displays a 'RECEIVE ITEM FORM' dialog box with the following fields:

- Item Code
- Item Name
- Make
- Model
- Category
- Uom
- Recv Qty
- Recv Dt
- Username: TIMOTHY
- Cancel
- Create

P&B Receive Item Table Structure

```
CREATE TABLE "RECEIVE_ITEMS"
(
    "ITEM_CODE" VARCHAR2(6),
    "ITEM_NAME" VARCHAR2(50),
    "MAKE" VARCHAR2(15),
    "MODEL" VARCHAR2(30),
    "CATEGORY" VARCHAR2(30),
    "UOM" VARCHAR2(6),
    "RECV_QTY" NUMBER,
    "RECV_DT" DATE,
    "USERID" VARCHAR2(10),
    "USERNAME" VARCHAR2(30)
) ;
```

P&B Receive Item Page View

The screenshot shows the Oracle APEX Page Designer interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The search bar contains the text 'Search'. On the right, there is a user profile for 'Rishav Prasad' (rishav07). The main workspace displays the 'RECEIVE ITEM FORM' page. The page title is 'RECEIVE ITEM FORM'. The body contains a region titled 'RECEIVE ITEM FORM' with several input fields: P17_ITEM_CODE, P17_ITEM_NAME, P17_MAKE, P17_MODEL, P17_CATEGORY, P17_UOM, P17_RECV_QTY, P17_RECV_DT, P17_USERID, and P17_USERNAME. Below the region is a 'REGION CONTENT' section with buttons for CLOSE, CANCEL, DELETE, and SAVE. The 'Item Name' field (P17_ITEM_NAME) is currently selected, highlighted with a green border. The left sidebar shows the page structure, including components like Pre-Rendering, Body, and Region Body. The right sidebar provides settings for the 'Item Name' field, such as 'Name' (P17_ITEM_NAME), 'Type' (Popup LOV), 'Label' (Item Name), and 'Settings' (Display As: Inline Popup, Initial Fetch: Automatic, etc.).

P&B Receive Item Dynamic actions Code For fetch details from master table

ITEM NAME SQL QUERY FOR FETCH DETAILS

```
SELECT DISTINCT ITEM_NAME AS DISPLAY_VALUE, ITEM_NAME AS RETURN_VALUE
FROM IT_ITEMS
ORDER BY ITEM_NAME
```

ITEM MODEL SQL QUERY FOR FETCH MODEL DETAILS

```
SELECT MODEL AS DISPLAY_VALUE, MODEL AS RETURN_VALUE
FROM IT_ITEMS
WHERE ITEM_NAME = :P22_ITEM_NAME
ORDER BY MODEL
```

BY FOLLOWING THIS STEP IT CAN SHOW ITEM CODE, MAKE, CATEGORY, UOM

P&B Receive Item Report Page

The screenshot shows a web-based application interface for managing inventory. On the left, there is a vertical navigation menu with the following items:

- Home
- P&B ALERT LOG
- P&B MASTER DETAILS
- P&B ISSUE ITEM
- P&B ISSUE REPORT
- P&B RECEIVE ITEM
- P&B RECEIVE REPORT
- P&B BALANCE REPORT
- P&B ONHAND ITEMS
- P&B RETURN ITEMS
- P&B RETURN ITEMS REP...
- P&B IWYC

The "P&B RECEIVE REPORT" item is currently selected, indicated by a highlighted background.

The main content area displays a table titled "Actions" with the following data:

Item Code	Item Name	Make	Model	Category	Uom	Recv Qty	Recv Dt	Userid	Username
P016	12U RACK	DLINK	NWR-5545-GR-CKD	CAPEX	NO	10	6/8/2024		TIMOTHY

Below the table, the text "Release 1.0" is visible. At the top right of the main area, there are several icons: a cloud icon, "Install App", a help icon, a user icon, and the username "timothy".

P&B Return Item Form Page

The screenshot shows a web-based application interface for managing inventory returns. On the left, a sidebar menu lists various P&B functions: Home, P&B ALERT LOG, P&B MASTER DETAILS, P&B ISSUE ITEM, P&B ISSUE REPORT, P&B RECEIVE ITEM, P&B RECEIVE REPORT, P&B RETURN ITEMS (which is selected), P&B RETURN ITEMS REP..., P&B BALANCE REPORT, P&B ONHAND ITEMS, and P&B IWYC. At the top right, there are links for 'Install App', help, and user 'timothy'. The main content area is titled 'RETURN ITEM FORM' and contains several input fields: 'Item Code', 'Item Name', 'Make', 'Model', 'Category', 'Uom', 'Return Qty', 'Return Dt', 'Userid', and 'Username' (with the value 'TIMOTHY' displayed). Below these fields are 'Cancel' and 'Create' buttons.

P&B Return Item Table Structure

```
CREATE TABLE "RETURN_ITEMS"
(
    "ITEM_CODE" VARCHAR2(6),
    "ITEM_NAME" VARCHAR2(50),
    "MAKE" VARCHAR2(15),
    "MODEL" VARCHAR2(30),
    "CATEGORY" VARCHAR2(30),
    "UOM" VARCHAR2(6),
    "RETURN_QTY" NUMBER,
    "RETURN_DT" DATE,
    "USERID" VARCHAR2(10),
    "USERNAME" VARCHAR2(30)
) ;
```

P&B Return Item Page View

The screenshot shows the Oracle APEX App Builder's Page Designer interface. The left sidebar lists components for 'Page 16: RETURN ITEM FORM'. The main area displays a form titled 'RETURN ITEM FORM' with fields for P16_ITEM_CODE, P16_ITEM_NAME, P16_MAKE, P16_MODEL, P16_CATEGORY, P16_UOM, P16_RETURN_QTY, P16_RETURN_DT, P16_USERID, and P16_USERNAME. Below the form are buttons for CLOSE, CANCEL, DELETE, and SAVE. On the right, the 'Page Item' panel shows validation settings (Value Required, Maximum Length 50 characters) and a 'List of Values' section with an SQL query:

```
SELECT DISTINCT ITEM_NAME AS DISPLAY_VALUE, ITEM_NAME AS RETURN_VALUE
FROM IT_ITEMS
ORDER BY ITEM_NAME
```

P&B Return Item Dynamic actions Code For fetch details from master table

ITEM NAME SQL QUERY FOR FETCH DETAILS

```
SELECT DISTINCT ITEM_NAME AS DISPLAY_VALUE, ITEM_NAME AS RETURN_VALUE
FROM IT_ITEMS
ORDER BY ITEM_NAME
```

ITEM MODEL SQL QUERY FOR FETCH MODEL DETAILS

```
SELECT MODEL AS DISPLAY_VALUE, MODEL AS RETURN_VALUE
FROM IT_ITEMS
WHERE ITEM_NAME = :P22_ITEM_NAME
ORDER BY MODEL
```

BY FOLLOWING THIS STEP IT CAN SHOW ITEM CODE, MAKE, CATEGORY, UOM

P&B Return Item Report Page

This screenshot shows the P&B Return Item Report Page. The left sidebar contains a navigation menu with items like Home, P&B ALERT LOG, P&B MASTER DETAILS, P&B ISSUE ITEM, P&B ISSUE REPORT, P&B RECEIVE ITEM, P&B RECEIVE REPORT, P&B RETURN ITEMS, P&B RETURN ITEMS REP..., P&B BALANCE REPORT, P&B ONHAND ITEMS, and P&B IWYC. The main content area features a search bar with a dropdown arrow, a 'Go' button, and an 'Actions' dropdown. Below is a table with the following data:

Item Code	Item Name	Make	Model	Category	Uom	Return Qty	Return Dt	Userid	Username
P017	CAT-6 CABLE	DLINK	NA	CAPEX	MTR	10	6/8/2024	Rishav123	TIMOTHY

At the bottom left is the text "Release 1.0".

P&B Balance Report

This screenshot shows the P&B Balance Report page. The left sidebar has the same navigation menu as the previous page. The main content area displays a table of transaction records with the following columns: Transaction ID, Item Code, Transaction Type, Quantity Before, ISSUE_QTY, BALANCE_QTY, Rate, and Transaction Date. The data is as follows:

Transaction ID	Item Code	Transaction Type	Quantity Before	ISSUE_QTY	BALANCE_QTY	Rate	Transaction Date
524	P003	ISS	4	3	1		6/7/2024
561	P015	ISS	3	2	1		6/7/2024
622	P017	RET	3050	10	3060		6/7/2024
581	P014	ISS	2	1	1		6/7/2024
542	P001	ISS	4	3	1		6/7/2024
601	P008	ISS	6	5	1		6/7/2024
621	P016	REC	10	10	20		6/7/2024

At the bottom left is the text "Release 1.0".

P&B Triggers for ISS,REC,RET

Issue Item Trigger

```
create or replace TRIGGER TR_ISSUE_ITEMS_UPDATE_OB
AFTER INSERT ON ISSUE_ITEMS
FOR EACH ROW
DECLARE
    l_quantity_before NUMBER;
    l_quantity_after  NUMBER;
BEGIN
    -- Fetch the last quantity after from the OPENING_BALANCE for this item
    code
    -- If there is no previous record, fetch the stock from IT_ITEMS
    SELECT COALESCE(
        (SELECT QUANTITY_AFTER
         FROM OPENING_BALANCE
         WHERE ITEM_CODE = :NEW.ITEM_CODE
         ORDER BY TRANSACTION_ID DESC FETCH FIRST 1 ROWS ONLY),
        (SELECT STOCK
         FROM IT_ITEMS
         WHERE ITEM_CODE = :NEW.ITEM_CODE)
    ) INTO l_quantity_before
    FROM DUAL;

    -- Calculate the quantity after the transaction
    l_quantity_after := l_quantity_before - :NEW.ISSUE_QTY;

    -- Insert into OPENING_BALANCE
    INSERT INTO OPENING_BALANCE (
        ITEM_CODE, TRANSACTION_TYPE, QUANTITY_BEFORE, QUANTITY,
        QUANTITY_AFTER, RATE
    ) VALUES (
        :NEW.ITEM_CODE, 'ISS', l_quantity_before, :NEW.ISSUE_QTY,
        l_quantity_after, NULL
    );
END;
/
```

Receive Item Trigger

```
create or replace TRIGGER TR_RECEIVE_ITEMS_UPDATE_OB
AFTER INSERT ON RECEIVE_ITEMS
FOR EACH ROW
DECLARE
    l_quantity_before NUMBER;
    l_quantity_after  NUMBER;
BEGIN
    -- Fetch the last quantity after from the OPENING_BALANCE for this
item code
    -- If there is no previous record, fetch the stock from IT_ITEMS
    SELECT COALESCE(
        (SELECT QUANTITY_AFTER
         FROM OPENING_BALANCE
         WHERE ITEM_CODE = :NEW.ITEM_CODE
         ORDER BY TRANSACTION_ID DESC FETCH FIRST 1 ROWS ONLY),
        (SELECT STOCK
         FROM IT_ITEMS
         WHERE ITEM_CODE = :NEW.ITEM_CODE)
    ) INTO l_quantity_before
    FROM DUAL;

    -- Calculate the quantity after the transaction
    l_quantity_after := l_quantity_before + :NEW.RECV_QTY;

    -- Insert into OPENING_BALANCE
    INSERT INTO OPENING_BALANCE (
        ITEM_CODE, TRANSACTION_TYPE, QUANTITY_BEFORE, QUANTITY,
        QUANTITY_AFTER, RATE
    ) VALUES (
        :NEW.ITEM_CODE, 'REC', l_quantity_before, :NEW.RECV_QTY,
        l_quantity_after, NULL
    );
END;
/
```

RETURN ITEM TRIGGER

```

create or replace TRIGGER TR_RETURN_ITEMS_UPDATE_OB
AFTER INSERT ON RETURN_ITEMS
FOR EACH ROW
DECLARE
    l_quantity_before NUMBER;
    l_quantity_after  NUMBER;
BEGIN
    -- Fetch the last quantity after from the OPENING_BALANCE for this
    item code
    -- If there is no previous record, fetch the stock from IT_ITEMS
    SELECT COALESCE(
        (SELECT QUANTITY_AFTER
         FROM OPENING_BALANCE
         WHERE ITEM_CODE = :NEW.ITEM_CODE
         ORDER BY TRANSACTION_ID DESC FETCH FIRST 1 ROWS ONLY),
        (SELECT STOCK
         FROM IT_ITEMS
         WHERE ITEM_CODE = :NEW.ITEM_CODE)
    ) INTO l_quantity_before
    FROM DUAL;

    -- Calculate the quantity after the return
    l_quantity_after := l_quantity_before + :NEW.RETURN_QTY;

    -- Insert into OPENING_BALANCE
    INSERT INTO OPENING_BALANCE (
        ITEM_CODE, TRANSACTION_TYPE, QUANTITY_BEFORE, QUANTITY,
        QUANTITY_AFTER, RATE
    ) VALUES (
        :NEW.ITEM_CODE, 'RET', l_quantity_before, :NEW.RETURN_QTY,
        l_quantity_after, NULL
    );

    -- Update the ONHAND_QUANTITY table
    UPDATE ONHAND_QUANTITY
    SET QUANTITY = l_quantity_after
    WHERE ITEM_CODE = :NEW.ITEM_CODE;
END;
/

```

P&B Onhand Items REPORT

Item Code	Item Name	Quantity
P003	24 PORT SWITCH	1
P015	6U RACK	1
P017	CAT-6 CABLE	3060
P001	8 PORT SWITCH	1
P014	9U RACK	1
P008	PATCH PANEL 24 PORT	1
P016	12U RACK	20

Release 1.0

P&B Trigger for Onhand Quantity

```

create or replace TRIGGER TR_UPDATE_ONHAND_QUANTITY
AFTER INSERT OR UPDATE ON OPENING_BALANCE
FOR EACH ROW
BEGIN
    -- Ensure the `ITEM_CODE` exists and `QUANTITY_AFTER` is provided
    IF :NEW.ITEM_CODE IS NOT NULL AND :NEW.QUANTITY_AFTER IS NOT NULL THEN
        MERGE INTO ONHAND_QUANTITY oq
        USING (
            SELECT :NEW.ITEM_CODE AS ITEM_CODE,
                   :NEW.QUANTITY_AFTER AS QUANTITY,
                   (SELECT ITEM_NAME FROM IT_ITEMS WHERE ITEM_CODE
                   = :NEW.ITEM_CODE) AS ITEM_NAME
            FROM DUAL
        ) src
        ON (oq.ITEM_CODE = src.ITEM_CODE)
        WHEN MATCHED THEN
            UPDATE SET oq.QUANTITY = src.QUANTITY, oq.ITEM_NAME =
src.ITEM_NAME
        WHEN NOT MATCHED THEN
            INSERT (oq.ITEM_CODE, oq.ITEM_NAME, oq.QUANTITY)
            VALUES (src.ITEM_CODE, src.ITEM_NAME, src.QUANTITY);
    END IF;
END;
/

```

P&B Item wise Yearly Consumption Report

The screenshot shows a user interface for an inventory management system. On the left, there's a dark sidebar with a navigation menu containing items like Home, P&B ALERT LOG, P&B MASTER DETAILS, P&B ISSUE ITEM, P&B ISSUE REPORT, P&B RECEIVE ITEM, P&B RECEIVE REPORT, P&B RETURN ITEMS, P&B RETURN ITEMS REP..., P&B BALANCE REPORT, P&B ONHAND ITEMS, and P&B IWYC. The main content area has a title "Item-wise Yearly Consumption report". Below it is a search bar with a dropdown arrow, a "Go" button, and an "Actions" dropdown. A table follows, with columns: Item Code, Item Name, Issue Year, and Total Issued Quantity. The data in the table is as follows:

Item Code	Item Name	Issue Year	Total Issued Quantity
P001	8 PORT SWITCH	2024	3
P003	24 PORT SWITCH	2024	3
P008	PATCH PANEL 24 PORT	2024	5
P014	9U RACK	2024	1
P015	6U RACK	2024	2

At the bottom right of the table, there's a page number "1 - 5".

P&B Item wise Yearly report View Created

```

CREATE OR REPLACE FORCE EDITIONABLE VIEW
"ITEMWISE_YEARLY_CONSUMPTION" ("ITEM_CODE", "ITEM_NAME",
"ISSUE_YEAR", "TOTAL_ISSUED_QUANTITY") AS
  SELECT
    ITEM_CODE,
    ITEM_NAME,
    EXTRACT(YEAR FROM ISSUE_DT) AS ISSUE_YEAR,
    SUM(ISSUE_QTY) AS TOTAL_ISSUED_QUANTITY
  FROM
    ISSUE_ITEMS
  GROUP BY
    ITEM_CODE, ITEM_NAME, EXTRACT(YEAR FROM ISSUE_DT);

```

CREATED A PROJECT FORM PAGE

CREATE A PROJECT

Cp No	Category CAPEX
Description	
Planned/Unplanned	Financial Year
Created Date	Created By TIMOTHY
<input type="button" value="Cancel"/> <input type="button" value="Create"/>	

CAPEX PROJECT DETAILS PAGE

Go Actions

Serial No	Cp No	Category	Description	Planned Unplanned	Financial Year	Created Date	Created By
1	ITD/CP/CAT1B/25/0004-123/18	CAPEX	Centralized Procurement of Laptops & Desktops for PAN Hindalco locations	Planned	2024	4 days ago	TIMOTHY
2	ITD/CP/CAT1B/24/0118	CAPEX	Capex_Lenovo Desktops from CIT for Muri FY24-25 Q1 & Q2	Planned	2024	4 days ago	TIMOTHY

STRUCTURE OF CAPEX PROJECT DETAILS

```

CREATE TABLE "CAPEX_PROJECTS"
(
    "SERIAL_NO" NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY
    MINVALUE 1 MAXVALUE 99999999999999999999999999999999 INCREMENT BY 1 START
    WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,
    "CP_NO" VARCHAR2(50) NOT NULL ENABLE,
    "CATEGORY" VARCHAR2(30),
    "DESCRIPTION" VARCHAR2(100),
    "PLANNED_UNPLANNED" VARCHAR2(20),
    "FINANCIAL_YEAR" VARCHAR2(20),
    "CREATED_DATE" DATE DEFAULT SYSDATE,
    "CREATED_BY" VARCHAR2(20),
    CONSTRAINT "PK_CAPEX_PROJECTS" PRIMARY KEY ("SERIAL_NO")
    USING INDEX ENABLE,
    CONSTRAINT "UQ_CP_NO" UNIQUE ("CP_NO")
    USING INDEX ENABLE
);

```

CAPEX PROJECT DETAILS PAGE

The screenshot shows the Oracle APEX Page Designer interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is logged in as Rishav Prasad (rishav07). The current page is titled "Page 21: CREATE A PROJECT". The left sidebar shows the page structure under "Components" with sections for Pre-Rendering, Body, and Footer. The main content area displays a "CREATE A PROJECT" page with a green header. The body contains a "REGION BODY" with several input fields (P21_CP_NO, P21_CATEGORY, P21_DESCRIPTION, P21_PLANNED_UNPLANNED, P21_FINANCIAL_YEAR, P21_CREATED_DATE, P21_CREATED_BY) and a "REGION CONTENT" section with buttons for CLOSE, CANCEL, DELETE, and SAVE.

CaPeX MASTER TABLE FORM PAGE

CAPEX MASTER FORM

Item Code CO21	Item Name		
Make	Model	Category	Uom --SELECT--
Created Date	Created By TIMOTHY		

Create

CaPeX MASTER TABLE REPORT PAGE

Actions

Serial No	Item Code	Item Name	Make	Model	Category	Created Date	Created By	Uom
3	C003	24 PORT SWITCH	ARUBA	6000	CAPEX	3 days ago	TIMOTHY	NO
1	C001	8 PORT SWITCH	ARUBA	1930	CAPEX	3 days ago		NO
2	C002	24 PORT SWITCH	ARUBA	1930	CAPEX	3 days ago		NO
4	C004	CASING CAPPING 1 IN	NA	1 IN	CAPEX	3 days ago		NO
5	C005	PVC CONDUIT PIPE 3M	NA	NA	CAPEX	3 days ago		MTR
6	C006	CASING CAPPING IN	NA	50X50 MM	CAPEX	3 days ago		MTR
7	C007	I/O BOX JACK	D LINK	NCB-C6UGRYR1-3	CAPEX	3 days ago		NO
8	C008	PATCH PANEL 24 PORT	D LINK	NPP-C61BLK241	CAPEX	3 days ago		NO
9	C009	CAT-6 PATCH CORD 3 M	D LINK	NA	CAPEX	3 days ago		NO
10	C010	RJ-45 CONNECTOR	D LINK	NA	CAPEX	3 days ago		NO

CaPeX MASTER TABLE STRUCTURE

```

CREATE TABLE "CAPEX_MASTER"
(
    "SERIAL_NO" NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY
    MINVALUE 1 MAXVALUE 99999999999999999999999999999999 INCREMENT BY 1
    START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL
    ENABLE,
    "ITEM_CODE" VARCHAR2(50) NOT NULL ENABLE,
    "ITEM_NAME" VARCHAR2(100) NOT NULL ENABLE,
    "MAKE" VARCHAR2(50),
    "MODEL" VARCHAR2(50),
    "CATEGORY" VARCHAR2(50),
    "CREATED_DATE" DATE DEFAULT SYSDATE,
    "CREATED_BY" VARCHAR2(50),
    "UOM" VARCHAR2(30),
    CONSTRAINT "PK_CAPEX_MASTER" PRIMARY KEY ("SERIAL_NO")
    USING INDEX ENABLE,
    CONSTRAINT "UQ_ITEM_CODE" UNIQUE ("ITEM_CODE")
    USING INDEX ENABLE
);

```

CaPeX ISSUE ITEM REPORT PAGE

Serial No	Cp No	Item Code	Item Name	Model	Make	Description	Issue Quantity	Uom	Issued Date	Issues To
12	ITD/CP/CAT1B/24/0118	C016	12U RACK	NWR-5545-GR-CKD	DLINK	Capex_Lenovo Desktops from CIT for Muri FY24-25 Q1 & Q2	1	NO	6/6/2024	IT DEPARTMENT
13	ITD/CP/CAT1B/24/0118	C016	12U RACK	NWR-5545-GR-CKD	DLINK	Capex_Lenovo Desktops from CIT for Muri FY24-25 Q1 & Q2	2	NO		ABHINANDAN
11	ITD/CP/CAT1B/24/0118	C016	12U RACK	NWR-5545-GR-CKD	DLINK	Capex_Lenovo Desktops from CIT for Muri FY24-25 Q1 & Q2	5	NO	6/6/2024	MATERIAL

CaPeX ISSUE ITEM FORM PAGE

The screenshot shows the 'CAPEX ISSUE ITEM FORM' page. The form includes fields for Cp No, Description, Item Name, Item Code, Model, Make, Uom, Onhand Stock, Issue Quantity, ISSUE TO, and Remarks. At the bottom are 'Cancel' and 'Create' buttons. The left sidebar has a 'CAPEX ISSUE ITEM FORM' link selected.

CaPeX ISSUE ITEM TABLE STRUCTURE

```

CREATE TABLE "CAPEX_ISSUE_ITEM"
(
    "SERIAL_NO" NUMBER NOT NULL ENABLE,
    "CP_NO" VARCHAR2(50) NOT NULL ENABLE,
    "ITEM_CODE" VARCHAR2(10) NOT NULL ENABLE,
    "ITEM_NAME" VARCHAR2(50),
    "MODEL" VARCHAR2(30),
    "MAKE" VARCHAR2(15),
    "DESCRIPTION" VARCHAR2(100),
    "ISSUE_QUANTITY" NUMBER NOT NULL ENABLE,
    "UOM" VARCHAR2(6),
    "ISSUED_DATE" DATE DEFAULT SYSDATE,
    "ISSUES_TO" VARCHAR2(100) NOT NULL ENABLE,
    "REMARKS" VARCHAR2(100),
    CONSTRAINT "PK_CAPEX_ISSUE_ITEM" PRIMARY KEY ("SERIAL_NO")
    USING INDEX ENABLE
);

ALTER TABLE "CAPEX_ISSUE_ITEM" ADD CONSTRAINT
"FK_CAPEX_ISSUE_ITEM_CP_NO" FOREIGN KEY ("CP_NO")
    REFERENCES "CAPEX_PROJECTS" ("CP_NO") ENABLE;
ALTER TABLE "CAPEX_ISSUE_ITEM" ADD CONSTRAINT
"FK_CAPEX_ISSUE_ITEM_ITEM_CODE" FOREIGN KEY ("ITEM_CODE")
    REFERENCES "CAPEX_MASTER" ("ITEM_CODE") ENABLE;

```

CaPeX ISSUE ITEM PAGE

CAPEX ISSUE ITEM FORM

Identification

- Name: CAPEX ISSUE ITEM FORM
- Alias: capex-issue-item-form
- Title: CAPEX ISSUE ITEM FORM
- Page Group: - Select -

Appearance

- Page Mode: Normal
- Page Template: Theme Default
- Template Options: Use Template Defaults
- CSS Classes:
- Media Type:

Serial No	Cp No	Item Code	Item Name	Model	Make	Description	Receive Quantity	Uom	Received Date	Received By	Remarks
2	ITD/CP/CAT1B/24/0118	C016	12U RACK	NWR-5545-GR-CKD	DLINK	Capex_Lenovo Desktops from CIT for Muri FY24-25 Q1 & Q2	10	NO	6/6/2024	TIMOTHY	BHHBH

CaPeX RECEIVE ITEM REPORT PAGE

Actions

Serial No	Cp No	Item Code	Item Name	Model	Make	Description	Receive Quantity	Uom	Received Date	Received By	Remarks
2	ITD/CP/CAT1B/24/0118	C016	12U RACK	NWR-5545-GR-CKD	DLINK	Capex_Lenovo Desktops from CIT for Muri FY24-25 Q1 & Q2	10	NO	6/6/2024	TIMOTHY	BHHBH

CaPeX RECEIVE ITEM FORM PAGE

The screenshot shows the 'CAPEX RECEIVE ITEM FORM' page. The form includes fields for Cp No, Item Code, Item Name, Model, Make, Uom, Description, Receive Quantity, Received Date, Received By, and Remarks. A 'Create' button is located at the bottom right. The left sidebar has a dark theme with various navigation links.

CaPeX RECEIVE ITEM TABLE STRUCTURE

```

CREATE TABLE "CAPEX_RECEIVE_ITEM"
(
    "SERIAL_NO" NUMBER NOT NULL ENABLE,
    "CP_NO" VARCHAR2(50) NOT NULL ENABLE,
    "ITEM_CODE" VARCHAR2(10) NOT NULL ENABLE,
    "ITEM_NAME" VARCHAR2(50),
    "MODEL" VARCHAR2(30),
    "MAKE" VARCHAR2(15),
    "DESCRIPTION" VARCHAR2(100),
    "RECEIVE_QUANTITY" NUMBER NOT NULL ENABLE,
    "UOM" VARCHAR2(6),
    "RECEIVED_DATE" DATE DEFAULT SYSDATE,
    "RECEIVED_BY" VARCHAR2(100) NOT NULL ENABLE,
    "REMARKS" VARCHAR2(100),
    CONSTRAINT "PK_CAPEX_RECEIVE_ITEM" PRIMARY KEY ("SERIAL_NO")
    USING INDEX ENABLE
);

ALTER TABLE "CAPEX_RECEIVE_ITEM" ADD CONSTRAINT
"FK_CAPEX_RECEIVE_ITEM_CP_NO" FOREIGN KEY ("CP_NO")
    REFERENCES "CAPEX_PROJECTS" ("CP_NO") ENABLE;
ALTER TABLE "CAPEX_RECEIVE_ITEM" ADD CONSTRAINT
"FK_CAPEX_RECEIVE_ITEM_ITEM_CODE" FOREIGN KEY ("ITEM_CODE")
    REFERENCES "CAPEX_MASTER" ("ITEM_CODE") ENABLE;

```

CaPeX RECEIVE ITEM PAGE

The screenshot shows the Oracle APEX Page Designer interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and user profile for Rishav Prasad (rishav07) are also present. The main area displays the 'CAPEX RECEIVE ITEM FORM' page. The left sidebar shows the page structure under 'Page 32: CAPEX RECEIVE ITEM FOR'. It includes sections for Pre-Rendering, Components (Body), and Region Body, which contains fields for P32_SERIAL_NO, P32_CP_NO, P32_ITEM_CODE, P32_ITEM_NAME, P32_MODEL, P32_MAKE, P32_UOM, P32_DESCRIPTION, P32_RECEIVE_QUANTITY, P32_RECEIVED_DATE, and P32_RECEIVED_BY. The right sidebar contains tabs for Page, Identification, Appearance, and Navigation Menu. Under Identification, the page is named 'CAPEX RECEIVE ITEM FOR' with alias 'capex-receive-item-form'. Under Appearance, the page mode is set to 'Normal' and the template is 'Theme Default'.

CaPeX RETURN ITEM REPORT PAGE

The screenshot shows a web browser displaying the 'CAPEX RETURN ITEM REPORT' page from apex.oracle.com. The left sidebar has a navigation menu with items like P&B IWYC, CAPEX PROJECT DETAILS, CREATE A PROJECT, CAPEX MASTER REPORT, CAPEX ISSUE ITEM REPORT, CAPEX ISSUE ITEM FORM, CAPEX RECEIVE ITEM RE..., CAPEX RECEIVE ITEM FO..., CAPEX RETURN ITEM RE..., CAPEX RETURN ITEM FO..., CAPEX OPENING BALAN..., and CAPEX ONHAND QUANT... The main content area shows a table titled 'Actions' with columns for Serial No, Cp No, Item Code, Item Name, Model, Make, Description, Return Quantity, Uom, Returned Date, Returned By, and Remarks. There is one row of data: Serial No 1, Cp No ITD/CP/CAT1B/24/0118, Item Code C016, Item Name 12U RACK, Model NWR-5545-GR-CKD, Make DLINK, Description Capex_Lenovo Desktops from CIT for Muri FY24-25 Q1 & Q2, Return Quantity 5, Uom NO, Returned Date 6/9/2024, Returned By RISHAV, and Remarks TESTING. The bottom of the page includes a footer with links for Release 10, App 217661, Page 37, Session, Debug, Quick Edit, Customize, and other system icons.

CaPeX RETURN ITEM FORM PAGE

The screenshot shows a web-based application interface for managing CAPEX return items. The main form is titled "CAPEX RETURN ITEM FORM". It includes fields for Cp No, Item Code, Item Name, Model, Make, Uom, Description, Return Quantity, Returned Date, Returned By, and Remarks. A "Create" button is located at the bottom right of the form. The left sidebar provides navigation links for various CAPEX-related modules.

CaPeX RETURN ITEM TABLE STRUCTURE

```

CREATE TABLE "CAPEX_RETURN_ITEM"
(
    "SERIAL_NO" NUMBER NOT NULL ENABLE,
    "CP_NO" VARCHAR2(50) NOT NULL ENABLE,
    "ITEM_CODE" VARCHAR2(10) NOT NULL ENABLE,
    "ITEM_NAME" VARCHAR2(50),
    "MODEL" VARCHAR2(30),
    "MAKE" VARCHAR2(15),
    "DESCRIPTION" VARCHAR2(100),
    "RETURN_QUANTITY" NUMBER NOT NULL ENABLE,
    "UOM" VARCHAR2(6),
    "RETURNED_DATE" DATE DEFAULT SYSDATE,
    "RETURNED_BY" VARCHAR2(100) NOT NULL ENABLE,
    "REMARKS" VARCHAR2(100),
    CONSTRAINT "PK_CAPEX_RETURN_ITEM" PRIMARY KEY ("SERIAL_NO")
    USING INDEX ENABLE
);

ALTER TABLE "CAPEX_RETURN_ITEM" ADD CONSTRAINT
"FK_CAPEX_RETURN_ITEM_CP_NO" FOREIGN KEY ("CP_NO")
    REFERENCES "CAPEX_PROJECTS" ("CP_NO") ENABLE;
ALTER TABLE "CAPEX_RETURN_ITEM" ADD CONSTRAINT
"FK_CAPEX_RETURN_ITEM_ITEM_CODE" FOREIGN KEY ("ITEM_CODE")
    REFERENCES "CAPEX_MASTER" ("ITEM_CODE") ENABLE;

```

CaPeX RETURN ITEM PAGE

The screenshot shows the Oracle APEX App Builder interface for designing a page. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and user profile 'Rishav Prasad' are also present. The main area displays the 'CAPEX RETURN ITEM FORM' page. The left sidebar shows the page structure with components like 'Pre-Rendering', 'Components' (Body, Region Body), and various input fields labeled P39_SERIAL_NO, P39_CP_NO, P39_ITEM_CODE, etc. The right sidebar contains sections for 'Identification' (Name: CAPEX RETURN ITEM FORM, Alias: capex-return-item-form, Title: CAPEX RETURN ITEM FORM, Page Group: Select), 'Appearance' (Page Mode: Normal, Page Template: Theme Default, Template Options: Use Template Defaults), and 'Navigation Menu'. The central body area shows a form with fields for item details and a footer with buttons for CLOSE, DELETE, CHANGE, CANCEL, DRAFT, and SAVE.

CaPeX OPENING BALANCE REPORT PAGE

The screenshot shows the Oracle APEX application running. The left sidebar menu includes 'P&B IWYC' and several report options: 'CAPEX PROJECT DETAILS', 'CREATE A PROJECT', 'CAPEX MASTER REPORT', 'CAPEX ISSUE ITEM REPORT', 'CAPEX ISSUE ITEM FORM', 'CAPEX RECEIVE ITEM RE...', 'CAPEX RECEIVE ITEM FO...', 'CAPEX RETURN ITEM RE...', 'CAPEX RETURN ITEM FO...', 'CAPEX OPENING BALAN...', and 'CAPEX ONHAND QUANTI...'. The main content area displays a report titled 'CAPEX OPENING BALANCE REPORT' with a table showing transaction details. The table has columns: Cp No, Item Code, Transaction Type, Quantity Before, Quantity(ISS,REC,RET), Quantity After, Rate, and Transaction Date. The data shows five transactions for item code C016, with dates ranging from 6/6/2024 to 6/9/2024. The bottom of the screen shows standard APEX navigation and status bars.

Cp No	Item Code	Transaction Type	Quantity Before	Quantity(ISS,REC,RET)	Quantity After	Rate	Transaction Date
ITD/CP/CATIB/24/0118	C016	REC	0	10	10		6/6/2024
ITD/CP/CATIB/24/0118	C016	ISS	10	5	5		6/6/2024
ITD/CP/CATIB/24/0118	C016	ISS	5	1	4		6/6/2024
ITD/CP/CATIB/24/0118	C016	ISS	4	2	2		6/8/2024
ITD/CP/CATIB/24/0118	C016	RET	2	5	7		6/9/2024

CaPeX OPENING BALANCE TABLE STRUCTURE

```

CREATE TABLE "CAPEX_OPENING_BALANCE"
(
    "TRANSACTION_ID" NUMBER GENERATED BY DEFAULT AS IDENTITY
        MINVALUE 1 MAXVALUE 99999999999999999999999999999999 INCREMENT BY 1
        START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL
        ENABLE,
    "CP_NO" VARCHAR2(50) NOT NULL ENABLE,
    "ITEM_CODE" VARCHAR2(6) NOT NULL ENABLE,
    "TRANSACTION_TYPE" VARCHAR2(3),
    "QUANTITY_BEFORE" NUMBER,
    "QUANTITY" NUMBER NOT NULL ENABLE,
    "QUANTITY_AFTER" NUMBER,
    "RATE" NUMBER,
    "TRANSACTION_DATE" DATE DEFAULT SYSDATE,
    CONSTRAINT "CHK_CAPEX_TRANSACTION_TYPE" CHECK
    (TRANSACTION_TYPE IN ('ISS', 'REC', 'RET')) ENABLE,
    PRIMARY KEY ("TRANSACTION_ID")
    USING INDEX ENABLE
);

```

CaPeX OPENING BALANCE PAGE

The screenshot shows the Oracle APEX App Builder interface in the Page Designer mode. The page title is "Page 33: CAPEX OPENING BALANCE". The main content area displays a report titled "CAPEX OPENING BALANCE REPORT" with a single row view selected. The left sidebar shows the report structure with columns: TRANSACTION_ID, CP_NO, ITEM_CODE, TRANSACTION_TYPE, QUANTITY_BEFORE, QUANTITY, QUANTITY_AFTER, RATE, and TRANSACTION_DATE. The right sidebar displays layout settings for the column.

CaPeX OPENING BALANCE TRIGGERS

CAPEX ISSUE ITEM TRIGGER

```

CREATE OR REPLACE EDITIONABLE TRIGGER "CAPEX_ISSUE_ITEM_TRG"
BEFORE INSERT ON CAPEX_ISSUE_ITEM
FOR EACH ROW
BEGIN
    IF :NEW.SERIAL_NO IS NULL THEN
        SELECT CAPEX_ISSUE_ITEM_SEQ.NEXTVAL INTO :NEW.SERIAL_NO FROM DUAL;
    END IF;
END;
/
ALTER TRIGGER "CAPEX_ISSUE_ITEM_TRG" ENABLE;
CREATE OR REPLACE EDITIONABLE TRIGGER "TR_CAPEX_ISSUE_ITEM_UPDATE_OB"
FOR INSERT ON CAPEX_ISSUE_ITEM
COMPOUND TRIGGER

TYPE t_issue_item_rec IS RECORD (
    item_code VARCHAR2(10),
    cp_no VARCHAR2(50),
    issue_quantity NUMBER
);

TYPE t_issue_item_tab IS TABLE OF t_issue_item_rec;
g_issue_items t_issue_item_tab := t_issue_item_tab();

AFTER EACH ROW IS
BEGIN
    g_issue_items.EXTEND;
    g_issue_items(g_issue_items.COUNT) :=
    t_issue_item_rec(:NEW.ITEM_CODE, :NEW.CP_NO, :NEW.ISSUE_QUANTITY);
END AFTER EACH ROW;

AFTER STATEMENT IS
BEGIN
    FOR i IN 1 .. g_issue_items.COUNT LOOP
        DECLARE
            l_quantity_before NUMBER;
            l_quantity_after NUMBER;
        BEGIN
            -- Fetch the last quantity after from the CAPEX_OPENING_BALANCE for this item code
            and CP_NO
            SELECT COALESCE(
                (SELECT QUANTITY_AFTER
                 FROM (SELECT QUANTITY_AFTER
                       FROM CAPEX_OPENING_BALANCE
                       WHERE ITEM_CODE = g_issue_items(i).item_code
                           AND CP_NO = g_issue_items(i).cp_no
                           ORDER BY TRANSACTION_ID DESC)
                  WHERE ROWNUM = 1),
                (SELECT ISSUE_QUANTITY
                 FROM CAPEX_ISSUE_ITEM
                 WHERE ITEM_CODE = g_issue_items(i).item_code AND ROWNUM = 1)
            ) INTO l_quantity_before
            FROM DUAL;

            -- Calculate the quantity after the issue
            l_quantity_after := l_quantity_before - g_issue_items(i).issue_quantity;

            -- Insert into CAPEX_OPENING_BALANCE
            INSERT INTO CAPEX_OPENING_BALANCE (
                ITEM_CODE, CP_NO, TRANSACTION_TYPE, QUANTITY_BEFORE, QUANTITY, QUANTITY_AFTER,
                TRANSACTION_DATE
            ) VALUES (
                g_issue_items(i).item_code, g_issue_items(i).cp_no, 'ISS', l_quantity_before,
                g_issue_items(i).issue_quantity, l_quantity_after, SYSDATE
            );
        END;
    END LOOP;
END;

```

CAPEX RECEIVE ITEM TRIGGER

```

CREATE OR REPLACE EDITIONABLE TRIGGER "CAPEX_RECEIVE_ITEM_TRG"
BEFORE INSERT ON CAPEX_RECEIVE_ITEM
FOR EACH ROW
BEGIN
    IF :NEW.SERIAL_NO IS NULL THEN
        SELECT CAPEX_RECEIVE_ITEM_SEQ.NEXTVAL INTO :NEW.SERIAL_NO FROM DUAL;
    END IF;
END;
/
ALTER TRIGGER "CAPEX_RECEIVE_ITEM_TRG" ENABLE;
CREATE OR REPLACE EDITIONABLE TRIGGER "TR_CAPEX_RECEIVE_ITEM_UPDATE_OB"
FOR INSERT ON CAPEX_RECEIVE_ITEM
COMPOUND TRIGGER

TYPE t_receive_item_rec IS RECORD (
    item_code VARCHAR2(10),
    cp_no VARCHAR2(50),
    receive_quantity NUMBER
);
TYPE t_receive_item_tab IS TABLE OF t_receive_item_rec;
g_receive_items t_receive_item_tab := t_receive_item_tab();

AFTER EACH ROW IS
BEGIN
    g_receive_items.EXTEND;
    g_receive_items(g_receive_items.COUNT) :=
t_receive_item_rec(:NEW.ITEM_CODE, :NEW.CP_NO, :NEW.RECEIVE_QUANTITY);
END AFTER EACH ROW;

AFTER STATEMENT IS
BEGIN
    FOR i IN 1 .. g_receive_items.COUNT LOOP
        DECLARE
            l_quantity_before NUMBER;
            l_quantity_after NUMBER;
        BEGIN
            -- Fetch the last quantity after from the CAPEX_OPENING_BALANCE for this item code
and CP_NO
            SELECT COALESCE(
                (SELECT QUANTITY_AFTER
                 FROM (SELECT QUANTITY_AFTER
                       FROM CAPEX_OPENING_BALANCE
                       WHERE ITEM_CODE = g_receive_items(i).item_code
                           AND CP_NO = g_receive_items(i).cp_no
                           ORDER BY TRANSACTION_ID DESC)
                  WHERE ROWNUM = 1),
                0 -- Assuming 0 as default initial stock if no previous record exists
            ) INTO l_quantity_before
            FROM DUAL;

            -- Calculate the quantity after the receive
            l_quantity_after := l_quantity_before + g_receive_items(i).receive_quantity;

            -- Insert into CAPEX_OPENING_BALANCE
            INSERT INTO CAPEX_OPENING_BALANCE (
                ITEM_CODE, CP_NO, TRANSACTION_TYPE, QUANTITY_BEFORE, QUANTITY, QUANTITY_AFTER,
                TRANSACTION_DATE
            ) VALUES (
                g_receive_items(i).item_code, g_receive_items(i).cp_no, 'REC',
                l_quantity_before, g_receive_items(i).receive_quantity, l_quantity_after, SYSDATE
            );
        END;
    END LOOP;
END;

```

CAPEX RETURN ITEM TRIGGER

```

CREATE OR REPLACE EDITIONABLE TRIGGER "CAPEX_RETURN_ITEM_TRG"
BEFORE INSERT ON CAPEX_RETURN_ITEM
FOR EACH ROW
BEGIN
    IF :NEW.SERIAL_NO IS NULL THEN
        SELECT CAPEX_RETURN_ITEM_SEQ.NEXTVAL INTO :NEW.SERIAL_NO FROM DUAL;
    END IF;
END;
/
ALTER TRIGGER "CAPEX_RETURN_ITEM_TRG" ENABLE;
CREATE OR REPLACE EDITIONABLE TRIGGER "TR_CAPEX_RETURN_ITEM_UPDATE_OB"
FOR INSERT ON CAPEX_RETURN_ITEM
COMPOUND TRIGGER

TYPE t_return_item_rec IS RECORD (
    item_code VARCHAR2(10),
    cp_no VARCHAR2(50),
    return_quantity NUMBER
);
TYPE t_return_item_tab IS TABLE OF t_return_item_rec;
g_return_items t_return_item_tab := t_return_item_tab();

AFTER EACH ROW IS
BEGIN
    g_return_items.EXTEND;
    g_return_items(g_return_items.COUNT) :=
t_return_item_rec(:NEW.ITEM_CODE, :NEW.CP_NO, :NEW.RETURN_QUANTITY);
END AFTER EACH ROW;

AFTER STATEMENT IS
BEGIN
    FOR i IN 1 .. g_return_items.COUNT LOOP
        DECLARE
            l_quantity_before NUMBER;
            l_quantity_after NUMBER;
        BEGIN
            -- Fetch the last quantity after from the CAPEX_OPENING_BALANCE for this item code
and CP_NO
            SELECT COALESCE(
                (SELECT QUANTITY_AFTER
                 FROM (SELECT QUANTITY_AFTER
                         FROM CAPEX_OPENING_BALANCE
                         WHERE ITEM_CODE = g_return_items(i).item_code
                             AND CP_NO = g_return_items(i).cp_no
                         ORDER BY TRANSACTION_ID DESC)
                         WHERE ROWNUM = 1),
                0 -- Assuming 0 as default initial stock if no previous record exists
            ) INTO l_quantity_before
            FROM DUAL;

            -- Calculate the quantity after the return
            l_quantity_after := l_quantity_before + g_return_items(i).return_quantity;

            -- Insert into CAPEX_OPENING_BALANCE
            INSERT INTO CAPEX_OPENING_BALANCE (
                ITEM_CODE, CP_NO, TRANSACTION_TYPE, QUANTITY_BEFORE, QUANTITY, QUANTITY_AFTER,
                TRANSACTION_DATE
            ) VALUES (
                g_return_items(i).item_code, g_return_items(i).cp_no, 'RET',
                l_quantity_before, g_return_items(i).return_quantity, l_quantity_after, SYSDATE
            );
        END;
    END LOOP;
END;

```

CaPeX ONHAND REPORT

The screenshot shows a web-based application interface for an Inventory Management System. The URL in the browser is <https://apex.oracle.com/pls/apex/r/rishav07/inventory-application/capex-onhand-quantity?session=111201782438766>. The main content area is titled "CaPeX ONHAND REPORT". On the left, there is a sidebar menu with the following items:

- P&B IWYC
- CAPEX PROJECT DETAILS** (highlighted in red)
- CREATE A PROJECT
- CAPEX MASTER REPORT** (highlighted in red)
- CAPEX ISSUE ITEM REPORT
- CAPEX ISSUE ITEM FORM
- CAPEX RECEIVE ITEM RE...
- CAPEX RECEIVE ITEM FO...
- CAPEX RETURN ITEM RE...
- CAPEX RETURN ITEM FO...
- CAPEX OPENING BALAN...
- CAPEX ONHAND QUANTI...

The main content area displays a table with the following data:

Cp No	Item Code	Item Name	Quantity
ITD/CP/CAT1B/24/0118	C016	12U RACK	7

Below the table, it says "1 - 1". At the bottom of the page, there is a note "Release 10" and a footer with links for "Debug", "Quick Edit", "Customize", and "Help".

CaPeX ONHAND TABLE STRUCTURE

```

CREATE TABLE "CAPEX_ONHAND_QUANTITY"
(
    "CP_NO" VARCHAR2(50) NOT NULL ENABLE,
    "ITEM_CODE" VARCHAR2(6) NOT NULL ENABLE,
    "ITEM_NAME" VARCHAR2(50),
    "QUANTITY" NUMBER,
    PRIMARY KEY ("CP_NO", "ITEM_CODE")
)
USING INDEX ENABLE
;

```

CaPeX ONHAND TRIGGER

```

create or replace TRIGGER TR_UPDATECAPEX_ONHAND_QUANTITY
AFTER INSERT OR UPDATE ON CAPEX_OPENING_BALANCE
FOR EACH ROW
BEGIN
    MERGE INTO CAPEX_ONHAND_QUANTITY coq
    USING (SELECT :NEW.CP_NO AS CP_NO,
                 :NEW.ITEM_CODE AS ITEM_CODE,
                 :NEW.QUANTITY_AFTER AS QUANTITY,
                 (SELECT ITEM_NAME FROM CAPEX_MASTER WHERE ITEM_CODE
      = :NEW.ITEM_CODE) AS ITEM_NAME
           FROM DUAL) src
    ON (coq.CP_NO = src.CP_NO AND coq.ITEM_CODE = src.ITEM_CODE)
    WHEN MATCHED THEN
        UPDATE SET coq.QUANTITY = src.QUANTITY, coq.ITEM_NAME =
src.ITEM_NAME
    WHEN NOT MATCHED THEN
        INSERT (coq.CP_NO, coq.ITEM_CODE, coq.ITEM_NAME,
coq.QUANTITY)
        VALUES (src.CP_NO, src.ITEM_CODE, src.ITEM_NAME,
src.QUANTITY);
    END;
/

```

CaPeX ONHAND STOCK PROCEDURE

```

create or replace FUNCTION get_capex_onhand_stock (
    p_item_code IN VARCHAR2,
    p_cp_no IN VARCHAR2
) RETURN NUMBER IS
    l_onhand_stock NUMBER;
BEGIN
    SELECT COALESCE(
        (SELECT QUANTITY_AFTER
         FROM CAPEX_OPENING_BALANCE
         WHERE ITEM_CODE = p_item_code
           AND CP_NO = p_cp_no
         ORDER BY TRANSACTION_ID DESC FETCH FIRST 1 ROWS
ONLY),
        0 -- Assuming 0 as default initial stock if no
previous record exists
    ) INTO l_onhand_stock
    FROM DUAL;

    RETURN l_onhand_stock;
END get_capex_onhand_stock;
/

```

P&B ALERT LOG PAGE

The screenshot shows a browser window for apex.oracle.com with the URL `apex.oracle.com/pls/apex/r/rishav07/inventory-application/p-b-alert-log?session=111201782438766`. The page title is "P&B ALERT LOG PAGE". The left sidebar contains a navigation menu with items like Home, P&B ALERT LOG, P&B MASTER DETAILS, etc. The main content area displays a table titled "P&B ALERT LOG" with the following data:

Item Code	Item Name	Quantity	Reorder Level	Status	Error Message	Alert Date
P012	FLEXIBLE PIPE 1IN	5	5	Alert Sent	Stock below reorder level	6/9/2024
P002	24 PORT SWITCH	5	5	Alert Sent	Stock below reorder level	6/9/2024
P012	FLEXIBLE PIPE 1IN	5	5	Alert Sent	Stock below reorder level	6/9/2024
P002	24 PORT SWITCH	5	5	Alert Sent	Stock below reorder level	6/9/2024
P012	FLEXIBLE PIPE 1IN	5	5	Alert Sent	Stock below reorder level	6/9/2024
P002	24 PORT SWITCH	5	5	Alert Sent	Stock below reorder level	6/9/2024
P012	FLEXIBLE PIPE 1IN	5	5	Alert Sent	Stock below reorder level	6/8/2024
P012	FLEXIBLE PIPE 1IN	5	5	Alert Sent	Stock below reorder level	6/9/2024
P002	24 PORT SWITCH	5	5	Alert Sent	Stock below reorder level	6/9/2024

P&B ALERT LOG

The screenshot shows the Oracle APEX App Builder interface. The top navigation bar includes "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The current page is "Page 19: P&B ALERT LOG". The left sidebar shows the page structure with components like "Pre-Rendering", "Components" (Body, Columns, etc.), and "Post-Rendering". The main workspace shows the "BODY" region with a table structure and a column "ITEM_NAME" highlighted. The right panel displays configuration options for the selected column:

- Identification:** Column Name: ITEM_NAME, Type: Plain Text
- Heading:** Heading: Item Name, Alignment: Center
- Single Row View:** Use Column Heading: On
- Layout:** Sequence: 2

P&B ALERT LOG PROCEDURE

```

CREATE OR REPLACE PROCEDURE CHECK_REORDER_LEVEL IS
    CURSOR reorder_cursor IS
        SELECT *
        FROM (
            SELECT I.ITEM_CODE, I.ITEM_NAME, I.REORDER_LEVEL,
                   NVL(I STOCK, 0) + NVL(R.RECV_QTY, 0) - NVL(ISS.ISSUE_QTY, 0) + NVL(RT.RETURN_QTY, 0) AS QUANTITY
            FROM IT_ITEMS I
            LEFT JOIN (
                SELECT ITEM_CODE, SUM(RECV_QTY) AS RECV_QTY
                FROM RECEIVE_ITEMS
                GROUP BY ITEM_CODE
            ) R ON I.ITEM_CODE = R.ITEM_CODE
            LEFT JOIN (
                SELECT ITEM_CODE, SUM(ISSUE_QTY) AS ISSUE_QTY
                FROM ISSUE_ITEMS
                GROUP BY ITEM_CODE
            ) ISS ON I.ITEM_CODE = ISS.ITEM_CODE
            LEFT JOIN (
                SELECT ITEM_CODE, SUM(RETURN_QTY) AS RETURN_QTY
                FROM RETURN_ITEMS
                GROUP BY ITEM_CODE
            ) RT ON I.ITEM_CODE = RT.ITEM_CODE
            WHERE I.REORDER_LEVEL > 0
        )
        WHERE QUANTITY <= REORDER_LEVEL;

    l_item_code IT_ITEMS.ITEM_CODE%TYPE;
    l_item_name IT_ITEMS.ITEM_NAME%TYPE;
    l_reorder_level IT_ITEMS.REORDER_LEVEL%TYPE;
    l_quantity NUMBER;

    -- Add a variable to hold the list of email addresses
    l_email_list VARCHAR2(4000);
BEGIN
    -- Fetch the email list from the ADMIN_EMAILS table
    SELECT LISTAGG(EMAIL, ',') WITHIN GROUP (ORDER BY EMAIL)
    INTO l_email_list
    FROM ADMIN_EMAILS;

    OPEN reorder_cursor;
    LOOP
        FETCH reorder_cursor INTO l_item_code, l_item_name, l_reorder_level, l_quantity;
        EXIT WHEN reorder_cursor%NOTFOUND;

        IF l_quantity <= l_reorder_level THEN
            -- Insert initial log entry with status "Scheduled at 8 AM"
            INSERT INTO REORDER_ALERT_LOG (ITEM_CODE, ITEM_NAME, QUANTITY, REORDER_LEVEL, STATUS,
            ERROR_MESSAGE, ALERT_DATE)
            VALUES (l_item_code, l_item_name, l_quantity, l_reorder_level, 'Scheduled at 8 AM', 'Stock
            below reorder level', SYSDATE);

            -- Send the email
            APEX_MAIL.SEND(
                p_to      => l_email_list, -- Use the dynamic email list
                p_from    => 'inventory@example.com',
                p_subj   => 'Reorder Alert',
                p_body   => 'Item Code: ' || l_item_code ||
                           ', Item Name: ' || l_item_name ||
                           ', Quantity: ' || l_quantity ||
                           ', Reorder Level: ' || l_reorder_level ||
                           '. Please reorder the item as soon as possible.'
            );
            -- Update the log entry status to "Alert Sent"
            UPDATE REORDER_ALERT_LOG
            SET STATUS = 'Alert Sent'
            WHERE ITEM_CODE = l_item_code
            AND ITEM_NAME = l_item_name
            AND ALERT_DATE = (SELECT MAX(ALERT_DATE) FROM REORDER_ALERT_LOG WHERE ITEM_CODE = l_item_code);
        END IF;
    END LOOP;
    CLOSE reorder_cursor;
END;
/

```

CONCLUSION

The Development of the Inventory Management System for the IT Department At Hindalco Industries Ltd. Has Successfully addressed the challenges associated with managing Inventory across multiple systems. The system provides a user-friendly interface for authorized users to efficiently track, control, and report Inventory Data. By leveraging technologies such as HTML, CSS, Oracle APEX, MySQL, PL/SQL, the system has streamlined inventory management processes and improved overall efficiency within the department.

Through the development and implementation of the Inventory Management System, several key achievements have been realized. This include:

- Improved Inventory Control and Accuracy: The system allows for Real Time Tracking of Inventory Items, ensuring better control and reducing instances of stockouts or excess inventory.
- Enhanced Reporting Capabilities: With the system's reporting functionality, users can generate detailed reports on Inventory status, transactions, and usage enabling informed decision-making.
- Increased Productivity : Automation of Inventory Management process has reduced manual efforts, enabling IT Department personnel to focus on other critical tasks.

The Successful deployment and utilization of the Inventory Management System have brought tangible benefits to Hindalco Industries Ltd's IT Department, fostering operational efficiency and cost optimization.

FUTURE SCOPE

While the Current Version of the Inventory Management System fulfills the Initial requirements, there is a scope for further enhancements and expansions. Some potential areas for future development include:

Integration with external System:

Integration with Procurement systems to automate the inventory replenishment process.

Integration with financial systems for better cost Tracking and financial analysis.

Advanced Analytics and Forecasting:

Incorporating data analysis and forecasting techniques to predictt inventory demand, optimize reorder points, and improve overall inventory Planning.

Mobile Application Development:

Creating mobile application for inventory management, providing on-the-go access and real time updates to authorized users.

Supplier Collaboration:

Implementing Features that enables better collaboration with Suppliers, facilitating efficient communication and streamlining the procurement process.

These Future enhancements would further strengthen the inventory management system, allowing Hindalco Industries Ltd's Department to maximize efficiency, reduce costs, and adapt to evolving business needs.