**Employee Training Tracker: Project Plan (Phases 4-10)**

**## Phase 4: Business Logic & Automation**

**Goal:** To automate processes, enforce data quality, and reduce manual work using Salesforce's point-and-click tools.

**### 1. Create Roll-Up Summary Fields**

* **On Employee\_\_c:** Create a field Total Trainings Completed that **COUNTs** Enrollment\_\_c records where **Status\_\_c = "Completed"**.
* **On Training\_\_c:** Create a field Total Enrolled that **COUNTs** all related Enrollment\_\_c records.

**### 2. Create a Validation Rule**

* **On Enrollment\_\_c:** Create a rule named Score\_Only\_When\_Completed.
* **Error Formula:** AND(NOT(ISBLANK(Score\_\_c)), NOT(ISPICKVAL(Status\_\_c, "Completed")))
* **Error Message:** "You can only enter a score after the training is completed."

**### 3. Create a Record-Triggered Flow**

* **Name:** Update Employee on Training Completion
* **Trigger:** When an **Enrollment\_\_c** record is **updated** and the **Status\_\_c = "Completed"**.
* **Action:** Update the related parent **Employee\_\_c** record.
* **Field Update:** Set a new date field on Employee, Last\_Training\_Completed\_Date\_\_c, to the current date ($Flow.CurrentDate).

**## Phase 5: Security & Reporting**

**Goal:** To control data access for different users and visualize the data for managers.

**### 1. Configure Permission Set**

* **Permission Set:** HR\_Permissions
* **Action:** Go to **Object Settings** within the permission set.
* **Assign Access:** Grant **Read, Create, and Edit** access to the Employee\_\_c, Training\_\_c, Certification\_\_c, and Enrollment\_\_c objects.
* **Assign to Users:** Ensure this is assigned to your hr.manager@example.com test user.

**### 2. Create a Custom Report Type**

* **Name:** Employees with Enrollments
* **Primary Object:** Employee\_\_c
* **Related Object:** Enrollment\_\_c (Select the option "Each 'A' record must have at least one related 'B' record").

**### 3. Build Reports**

* **Report 1:** Training Status by Department
  + **Report Type:** Employees with Enrollments
  + **Format:** Matrix
  + **Group Rows:** Department\_\_c, Status\_\_c
  + **Group Columns:** Training: Training Name
* **Report 2:** All In Progress Trainings
  + **Report Type:** Enrollments
  + **Filter:** Status\_\_c **Equals** In Progress

**### 4. Build a Dashboard**

* **Name:** HR Training Dashboard
* **Component 1:** Use the Training Status by Department report to create a **Donut Chart** showing the count of enrollments by status.
* **Component 2:** Use the All In Progress Trainings report to create a **Lightning Table** showing a list of employees and their active courses.

**## Phase 6: User Experience (UX) Enhancement**

**Goal:** To create a branded, user-friendly experience using the Lightning App Builder.

**### 1. Create a Custom Lightning App**

* **Name:** Training Tracker
* **Action:** Go to **Setup > App Manager > New Lightning App**.
* **Navigation Items:** Add the **Home**, **Employees**, **Trainings**, **Certifications**, **Reports**, and **Dashboards** tabs.
* **Assign:** Assign this app to your **System Administrator** and **HR\_Permissions** profiles/permission sets.

**### 2. Create a Custom Employee Record Page**

* **Name:** Employee Record Page
* **Action:** Go to **Object Manager > Employee > Lightning Record Pages > New**.
* **Template:** Use **Header and Right Sidebar**.
* **Layout:**
  + Place the **Highlights Panel** in the header.
  + Place the **Record Detail** component in the main (left) section.
  + Place the **Related Lists** component in the right sidebar.
* **Activation:** Activate this page as the **Org Default** for the Employee\_\_c object.

**### 3. Create a Custom App Home Page**

* **Name:** Training Home Page
* **Action:** Go to **Setup > Lightning App Builder > New > App Page**.
* **Layout:** Add the **Dashboard** component and configure it to display your HR Training Dashboard. Add a **Recent Items** component to show recently viewed records.
* **Activation:** Add this page as the first tab in your **Training Tracker** Lightning App.

**## Phase 7: Custom Code & Deployment**

**Goal:** To implement complex logic that is not possible with flows and to package the entire project for deployment from Sandbox to Production.

**### 1. Create an Apex Trigger**

* **Name:** EnrollmentTrigger on the Enrollment\_\_c object.
* **Event:** before insert
* **Logic:** To prevent a user from being enrolled in the same active training more than once. (See Phase 8 for a more detailed handler class).

**### 2. Create an Apex Test Class**

* **Name:** EnrollmentTriggerTest
* **Action:** Write a test method that creates sample data and tries to insert a duplicate enrollment.
* **Assert:** Use System.assert() to confirm that the duplicate enrollment fails as expected.
* **Goal:** Achieve at least 75% code coverage for your EnrollmentTrigger (and its handler).

**### 3. Create an Outbound Change Set**

* **Action:** In your **Sandbox**, go to **Setup > Outbound Change Sets > New**.
* **Name:** Employee Tracker v1.0
* **Add All Components:** Manually add every component you have built:
  + **Custom Objects** (all 4)
  + **Custom Fields** (all of them)
  + **Flow** (Update Employee on Training Completion)
  + **Validation Rule**
  + **Permission Set** (HR\_Permissions)
  + **Custom Report Type** (Employees with Enrollments)
  + **Reports & Dashboard**
  + **Lightning App** (Training Tracker)
  + **Lightning Pages** (Employee Page, Home Page)
  + **Apex Class & Apex Trigger** (and your test class)
* **Upload:** Upload the completed change set to your Production org.

**### 4. Deploy to Production**

* **Action:** In your **Production** org, go to **Setup > Inbound Change Sets**.
* **Validate:** Find your change set and click **Validate**. This will run all your Apex tests.
* **Deploy:** Once validation succeeds, click **Deploy**.

**## Phase 8: Advanced UX with Lightning Web Components (LWC)**

**Goal:** To build a fully custom component that improves user efficiency beyond what standard layouts allow.

**### 1. Create an Apex Controller**

* **Name:** TrainingEnrollmentController
* **Methods:**
  + @AuraEnabled(cacheable=true) method to search for employees *not* already enrolled in a specific training.
  + @AuraEnabled method to receive a list of Employee IDs and a Training ID, and create new Enrollment\_\_c records.

**### 2. Create an LWC**

* **Name:** quickEnrollmentManager
* **HTML:** Build a component with a search bar, a data table to show results, and an "Enroll Selected" button.
* **JavaScript:** Import the Apex methods. Wire the search bar to the search method. On button click, call the enrollment method.
* **XML:** Expose the component to be used on a lightning\_\_RecordPage for the Training\_\_c object.

**### 3. Update the Training Record Page**

* **Action:** Edit the Training\_\_c Lightning Record Page.
* **Layout:** Add a new tab called "Mass Enroll" and drag your quickEnrollmentManager LWC into it.

**## Phase 9: Asynchronous Apex & Integration**

**Goal:** To handle large data processing and connect Salesforce to an external system.

**### 1. Create a Batch Apex Class**

* **Name:** BatchUpdateExpiredCerts
* **Logic:**
  + start(): Queries for all Certification\_\_c records where Expiry\_Date\_\_c < TODAY and Certification\_Status\_\_c = 'Active'.
  + execute(): Updates this batch of records to set Certification\_Status\_\_c = 'Expired'.
  + finish(): (Optional) Send an email to the admin.

**### 2. Schedule the Batch Job**

* **Action:** Create a new **Schedulable Apex Class** that calls Database.executeBatch(new BatchUpdateExpiredCerts()).
* **Schedule:** In **Setup > Apex Classes**, schedule this class to run nightly (e.g., at 2 AM).

**### 3. Create an Asynchronous REST Callout**

* **Use Case:** When a new Employee is hired in the 'IT' department, create a user account in an external system (use a free mock API like https://reqres.in/api/users).
* **Apex Class:** Create a class ITHelpdeskIntegration with an @future(callout=true) method. This method builds an HTTP request and sends a 'POST' callout with the employee's name and email.
* **Apex Trigger:** Create/modify the EmployeeTrigger to run after insert. If the new Employee\_\_c.Department\_\_c == 'IT', call the @future method.

**## Phase 10: Go-Live & Post-Production**

**Goal:** To successfully launch the application to users, manage the initial data load, and establish a plan for long-term support.

**### 1. Data Migration**

* **Action:** Prepare CSV files of all existing employees, training courses, and historical certifications.
* **Tool:** Use **Salesforce Data Loader** to upload the data.
* **Order:** You must load data in the correct order:
  1. Employee\_\_c
  2. Training\_\_c
  3. Certification\_\_c
  4. Enrollment\_\_c (using the Salesforce IDs from the previous files).

**### 2. User Training**

* **Deliverable:** Create a 1-2 page user guide (or "Quick Reference Guide") for the HR Manager.
* **Action:** Conduct a live training session to walk them through:
  + Creating new employees.
  + Enrolling employees using the standard related list.
  + Enrolling employees using the "Mass Enroll" LWC.
  + Reading the HR Training Dashboard.

**### 3. Support & Iteration**

* **Support:** Establish a "hypercare" period (e.g., the first two weeks post-launch) for immediate bug fixes.
* **Iteration:** Create a "v2.0" backlog. Gather feedback from your HR users on new features they'd like to see (e.g., "automated email reminders for expiring certifications").