**Employee Recognition App**

**Phase 1: Problem Understanding & Industry Analysis**

**Problem Statement:**

In many organizations, recognition is informal and untracked, leaving HR unable to measure engagement or identify top performers. A Salesforce-based Employee Recognition App will allow employees to send Kudos, track points, and provide managers with dashboards for visibility.

**1.Requirement Gathering**

* Primary Requirement: Build a system inside Salesforce where employees can recognize peers by sending “Kudos.”
* Detailed Needs:
  + Employees should be able to select a colleague, choose a recognition type (e.g., Teamwork, Innovation), and add a short message.
  + The recipient should be notified instantly (email or Salesforce notification).
  + Each Kudos should add points to the recipient’s profile.
  + HR managers should be able to track recognition trends, top performers, and engagement data.
  + Optional: Integrate with collaboration tools (Slack, Teams) for real-time recognition announcements.

**2.Stakeholder Analysis**

* **Employees:** Main users of the app → They create Kudos.
* **Managers/HR:** Consumers of reports and dashboards → They use insights to reward and recognize employees formally.
* **Salesforce Admin:** Configures objects, fields, automation, dashboards.
* **Salesforce Developer:** Builds Apex trigger, custom LWC for “Kudos Wall,” or Slack integration if required.

Stakeholder mapping ensures who benefits, who manages, who builds.

**3.Business Process Mapping**

Current Process (Before App):

* Recognition happens informally (verbal thanks, emails, or chats).
* No tracking, so HR cannot measure engagement or reward data-driven recognition.

Future Process (With Salesforce App):

1. Employee A opens “Give Kudos” form.
2. Selects recipient → adds message + type → submits.
3. Kudos record created → Flow/Apex updates points → recipient notified.
4. Reports/Dashboards updated automatically.

This mapping shows how Salesforce streamlines and digitizes the process.

**4.Industry-Specific Use Case Analysis**

* **HR & Employee Engagement:** Many companies invest in recognition platforms to improve morale. This project is like a mini version of Bonusly/Reward Gateway, but built natively in Salesforce.
* **Corporate Culture:** Recognition improves teamwork, reduces attrition, and encourages healthy competition.
* **Scalability:** If the org expands, the same app can integrate with payroll or HR systems to influence appraisals.

**5.AppExchange Exploration**

* Analysis shows they may have licensing costs or too many features for a small org.
* Decision: Build a lightweight in-house custom app tailored to company needs.

# **Phase 2: Org Setup & Configuration**

# **1. Salesforce Edition & Org Setup**

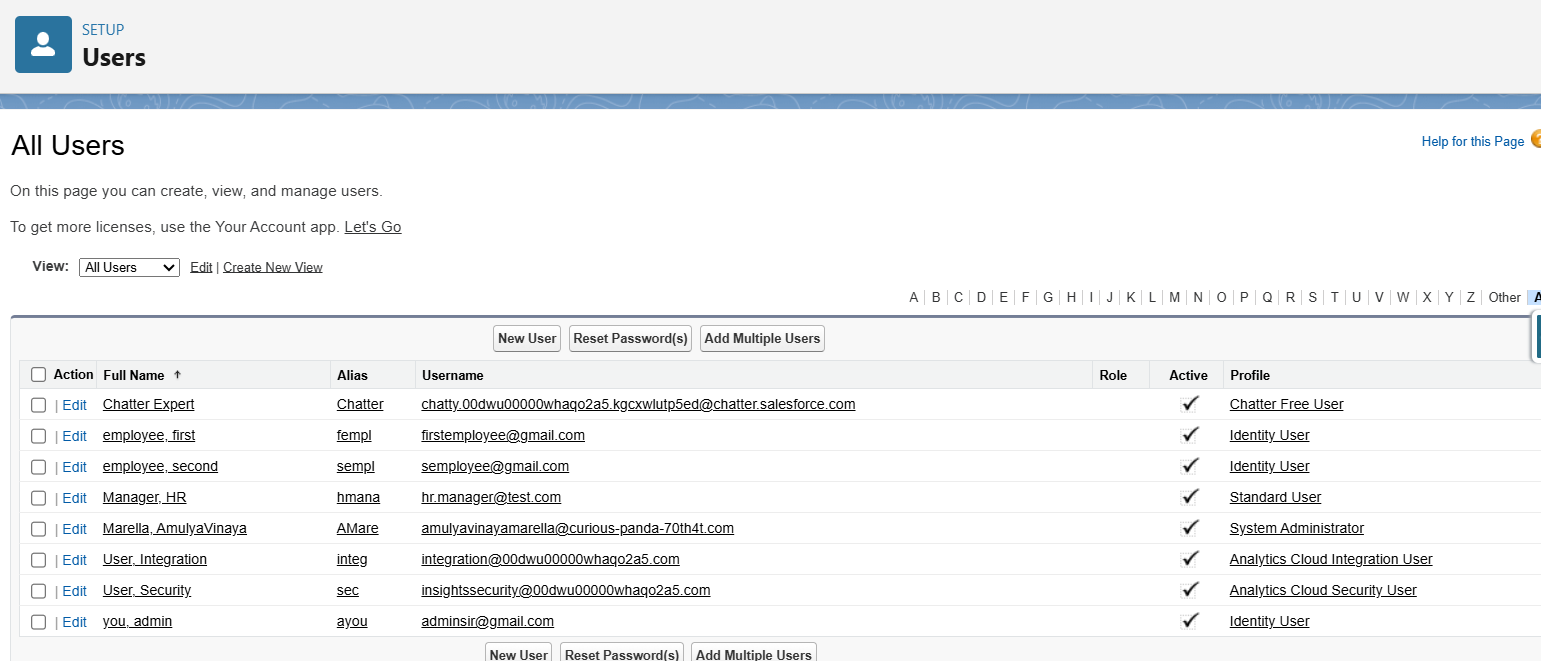
* Use a **Developer Edition** (for practice) or Sandbox (if in an enterprise org).
* Define **Company Information** (e.g., Company Name, Locale, Fiscal Year).
* Set **Business Hours & Holidays** (useful if recognition reports are time-based).

### **2. User Setup & Licenses**

* Create sample **user accounts** representing Employees, HR Managers, and Admins.
* Assign **Salesforce Platform licenses** (sufficient for custom apps).

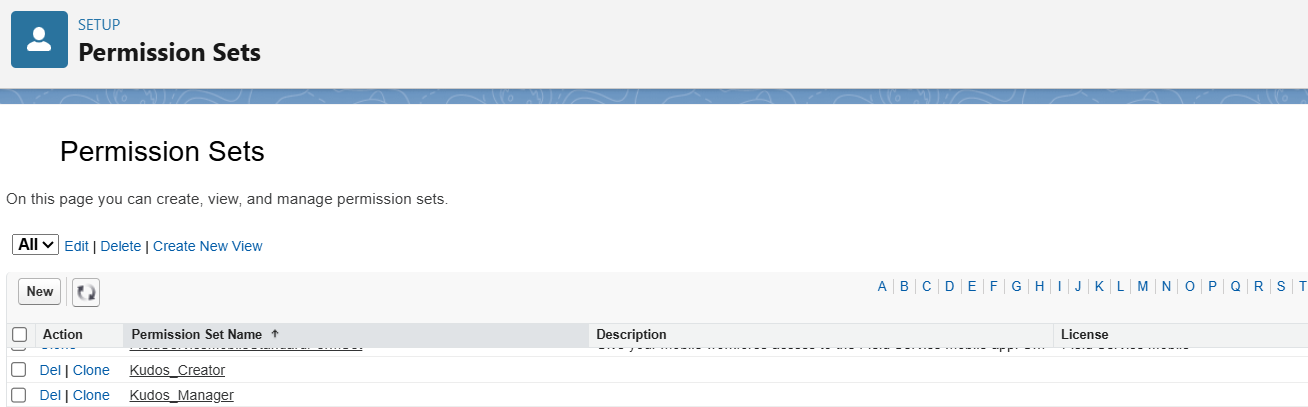
Example:

* Employee Users → Standard access.
* HR Managers → Extra reporting/dashboard permissions.
* Admin → Full system access.



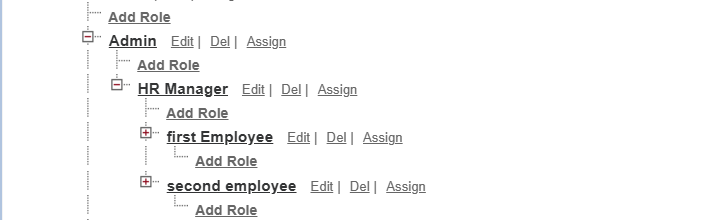
### **3. Profiles & Permission Sets**

* **Profiles** control baseline access:
  + Employee Profile → Can create Kudos, view own records.
  + HR Manager Profile → Can view all Kudos, run reports.
  + Admin Profile → Full CRUD rights.
* **Permission Sets** for extra flexibility:
  + “Kudos Dashboard Access” → Assign only to managers who need insights.
  + “Give Kudos” quick action → All employees.



**4. Roles & Hierarchy**

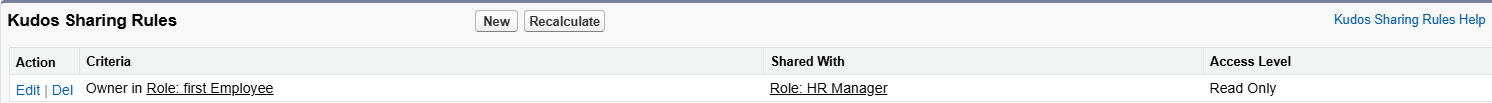
* Roles reflect org structure for visibility:
  + Admin (Top)
  + HR Manager
  + Employees



This ensures managers can see recognition given to their team, while employees only see their own.

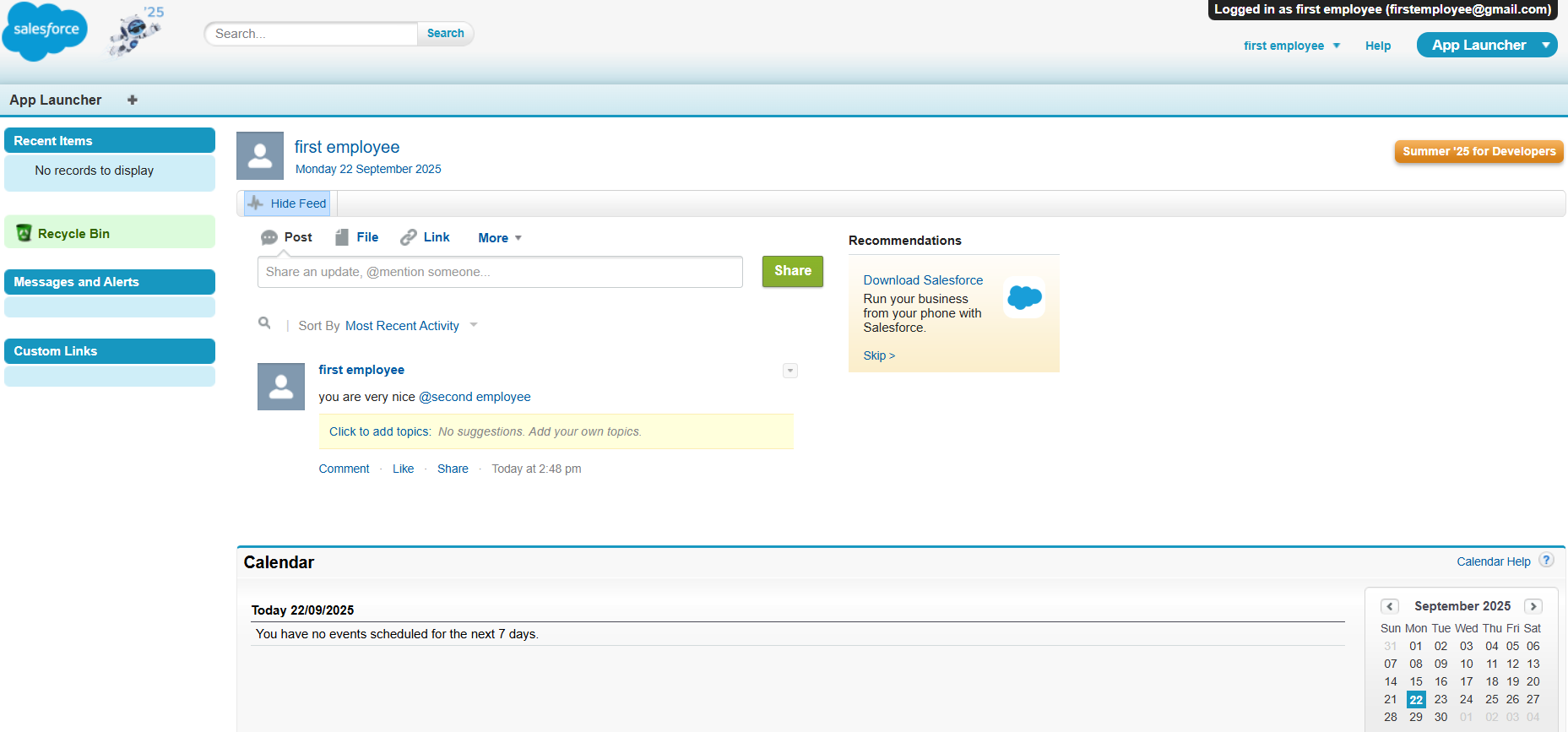
### **5. Org-Wide Defaults (OWD) & Sharing Rules**

* OWD for **Kudos Object** → Private (so employees can’t see all Kudos).
* **Sharing Rule:** HR Managers get read access to all Kudos for reporting.



### **6. Login Access Policies**

* Define security policies → trusted IP ranges for Admin login.
* Enable login-as feature for Admins to troubleshoot user issues.



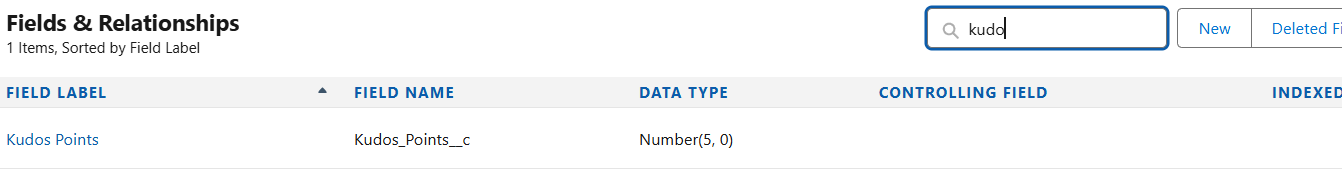
### **7. Sandbox / Deployment Basics**

* Development done in Sandbox/Dev Org.
* Features deployed to Production via **Change Sets.**

# **Phase 3: Data Modeling & Relationships**

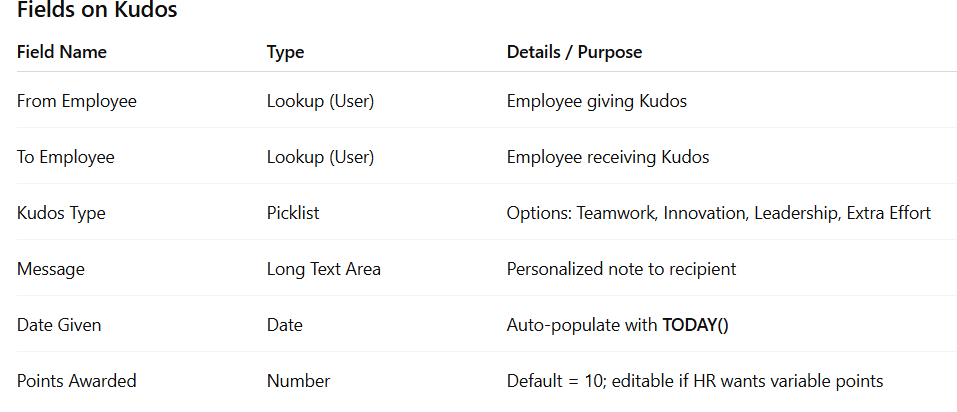
### **1. Standard Objects Used**

* **User (Standard Object)**: Already exists in Salesforce.
  + **Extension**: Add a custom field to track recognition:
    - **Kudos Points**
      * Type: Number
      * Default: 0
      * Purpose: Accumulates the points each employee earns from receiving Kudos.
  + This allows reports like “Top recognized employees” or dashboards showing employee engagement.



### **2. Custom Object: Kudos**

* Core object of the app → each record represents one recognition event.

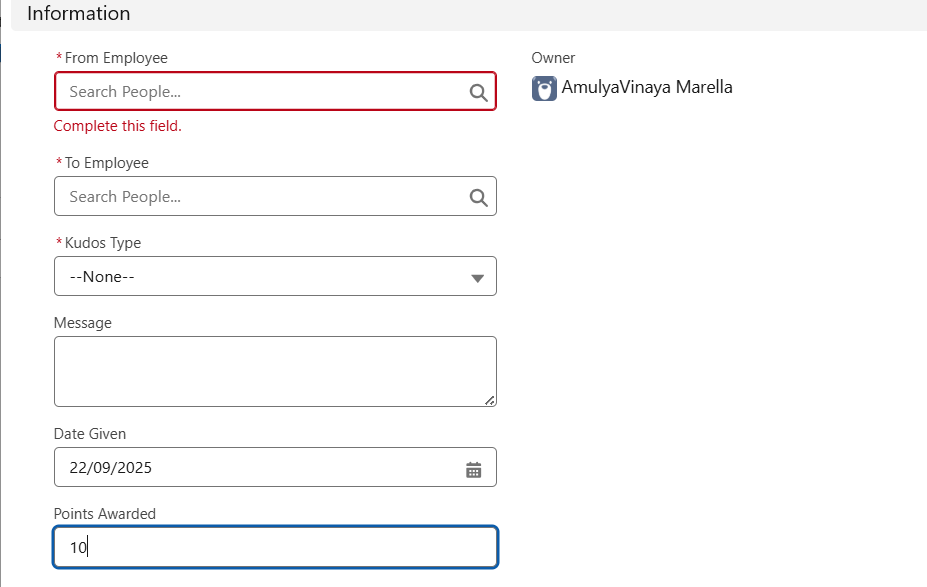


### **3. Relationships**

* **Lookup Relationship:**
  + From Kudos → User (From Employee).
  + From Kudos → User (To Employee).
* This makes Kudos a child record of the User object in two different roles.
* Benefit → You can run reports like “How many Kudos did each employee give/receive?”

### **4. Page Layouts & Record Types**

* **Page Layout (for Kudos):**
  + Show fields: From Employee, To Employee, Kudos Type, Message, Points.
  + Compact Layout for mobile → Show To Employee, Kudos Type, Points.
* **Record Types (Optional):**
  + Different types of recognition (e.g., “Peer Kudos,” “Manager Kudos”) with separate layouts.



### **5. Schema Builder (Visualization)**

In **Schema Builder**, we can see:

* User Object (Standard).
* Kudos Object (Custom).
* Lookup relationships linking Kudos to User (From + To).

This helps HR/Managers understand the data model visually.

### **6. Junction Objects (Not Needed Here, but Conceptual)**

* If we wanted to track **“Kudos given for a project”,** we could create a **Project object** and use Kudos as a junction object between User and Project.
* For now, a direct lookup between Kudos and User is sufficient.

### **7. External Objects (Optional Future Scope)**

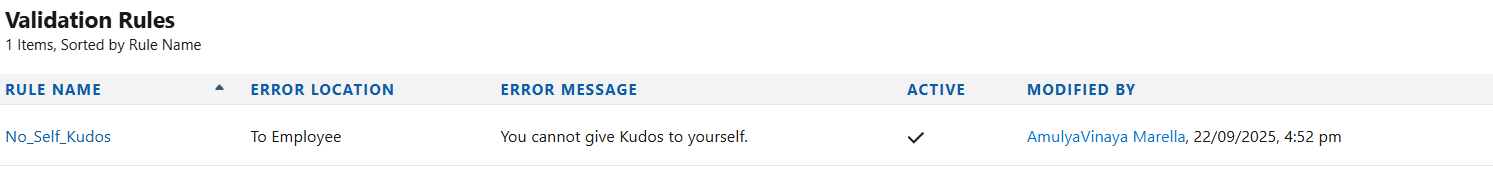
* If integrated with an external HR system (like Workday), we could create an **External Object** for HR data and relate Kudos with employee records outside Salesforce.

# **Phase 4: Process Automation**

### **1. Validation Rules**

* **Rule 1: No Self-Kudos**
  + Prevent employees from sending Kudos to themselves.
  + Formula Example:



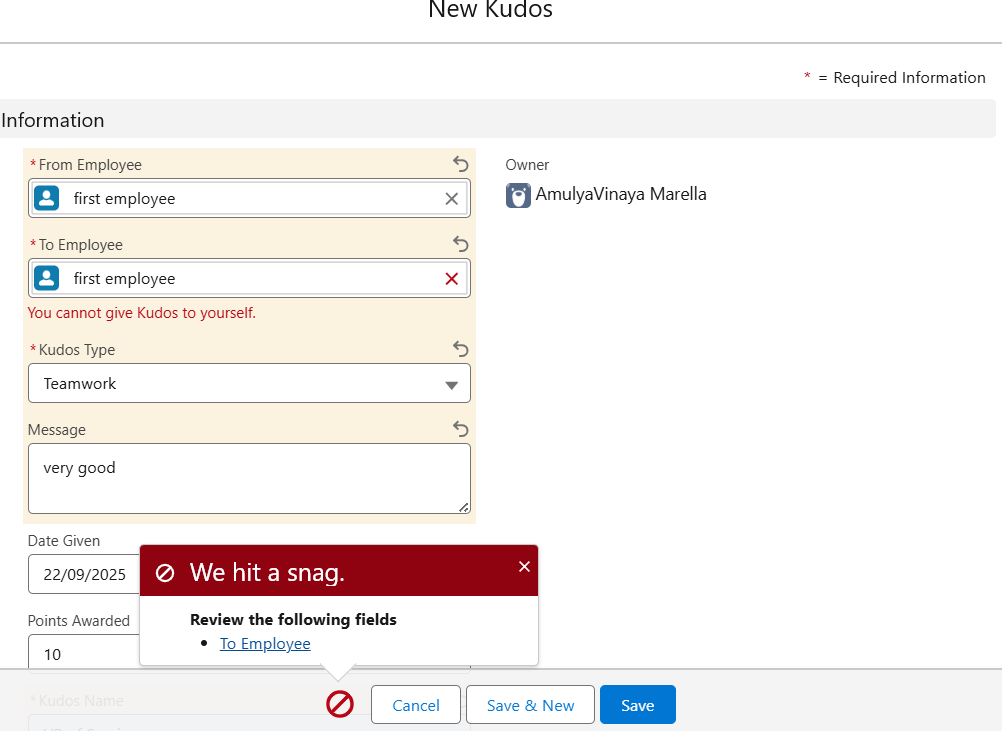


* + Error Message: “You cannot give Kudos to yourself.”
* **Rule 2: Mandatory Message**
  + Ensure that every Kudos has a recognition note.
  + Formula:



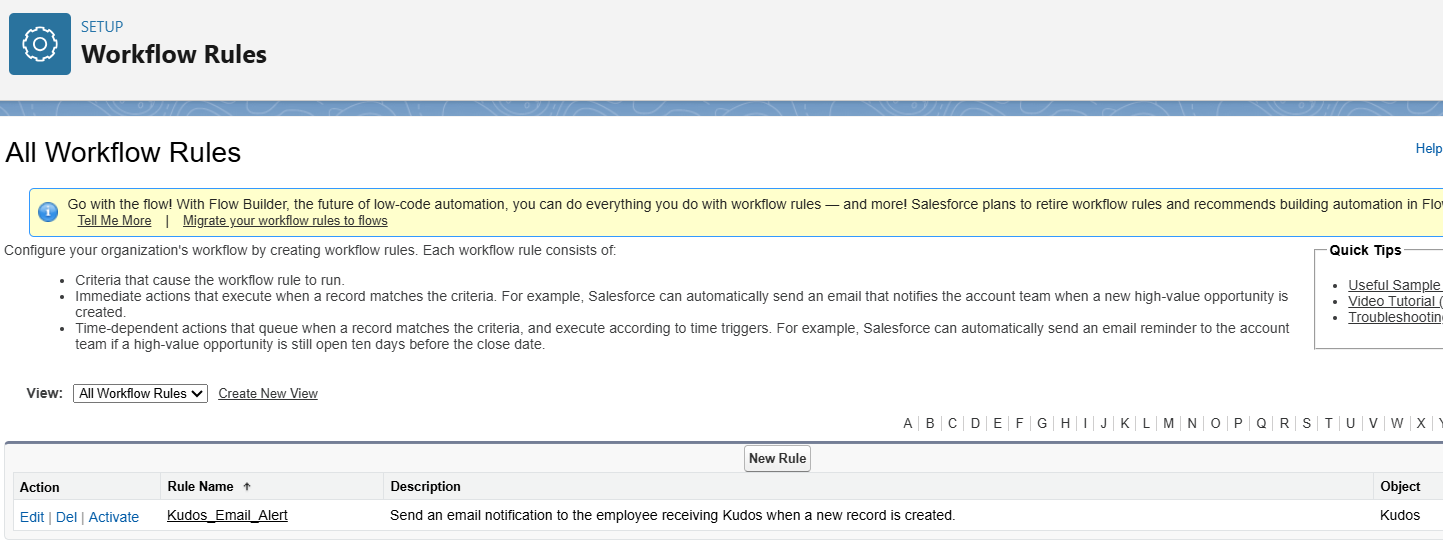
**Testing the Rule**

* Create a new Kudos record.
* Try selecting From Employee = To Employee.
* Save → You should see the error:



### **2. Workflow Rules (Basic Option)**

* Although **Flows** are recommended now, you can use Workflow Rules for simple automation:
  + Send an **Email Alert** to the recipient when Kudos is given.
  + Create a **Task** for managers to review monthly Kudos.



### **3. Process Builder (Legacy Option)**

Can update Kudos Points on the User record automatically.

Steps:

1. Setup → Process Builder → New Process
2. Object = Kudos → Trigger = when a record is created
3. Action: Update related User record → add Points Awarded to Kudos Points



### **4. Flow Builder (Recommended Modern Approach)**

We’ll use **Record-Triggered Flows** for automation:

* **Flow 1: Update Points**
  + Trigger: On insert of a Kudos record.
  + Action: Add Points Awarded value to the recipient’s Kudos Points field on the User record.

**Steps:**

1. Create a Record-Triggered Flow on the Kudos object.
2. Trigger: When a record is created.
3. Get Records: Retrieve the To Employee (User) record.
4. Update Records:
   * Field: Kudos\_Points\_\_c
   * Formula: Current Kudos\_Points + Points\_Awarded\_\_c
5. Activate the flow.

* **Flow 2: Notify Recipient**
  + Trigger: Same as above.
  + Action: Send an **Email Alert + In-App Notification** (“You received Kudos from [User]”).

**Steps:**

1. Create a Record-Triggered Flow on the Kudos object.
2. Trigger: When a record is created.
3. Send Email Alert: “You received Kudos from [User]”.
4. Send Custom Notification: Lightning pop-up to the To Employee.
5. Activate the flow.

* **Flow 3: Scheduled Flow (Optional)**
  + Run monthly to send HR a report *of* Top 5 employees with highest Kudos points*.*

**Steps:**

1. Create a Scheduled-Triggered Flow.
2. Frequency: Monthly
3. Get Records: Query Kudos points or User records.
4. Sort/Filter: Top 5 employees by total points.
5. Action: Send email or report to HR.
6. Activate the flow.

### **5. Approval Process (Optional)**

* For special recognition (e.g., “Employee of the Month”), Kudos records can go through an **approval process** where a manager reviews before finalizing.

### **6. Custom Notifications**

* Create a **Custom Notification** (Lightning) that pops up for recipients in the Salesforce UI when Kudos are given.
* Example: “You received Kudos for Innovation from Priya!”

**Create Notification Type**

* Setup → Custom Notifications → New
* Name: Kudos\_Notification
* Channels: Desktop and Mobile → Save

**Add Action in Flow**

* Open Kudos Record-Triggered Flow
* Click **+ → Action → Send Custom Notification**

**Set Notification Details**

* Notification Type: Kudos\_Notification
* Recipient: {!Kudos.Recipient\_\_c}
* Message: You received Kudos for {!Kudos.Reason\_\_c} from {!Kudos.CreatedBy.Name}!

**Activate Flow**

* Save and Activate

**Test**

* Create a Kudos record → recipient sees Lightning pop-up

### **7. Tasks & Reminders**

* Auto-create a **task** for HR every quarter → “Review Recognition Reports for Rewards.”

# **Phase 5: Apex Programming**

### **1. Apex Classes & Objects**

* Create an **Apex Utility Class** (e.g., KudosHandler.cls) to handle:
  + Updating Kudos Points on the recipient’s User record.
  + Fetching top recognized employees.
  + Sending custom notifications programmatically.

### **2. Apex Triggers**

Triggers let us run code **before or after DML events** (insert, update, delete).

* **Trigger: KudosTrigger**
  + **After Insert:**
    - Add Points Awarded to the recipient’s total Kudos Points on the User object.
    - Send a notification/email.
  + **After Delete (Optional):**
    - Subtract points if Kudos was deleted (e.g., in error).

### **3. Trigger Design Pattern**

* Instead of putting logic in the trigger, we move it into a **Handler Class** for maintainability.
* Benefits: Cleaner code, easier to test, reusable methods.

### **4. SOQL & Collections**

* Use **SOQL** queries to:
  + Get all Kudos for a user.
  + Fetch Top 5 Employees (for dashboard/LWC).

### **5. Batch Apex (Optional, for Scale)**

* If thousands of Kudos records exist, use **Batch Apex** to:
  + Recalculate Kudos Points monthly.
  + Generate summary records for reports.

### **6. Queueable & Scheduled Apex**

* **Queueable Apex:** Send asynchronous notifications for large batches of Kudos.
* **Scheduled Apex:** Run monthly job → Email HR the “Top 10 Recognized Employees” list.

### **7. Exception Handling**

* Wrap SOQL/DML in **try-catch blocks** to handle errors gracefully.
* Log failures (e.g., if updating points fails).

### **8. Test Classes (Mandatory in Salesforce)**

* Write **unit tests** for:
  + Trigger adding/removing Kudos points.
  + Utility class fetching top employees.
* Ensure at least **75% code coverage** before deployment.

# **Phase 6: User Interface Development**

### **1. Lightning App Builder**

* Create a dedicated **App** *→* Employee Recognition App.
* Includes:
  + **Kudos Tab** (custom object).
  + **Reports & Dashboards Tab** (for managers/HR).
  + **Home Page** → Highlight top employees.

### **2. Record Pages**

* **Kudos Record Page:**
  + Display fields → From Employee, To Employee, Kudos Type, Message, Date, Points.
  + Add **Related List** → Kudos given/received by the same employee.
  + Add **Quick Action → “Give Kudos”** directly on User records.
* **User Record Page Enhancement:**
  + Show total “Kudos Points.”
  + Add related list → “Kudos Received.”
  + Display a small leaderboard (top 3 employees).

### **3. Tabs & Navigation**

* Create a **Custom Tab** for Kudos.
* Add it to the navigation bar of the app so employees can easily submit/view Kudos.

### **4. Home Page Layouts**

* Customize the Home Page with components:
  + **Kudos Leaderboard (LWC/Report Chart).**
  + **Recent Kudos Feed (List View or LWC).**
  + **Motivational Quote/Message (Static Resource).**

### **5. Utility Bar (Optional)**

* Add a **Utility Bar Component** → “Give Kudos” quick access button from anywhere in Salesforce.

### **6. Lightning Web Components (LWC)**

To make the app engaging, build custom LWCs:

* **Kudos Wall Component:**
  + Displays the 10 most recent Kudos given.
  + Shows “From Employee → To Employee + Message + Type.”
  + Uses SOQL in Apex to fetch data.
* **Leaderboard Component:**
  + Displays Top 5 Employees with highest Kudos Points.
  + Pulls data from aggregated SOQL query (from Apex).
  + Can include dynamic visuals like stars .

Example LWC: KudosLeaderboard

* JS fetches Apex method → returns top employees.
* HTML displays names + points in a styled leaderboard.

### **7. Apex with LWC**

* Apex methods used in LWCs to fetch Kudos data dynamically.
* Example: @AuraEnabled method → fetch Top 5 employees.

### **8. Navigation Service**

* Enable navigation from LWC → User Record → View all Kudos received.

# **Phase 7: Integration & External Access**

### **1. Use Case for Integration**

* Many employees work in **Slack, Microsoft Teams, or Email** more than Salesforce.
* To increase adoption, we want **Kudos notifications** to appear in these tools.
* HR may also want **Kudos data synced** with an HR system (like Workday or SAP SuccessFactors) for performance reviews.

### **2. Named Credentials & Authentication**

* Set up **Named Credentials** in Salesforce to securely store authentication details for external services (e.g., Slack API, Teams Webhook).
* Ensures no sensitive info is hardcoded in Apex.

### **3. REST API Callouts (External Services)**

* Use **Apex Callouts** to send Kudos notifications:
  + Example: When a Kudos is created, Salesforce sends a message → Slack channel:  
    “ John gave Kudos to Priya for Teamwork: Great support on project delivery!”

Sample Callout Flow:

1. User creates Kudos in Salesforce.
2. Apex Trigger/Flow calls Slack API (via REST endpoint).
3. Message appears in #kudos-channel.

### **4. Platform Events (Real-Time Messaging)**

* Create a **Platform Event: Kudos\_Event\_\_e.**
* Publish an event when a Kudos is given.
* External systems (via CometD or MuleSoft) can subscribe to this event and act in real time (e.g., HR system logs recognition).

### **5. Salesforce Connect / External Objects (Optional)**

* If HR system stores employee data externally, use **Salesforce Connect** to link external employee records with Salesforce Kudos without duplicating data.

### **6. Change Data Capture (CDC)**

* Enable CDC on Kudos object to push recognition data to downstream systems (like a rewards platform or BI tool).
* Example: A BI dashboard in Tableau automatically updates with Kudos points.

### **7. API Limits Consideration**

* Sending notifications to external systems should respect Salesforce API governor limits.
* Use **Batching/Queueable Apex** if sending multiple Kudos at once.

### **8. Security & OAuth**

* Use **OAuth 2.0** for secure integrations with Slack/Teams.
* Ensure only authorized apps can access Kudos data.

# **Phase 8: Data Management & Deployment**

### **1. Data Import Wizard (Small Data Loads)**

* Used for **initial setup** of employees and test Kudos records.
* Example:
  + Import employee list into **User Object** (if sandbox users not created).
  + Import initial Kudos records (sample data for demo).

Limitation → Best for up to ~50k records.

### **2. Data Loader (Large Data Loads)**

* For larger datasets (e.g., thousands of employees or Kudos history migration).
* Use **CSV files** to insert, update, or delete records.
* Example: If company already had Kudos in spreadsheets, import them into the custom **Kudos Object.**

### **3. Duplicate Rules**

* Prevents duplicate Kudos records.
* Example Rule:
  + Same sender giving the same Kudos to the same receiver on the same day → block record creation.

### **4. Data Export & Backup**

* Use **Data Export Service** or **Weekly Export** to back up Kudos records and User recognition points.
* Ensures HR always has historical recognition data.

### **5. Deployment Tools**

* **Change Sets** (Admin-Friendly):
  + Move custom object (Kudos), fields, flows, reports, and dashboards from Sandbox → Production.
* **Unmanaged Packages:**
  + Share the app with other orgs (good for portfolio/demo).
* **VS Code with SFDX / ANT Migration Tool:**
  + Developer-friendly deployment for Apex classes, LWCs, and complex metadata.

### **6. Testing Before Deployment**

* **UAT (User Acceptance Testing):**
  + Employees test “Give Kudos.”
  + HR tests dashboards and reports.
  + Admin checks security/sharing settings.
* **Test Data:** Import 50–100 Kudos records to validate flows, triggers, and dashboards.

### **7. Rollout Plan**

* **Pilot Phase:** Release app to one department (e.g., Sales team).
* **Org-Wide Rollout:** After feedback, deploy app to all departments.
* **Training:** Short session to teach employees how to give Kudos & view dashboards.

### **8. Post-Deployment Data Strategy**

* Set up **Quarterly Data Archive** → Export Kudos data for compliance.
* Automate **Data Clean-Up Flow** → E.g., mark inactive users but retain their Kudos history.

# **Phase 9: Reporting, Dashboards & Security Review**

## **1. Reports (Analytics)**

Reports turn raw Kudos data into insights.

* **Tabular Report (Simple List):**
  + List of Kudos given/received by each employee.
  + Useful for HR to filter recognition by department or date.
* **Summary Report (Aggregated):**
  + Group by To Employee → Total Kudos points received.
  + Shows who has the most recognition.
* **Matrix Report (Comparison):**
  + Rows: Departments
  + Columns: Kudos Type (Teamwork, Innovation, Leadership, etc.)
  + Insight: Which department leads in collaboration or innovation.
* **Joined Report (Advanced):**
  + Combine “Kudos Given” vs. “Kudos Received.”
  + Identify employees who both give and receive recognition (balanced engagement).

## **2. Dashboards (Visual Insights)**

Dashboards make reports engaging and easy for HR/Managers.

* **Leaderboard Component (Bar Chart):** Top 5 employees by Kudos Points.
* **Kudos by Type (Pie Chart):** Distribution of recognition categories.
* **Kudos Trend Over Time (Line Chart):** Engagement growth weekly/monthly.
* **Department-wise Recognition (Stacked Bar):** Compare Kudos activity across teams.
* **Manager Dashboard:** Recognition given within their reporting hierarchy.

Dashboards can be **dynamic** → Managers only see their team’s data, HR sees all.

## **3. Security Review (Data & Access)**

Security ensures employees see only what they’re supposed to.

* **Sharing Settings (OWD):**
  + Kudos Object → Private (only creator & recipient can see).
  + HR Managers → Access via Sharing Rules.
* **Field-Level Security (FLS):**
  + Employees can’t edit “Kudos Points.”
  + Only HR/Admin can adjust points manually (if needed).
* **Permission Sets:**
  + “Kudos User” → Can create Kudos.
  + “Kudos Manager” → Access to dashboards and full reporting.
* **Session Settings:**
  + Set session timeout policies to prevent unauthorized access.
* **Audit Trail:**
  + Track changes to Kudos records → Who gave, who deleted, etc.
* **Login IP Ranges:**
  + Restrict Admin/HR logins to office IPs for compliance.

## **4. Review for AppExchange-Style Packaging (Optional)**

* If you want to publish or reuse → conduct a **Security Review** for best practices:
  + Ensure Apex classes use **with sharing.**
  + Validate inputs to prevent SOQL injection.
  + Bulkify triggers & Flows for performance.

# **Phase 10: Final Presentation & Demo Day**

### **1. Pitch Presentation**

* **Objective:** Convince stakeholders (HR, Managers, Leadership) of the app’s value.
* **Key Talking Points:**
  + Problem → Recognition is informal, untracked, and invisible to HR.
  + Solution → Salesforce-based Kudos app with automation, points, and dashboards.
  + Benefits → Boosts morale, improves engagement, provides HR insights.

Use **slides or a live demo** to explain phases 1–9 in simple terms.

### **2. Demo Walkthrough**

Do a **live demo** of the app inside Salesforce:

1. **Give Kudos:** Show how an employee selects a colleague, adds message/type, and submits.
2. **Automation in Action:**
   * Recipient instantly receives an email/notification.
   * Kudos Points auto-updated on User record.
3. **Leaderboard Dashboard:** Display Top 5 recognized employees.
4. **Kudos Wall (LWC or List View):** Show recent recognition messages.
5. **Manager View:** Show department-wise reports.

This makes the app feel **real and useful.**

### **3. Feedback Collection**

* Ask employees: “Was the Kudos submission easy to use?”
* Ask HR: “Does the dashboard give you the visibility you need?”
* Collect suggestions for new features → e.g., badges, certificates, rewards.

### **4. Handoff Documentation**

Prepare a simple guide for end-users & admins:

* **User Guide:** How to give Kudos, view dashboards, and track points.
* **Admin Guide:** Managing Kudos object, updating flows, and adjusting points if needed.
* **Developer Notes:** Apex classes, triggers, and LWC components used.

### **5. LinkedIn/Portfolio Showcase**

* Write a **case study** describing the project phases.
* Add **screenshots of the Kudos app & dashboards.**
* Share on LinkedIn or GitHub as a **portfolio project** → shows real Salesforce implementation skills.