

Phase 1 - Problem Understanding & Industry Analysis

Requirement Gathering

The Helping Hands Foundation project requires a Salesforce-based solution to streamline the management of volunteers, donations, and events. The primary needs identified include:

- Tracking volunteer details, skills, and availability.
- Recording and monitoring donations with amount, date, and donor details.
- Organizing events and associating them with volunteers.
- Maintaining transparency and generating reports for stakeholders.
- Ensuring secure access based on roles.

Stakeholder Analysis

Key stakeholders in the Helping Hands Foundation include:

- CEO: Oversees the overall operations and ensures alignment with the organization's mission.
- CFO: Manages financial aspects, donation tracking, and compliance.
- Financial Officer: Records transactions, monitors donation flow, and ensures proper allocation.
- Donation Manager: Handles donor engagement, donation collection, and donor relations.
- Volunteer Coordinator: Manages volunteer recruitment, assignments, and scheduling.
- Donors: External stakeholders providing financial support.
- Volunteers: Community members who actively participate in events and initiatives.

Business Process Mapping

The current manual processes include:

1. Volunteers register via paper forms or spreadsheets.
2. Donations are tracked manually in ledgers or Excel files.
3. Event planning and volunteer assignments are managed over phone calls or emails.

Proposed Salesforce Solution:

- Volunteers can be registered in Salesforce with their availability and skills.
- Donations are logged in Salesforce and linked to donors and volunteers.
- Events are created in Salesforce with capacity, location, and date, and linked to volunteers via a junction object.
- Automation rules ensure approvals, notifications, and reminders are triggered.
- Reports and dashboards provide real-time visibility into donations and volunteer activities.

Industry-specific Use Case Analysis

Non-profit organizations, like Helping Hands Foundation, face challenges in managing large amounts of donor and volunteer data. Common industry challenges include donor retention, transparent reporting, and efficient resource allocation. Salesforce Nonprofit Success Pack (NPSP) is a popular industry solution, offering capabilities like donor management, household tracking, and grant management.

For Helping Hands Foundation:

- Donor management ensures donor history and preferences are maintained.
- Volunteer management ensures events have adequate staffing.
- Donation transparency builds trust among stakeholders and encourages repeated contributions.

AppExchange Exploration

AppExchange, Salesforce's marketplace, provides pre-built solutions for nonprofits. Relevant apps include:

- Nonprofit Success Pack (NPSP): Tailored for donor and volunteer management.
- Classy for Salesforce: Helps with fundraising campaigns.
- Volunteers for Salesforce: Streamlines volunteer scheduling and hours tracking.
- FormAssembly: Simplifies data collection via online forms.

Exploring these solutions can accelerate implementation by leveraging best practices and reducing custom development efforts.

Phase 2 - Org Setup & Configuration

This phase covers the complete setup and configuration of the proper environment setup, security, data modeling readiness, and prepares the organization for future development, automation, and deployment.

Step 1: Salesforce Editions

We are using Salesforce Developer Edition as our working environment. This edition provides the necessary licenses, features, and tools to build and test our Helping Hands Foundation project without production costs.

Step 2: Company Profile Setup

Configured Company Profile with organization details such as Company Name (Helping Hands Foundation), Locale, Default Language, Time Zone, and Currency to ensure consistency across the org.

The screenshot shows the 'Company Information' page in the Salesforce Setup interface. The page title is 'Company Information' under the 'SETUP' tab. The organization's name is listed as 'Helping Hands Foundation'. A note says 'The organization's profile is below.' Below the profile section, there are tabs for 'User Licenses [10+] | Permission Set Licenses [10+] | Feature Licenses [11+] | Usage-based Entitlements [10+]' and a 'Help for this Page' button.

Organization Detail

Organization Name	Helping Hands Foundation	Phone
Primary Contact	OrgFarm EPIC	Fax
Division		Default Locale
Address	Telangana India	English (India)
Fiscal Year Starts In	January	Default Language
Activate Multiple Currencies	<input type="checkbox"/>	English
Enable Data Translation	<input type="checkbox"/>	Default Time Zone
Newsletter	<input checked="" type="checkbox"/>	(GMT+05:30) India Standard Time (Asia/Kolkata)
Admin Newsletter	<input checked="" type="checkbox"/>	Currency Locale
Hide Notices About System Maintenance	<input type="checkbox"/>	Telugu (India) - INR
Hide Notices About System Downtime	<input type="checkbox"/>	Used Data Space
Locale Formats	ICU	402 KB (8%) [View]
		Used File Space
		31 KB (0%) [View]
		API Requests, Last 24 Hours
		0 (15,000 max)
		Streaming API Events, Last 24 Hours
		0 (10,000 max)
		Restricted Logins, Current Month
		0 (0 max)
		Salesforce.com Organization ID
		00DgK000007YDV7
		Organization Edition
		Developer Edition
		Instance
		CAN96

Step 3: Business Hours & Holidays

Defined standard business hours (Mon-Fri, 9 AM – 6 PM) and added holidays (Independence Day, Republic Day, New Year, etc.) to support service-level agreements and volunteer scheduling.

The screenshot shows the 'Business Hours' setup page under 'SETUP'. It displays the 'Organization Business Hours' section with a table of standard business hours from Monday to Saturday. A 'Holidays' section is also present, showing a single entry for 'Republic Day' on January 26, 2026, marked as an 'All Day' event.

Step 4: Fiscal Year Settings

Enabled a Standard Fiscal Year setup aligned with the calendar year (Jan – Dec). This ensures proper reporting of donations, expenses, and volunteer funding cycles.

The screenshot shows the 'Fiscal Year' setup page under 'SETUP'. It displays the 'Organization Fiscal Year Edit' section for 'Helping Hands Foundation'. The 'Standard Fiscal Year' option is selected, and the 'Change Fiscal Year Period' dialog is open, showing the start month as 'January' and the ending month as 'The ending month'.

Step 5: User Setup & Licenses

Created project-specific users and assigned appropriate licenses and profiles:

- CEO: System Administrator License
- CFO: Salesforce Platform License
- Financial Officer: System Administrator License
- Donation Manager: Salesforce Platform License
- Volunteer Coordinator: Salesforce Platform License

The screenshot shows the Salesforce 'Users' page under the 'SETUP' tab. The page title is 'All Users'. It displays a table of users with columns for Action, Full Name, Alias, Username, Role, Active, and Profile. The users listed are:

Action	Full Name	Alias	Username	Role	Active	Profile
<input type="checkbox"/> Edit	Chatter Expert	Chatter	chatty_00dgk000007ydv7uao.yy02n1syak@chatter.salesforce.com		<input checked="" type="checkbox"/>	Chatter Free User
<input type="checkbox"/> Edit Login	Coordinator_Volunteer	vcoord	vc.volunteer@ngo.org	Volunteer Coordinator	<input checked="" type="checkbox"/>	Volunteer Coordinator Profile
<input type="checkbox"/> Edit	Kovi_Venkata_Ukith_Sai	sai	saiukith511187@agentforce.com		<input checked="" type="checkbox"/>	System Administrator
<input type="checkbox"/> Edit Login	Manager_Donation	dmanag	dm.manager@ngo.org	Donation Manager	<input checked="" type="checkbox"/>	Donation Manager Profile
<input type="checkbox"/> Edit Login	Officer_Finance	foffice	fo.officer@ngo.org	Financial Officer	<input checked="" type="checkbox"/>	System Administrator
<input type="checkbox"/> Edit	User_Integration	integ	integration@00dgk000007ydv7uao.com		<input checked="" type="checkbox"/>	Analytics Cloud Integration User
<input type="checkbox"/> Edit	User_Security	sec	insightssecurity@00dgk000007ydv7uao.com		<input checked="" type="checkbox"/>	Analytics Cloud Security User

Step 6: Profiles

Profiles define baseline access. Created custom profiles for Donation Manager and Volunteer Coordinator with object-level and field-level access tailored to their roles.

The screenshot shows the Salesforce 'Profiles' page under the 'SETUP' tab. It contains two tables for different profile types:

Top Table (Volunteer Coordinator Profile):

Action	Profile Name	User License	Custom
<input type="checkbox"/> Edit Del ...	Volunteer Coordinator Profile	Salesforce Platform	<input checked="" type="checkbox"/>

Bottom Table (Donation Manager Profile):

Action	Profile Name	User License	Custom
<input type="checkbox"/> Edit Del ...	Donation Manager Profile	Salesforce Platform	<input checked="" type="checkbox"/>

Step 7: Roles

Set up the role hierarchy to mirror the organization:

- CEO
 - ↳ CFO
 - ↳ Financial Officer
 - ↳ Donation Manager
 - ↳ Volunteer Coordinator

The screenshot shows the 'Roles' page in the Salesforce Setup. The title is 'Creating the Role Hierarchy'. It displays a tree view of roles under 'Helping Hands Foundation':

- CEO
 - ↳ CFO
 - ↳ Financial Officer
 - ↳ Donation Manager
 - ↳ Volunteer Coordinator

Buttons for 'Add Role' are visible next to each role node.

Step 8: Permission Sets

Create additional permission sets for users requiring extra access beyond their profiles, e.g., 'Edit Access and Read/Write/Edit Access' for generating advanced reports.

The screenshot shows the 'Permission Sets' page in the Salesforce Setup. The title is 'Permission Set Overview > Object Settings > Donations'. The 'Edit' button is highlighted.
The 'Tab Settings' section shows the 'Visible' tab selected for the 'Available' tab.
The 'Object Permissions' section lists permissions for the 'Donations' object:

Permission Name	Enabled
Read	<input checked="" type="checkbox"/>
Create	<input checked="" type="checkbox"/>
Edit	<input checked="" type="checkbox"/>
Delete	<input type="checkbox"/>
View All Records	<input type="checkbox"/>
Modify All Records	<input type="checkbox"/>
View All Fields	<input type="checkbox"/>

The 'Field Permissions' section lists permissions for fields:

Field Name	Field API Name	Read Access	Edit Access
Created By	CreatedById	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Donation Number	Name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Last Modified By	LastModifiedById	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Owner	OwnerId	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Step 9: Organization-Wide Defaults (OWD)

Configured OWD:

- Donations: Private (only owner & above roles can see)
- Volunteers: Private (restricted by default)
- Accounts/Contacts: Controlled by Parent
- Other standard objects: Default Salesforce settings

Sharing Settings Help for this Page 

This page displays your organization's sharing settings. These settings specify the level of access your users have to each others' data. Go to [Background Jobs](#) to monitor the progress of a change to an organization-wide default or a parallel sharing recalculation.

Manage sharing settings for: [All Objects](#)

[Disable External Sharing Model](#)

Default Sharing Settings			
Organization-Wide Defaults			
Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Lead	Public Read/Write/Transfer	Private	✓
Account and Contract	Public Read/Write	Private	✓
Contact	Controlled by Parent	Controlled by Parent	✓

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Donation	Private	Private	✓
Event	Public Read/Write	Private	✓
Mentor	Public Read/Write	Private	✓
Student	Public Read/Write	Public Read/Write	✓
Volunteer	Private	Private	✓

Step 10: Sharing Rules

Created Sharing Rules to support collaboration:

- Donation Sharing Rule: Records owned by Volunteer Coordinators are shared with Donation Managers (Read-Only).
- Volunteer Sharing Rule: Volunteer records owned by Volunteer Coordinators are shared with all Volunteer Coordinators (Read/Write).

SETUP Sharing Settings

No sharing rules specified.

Work Type Group Sharing Rules			
New	Recalculate	Work Type Group Sharing Rules Help 	
No sharing rules specified.			

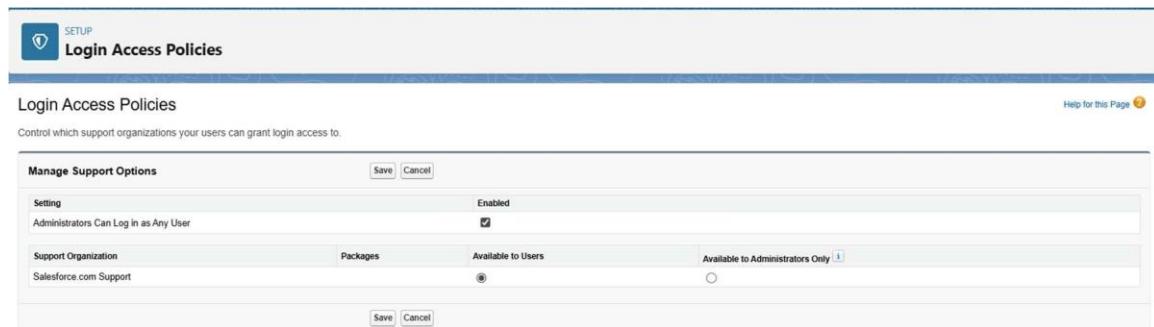
Donation Sharing Rules			
New	Recalculate	Donation Sharing Rules Help 	
Action	Criteria	Shared With	Access Level
Edit Del	Owner in Role and Internal Subordinates: Volunteer Coordinator	Role: Donation Manager	Read Only

Event Sharing Rules			
New	Recalculate	Event Sharing Rules Help 	
No sharing rules specified.			

Volunteer Sharing Rules			
Action	Criteria	Shared With	Access Level
Edit Del	Owner in Role and Internal Subordinates: Volunteer Coordinator	Role and Internal Subordinates: Volunteer Coordinator	ReadWrite

Step 11: Login Access Policies

Enabled administrators to log in as any user for troubleshooting. This allows quick issue resolution and ensures smooth testing during the project.

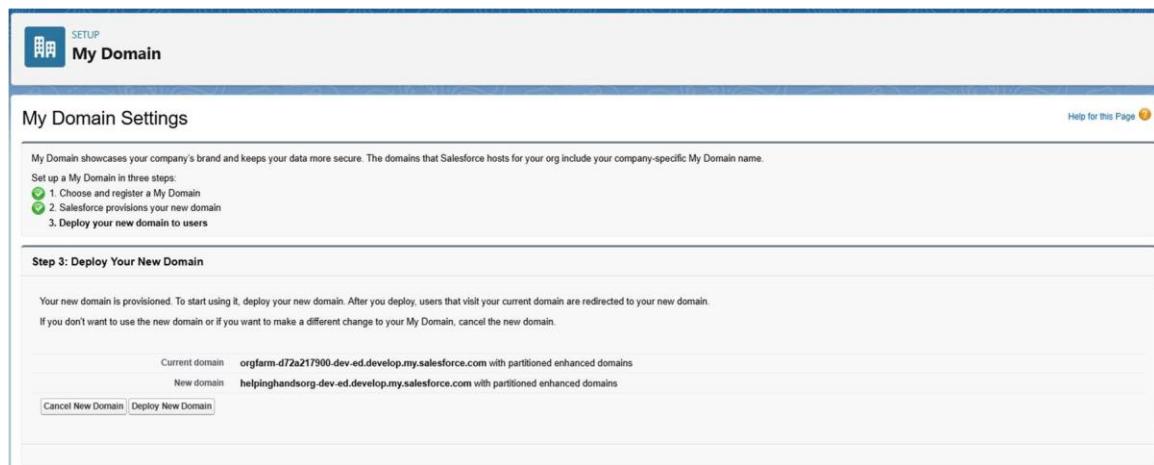


Setting	Packages	Available to Users	Available to Administrators Only
Administrators Can Log in as Any User		<input checked="" type="checkbox"/>	
Support Organization		<input type="radio"/>	<input type="radio"/>

Step 12: Development Org Setup

Confirmed the Developer Org setup and enabled

My Domain :(**helpinghandsorg-dev-ed.develop.my.salesforce.com**).



My Domain Settings

My Domain showcases your company's brand and keeps your data more secure. The domains that Salesforce hosts for your org include your company-specific My Domain name.

Set up a My Domain in three steps:

1. Choose and register a My Domain
2. Salesforce provisions your new domain
3. Deploy your new domain to users

Step 3: Deploy Your New Domain

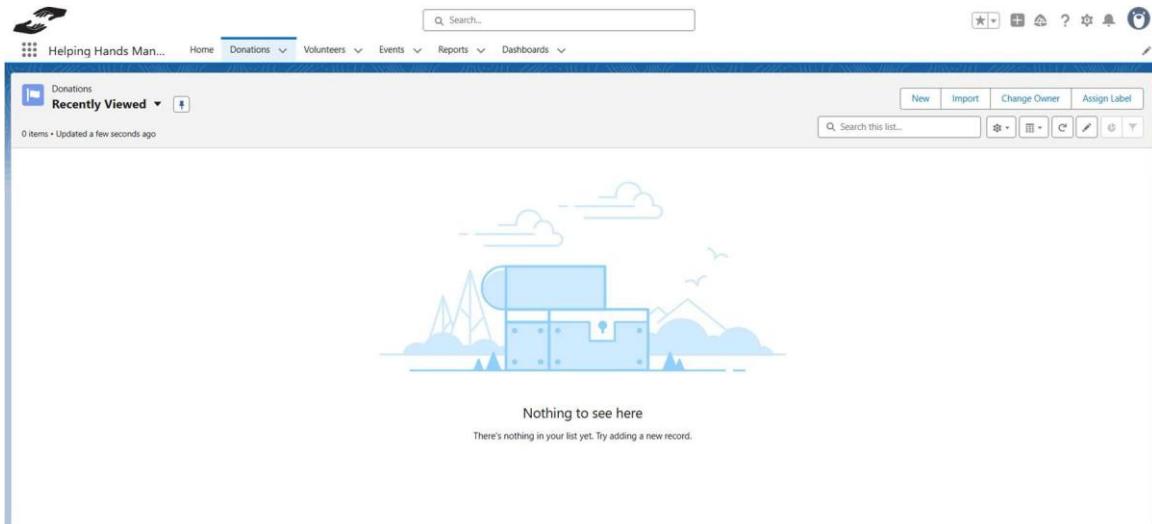
Your new domain is provisioned. To start using it, deploy your new domain. After you deploy, users that visit your current domain are redirected to your new domain.

If you don't want to use the new domain or if you want to make a different change to your My Domain, cancel the new domain.

Current domain	orgfarm.d72a217900-dev-ed.develop.my.salesforce.com with partitioned enhanced domains
New domain	helpinghandsorg-dev-ed.develop.my.salesforce.com with partitioned enhanced domains

[Cancel New Domain](#) [Deploy New Domain](#)

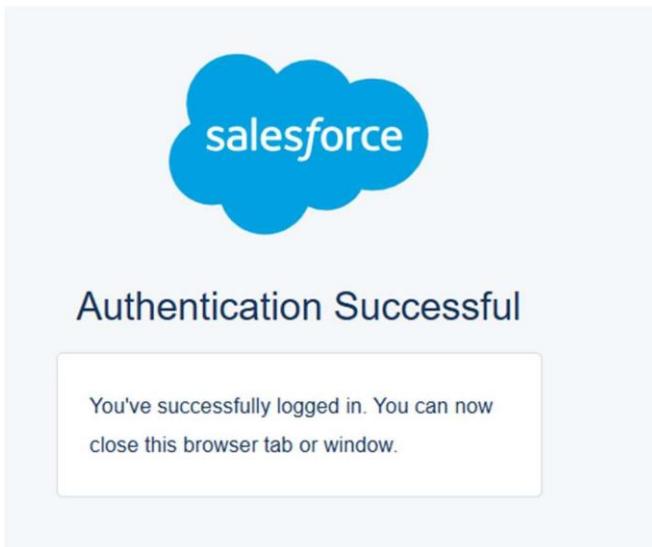
Created a dedicated app 'Helping Hands Management' for testing all project configurations and customizations.



Step 13: Deployment Basics

Deployment options:

- Change Sets: For Admin-driven migration (not available in Developer Edition).
- Salesforce DX (SFDX): Setup VS Code with Salesforce CLI, connect Dev Org, retrieve and deploy metadata to another org.



```
PS C:\Users\HP\OneDrive\Documents\GitHub\salesforce-capstone-volunteer-donation> sfdx force:auth:web:logon -a DevOrg
>>> Successfully authorized sailikith511187@agentforce.com with org ID 00DgK000007YDV7UA0
PS C:\Users\HP\OneDrive\Documents\GitHub\salesforce-capstone-volunteer-donation>
```

```
File Edit Selection View ... ← → ⌘ salesforce-capstone-volunteer-donation ⌘ ... 08 □ □ □ - ○ x
EXPLORER ... Welcome x
SALEFORCE-CAPSTONE... D+ U ⌂
HelpingHandsProject ...
> .husky
> .vscode
> config
> force-app
> scripts
  ◆ .forceignore
  ◆ .gitignore
  ◆ .prettierignore
  { .prettierrc
  @ eslint.config.js
  JS jest.config.js
  { package.json
  README.md
  { sfdsx-project.json
  Phase-1.md

  > OUTLINE
  > TIMELINE
  > RUNNING TASKS
Start Walkthroughs
New File...
Open File...
Open Folder...
Clone Git Repository...
More...
Connect to
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS pwsh Δ + ⌂ ⌂ ... | ⌂ x
PS C:\GitHub\salesforce-capstone-volunteer-donation> sf project generate --name HelpingHandsProject
--template standard
>>>
create HelpingHandsProject\eslint.config.js
create HelpingHandsProject\scripts\sql\account.sql
create HelpingHandsProject\scripts\apex\hello.apex
create HelpingHandsProject\.forceignore
create HelpingHandsProject\.gitignore
create HelpingHandsProject\.prettierignore
create HelpingHandsProject\jest.config.js
create HelpingHandsProject\package.json
PS C:\GitHub\salesforce-capstone-volunteer-donation>
```

For this project, deployment will be demonstrated using SFDX and a secondary Developer Org as Production simulation.

Phase 3 - Data Modeling & Relationships

1. Purpose & Objectives

Purpose: Develop the data model necessary to manage Volunteers, Events, Donations, Volunteer assignments, and tasks, enabling the Helping Hands Foundation to track volunteer participation, donations, and event activities.

Objectives:

- Create and configure custom objects and fields
- Establish relationships (lookup & master-detail, junction object)
- Configure Page Layouts, Compact Layouts, and Related Lists
- Verify model in Schema Builder
- Create sample data and test basic SOQL queries
- Pull metadata into SFDX and commit to Git

2. General Conventions & Notes

- All custom objects use the suffix `_c` in the API name (e.g., `Volunteer_c`).
- Use Lookup relationships for flexibility and use Master-Detail when you need roll-up summaries and ownership inheritance.
- System fields like `CreatedById`, `LastModifiedById`, `OwnerId` are created automatically — you don't need to add them.

3. Full Object & Field Definitions

3.1 Volunteer (`Volunteer_c`)

Object settings

- Label: Volunteer
- Plural Label: Volunteers
- Record Name: Volunteer ID (Auto Number)
- Display Format: VOL-{0000}

Fields to create (custom object-volunteer):

Label	API Name	Data Type	Notes
Volunteer Name	Volunteer_Name_c	Text (255)	Full person name
Email	Email_c	Email	--
Phone	Phone_c	Phone	--
Skills	Skills_c	Multi-Select Picklist	e.g., Event Management; Fundraising; Teaching; Medical
Availability	Availability_c	Picklist	Values: Weekdays, Weekends, Evenings, Flexible
Joined Date	Joined_Date_c	Date	--
Total Hours	Total_Hours_c	Number (5,1)	Sum of hours contributed (see roll-up notes)
Status	Status_c	Picklist	Active, Inactive, Onboarding
Linked Account (optional)	Account_c	Lookup(Account)	If volunteers are linked to organizations

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Availability	Availability_c	Picklist		
Created By	CreatedById	Lookup(User)		
Email	Email_c	Email		
Joined Date	Joined_Date_c	Date		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User/Group)		✓
Phone Number	Phone_Number_c	Phone		
Skills	Skills_c	Picklist (Multi-Select)		
Status	Status_c	Picklist		
Total Hours	Total_Hours_c	Number(3, 1)		
Volunteer ID	Name	Auto Number		✓
Volunteer Name	Volunteer_Name_c	Text(100)		

3.2 Event (Event_c)

Object settings

- Label: Event
- Plural Label: Events
- Record Name: Event ID (Auto Number)

- Display Format: EVT-{0000}

Fields to create (custom object - Event):

Label	API Name	Data Type	Notes
Event Name	Event_Name__c	Text (255)	Human-readable name
Event Date	Event_Date__c	Date	--
Location	Location__c	Text (255)	--
Description	Description__c	Long Text Area	--
Event Type	Event_Type__c	Picklist	Fundraising, Awareness, Education, Other
Organizer (Volunteer)	Organizer__c	Lookup(Volunteer__c)	Who organizes the event

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Description	Description__c	Long Text Area(32768)		
Event Date	Event_Date__c	Date		
Event ID	Name	Auto Number		✓
Event Name	Event_Name__c	Text(255)		
Last Modified By	LastModifiedById	Lookup(User)		
Location	Location__c	Text(255)		
Owner	OwnerId	Lookup(User,Group)		✓

3.3 Donation (Donation__c)

Object settings

- Label: Donation

- Plural Label: Donations
- Record Name: Donation ID (Auto Number) — DON-{0000}

Fields present in my org:

- **Donation ID** — Name (Auto Number).
- **Amount** — Amount_c — Currency.
- **Donation Date** — Donation_Date_c — Date.
- **Donor Name** — Donor_Name_c — Text(100).
- **Donor Email** — Donor_Email_c — Email.
- **Linked Volunteer** — Linked_Volunteer_c — Lookup(Volunteer_c).
- **Payment Method** — Payment_Method_c — Picklist. (Values you used e.g., Cash, Bank Transfer, Credit Card, UPI, Cheque, Other)
- **Status** — Status_c — Picklist. (e.g., Pledged, Received, Refunded)
- **Notes** — Notes_c — Long Text Area.
- **Owner** — standard OwnerId (User/Queue).
- **CreatedBy, LastModifiedBy** — system fields.

Recommended additional field(s):

- **Receipt Sent** — Receipt_Sent_c — Checkbox (default false).
- **Related Event** — Related_Event_c — Lookup(Event_c)

Validation Rule (recommended):

- **Name:** Amount_Must_Be_Positive
- **Formula:** Amount_c <= 0
- **Error message:** Donation Amount must be greater than 0.
- **Error location:** Amount_c field

Fields & Relationships					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
	Amount	Amount__c	Currency(18, 0)		✓
	Created By	CreatedById	Lookup(User)		✓
	Donation Date	Donation_Date__c	Date		✓
	Donation ID	Name	Auto Number		✓
	Donor Email	Donor_Email__c	Email		✓
	Donor Name	Donor_Name__c	Text(100)		✓
	Last Modified By	LastModifiedById	Lookup(User)		✓
	Linked Volunteer	Linked_Volunteer__c	Lookup(Volunteer)		✓
	Notes	Notes__c	Long Text Area(22760)		✓
	Owner	OwnerId	Lookup(User/Group)		✓
	Payment Method	Payment_Method__c	Picklist		✓
	Status	Status__c	Picklist		✓

3.4 Volunteer Event (Junction) (VolunteerEvent_c)

Object settings

- Label: Volunteer Event
- Plural Label: Volunteer Events
- Record Name: Volunteer Event ID (Auto Number) — VE-{0000}

Relationships (required)

- Master-Detail → Volunteer (Volunteer_c) — Volunteer_c (Master-Detail)
- Master-Detail → Event (Event_c) — Event_c (Master-Detail)

Optional fields on the junction:

- Role — Role_c — Picklist (Helper, Organizer, Coordinator)
- Hours Contributed — Hours_Contributed_c — Number (5,1)
- Feedback — Feedback_c — Long Text Area

Fields & Relationships					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
	Created By	CreatedById	Lookup(User)		✓
	Event	Event__c	Master-Detail(Event)		✓
	Last Modified By	LastModifiedById	Lookup(User)		✓
	Volunteer	Volunteer__c	Master-Detail(Volunteer)		✓
	Volunteer Event Name	Name	Auto Number		✓

3.5 Volunteer Task (Volunteer_Task_c) — custom task object

Object settings

- Label: Volunteer Task
- Plural Label: Volunteer Tasks
- Record Name: Task ID (Auto Number) or Task Name (Text) – choose Task Name (Text) if you want descriptive titles.

Fields:

- Task Name — Name — Text (if used as Name)
- Assigned Volunteer — Assigned_Volunteer_c — Lookup(Volunteer_c)
- Related Event — Related_Event_c — Lookup(Event_c)
- Status — Status_c — Picklist (Not Started, In Progress, Completed)
- Priority — Priority_c — Picklist (High, Medium, Low)
- Due Date — Due_Date_c — Date
- Notes — Notes_c — Long Text Area

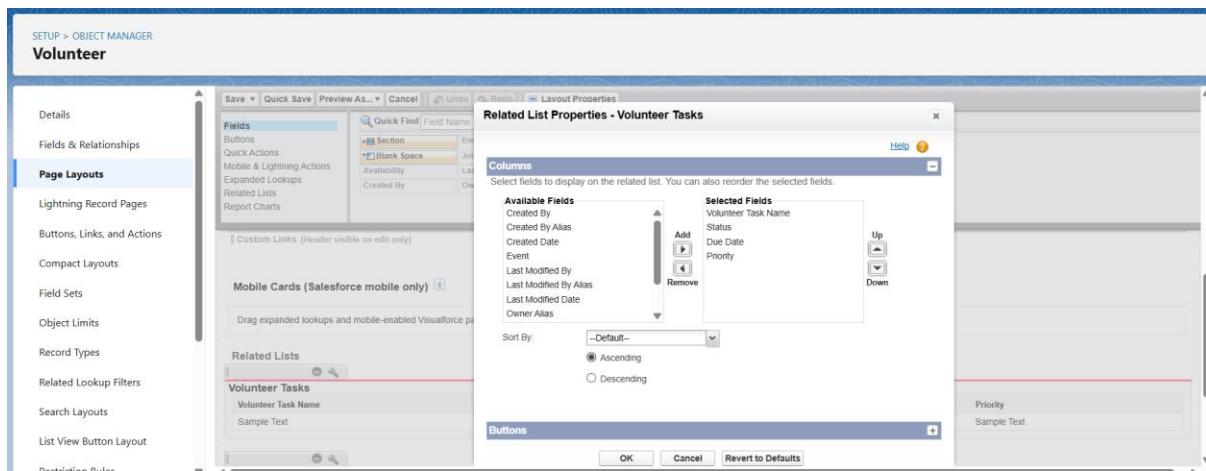
The screenshot shows the Salesforce Object Manager interface for the 'Volunteer Task' object. The left sidebar has a 'Fields & Relationships' section selected. The main area displays a table titled 'Fields & Relationships' with 9 items, sorted by Field Label. The table includes columns for FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The fields listed are: Created By (CreatedBy), Due Date (Due_Date_c), Event (Event_c), Last Modified By (LastModifiedBy), Owner (OwnerId), Priority (Priority_c), Status (Status_c), Volunteer (Volunteer_c), and Volunteer Task Name (Name). The 'CONTROLLING FIELD' column shows that most fields are controlled by the Volunteer_c field, except for 'Created By' which is controlled by User. The 'INDEXED' column indicates that several fields are indexed (Priority, Status, Volunteer, and Volunteer Task Name).

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
Due Date	Due_Date_c	Date		
Event	Event_c	Lookup(Event)		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		
Priority	Priority_c	Picklist		
Status	Status_c	Picklist		
Volunteer	Volunteer_c	Lookup(Volunteer)		
Volunteer Task Name	Name	Text(80)		

4. Page Layouts

Configure Page Layouts & Related Lists

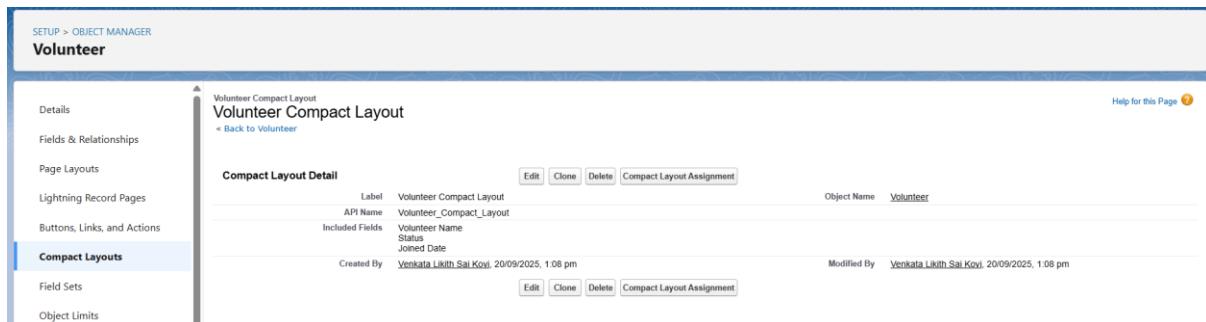
1. Object Manager → Volunteer → **Page Layouts** → Edit default layout.
2. Drag key fields at the top (Volunteer Name, Email, Phone, Status, Skills), and add related lists: Volunteer Events, Volunteer Tasks, Donations (if Donations link to Volunteer).
3. Save.
4. Repeat for Event and Donation (add related lists and important fields).



5. Compact Layouts

Configure Compact Layouts (Mobile / Highlights Panel)

- Object Manager → Volunteer → **Compact Layouts** → New → choose key fields (Volunteer Name, Status, Joined Date).
- Save → Click **Compact Layout Assignment** → Set as Primary.



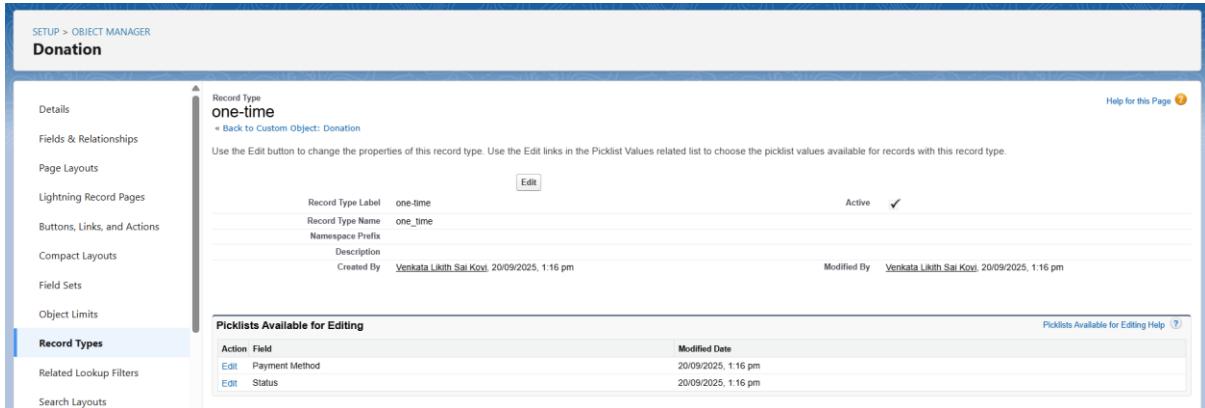
6. Field Level Security & Profile Access

- For each field: Object Manager → {Object} → Fields & Relationships → click field → **Set Field-Level Security** → Select profiles that should have visibility/edit access → Save.
- For each object: Object Manager → {Object} → **Object Settings** in profile to control CRUD (Create, Read, Edit, Delete). Or edit the cloned profiles created in Phase 2 and set Object Permissions.

7. Record Types & Page Layout Assignments

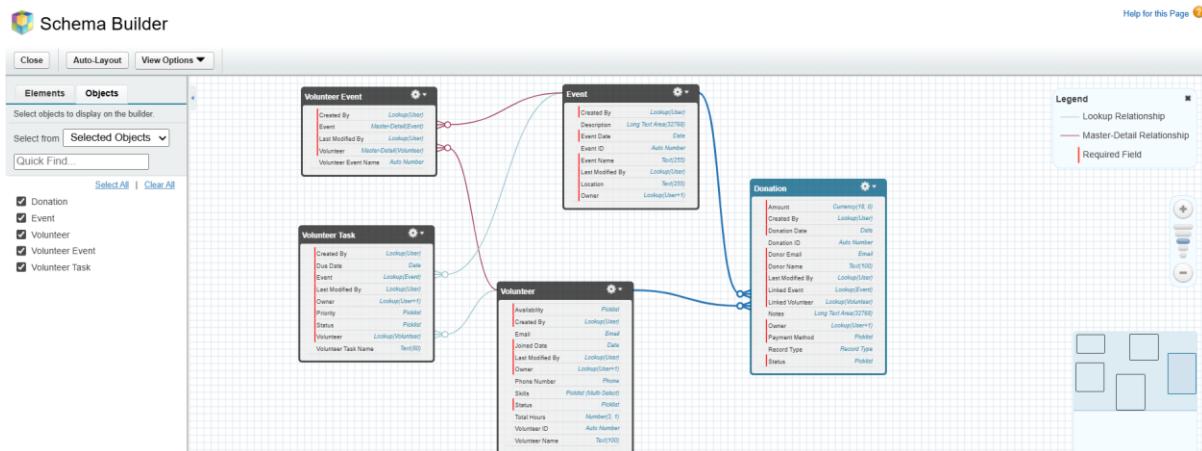
Object Manager → Donation → **Record Types** → New: One-Time, Recurring.

1. Assign different page layouts if you want different fields visible for types.
2. Save and assign to profiles.



8. Schema Builder

1. Setup → **Schema Builder** → Check the left panel boxes for Volunteer, Event, Donation, Volunteer Event, Volunteer Task.
2. Layout the schema on the canvas and verify relationships (look for lines representing Lookup and Master-Detail).
3. Take a screenshot of the schema diagram for documentation.



9. Sample Data (Test Records)

Create sample records to validate relationships and reporting:

Volunteers:

 Helping Hands Man...

Home Donations Volunteers Events Reports Dashboards

Volunteer A-001

Volunteer ID	A-001	Owner	Venkata Likith Sai Kovi
Volunteer Name	Rahul		
Email	rahul73@gmail.com		
Phone Number	7479546738		
Skills	Event Management;Teaching		
Availability	Evenings		
Joined Date	08/09/2025		
Status	Active		
Total Hours	12.0		
Created By	Venkata Likith Sai Kovi	Last Modified By	Venkata Likith Sai Kovi
19/09/2025, 10:50 pm		19/09/2025, 10:50 pm	

New Contact Edit New Opportunity

Events:

 Helping Hands Man...

Home Donations Volunteers Events Reports Dashboards

Event A-0001

Related	Details		
Event ID	A-0001	Owner	Venkata Likith Sai Kovi
Event Name	Food Distribution Camp		
Event Date	21/09/2025		
Location	Gachibowli,Hyderabad		
Description	It is a Food distribution camp,where the organic and protein foods are handovered to the poor people and daily wage workers.		
Created By	Venkata Likith Sai Kovi	Last Modified By	Venkata Likith Sai Kovi
19/09/2025, 10:48 pm		19/09/2025, 10:48 pm	

VolunteerEvent (junction):

 Helping Hands Man...

Home Donations Volunteers Volunteer Tasks Events Reports Dashboards Volunteer Events

Volunteer Event VE-0001

Related	Details		
Volunteer Event Name	VE-0001		
Volunteer	A-002		
Event	A-0001		
Created By	Venkata Likith Sai Kovi	Last Modified By	Venkata Likith Sai Kovi
20/09/2025, 1:29 pm		20/09/2025, 1:29 pm	

Donations:

The screenshot shows a donation record for "DON-0001". The "Details" tab is selected, displaying fields such as Donation ID, Amount (\$1,000), Date (19/09/2025), Donor Name (Likith sai), and Donor Email (salikith51@gmail.com). The "Activity" section is also visible, showing a list of upcoming and overdue activities.

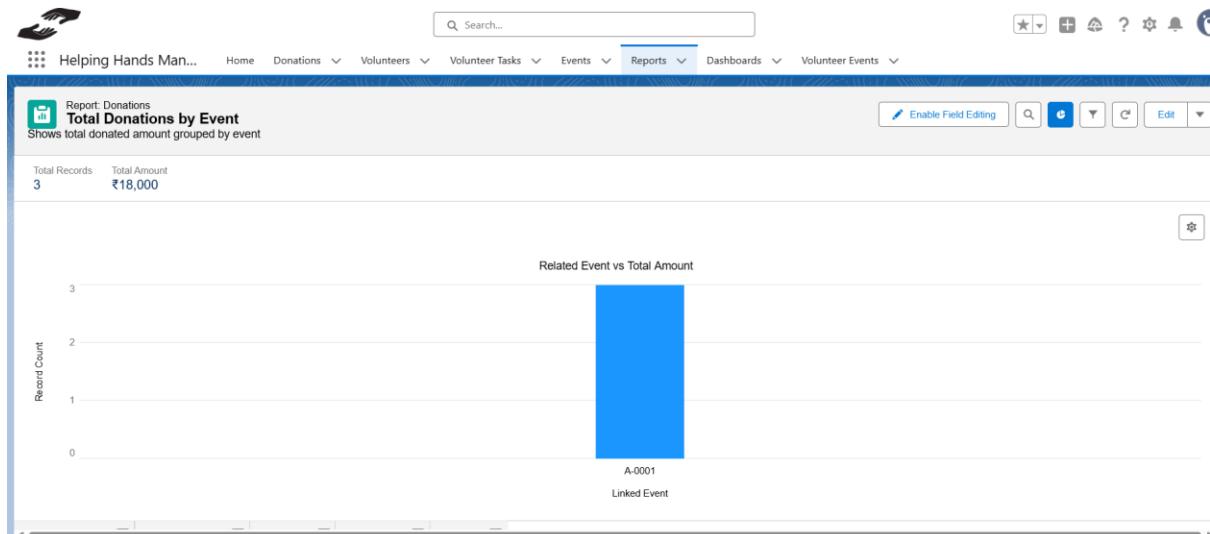
Volunteer Task(Food Donation):

The screenshot shows a volunteer task record for "Collect Food Packets". The "Details" tab is selected, displaying fields such as Volunteer Task Name, Volunteer (A-001), Status (In Progress), and Priority (High). The "Activity" section is also visible, showing a list of upcoming and overdue activities.

10. Reports & Dashboards

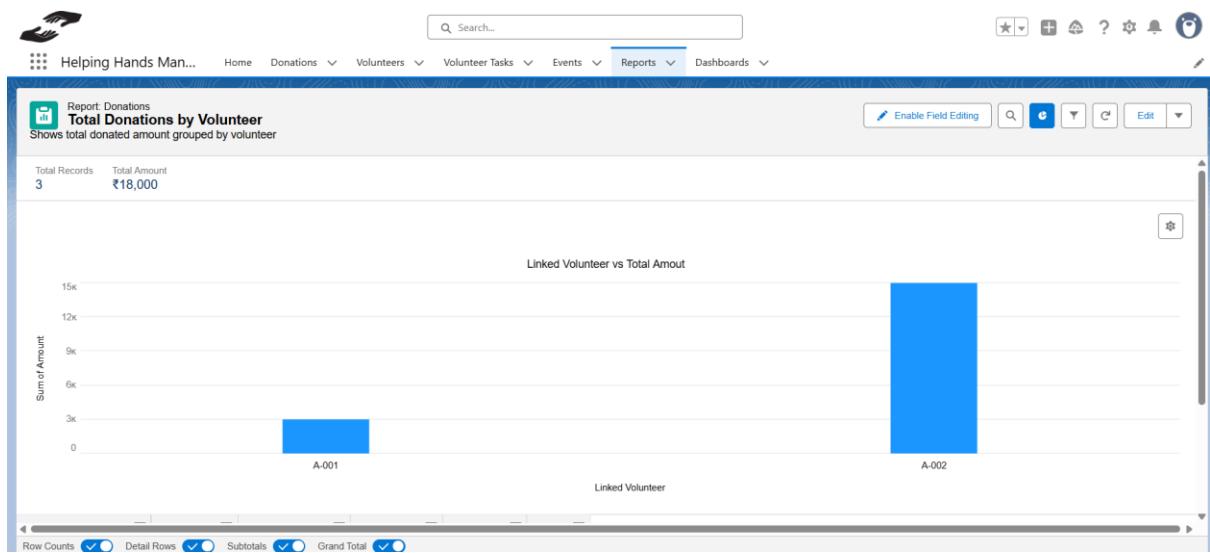
Reports to create:

1. Total Donations by Event
 - Group by Related Event, summarize Sum(Amount)
 - add chart (bar).



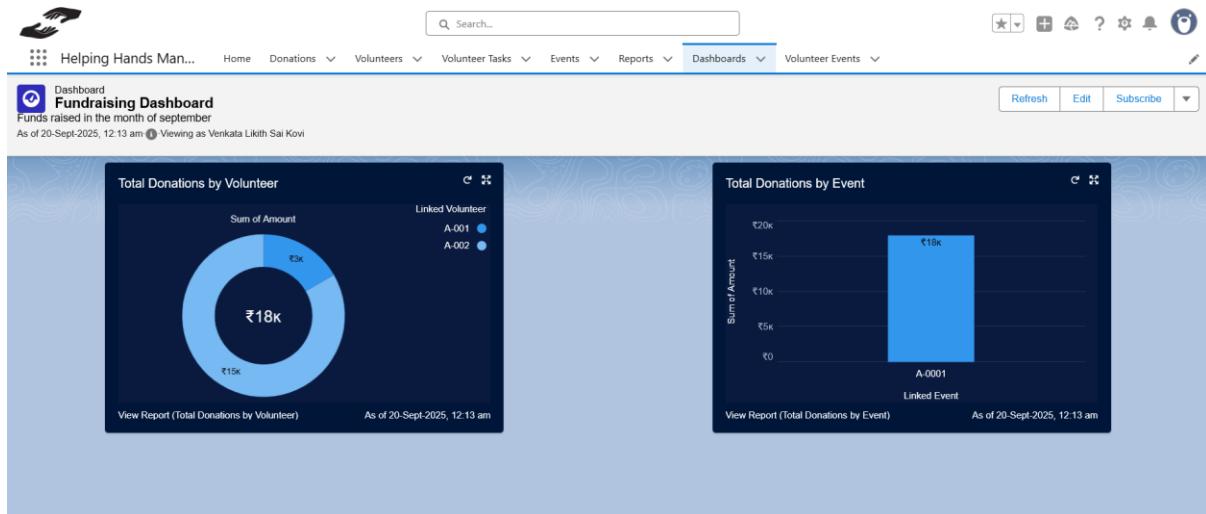
2. Donations by Volunteer

- Group by Linked Volunteer
- Sum Amount.



Dashboard:

- Dashboards → New Dashboard
- Name: Fundraising Dashboard
- Add component: select report Total Donations by Event
- choose Bar Chart
- Total Donations by Event
- Total Donations by Volunteer → Save.



11. Developer: Retrieve Metadata to SFDX & Version Control

Authenticate to Dev Org

```
sfdx force:auth:web:login -a DevOrg
```

Retrieve specific custom objects

```
sfdx force:source:retrieve -m
CustomObject:Volunteer__c,CustomObject:Event__c,CustomObject:Donation__c,CustomObject:VolunteerEvent__c,CustomObject:VolunteerTask__c -u DevOrg
```

Commit to Git

```
git add .
```

```
git commit -m "Phase 3: Add custom objects Volunteer, Event, Donation, VolunteerEvent, VolunteerTask"
```

```
git push origin main
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
<fullName>Amount__c</fullName>
<description>Enter the donation amount you collected</description>
<label>Amount</label>
<precision>18</precision>
<required>true</required>
<scale>0</scale>
<trackTrending>false</trackTrending>
<type>Currency</type>
</CustomField>
```

Phase 4 - Automation, Approval Processes, and Reporting

Summary of Phase 4 Deliverables

Phase 4 implements automation (Flows), validation rules, an approval process, scheduled summary flows, email alerts and reporting dashboards. The key deliverables are:

- Validation Rules (Donation amount > 0)
- Record-triggered Flows (Orientation task for new volunteers; Large-donation email notifications)
- Approval Process for large donations (threshold \geq ₹100,000)
- Scheduled Flow that summarises monthly donations into Donation_Summary_c
- Reports & Dashboards (Monthly Donation Summary Report, Fundraising Dashboard)
- Debug logs, test cases, and fixes for duplicate month records.

This doc reflects what was implemented in your Dev Org and includes exact resource names, labels, variables, and formulas you used.

Step 1: Validation Rule — Donation Amount Must Be Greater Than Zero

Purpose: Prevent invalid donation records where the amount is zero or negative.

This preserves reporting accuracy.

Steps:-

1. Click the Gear icon → Setup.
2. In Quick Find type: Object Manager → click Object Manager.
3. Find and click: Donation (Donation_c).
4. In left menu click: Validation Rules → New.
5. Fill fields exactly as below:
 - Rule Name: Validate_Donation_Amount
 - Description: Ensure donation amount is positive
 - Error Condition Formula:
Amount_c <= 0
 - Error Message: Donation amount must be greater than 0.
 - Error Location: Field → Amount

6. Click Save, then Activate (if not active automatically).

The screenshot shows the 'Validation Rule Detail' section of a page titled 'Donation Validation Rule'. The rule name is 'Donation_Amount_Positive'. The 'Active' checkbox is checked. The error condition formula is 'Amount__c <= 0'. The error message is 'Donation Amount must be greater than 0.' The description is 'Ensure donation amount is positive'. The created by field shows 'Venkata Likith Sai Kovi, 20/09/2025, 7:46 pm'. The modified by field shows 'Venkata Likith Sai Kovi, 22/09/2025, 9:48 am'. There are 'Edit' and 'Clone' buttons at the bottom.

Test case :

- Create a Donation record with Amount = -1
 - Attempt to save
 - Expected Result: Validation error displayed near the Amount field.

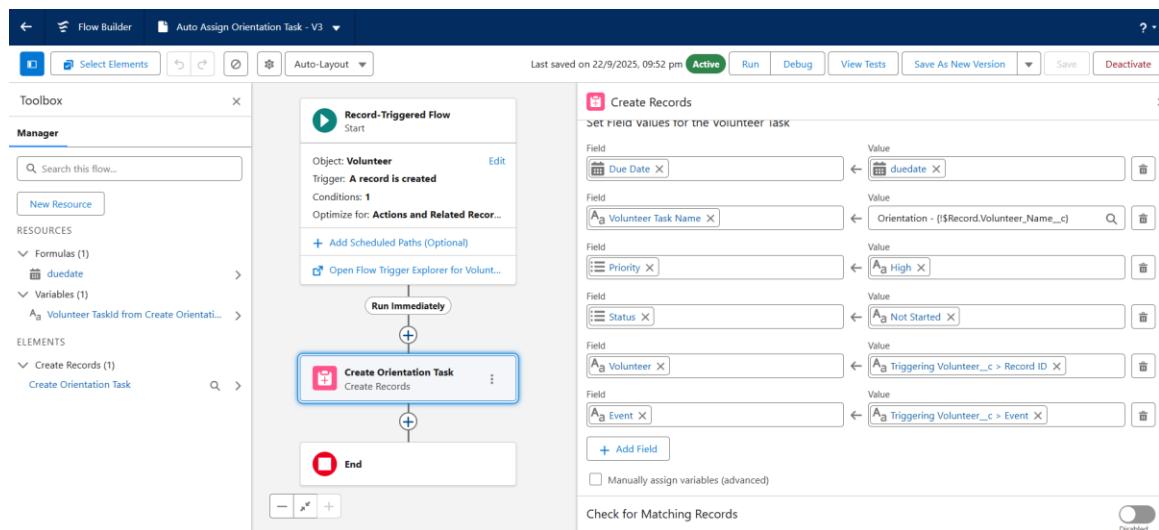
The screenshot shows the 'Information' tab of a 'New Donation: one-time' form. The 'Amount' field contains '-₹1'. A validation error message 'Donation Amount must be greater than 0.' is displayed below it. The 'Owner' field is set to 'Venkata Likith Sai Kovi'. Other fields like 'Donation Date' (14/09/2025), 'Donor Name' (vivek), and 'Donor Email' (vivek@gmail.com) are filled. A modal window titled 'We hit a snag.' displays the error message 'Review the following fields • Amount'. At the bottom, there are 'Cancel', 'Save & New', and 'Save' buttons.

Step 2: Record-Triggered Flow — Auto-assign 'Orientation' Volunteer Task

Purpose: When a new Volunteer_c record is created, automatically create a VolunteerTask_c record for orientation so onboarding is tracked and consistent.

Implementation details (exact):

1. Setup → Quick Find → Flows → Flows → New Flow.
2. Choose: Record-Triggered Flow → Create.
3. Configure Start:
 - Object: Volunteer (Volunteer_c)
 - Trigger: A record is created
 - Condition Requirements: None (or Status_c = 'Onboarding' if you only want to run for that status)
 - Optimize the Flow for: Actions and Related Records
4. Click Done.
5. Add a Create Records element:
 - Label: Create Orientation Task
 - Create One Volunteer Task (VolunteerTask_c)
 - Set Field Values:
 - Name (Task Name): 'Orientation - ' & {!\$Record.Volunteer_Name_c}
 - Assigned_Volunteer_c: {!\$Record.Id}
 - Status_c: Not Started
 - Priority_c: High
 - Due_Date_c: Use a Date Formula Resource (DueDateFormat) = TODAY() + 7
6. Save Flow Label: Auto_Assign_Orientation_Task. Save and Activate.



Test steps :

1. App Launcher → Helping Hands → Volunteers → New Volunteer.
2. Fill Volunteer_Name_c = 'Test Volunteer Onboard', other fields as needed → Save.
3. Open the volunteer record → scroll to Volunteer Tasks related list → Confirm 'Orientation - Test Volunteer Onboard' exists.

The screenshot shows the Salesforce Lightning Experience. At the top, there's a navigation bar with icons for Home, Donations, Volunteers, Volunteer Tasks, Events, Volunteer Events, Reports, Dashboards, and a Home button. Below the navigation is a search bar and a toolbar with buttons for New Contact, Edit, and New Opportunity. The main area displays a volunteer record for 'A-017'. Under the 'Related' tab, there's a section for 'Volunteer Tasks (1)'. A table shows one task: 'Orientation - aravind' with Status 'Not Started', Due Date '29/9/2025', and Priority 'High'. There are 'New' buttons for each section: Volunteer Tasks, Donations (0), and Volunteer Events (0). A 'View All' link is also present.

Step 3: Record-Triggered Flow — Email Alert for Large Donations (>= ₹10,000)

Purpose: Notify Finance Officer (or a Role) when any donation equal to or greater than ₹10,000 is created so that high-value gifts receive timely attention.

Implementation steps :

- A. Create Email Template (Lightning Email Template):
 1. Setup → Email Templates → New Email Template.
 2. Template Name: Large Donation Alert
 3. Subject: Large donation received: {!Donation_c.Donor_Name_c} - {!Donation_c.Amount_c}
 4. Body: include merge fields for Donation_c.Donor_Name_c, Donation_c.Amount_c, Donation_c.Donation_Date_c and link to the Donation record.
 5. Save in folder: Helping Hands Templates (or Unfiled Public Email Templates).

Email Template
Large_Donation_Alert

Details Related

Information

Email Template Name Large_Donation_Alert	Related Entity Type Folder Helping Hands Project
Description	
Made in Email Template Builder <input type="checkbox"/>	

Message Content

Subject {{Donation__c.Donor_Name_c}} - {{Donation__c.Amount_c}}	Enhanced Letterhead
HTML Value Subject: {{Donation__c.Donor_Name_c}} - {{Donation__c.Amount_c}}	
Hello Finance Team, A donation of {{Donation__c.Amount_c}} (Donation ID: {{Donation__e.Name}}) was received on {{Donation__c.Donation_Date_c}} by {{Donation__c.Donor_Name_c}}. Linked Volunteer: {{Donation__c.Linked_Volunteer_c}} Please review and take necessary action. Thanks, Helping Hands Management	

B. Create Email Alert:

1. Setup → Quick Find → Email Alerts → New Email Alert.
2. Object: Donation
3. Email Template: Large Donation Acknowledgement
4. Recipient Type: Role → Financial Officer (or specific user)
5. Save

Email Alert
Alert message in Large_Donation_To_Finance

Rules Using This Email Alert | Approval Processes Using This Email Alert | Entitlement Processes Using This Email Alert

Help for this Page

Email Alert Detail

Description Unique Name From Email Address	Alert message in Large_Donation_To_Finance Alert_Large_Donation_To_Finance Default Workflow User's email address	Email Template Object
Recipients	User_Finance Officer User_Volunteer User_Donation Manager	Large_Donation_Alert Donation
Additional Emails	sailikm511@gmail.com	Modified By Venkata Likith Sai Koyi 23/09/2025, 10:55 pm
Created By	Venkata Likith Sai Koyi 20/09/2025, 10:51 pm	

Rules Using This Email Alert
This alert is currently not used by any rules

Approval Processes Using This Email Alert
This alert is currently not used by any approval processes

Entitlement Processes Using This Email Alert
This alert is currently not used by any entitlement processes

Flows Using This Email Alert

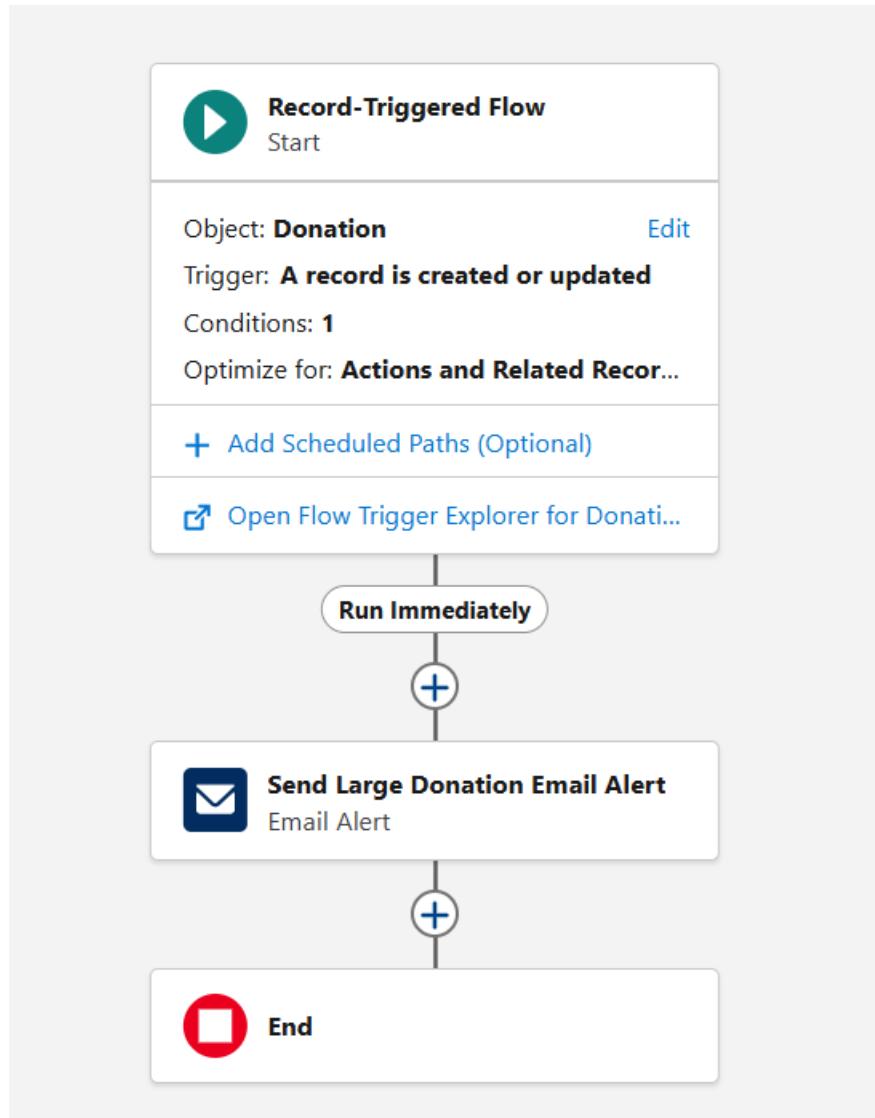
Flow Name	Version	Description	Object	Active
Notify_Finance_on_Large_Donation	1, 2		01gK000002KSIR	✓

C. Record-Triggered Flow:

1. Setup → Flows → New Flow → Record-Triggered Flow
2. Object: Donation (Donation__c)
3. Trigger: A record is created or updated
4. Condition Requirements: Amount__c >= 10000
5. Run After Save (so email actions can execute)
6. Add an Action element → select the Email Alert

(EA_LargeDonation_NotifyFinance)

7. Save Flow Label: Notify_Finance_on_Large_Donation. Activate.



Test example :

- Create Donation record with Amount = 15000, Donor_Name_c = 'Test Donor Email'.
- Save → Check the Financial Officer's email inbox or the email logs in Salesforce.

Donation
DON-0011

Related	Details
Donation ID	DON-0011
Amount	₹35,000
Donation Date	22/09/2025
Donor Name	Sri Lakshmi
Donor Email	srilakshmi@gmail.com
Payment Method	Card
Status	Received
Linked Volunteer	A-017
Notes	
Linked Event	A-0001
Created By	Venkata Likith Sai Kovi, 23/09/2025, 11:28 pm
Last Modified By	Venkata Likith Sai Kovi, 23/09/2025, 11:28 pm

Sri Lakshmi - ₹35,000 [Inbox](#)

◆ Summarise this email

L Finance Officer via h5hh4dc2zvbljj.gk-7ydv7ua0.can96.bnc.salesforce.com
to me, saililkith57@gmail.com ▾
Subject: Sri Lakshmi - ₹35,000

23:28 (0 minutes ago) [Star](#) [Reply](#) [Forward](#) [More](#)

Hello Finance Team,

A donation of ₹35,000 (Donation ID: DON-0011) was received on 22/09/2025 by Sri Lakshmi.
Linked Volunteer: A-017

Please review and take necessary action.

Thanks,
Helping Hands Management

Step 4: Approval Process — Large Donation Approval (>= ₹100,000)

Purpose: Implement a formal approval process for very large donations to ensure financial oversight. You implemented an approval process that routes donations >= ₹1,00,000 to the Finance Officer.

Steps performed (Jump Start Wizard approach):

1. Setup → Quick Find → Approval Processes → Approval Processes.
2. Select Object: Donation.
3. Click Create New Approval Process → Use Jump Start Wizard.
4. Fill: Name = Large Donation Approval, Description as needed.
5. Entry Criteria: Amount_c greater or equal 100000.
6. Select Approver: Automatically assign to approver(s) → User: choose Finance Officer (or CFO). For testing you used System Administrator user.
7. Final Approval Action → Add New → Field Update: set Status_c = 'Approved'
8. Final Rejection Action → Add New → Field Update: set Status_c = 'Rejected'
9. Save → Activate the approval process.

The screenshot shows the 'Approval Processes' page in Salesforce. The process is named 'Large Donation Approval'. It has the following details:

- Process Name:** Large Donation Approval
- Unique Name:** Large_Donation_Approval
- Description:** Approval process for Donations requiring admin approval when amount exceeds 1,00,000.
- Entry Criteria:** Amount_c >= 100000
- Record Editability:** Administrator ONLY
- Next Automated Approver Determined By:** Active (checked)
- Allow Submitters to Recall Approval Requests:** Unchecked
- Initial Submitters:** Donation Center
- Created By:** Venkata Likhith Koyi (23/08/2025, 11:54 pm)
- Modified By:** Venkata Likhith Koyi (24/09/2025, 12:18 am)

Initial Submission Actions:

Action	Type	Description
Record Lock		Lock the record from being edited

Approval Steps:

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions Edit	1	High Value Donation Approval Step	Approves donations of ₹100,000 or more	Amount_c >= 100000, else Approve	User: Finance Officer	Final Rejection

Final Approval Actions:

Action	Type	Description
Edit Remove	Record Lock	Lock the record from being edited
Edit Remove	Field Update	Mark Donation as Approved

Final Rejection Actions:

Action	Type	Description
Edit Remove	Record Lock	Unlock the record for editing
Edit Remove	Field Update	Mark Donation as Rejected

Approval Step (added):

- Created an Approval Step named 'High Value Donation Approval Step'.
- Step Criteria: Amount_c >= 100000
- Assigned Approver: System Administrator (for testing), later set to Financial Officer role.

The screenshot shows an email inbox with one new message from 'Finance Officer' via y7s6w7yoxowwbqm.gk-7ydv7ua0.can96.bnc.salesforce.com. The message subject is 'Ram - ₹2,00,000' and the body contains the following text:

Hello Finance Team,

A donation of ₹2,00,000 (Donation ID: DON-0012) was received on 25/09/2025 by Ram.
Linked Volunteer: A-002

Please review and take necessary action.

Thanks,
Helping Hands Management

Step 5: Scheduled Flow — Monthly Donation Summary (Detailed)

Purpose: Create a scheduled automation that computes monthly donation totals and stores them in `Donation_Summary__c`. Because Lightning Scheduled Flow may not show 'Monthly' frequency, we scheduled it Daily and used a Decision to run only on the first of the month.

Flow design steps:-

1. Flow type: Schedule-Triggered Flow (configured as Daily; Decision checks `DAY(TODAY()) = 1`)

2. Flow Label: Monthly Donation Summary (Daily Check)

3. Manager Resources created:

- Variable: `TotalAmount` (Number, scale 2, default 0)
- Variable: `DonationCount` (Number, scale 0, default 0)
- Formula: `MonthYearText` (Text) = `TEXT(MONTH(TODAY())) & "-" & TEXT(YEAR(TODAY()))`
- Formula (`IsFirstOfMonth`) or Decision: `DAY(TODAY()) = 1`

4. Schedule: Start Date = (for testing) 24-Sep-2025; Frequency = Daily; Start Time = as selected

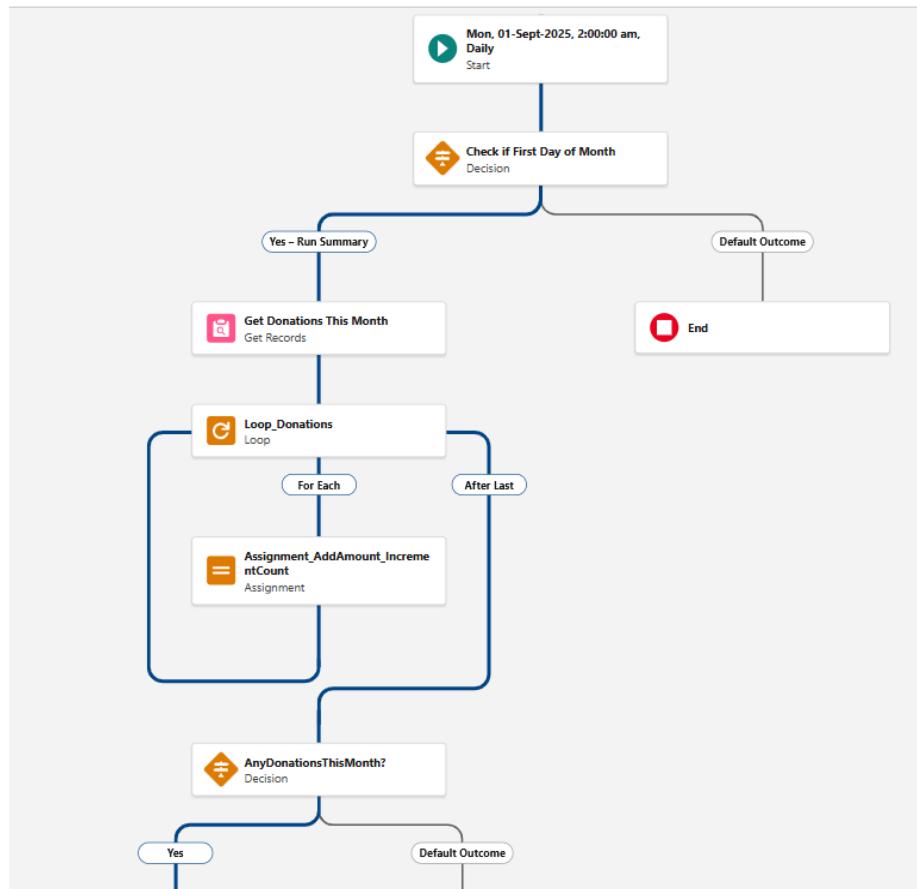
5. Flow Elements and exact labels used:

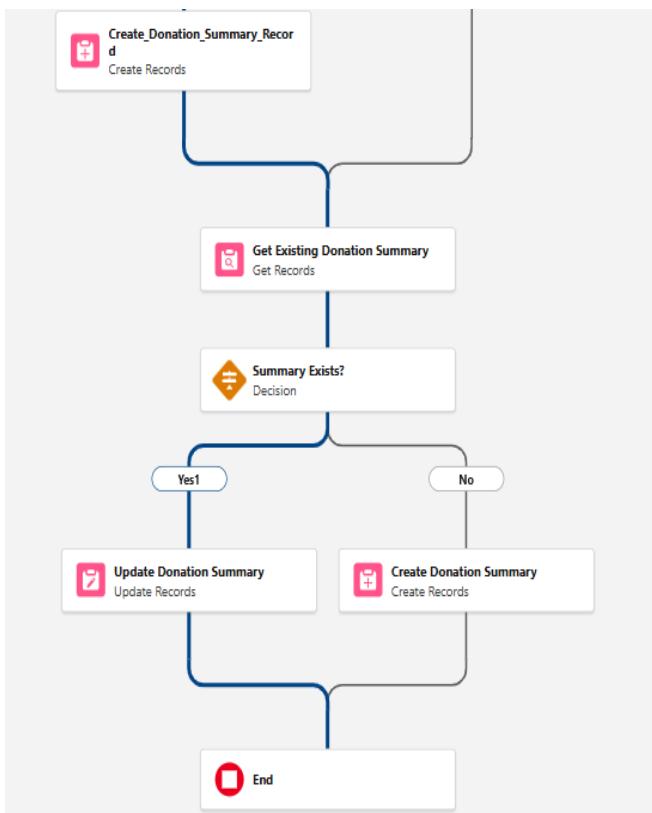
- Start (Schedule) -> Decision (Check if First Day of Month)
Decision Outcome: `Yes_Run_Summary` if `DAY(TODAY()) = 1` OR `RunNow` boolean is True

- Get Records: `Get_Donations_This_Month`
 - * Object: `Donation__c`
 - * Condition: `Donation_Date__c` equals `THIS_MONTH` (or use date range variables `StartOfPrevMonth/EndOfPrevMonth` depending on design)
 - * How Many: All records; Store: All fields
- Loop: `Loop_Donations` (iterate `Get_Donations_This_Month` collection)
 - Assignment element inside loop: `Assignment_AddAmount_IncrementCount`
 - * `TotalAmount = TotalAmount + {!Loop_Donations.Amount__c}`
 - * `DonationCount = DonationCount + 1`

- After loop: Get Records: Get_Existing_Donation_Summary
 - * Object: Donation_Summary_c
 - * Condition: Month_c equals {!MonthYearText} (or {!date} depending on naming)
 - * How many: Only the first record (we check existence)
- Decision: Summary Exists?
 - * Yes path -> Update Donation Summary
 - Update record (where Id = {!Get_Existing_Donation_Summary.Id}) set:
 - Total_Amount_c = {!TotalAmount}
 - Total_Donations_c = {!DonationCount}
 - * No path -> Create Donation Summary
 - Create record Donation_Summary_c with fields:
 - Month_c = {!MonthYearText}
 - Total_Amount_c = {!TotalAmount}
 - Total_Donations_c = {!DonationCount}

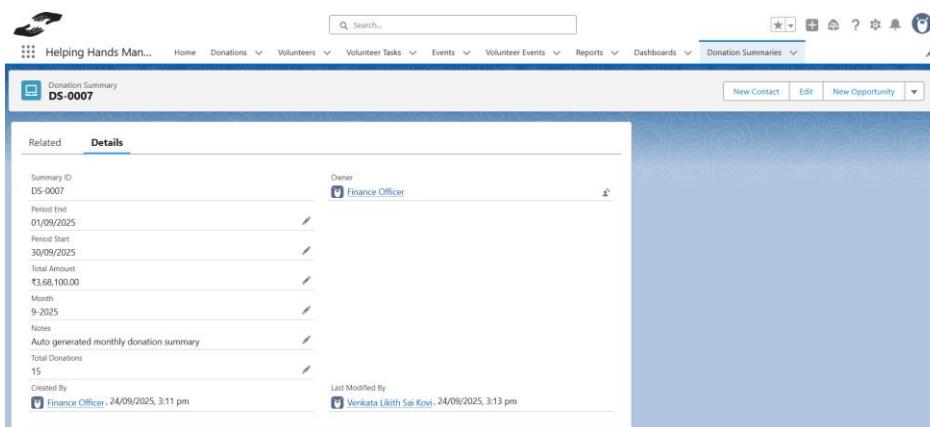
6. Save as 'Monthly Donation Summary (Daily Check)' and Activate.





Test examples & debug steps :

- Created test donations for Sept-2025 (various amounts) and ran Debug with RunNow = True to bypass date check.
- Observed Debug log showing Decision 'Summary Exists?' = Yes and Update path executed.
- Debug outputs showed values: TotalAmount = 3,21,100.00; DonationCount = 12 (example run). Later run produced TotalAmount = 3,68,100.00; DonationCount = 15.
- Important: Debug default runs in rollback mode; uncheck 'Run flow in rollback mode' to persist changes.



Step 6: Reports & Dashboards — Detailed Build

Objective: Provide Finance and Executive stakeholders with actionable visuals summarizing donations, donors, and volunteer contributions.

Report : Monthly Donation Summary Report (detailed build):

1. App Launcher → Reports → New Report.

2. Select Report Type: Donations (Donation_c) → Continue.

3. In the Report Builder:

- Add Columns: Donation ID (Name), Donor_Name_c, Amount_c, Donation_Date_c, Linked_Volunteer_c, Payment_Method_c, Status_c
- Filters: Date Range = This Year (or All Time for historical). Remove default 'My Donations'.
- Group Rows by: Calendar Month (Donation Date) -> To do this: click Add Group -> Donation Date -> select 'Calendar Month'
- Summaries: click the chevron on Amount_c -> Summarize -> Sum. Also add Row Count for number of donations.
- Optional: add filter Status_c = Received to show only completed donations.

4. Save Report Name: Monthly Donation Summary Report. Folder: Helping Hands Reports.

Report: Donations Monthly Donation Summary Report				
	Amount	Donor Name	Status	Donation: Donation ID
□ 19/09/2025 (1)	₹3,000	Likhil sai	Received	DON-0001
Subtotal				₹3,000
□ 22/09/2025 (2)	₹18,000	Sumith	Received	DON-0007
	₹35,000	Sri Lakshmi	Received	DON-0011
Subtotal				₹53,000
□ 23/09/2025 (1)	₹15,000	Umesh	Received	DON-0006
Subtotal				₹15,000
□ 24/09/2025 (7)	₹25,000	Sampath	Pledged	DON-0008
	₹6,500	test3	Received	DON-0015
	₹2,500	test2	Received	DON-0014
	₹1,000	test1	Pledged	DON-0013
	₹10,000	test a	Pledged	DON-0016
	₹25,000	testb	Pledged	DON-0017
	₹12,000	testc	Pledged	DON-0018
Subtotal				₹82,000
□ 25/09/2025 (1)	₹2,00,000	Ram	Received	DON-0012
Subtotal				₹2,00,000
Total (15)				₹3,68,100

Row Counts Detail Rows Subtotals Grand Total

Dashboard: 2025-September Month Donation Dashboard

1. App Launcher → Dashboards → New Dashboard.

2. Name: Helping Hands Donation Dashboard. Folder: Helping Hands Reports.

3. Add Components:

- Component 1: Bar Chart -> Source report: Monthly Donation Summary Report -> X-axis: Month, Y-axis: Sum of Amount

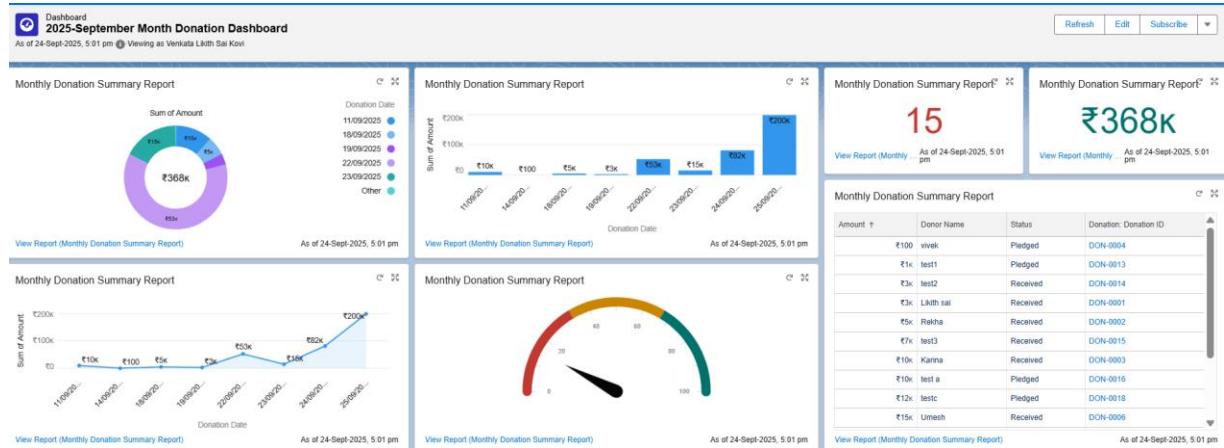
- Component 2: Donut/Pie -> Source report: Donations by Payment Method (create an ad-hoc report grouped by Payment_Method_c)

- Component 3: Metric tile -> Source report: Monthly Donation Summary -> configure to show total for current month

- Component 4: Table -> Top 10 donors by amount (create a report that sorts Amount desc and limit to 10)

4. Configure component titles, display units, and filters as needed.

5. Save Dashboard and share folder permissions with Finance Officer and Admins.



Phase - 5 Apex Programming

Part 1: Creating the Apex Files

First, we need to create each of the necessary Apex files in your Salesforce org.

Resource:- using the Developer Console.

Step 1.1: Open the Developer Console

1. Log in to your Salesforce org.
2. Click the **Gear Icon** in the top-right corner.
3. Select **Developer Console** from the dropdown menu.

Step 1.2: Create Each Apex File

For each of the six files listed below, you will follow the same process:

- In the Developer Console, click **File → New → Apex Class** (or **Apex Trigger** for the .trigger file).
- Enter the exact file name provided.
- Delete any default text in the editor window.
- Copy and paste the complete code for that file.
- Click **File → Save** or press Ctrl + S.

Create the files in this order:

1. **VolunteerTriggerHandler.apxc (Class)**

```

1 * public class VolunteerTriggerHandler {
2     private static Boolean isRunning = false;
3
4     public static void updateVolunteerTotals(List<Volunteer__c> volunteers){
5         if(isRunning) return;
6         isRunning = true;
7
8         Set<Id> volunteerIds = new Set<Id>();
9         for(Volunteer__c v : volunteers){
10             if(v.Id != null){
11                 volunteerIds.add(v.Id);
12             }
13         }
14         if(volunteerIds.isEmpty()) {
15             isRunning = false;
16             return;
17         }
18
19         // 1. Aggregate Donations
20         Map<Id, Decimal> donationTotals = new Map<Id, Decimal>();
21         for(AggregateResult ar : [SELECT Linked_Volunteer__c vId, SUM(Amount__c) total FROM Donation__c WHERE Linked_Volunteer__c IN :volunteerIds GROUP BY Linked_Volunteer__c]){
22             donationTotals.put((Id)ar.get('vId'), (Decimal)ar.get('total'));
23         }
24
25         // 2. Aggregate Hours using your confirmed field names
26         Map<Id, Decimal> hourTotals = new Map<Id, Decimal>();
27         for(AggregateResult ar : [SELECT Volunteer__c vId, SUM(Hours_Worked__c) totalHours FROM Volunteer_Task__c WHERE Volunteer__c IN :volunteerIds GROUP BY Volunteer__c]){
28             hourTotals.put((Id)ar.get('vId'), (Decimal)ar.get('totalHours'));
29         }
30
31         // 3. Prepare updates
32         List<Volunteer__c> updates = new List<Volunteer__c>();
33         List<Volunteer__c> existingVolunteers = [SELECT Id, Total_Donations__c, Total_Hours__c FROM Volunteer__c WHERE Id IN :volunteerIds];
34
35         for(Volunteer__c existing : existingVolunteers){
36             Decimal newDonations = donationTotals.get(existing.Id) == null ? 0 : donationTotals.get(existing.Id);
37             Decimal newHours = hourTotals.get(existing.Id) == null ? 0 : hourTotals.get(existing.Id);
38
39             if(existing.Total_Donations__c != newDonations || existing.Total_Hours__c != newHours){
40                 Volunteer__c vUpdate = new Volunteer__c(Id = existing.Id);
41                 vUpdate.Total_Donations__c = newDonations;
42                 vUpdate.Total_Hours__c = newHours;
43                 updates.add(vUpdate);
44             }
45         }
46
47         if(!updates.isEmpty()){
48             update updates;
49         }
50
51         isRunning = false;
52     }
53 }

```

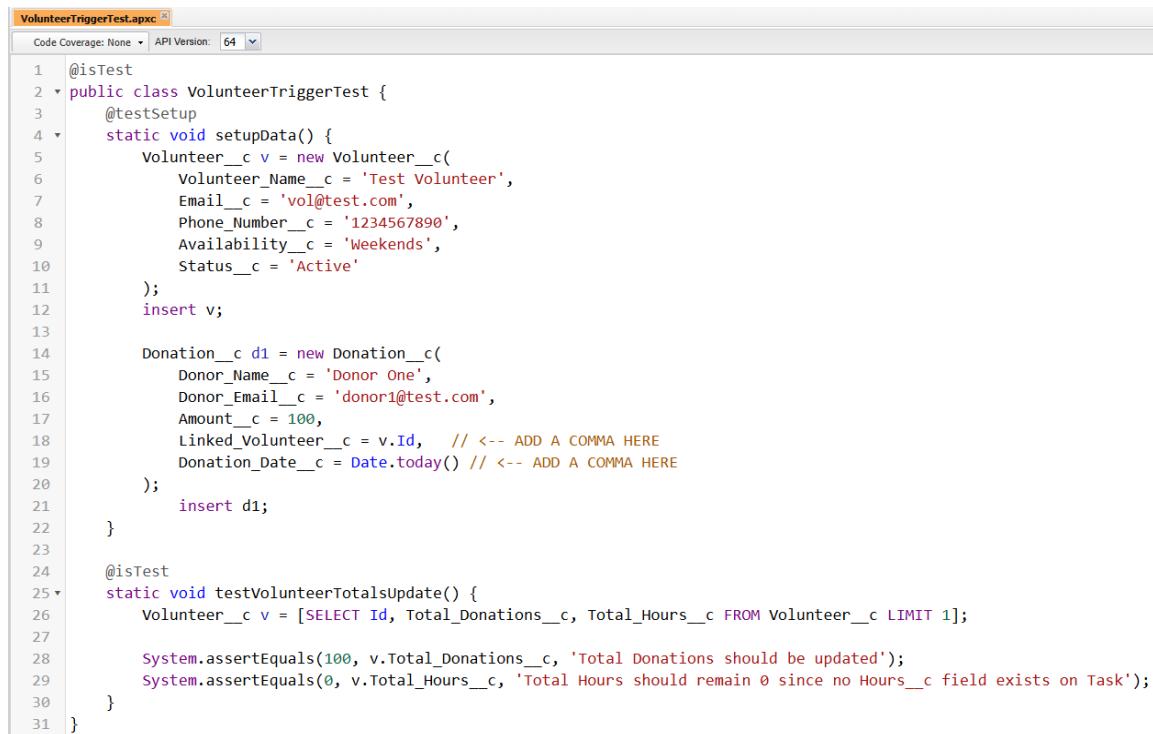
2. VolunteerTrigger.apxt (Trigger)

```

1 trigger VolunteerTrigger on Volunteer__c (after insert, after update) {
2     if(Trigger.isAfter){
3         VolunteerTriggerHandler.updateVolunteerTotals(Trigger.new);
4     }
5 }

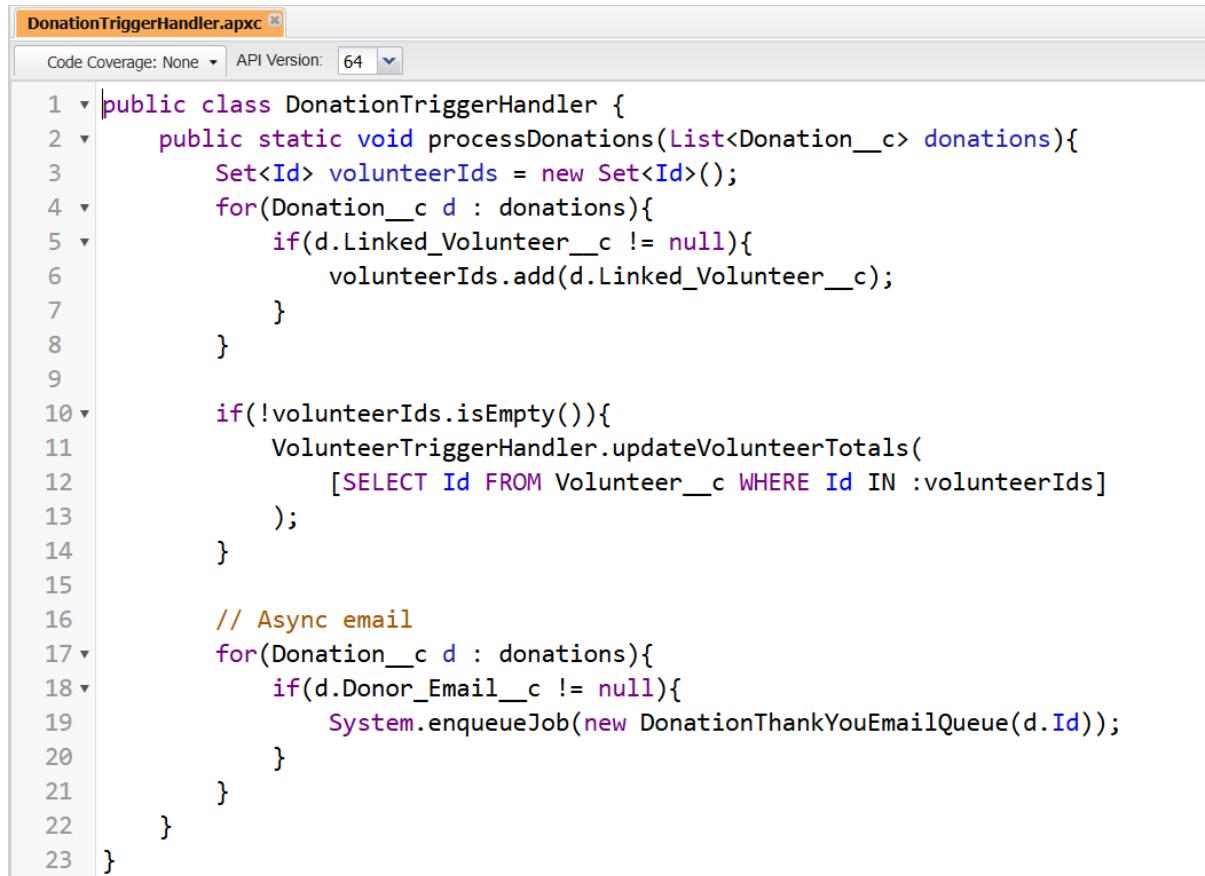
```

3. VolunteerTriggerTest.apxc (Class)



```
1  @isTest
2  public class VolunteerTriggerTest {
3      @testSetup
4      static void setupData() {
5          Volunteer__c v = new Volunteer__c(
6              Volunteer_Name__c = 'Test Volunteer',
7              Email__c = 'vol@test.com',
8              Phone_Number__c = '1234567890',
9              Availability__c = 'Weekends',
10             Status__c = 'Active'
11         );
12         insert v;
13
14         Donation__c d1 = new Donation__c(
15             Donor_Name__c = 'Donor One',
16             Donor_Email__c = 'donor1@test.com',
17             Amount__c = 100,
18             Linked_Volunteer__c = v.Id, // <-- ADD A COMMA HERE
19             Donation_Date__c = Date.today() // <-- ADD A COMMA HERE
20         );
21         insert d1;
22     }
23
24     @isTest
25     static void testVolunteerTotalsUpdate() {
26         Volunteer__c v = [SELECT Id, Total_Donations__c, Total_Hours__c FROM Volunteer__c LIMIT 1];
27
28         System.assertEquals(100, v.Total_Donations__c, 'Total Donations should be updated');
29         System.assertEquals(0, v.Total_Hours__c, 'Total Hours should remain 0 since no Hours__c field exists on Task');
30     }
31 }
```

4. DonationTriggerHandler.apxc (Class)



```
1  public class DonationTriggerHandler {
2      public static void processDonations(List<Donation__c> donations){
3          Set<Id> volunteerIds = new Set<Id>();
4          for(Donation__c d : donations){
5              if(d.Linked_Volunteer__c != null){
6                  volunteerIds.add(d.Linked_Volunteer__c);
7              }
8          }
9
10         if(!volunteerIds.isEmpty()){
11             VolunteerTriggerHandler.updateVolunteerTotals(
12                 [SELECT Id FROM Volunteer__c WHERE Id IN :volunteerIds]
13             );
14         }
15
16         // Async email
17         for(Donation__c d : donations){
18             if(d.Donor_Email__c != null){
19                 System.enqueueJob(new DonationThankYouEmailQueue(d.Id));
20             }
21         }
22     }
23 }
```

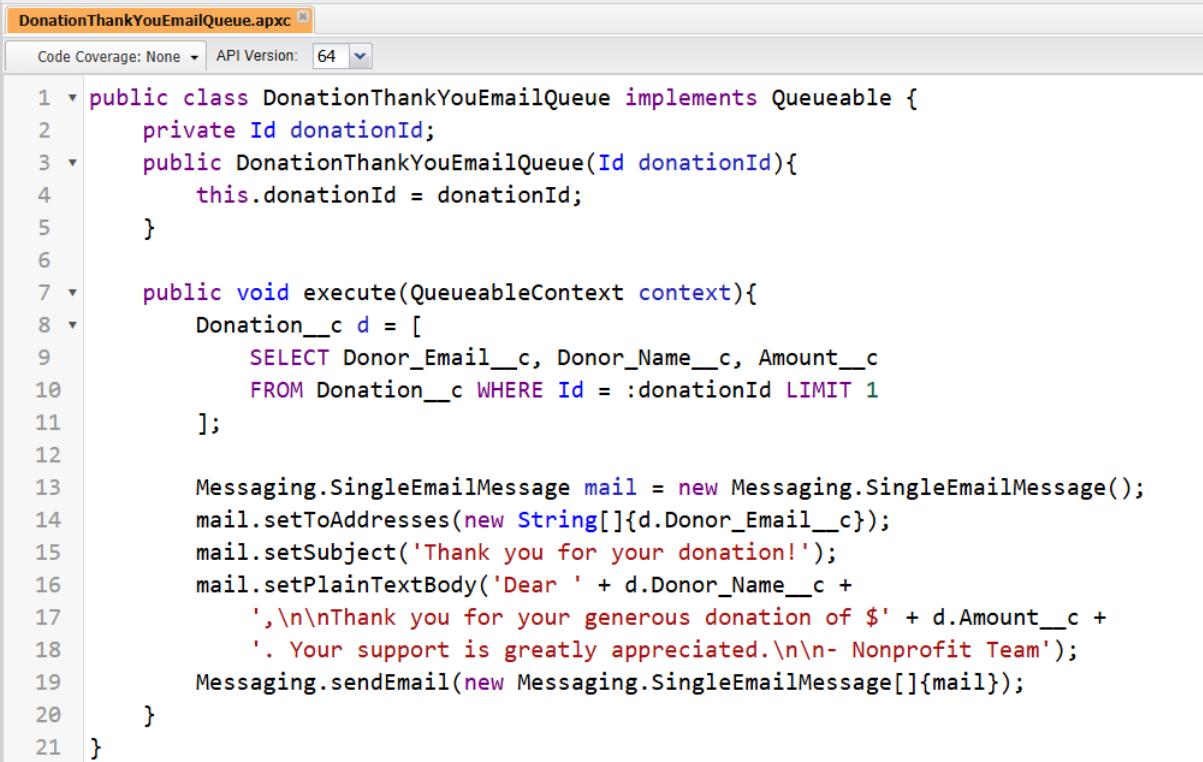
5. DonationTrigger.apxt (Trigger)

```
DonationTrigger.apxt
Code Coverage: None API Version: 64
1 trigger DonationTrigger on Donation__c (after insert, after update) {
2     if(Trigger.isAfter){
3         DonationTriggerHandler.processDonations(Trigger.new);
4     }
5 }
```

6. DonationTriggerTest.apxc (Class)

```
File Edit Debug Test Workspace Help < >
DonationTriggerTest.apxc
Code Coverage: None API Version: 64
1 @isTest
2 public class DonationTriggerTest {
3     @isTest
4     static void testDonationProcessingAndEmail() {
5         Volunteer__c v = new Volunteer__c(
6             Volunteer_Name__c = 'Donor Volunteer',
7             Email__c = 'donorvol@test.com',
8             Phone_Number__c = '9999999999',
9             Availability__c = 'Weekdays',
10            Status__c = 'Active'
11        );
12        insert v;
13
14        Test.startTest();
15        Donation__c d = new Donation__c(
16            Donor_Name__c = 'Test Donor',
17            Donor_Email__c = 'thankyou@test.com',
18            Amount__c = 200,
19            Linked_Volunteer__c = v.Id,
20            Donation_Date__c = Date.today()
21        );
22        insert d;
23        Test.stopTest(); // ensures async Queueable job executes
24
25        v = [SELECT Total_Donations__c FROM Volunteer__c WHERE Id = :v.Id];
26        System.assertEquals(200, v.Total_Donations__c, 'Volunteer total donations should update after donation');
27    }
28 }
```

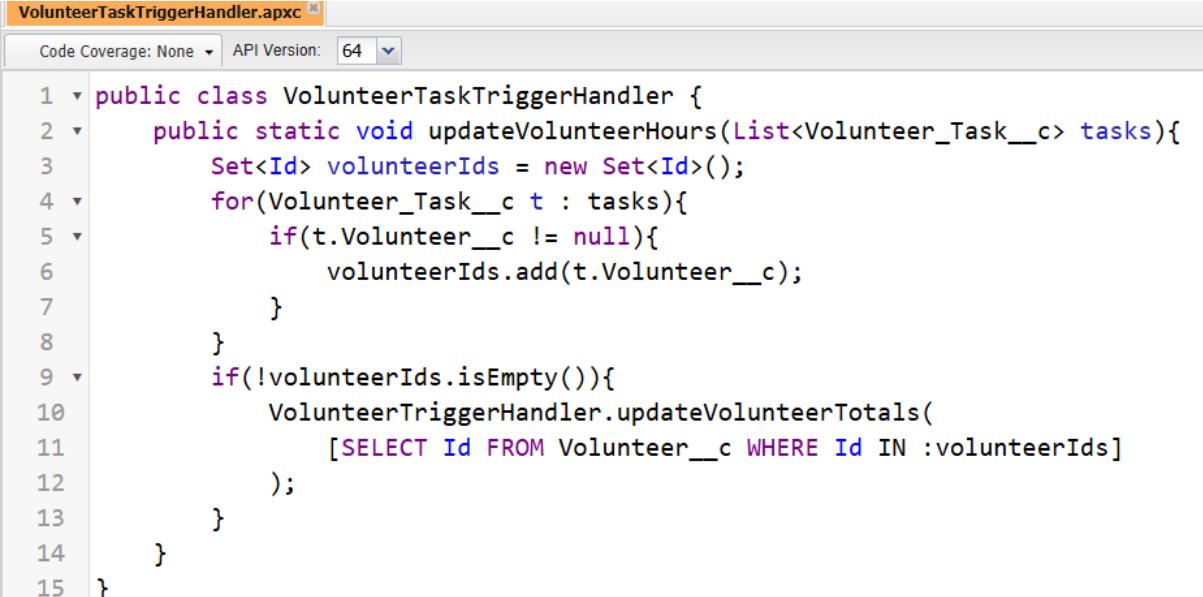
7. DonationThankYouEmailQueue.apxc (Class)



The screenshot shows the Salesforce code editor with the file 'DonationThankYouEmailQueue.apxc' open. The code implements a Queueable interface to send a thank-you email to a donor based on a donation ID.

```
1 public class DonationThankYouEmailQueue implements Queueable {
2     private Id donationId;
3     public DonationThankYouEmailQueue(Id donationId){
4         this.donationId = donationId;
5     }
6
7     public void execute(QueueableContext context){
8         Donation__c d = [
9             SELECT Donor_Email__c, Donor_Name__c, Amount__c
10            FROM Donation__c WHERE Id = :donationId LIMIT 1
11        ];
12
13        Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
14        mail.setToAddresses(new String[]{d.Donor_Email__c});
15        mail.setSubject('Thank you for your donation!');
16        mail.setPlainTextBody('Dear ' + d.Donor_Name__c +
17            ',\n\nThank you for your generous donation of $' + d.Amount__c +
18            '. Your support is greatly appreciated.\n\n- Nonprofit Team');
19        Messaging.sendEmail(new Messaging.SingleEmailMessage[]{mail});
20    }
21 }
```

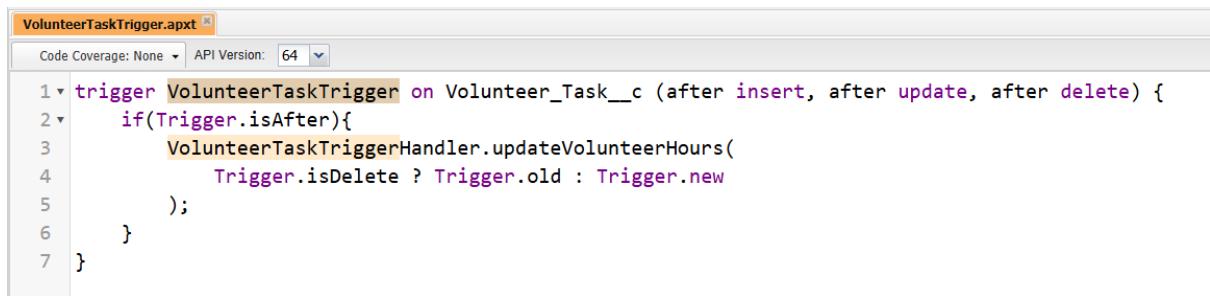
8. VolunteerTaskTriggerHandler.apxc (Class)



The screenshot shows the Salesforce code editor with the file 'VolunteerTaskTriggerHandler.apxc' open. This class contains a static method 'updateVolunteerHours' that updates volunteer hours based on tasks.

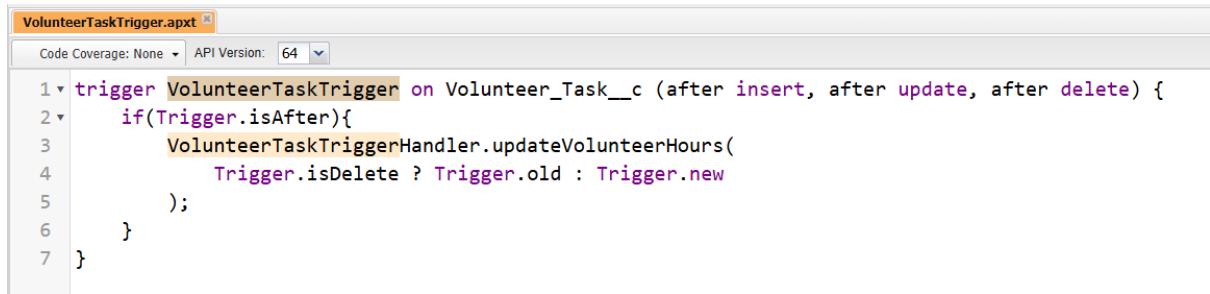
```
1 public class VolunteerTaskTriggerHandler {
2     public static void updateVolunteerHours(List<Volunteer_Task__c> tasks){
3         Set<Id> volunteerIds = new Set<Id>();
4         for(Volunteer_Task__c t : tasks){
5             if(t.Volunteer__c != null){
6                 volunteerIds.add(t.Volunteer__c);
7             }
8         }
9         if(!volunteerIds.isEmpty()){
10             VolunteerTriggerHandler.updateVolunteerTotals(
11                 [SELECT Id FROM Volunteer__c WHERE Id IN :volunteerIds]
12             );
13         }
14     }
15 }
```

9. VolunteerTaskTrigger (Trigger)



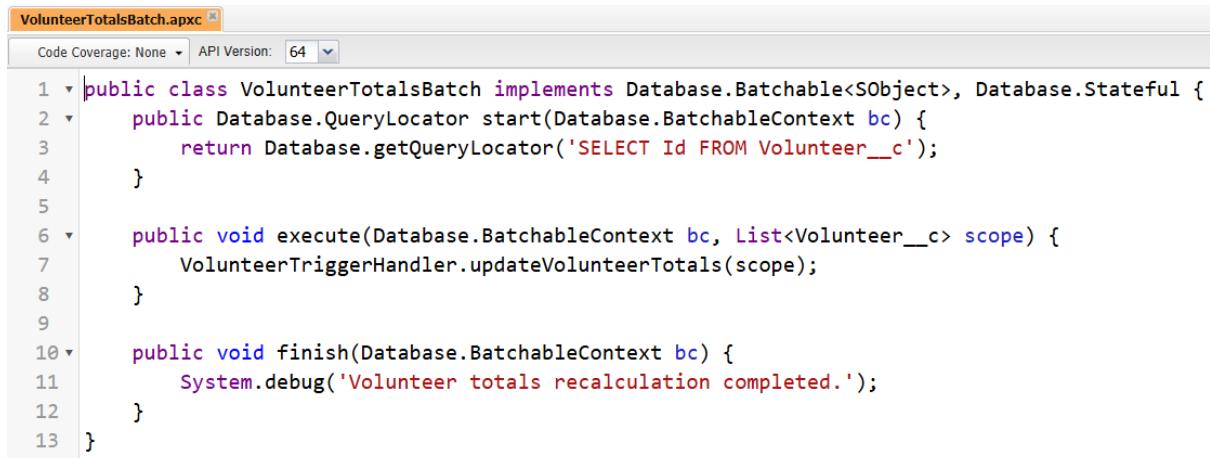
```
VolunteerTaskTrigger.apxt
Code Coverage: None API Version: 64
1 trigger VolunteerTaskTrigger on Volunteer_Task__c (after insert, after update, after delete) {
2     if(Trigger.isAfter){
3         VolunteerTaskTriggerHandler.updateVolunteerHours(
4             Trigger.isDelete ? Trigger.old : Trigger.new
5         );
6     }
7 }
```

10. VolunteerTaskTriggerTest (Class)



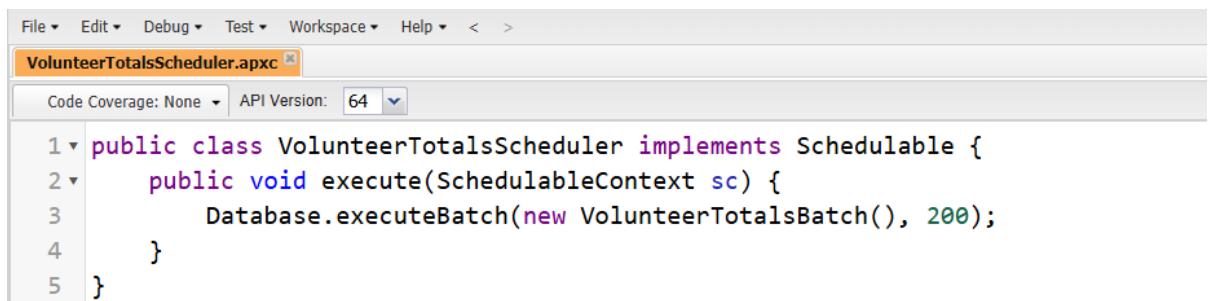
```
VolunteerTaskTriggerTest.apxt
Code Coverage: None API Version: 64
1 trigger VolunteerTaskTrigger on Volunteer_Task__c (after insert, after update, after delete) {
2     if(Trigger.isAfter){
3         VolunteerTaskTriggerHandler.updateVolunteerHours(
4             Trigger.isDelete ? Trigger.old : Trigger.new
5         );
6     }
7 }
```

11. VolunteerTotalBatch.apxc (Class)



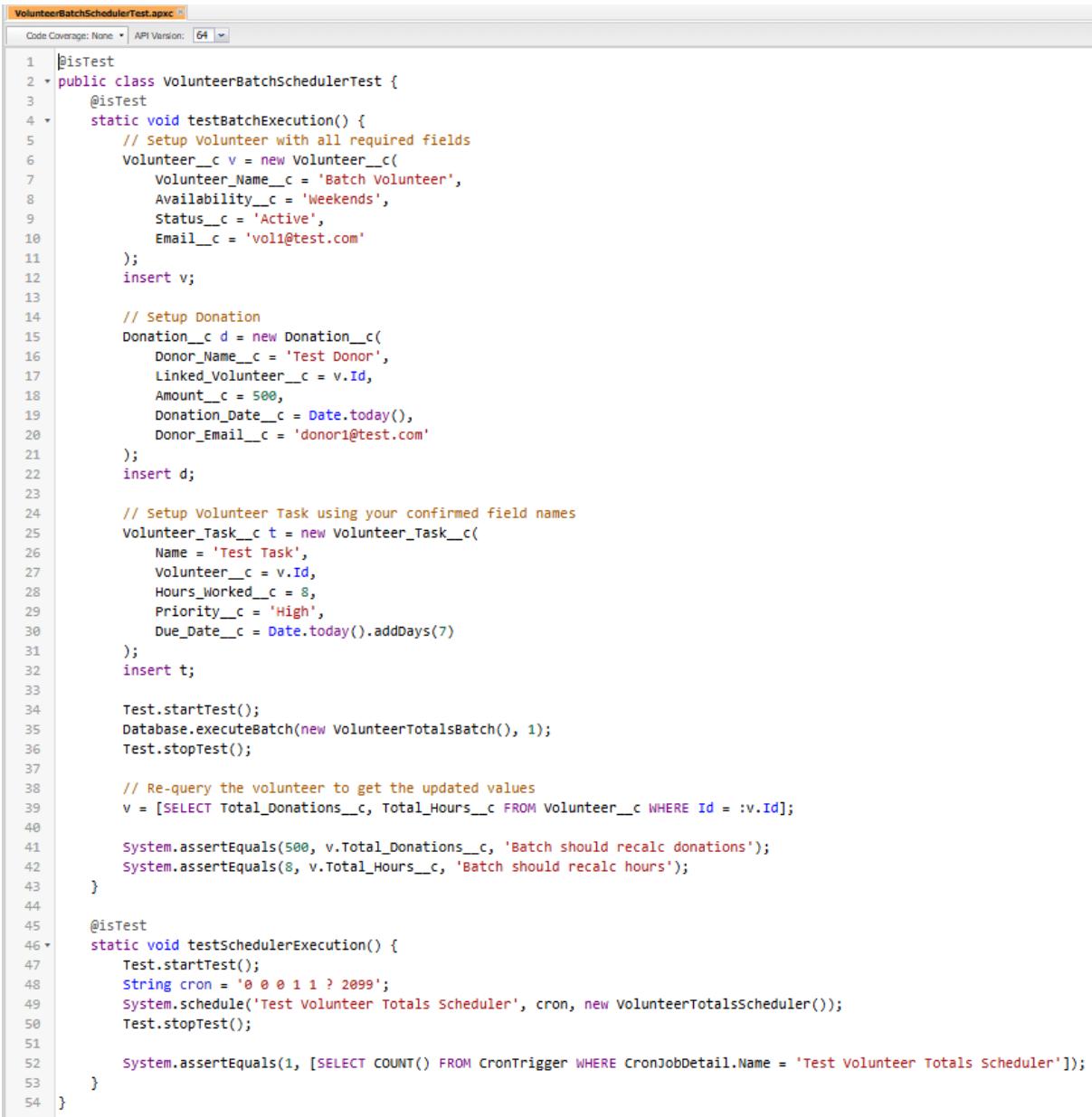
```
VolunteerTotalBatch.apxc
Code Coverage: None API Version: 64
1 public class VolunteerTotalBatch implements Database.Batchable<SObject>, Database.Stateful {
2     public Database.QueryLocator start(Database.BatchableContext bc) {
3         return Database.getQueryLocator('SELECT Id FROM Volunteer__c');
4     }
5
6     public void execute(Database.BatchableContext bc, List<Volunteer__c> scope) {
7         VolunteerTriggerHandler.updateVolunteerTotals(scope);
8     }
9
10    public void finish(Database.BatchableContext bc) {
11        System.debug('Volunteer totals recalculation completed.');
12    }
13 }
```

12. VolunteerTotalScheduler.cls (Class)



```
File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
VolunteerTotalScheduler.apxc
Code Coverage: None API Version: 64
1 public class VolunteerTotalScheduler implements Schedulable {
2     public void execute(SchedulableContext sc) {
3         Database.executeBatch(new VolunteerTotalBatch(), 200);
4     }
5 }
```

13. VolunteerTotalsScheduler.cls (Class)



```
1  @isTest
2  public class VolunteerBatchSchedulerTest {
3      @isTest
4      static void testBatchExecution() {
5          // Setup Volunteer with all required fields
6          Volunteer__c v = new Volunteer__c{
7              Volunteer_Name__c = 'Batch Volunteer',
8              Availability__c = 'Weekends',
9              Status__c = 'Active',
10             Email__c = 'voli@test.com'
11         );
12         insert v;
13
14         // Setup Donation
15         Donation__c d = new Donation__c{
16             Donor_Name__c = 'Test Donor',
17             Linked_Volunteer__c = v.Id,
18             Amount__c = 500,
19             Donation_Date__c = Date.today(),
20             Donor_Email__c = 'donori@test.com'
21         );
22         insert d;
23
24         // Setup Volunteer Task using your confirmed field names
25         Volunteer_Task__c t = new Volunteer_Task__c{
26             Name = 'Test Task',
27             Volunteer__c = v.Id,
28             Hours_Worked__c = 8,
29             Priority__c = 'High',
30             Due_Date__c = Date.today().addDays(7)
31         );
32         insert t;
33
34         Test.startTest();
35         Database.executeBatch(new VolunteerTotalsBatch(), 1);
36         Test.stopTest();
37
38         // Re-query the volunteer to get the updated values
39         v = [SELECT Total_Donations__c, Total_Hours__c FROM Volunteer__c WHERE Id = :v.Id];
40
41         System.assertEquals(500, v.Total_Donations__c, 'Batch should recalc donations');
42         System.assertEquals(8, v.Total_Hours__c, 'Batch should recalc hours');
43     }
44
45     @isTest
46     static void testSchedulerExecution() {
47         Test.startTest();
48         String cron = '0 0 0 1 1 ? 2099';
49         System.schedule('Test Volunteer Totals Scheduler', cron, new VolunteerTotalsscheduler());
50         Test.stopTest();
51
52         System.assertEquals(1, [SELECT COUNT() FROM CronTrigger WHERE CronJobDetail.Name = 'Test Volunteer Totals Scheduler']);
53     }
54 }
```

After saving all 13 files without any errors, you are ready to run the tests.

Part 2: Running Tests and Capturing Results

Now we will execute the test classes to verify all the logic works correctly and to check your code coverage.

Step 2.1: Navigate to the Test Execution Tab

1. In the Developer Console, click on the **Test** tab in the menu bar.
2. Select **New Run**.

Step 2.2: Run the Trigger Test

1. From the "Test Classes" list, select **VolunteerTriggerTest**.
2. Click the **Add Selected** button to move it to the right-hand column.
3. Click the **Run** button.

The test will execute, and the results will appear in the "Tests" tab.

The screenshot shows the Salesforce IDE interface. At the top, there's a header bar with tabs for 'Code Coverage: None' and 'API Version: 64'. Below the header is the code editor window titled 'VolunteerTriggerTest.apxc'. The code contains two test methods: `@isTest public class VolunteerTriggerTest { ... }` and `@isTest static void testVolunteerTotalsUpdate() { ... }`. The second method includes assertions for Total_Donations__c and Total_Hours__c fields. At the bottom of the interface is the 'Tests' tab, which displays a table with one row: 'Status: Test Run' and 'TestRun @ 5:13:46 pm'.

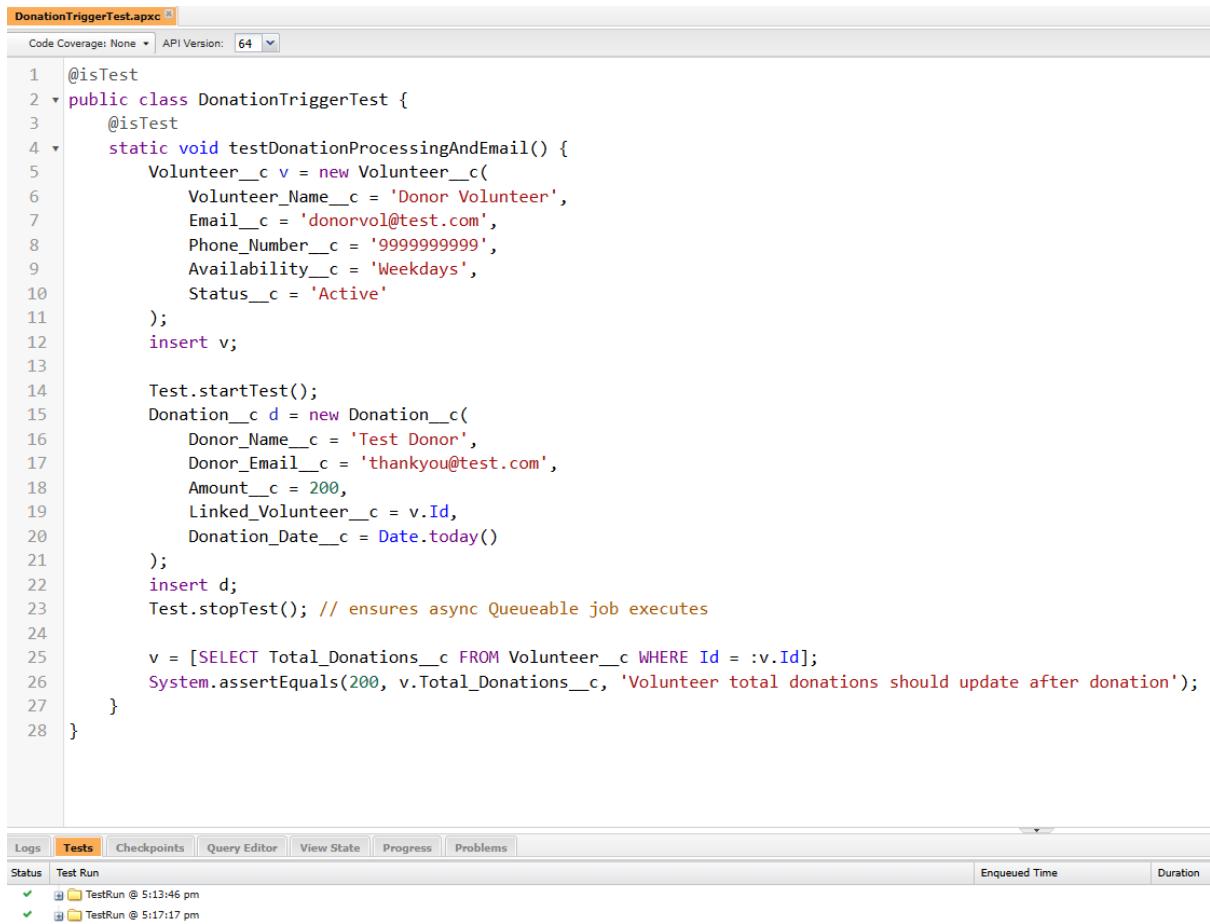
```
1  @isTest
2  public class VolunteerTriggerTest {
3      @testSetup
4      static void setupData() {
5          Volunteer__c v = new Volunteer__c(
6              Volunteer_Name__c = 'Test Volunteer',
7              Email__c = 'vol@test.com',
8              Phone_Number__c = '1234567890',
9              Availability__c = 'Weekends',
10             Status__c = 'Active'
11         );
12         insert v;
13
14         Donation__c d1 = new Donation__c(
15             Donor_Name__c = 'Donor One',
16             Donor_Email__c = 'donor1@test.com',
17             Amount__c = 100,
18             Linked_Volunteer__c = v.Id, // <-- ADD A COMMA HERE
19             Donation_Date__c = Date.today() // <-- ADD A COMMA HERE
20         );
21         insert d1;
22     }
23
24     @isTest
25     static void testVolunteerTotalsUpdate() {
26         Volunteer__c v = [SELECT Id, Total_Donations__c, Total_Hours__c FROM Volunteer__c LIMIT 1];
27
28         System.assertEquals(100, v.Total_Donations__c, 'Total Donations should be updated');
29         System.assertEquals(0, v.Total_Hours__c, 'Total Hours should remain 0 since no Hours__c field exists on Task');
30     }
31 }
```

Status	Test Run	Enqueued Time	Duration
✓	TestRun @ 5:13:46 pm		

Step 2.3: Run the Trigger Test

1. From the "Test Classes" list, select **DonationTriggerTest**.
2. Click the **Add Selected** button to move it to the right-hand column.
3. Click the **Run** button.

The test will execute, and the results will appear in the "Tests" tab.



```

1  @isTest
2  public class DonationTriggerTest {
3      @isTest
4      static void testDonationProcessingAndEmail() {
5          Volunteer__c v = new Volunteer__c(
6              Volunteer_Name__c = 'Donor Volunteer',
7              Email__c = 'donorvol@test.com',
8              Phone_Number__c = '9999999999',
9              Availability__c = 'Weekdays',
10             Status__c = 'Active'
11         );
12         insert v;
13
14         Test.startTest();
15         Donation__c d = new Donation__c(
16             Donor_Name__c = 'Test Donor',
17             Donor_Email__c = 'thankyou@test.com',
18             Amount__c = 200,
19             Linked_Volunteer__c = v.Id,
20             Donation_Date__c = Date.today()
21         );
22         insert d;
23         Test.stopTest(); // ensures async Queueable job executes
24
25         v = [SELECT Total_Donations__c FROM Volunteer__c WHERE Id = :v.Id];
26         System.assertEquals(200, v.Total_Donations__c, 'Volunteer total donations should update after donation');
27     }
28 }

```

The screenshot shows the Salesforce IDE interface with the 'Tests' tab selected. Below the code editor, there is a table titled 'Status' showing two test runs: 'Test Run' at 5:13:46 pm and another 'Test Run' at 5:17:17 pm, both marked as successful.

Step 2.3: Run the Trigger Test

1. From the "Test Classes" list, select **VolunteerTaskTriggerTest**.
2. Click the **Add Selected** button to move it to the right-hand column.
3. Click the **Run** button.

The test will execute, and the results will appear in the "Tests" tab.

The screenshot shows the Salesforce IDE interface. The top navigation bar includes File, Edit, Debug, Test, Workspace, Help, and a back/forward button. The code editor window has "VolunteerTaskTriggerTest.apxc" as the active file. The code itself is a test class for a trigger. It creates a volunteer record and a task record, then asserts that the volunteer's total hours remain at zero after the task is processed. The test passes with three successful runs listed in the test results table.

```

1  @isTest
2  public class VolunteerTaskTriggerTest {
3      @isTest
4      static void testTaskTriggerDoesNotBreak() {
5          Volunteer__c v = new Volunteer__c(
6              Volunteer_Name__c = 'Task Volunteer',
7              Email__c = 'taskvol@test.com',
8              Phone_Number__c = '8888888888',
9              Availability__c = 'Evenings',
10             Status__c = 'Active'
11         );
12         insert v;
13
14         Test.startTest();
15         Volunteer_Task__c t = new Volunteer_Task__c(
16             Name = 'Old Age Homes Food Supply',
17             Volunteer__c = v.Id,
18             Status__c = 'Not Started',
19             Priority__c = 'Medium',           // Provide a valid picklist value (e.g., 'Low', 'Medium', 'High')
20             Due_Date__c = Date.today().addDays(7) // Set a future due date
21         );
22         insert t;
23         update t;
24         delete t;
25         Test.stopTest();
26
27         v = [SELECT Total_Hours__c, Total_Donations__c FROM Volunteer__c WHERE Id = :v.Id];
28         System.assertEquals(0, v.Total_Hours__c, 'Volunteer hours should remain 0');
29     }
30 }

```

Logs	Tests	Checkpoints	Query Editor	View State	Progress	Problems
	Test Run					
	✓ TestRun @ 5:13:46 pm					
	✓ TestRun @ 5:17:17 pm					
	✓ TestRun @ 5:19:14 pm					
					Enqueued Time	Duration

Step 2.4: Run the Batch and Scheduler Test

1. Just like before, click **Test → New Run**.
2. This time, select **VolunteerBatchSchedulerTest** from the list.
3. Click **Add Selected**, then click **Run**.

```

1  @isTest
2  public class VolunteerBatchSchedulerTest {
3      @isTest
4      static void testBatchExecution() {
5          // Setup Volunteer with all required fields
6          Volunteer__c v = new Volunteer__c(
7              Volunteer_Name__c = 'Batch Volunteer',
8              Availability__c = 'Weekends',
9              Status__c = 'Active',
10             Email__c = 'vol1@test.com'
11         );
12         insert v;
13
14         // Setup Donation
15         Donation__c d = new Donation__c(
16             Donor_Name__c = 'Test Donor',
17             Linked_Volunteer__c = v.Id,
18             Amount__c = 500,
19             Donation_Date__c = Date.today(),
20             Donor_Email__c = 'donor@test.com'
21         );
22         insert d;
23
24         // Setup Volunteer Task using your confirmed field names
25         Volunteer_Task__c t = new Volunteer_Task__c(
26             Name = 'Test Task',
27             Volunteer__c = v.Id,
28             Hours_Worked__c = 8,
29             Priority__c = 'High',
30             Due_Date__c = Date.today().addDays(7)
31         );
32         insert t;
33
34         Test.startTest();
35         Database.executeBatch(new VolunteerTotalsBatch(), 1);
36         Test.stopTest();
37
38         // Re-query the volunteer to get the updated values
39         v = [SELECT Total_Donations__c, Total_Hours__c FROM Volunteer__c WHERE Id = :v.Id];
40
41         System.assertEquals(500, v.Total_Donations__c, 'Batch should recalc donations');
42         System.assertEquals(8, v.Total_Hours__c, 'Batch should recalc hours');
43     }
44
45     @isTest
46     static void testSchedulerExecution() {
47         Test.startTest();
48         String cron = '0 0 0 1 ? *';
49         System.schedule('Test Volunteer Totals Scheduler', cron, new VolunteerTotalsScheduler());
50         Test.stopTest();
51
52         System.assertEquals(1, [SELECT COUNT() FROM CronTrigger WHERE CronJobDetail.Name = 'Test Volunteer Totals Scheduler']);
53     }
54 }

```

The screenshot shows the "Tests" tab of the Developer Console. The status bar at the bottom indicates "Logs" (selected), "Tests" (highlighted in orange), "Checkpoints", "Query Editor", "View State", "Progress", and "Problems". The main area displays a list of test runs with their status (green checkmark) and run time (e.g., 5:13:46 pm, 5:17:17 pm, 5:19:14 pm, 5:24:44 pm).

Step 2.4: Check Overall Code Coverage

After all tests have passed, you can view your organization's overall Apex code coverage.

1. In the "Tests" tab of the Developer Console, look at the top-right corner. You will see "Overall Code Coverage".
2. Click the name of a class to see which lines were covered by your tests.

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage
						Class
						Percent
✓	[TestRun @ 5:13:46 pm]			0	1	97%
✓	[TestRun @ 5:17:17 pm]			0	1	100%
✓	[TestRun @ 5:19:14 pm]			0	1	100%
✓	[TestRun @ 5:24:44 pm]			0	1	100%
✓	70/gk000000E0SJ7	Thu Sep 25 2025 17:27:11 GM...		0	2	100%

Part 3: Scheduling the Nightly Job

The final step is to run the script that officially schedules your batch job to run every night.

Step 3.1: Run the Anonymous Apex Script

1. In the Developer Console, click **Debug** → **Open Execute Anonymous Window**.
2. Copy and paste the following script into the window:

Apex

```
VolunteerTotalsScheduler job = new VolunteerTotalsScheduler();  
  
String cron = '0 0 2 * * ?'; // Runs daily at 2 AM  
  
System.schedule('Nightly Volunteer Totals Job', cron, job);
```

3. Click the **Execute** button at the bottom right.

Step 3.2: Verify the Job is Scheduled

1. Go back to the main Salesforce window (not the Developer Console).
2. Click the **Gear Icon** and select **Setup**.
3. In the "Quick Find" box on the left, type Scheduled Jobs.
4. Click on the **Scheduled Jobs** link under the "Jobs" heading.

You will see a list of all scheduled jobs in your org.

The All Scheduled Jobs page lists all of the jobs scheduled by your users. Multiple job types may display on this page. You can delete scheduled jobs if you have the permission to do so.

Percentage of Scheduled Jobs Used: 1%
You have currently used 1 scheduled Apex jobs out of an allowed organization limit of 100 active or scheduled jobs. To learn about how this limit is calculated and what contributes to it see the [Lightning Platform Apex Limits](#) topic.

Action **Job Name** **Submitted By** **Submitted** **Started** **Next Scheduled Run** **Type** **Cron Trigger ID**

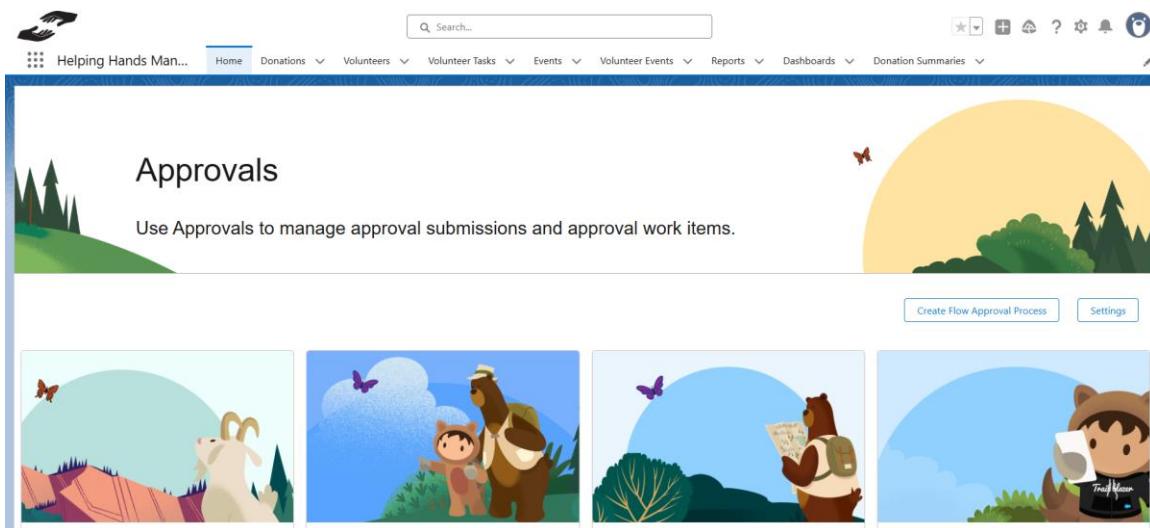
Action	Job Name	Submitted By	Submitted	Started	Next Scheduled Run	Type	Cron Trigger ID
Manage Del Pause Job	Nightly Volunteer Totals Job	Kovi Venkata Likith Sai	25/09/2025, 3:16 pm		26/09/2025, 2:00 am	Scheduled Apex	08egK00000C6BDD

Phase 6 - User Interface Development

In this phase, we will focus on building the User Interface (UI) within the Salesforce platform. The goal is to make it easy for users to view, create, and interact with the Donation Summary and Donor information without using external tools like VS Code. This phase ensures that everything is built directly within Salesforce using Lightning App Builder, Lightning Record Pages, and Custom Tabs.

Step 1: Open Lightning App Builder

1. Go to **Setup** (gear icon → Setup).
2. In the **Quick Find** box, type **Lightning App Builder**.
3. Click **New**.
4. Choose **App Page** → Next.
5. Name it: Helping Hands Management
6. Select **One Region Layout** (simple layout to start).
7. Click **Finish**.



Step 2: Add Custom Tabs for Objects

1. Go to **Setup** → **Tabs** (Quick Find → Tabs).
2. Under **Custom Object Tabs** → **New**.
3. Create Tabs for:
 - Donation
 - Volunteer
 - Event
 - Donation Summary

Select a Tab Style (e.g., “Money Bag” for Donation).

4. Save each.

Custom Object Tabs		New What Is This?
Action	Label	Tab Style
Edit Del	<u>Donations</u>	 Flag
Edit Del	<u>Donation Summaries</u>	 Laptop
Edit Del	<u>Events</u>	 Building Block
Edit Del	<u>Mentors</u>	 Hexagon
Edit Del	<u>Students</u>	 Desk
Edit Del	<u>Volunteer Events</u>	 Books
Edit Del	<u>Volunteers</u>	 Bell
Edit Del	<u>Volunteer Tasks</u>	 Chalkboard

Tabs Output:-



Step 3: Add Record Pages

1. Go to **Object Manager** → **Donation**.
2. Under **Lightning Record Pages** → **New**.
3. Select **Record Page** → Name: **Donation Record Page**.
4. Choose **Header and 2 Columns** layout.
5. Drag **Highlights Panel**, **Related Lists**, and **Details** into the layout.
6. Save & Activate → Assign as **Org Default**.

Donation Record Page

The screenshot displays the 'Donation Record Page' in the Salesforce interface. At the top, it shows the object name 'Donation' and the record ID 'DON-0018'. The page has a header with 'New Contact', 'Edit', and 'New Opportunity' buttons. On the left, there's a sidebar titled 'Approval History (0)'. The main content area contains the following fields:

Field	Value
Donation ID	DON-0018
Amount	₹12,000
Donation Date	24/09/2025
Donor Name	testc
Donor Email	testc@gmail.com
Payment Method	Cash
Status	Pledged
Linked Volunteer	A-017
Notes	(empty)
Linked Event	A-0001
Created By	Venkata Likith Sai Kovi, 24/09/2025, 3:04 pm
Last Modified By	Venkata Likith Sai Kovi, 24/09/2025, 3:04 pm

Volunteer Record Page

Volunteer A-017

Volunteer Tasks (1)

Volunteer Task Name	Status	Due Date	Priority
Orientation - aravind	Not Started	29/09/2025	High

[View All](#)

Donations (3)

Donation ID
DON-0011
DON-0014
DON-0018

[View All](#)

Volunteer Events (0)

Volunteer ID
A-017

Owner
[Venkata Likith Sai Kovi](#)

Volunteer Name
aravind

Email
aravind@gmail.com

Phone Number
6754475897

Skills
Fundraising

Availability
Evenings

Joined Date
15/09/2025

Status
Onboarding

Total Hours

Event
[A-0001](#)

Total Donations

Created By
[Venkata Likith Sai Kovi](#),
22/09/2025, 9:58 pm

Last Modified By
[Venkata Likith Sai Kovi](#),
22/09/2025, 9:58 pm

Event Record Page

Event A-0001

Volunteer Tasks (2)

Volunteer Task Name	Volunteer	Status	Due Date
Collect Food Packets	A-001	In Progress	30/09/2025
Orientation - aravind	A-017	Not Started	29/09/2025

[View All](#)

Volunteer Events (1)

Volunteer Event: Volunteer Event Name
VE-0001

[View All](#)

Donations (6+)

Donation ID
DON-0003
DON-0001
DON-0002
DON-0004
DON-0006
DON-0007

[View All](#)

Volunteers (1)

Volunteer ID
A-017

[View All](#)

Event ID
A-0001

Owner
[Venkata Likith Sai Kovi](#)

Event Name
Food Distribution Camp

Event Date
21/09/2025

Location
Gachibowli,Hyderabad

Description
It is a Food distribution camp,where the organic and protein foods are handedover to the poor people and daily wage workers.

Created By
[Venkata Likith Sai Kovi](#),
19/09/2025, 10:48 pm

Last Modified By
[Venkata Likith Sai Kovi](#),
19/09/2025, 10:48 pm

Step 4: Add Utility Bar (for Quick Access)

1. Go to **App Manager** → **Helping Hands App** → **Edit**.
2. Click **Utility Items** (left panel).
3. Add:
 - **Recent Items**
 - **Notes**
 - **Chatter Feed**

Set **Start Automatically** → **On**.

Utility Items (Desktop Only)

Give your users quick access to productivity tools and add background utility items to your app.

Add Utility Item Utility Bar Alignment ⓘ Default

Recent Items

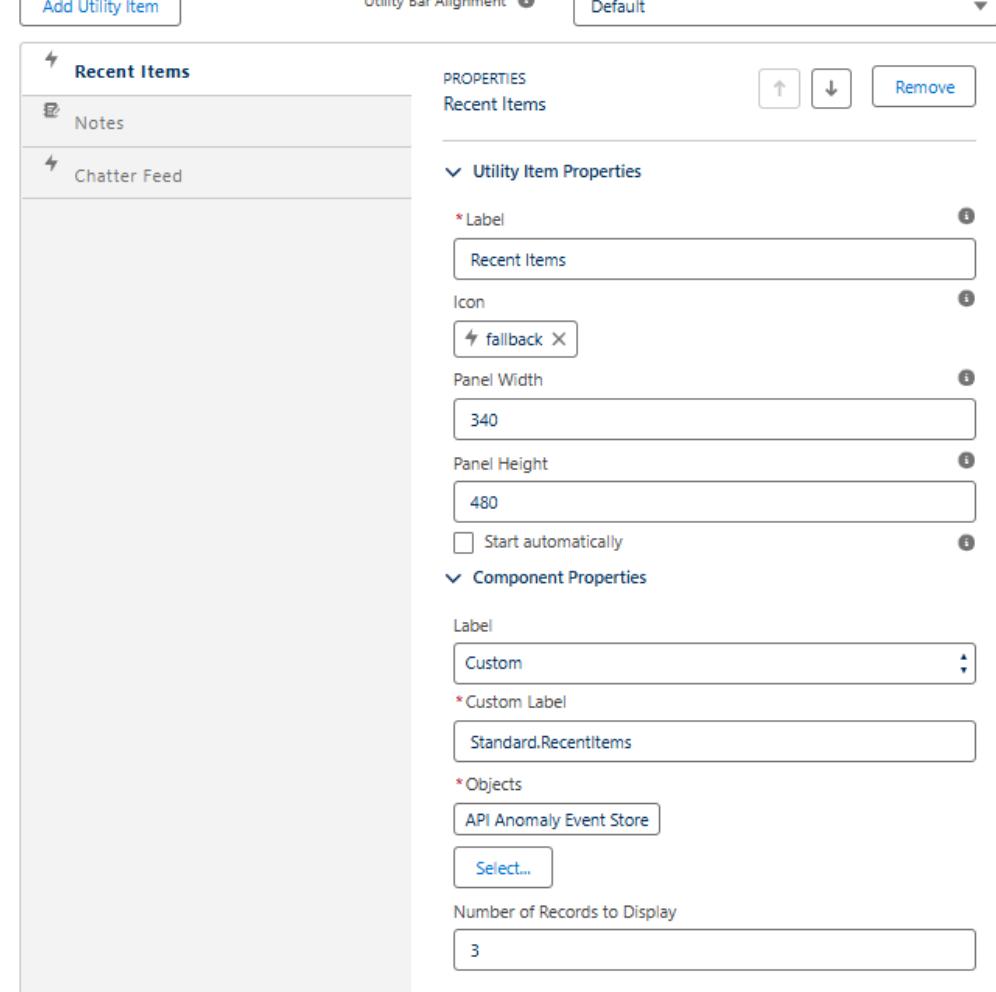
Notes	PROPERTIES	Recent Items	↑ ↓ Remove
-------	------------	--------------	------------

Utility Item Properties

* Label: Recent Items
Icon: fallback X
Panel Width: 340
Panel Height: 480
 Start automatically

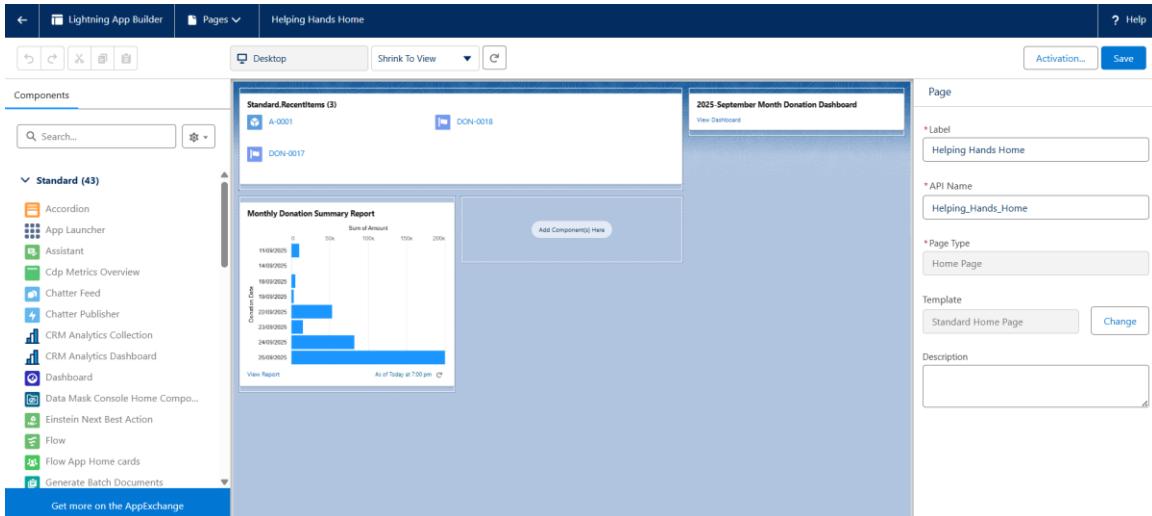
Component Properties

Label: Custom
* Custom Label: Standard.Recentitems
* Objects: API Anomaly Event Store
Select...
Number of Records to Display: 3



Step 5: Create a Custom Home Page

1. Go to **Setup → Lightning App Builder → New → Home Page.**
2. Name it: Helping Hands Home.
3. Choose **Standard Home Page**.
4. Drag components:
 - **Recent Items (Donations, Events)**
 - **Reports (Donation Summary Report)**
 - **Dashboard (Donation Trends)**
5. Save & Activate → Assign to Helping Hands App.



Step 6: Basic LWC (Lightning Web Component)

1. Open **Setup → Developer Console**.
2. Click **File → New → Lightning Component**.
3. Name: donationSummaryCard.
4. In the markup, paste simple code:

```

donationSummaryCardController.js [x] donationSummaryCard.cmp [x] DonationController.apxc [x]
1 <aura:component controller="DonationController" implements="flexipage:availableForAllPageTypes" access="global">
2   <!-- attributes to hold values -->
3   <aura:attribute name="totalAmount" type="String" />
4   <aura:attribute name="totalDonations" type="String" />
5   <aura:attribute name="monthLabel" type="String" />
6
7   <!-- init handler -->
8   <aura:handler name="init" value="{!this}" action=" {!c.doInit}"/>
9
10 <lightning:card title="Donation Summary">
11   <div class="slds-p-around_medium">
12     <p><strong>Month:</strong> {!v.monthLabel}</p>
13     <p><strong>Total Donation Amount:</strong> {!v.totalAmount}</p>
14     <p><strong>Total Donations:</strong> {!v.totalDonations}</p>
15   </div>
16 </lightning:card>
17 </aura:component>
-->

```

5. We will create an Apex class that returns the most recent `Donation_Summary__c` record.

1. Setup (⌚) → Quick Find → **Apex Classes** → click **Apex Classes**.
2. Click **New** (top-right).
3. In the editor, paste the exact code below.
4. Click **Save**.

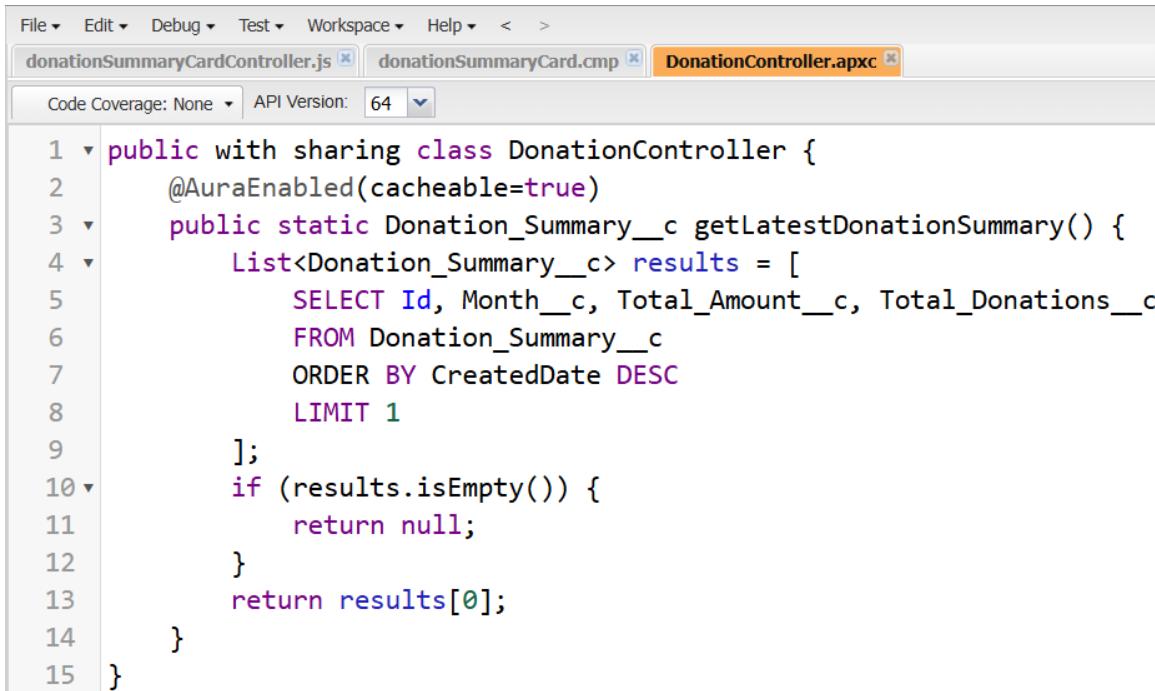
Apex class:

```

donationSummaryCardController.js [x] donationSummaryCard.cmp [x] DonationController.apxc [x]
1 <{(
2   doInit : function(component, event, helper) {
3     var action = component.get("c.getLatestDonationSummary");
4     action.setCallback(this, function(response) {
5       var state = response.getState();
6       if (state === "SUCCESS") {
7         var data = response.getReturnValue();
8         if (data) {
9           component.set("v.totalAmount", data.Total_Amount__c ? data.Total_Amount__c : '0');
10          component.set("v.totalDonations", data.Total_Donations__c ? data.Total_Donations__c : '0');
11          component.set("v.monthLabel", data.Month__c ? data.Month__c : 'N/A');
12        } else {
13          component.set("v.totalAmount", '0');
14          component.set("v.totalDonations", '0');
15          component.set("v.monthLabel", 'N/A');
16        }
17      } else {
18        // handle errors for debugging
19        var errors = response.getError();
20        console.error('Error from server-side action', errors);
21        component.set("v.totalAmount", 'Error');
22        component.set("v.totalDonations", 'Error');
23        component.set("v.monthLabel", 'Error');
24      }
25    });
26    $A.enqueueAction(action);
27  }
28 )>

```

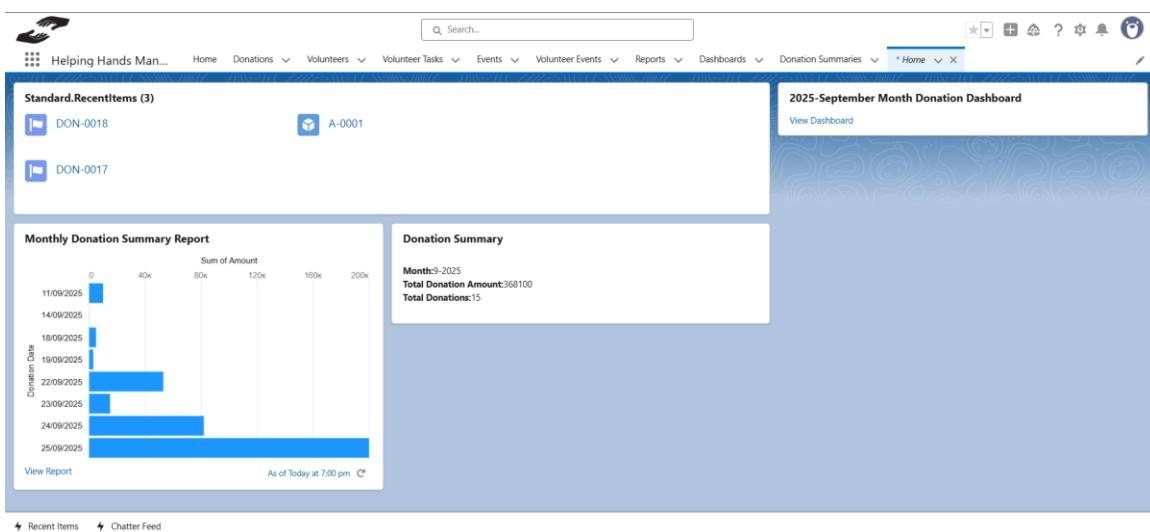
6. In Apex Controller:



```
1 public with sharing class DonationController {
2     @AuraEnabled(cacheable=true)
3     public static Donation_Summary__c getLatestDonationSummary() {
4         List<Donation_Summary__c> results = [
5             SELECT Id, Month__c, Total_Amount__c, Total_Donations__c
6             FROM Donation_Summary__c
7             ORDER BY CreatedDate DESC
8             LIMIT 1
9         ];
10        if (results.isEmpty()) {
11            return null;
12        }
13        return results[0];
14    }
15 }
```

Step 7: Place LWC on the Home Page

1. Go to **Lightning App Builder** → **Home Page** → **Edit**.
2. Drag **donationSummaryCard** LWC into the Home Page.
3. Save & Activate.



The screenshot shows the Salesforce Lightning App Builder interface with the following components visible:

- Header:** Shows the organization logo, "Helping Hands Man...", and various navigation links like Home, Donations, Volunteers, Events, Reports, Dashboards, and a search bar.
- Left Sidebar:** Displays "Standard.RecentItems (3)" with entries for "DON-001" and "DON-0017".
- Right Sidebar:** Shows a "2025-September Month Donation Dashboard" with a "View Dashboard" button.
- Content Area:** Contains:
 - A "Monthly Donation Summary Report" chart showing the sum of amounts for dates from 11/09/2025 to 25/09/2025. The chart has a Y-axis for "Donation Date" and an X-axis labeled "Sum of Amount" with ticks at 0, 40k, 80k, 120k, 160k, and 200k. The bars show increasing values over time, with a significant spike on 25/09/2025.
 - A "Donation Summary" card with the following details:
 - Month: 9-2025
 - Total Donation Amount: 368100
 - Total Donations: 15

Phase 7 - Integration & External Access

Contents

- A. Remote Site Settings
- B. Named Credentials (preferred) & Authentication
- C. Connected App & Auth Provider (OAuth flow)
- D. Apex Callouts (HTTP) + HttpCalloutMock test
- E. External Services (Register OpenAPI)
- F. Salesforce Connect (External Data Source & External Objects)
- G. Platform Events (publish & subscribe)
- H. Change Data Capture (enable & subscribe)
- I. API Limits & Monitoring

A. Remote Site Settings (Quick Allow List)

Purpose: If you will call external endpoints with simple HTTP requests (and are not using Named Credentials), you must register the external domain in Remote Site Settings.

1. Click the Gear icon → Setup.
2. In Quick Find box type 'Remote Site Settings' → Click 'Remote Site Settings'.
3. Click New Remote Site.
4. Fill the form exactly:
 - Remote Site Name: Example_API
 - Remote Site URL: <https://jsonplaceholder.typicode.com>
 - Active: checked
 - Description: Test public API for callouts
5. Click Save.

The screenshot shows the 'Remote Site Settings' page in Salesforce. At the top, there's a 'SETUP' button and a 'Remote Site Settings' link. Below that is a section titled 'Remote Site Details'. A table displays the 'Remote Site Detail' with the following data:

Remote Site Detail		Edit Delete Clone
Remote Site Name	Example_API	Modified By Venkata Likith Sai Kovil 25/09/2025, 8:32 pm
Remote Site URL	https://jsonplaceholder.typicode.com	
Disable Protocol Security	<input type="checkbox"/>	
Description	Test public API for callouts	
Active	<input checked="" type="checkbox"/>	
Created By	Venkata Likith Sai Kovil, 25/09/2025, 8:32 pm	

At the bottom of the table, there are three buttons: 'Edit', 'Delete', and 'Clone'.

B. Named Credentials (Preferred for Secure Callouts)

Purpose: Named Credentials centralize endpoint URL and authentication. Use them instead of Remote Site Settings whenever possible — they simplify authentication and secure credentials.

Exact steps to create a Named Credential (no auth example):

1. Setup → Quick Find → Named Credentials → Click Named Credentials.
2. Click New Named Credential.
3. Fill values:
 - Label: JSONPlaceholder API
 - Name: JSONPlaceholder_API
 - URL: <https://jsonplaceholder.typicode.com>
 - Identity Type: Named Principal
 - Authentication Protocol: No Authentication
 - Allow Merge Fields in HTTP Header: unchecked
4. Click Save.

The screenshot shows the Salesforce setup interface for creating a new named credential. The top navigation bar includes 'SETUP' and the 'Named Credentials' section. Below the header, a sub-header reads 'Named Credential: JSONPlaceholder API'. A note below the sub-header says 'Specify the callout endpoint's URL and the authentication settings that are required for Salesforce to make callouts to the remote system.' A link '« Back to Named Credentials' is present. The main form contains the following data:

Label	JSONPlaceholder API
Name	JSONPlaceholder_API
URL	https://jsonplaceholder.typicode.com

Below the form, two sections are expanded: 'Authentication' and 'Callout Options'.

Authentication:

- Certificate
- Identity Type: Named Principal
- Authentication Protocol: No Authentication

Callout Options:

- Generate Authorization Header: checked
- Allow Merge Fields in HTTP Header: unchecked
- Allow Merge Fields in HTTP Body: unchecked
- Outbound Network Connection: unchecked

C. Connected App & Auth Provider (for OAuth 2.0)

Purpose: Use a Connected App to allow external systems to authenticate with Salesforce, or to connect Salesforce to an external OAuth provider via Named Credential + Auth Provider.

[Create a Connected App \(for external OAuth client\):](#)

1. Setup → Quick Find → App Manager → Click New Connected App (top-right).

2. Fill the basics:

- Connected App Name: HelpingHands Integration App
- API Name: HelpingHands_Integration_App
- Contact Email: your-email@example.com

3. Under 'API (Enable OAuth Settings)'

check the box: 'Enable OAuth Settings'.

- Callback URL: <https://login.salesforce.com/services/oauth2/success>
- Selected OAuth Scopes: Add 'Full access (full)', 'Perform requests on your behalf at any time (refresh_token, offline_access)'

4. Save the Connected App. Wait ~2–10 minutes for the app to be active.

5. After saving, you will get Consumer Key (Client Id) and Consumer Secret (Client Secret) on the page — copy them for external use.

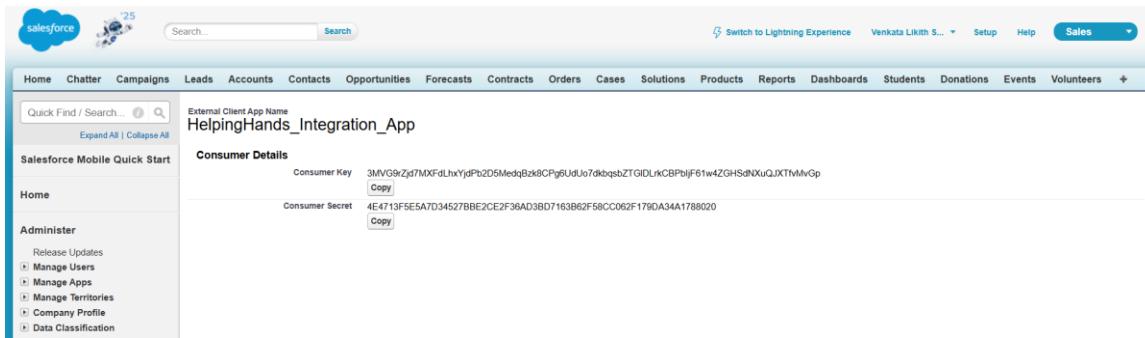
[Create an Auth. Provider \(if you want Salesforce to use an OAuth provider like Google or Custom OAuth\):](#)

1. Setup → Quick Find → Auth. Providers → New.

2. Provider Type: select the provider (e.g., Google) or choose 'OpenID Connect'/'OAuth 2.0' for custom providers.

3. Fill name and the client Id/secret if required.

4. Save.



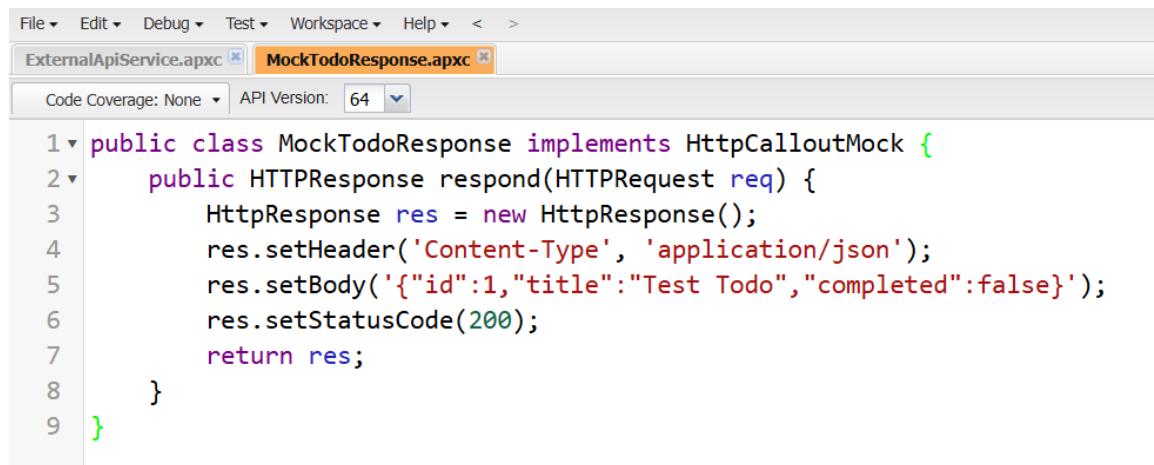
D. Apex HTTP Callouts and Testing (Developer Console)

Purpose: This section shows a complete pattern for doing a callout from Apex using a Named Credential and how to test it safely with HttpCalloutMock so tests run without issues.

Step D.1 — Create an Apex class that performs the callout:(MockTodoResponse.apxc)

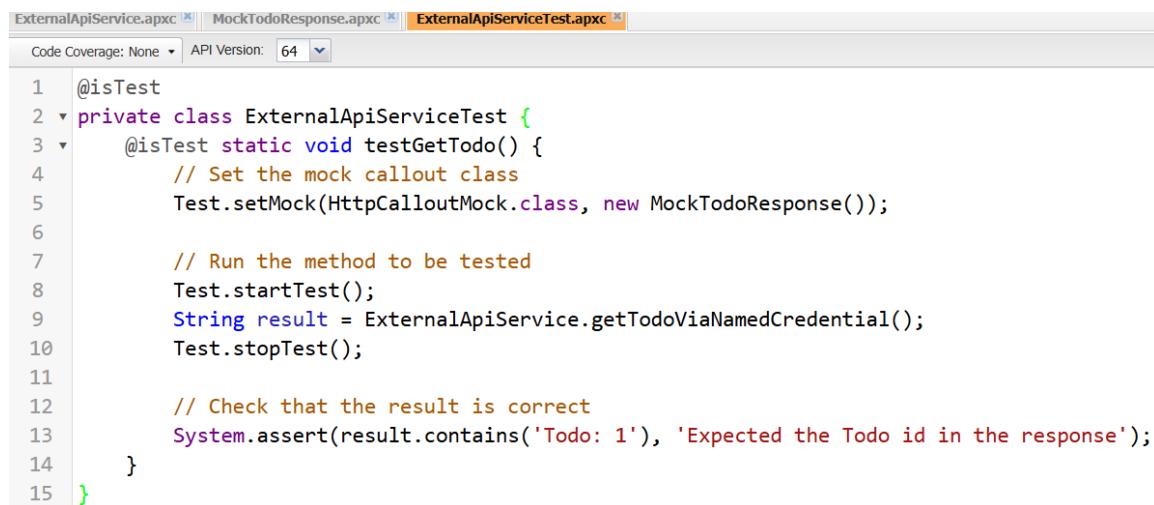
A screenshot of the Salesforce Developer Console code editor. The top navigation bar includes File, Edit, Debug, Test, Workspace, Help, and tabs for ExternalApiService.apxc and MockTodoResponse.apxc. The code editor shows the following Apex code:1 public with sharing class External ApiService {
2 public class Todo { public Integer id; public String title; public Boolean completed; }
3
4 @AuraEnabled
5 public static String getTodoViaNamedCredential() {
6 HttpRequest req = new HttpRequest();
7 req.setEndpoint('callout:JSONPlaceholder_API/todos/1');
8 req.setMethod('GET');
9 Http http = new Http();
10 HttpResponse res = http.send(req);
11 if (res.getStatusCode() == 200) {
12 Todo t = (Todo) JSON.deserialize(res.getBody(), Todo.class);
13 return 'Todo: ' + t.id + ' - ' + t.title;
14 }
15 return 'Error: ' + res.getStatus();
16 }
17}The bottom of the screen shows tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems, with 'Problems' being the active tab.

Step D.2 — Create an HttpCalloutMock for tests(MockTodoResponse.apxc)



```
File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
External ApiService.apxc MockTodoResponse.apxc
Code Coverage: None API Version: 64
1 public class MockTodoResponse implements HttpCalloutMock {
2     public HttpResponse respond(HTTPRequest req) {
3         HttpResponse res = new HttpResponse();
4         res.setHeader('Content-Type', 'application/json');
5         res.setBody('{"id":1,"title":"Test Todo","completed":false}');
6         res.setStatusCode(200);
7         return res;
8     }
9 }
```

Step D.3 — Create a Test Class that uses the mock and validates behavior:

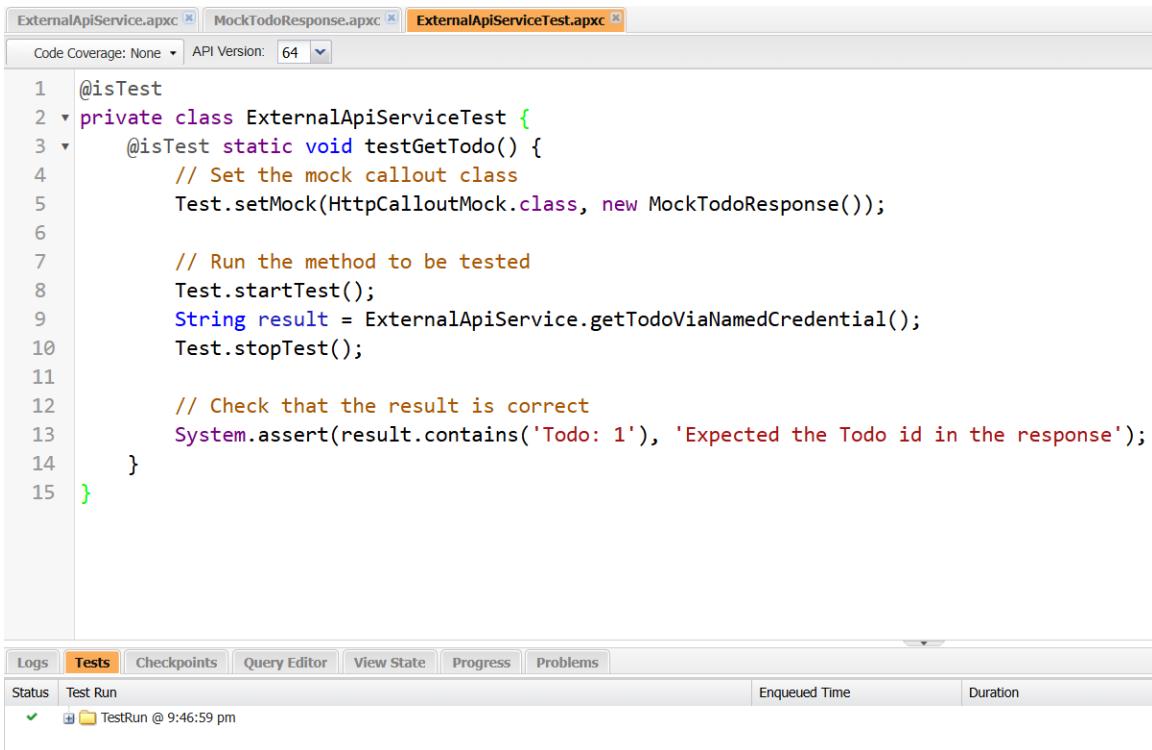


```
External ApiService.apxc MockTodoResponse.apxc External ApiServiceTest.apxc
Code Coverage: None API Version: 64
1 @isTest
2 private class External ApiServiceTest {
3     @isTest static void testGetTodo() {
4         // Set the mock callout class
5         Test.setMock(HttpCalloutMock.class, new MockTodoResponse());
6
7         // Run the method to be tested
8         Test.startTest();
9         String result = External ApiService.getTodoViaNamedCredential();
10        Test.stopTest();
11
12        // Check that the result is correct
13        System.assert(result.contains('Todo: 1'), 'Expected the Todo id in the response');
14    }
15 }
```

Step D.4.

open Developer Console → File → New → Apex Class (MockTodoResponse) → Save;
then create External ApiService class and save;

Then create External ApiServiceTest class and save. Run the Apex test via Setup → Apex Test Execution or Developer Console → Test → New Run.



```

1  @isTest
2  private class External ApiServiceTest {
3  @isTest static void testGetTodo() {
4      // Set the mock callout class
5      Test.setMock(HttpCalloutMock.class, new MockTodoResponse());
6
7      // Run the method to be tested
8      Test.startTest();
9      String result = External ApiService.getTodoViaNamedCredential();
10     Test.stopTest();
11
12     // Check that the result is correct
13     System.assert(result.contains('Todo: 1'), 'Expected the Todo id in the response');
14 }
15 }
```

The screenshot shows the Mule Studio interface with the 'Tests' tab selected. A single test run named 'TestRun @ 9:46:59 pm' is listed under the 'Status' column.

E. External Services (Register an OpenAPI / Swagger)

Purpose: External Services lets you import a REST API described by an OpenAPI (Swagger) schema and then use the generated actions in declarative tools (Flows).

Steps to Register the External Service

Your guide doesn't provide the required API specification, so I've created a simple one for you to use.

1. Go to **Setup**, use the Quick Find box to search for External Services, and open it.
2. Click the **New External Service** button.
3. On the first screen, select **From API Specification** and click **Next**.
4. Fill out the registration details:
 - **External Service Name:** JSONPlaceholder_Service

- **Named Credential:** Select the JSONPlaceholder_API you created earlier.
- For the **Service Schema**, choose the option Complete JSON. This will open a text box.

5. JSON

```
{  
  "openapi": "3.0.0",  
  
  "info": {  
    "title": "JSONPlaceholder Todos API",  
    "version": "1.0.0"  
  },  
  
  "paths": {  
    "/todos/{id)": {  
      "get": {  
        "summary": "Get a single todo item by ID",  
        "parameters": [  
          {  
            "name": "id",  
            "in": "path",  
            "required": true,  
            "schema": {  
              "type": "integer"  
            }  
          }  
        ]  
      },  
      "responses": {  
        "200": {  
          "content": {  
            "application/json": {  
              "schema": {  
                "type": "object",  
                "properties": {  
                  "id": {  
                    "type": "integer",  
                    "description": "The ID of the todo item."  
                  },  
                  "title": {  
                    "type": "string",  
                    "description": "The title of the todo item."  
                  },  
                  "body": {  
                    "type": "string",  
                    "description": "The body of the todo item."  
                  },  
                  "completed": {  
                    "type": "boolean",  
                    "description": "A flag indicating if the todo item is completed."  
                  }  
                }  
              }  
            }  
          }  
        }  
      }  
    }  
  }  
}
```


}

6. Save

The screenshot shows the 'External Services' setup page in Salesforce. At the top, there are five progress bars: 'Total Registrations' (1 used of 150), 'Active Operations' (1 used of 1,250), 'Total Operations' (1 used of 10,000), 'Active Objects' (1 used of 1,250), and 'Total Objects' (1 used of 10,000). Below the bars, a message says '1 Items - Sorted by External Service Name · Last refreshed a few seconds ago'. A table lists one item: 'JSONPlaceholderService' with 'D' status, 1 Active Operation, 1 Total Operation, 1 Active Object, 1 Total Object, 'JSONPlaceholder_API' Credentials, 'Complete' Configuration, and a dropdown Actions menu.

F. Salesforce Connect — External Data Source & External Objects

Purpose: Salesforce Connect lets you surface external data as external objects without copying data into Salesforce.

1. Setup → Quick Find → External Data Sources → New External Data Source.

2. Fill values:

- External Data Source Name: Northwind_OData
- Type: OData 4.0 (or 2.0 depending on the endpoint)
- URL: <https://services.odata.org/V4/Northwind/Northwind.svc/>
- Identity Type: Anonymous (or Named Principal with Named Credential if auth required)

3. Click Save and then click 'Validate and Sync'.

4. In the Sync UI, select the Entities you want to sync (Products, Orders, etc.) and click 'Sync'.

5. Salesforce will create External Objects (e.g., Product_x) that map to external entities.

The screenshot shows the 'External Data Sources' setup page in Salesforce. At the top, there's a 'SETUP' icon and the title 'External Data Sources'. Below the title, it says 'External Data Source: Northwind_OData'. A note below that says 'Connect to another Salesforce org or a third-party database or content system.' and a link to 'Back to External Data Sources'.

The main area contains a table with the following details:

External Data Source	Northwind_OData
Name	Northwind_OData
Type	Salesforce Connect: OData 4.0

Below this is a section titled 'Parameters' with various configuration options:

- URL: <https://services.odata.org/V4/Northwind/Northwind.svc/>
- Connection Timeout (Seconds): 120
- Writable External Objects:
- High Data Volume:
- Server Driven Pagination:
- Request Row Counts:
- Compress Requests:
- Enable Search:
- Use Free-Text Search Expressions:
- Format: JSON
- Special Compatibility: None
- Display Server Errors:
- Eligible for External Change Data Capture:

Below this is a section titled 'Authentication' with the following settings:

- Certificate:
- Identity type: Anonymous
- Authentication Protocol: No Authentication

At the bottom right of the main form are three buttons: 'Edit', 'Validate and Sync', and 'Delete'.

The final section at the bottom is titled 'External Objects' and lists several objects with their labels, namespace prefixes, descriptions, and table names:

Action	Label	Namespace Prefix	Description	Table Name
Edit Erase Validate	Alphabetical_list_of_product	Alphabetical_list_of_products	Alphabetical_list_of_products	Alphabetical_list_of_products
Edit Erase Validate	Category	Categories	Categories	Categories
Edit Erase Validate	Category_Sales_for_1997	Category_Sales_for_1997	Category_Sales_for_1997	Category_Sales_for_1997
Edit Erase Validate	Customer	Customers	Customers	Customers
Edit Erase Validate	Employee	Employees	Employees	Employees
Edit Erase Validate	Order_Detail	Order_Details	Order_Details	Order_Details
Edit Erase Validate	Order	Orders	Orders	Orders
Edit Erase Validate	Alphabetical_list_of_product	Alphabetical_list_of_products	Alphabetical_list_of_products	Alphabetical_list_of_products

Test Example:

Click the new External Object tab (create a Tab via Setup → Tabs → New → External Object Tab) or browse via App Launcher to view records.

You can also create external lookup relationships between standard/custom objects and external objects.

 **SETUP**
Tabs

New Custom Object Tab

Step 1. Enter the Details

Choose the custom object for this new custom tab. Fill in other details.

New Custom Object Tab

Select an existing custom object or [create a new custom object now](#).

Object	Alphabetical_list_of_product
Tab Style	

(Optional) Choose a Home Page Custom Link to show as a splash page the first time your users click on this tab.

Splash Page Custom Link	--None--
-------------------------	----------

Enter a short description.

Description	<input type="text"/>
-------------	----------------------

Custom Object Tabs

Action	Label	Tab Style
Edit Del	Alphabetical_list_of_products	Airplane
Edit Del	Donations	Flag
Edit Del	Donation Summaries	Laptop
Edit Del	Events	Building Block
Edit Del	Mentors	Hexagon
Edit Del	Students	Desk
Edit Del	Volunteer Events	Books
Edit Del	Volunteers	Bell
Edit Del	Volunteer Tasks	Chalkboard

G. Platform Events — Publish & Subscribe

Purpose: Platform Events deliver custom event messages within Salesforce (or to external systems). They are great for near real-time integration patterns.

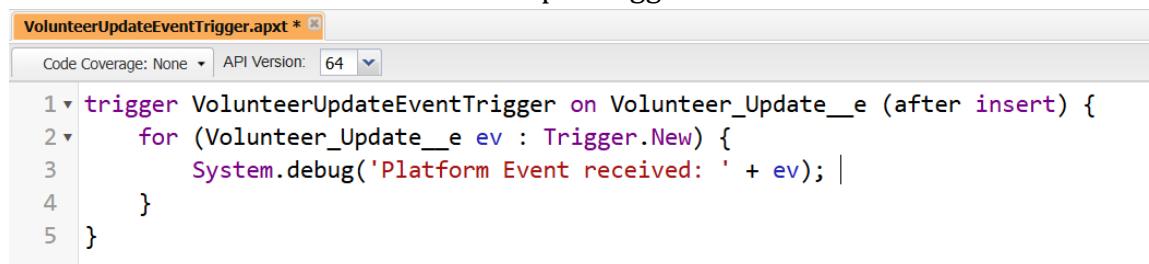
Create a Platform Event: exact clicks:

1. Setup → Quick Find → Platform Events → Click Platform Events.
2. Click 'New Platform Event'.
3. Fill:
 - Label: Volunteer_Update
 - Plural Label: Volunteer_Updates
 - Object Name: Volunteer_Update_e
 - Publish Behavior: Publish After Commit (recommended)
4. Click Save.

Add fields to the Platform Event: e.g., VolunteerId_c (Text), Update_Type_c (Text), Notes_c (Long Text).

Publish a Platform Event via Apex (Developer Console):

Subscribe to Platform Events with an Apex trigger:



```
VolunteerUpdateEventTrigger.apxt *
Code Coverage: None API Version: 64
1 trigger VolunteerUpdateEventTrigger on Volunteer_Update_e (after insert) {
2     for (Volunteer_Update_e ev : Trigger.New) {
3         System.debug('Platform Event received: ' + ev);
4     }
5 }
```

5. Publish an Event to Test It

Now I manually publish one event to see if your trigger is working.

1. In the **Developer Console**, click **Debug > Open Execute Anonymous Window**.
2. In the window that pops up, paste the following code. (The VolunteerId_c is just a placeholder, you don't need to change it).

Apex

```
Volunteer_Update_e ev = new Volunteer_Update_e(
```

```
    VolunteerId_c = 'a0B...',
```

```

Update_Type_c = 'Assigned',
Notes_c = 'Auto-assigned task'
);
Database.SaveResult sr = EventBus.publish(ev);

```

6. Check the **Open Log** box at the bottom and click **Execute**.

User	Application	Operation	Time	Status	Read	Size
Venkata Likith Sai Kovi	Unknown	/services/data/v64.0/to...	9/25/2025, 10:41:13 PM	Success	Unread	3.26 KB

H. Change Data Capture (CDC) — Enable & Subscribe

Purpose: CDC publishes change events for record lifecycle changes (create/update/delete/undelete).

Enable CDC for `Donation__c`:

1. Setup → Quick Find → Change Data Capture → Click 'Change Data Capture'.
2. From the list of objects, check 'Donation' (`Donation__c`) and click Save.
3. Now Salesforce will produce `Donation__ChangeEvent` events whenever `Donation__c` records change.

Subscribe in Apex by creating a trigger for the change event (Developer Console → New → Apex Trigger):

User	Application	Operation	Time	Status	Read	Size
Venkata Likith Sai Kovi	Unknown	/services/data/v64.0/to...	9/25/2025, 10:41:13 PM	Success	Unread	3.26 KB

Test: Update a Donation record. Then check Debug Logs or the trigger's System.debug output to confirm the change event fired.

The screenshot shows the Salesforce IDE interface. At the top, there is a header bar with the file name 'DonationChangeEventTrigger.apxt'. Below it, a toolbar has 'Code Coverage: None' and 'API Version: 64'. The main area contains the trigger code:

```

1 trigger DonationChangeEventTrigger on Donation__ChangeEvent (after insert) {
2     for (Donation__ChangeEvent evt : Trigger.new) {
3         System.debug('ChangeEvent header: ' + evt.ChangeEventHeader);
4     }
5 }

```

Below the code editor is a navigation bar with tabs: Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The 'Logs' tab is selected. A table below lists log entries:

User	Application	Operation	Time	Status	Read	Size
Venkata Likith Sai Kovi	Unknown	QueueableHandler	9/25/2025, 10:57:32 PM	Success	Unread	4.57 KB
Venkata Likith Sai Kovi	Browser	/aura	9/25/2025, 10:57:31 PM	Success	Unread	25.17 KB
Venkata Likith Sai Kovi	Unknown	common.api.soap.Direct...	9/25/2025, 10:57:31 PM	Success	Unread	516 bytes

I. API Limits & Monitoring

Purpose: Understand API usage and limits so you design efficient integrations.

How to check API usage in Salesforce (click-by-click):

1. Setup → Quick Find → System Overview → Click System Overview. Review 'API Requests, Last 24 Hours' and other stats.



2. To get programmatic view of limits, you can call the REST endpoint /services/data/vXX.0/limits (use Workbench or Postman with Salesforce authentication).

Quick steps to call the REST limits endpoint (Workbench method):

1. Go to <https://workbench.developerforce.com> → Login with your dev org credentials.

2. Navigate to Utilities → REST Explorer.
3. Use GET /services/data/v54.0/limits (replace v54.0 with your API version).
4. Review the JSON result which shows Concurrent Async, Daily API Calls, etc.

The screenshot shows the Salesforce REST Explorer interface. At the top, there's a navigation bar with links for workbench, info, queries, data, migration, and utilities. Below that, it says "VENKATA LIKITH SAI KOVI AT HELPING HANDS FOUNDATION ON API 62.0". A red banner at the top says "Try the Salesforce APIs for Postman.". Below that, it asks "Choose an HTTP method to perform on the REST API service URI below:" with options for GET (selected), POST, PUT, PATCH, DELETE, HEAD, Headers, Reset, and Up. The URL input field contains "/services/data/v62.0" and has an "Execute" button to its right. Below the URL field, there are links for "Expand All", "Collapse All", and "Show Raw Response". The main area lists numerous API endpoints starting with "/services/data/v62.0/". Some examples include: metadata, eclair, folders, jsonform, appMenu, iot, analytics, smartdatadiscovery, composite, parameterizedSearch, fingerprint, scheduling, domino, serviceTemplates, recent, dedupe, query, ai, consent, digitalwallet, compactLayouts, knowledgeManagement, actions, support, tooling, prechatForms, contact-tracing, chatter, payments, tabs, quickActions, queryAll, commerce, wave, search, identity, theme, nouns, event, connect, licensing, limits, process, jobs, match, localizedvalue, mobile, emailConnect, tokenizer, async, externalservices, and subjects. At the bottom, it says "Requested in 0.079 sec" and "Workbench 62.0.0".

Phase 8 - Testing & Deployment

1. Unit Testing with Apex

Step 1: Go to Setup → Apex Test Execution.

Step 2: Click 'Select Tests' → choose all test classes (DonationTriggerTest, VolunteerTriggerTest, etc.) and Phase-7 (HttpCalloutMock tests).

Step 3: Run them and wait for results.

The screenshot shows the 'Apex Test Execution' page. At the top, there are tabs for 'Select Tests...', 'Developer Console', 'Options...', and 'View Test History'. A 'Help for this Page' link is in the top right. Below the tabs, there's a section titled 'Test Run: 2025-09-25 23:24:55, sailikith511187@agentforce.com, (5 test classes run)'. This section lists five test classes with their results: DonationTriggerTest (1/1 Test Methods Passed), ExternalApiServiceTest (1/1 Test Methods Passed), VolunteerBatchSchedulerTest (2/2 Test Methods Passed), VolunteerTaskTriggerTest (1/1 Test Methods Passed), and VolunteerTriggerTest (1/1 Test Methods Passed). At the bottom, there are columns for 'Detail', 'Duration', 'Class', 'Method', 'Pass/Fail', 'Error Message', and 'Stack Trace'.

2. Debugging with Debug Logs

Step 1: In Setup → Debug Logs.

Step 2: Add yourself as a Monitored User.

Step 3: Run a test or perform actions (like creating a Donation record).

Step 4: Open the generated log → filter for USER_DEBUG to check values.

Debug Logs

Help for this Page 

A debug log records database operations, system processes, and errors that occur when executing a transaction or while running unit tests. The system generates a debug log for a user every time that user executes a transaction and the user has a trace flag with start and expiration dates that contain the transaction's start time. You can monitor and retain debug logs for the users specified below. One SFDC_DevConsole debug level is shared by all DEVELOPER_LOG trace flags in your org.

User Trace Flags		New				
Action	Name	Log Type	Requested By	Start Date	Expiration Date	Debug Level Name
Delete Edit Filters	Kovi_Venkata Likith Sai	USER_DEBUG	Venkata Likith Sai Kovi	25/09/2025, 11:34 pm	26/09/2025, 12:04 am	SFDC_DevConsole
Delete Edit Filters	Officer_Finance	DEVELOPER_LOG	Finance Officer	07/04/2022, 6:54 pm	07/04/2022, 6:59 pm	SFDC_DevConsole

Previous Page | Next Page

Debug Logs		Delete All					
User	Request Type	Application	Operation	Status	Duration (ms)	Log Size (bytes)	Start Time
View Download Delete Venkata Likith Sai Kovi	Api	Unknown	common.api.soap.DirectSoap	Success	68	516	09/25 23:38:51
View Download Delete Venkata Likith Sai Kovi	Application	Browser	/aura	Success	423	17,797	09/25 23:38:51

[Delete All](#)

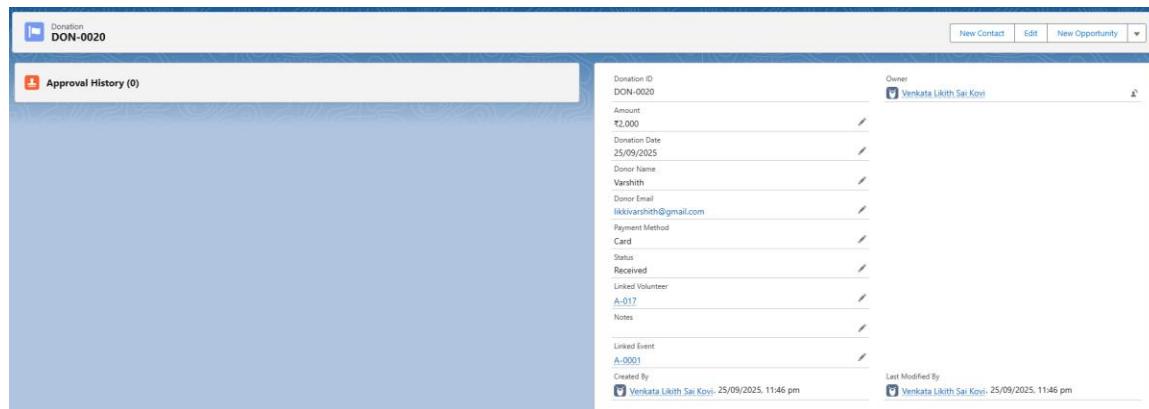
3. Validation of Automation

This process involves manually testing the key features you've built to ensure they work as a user would expect them to.

Test 1: Flow & Lightning Component

This test validates that creating a donation correctly updates the summary object and that the Lightning component shows the new data.

- Action:** Navigate to the Donations tab and create a new **Donation record**. Fill in all the necessary details, such as the amount and related account.
- What to Check:**
 - Go to the **Donation Summary** object and find the relevant summary record. Verify that its values have been correctly updated to include your new donation.



The screenshot shows a Salesforce page for a 'Donation' record with ID DON-0020. The page includes sections for 'Approval History (0)', 'Donation ID (DON-0020)', 'Owner (Venkata Likith Sai Kovi)', and various fields like 'Amount (\$2,000)', 'Donation Date (25/09/2025)', 'Donor Name (Varshith)', 'Donor Email (likkivarshith@gmail.com)', 'Payment Method (Card)', 'Status (Received)', 'Linked Volunteer (A-017)', 'Notes', 'Linked Event (A-0001)', and 'Created By (Venkata Likith Sai Kovi, 25/09/2025, 11:46 pm)'. A 'Last Modified By' field also shows Venkata Likith Sai Kovi.

- Open the page where the **Donation Summary Card** Lightning component is located and check that it displays the updated values.

Donation Summary record after your test donation.

Test 2: Approval Process

This test ensures that the approval workflow functions correctly.

- Action:** Find a Donation record that meets the criteria for your approval process. Submit it for approval. Then, as the designated approver, find the approval request and **approve the Donation**.

Step Name	Date	Status	Assigned To	Actual Approver	Comments
1 High Value Donation Approval Step	26/09/2025, 1:21 am	Pending	Finance Officer	Finance Officer	
2 Approval Request Submitted	26/09/2025, 1:21 am	Submitted	Finance Officer	Finance Officer	submit

- What to Check:** After approving, view the Donation record and confirm that its status or relevant fields have been updated to reflect the approval.

Submitter	Date Submitted	Actual Approver	Assigned To
Finance Officer	26-Sept-2025	Finance Officer	Finance Officer

Result at User's Side:-

The screenshot shows a Salesforce page for a donation record. At the top left is the icon and ID 'Donation DON-0024'. At the top right are buttons for 'New Contact', 'New Opportunity', and 'New Case'. The main content area has two sections: 'Approval History (2)' on the left and detailed record information on the right.

Step Name	Date	Status	Assigned To
High Value Donation A...	26/09/2025, 1:42 am	Approved	Finance Officer
Approval Request Sub...	26/09/2025, 1:21 am	Submitted	Finance Officer

[View All](#)

Donation ID: DON-0024
Owner: Finance Officer

Amount: ₹1,23,000
Donation Date: 23/09/2025
Donor Name: swetha
Donor Email: swetha@gmail.com
Payment Method: Cash
Status: Approve
Linked Volunteer: A-019
Notes:
Linked Event:

Recent Items Chatter Feed

Test 3: Trigger Logic

This test verifies that your trigger's cleanup logic is working when a record is deleted.

- Action:** First, create a sample Volunteer record and ensure it has related records (like Volunteer Tasks). Then, go to that Volunteer record and **delete it**.

The screenshot shows a Salesforce page for a volunteer record. At the top left is the icon and ID 'Volunteer A-020'. At the top right are buttons for 'New Contact', 'Edit', and 'New Opportunity'. The main content area has three sections: 'Volunteer Tasks (1)', 'Donations (0)', and 'Volunteer Events (0)'. On the right is a detailed view of the volunteer record.

Volunteer Task Name	Status	Due Date	Priority
Money Collection	In Progress	30/9/2025	High

[View All](#)

Donation ID	Owner
A-020	Venkata Likith Sai Kovi

Volunteer Name: volunteer test
Email: volunteer@gmail.com
Phone Number:
Skills: Fundraising
Availability: Weekends
Joined Date: 25/09/2025
Status: Active
Total Hours: 4.0
Event:

- What to Check:** After deletion, confirm that the trigger's cleanup logic has run successfully. For example, check to see if the related Volunteer Task records were also deleted or reassigned as expected

The screenshot shows the 'Volunteer Tasks' list view. At the top, there's a header with 'Volunteer Tasks' and a 'Recently Viewed' dropdown. Below the header, it says '1 item • Updated a few seconds ago'. A single task is listed: 'Collect Food Packets' (Type: Volunteer). The task has a checkbox next to its name. At the bottom right of the list area, there are buttons for 'New', 'Import', 'Change Owner', and 'Assign Label'. To the right of the list, there's a search bar with placeholder text 'Search this list...' and various filter and sort icons.

The volunteer task has also been deleted after the deletion of volunteer A-020.

Recycle Bin Check:-

The screenshot shows the 'Recycle Bin' list view. At the top, it says 'Recycle Bin' and 'My Recycle Bin'. It indicates '1 item • Sorted by Deleted Date • Filtered by My recycle bin • Updated a few seconds ago'. A single item is listed: 'A-020' (Record Name, Type: Volunteer, Deleted By: Venkata Likith Sai Kovi, Deleted Date: 26/09/2025, 2:37 am). At the top right, there are buttons for 'Restore', 'Delete', and 'Empty Org Recycle Bin'. Below the list are standard filter and sort icons.

4. Code Coverage Requirement

Salesforce requires $\geq 75\%$ code coverage for deployment.

Step 1: Go to Setup → Apex Classes → Code Coverage.

Step 2: Ensure each trigger/class has coverage.

The screenshot shows the 'Apex Classes' page. At the top, it says 'Apex Classes' and 'Help for this Page'. It provides information about Apex Code usage: 'Percent of Apex Used: 0.13%', 'You are currently using 7,577 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization.', 'Code Coverage: 84%', 'Your overall code coverage is currently 84%. To deploy code to production, you must have at least 75%.', and 'Compilation Complete' with 'Compilation Successful'. Below this, there are buttons for 'View: All' and 'Create New View'. The main area is a table listing Apex classes. The columns are: Action, Name +, Namespace Prefix, Api Version, Status, Size Without Comments, Last Modified By, and Has Trace Flags. The table lists four classes: 'DonationController', 'DonationThankYouEmailQueue', 'DonationTriggerHandler', and 'DonationTriggerTest'. Each class has an 'Edit | Del | Security' link, its namespace prefix, API version (64.0), status (Active), size (e.g., 471, 885, 702, 917), last modified by (Venkata Likith Sai Kovi), and a 'Has Trace Flags' checkbox.

5. Post-Deployment Steps

Step 1: Assign Permissions:

- Create a HelpingHands User Permission Set.
- Add CRUD access to all HelpingHands objects.

Step 2: Test with a normal user:

- Log in as them → check if they can Create/Update Donation.

Permission Set Information										
See the permissions enabled for this permission set and the permission set groups it's added to.										
Related Permission Set Groups		User Permissions		Object Permissions		Field Permissions		Custom Permissions		Tabs
6 items										Edit <input type="text"/> Search this list...
Label	Object API N...	Read	Create	Edit	Delete	View All Rec...	Modify All R...	View All Fields		
Donation	Donation__c	✓	✓	✓	✓	✗	✗	✗		
Donation Summary	Donation_Summar...	✓	✓	✓	✓	✗	✗	✗		
Event	Event__c	✓	✓	✓	✓	✗	✗	✗		
Volunteer	Volunteer__c	✓	✓	✓	✓	✗	✗	✗		
Volunteer Event	VolunteerEvent__c	✓	✓	✓	✓	✗	✗	✗		
Volunteer Task	Volunteer_Task__c	✓	✓	✓	✓	✗	✗	✗		

6. Data Import Wizard

The Data Import Wizard is a tool built directly into Salesforce Setup. It's designed for simple data imports of up to 50,000 records. Think of it as the perfect tool for a non-technical user, like a manager, who needs to upload a list of new leads or contacts from a spreadsheet.

Importing New Donations

Your task is to import a small list of new Donation records using the Data Import Wizard.

Step 1: Create Your CSV File

First, you need a data file to import. Open a plain text editor (like Notepad orTextEdit) and paste the following text. Save the file on your computer with the name donations.csv.

Code snippet

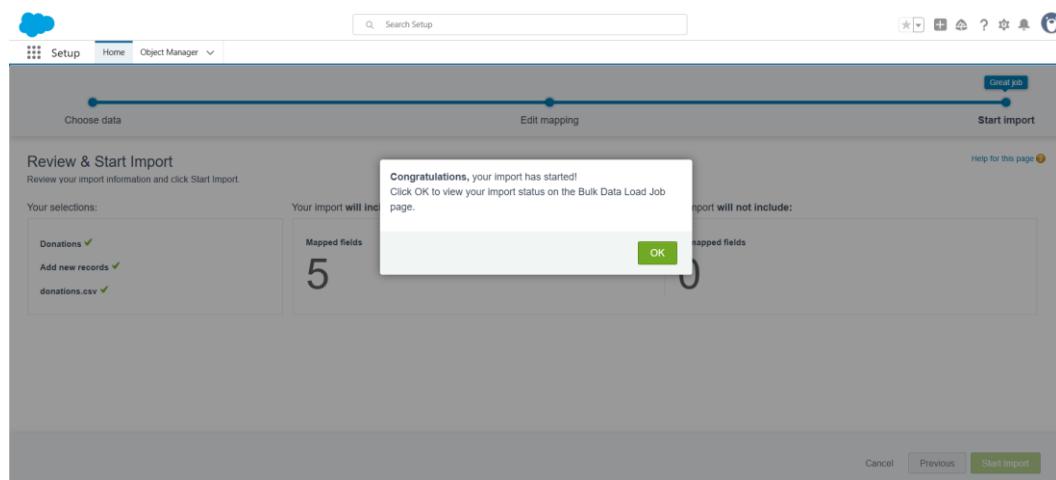
```
Donor_Name__c,Donor_Email__c,Amount__c,Payment_Method__c,Status__c  
Alice Johnson,alice.j@example.com,250,Card,Received  
Bob Williams,bob.w@example.com,500,Cash,Received  
Charlie Brown,charlie.b@example.com,75,Online,Pledged
```

Note: The column headers in the CSV must exactly match the API names of your fields on the Donation object.

Step 2: Use the Data Import Wizard

1. Go to Setup and type Data Import Wizard in the Quick Find box.
2. Click Launch Wizard!
3. On the first screen, choose the Donations custom object.
4. Choose to Add new records. You can leave the other options as they are.
5. Upload your donations.csv file.
6. On the next screen, Salesforce will ask you to map your columns. It should automatically match them, but ensure that Donor_Name__c from your CSV maps to the Donor Name field in Salesforce, and so on.
7. Click Next, then Start Import.

You'll receive an email when the import is complete. Go to your Donations tab to see the three new records.

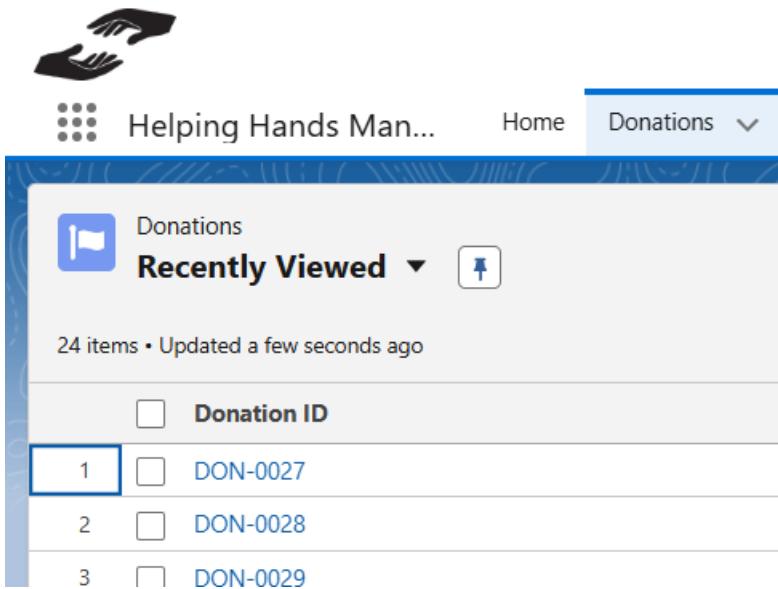


Bulk Data Load Job Detail		Reload
Job ID	750gK00000DqAS1	Job Type Bulk V1
Submitted By	Venkata Likhith Sai Kovil	Operation Insert
Start Time	26/09/2025, 9:58 am IST	Queued Batches 0
End Time	26/09/2025, 9:58 am IST	In Progress Batches 0
Time to Complete (hh:mm:ss)	00:01	Completed Batches 1
Object	Donation	Failed Batches 0
External ID Field		Progress 100%
Content Type	CSV	Records Processed 3
Concurrency Mode	Parallel	Records Failed 0
API Version	64.0	Retries 0

Reload

Batches												
View Request	View Result	Batch ID	Start Time	End Time	Processing Time (ms)	API Processing Time (ms)	Apex Processing Time (ms)	Records Processed	Records Failed	Retry Count	State Message	Status
View Request	View Result	751gK00000BCxag	26/09/2025, 9:58 am	26/09/2025, 9:58 am	245	163	79	3	0	0	Completed	

Results



The screenshot shows the 'Helping Hands Management System' interface. At the top, there's a logo of two hands, a navigation bar with 'Home' and 'Donations' (which is currently selected), and a search bar. Below the header, the 'Donations' section has a title 'Recently Viewed' with a dropdown arrow. It displays a list of 24 items, with the first item highlighted. The list includes columns for a checkbox, the donation ID, and the record number.

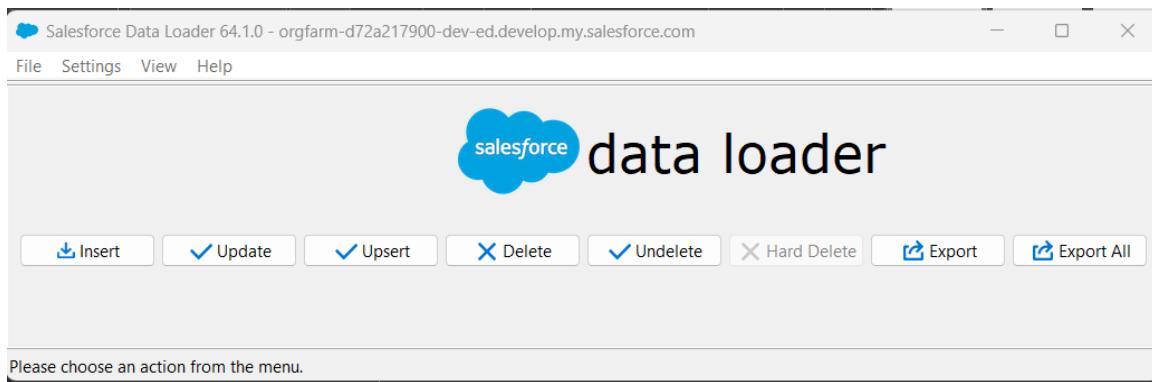
1	DON-0027
2	DON-0028
3	DON-0029

7. Data Loader

Step 1: Install Data Loader

Data Loader is a program you need to download and install on your computer.

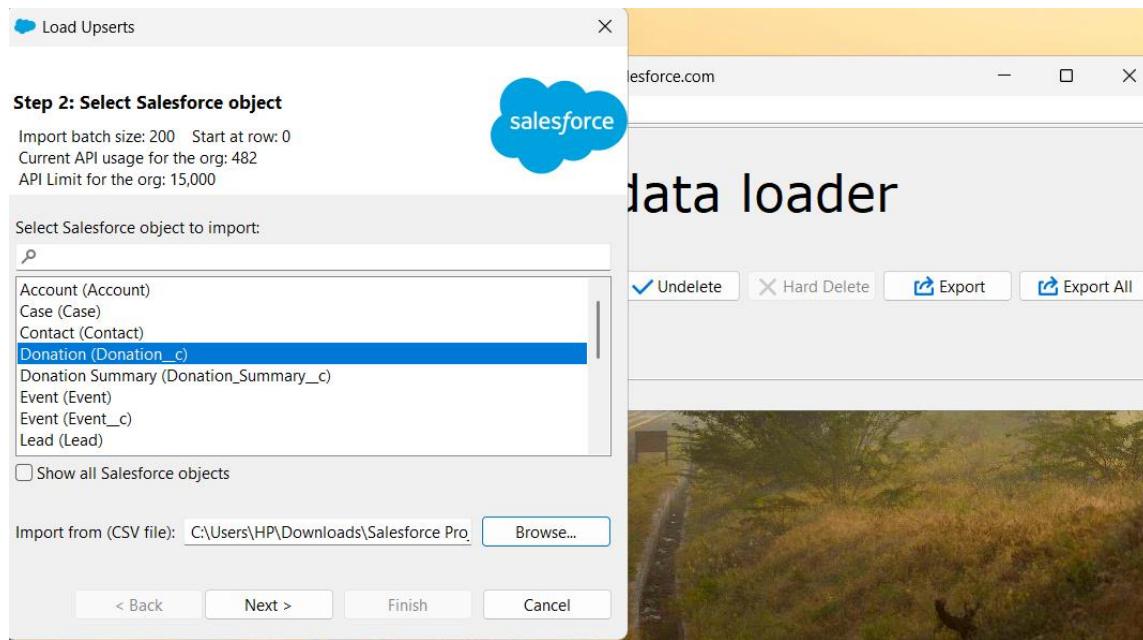
1. In Salesforce Setup, use the Quick Find box to search for and open Data Loader.
2. On this page, you'll find links to download the installer for Windows or macOS.
3. Download the correct version for your computer and follow the on-screen instructions to install it. You may also need to install a Java runtime if prompted.



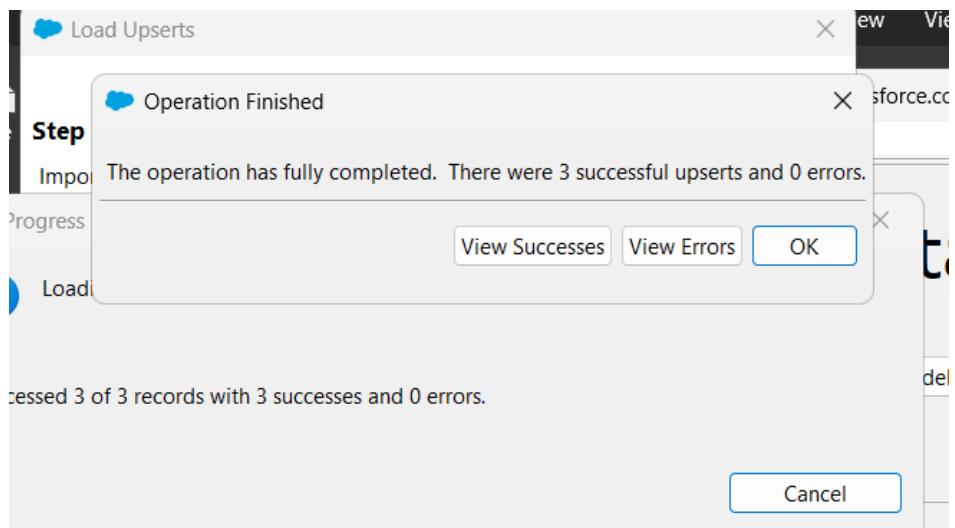
Step 2: Perform the Upsert

Once Data Loader is installed, you can perform the upsert operation.

1. Open Data Loader and click Upsert.
2. Log in:
 - o Environment: Production (Developer orgs use this)
 - o Click Log in and use the username and password for your project org.
3. Choose SObject and CSV:
 - o Select the Donation (Donation__c) object.
 - o Browse for and select your donations_upsert.csv file.



4. Map External ID: For an upsert, Data Loader needs a way to match existing records. Since you don't have an External ID field, you'll use the Salesforce ID itself. Select the Id field from the list.
5. Map Fields: On the mapping screen, click Create or Edit a Map. Drag and drop the field names from the top box to the bottom box to match your CSV columns to the Salesforce fields.
6. Run the Operation: Click Next, choose a directory to save the success and error files, and click Finish.



ID	Donor_Name__c	Donor_Email__c	Amount__c	Payment_Method__c	Status__c	STATUS
a02gK000005pyQfQAI	Alice Johnson	alice.j@example.com	250	Card	Received	Item Created
a02gK000005pyQgQAI	Bob Williams	bob.w@example.com	500	Cash	Received	Item Created
a02gK000005pyQhQAI	Charlie Brown	charlie.b@example.com	75	Online	Pledged	Item Created

8. Duplicate Rules

After importing a lot of data, one of the biggest challenges is preventing duplicate records. Duplicate Rules are Salesforce's built-in tool to automatically catch and block duplicates as they are being created.

This feature works in two parts:

1. **Matching Rule:** Defines *what counts* as a duplicate (e.g., two records with the same email).
2. **Duplicate Rule:** Defines *what happens* when a duplicate is found (e.g., block the user or just show a warning).

Prevent Duplicate Donations

Our goal is to create a rule that blocks users from creating a new Donation if another Donation with the same **Donor Email** and **Donation Date** already exists.

Step 1: Create the Matching Rule

1. Go to **Setup** and search for **Matching Rules**.
2. Click **New Rule** and select the **Donation** object.
3. Name the rule **Donation_Matching_Rule**.
4. For the criteria, select the **Donor Email** and **Donation Date** fields. Set the Matching Method for both to **Exact**.
5. **Save** and **Activate** the rule.

The screenshot shows the 'Matching Rules' page in Salesforce. The top navigation bar has 'SETUP' selected. Below it, the page title is 'Matching Rules'. A sub-header 'Matching Rule Detail' is visible. The main section displays the details of a new rule:
Object: Donation
Rule Name: Donation_Matching_Rule
Unique Name: Donation_Matching_Rule
Description: (Donation: Donor_Name EXACT MatchBlank = FALSE) AND (Donation: Donor_Email EXACT MatchBlank = FALSE)
Matching Criteria: (Donation: Donor_Name EXACT MatchBlank = FALSE) AND (Donation: Donor_Email EXACT MatchBlank = FALSE)
Status: Activating
Created By: Venkata Likith Sai Kovi, 26/09/2025, 10:40 am
Modified By: Venkata Likith Sai Kovi, 26/09/2025, 10:40 am

Step 2: Create the Duplicate Rule

1. Go to **Setup** and search for **Duplicate Rules**.
2. Click **New Rule** and select the **Donation** object.
3. Name the rule **Prevent_Duplicate_Donations**.
4. Under "Matching Rule," select the **Donation_Matching_Rule** you just created.
5. Under "Actions," for the "Action on Create" option, select **Block**.
6. **Save** and **Activate** the rule.

The screenshot shows the 'Duplicate Rule Detail' page for a rule named 'Prevent_Duplicate_Donations'. The page includes fields for Rule Name, Description, Object, Record-Level Security, Action On Create, Action On Edit, Alert Text, Active status, Matching Rule (selected 'Donation_Matching_Rule'), Conditions, and Created By. It also shows Operations On Create and Operations On Edit settings, Matching Criteria (using AND logic for Donor_Name and Donor_Email), and Modification history. Buttons for Edit, Delete, Clone, and Deactivate are at the top and bottom of the page.

9. Data Export and Backup

Just as it's important to get data into Salesforce, you also need ways to get it out for reporting, archiving, or backup purposes. Salesforce provides two primary methods for this.

Method 1: Data Export Service (Weekly/Monthly Backup)

This is a built-in service for creating a complete backup of your org's data. You can schedule it to run automatically every week or month.

Practice:

1. Go to **Setup** and search for **Data Export**.

2. Click **Export Now**. (If that's not available, click **Schedule Export**).
3. You can leave the default settings to "Include all data".
4. Click **Start Export**.

The screenshot shows the 'Data Export' page in the Salesforce setup. At the top, there's a message about the monthly export service. Below it, a yellow bar indicates a 'Next scheduled export' on 01/10/2025 at 11:00 am. There are buttons for 'Export Now' and 'Schedule Export'. Underneath, details of the scheduled export are listed: 'Scheduled By' Venkata Utkarsh Sai Kovi, 'Schedule Date' 26/09/2025, and 'Export File Encoding' ISO-8859-1 (General US & Western European, ISO-LATIN-1). A table at the bottom shows the export file information: Action (download), File Name (WE_00Dgk000007YDV7UAO_1.ZIP), and File Size (247.4K).

Method 2: Data Loader (Specific Export)

You use the same Data Loader application to export specific data using a query. This is useful when you need a particular set of records right away.

Practice:

1. Open **Data Loader** and click **Export**.
2. Log in and select the **Donation (Donation__c)** object.
3. Choose a name and location for the export file (e.g., donations_export.csv).
4. Data Loader will now show you a query editor. You can write a SOQL query here to select exactly what you want. Copy and paste the following query:

SQL

```
SELECT Id, Donor_Name__c, Amount__c, Status__c FROM Donation__c
```

5. Click **Finish** and then **Yes**.

The operation will run, and the donations_export.csv file will be saved to your computer. Open it to see the data you exported

 Export

Step 3: Edit your Query

Export batch size: 500
Current API usage for the org: 492
API Limit for the org: 15,000



Choose the query fields below:

Amount_c (currency)
 CreatedById (reference)
 CreatedDate (datetime)

Select all fields Clear all fields

Create the where clauses to your query below

Field:
Operation: equals
Value:

Add condition Clear all conditions

The generated query will appear below. You may edit it before finishing.

```
SELECT Id, Donor_Name_c, Amount_c, Status_c FROM Donation_c
```

< Back Next > **Finish** Cancel

Result:-

 Export

 Operation Finished

Step
The operation has fully completed. There were 28 successful extractions and 0 errors.

View Extraction OK

Extracted 28 of 28 records with 28 successes and 0 errors.

Select all fields Clear all fields

The generated query will appear below. You may edit it before finishing.

```
SELECT Id, Donor_Name_c, Amount_c, Status_c FROM Donation_c
```

< Back Next > **Finish** Cancel

10. Packages

Unmanaged vs. Managed Packages

While Change Sets are used to move metadata between your *own connected orgs*, Packages are used to bundle and distribute your customizations to *other, unrelated Salesforce orgs*.

If a Change Set is like an internal company mail envelope sent between two departments, a Package is like a product you put in a box to ship to any customer in the world.

There are two main types of packages.

Unmanaged Packages

Think of these as an open-source template.

- **Purpose:** Best for one-time distributions or for sharing code templates.
- **Key Feature:** Once an unmanaged package is installed, all of its components (objects, code, etc.) are fully unlocked and can be edited by the administrator of that org. They essentially become part of the new org.
- **Upgrades:** They cannot be upgraded. To release a new version, the installer has to uninstall the old package and install the new one.
- **Use Case:** You create a set of useful reports and dashboards and want to share them with another department to use as a starting point.

Managed Packages

Think of these as a professional, sealed software product, like an app you'd buy.

- **Purpose:** Used by Salesforce partners to sell and distribute applications on the AppExchange.
- **Key Feature:** The components are "locked" and cannot be viewed or edited by the installer. This protects the developer's intellectual property.
- **Upgrades:** They are fully upgradeable. The developer can push new versions and bug fixes to customers who have the package installed.
- **Use Case:** You install an application like DocuSign or a financial calculator from the AppExchange.

11. ANT Migration Tool

The Ant Migration Tool is a command-line utility for moving metadata between your local computer and a Salesforce org. It's a step up from the point-and-click interface of Change Sets and requires a more technical setup.

Think of it this way: if Change Sets are like mailing a package at the post office, the Ant Migration Tool is like using a professional shipping service where you define everything in a manifest (package.xml) and use commands to script the entire process.

The High-Level Process

1. **Define package.xml:** You create an XML file that lists every single component you want to move (e.g., ApexClass, CustomObject, Flow).
2. **Retrieve:** You run a command in your computer's terminal to **retrieve** the metadata from a Salesforce org. This downloads all the components as source files into a folder on your computer.
3. **Deploy:** You then run another command to **deploy** those files from your computer to a different target org.

Why Use It?

It's more powerful than Change Sets because it's scriptable, repeatable, and can be integrated into automated systems (CI/CD). For many years, this was the primary tool for Salesforce developers.

While the Ant Migration Tool is still used in some cases, it has been largely replaced by the more modern **VS Code & SFDX**

Phase 9: Reporting, Dashboards & Security Review

This phase focuses on building meaningful reports and dashboards for the Helping Hands Foundation application and performing a final security review of the Salesforce org.

Step 1: Create Reports

1. Navigate to Reports → App Launcher → Search 'Reports' → Open Reports tab.

The screenshot shows the Salesforce Reports tab. The top navigation bar includes Home, Donations, Volunteers, Volunteer Tasks, Events, Volunteer Events, Reports, Dashboards, and Donation Summaries. The Reports tab is selected. On the left, there's a sidebar with 'Recent' and '4 items' under 'REPORTS'. The main area displays a table with columns: Report Name, Description, Folder, Created By, Created On, and Subscribed. The table contains five rows:

Report Name	Description	Folder	Created By	Created On	Subscribed
Monthly Donation Summary Report	Helping Hands Reports	Venkata Likith Sai Kovi	24/9/2025, 4:08 pm		
Total Donations by Volunteer	Shows total donated amount grouped by volunteer	Helping Hands Reports	Venkata Likith Sai Kovi	20/9/2025, 12:05 am	
Total Donations by Event	Shows total donated amount grouped by event	Helping Hands Reports	Venkata Likith Sai Kovi	20/9/2025, 12:00 am	
Program Overview by User	Analyze how different users are progressing towards a program and its outcome.	Enablement Dashboard Reports Spring '24	Automated Process	17/7/2025, 1:57 pm	

2. Create a New Report → Select object: Donations → Continue.

The screenshot shows the 'Create Report' dialog box. The title bar says 'Create Report'. The left sidebar has a 'Category' section with 'Recently Used' expanded, showing options like All, Accounts & Contacts, Opportunities, etc. Below that is a 'Select a Report Type' section with a search bar and a table of recently used report types. The table has columns: Report Type Name and Category. One row is selected: 'Donations' under 'Standard'. The right side of the dialog box is a 'Details' panel for the 'Donations' report type. It shows the report icon, name ('Donations'), status ('Standard Report Type'), a 'Start Report' button, and a list of fields (22). Below that are sections for 'Created By You' (listing three reports) and 'Created By Others' (listing 'No Reports Yet'). At the bottom is a section for 'Objects Used in Report Type' (listing 'Owner' and 'Donation').

3. Add Columns → Donor Name, Donation Amount, Donation Date, Payment Status.

Columns ▾

Add column...

Donor Name

Amount

Donation Date

Status

4. Apply Filters → Donation Date = Current Fiscal Year.

REPORT ▾

New Donations Report Donations

Fields > Outline Filters 2

Filters ▾

Add filter...

Show Me My donations

Donation Date
Current FY (01-Jan-2025 - 31-Dec-2025)

5. Group Records → Group by Donor Name, Group by Month (Donation Date).

Groups ▾

GROUP ROWS

Add group...

Donor Name

Donation Date

6. Save & Run Report → Name: Donation Summary Test Report → Save to Helping Hands Reports Folder.

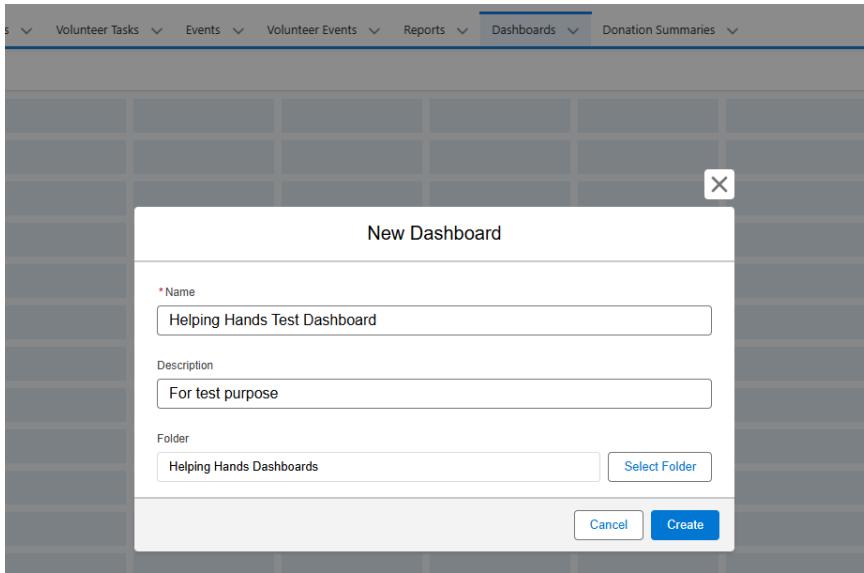


Report: Donations Donation Summary Test Report			
Total Records	Total Amount		
20 ₹5,83,101			
□ Donor Name ↑ ↓	Donation Date ↑ ↓	Amount ↑ ↓	Status ↑ ↓
□ 10000 (1)	25/09/2025 (1)	₹20,000	Pledged
	Subtotal	₹20,000	
Subtotal		₹20,000	
□ Aravind (1)	23/09/2025 (1)	₹3,000	Pledged
	Subtotal	₹3,000	
Subtotal		₹3,000	
□ Chandana (1)	23/09/2025 (1)	₹2,00,000	Received
	Subtotal	₹2,00,000	
Subtotal		₹2,00,000	
□ Karina (1)	11/09/2025 (1)	₹10,000	Received
	Subtotal	₹10,000	
Subtotal		₹10,000	
□ Likith sai (1)	19/09/2025 (1)	₹3,000	Received
	Subtotal	₹3,000	
Subtotal		₹3,000	
□ Ram (1)	25/09/2025 (1)	₹2,00,000	Received
	Subtotal	₹2,00,000	
Subtotal		₹2,00,000	
Row Counts	✓	Detail Rows	✓
Subtotals	✓	Grand Total	✓

⚡ Recent Items ⚡ Chatter Feed

Step 2: Create Dashboard

1. Navigate to Dashboards tab → New Dashboard → Name: Helping Hands Test Dashboard.



2. Add Components → Link reports to visualizations:

- Donations by Month (Bar Chart)
- Top Donors (Donut Chart)
- Total Donations (Number widget)

3. Configure Dashboard → Running User = Financial Officer → Save & Refresh.

Add Widget

Report: Donation Summary Test Report

Use chart settings from report

Display As: Bar Chart

X-Axis: Donor Name

Y-Axis: Sum of Amount

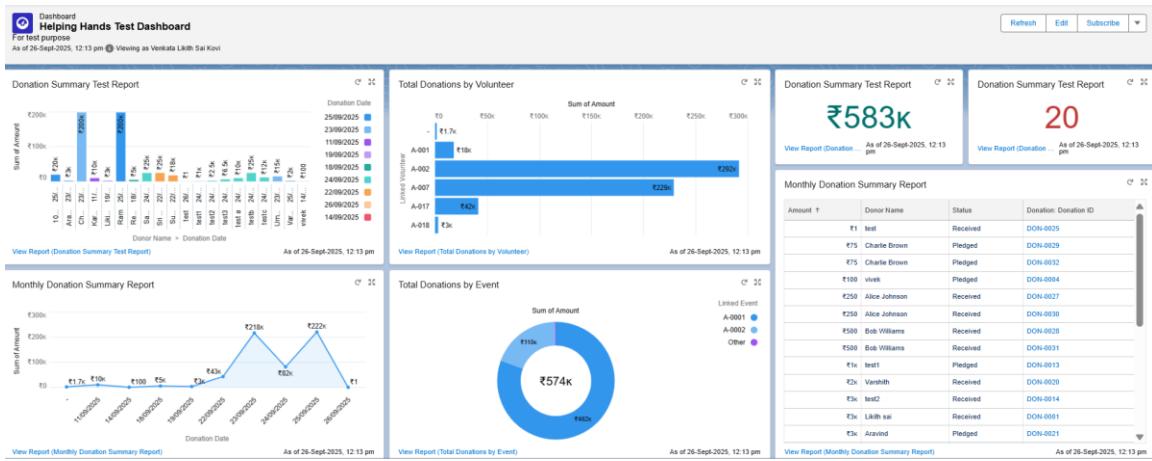
Preview:
Donation Summary Test Report

Donation Date	Sum of Amount (₹)
25/09/2025	₹20k
23/09/2025	₹3k
11/09/2025	₹10k
19/09/2025	₹3k
18/09/2025	₹25k
24/09/2025	₹5k
22/09/2025	₹25k
26/09/2025	₹18k
14/09/2025	₹10k

[View Report \(Donation Summary Test Report\)](#)

Buttons: Cancel, Add

4. Create Dynamic Dashboard → Run as logged-in user.



Step 3: Security Review

1. **Sharing Settings** : Setup → Sharing Settings → Ensure Donations = Private, Volunteer = Public Read Only.



2. **Field-Level Security** : Setup → Object Manager → Fields → Set Field-Level Security for sensitive fields.

Field Label	Amount	Visible	Read-Only
Data Type	Currency(18, 0)		
Analytics Cloud Integration User		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Analytics Cloud Security User		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Anypoint Integration		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Contract Manager		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Cross Org Data Proxy User		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Custom: Marketing Profile		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Custom: Sales Profile		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Custom: Support Profile		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Donation Manager Profile		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Einstein Agent User		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Force.com - App Subscription User		<input checked="" type="checkbox"/>	<input type="checkbox"/>

3. Session Settings : Setup → Session Settings → Timeout 2 hrs, Require HTTPS.

Session Settings

Set the session security and session expiration timeout for your organization.

Session Timeout

Timeout Value: 2 hours

Disable session timeout warning popups

Force logout on session timeout

4. Login IP Ranges : Setup → Profiles → System Administrator → Add allowed IPs.

Profiles

Login Hours

No login hours specified

Login IP Ranges

Action	IP Start Address	IP End Address
Edit Del	103.255.146.242	103.255.146.242

5. Audit Trail : Setup → View Setup Audit Trail → Review all admin actions.

View Setup Audit Trail

The last 20 entries for your organization are listed below. You can [download](#) your organization's setup audit trail for the last six months (Excel .csv file).

View Setup Audit Trail

Date	User	Source Namespace Prefix	Action	Section	Delegate User
26/09/2025, 12:30:59 pm IST	sailikith511187@agentforce.com		Added Login Ip Range to System Administrator from 103.255.146.242 to 103.255.146.242	Manage Users	
26/09/2025, 10:49:13 am IST	sailikith511187@agentforce.com		Requested an export	Data Export	
26/09/2025, 10:44:25 am IST	sailikith511187@agentforce.com		For the 01lgK000002KSIR duplicate rule Prevent_Duplicate_Donations, changed "Active" from "false" to "true"	Duplicate Rule	
26/09/2025, 10:44:22 am IST	sailikith511187@agentforce.com		For duplicate rule Prevent_Duplicate_Donations, changed matching rules.	Duplicate Rule	
26/09/2025, 10:44:22 am IST	sailikith511187@agentforce.com		Created new 01lgK000002KSIR duplicate rule "Prevent_Duplicate_Donations". Set "Record-Level Security" to "Enforce sharing rules"	Duplicate Rule	
26/09/2025, 10:40:58 am IST	sailikith511187@agentforce.com		Donation matching rule, Donation_Matching_Rule, activating by Venkata Likith Sai Kovi	Matching Rule	
26/09/2025, 10:40:46 am IST	sailikith511187@agentforce.com		For matching rule Donation_Matching_Rule, added matching criteria where matching method is Exact, the field is Donor_Email and match blank fields is "Does Not Match If Null"	Matching Rule	
26/09/2025, 10:40:46 am IST	sailikith511187@agentforce.com		For matching rule Donation_Matching_Rule, added matching criteria where matching method is Exact, the field is Donor_Name and match blank fields is "Does Not Match If Null"	Matching Rule	
26/09/2025, 10:40:46 am IST	sailikith511187@agentforce.com		For matching rule Donation_Matching_Rule, matching engine set to Exact Match Engine.	Matching Rule	
26/09/2025, 10:40:46 am IST	sailikith511187@agentforce.com		Created new Donation matching rule Donation_Matching_Rule	Matching Rule	

Step 4: Testing Reports & Security

1. Login as Volunteer Coordinator → Open Dashboard → Check restricted data hidden.

The screenshot shows a dashboard titled "Enablement Dashboard". It contains eight report cards arranged in two rows of four. Each report card has a red error icon and the message "The source report is based on a report type that is inaccessible to the dashboard's running user." Below each card is a "View Report" button.

2. Login as Financial Officer → Verify donation details are visible.

The screenshot shows a dashboard for a Financial Officer. It includes a "Helping Hands Man..." logo and various navigation links. The main area displays several reports:

- "Monthly Donation Summary Report": A pie chart showing a total of ₹584k, a bar chart by date, and a line chart.
- A summary box showing "₹584k" with "View Report (Mo... As of 26-Sept-2025, 10:08 am)".
- A table of donation details with 15 entries:

Amount	Donor Name	Status	Donation ID
₹1 test		Received	DON-0025
₹75 Charlie Brown		Pledged	DON-0029
₹100 vivek		Pledged	DON-0004
₹250 Alice Johnson		Received	DON-0027
₹500 Bob Williams		Received	DON-0026
₹500 Bob Williams		Received	DON-0028
₹100 test1		Pledged	DON-0013
₹2k Vanshith		Received	DON-0020
₹3k test2		Received	DON-0014
₹3k Likith sai		Received	DON-0001
₹1000 test3		Pledged	DON-0015
₹1000 test4		Pledged	DON-0016
₹1000 test5		Pledged	DON-0017
₹1000 test6		Pledged	DON-0018
₹1000 test7		Pledged	DON-0019