Phase 2 – Org Setup & Configuration

This phase involved configuring the **Salesforce Developer Org** to build a secure, structured, and role-based environment for the **Insurance Claim and Policy Management System**.
The goal was to ensure proper **user access, data visibility, and control** before creating custom objects, relationships, and automation.

---

1. Salesforce Developer Edition Setup

The project was developed entirely in **Salesforce Lightning Developer Edition**, which supports:

- Custom Object and Field creation
- Workflow and Validation Rules
- Profiles, Roles, Permission Sets
- Reports and Dashboards

A single Developer Org was used for end-to-end development and testing.
This environment provided full admin access to configure the data-model and security settings.

Procedure:

1. Logged into Salesforce Developer Org.
2. Enabled **Lightning Experience** via Setup → User Interface.
3. Verified access to Object Manager, Security, and Setup tools.
4. Created a dedicated App for **Insurance Policy Management**.

Brief Purpose:
To establish a fully functional development workspace with all customization features enabled.

---

2. User Setup

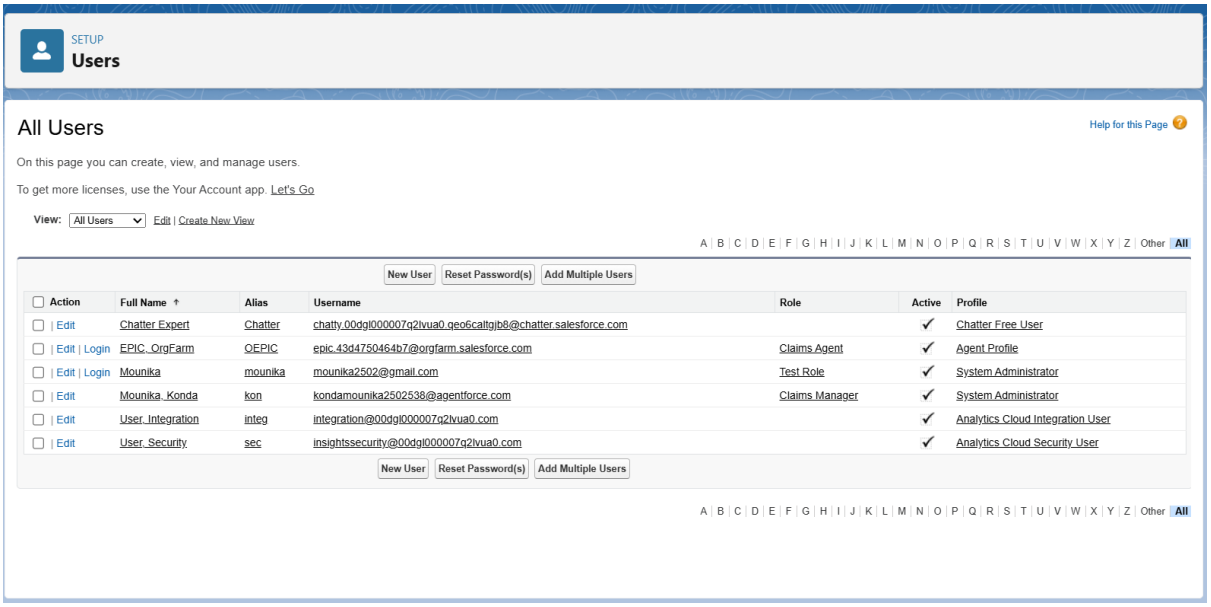
Three users were created to simulate the working hierarchy:

- **Administrator** – complete access to configure and monitor all modules
- **Manager** – reviews and approves claims, monitors agent activities
- **Agent** – creates and manages customers, policies, and claim records

Each user account was assigned a role and profile to reflect real-world responsibilities.
Test logins were used to verify access privileges for each type of user.

Procedure:

1. Setup → Users → New User.
2. Created three users and assigned appropriate profiles and roles.
3. Logged in as each user to validate access and permissions.



---

3. Profiles Configuration

Custom profiles were set up to define **object-level and field-level access**:

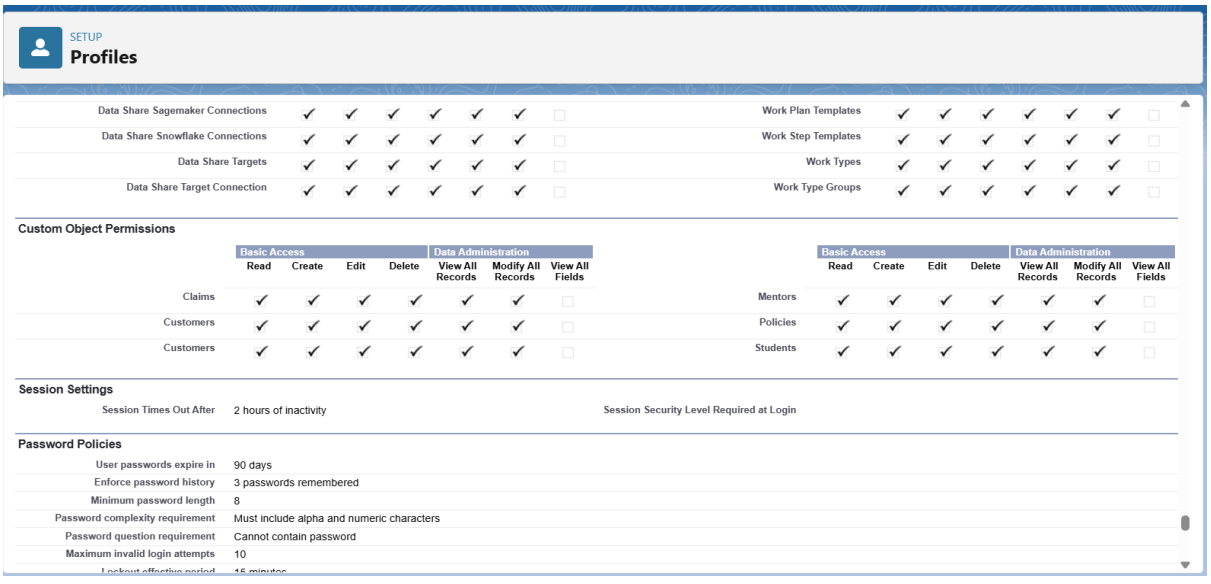
- **System Administrator Profile:** Full CRUD access on all objects (Customers, Policies, Claims, Mentors)
- **Manager Profile:** Read and Edit access on records created by their agents
- **Agent Profile:** Create and View access only for their own records

Field-level security was applied to hide or make read-only sensitive data such as **Claim Amount** or **Policy Coverage** for lower roles.

Procedure:

1. Setup → Profiles → Clone Standard User profile to create Agent profile.
2. Modified Object Settings and Field-Level Security per role.
3. Saved and tested profile functionality.

Brief Purpose:
Profiles control baseline permissions and ensure sensitive policy or claim data cannot be modified by unauthorized users.



4. Role Hierarchy

A hierarchical structure was implemented as:

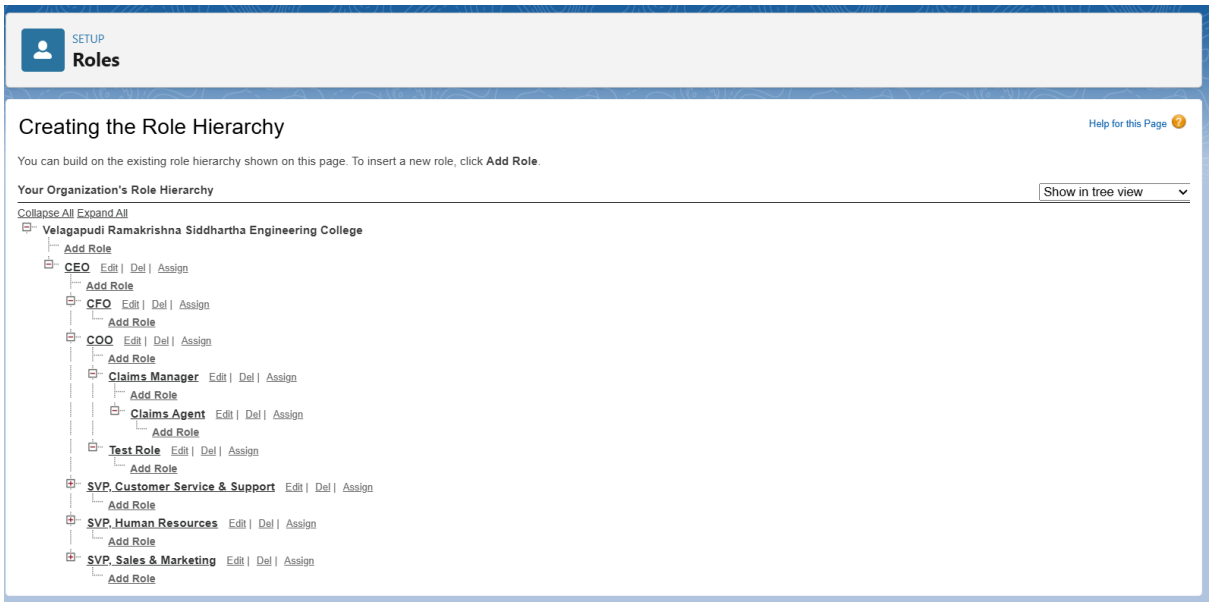
Admin → Manager → Agent

- Agents can view and edit only their own records
- Managers can view and approve the records of their reporting agents
- Admin has full organization-wide visibility

Procedure:

1. Setup → Roles → Set Up Roles → Add Role.
2. Created roles and assigned them to respective users.
3. Verified record visibility according to hierarchy.

Brief Purpose:
Role hierarchy establishes the approval and reporting chain, enabling upward visibility for monitoring and validation.



5. Permission Sets

Created an additional **"Claims Extended Access"** permission set:

- Temporarily granted agents edit permission on claim status during workflow testing
- Allowed controlled flexibility without changing the base profile

Procedure:

1. Setup → Permission Sets → New.
2. Enabled Edit permission on Claim records.
3. Assigned permission set to selected Agent users.

Brief Purpose:
Permission sets were used to temporarily extend access without altering default profile permissions.

6. Organization-Wide Defaults (OWD)

Configured base record-level security for all custom objects:

Object	Default Access	Description
Customers	Private	Only owner (agent) and their manager can view/edit
Policies	Controlled by Parent	Access inherited from related Customer
Claims	Private	Only record owner or approver can view/edit

Procedure:

1. Setup → Sharing Settings → Organization-Wide Defaults.
2. Set access levels per object.
3. Saved changes.

Brief Purpose:
OWD ensures data confidentiality and prevents unauthorized data visibility between different agents.

---

7. Sharing Rules

Created a **Manager-Level Sharing Rule** to automatically share agent-owned records with their respective managers:

- Enabled Managers to review, verify, and approve claim or policy details created by agents

Procedure:

1. Setup → Sharing Settings → Sharing Rules → New.
2. Set criteria: Owner Role = Agent → Share with Manager role.
3. Access Level: Read/Write.
4. Repeated for Claims and Policies.

Brief Purpose:
Sharing rules extend record access upward in the hierarchy for supervision and claim verification.

---

8. Login Access & Testing

Enabled Admin login access for troubleshooting user-level issues.
Each configuration was tested by logging in as **Admin, Manager, and Agent** to confirm proper visibility and permission flow.

Procedure:

1. Logged in as Agent — created sample Customer, Policy, and Claim.

2. Logged in as Manager — verified access to agent-created records.
3. Logged in as Admin — checked organization-wide visibility.
4. Confirmed restricted fields (Claim Amount, Policy Coverage) were hidden for Agents.

Brief Purpose:
Testing verified that security and record-sharing behaviors worked correctly before moving to customization and automation.

---

9. Developer Org Verification

All configurations, metadata, and role-based rules were created and tested inside a single Salesforce Developer Org. Verified successful:

- Role-based access control
- Record visibility flow
- Field-level security restrictions
- Claim approval visibility from **Agent → Manager → Admin**

Brief Purpose:
Ensures the Salesforce org is stable, secure, and ready for next phases.

The Salesforce Developer Org was fully configured with:

- Secured access
- Hierarchical data visibility
- Controlled sharing mechanisms