Finance & Accounting Manual

Odoo Manual

Author: Axon System

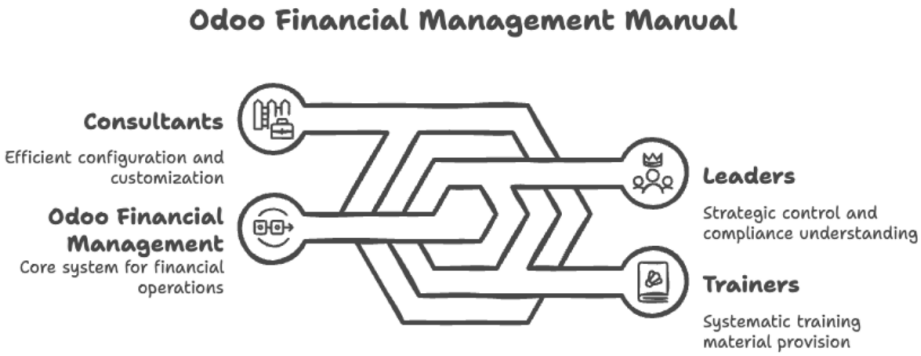
Date: September 14, 2025

Contents

1 Introduction

1.1 Purpose of this Manual

This manual is meant for leaders, trainers, and consultants working with financial management in Odoo. Its objectives are:



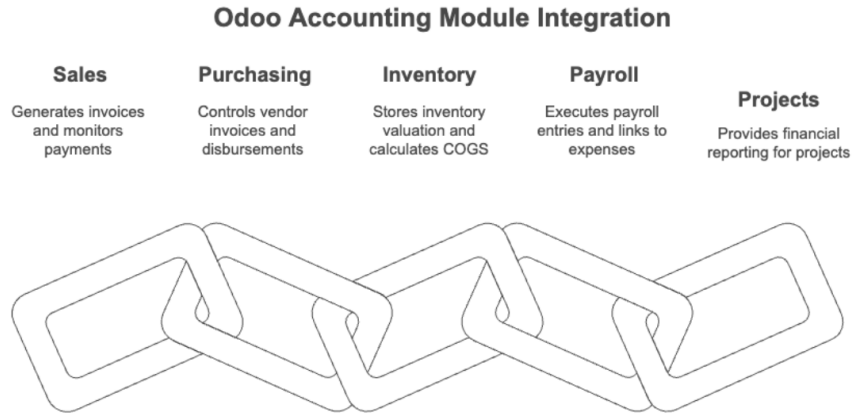
Leaders: To help you understand strategic financial control and compliance using Odoo.
Trainers: For the purpose of providing systematic material for training accountants.
Consultants: To enable efficient configuration, localization, and customization of Odoo for financial management.

1.2 Enterprise Architecture Context

This section describes the place of the Accounting module in the broader Odoo enterprise architecture and delineates the underlying technology stack that supports its functionality.

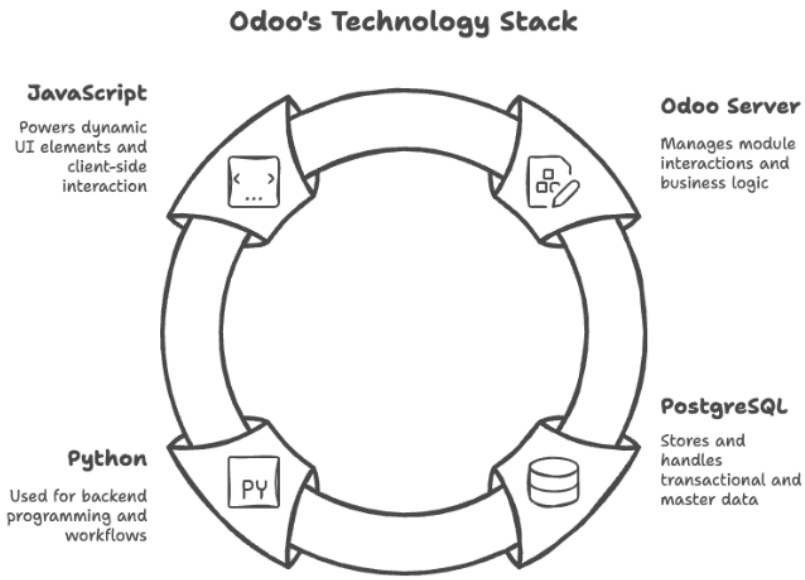
Position of the Accounting Module
The Accounting module is the fiscal backbone of the Odoo world. It brings together financial transactions and reporting, ensuring smooth fiscal control throughout departments. Thanks to its integration with several functional modules, it offers real-time financial visibility and control throughout the entire organization.

Integration Points with Other Modules
The Accounting module has strong integration points with some of the most significant Odoo modules, enabling end-to-end financial process automation:



Sales: Generates customer invoices automatically and monitors payment follow-ups.
Purchasing: Controls vendor invoices and disbursements going out.
Inventory: Stores inventory valuation in real time and on-the-fly calculation of COGS.
Payroll: Executes payroll journal entries and link to employee expense capture.
Projects: Provides financial reporting for project-level expense, revenue, and profitability.

Technology Stack Overview
Odoo's solid, modular design supports the Accounting module. Among the key constituents of the technology stack are:



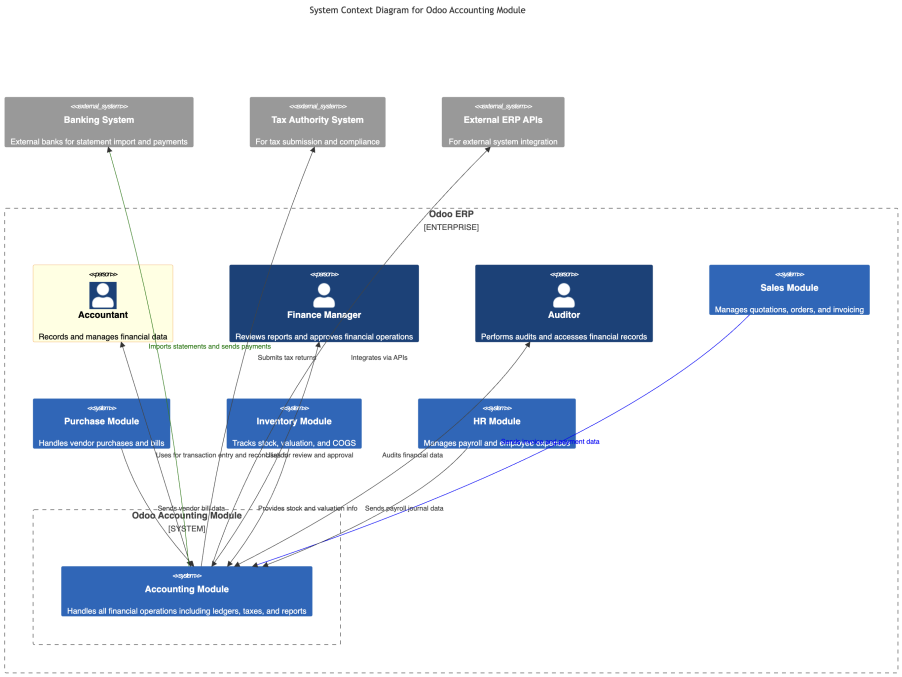
Odoo Server: The main framework that manages all module interactions and business logic.
PostgreSQL: A powerful open-source relational database management system used to store and handle transactional and master data.
Python: The primary backend programming language used for business logic coding and workflows.
JavaScript (JS): Used for dynamic user interface elements and client-side interaction within the web client.

1.3 Supported Business Capabilities

Odoo Accounting module provides organizations with an integrated suite of finance management capabilities, enabling effective, accurate, and compliant financial activities. Some of the key business capabilities are:

General Ledger Accounting
Maintains a structured chart of accounts and records all financial transactions, which form the foundation for proper financial reporting.
Accounts Receivable & Accounts Payable
Monitors payments received from customers and payments made to vendors, offering visibility and control over outstanding balances and obligations.
Bank Reconciliation
Automates reconciliation of bank statements with company records to identify discrepancies and maintain data integrity.
Tax Reporting & Compliance
Simplifies the generation of tax reports and maintains compliance with local and foreign tax legislation, such as VAT, GST, and other statutory requirements.
Financial Reporting & Audits
Prepares real-time financial reports such as balance sheets, income statements, and cash flow statements, and supports internal and external audit processes.
Multi-Currency & Multi-Company Accounting
Enables global operations by permitting accounting in multiple currencies and processing consolidated or separate financials for several companies in a single Odoo instance.

1.4 C4 Model - System Context Diagram (C1)



The C1 diagram shows the big picture of how the Odoo Accounting Module fits into the Odoo system. It outlines the users, internal modules in communication, and external systems, but also focuses on how each talks to or depends upon the accounting module. **Interactions:**

Users:

i. **Accountant:** Uses the accounting module to enter journal entries, process financial transactions, reconcile bank statements, and generate reports.

ii. **Finance Manager:** Generates financial reports, authorizes transactions, monitors cash flows, and verifies overall finance compliance via the module.

iii. **Auditor:** Puts accounting records into perspective to perform audits, ensures compliance checks, and submits regulatory ones.

Internal Odoo Modules:

i. **Sales Module:** Sends customers' invoice and payment information to the accounting module for accounting revenues and monitoring receivables.

ii. **Purchase Module:** Puts vendor bill and purchase transaction records into the module to process accounts payable.

iii. **Inventory Module:** Provides stock value and COGS information to the financial ledgers for updating.

iv. **HR Module:** Updates payroll, employee expense reimbursements, and salary-related journal entries into the accounting system.

External Systems:

i. **Banking System:** Exchanges information bi-directionally with the accounting module. Imports bank statements for reconciling. Exports payment instructions for vendor payments and salary payments.

ii. **Tax Authority System:** Accepts tax filings and compliance reports sent via the accounting module.

iii. **External ERP APIs:** Allows integration with third-party ERP systems for data exchange, e.g., invoices, payments, or financial summaries.

Dependencies:

i. **Data from other Odoo modules:** Depends on the Sales, Purchase, Inventory, and HR modules to automatically receive transaction data, minimizing manual data entry and errors.

ii. **External banking integration:** Depends on connectivity to banking systems for statement imports and automatic payments.

iii. **Tax system access:** Depends on integration with government or regional tax authority systems for compliant reporting.

iv. **API integration:** Connects through APIs to connect with external ERPs, enhancing interoperability and facilitating real-time financial visibility across the platform.

v. **User roles and access control:** Requires proper user role configurations to enable secure and compliant access for accountants, managers, and auditors.

Summary
Interactions determine who uses the module and what systems send/receive data. Dependencies emphasize systems and sources of data required by the accounting module to accurately and efficiently operate. Combined, these linkages place the accounting module at the center of business operations, facilitating smooth financial management within and between departments and systems.

2 Finance & Accounting Architecture Context

2.1 Module Architecture

The Odoo Accounting module consists of several combined elements which, collectively, make a solid and combined finance administration solution. They are designed so that they support any number of accounting processes and they stay precise, compliant, and easy to use:

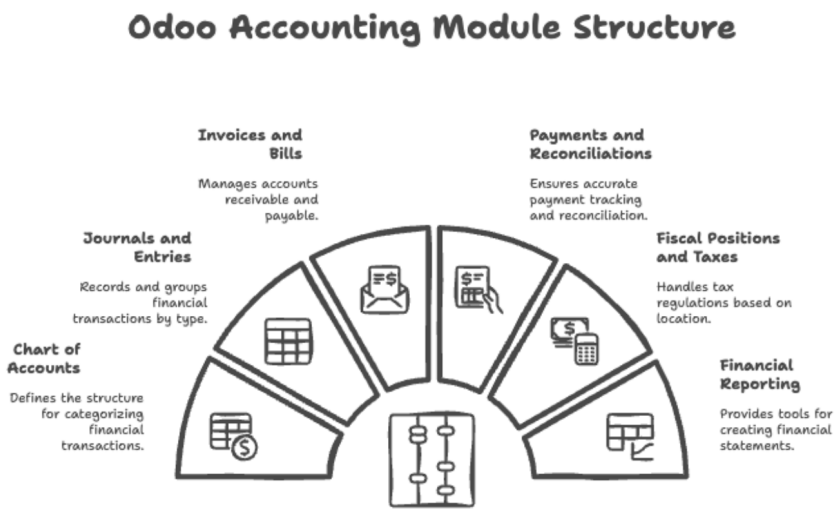


Chart of Accounts
As the backbone of the accounting system, the Chart of Accounts defines the accounts that are used to categorize and post all financial transactions. It is a hierarchical and standardized structure that facilitates structured financial information across the organization.

Journals and Journal Entries
Financial transactions are entered in journals, with one journal entry representing a definite business transaction (e.g., sale, purchase, payment). Different types of journals—such as sales, purchases, cash, or miscellaneous—help to group and trace transactions by their nature and source.

Invoices and Bills
Invoices are issued to customers to request payment for goods or services, while bills are received from suppliers as account of amounts owed. The documents are crucial in managing accounts receivable and accounts payable to record both incoming and outgoing payments correctly.

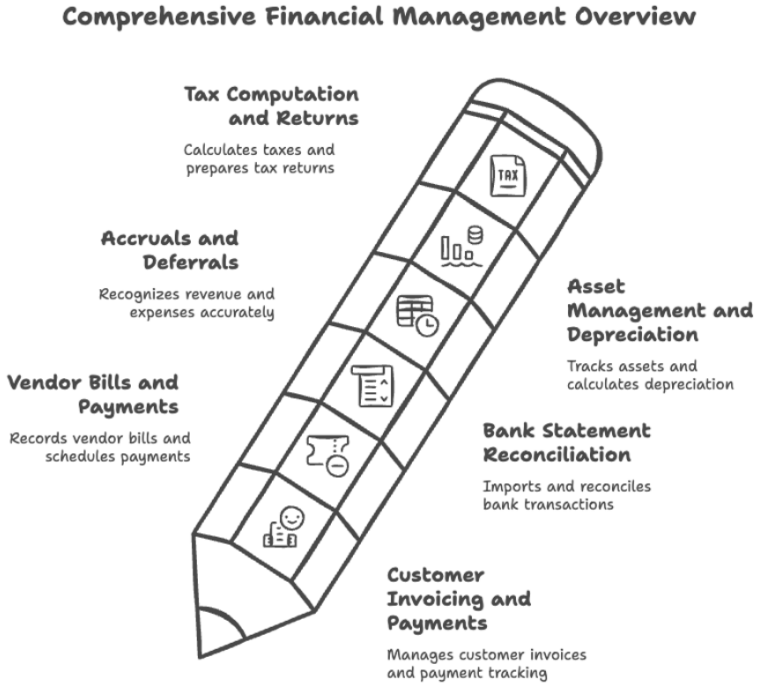
Payments and Reconciliations
This module manages capture of payments received from customers and payments made to suppliers. Reconciliation features embedded in the application allow users to reconcile payments with their corresponding invoices or bills for proper, up-to-date financial records and proper bank reconciliations without any errors.

Fiscal Positions and Tax Configuration
Fiscal positions allow for dynamic handling of tax regulations based on customers' and vendors' location or status. This ensures that tax calculations are correct and in accordance with local and international tax legislation, especially in multi-jurisdictional settings.

Financial Reporting Tools
Odoo Accounting has a robust set of reporting tools which enable the creation of financial statements such as balance sheet, profit & loss, tax reports, audit-ready transaction histories, and analytical reports customized to specific requirements. The above-mentioned information helps in strategic decision-making and facilitates financial transparency.

2.2 Key Features and Workflows

The Odoo Accounting module offers a comprehensive suite of features and workflows to manage various financial processes. Here's an overview:

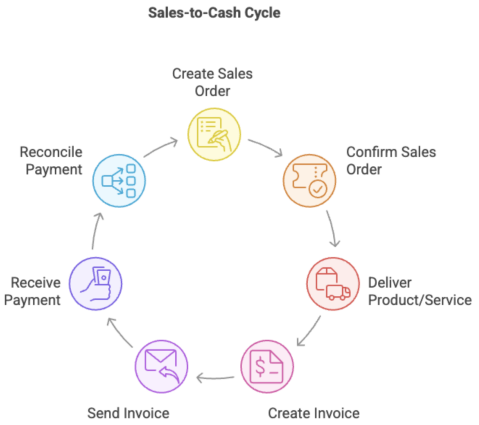


Customer invoicing and payments: Create and manage customer invoices, track payments, and automate payment reminders.
Vendor bills and payments: Record and manage vendor bills, schedule payments, and track payment history.
Bank statement reconciliation: Import bank statements, match transactions, and reconcile bank accounts.
Accruals and deferrals: Recognize revenue and expenses in the correct accounting period, regardless of when cash changes hands.
Asset management and depreciation: Track fixed assets, calculate depreciation, and manage asset disposal.
Tax computation and returns: Calculate taxes on transactions, generate tax reports, and prepare tax returns.

Key Workflows:

Sales-To-Cash:

1. Create a sales order.
2. Confirm the sales order.
3. Deliver the product/service.
4. Create an invoice.
5. Send the invoice to the customer.
6. Receive payment from the customer.
7. Reconcile the payment with the invoice.



Example:

1. **Create a Sales Order**
   Corporate customer, TechHive Ltd., contacts Big Business Cafe to order coffee and snacks for a seminar.

   Salesperson creates a Sales Order in Odoo:

Product	Quantity	Price
Espresso	100 cups	Rs 200
Croissants	50 pieces	Rs 150
	Total	**Rs 350**

2. **Validate the Sales Order**
   Sales Order is validated in Odoo. This action keeps stock reserved and triggers delivery process.

3. **Deliver the Product/Service**
   Staff delivers the 100 cups of coffee and 50 croissants to TechHive's seminar room on the day of the event. The Delivery Order status is marked as completed in Odoo.

4. **Generate an Invoice**
   As soon as the delivery is confirmed, the system allows generating an invoice. An invoice for Rs 350 can be generated and is ready to send.

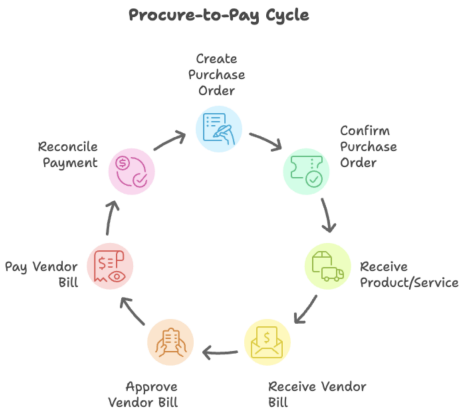
5. **Send the Invoice to the Customer**
   The invoice is sent automatically from Odoo to TechHive's accounts department. A PDF invoice is automatically attached.

6. **Get Payment from the Customer**
   After one week, TechHive Ltd., pays the invoice via bank transfer. The payment is recorded in Odoo as Customer Payments.

7. **Reconcile the Payment with the Invoice**
   Once the bank statement is imported or synced, the payment is reconciled with the invoice. The invoice status is now Paid.

Procure-to-Pay:

1. Create a purchase order.
2. Confirm the purchase order.
3. Receive the product/service.
4. Receive a vendor bill.
5. Approve the vendor bill.
6. Pay the vendor bill.
7. Reconcile the payment with the bill.



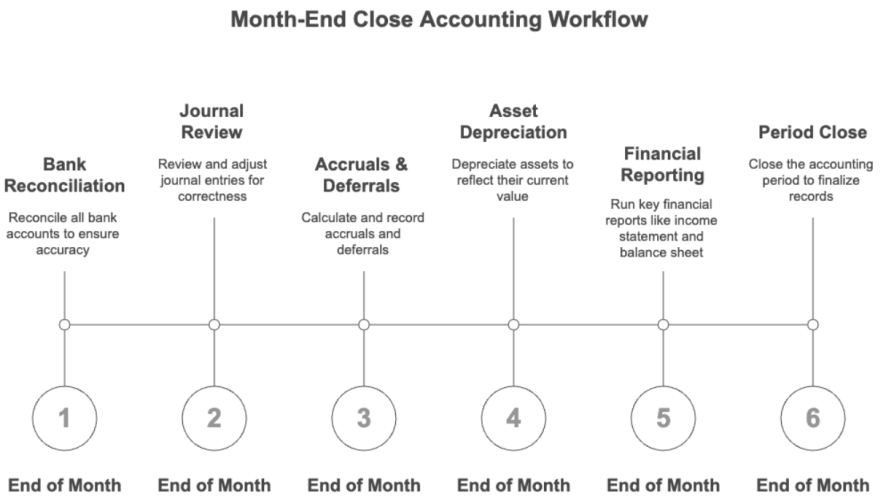
Example:

1. **Create a Purchase Order**
   Big Business Cafe requires additional coffee beans and croissants. The purchasing manager creates a Purchase Order in Odoo to their supplier, FreshRoast Suppliers:

Product	**Quantity**	**Price**
Premium Coffee Beans	20 kg	Rs 300
Frozen Croissants	200 pieces	Rs 400
	Total	**Rs 700**

2. **Confirm the Purchase Order**
   The PO is confirmed in Odoo. This notifies the supplier and starts expected delivery.
3. **Receive the Product/Service**
   The supplier delivers the coffee beans and croissants to the cafe. The receiving team logs the delivery in Odoo by validating the Receipt.
4. **Receive a Vendor Bill**
   FreshRoast Suppliers sends a Vendor Bill for Rs 700. The bill is posted in Odoo and matched with the original PO and receipt.
5. **Approve the Vendor Bill**
   The bill is validated in Odoo by the finance manager for payment.
6. **Pay the Vendor Bill**
   The payment (by bank transfer) is done and registered in Odoo in Vendor Payments.
7. **Reconcile the Payment with the Bill**
   When the bank statement is synced or imported, the payment is reconciled with the vendor bill. The status of the bill is now Paid, and accounting is updated.

Month-End Close:



1. Reconcile all bank accounts.
2. Review and adjust journal entries.
3. Calculate and record accruals and deferrals.
4. Depreciate assets.
5. Run financial reports (e.g., income statement, balance sheet).
6. Close the accounting period.

Example:

1. **Reconcile All Bank Accounts**
   Bank statements are downloaded by the accounting team using bank synchronization or CSV upload to Odoo. Customer payments (e.g., TechHive) and vendor payments (e.g., FreshRoast) are reconciled to invoices and bills. Transactions are reconciled to ensure no unmatched or outstanding entries exist.
2. **Review and Adjust Journal Entries**

   The accountant verifies accuracy in the following entries:

   (a) Sales revenue
   (b) Expense allocations (e.g., utilities, salaries, supplies)
   (c) Manual rounding adjustments or corrections

   Adjusting entries are posted as necessary in Odoo to account for real economic activity.
3. **Accrue and Record Accruals and Deferrals**
   Accruals: Rent owed this month but not yet received — a journal entry is recorded to accrue Rs 2,000.
   Deferrals: Prepaid catering service from a customer for the next month — revenue is deferred next period.
   These items are entered through Odoo's manual journal entry feature.

4. **Depreciate Assets**
   The cafe has a coffee machine and some furniture that were indicated as fixed assets. With the Assets module in Odoo, depreciation each month is automatically calculated and posted as follows:
   (a) Coffee Machine - Rs 100 per month
   (b) Furniture - Rs 50 per month
5. **Print Financial Reports**
   The following are printed by the finance manager inside Odoo:
   (a) Income Statement (Profit & Loss): revenue versus expenses
   (b) Balance Sheet: verify assets, liabilities, and equity
   (c) Cash Flow Statement: knowing liquidity
6. **Close the Accounting Period**
   Upon review and approval, the accounting period is closed out in Odoo. This prevents further backdated entries and protects the period to be audited.

2.3 User Roles and Permissions

Odoo has a fine-grained, customizable user roles and permissions system that ensures access to accounting functionality according to each user's responsibility. This structure provides protection for data, regulatory adherence, and effectiveness in the financial workflows. **Default**

Roles
Odoo is equipped with predefined default roles designed to meet common accounting tasks:

Accountant:
Accountants have wide access to the core functions of the accounting module. They can:

1. Create and maintain customer invoices and vendor bills
2. Record and confirm payments
3. Post and amend journal entries
4. Reconcile bank statements
5. Generate financial and tax reports

Advisor:
The Advisor role provides the highest level of access within the accounting module. In addition to all Accountant capabilities, Advisors can:

1. Set up accounting parameters
2. Maintain the chart of accounts and fiscal positions
3. Perform sophisticated analysis and close accounting periods
4. Manage financial compliance and audit processes

Role Mapping through Security Groups:
Odoo uses security groups to define and control the access levels by role. Security groups dictate what data a user can view or edit, and what menus and actions are available to them. Examples:

1. Accounting & Finance / Accountant: Basic accounting access.
2. Accounting & Finance / Advisor: All access including configuration and sensitive actions

Security groups are assigned to users through their role setup and can be changed accordingly.

Custom Roles:
Custom roles can be created by companies to fit access for specific internal business processes or regulatory requirements. For instance:

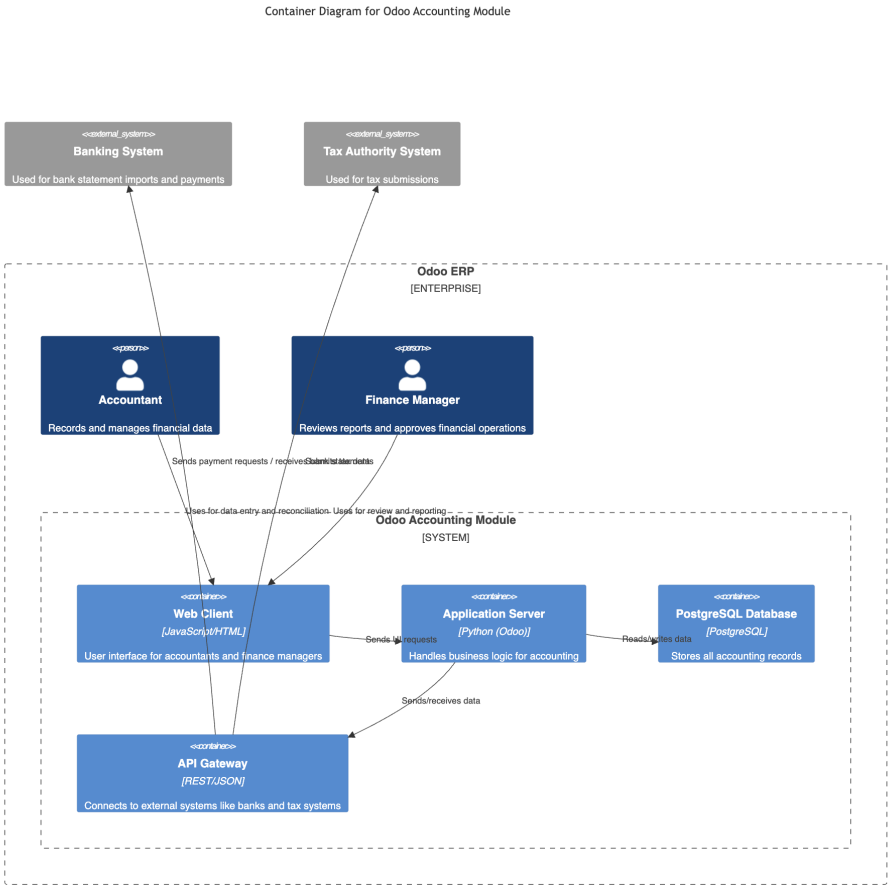
1. **Auditor**
   Read-only role that grants access to all relevant financial data but cannot be changed. Ideal for internal or external auditors to look at records and generate audit reports without fear of data modification.
2. **Junior Accountant**
   A restricted role to perform core data entry activities such as posting invoices, bills, and performing simple reconciliations. Access rights can restrict visibility to confidential activities such as approving payments, modifying the chart of accounts, or period-end closings.

Creating a Custom Role in Steps

1. **Copy an Existing Role**
   Begin from an existing role (e.g., Accountant) so that there is a foundation to start from a set of common permissions.
2. **Modify Associated Security Groups**
   Set up the security groups to fine-tune the degree of access. You may turn on or turn off access to specific models, views, menus, and reports.
3. **Assign the Role to Users**
   Once you've established the custom role, assign the role to the designated users via their user profile.

This roles-and-permissions framework enables companies to enforce proper segregation of duties and responsibility upon accounting procedures, but remain flexible enough to compensate for shifting organization needs.

2.4 C4 Model - Container Diagram (C2)



The C2 diagram highlights Odoo Accounting module technical architecture by depicting its major containers and how they interact. They include the Odoo server, the PostgreSQL database, and the Web client. Together, they form the system runtime and data-processing environment core.

In the middle lies the Odoo Server, built using Python for business logic and XML for views and configuration. The server addresses all critical accounting tasks, including invoice management, journal entry, reconciliation of accounts, and reporting. Workflow logic is managed by the server, and communication from the database to the user interface is planned.

The PostgreSQL database serves as the persistent storage for all the accounting information customer invoices, vendor bills, journal entries, tax information, and reporting information. All of the transactions initiated on the server are stored and retrieved within this robust relational database system, ensuring data integrity and transaction validity.

The Web client, developed using JavaScript, provides the front-end interface through which users like finance managers and accountants interact with the system. These include the generation of invoices, looking at financial dashboards, bank statement reconciliation, and the generation of financial reports. The web client communicates in real-time with the Odoo server via RPC and REST calls to fetch or send data.

Data Flows

Invoice Creation → Journal Entry Generation: When a user posts a customer invoice through the web client, the data is sent to the Odoo server, and journal entries for the invoice are created automatically. The entries are committed to the PostgreSQL database.

Real-Time Posting and Report Generation: While transactions are being posted, financial records are updated in real-time by the system. Users can instantly view updated balance sheets, profit & loss statements, and tax summaries from the web interface due to the smooth data sync between the Odoo server and PostgreSQL.

This kind of architecture delivers a responsive, scalable, and highly integrated accounting experience within the Odoo ERP environment, enabling accurate and real-time financial management.

3 Data Structure and Relationships

3.1 Key Data Models

Odoo's Accounting module is developed around several major models that complement one another to form a complete and accurate financial system. Below is an improved analysis of the most important models, their business impact, primary fields, and relation.

1. **account.move (Journal Entries)**

   Business Significance:

   Is a journal entry, the most basic double-entry bookkeeping unit. It records economic transactions between accounts.

   Key Fields:

   name (Char): Reference to journal entry (auto-generated sequence).

   date (Date): Date of transaction.

   journal_id (Many2one): Journal it belongs to (e.g., Sales, Purchase).

   company_id (Many2one): Company owning the entry.

   state (Selection): Status (e.g., Draft, Posted).

   line_ids (One2many): Journal lines belonging to this entry.

   amount_total (Monetary): Total value of the transaction.

   Relationships:

   One-to-many with account.move.line

   Many-to-one with account.journal, res.company

2. **account.move.line (Journal Items)**

   Business Significance:

   A line is a specific debit or credit in a journal entry. It defines the amount, account, and optionally, the related partner.

   Key Fields:

   move_id (Many2one): Parent journal entry.

   account_id (Many2one): Ledger account involved.

   partner_id (Many2one): Close customer/vendor.

   name (Char): Description of entry.

   debit, credit (Monetary): Amounts.

   amount_currency (Monetary), currency_id: Currency details.

   tax_ids (Many2many): Taxes applied.

   Relationships:

   Many-to-one with account.move, account.account, res.partner

   Many-to-many with account.tax

3. **account.account (Chart of Accounts)**

   Business Significance:

   Creates individual accounts in the chart of accounts (COA) utilized to categorize transactions.

   Key Fields:

   code (Char): Alphanumeric account code.

   name (Char): Description of account.

   user_type_id (Many2one): Account type (income, expense, etc.).

   reconcile (Boolean): Whether reconcilable or not.

   company_id (Many2one): The company it belongs to.

   Relationships:

   One-to-many with account.move.line

   Many-to-one with account.account.type, res.company

4. **account.payment (Payments)**

   Business Significance:

   Captures incoming/outgoing money flow for customer receipts and vendor payments.

   Key Fields:

   date (Date): Date of the payment.

   payment_type (Selection): In or out.

   partner_type (Selection): Customer or vendor.

   partner_id (Many2one): Concerned party.

   amount (Monetary): Amount paid.

   currency_id (Many2one): Used currency.

   journal_id (Many2one): Origin journal.

   payment_method_id (Many2one): Means (e.g., check, wire).

   move_id (Many2one): Related journal entry.

   Relationships:

   Many-to-one with res.partner, account.journal, account.payment.method, account.move

5. **account.tax and account.fiscal.position**

   account.tax

   Business Significance:

   Specifies tax rules and rates for sales and purchases.

   Key Fields:

   name (Char): Tax name (e.g., VAT 13%).

   amount (Float): Rate.

   amount_type (Selection): Fixed, percentage, or division.

   type_tax_use (Selection): Sale, purchase, or both.

account_id (Many2one): Linked account for tax posting.

Relationships:

One-to-many with account.move.line

account.fiscal.position

Business Significance:

Applies tax mappings for specific partner location or type.

Key Fields:

name (Char): Position name (e.g., Export).

company_id (Many2one): Related company.

tax_ids (One2many): Tax replacement rules.

Relationships:

Many-to-one with res.company, res.partner

One-to-many with account.fiscal.position.tax

6. **res.partner (Partners)**

   Business Significance:

   Stores contact data for all third parties—customers, vendors, etc.

   Key Fields:

   name, is_company, street, city, country_id

   vat: VAT number

   customer_rank, supplier_rank

   property_account_receivable_id: Default AR account

   property_account_payable_id: Default AP account

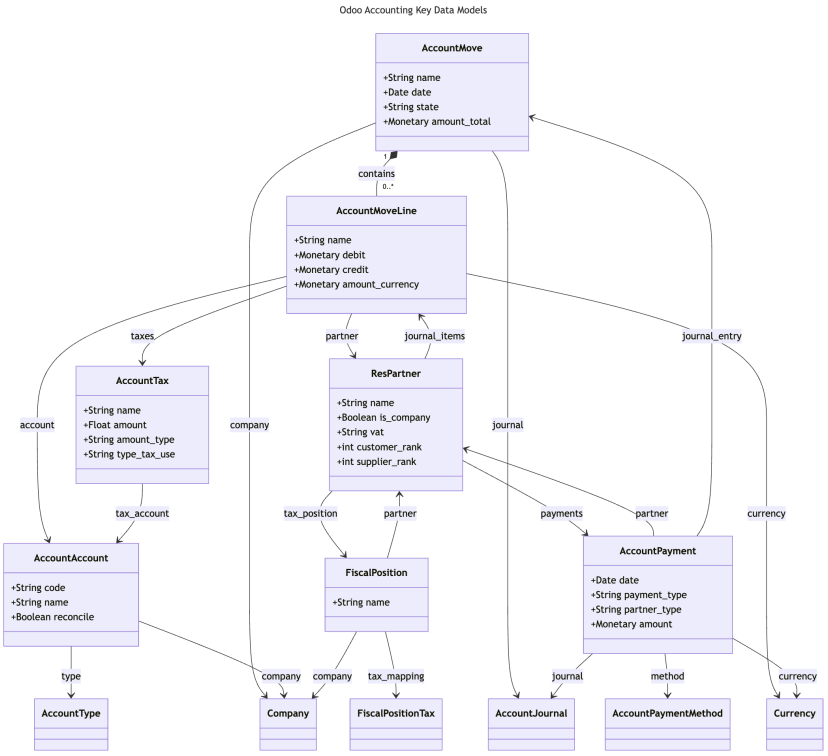
   property_payment_term_id, property_supplier_payment_term_id

   fiscal_position_id: Default tax rule mapping

   Relationships:

   One-to-many with account.move.line, account.payment

   Many-to-one with account.fiscal.position



3.2 Database Schema Overview

This chapter explains the top database tables in Odoo's Accounting module and the foreign key relationships and constraints that ensure data integrity across the system.

Core Tables

Table	Description
`account_move`	Contains journal entries, the basic unit of double-entry accounting.
`account_move_line`	Contains individual debit/credit lines of a journal entry.
`account_account`	Holds the Chart of Accounts — ledger accounts used for classifying transactions.
`account_journal`	Stores definitions of journals such as Sales Journal, Purchase Journal, etc.
`account_payment`	Stores payment postings such as customer receipts and vendor payments.
`account_tax`	Declares tax regulations (e.g., VAT, service tax) for financial postings.
`account_fiscal_position`	Maps fiscal rules for partners by location or type of business.
`res_partner`	Stores customer, vendor, and other third-party information.
`res_company`	Stores company data, allowing multi-company setups.
`account_account_type`	Categorizes accounts into types such as Income, Expenses, Payable, etc.

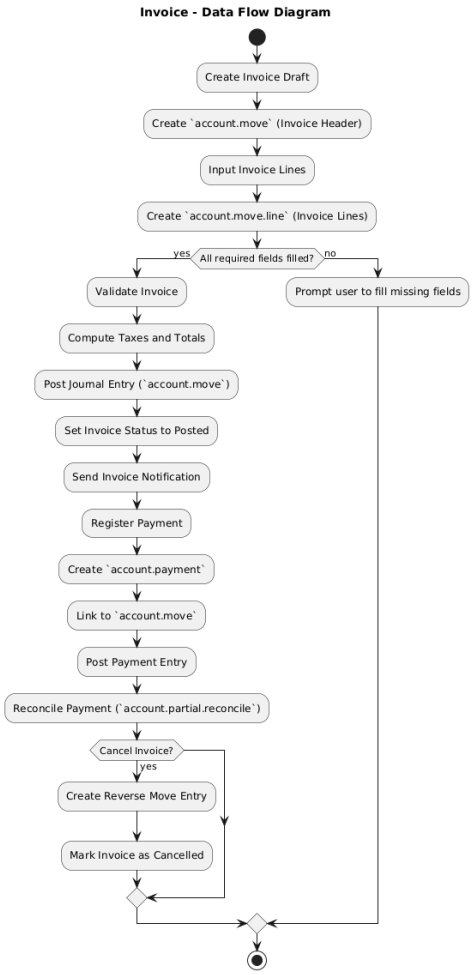
3.3 Data Flow Diagrams

Invoice

Purpose: Shows how an invoice is created, validated, paid, and optionally cancelled.

Steps:

1. Create Invoice Draft (account.move): An invoice draft is made.
2. Add Invoice Lines (account.move.line): Line items like products or services are added.
3. Validate Invoice: The invoice is posted, which makes accounting entries.
4. Register Payment (account.payment): A payment is recorded for the invoice.
5. Update Invoice Record: The invoice is set as paid.
6. Reconcile Payment (account.partial.reconcile): Debits and credits are matched.
7. Cancel Invoice (optional): Invoice is cancelled, reverting changes.

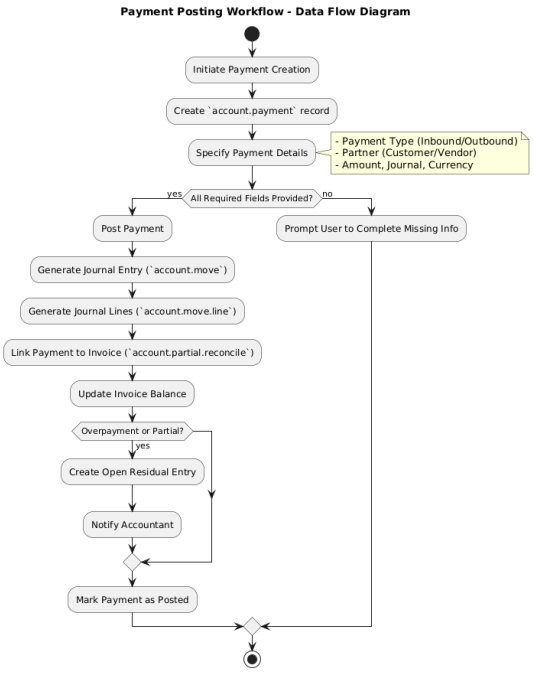


Payment

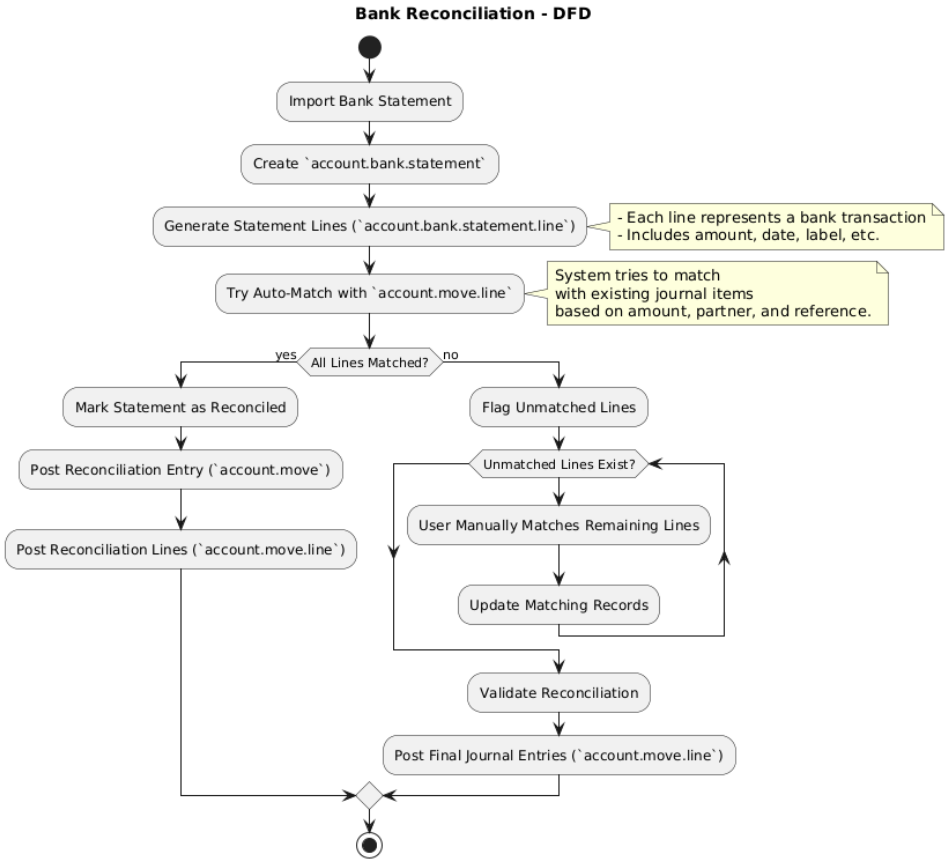
Purpose: Shows payments posted and processed to the general ledger.

Steps:

1. Create Payment (account.payment): Payment, either received or paid, is posted.
2. Specify Payment Details: Currency, partner, and method are defined.
3. Post Payment: Payment is checked.
4. Generate Journal Entry (account.move): Accounting records for transaction are generated.
5. Generate Lines (account.move.line): Debit and credit lines are posted.
6. Link to Invoice (account.partial.reconcile): Reconciles payment to invoice.



Bank Reconciliation



Purpose: Imagines matching bank statement lines to accounting records.

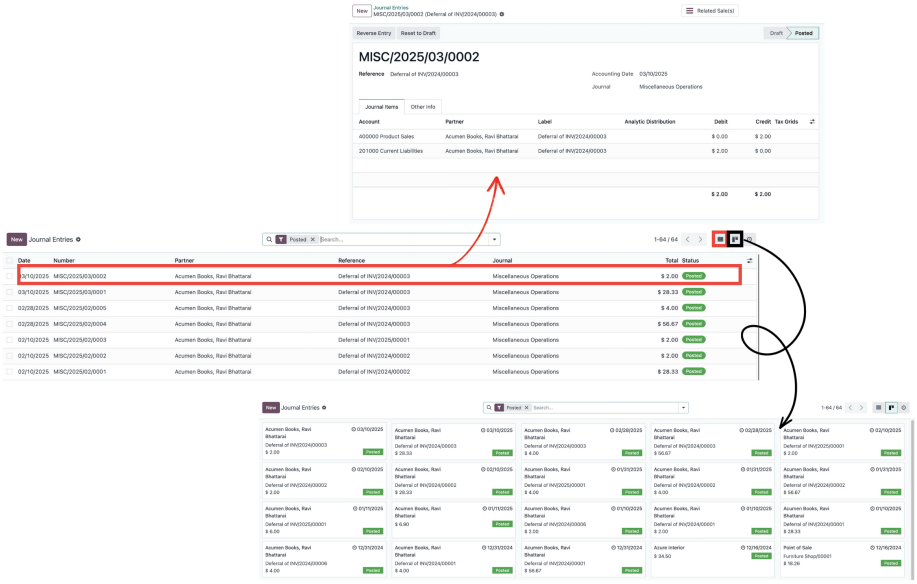
Steps:
-Import Bank Statement (account.bank.statement): Bank statements are imported.
-Create Statement Lines (account.bank.statement.line): Each transaction is split into lines.
-Match with Invoices (account.move.line): The system suggests matches with unpaid invoices.
-Validate Reconciliation (account.move): Reconciliation is validated.
-Post Journal Entries (account.move.line): Final accounting postings are posted if needed.

4 Odoo Features and Customization

4.1 Odoo UI & UX for Accounting

Odoo's user interface (UI) and user experience (UX) for the Accounting module are designed to be intuitive and efficient, providing users with easy access to financial data and tools. Here's an overview:

List/Form/Kanban Views for Financial Records:

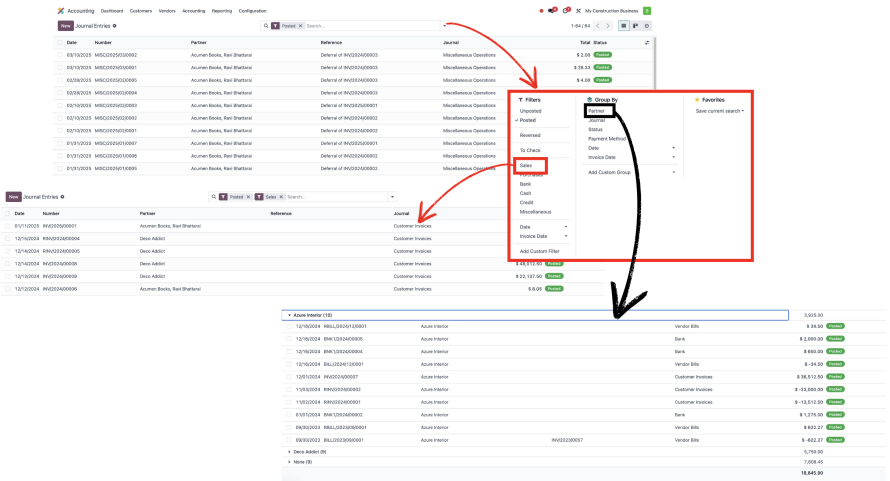


List View: Provides a tabular display of records, allowing users to quickly scan and sort data. This view is commonly used for journal entries, invoices, bills, and payments.

Form View: Displays detailed information for a single record, allowing users to edit and update data. This view is used for creating and modifying journal entries, invoices, bills, payments, and other financial records.

Kanban View: Presents records as cards in columns, allowing users to visually track the progress of financial processes. This view can be used for managing invoices, bills, and payments.

Filters and Groupings for Reports:



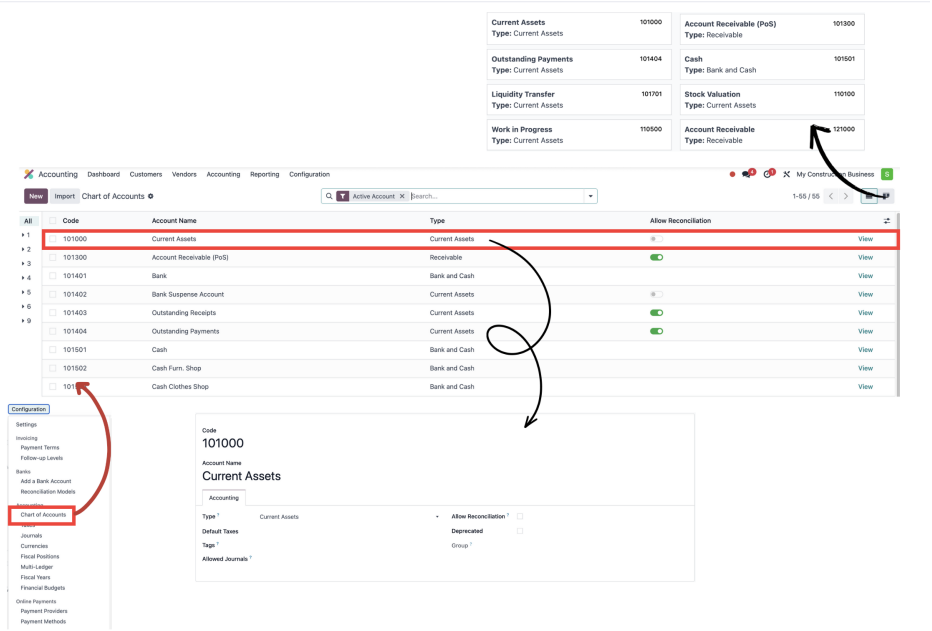
Filters: Allow users to narrow down the data displayed in reports based on specific criteria (e.g., date range, customer, product).

Groupings: Allow users to aggregate data in reports based on specific fields (e.g., by customer, by product, by month).

4.2 Configuration Options

The Odoo Accounting module comes with enormous configurations to enable the system to suit your requirements in your business and remain compliant with accounting standards and regulations. Take a look at the overview below:

Chart of Accounts:

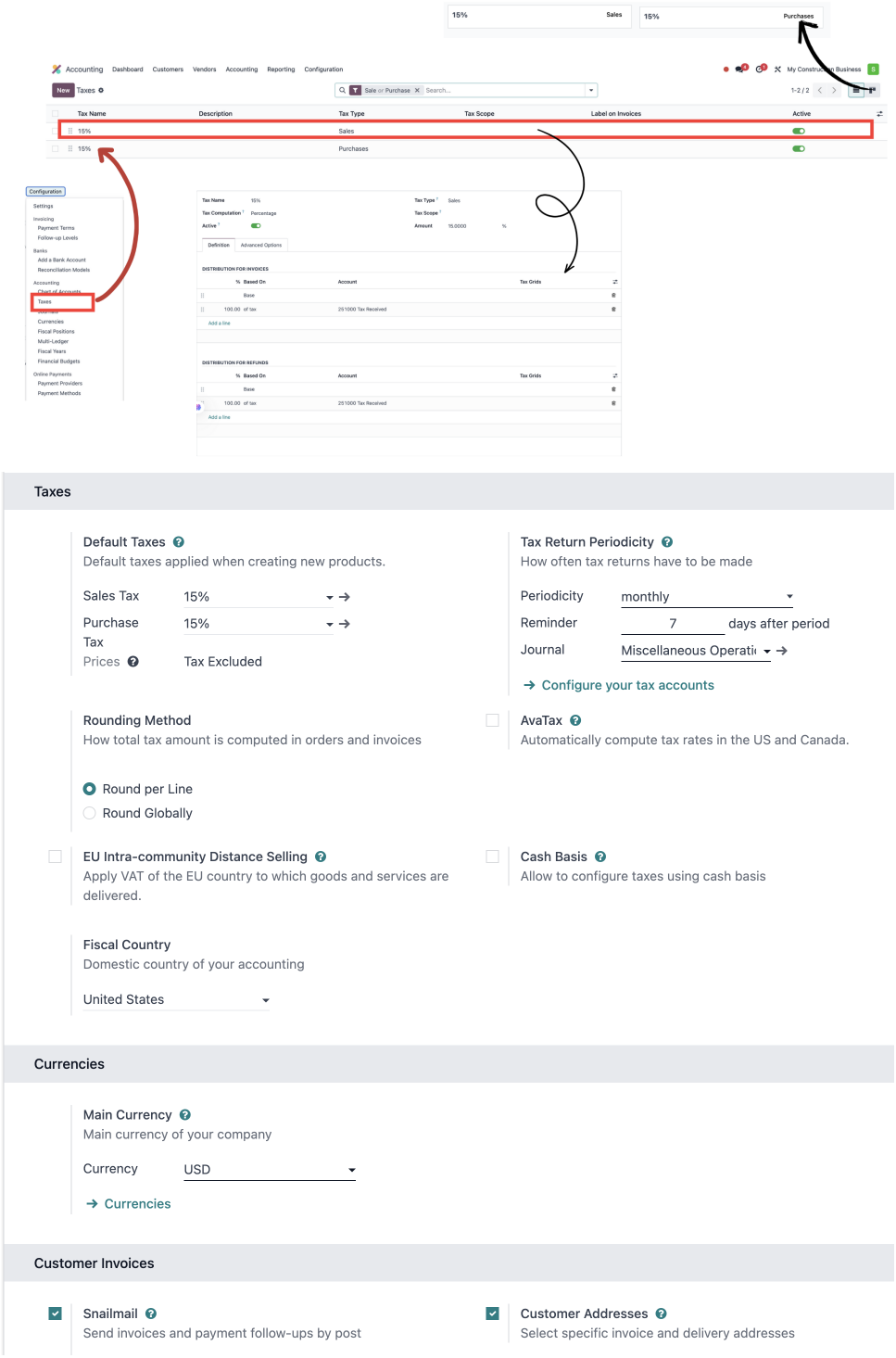


Configuration: Odoo allows you to create your chart of accounts so you can break down the account which you utilized in recording fiscal transactions. New accounts can be created, edit or alter existing accounts can be specified through account types.

Best Practices:

1. Start with a typical chart of accounts (e.g., US GAAP, IFRS).
2. Tailor the chart of accounts to your own business needs.
3. Assign each account with a classification and linked type of account.

Taxes and Fiscal Positions:

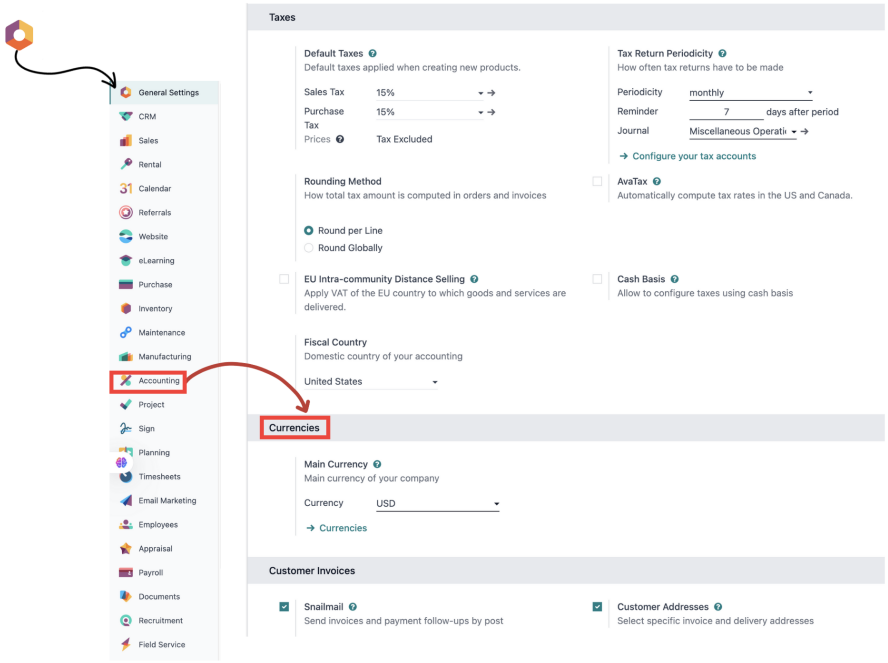


Configuration: Odoo allows you to set up taxes and fiscal positions, specifying the tax regulations for specific transactions based on the involved parties.

Best Practices:

1. Define all applicable taxes and tax rates.
2. Set fiscal positions for different customer and supplier locations.
3. Ensure tax rules are correctly configured to comply with tax regulations.

Multi-Currency and Exchange Rates:

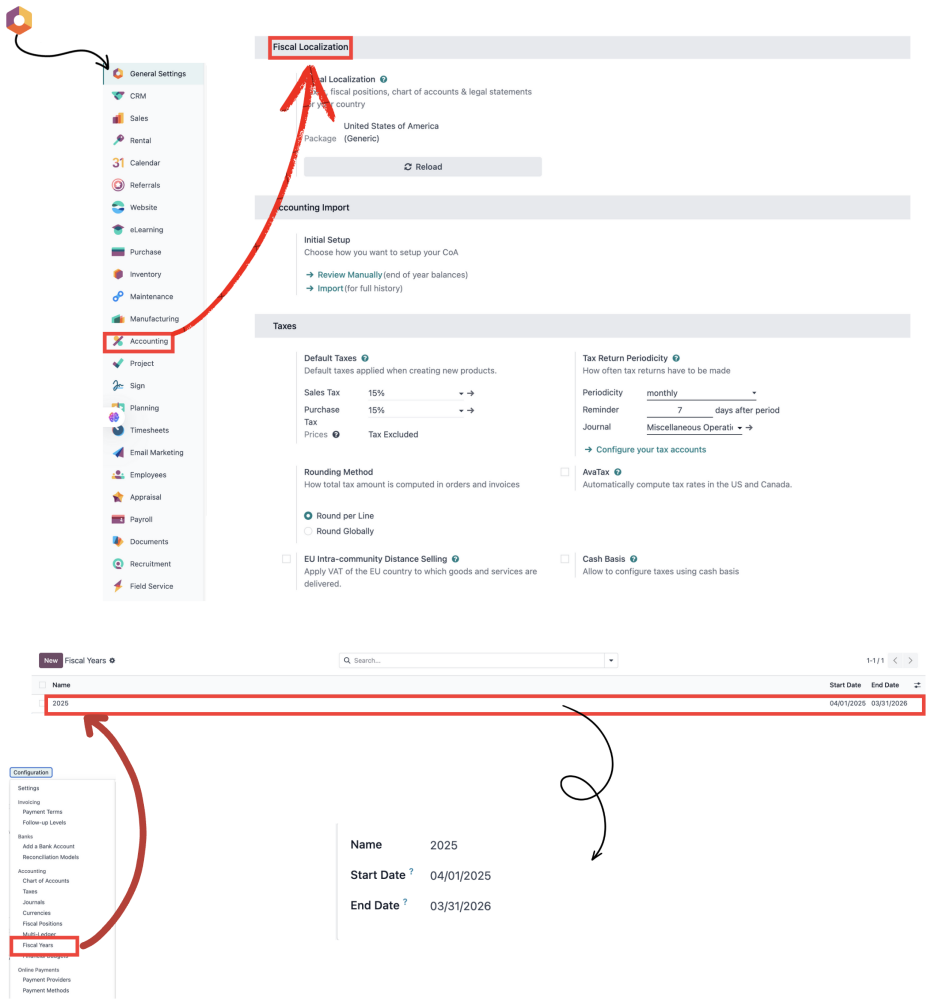


Configuration: Odoo supports multi-currency accounting, where you can conduct transactions in different currencies. You can configure exchange rates and automatically update exchange rates.

Best Practices:

1. Enable multi-currency functionality if you operate in different currencies.
2. Automatically update exchange rates to ensure proper currency conversions.
3. Manually check and update exchange rates at intervals.

Company Setup (Fiscal Years, Lock Dates):



Configuration: Odoo allows you to define fiscal years and lock dates, which define the years for which financial statements are to be prepared and the dates after which no further changes may be done to the accounting records.

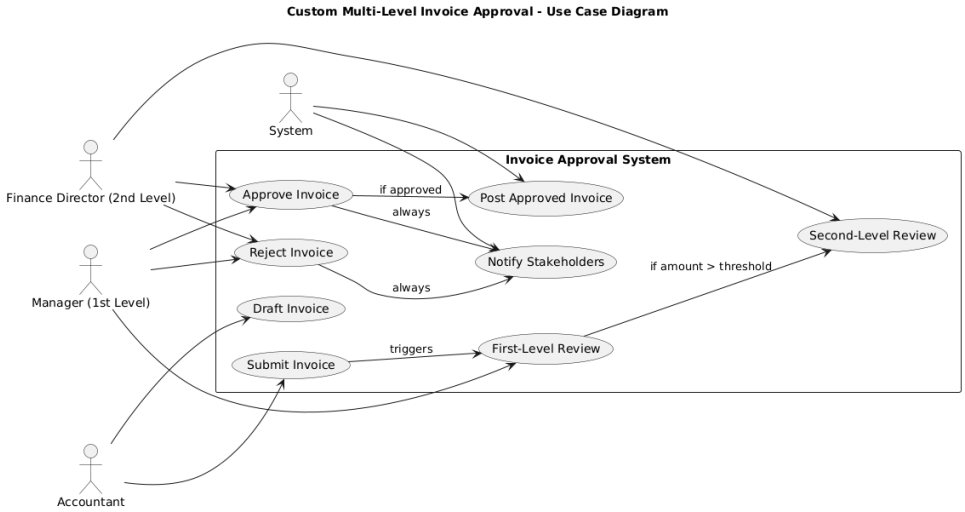
Best Practices:

1. Define fiscal years that correspond to your company cycle.
2. Set lock dates to prevent unauthorized changes to accounting records.
3. Close accounting periods on a regular schedule to offer accurate financial reporting.

4.3 Customization and Development

4.3.1 UML for customization

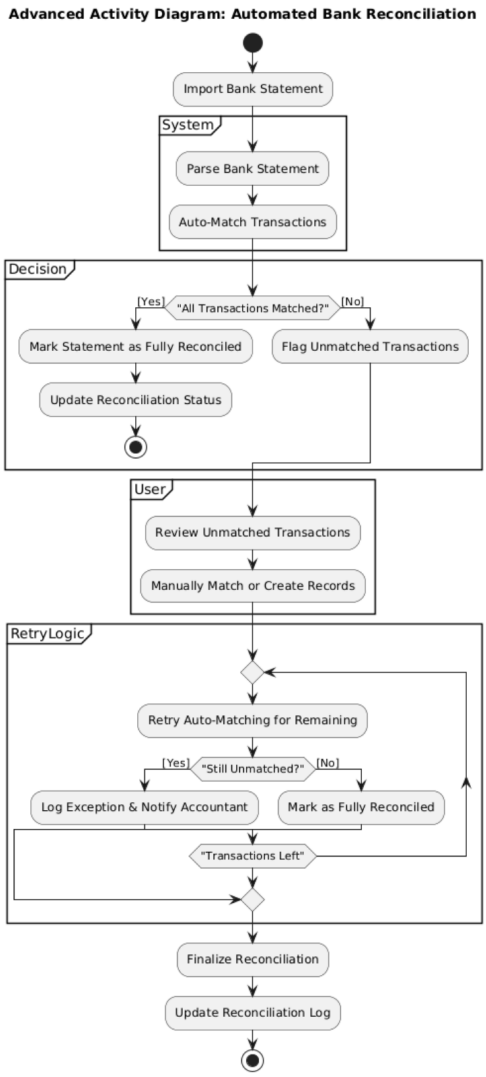
Use Case: Custom Invoice approval process



The Custom Multi-Level Invoice Approval workflow in Odoo extends the standard invoice workflow with formal roles and a multi-level review. It begins with the Accountant, where an invoice is prepared and submitted for approval. The invoice enters a First-Level Review by a Manager upon submission, ensuring that all the required information is accurate and complies with company policies. If the invoice amount is higher than a certain value or requires higher authorization, it goes to a Second-Level Review by the Finance Director. The Director and Manager can both approve or reject the invoice. Rejection must be with a reason, which is used by the System to notify the Accountant to take necessary action. On approval, the System automatically posts the invoice and makes the necessary adjustments to the accounting records.

This setup maximizes financial control, reduces the potential for error, and ensures high-value or sensitive transactions are correctly scrutinized. The workflow is nicely captured in a Use Case UML diagram that outlines the interaction of each actor with the system, approval flow, and escalation and system automation criteria. This is the ideal solution for organizations that need accountability, audit trails, and compliance within their invoicing process.

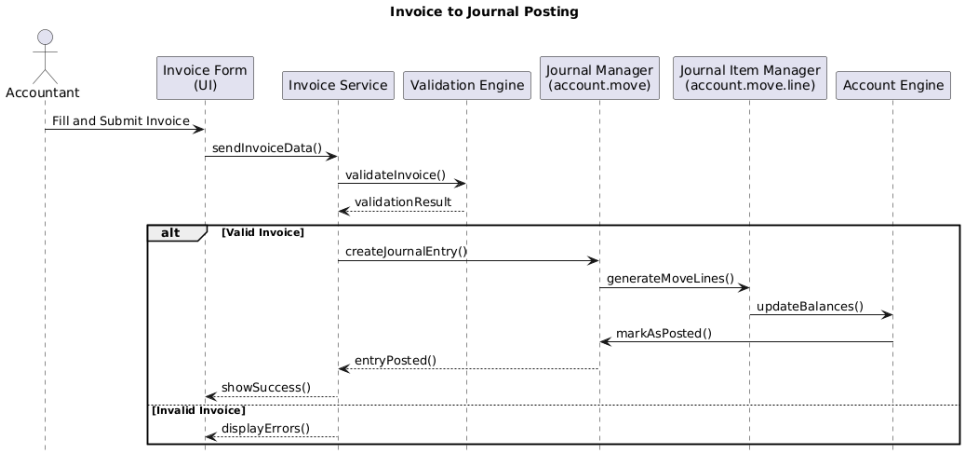
Activity Diagram: Automated bank reconciliation
The higher-level activity diagram for bank automatic reconciliation explains an elaborate process through which both users and the system collaborate to match bank statements appropriately and with maximum efficiency. The process initiates with importing a bank statement after which the system reads and attempts to automatically associate transactions with pre-existing accounting records. A decision node checks whether all the transactions are successfully matched. If they are, the system then proceeds to stamp the statement as fully reconciled and change its status, and it stops the process.



When not all the transactions match, the system stamps the unmatched ones, causing a manual intervention step. At this step, a user—in most cases, an accountant—scans the unmatched transactions, matches them by hand or generates new records where necessary. Following this, a retry logic loop is triggered where the system tries to auto-match any other unmatched items again. If the items continue to fail matching, the system raises an exception and notifies the accountant so that unsolved items can be traced properly and escalated accordingly.

It stops when reconciliation is finalized and the greatest number of permitted attempts at matching have been made. Finally, the system completes reconciling and refreshes the log of reconciliation to be utilized as an audit source. It is a sophisticated flow that enables greater data integrity, accountability and minimizes the scope of reconciliation error by finding a balance between auto-mation, human interaction and exception handling.

Sequence Diagram: Invoice to Journal Posting



This advanced sequence diagram illustrates the process of posting an invoice into the accounting system in Odoo. It starts when an Accountant posts an invoice from the user interface, which in turn calls the Invoice Service to check the data through the Validation Engine. If valid, a Journal Entry (account.move) is created, followed by the creation of corresponding Journal Items (account.move.line). These postings update the account balances through the Account Engine

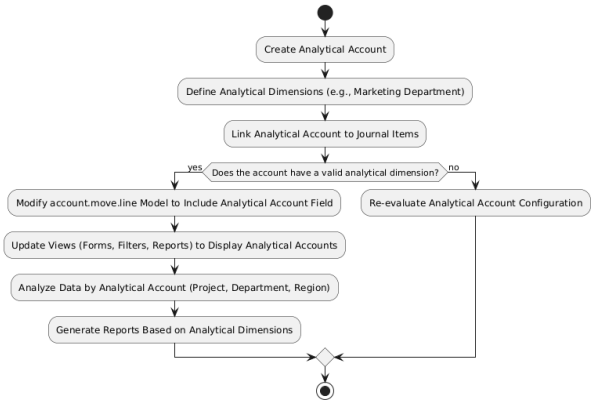
and mark the journal entry as posted. If an invoice fails to validate, the user is presented with accurate error messages. This modular sequence ensures data consistency, transaction integrity, and direct feedback throughout the accounting process.

4.3.2 Adding Analytical Dimensions

Odoo has good flexibility to extend the data structure of the Accounting module so that companies can monitor certain financial data and meet certain reporting or operation needs. Below are some common examples of these customizations:

1. **Adding Analytical Dimensions**

   Purpose: To give financial monitoring with the inclusion of dimensions such as department, project, or region—beyond the default chart of accounts.

   Implementation Steps:

   (a) Create Analytical Accounts: Utilize the Analytic Accounting functionality to create analytical accounts (e.g., M̈arketing Departmentör P̈roject A).

   (b) Extend Journal Items: Add a many2one field to the account.move.line model that connects journal lines with specific analytic accounts.

   (c) Customize Views: Modify form and list views to allow users to select and show analytic accounts.

   (d) Enable Analytical Reporting: Utilize Odoo's internal reporting capabilities (or custom reports) to filter, group, and report finance data by analytic dimension.

2. **Customizing Tax Fields or Journals**

   (a) **Custom Tax Fields Purpose:**

   Save additional information relating to tax reporting such as regulatory codes or reporting categories.

   Implementation Steps:

   i. Extend account.tax: Add custom fields (e.g., x_tax_code, x_reporting_category) to the account.tax model with Odoo Studio or custom modules.
   ii. Modify Views: Modify tax configuration views so that users can manage these new fields.
   iii. Adjust Tax Logic: Depending on the requirement, modify the tax calculation or report logic to incorporate new field values.

   (b) **Custom Journals Purpose:**

   To divide and process specific types of financial entries, such as payroll or grants.

   Implementation Steps:

i. Create a Journal: Navigate to Accounting > Configuration > Journals and create a new journal (e.g., "Payroll Journal").
ii. Define Journal Type and Sequence: Establish the journal type (e.g., General, Sale, Purchase) and specify an entry sequence for sequential numbering.
iii. Set up Journal Behavior: Establish default accounts, tax setups, and analytic setups as required to minimize data entry and enhance reporting.

4.3.3 Odoo ORM and Python

Odoo's Object-Relational Mapping (ORM) and Python give limitless flexibility to add and tailor the business logic of the Accounting module. Using these tools, you can develop custom behavior, automate tasks, and tailor procedures according to your business needs. Some instances of utilizing these tools for sophisticated customizations are described below:

Auto-Posting of Entries Based on Rules Purpose: Automate the automatic generation and posting of journal entries in Odoo, according to some predefined business rules or conditions.

1. **Create a Python Method for Auto-Posting**

   A Python method will be created in the respective Odoo model (typically, the `account.move` or `account.move.line` model) to define the rules under which entries are to be posted automatically.

   (a) **Specify Conditions:** The method will set conditions such as specific types of journals, transaction value thresholds, or posting dates to determine when entries should be posted automatically.

   (b) **Set up Journal Entries:** If conditions are met, the method will create new `account.move` and `account.move.line` records based on predefined configurations.

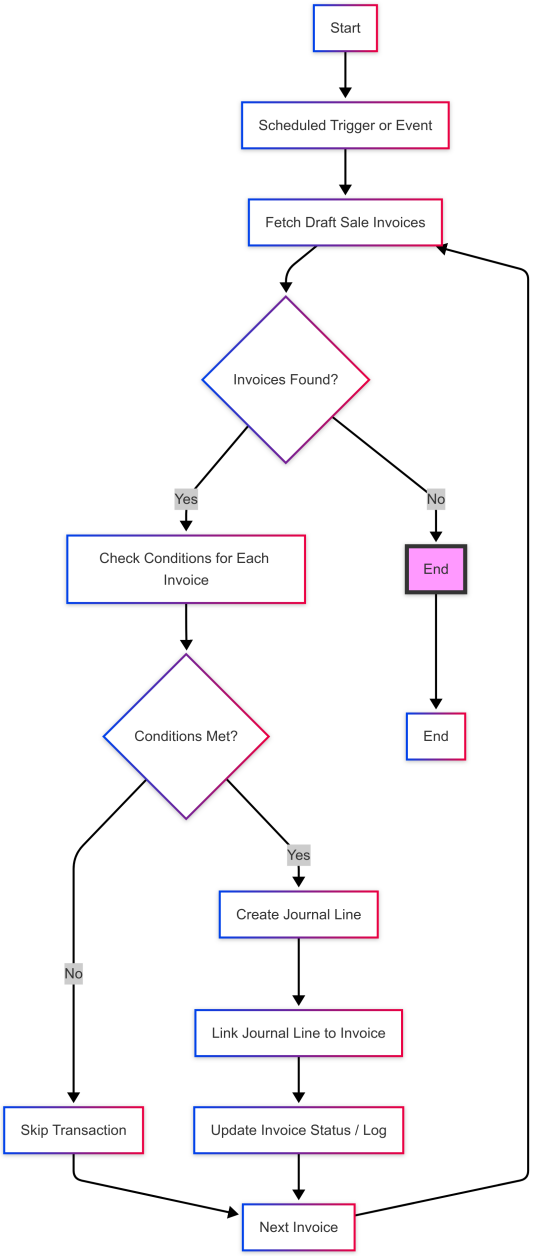
   (c) **Post Journal Entries:** Finally, the method will automatically post the entries, ensuring the accounting records are up to date.

2. **Automate the Python Method:**

   Once the Python method has been written, it must be automated through a server action or a scheduled action.

   (a) A scheduled action may call the method at a specific interval (e.g., daily or monthly).

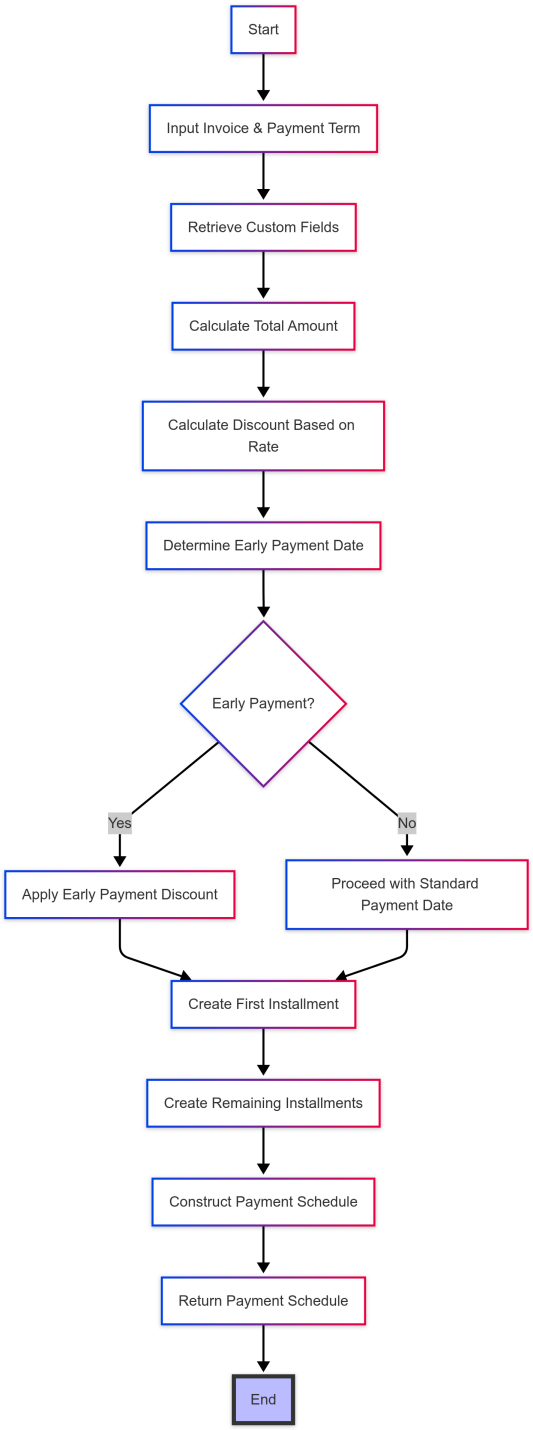
   (b) A server action can execute the method based on specific events, such as when a new invoice or sale order is created.



Custom Logic to Calculate Payment Terms
Purpose: Use custom logic to determine due dates and payment amounts based on special payment terms. This proves useful when special payment schemes are present in business, such as offering a discount for early settlement or special payment arrangements.

Implementation Steps:

1. Extend the account.payment.term Model: The account.payment.term model, which governs payment terms, will be updated to hold custom fields that enable the storage of unique rules governing the payment term, such as:
   (a) Discount rate
   (b) Early payment discount
   (c) Custom payment frequency or amounts
2. Override the Compute Method: The compute method of the account.payment.term model is overridden to include custom logic in computing due dates and payment amounts from the unique rules defined in the new fields.
3. Return Custom Payment Terms: The overridden method will give a list of tuples consisting of a due date and the amount due for each period.



4.3.4 Customization of Views and Actions

Odoo's highly modular architecture allows for extensive customization of views and actions to enhance usability and adapt to specific business requirements. This chapter presents two practical illustrations of view customization in the Accounting module: enhancing invoice forms

and designing tailored list views of aged receivables.

1. Enhancing Invoice Views by Adding Custom Fields **Objective:**

   To improve invoice forms to display additional information such as custom business fields or related model data, improving data visibility and productivity.

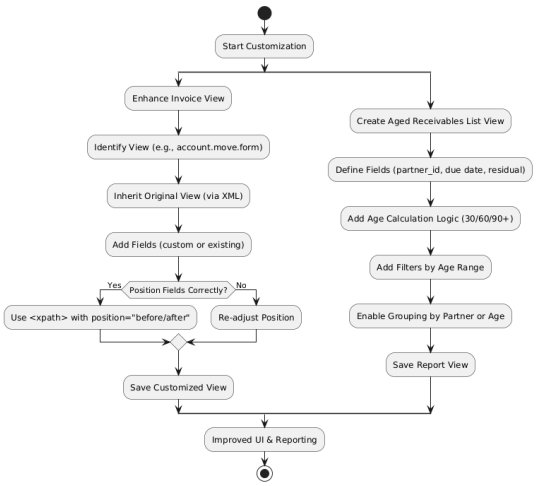
   Implementation Steps:

   (a) Identify the Target View: Recognize the view to be customized (e.g., account.move form view for invoices).

   (b) Inherit the View: Use XML to inherit the base view by using the inherit_id tag. This avoids overwriting base views and maintains upgrade compatibility.

   (c) Add Custom Fields: Insert new fields from existing fields in the account.move model or add new fields (e.g., x_branch_id, x_customer_ref).

   (d) Place Fields Strategically: Use Odoo's XML positioning attributes (before, after, inside) to place new fields in a convenient position.



   Result:

   Users can see additional invoice details without having to navigate away from the page, supporting enhanced decision-making and efficient workflow.

2. Custom List Views for Aged Receivables

   Objective:

   To provide a list view that presents overdue customer invoices segmented by age, to facilitate quicker collection follow-up and cash flow visibility.

   Implementation Steps:

   (a) Create a New List View: Specify a new list view (tree type) for the account.move model, filtered on customer invoices.

   (b) Add Key Fields:

      i. partner_id: Customer name
      ii. invoice_date_due: Payment due date
      iii. residual: Outstanding amount
      iv. age: A computed field reflecting how overdue the invoice is (e.g., 0–30, 31–60, 61–90, 90+ days)

   (c) Add Domain Filters:

   Support filtering by invoice age buckets with filter elements in XML or custom search screens.

   (d) Support Grouping:

   Support grouping by customer (partner_id) or age buckets to quickly analyze customer debt composition.

Result:

Accounting users are provided with a tailored UI to monitor past-due receivables, prioritize collections, and export structured data for reporting.

4.3.5 Reporting Customization

Odoo has a very flexible reporting engine that may be customized to generate specialized financial reports to serve specific business requirements. The customizations may either be done over standard financial reports or completely customized report templates.

1. **Changing Standard Profit & Loss and Balance Sheet Reports Purpose:**

   To alter the organization, content, and logic of generic financial reports (e.g., the Profit & Loss or Balance Sheet) to suit business-specific metrics and classifications.

   Implementation Steps:

   (a) Find the Report to Change: Identify the target report, for instance:
      i. account.account_financial_report_profitandloss
      ii. account.account_financial_report_balancesheet

   (b) Inherit the Report Template: Create a custom report by inheriting the underlying financial report XML or Python code.

   (c) Change Report Lines:
      i. Add new report lines to show additional financial information.
      ii. Adjust existing lines to alter calculations, formulas, or display settings.
      iii. Remove unnecessary lines that do not apply to the company reporting needs.

   (d) Change Report Options: Modify filters such as:
      i. Date ranges
      ii. Comparison periods (e.g., year-over-year, month-over-month comparisons)
      iii. Analytic filters on projects, departments, or cost centers

2. **Creating Custom QWeb or Excel Reports**

   (a) QWeb-Based Reports

   Purpose:

   To produce custom PDF reports with branded and properly formatted output through Odoo's templating engine.

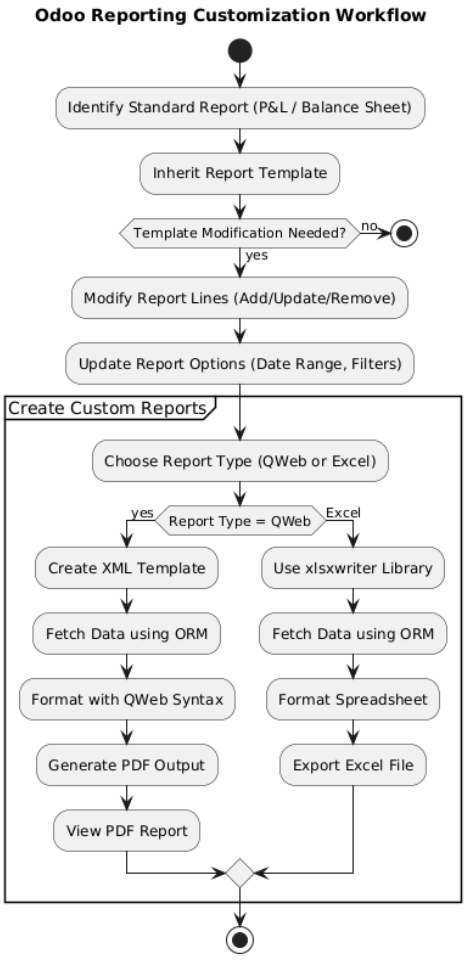
   Implementation Steps:

   i. Create a new QWeb template by XML.

   ii. Use Odoo ORM within a matching Python report class in order to pull necessary financial data.

   iii. Display and format the data through QWeb syntax (loops, conditions, formatting).

   iv. Bind the report to an action to enable users to create using the UI.



   (b) Excel-Based Reports (XLSX)

   Purpose:

   Create spreadsheet reports, which may further be edited by users, exchanged, or uploaded into third-party systems.

   Implementation Steps:

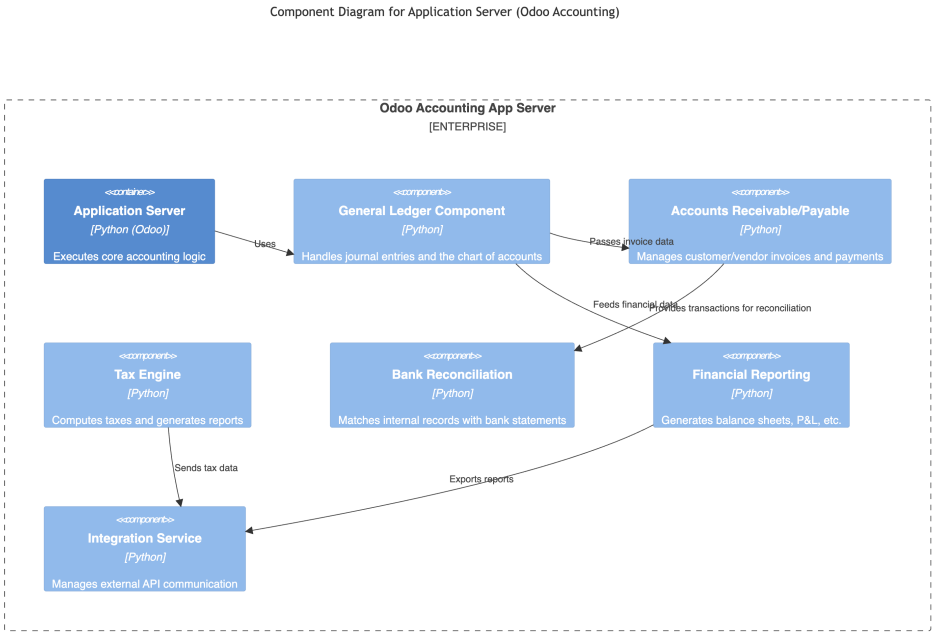
   (a) Make use of the xlsxwriter Python library for generating a dynamic Excel document.

   (b) Retrieve needed fiscal records via Odoo ORM practices (e.g., journal entry, balances).

   (c) Render headers, insert formulas, and insert rows of data by utilizing functions through xlsxwriter.

   (d) Give the developed Excel file as a downloadable attachment or in a report that is scheduled.

4.4 C4 Model - Component Diagram (C3 - Accounting Internals)



This component diagram provides a high-level description of the Odoo Accounting Module, breaking it down into its primary functional components and how they interact with each other. Each component embodies a specific set of duties that collectively enable end-to-end financial management within the system.

1. **Invoice Management**

   Description: Manages the creation, modification, and life cycle of customer and vendor invoices.

   Key Responsibilities:

   (a) Create, update, and delete invoice records.
   (b) Create and export invoice documents (PDFs).
   (c) Control invoice statuses (e.g., Draft, Posted, Paid, Cancelled).

   Interfaces:

   (a) Create and fetch invoice data.
   (b) Update invoice state and related fields.

   Dependencies:

   (a) Tax Engine – for calculation of tax in real-time.
   (b) Payment Handling – for linking payments and for updation of invoice payment status.

2. **Payment Handling**

   Description: Responsible for handling the processing and recording of payments that come in and go out.

   Key Responsibilities:

   (a) Record customer payments and vendor disbursements.

(b) Link payments with related invoices or bills.

(c) Generate payment receipts and reconcile books of accounts.

Interfaces:

(a) Register and read payment information.

(b) Associate payments with accounting documents.

Dependencies:

(a) Invoice Management – to account for payments.

(b) Reconciliation Engine – to reconcile payments with bank accounts.

3. **Reconciliation Engine**

Description: Reconciles bank statement line with internal accounting postings.

Key Responsibilities:

(a) Import and interpret bank statements.

(b) Automatically match transactions according to configurable rules.

(c) Offer manual and auto-recommended reconciliations.

Interfaces:

(a) Import bank information and define reconciliation logic.

(b) Review and approve reconciliations.

Dependencies:

(a) Payment Handling – to read payment records.

(b) Reporting & Analytics – to enable reconciliation reporting and auditing.

4. **Reporting & Analytics**

Description: Generates financial and analytical reports to external and internal stakeholders.

Key Responsibilities:

(a) Prepare primary financial statements (e.g., P&L, Balance Sheet).

(b) Provide dynamic dashboards and drill-down.

(c) Generate customized financial reports.

Interfaces:

(a) Configure report parameters and layouts.

(b) Export data to PDF, Excel, or web views.

Dependencies:

(a) Dependent on all other components for data aggregation and metrics.

5. **Tax Engine**

Description: Verifies compliance with local tax legislation and calculates taxes.

Key Responsibilities:

(a) Calculate recoverable taxes upon invoice or bill processing.

(b) Schedule tax codes, tax rates, and fiscal postings.

(c) Prepare statutory tax reports.

Interfaces:

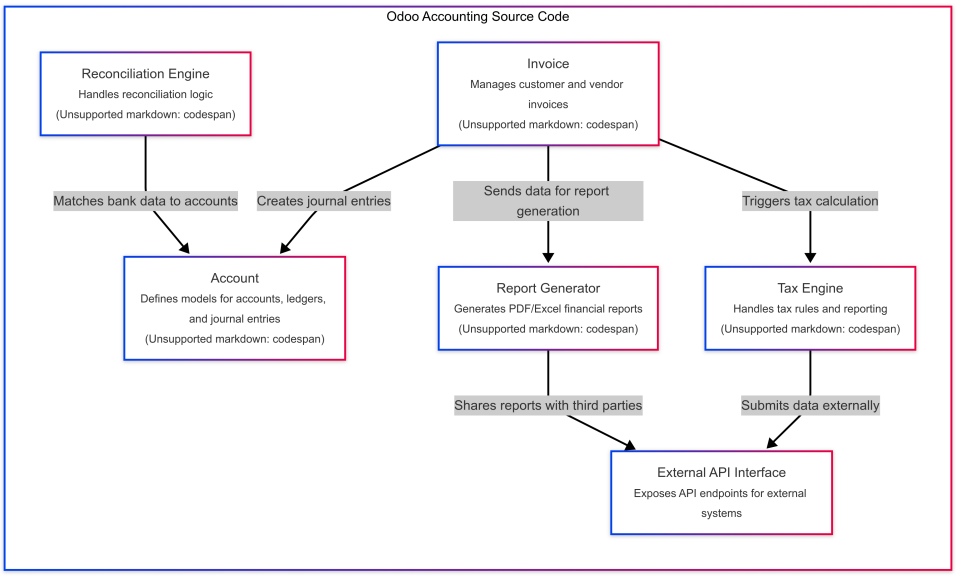
(a) Develop and implement tax regulations.

(b) Generate reports (e.g., VAT, GST summaries).

Dependencies:

(a) Invoice Management – in aid of calculation of tax on transactions.

4.5 C4 Model - Code Diagram (C4 - Illustrative Examples)

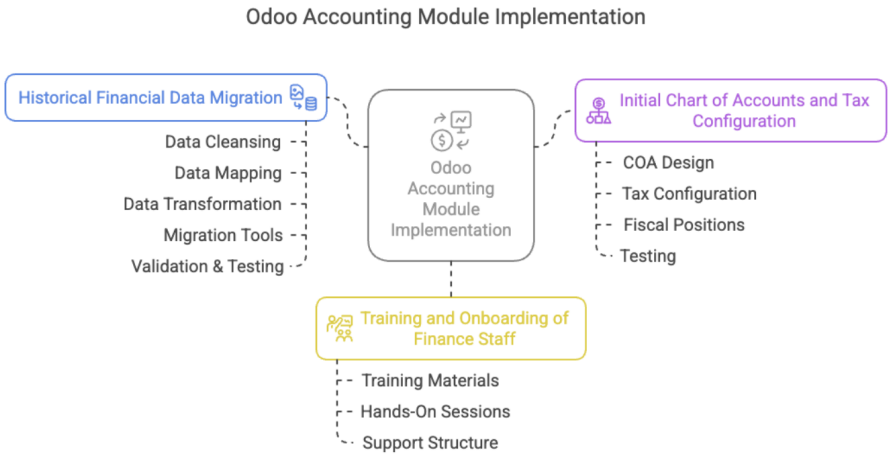


This diagram illustrates the core components of the Odoo Accounting source code and their interactions. The Invoice component manages customer and vendor invoices, creating journal entries in the Account module, which defines models for accounts, ledgers, and journal entries. The Reconciliation Engine handles matching bank data to existing account records to ensure accurate financial reconciliation. The Report Generator component is responsible for producing financial reports in PDF or Excel formats and shares these reports with third parties. The Tax Engine manages tax calculations and rules, triggering tax computations during invoice processing and submitting relevant tax data externally. All report sharing and tax submission activities interact with the External API Interface, which exposes API endpoints to enable third-party systems to access reports and tax-related data. Overall, these components work together to facilitate robust, automated accounting processes within Odoo.

5 Deployment

5.1 Deployment Considerations

Implementation of the Odoo Accounting module requires careful planning and strategic deployment to prevent a disruption, comply with finance standards, and smooth adoption by users. Some key areas to explore are outlined below:



1. **Historical Financial Data Migration**

   Goal: Ensure continuity of financial reporting and an auditable complete financial history in Odoo.

   Key Considerations:

   (a) Data Cleansing: Clean and verify current legacy financial records for the elimination of duplicates, errors, or inconsistencies.

   (b) Data Mapping: Map your existing data fields onto Odoo's data model (e.g., accounts, partners, transactions).

   (c) Data Transformation: Convert legacy forms into Odoo-compatible formats.

   (d) Migration Tools: Use Odoo's native import features or third-party tools (e.g., ETL tools or scripts).

   (e) Validation & Testing: Verify the integrity and accuracy of imported data through extensive reconciliation and test cases.

   Best Practices:

   (a) Run a pilot migration using a small data set.

   (b) Involve finance personnel early on for validation and feedback.

   (c) Data Transformation: Convert legacy forms into Odoo-compatible formats.

   (d) Document the entire process for transparency and future audits.

2. **Initial Chart of Accounts (COA) and Tax Configuration**

   Goal: Establish a solid financial foundation that is compliant, reduces reporting, and supports business-specific workflows.

   Key Considerations:

(a) COA Design: Create a rational and scalable chart of accounts based on accounting standards and reporting needs.

(b) Tax Configuration: Set up all relevant taxes, like sales tax, VAT, GST, etc., with the correct percentages and rules of usage.

(c) Fiscal Positions: Set up fiscal positions to handle regional tax variations by customer/vendor locations or exemptions.

(d) Testing: Test common transactions (e.g., invoices, vendor bills) to check tax calculations and account mappings.

Best Practices:

(a) Use a standard COA template as a base and then adapt it to your business.

(b) Test configurations with a certified accountant.

(c) Document your COA design and tax logic for regulatory purposes.

3. **Training and Onboarding of Finance Staff**

Goal: Get the finance team ready to learn and utilize the Odoo Accounting module efficiently.

Key Considerations:

(a) Training Materials: Provide step-by-step guides, quick reference materials, and video tutorials that are workflow-specific.

(b) Hands-On Sessions: Offer hands-on training in real-life scenarios to support learning.

(c) Support Structure: Set up support structures for maintenance support (e.g., internal helpdesk, FAQs, power users).

Best Practices:

(a) Start with fundamental concepts and build up to advanced features (e.g., reconciliation rules, financial reports).

(b) Offer training in multiple formats (live sessions, recorded webinars, interactive documentation).

(c) Encourage open communication and feedback to continually improve the onboarding experience.

6 Maintenance

6.1 Maintenance and Updates

Routine upkeep and updates of the Odoo Accounting module are necessary to ensure proper functionality, security, regulatory compliance, and responding to evolving business requirements. Below are some of the key areas that should be addressed:

1. **COA upgrade**

   Purpose: To support restructurings within the organization, future expansion, or to meet updated accounting standards.

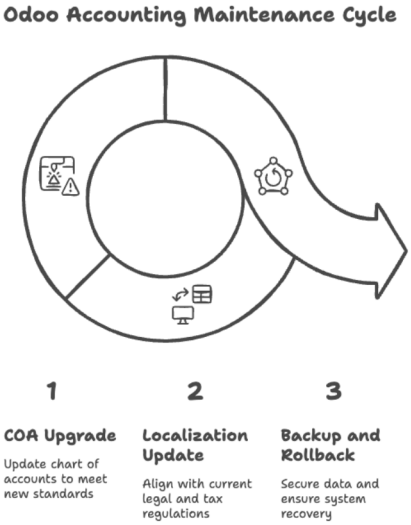
   Key Considerations:

   (a) Strategic Planning: Plan for how the COA updates will affect historical data, reporting, and integrations.

   (b) Mapping Old to New COA: Create a clean mapping of the old accounts to the new structure to ensure consistency.

   (c) Data Migration: Utilize data migration scripts or tools to realign the historical entries with the new COA.

   (d) Testing & Validation: Perform extensive tests to ensure reports, journal entries, and account balances are correct.



   Best Practices:

   (a) Upgrade first in a staging/test environment.

   (b) Engage your accounting team during the upgrade process for review and validation.

   (c) Maintain complete records of all updates, including before-and-after mapping and impact analysis.

2. **Localization and Compliance Update Management**

   Purpose: Having the system aligned with most recent legal, tax, and financial regulations of each operating country.

   Key Considerations:

   (a) Regulatory Monitoring: Monitor changes in national or international accounting regulations (e.g., VAT changes, reporting obligations) constantly.

   (b) Applying Localization Updates: Install Odoo localization modules and patches relevant to your country from time to time (e.g., electronic invoice, SAF-T, GST).

   (c) Testing Post-Updates: Validate the installed updates are functional and do not interfere with running workflows or reporting.

   Best Practices:

(a) Subscribe to Odoo enterprise updates or certified localization partners.

(b) Utilize a local certified accountant to validate all compliance updates.

(c) Log each compliance update and its effect on reporting, taxation, and workflow.

3. **Backup and Rollback Strategies**

Purpose: To safeguard critical financial data and system recovery in case of failures, data loss, or corruption.

Key Considerations:

(a) Backup Schedule: Have a regular schedule (e.g., every day, weekly) based on volume and urgency of transactions.

(b) Secure Storage Location: Store backups securely and redundantly (e.g.,encrypted cloud, offsite servers).

(c) Rollback Process: Set up and follow explicit rollback processes to restore backups when something goes wrong during emergencies or after failed updates.

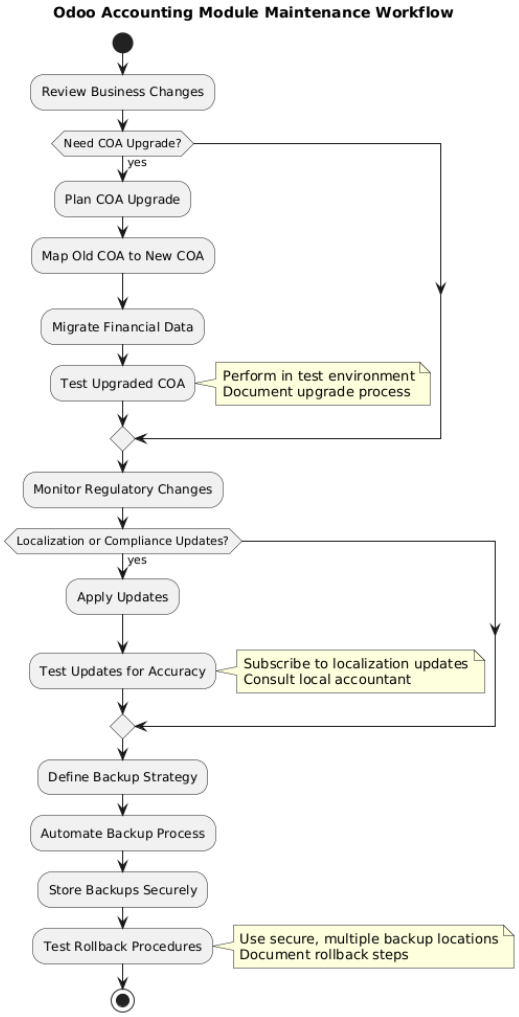
(d) Backup Verification: Periodically test the backup restoration process to guarantee the integrity of the backup and rollback.

Best Practices:

(a) Make backup procedures automated with scripts or Odoo-compatible utilities.

(b) Include versioning and snapshot backups for quick recovery alternatives.

(c) Have at least one offsite backup and detail the entire disaster recovery process.



7 Appendices

7.1 Glossary of Terms

Term	Definition
Account	A record in the general ledger that is used to sort and store financial transactions (e.g., Cash, Sales, Expense).
Chart of Accounts (COA)	A structured list of all the accounts used by an organization to classify and record financial transactions.
Fiscal Position	A configuration in Odoo that determines which taxes and accounts are applied based on a customer's or vendor's country, company type, or location.
Journal Entry (Move)	A complete record of a financial transaction in the general ledger. In Odoo, this is called a "Move."
Move Line	An individual line within a journal entry that represents a debit or credit to an account.
Journal	A record that groups financial entries of a specific type (e.g., Sales Journal, Purchase Journal, Bank Journal).
Reconciliation	The process of matching and validating transactions in the bank statement against the accounting entries.
Analytic Account	A secondary axis of financial analysis used to group transactions for projects, departments, or cost centers.
Partner	A contact record in Odoo, which can be a customer, vendor, or both.
Invoice	A commercial document issued by a seller to a buyer indicating the products, quantities, and agreed prices.
Bill	An invoice received from a supplier, which records the company's payables.
Credit Note	A document issued to reduce the amount owed on a previously issued invoice.
Debit Note	A document used to increase the amount payable on a bill, often due to errors or additional charges.
Payment Term	A set of rules that determine how and when an invoice should be paid, including possible early-payment discounts.
Tax	A financial charge calculated on a transaction, managed through tax rules and tax groups in Odoo.
Tax Report	A summary of tax collected and paid, typically used for VAT or GST filing.
Bank Statement	A record of transactions from a bank account, which can be imported or manually created in Odoo.
Asset	A long-term resource owned by a company, managed through Odoo's Asset Management for depreciation tracking.
Deferred Revenue / Expense	Revenues or expenses that are recognized in future periods and managed via deferral journals.
Lock Date	A date before which no changes can be made to accounting records, used to close financial periods.
Audit Trail	The log of changes and transactions recorded for traceability and compliance.
Cash Basis Accounting	An accounting method where revenues and expenses are recorded when cash is received or paid.

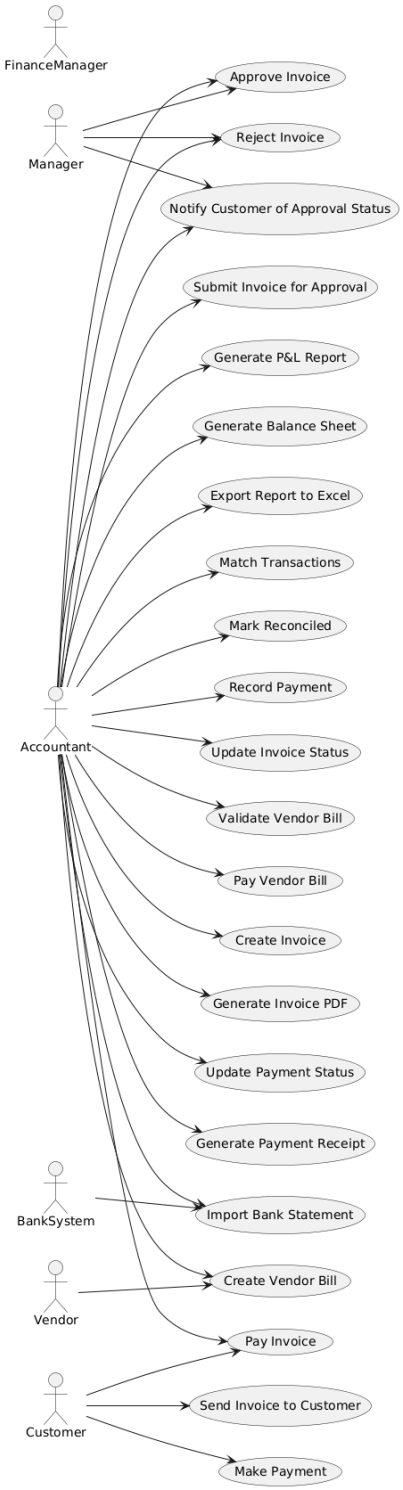
Accrual Accounting	An accounting method where transactions are recorded when they occur, regardless of cash movement.
Multi-Currency	The ability to handle transactions, journals, and reporting in different currencies.
Opening Balance	The amount of money in an account at the beginning of a new fiscal period.
Closing Entry	An entry made at the end of an accounting period to close temporary accounts (e.g., revenues, expenses).
Budget	A financial plan that estimates revenue and expenses for a future period, which can be tracked via Odoo's budgeting feature.
Consolidation	The process of combining financial data from multiple companies/entities into one unified report.

7.2 Acronyms and Abbreviations

Acronym	Definition
AP	Accounts Payable – Money a business owes to suppliers.
AR	Accounts Receivable – Money owed to a business by customers.
BS	Balance Sheet – A financial statement showing assets, liabilities, and equity.
COA	Chart of Accounts – Structured list of accounts used to categorize financial transactions.
CFO	Chief Financial Officer – The senior executive responsible for managing the financial actions of a company.
CPA	Certified Public Accountant – A qualified accountant licensed by a governing body.
CR	Credit – An accounting entry that either increases a liability or equity account or decreases an asset or expense account.
DB	Debit – An accounting entry that increases an asset or expense account or decreases a liability or equity account.
DFA	Depreciation, Fixed Assets – Related to the depreciation and management of fixed assets.
ERP	Enterprise Resource Planning – Integrated software used to manage business processes.
FY	Fiscal Year – A one-year period used for financial reporting.
GAAP	Generally Accepted Accounting Principles – Standard framework of guidelines for financial accounting.
GL	General Ledger – The primary accounting record summarizing all transactions.
IFRS	International Financial Reporting Standards – Global accounting standards.
INV	Inventory – The goods and materials a business holds for sale.
IS	Income Statement – Also known as Profit & Loss Statement.
JE	Journal Entry – A record of a transaction in the accounting books.
KPI	Key Performance Indicator – Metrics that evaluate the success of an activity.
P&L	Profit and Loss Statement – A financial report summarizing revenues and expenses.
POS	Point of Sale – System where sales transactions are completed.
ROI	Return on Investment – A measure of profitability.
TAX/VAT	Value Added Tax – A consumption tax added to the sale of goods and services.
UoM	Unit of Measure – Measurement unit for quantities in transactions.

WHT	Withholding Tax – Tax deducted at source on payments like salaries or dividends.
API	TApplication Programming Interface – Interface for communication between applications.
CSV	Comma-Separated Values – Format used to import/export tabular data.
DB	Database – Structured collection of data.
ORM	Object-Relational Mapping – Converts database data to object-oriented code.
QWeb	Odoo's templating engine for reports and web views.
SAAS	Software as a Service – Software delivery model via the internet.
UI	User Interface – Visual part of the application.
UX	User Experience – Overall user interaction quality.
UML	Unified Modeling Language – Visual modeling language for systems.
OCA	Odoo Community Association – Organization supporting Odoo open-source development.
Odoo.sh	Odoo's managed hosting platform for deployment.

7.3 Sample UML Diagrams



Use Case Diagram This is a UML activity diagram illustrating the process flow with regard to handling invoices, payments, and communication with vendors in terms of different roles:

Finance Manager, Manager, Accountant, Bank System, Vendor, and Customer.

Roles and Activities:
Finance Manager: Approve or decline invoice.
-Approve Invoice
-Reject Invoice

Manager: Inform customer of approval status, request approval of invoice.
-Inform Customer of Approval Status
-Request Approval of Invoice

Accountant: Performs multiple tasks with regard to handling invoices, financial reports, and payments.
-Generate P&L Report
-Generate Balance Sheet
-Export Report to Excel
-Match Transactions
-Mark Reconciled
-Record Payment
-Update Invoice Status
-Validate Vendor Bill
-Pay Vendor Bill
-Create Invoice
-Generate Invoice PDF
-Update Payment Status
-Generate Payment Receipt
-Send Invoice to Customer

Bank System: Handles bank statements and vendor bill payments. -Import Bank Statement
-Create Vendor Bill
-Pay Invoice

Vendor: Prepares vendor bills and pays invoices.
-Create Vendor Bill
-Pay Invoice

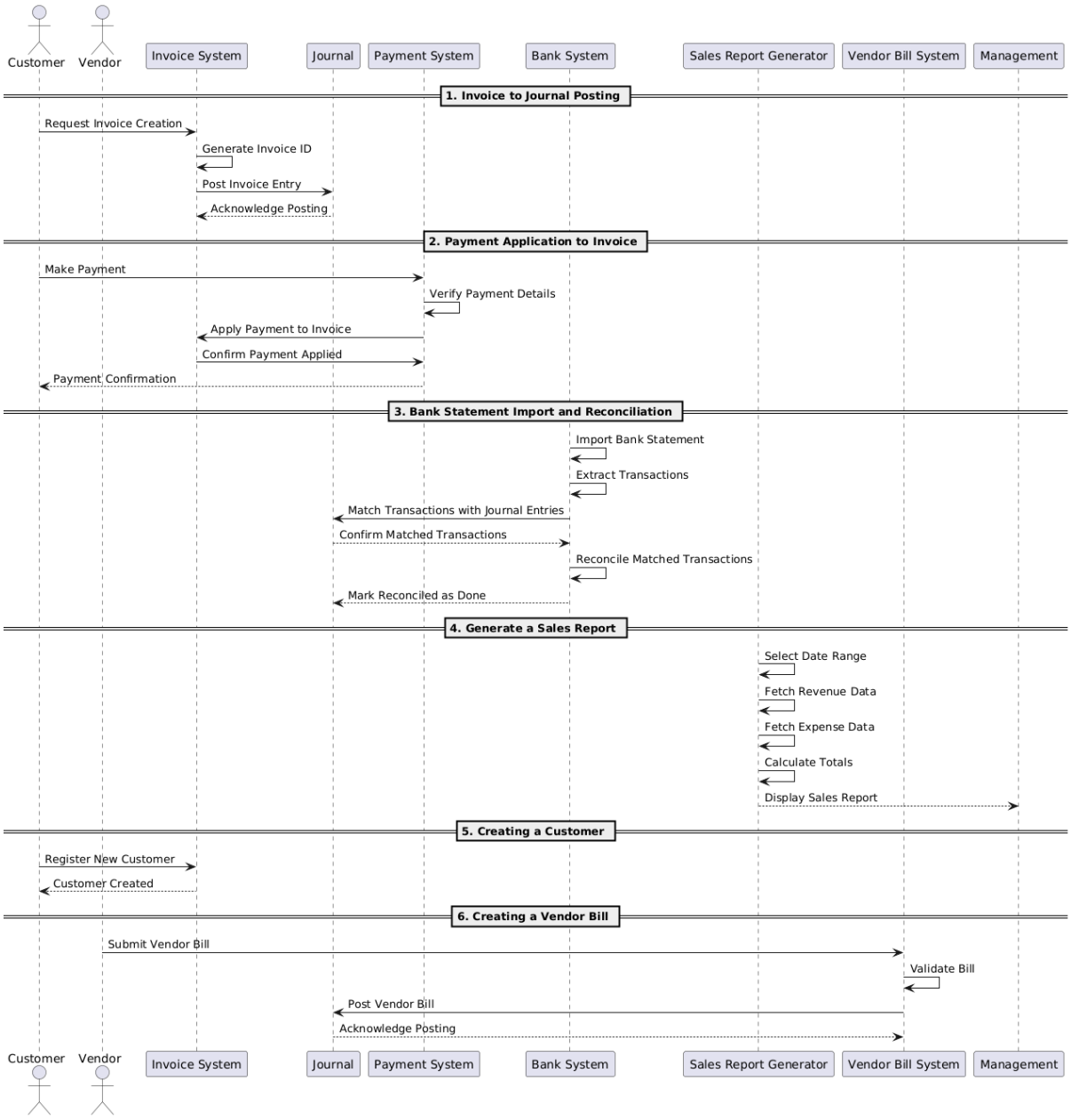
Customer: Receives invoices and makes payments.
-Send Invoice to Customer
-Make Payment

Key Flow:

- The invoice is approved or rejected by the Finance Manager.
- The Manager initiates customer notification about approval status and sends the invoice for approval.
- The Accountant handles several finance document generation, transaction matching, payment recording, and invoice create/update tasks.
- Bank System allows importing bank statements, vendor bill generation, and invoice payment.
- Vendors issue bills and settle invoices.
- Customers receive bills and pay.

Generally, this diagram illustrates an end-to-end invoice processing, approval, financial report, and payment task workflow between involved roles.

Sequence Diagram



Overview of key processes:

1. **Invoice to Journal Posting:**

   -Customer asks system to generate invoice from the Invoice System.

   -System generates invoice ID, posts invoice to Journal, and receives acknowledgment.

2. **Payment Application to Invoice:**

   -Customer applies payment via the Payment System.

   -Payment details are verified, applied to invoice, and payment confirmation is signaled.

3. **Bank Statement Import & Reconciliation:**

   -The Bank System imports bank statements, extracts and reconciles transactions with journal entries.

   -Confirmed and reconciled matched transactions, and status is done.

4. **Generating a Sales Report:**

   -The system facilitates choosing a date range, fetching revenue and expenses, summing totals, and returning the sales report.

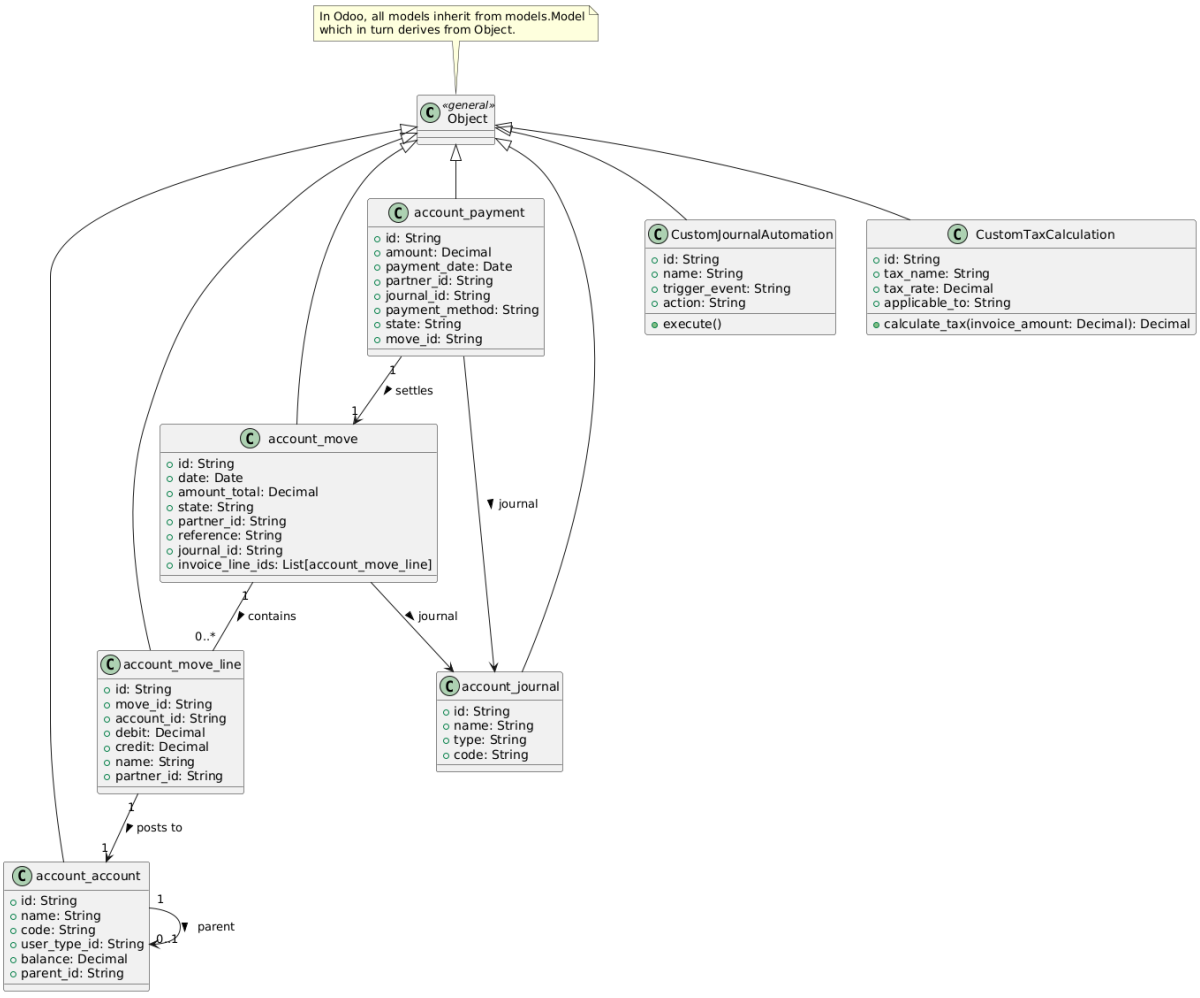
5. **Creating a Customer:**

   -New customers are inserted via the system, and creating a customer is triggered.

6. **Creating a Vendor Bill:**

   -Vendors submit bills, which are validated and posted by the Vendor Bill System, with acknowledgment.

Class Diagram



This is a class diagram illustrating the static structure and relations of a financial accounting system that has been designed with UML.

Key Components:
Object (Base Class): Serves as the superclass to which other classes are extended, as indicated by the note referring to Odoo's model inheritance.
Classes and Attributes:

- account_move: Employed to represent invoice headers with attributes like id, date, amount_total, state, partner_id, reference, and a list of associated invoice lines (invoice_line_ids).
- account_move_line: Represents individual invoice line items against an invoice (move_id), with information like account_id, debit, credit, name, and partner_id.

- account_journal: Represents journal entries containing id, name, type, and code.
- account_payment: Contains payments, related to invoices, with characteristics like id, amount, payment_date, partner_id, journal_id, payment_method, state, and the related move_id.
- account_account: Represents chart of accounts, with identifiers, name, code, user_type, balance, and parent account relation.
- CustomJournalAutomation: Custom class for journal entry automation, with id, name, trigger_event, action; and an execute() method.
- CustomTaxCalculation: Class for tax information, with fields for the tax name, rate, applicable-to info, and a calculate_tax() method to compute taxes on invoice amounts.

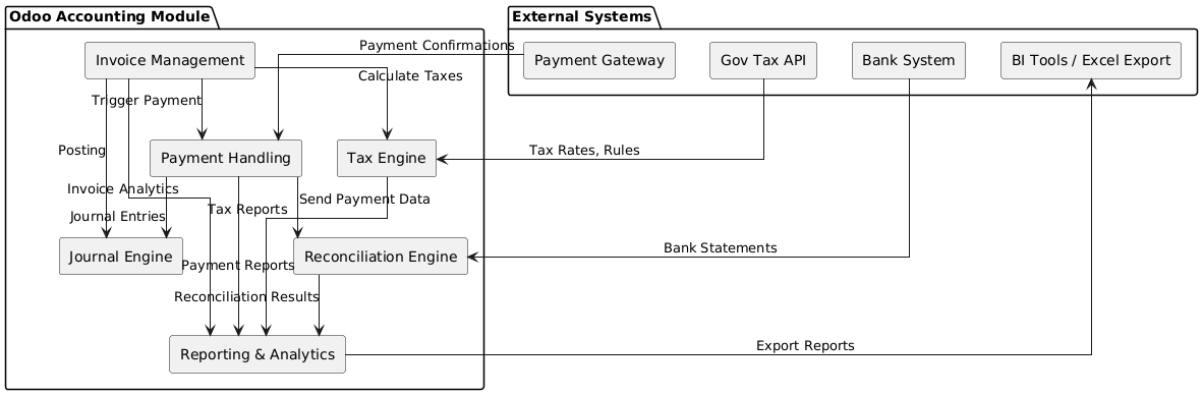
Relationships:
Inheritance: A number of classes inherit from Object, i.e., they are models in a hierarchy.
Associations:

- account_move has multiple account_move_line items (invoice_line_ids).
- account_move is associated with account_journal.
- account_payment is applied to account_move.
- account_move_line posts to account_account.
- CustomJournalAutomation is associated with account_move.
- CustomTaxCalculation is used on invoices (account_move).

Overall:

This UML class diagram defines the key entities involved in invoice processing, payments, accounting, and custom automation/tax calculation in an enterprise finance system, their attributes, and how they relate to each other.

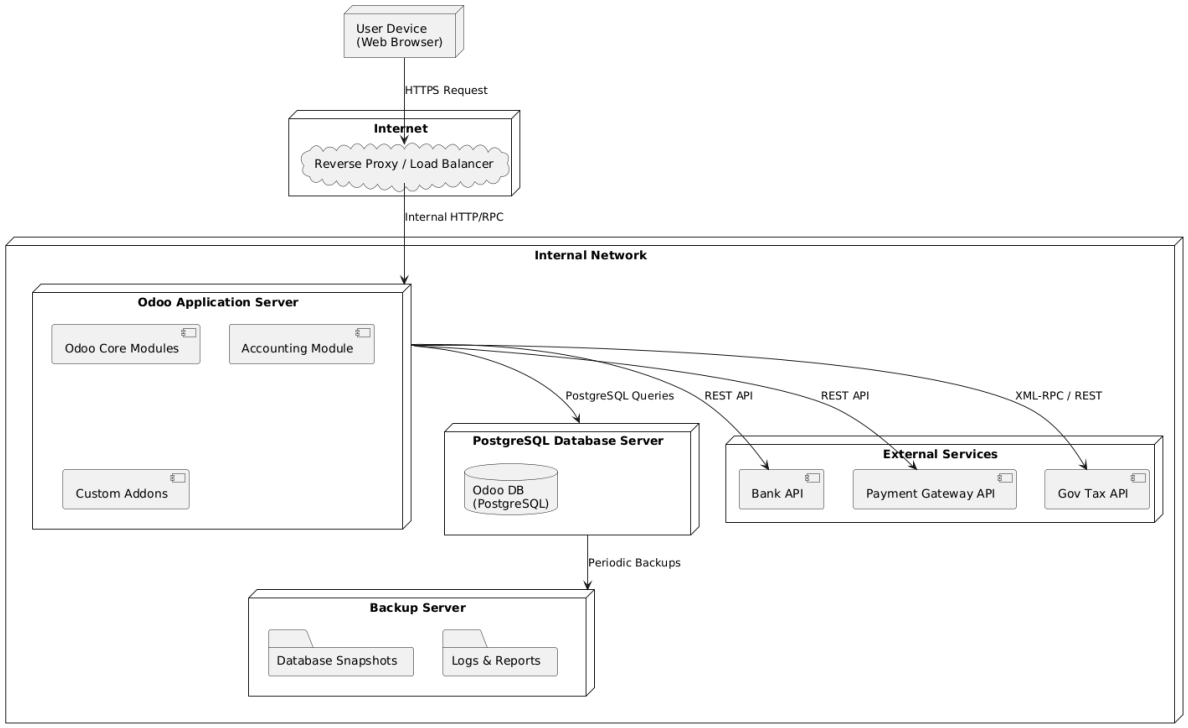
Component Diagram



The component diagram of the Odoo Accounting module provides a comprehensive view of the overall structure of the system, describing internal components and how they communicate with external systems. The module internally divides into distinct components such as Invoice Management, Payment Management, Reconciliation Engine, Reporting and Analytics, Tax Engine, and Journal Engine. Every module contains specialized duties, e.g., Invoice Management is in

charge of invoice creation and tracking and the Tax Engine handles tax rule application and reporting. The modules communicate with each other; the Payment Handling system relies on the Invoice Management and Journal Engine modules, for example, to properly process and book financial transactions. Externally, the system integrates with third-party systems like Bank Systems (reconciliation), Payment Gateways (receipt and payout of payments), Government Tax APIs (tax reporting and compliance), and BI Tools or Excel (advanced reporting and analysis). The integrated and modular architecture allows the Odoo Accounting system to be extensible and solid, enabling seamless integration and efficient financial management.

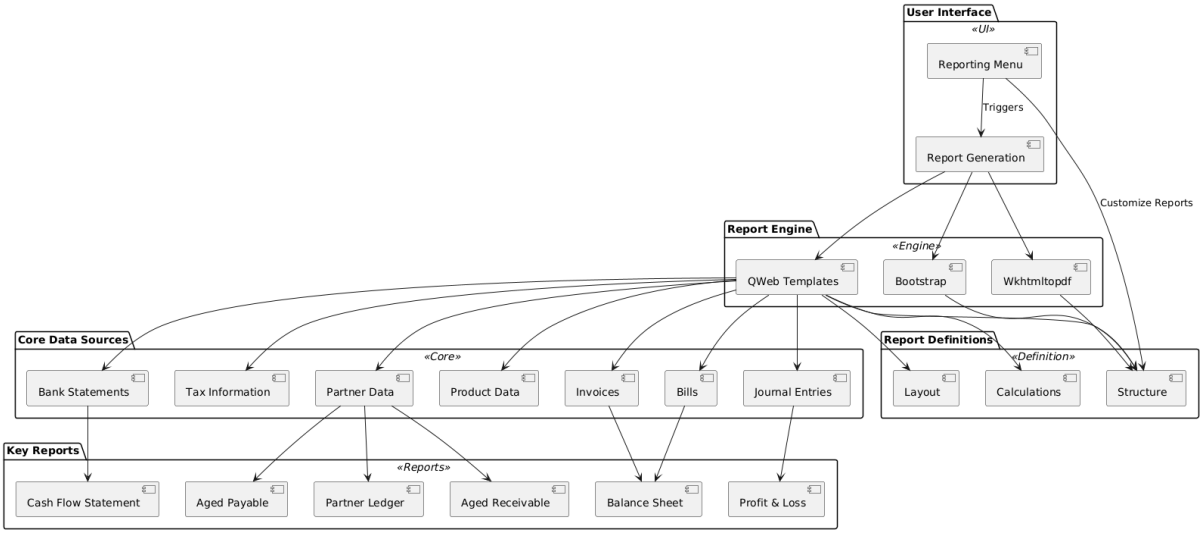
Deployment Diagram



This deployment diagram illustrates the physical structure of an Odoo system with emphasis on the Accounting Module. Users access the application through a web browser over the internet through a reverse proxy or load balancer to distribute traffic and terminate SSL. The Odoo Application Server hosts the base Odoo modules, including the Accounting Module and any addons developed custom. It interacts with a PostgreSQL Database Server that contains all transactional and config data.

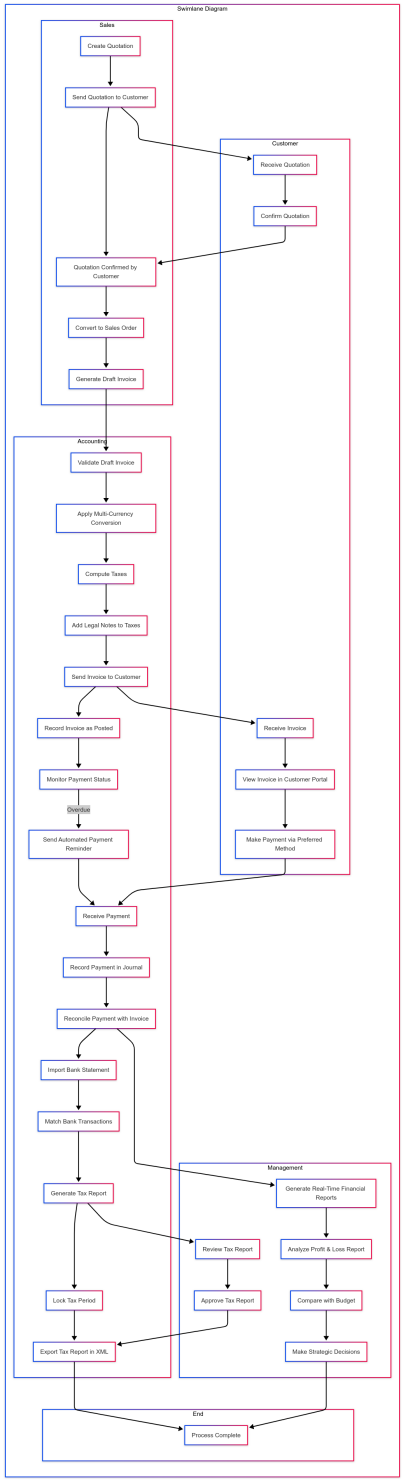
The Odoo software also integrates with various external services such as banking APIs, payment gateways, and government taxation systems to provide support for functionality such as automatic reconciliation, online payments, and regulatory reporting. The architecture also includes a backup server containing snapshots of the databases and significant logs at periodic intervals to ensure data security and recovery alternatives. The secure and modular deployment design facilitates scalability, external integration, and financial compliance operations.

Report Architecture Diagram



The system architecture of the Odoo reporting system for accounting and invoicing modules is presented in the diagram. The Data Sources form the core of the system and include key financial documents that are associated with invoices, bills, journal entries, bank statements, tax information, partner details, and product details. The Report Engine takes input from these data sources and utilizes reporting templates like QWeb, Bootstrap, and Wkhtmltopdf to generate reports in various formats. The user interaction is facilitated by the User Interface, where users launch the reporting menu, customize reports, and trigger report generation. The report generation revolves around the Report Definitions, where the structure, format, and calculations needed for every report are defined. The report engine then takes this information and produces the final reports, including key financial reports like the Balance Sheet, Profit &Loss, Cash Flow Statement, and others. Overall, this architecture enables customized, data-based reporting with a logical evolution from data collection to final report generation.

Activity Diagram



The diagram illustrates the end-to-end accounting and billing process in Odoo 18, integrating the activities of various departments: Sales, Accounting, Customer, and Management. The process begins when the Sales department quotes a customer. The quote is then submitted to the customer for approval. Upon confirmation by the customer, the quote is converted into a sales order. Odoo 18 also generates an automatically created draft invoice based on this sales order (Web ID: 2, 8).

At the Accounting step, the draft invoice is approved and validated by the accounting staff (Web ID: 2). If there is a foreign currency transaction, Odoo 18 refreshes with the latest exchange rate and records the base and foreign currencies (Web ID: 0, 18). Taxes are then calculated against the relevant country localization, accommodating price included or excluded, diverse tax rates, and partial exemptions (Web ID: 1). Legal remarks can also be added to specific taxes to assist with regulatory adherence (Web ID: 5). After this, the invoice is sent to the customer through their desired delivery method such as email or Peppol (Web ID: 4, 16), and its status is set to "Posted," posting journal entries automatically using double-entry bookkeeping (Web ID: 0, 16).

Accounting then goes on to monitor the status of payment. If the payment is overdue, Odoo 18 triggers its dunning procedure, sending automatic reminders to the customer (Web ID: 3, 18). Upon acceptance of the payment, it is recorded as a journal entry (Web ID: 1) and matched against the corresponding invoice using Odoo's smart matching algorithm, which does 95further minimize manual intervention (Web ID: 1, 18). A tax report is subsequently generated, cash or accrual basis (Web ID: 0), the tax period is locked to prevent additional VAT-related postings (Web ID: 0), and the report is exported to XML format for tax authority filing (Web ID: 0).

On the side of the Customer, they validate and accept the quotation, which triggers the creation of the sales order. On receiving approved invoice, customers can view and download it through Odoo's Customer Portal (Web ID: 1) and subsequently arrange for payment through their preferred medium such as SEPA or online gateway like Stripe (Web ID: 1, 4).

Finally, Management reviews the tax report, signs it off for filing, and uses Odoo's real-time financial reporting features to generate vital reports such as the profit and loss statement (Web

ID: 7, 18). They review financial performance and contrast it with budgeted (Web ID: 1), using these insights to inform strategic business decisions. The cycle ends when all the accounting and invoicing functions are finalized and closed.

7.4 Useful Odoo Resources and Documentation links

Here are some useful Odoo resources and links to documentation for finance:

General Finance Resources:

- **Finance:** This is the main landing page for Odoo's finance documentation, providing an overview of the various finance-related topics.

  https://www.odoo.com/documentation/18.0/applications/finance.html

- **Accounting and Invoicing:**
  - **Accounting and Invoicing:** This page describes the fundamental accounting features of Odoo, including invoicing, managing payments, and setting up your chart of accounts.

    `https://www.odoo.com/documentation/18.0/applications/finance/accounting.html`
  - **Customer Invoices:** Discover how to create, send, and manage customer invoices in Odoo.

    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/customer_invoices.html`
  - **Invoicing Processes:** This section explains how to automate customer invoice generation in Odoo, for example, from sales order, delivery order, and contract.

    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/customer_invoices/overview.html`
  - **Electronic Invoicing (EDI):** earn about electronic data interchange for business documents like invoices.

    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/customer_invoices/electronic_invoicing.html`
  - **Odoo electronic invoicing in Austria:** Information about legally compliant e-invoicing solutions for Austria.

    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/customer_invoices/electronic_invoicing/austria.html`
  - **Invoicing Method:** Explains various invoicing methods, for example, down payments, pro-forma invoices, and time and material-based invoicing.

    `https://www.odoo.com/documentation/18.0/applications/sales/sales/invoicing.html`
  - **Invoicing by time and materials:** Information on how to invoice customers by time and/or materials.

    `https://www.odoo.com/documentation/18.0/applications/sales/sales/invoicing/time_materials.html`

- **Accounting Features and Configuration:**
  - **Chart of Accounts:** This is the explanation of the chart of accounts, which is the list of all accounts used to record financial transactions.

    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/get_started/chart_of_accounts.html`

  - **Bank and Cash Accounts:** Learn how to manage your bank and cash accounts in Odoo.
    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/bank.html`
  - **Bank Synchronization:** Find out how to synchronize Odoo with your bank to automatically import bank statements.
    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/bank/bank_synchronization.html`
  - **Loans Management:** This section describes how to manage loans taken by your company.
    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/bank/loans.html`
  - **Online Payments:** Find out how to install and manage online payments in Odoo.
    `https://www.odoo.com/documentation/18.0/applications/finance/payment_providers.html`
  - **Payments:** This section describes how payments can be linked to invoices or as standalone records.
    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/payments.html`

- **Reporting:**
  - **Reporting:** This chapter has details about Odoo's various financial reports, such as the Balance Sheet, Profit and Loss account, and Cash Flow statement.
    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/reporting.html`
  - **Year-End Closing:** Learn about the year-end closing process in Odoo.
    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/reporting/year_end.html`

- **Getting Started:**
  - **Get Started:** It takes you through the Odoo Accounting initial setup.
    `https://www.odoo.com/documentation/18.0/applications/finance/accounting/get_started.html`