Requirements Document - EZShop

Date: 24/10/2025

Version: 1.0.0

Version number	Change

Contents

Informal description

Small shops require a simple application to support the owner or manager. A small shop (ex a food shop) occupies 50-200 square meters, sells 500-2000 different item types, has two or a few more cash registers. EZShop is a software application to:

- manage sales
- manage inventory
- manage orders to suppliers
- support accounting

In the following describe the requirements of the EZShop application. You are free to define the application as you deem more useful and effective for the stakeholders. You are also free to modify the structure of the document when needed. The document will be evaluated considering the typical defects in requirements (omissions, ambiguities, contradictions, etc), and syntactic errors in the formalism used (UML diagrams). Consider that the document should be delivered to another team (unknown to you) which will be in charge of designing and implementing the system. The design team should be able to proceed only with the information in the document.

Business Model

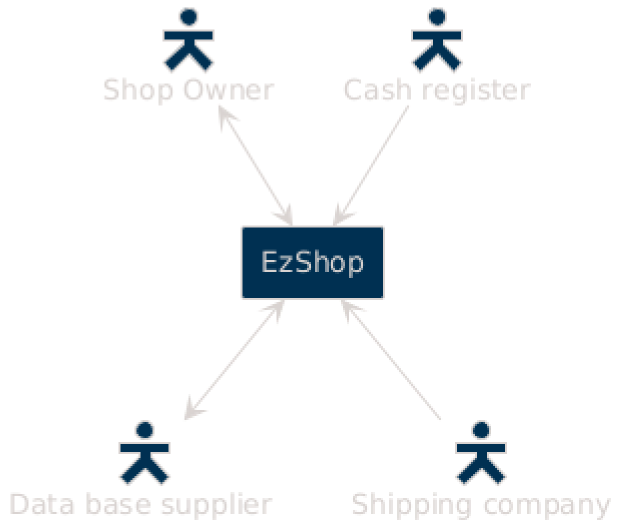
- Customer segment
  - Shop manager / owner
- Value proposition
  - Statistics reports
  - Accounting (revenue, costs, debts, ...)
  - Decision support
  - Assistance with logistics
  - Order management to suppliers
- Revenue Stream
  - Yearly fee
  - Privately founded

Stakeholders

Stakeholder name	Description
Shop owner	Reviews accounting information and initiates orders to suppliers
Cashier	Handles transactions and enters sales info through the cash register
Cash register	Feeds data to the application
Desktop OS	The device where the application resides
Product suppliers	Business who supply the shop with items
Shipping company	Company who delivers the items to the shop
DB service supplier	Stores data
Development team	Team responsible for the development of the application
Maintenance team	Team responsible for the long term support of the application
Competitors	Other companies that sell a similar product

Context Diagram and interfaces

Context Diagram



Interfaces

Actor	Logical Interface	Physical Interface
Shop manager	GUI	Desktop computer
Data base	API	Internet connection
Cash register	API	Internet connection
Shipping company	API	Internet connection

Functional and non functional requirements

Functional Requirements

<In the form DO SOMETHING, or VERB NOUN, describe high level capabilities of the system>

<they match to high level use cases>

ID	Description
FR1	Manage sales
FR2	Manage inventory
FR3	Manage orders
FR4	Manage accounting

- Manage sales
    - Read sale form cash register
    - Save last sale
    - Update inventory (with contents of the sale)
    - Export list as .csv
    - Display sale stats
- Manage inventory
    - Manage items
        - Delete item from inventory
        - Add new item to inventory

        - Edit item characteristics
    - View items
        - Display list of available items
        - Search items by selected fields
        - Sort items by selected fields
        - Group items by selected fields
        - Filter items by selected fields
    - Display inventory stats
    - Change item status when expiration date is passed
    - Notify user when items are past the expiration date
    - Export list as .csv
- Manage orders
    - Add new order
    - Delete order
    - Edit order
    - Automatically *track order* status for supported suppliers
    - Suggest order when quantity of certain item is below threshold
        - Accept, delete, edit suggested order
    - View orders
        - Display list of orders
        - search orders by selected fields
        - sort orders by selected fields
        - group orders by selected fields
        - filter orders by selected fields
    - Display order stats
    - Export list as .csv
- Manage accounting
    - Track Invoices
    - Track Incomes
    - Track balance
    - Track taxes
    - Display accounting stats
- Authenticate owner
    - Set password
    - Change password
    - Verify password

Usecase:

- item fields
    - code
    - issued date
    - delivery date
    - name
    - category
    - brand
    - status (pending, ordered, shipped, delivered, unloaded, damaged, expired)
    - Supplier
    - Shipping company
    - price
    - number of items ordered

- orders fields
    - code
    - supplier name
    - date
    - status
    - quantity
    - items

Non Functional Requirements

<Describe constraints on functional requirements>

ID	Type (efficiency, reliability, ..)	Description	Refers to
NFR1			
NFR2			
NFR3			
NFRx ..			

Table of rights

Actor	FR1	FRx

Use case diagram and use cases

Use case brief

UC name	Goal	Description

Use case diagram

<define here UML Use case diagram UCD summarizing all use cases, and their relationships>

<next describe here each use case in the UCD>

Use case 1, UC1

Actors Involved	
Precondition	<Boolean expression, must evaluate to true before the UC can start>
Post condition	<Boolean expression, must evaluate to true after UC is finished>
Nominal Scenario	<Textual description of actions executed by the UC>
Variants	<other normal executions>
Exceptions	<exceptions, errors >

Scenario 1.1

<describe here scenarios instances of UC1>

<a scenario is a sequence of steps that corresponds to a particular execution of one use case>

<a scenario is a more formal description of a story>

<only relevant scenarios should be described>

Scenario 1.1	
Precondition	<Boolean expression, must evaluate to true before the scenario can start>
Post condition	<Boolean expression, must evaluate to true after scenario is finished>

Steps

Actor's action	System action	FR needed

Scenario 1.2

Scenario 1.x

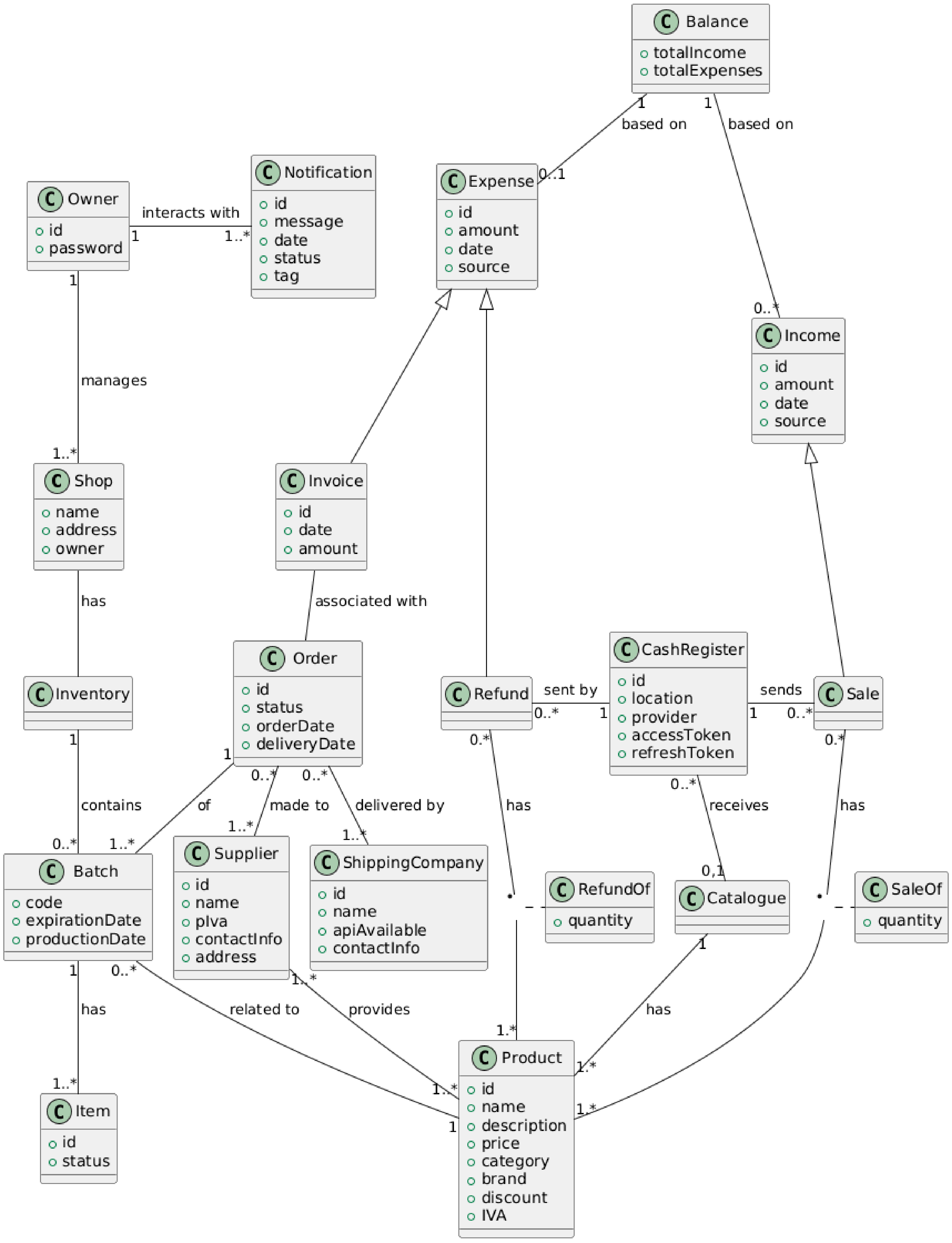
Use case 2, UC2

Use case x, UCx

Glossary

- **Shop**
  - The small business entity that uses the EzShop software to manage its operations, including sales, inventory, orders, and accounting. A shop typically has one owner, two or more cash registers, and several suppliers. In the current scope, EzShop manages a single shop.
- **Product**
  - An entity representing a type of product sold or stocked by the shop; not a single physical object. Each product describes the characteristics shared by all physical units of that product (e.g., 1L bottle of "GoodMilk!").
- **Batch**
  - A batch represents a specific group of items associated with a product in the inventory. Multiple batches can exist for the same product, each identified by its own quantity and expiration date.
- **Item**
  - An item represents a single physical unit of a product that exists in the shop's inventory. Each item belongs to a specific batch.
- **Sale**
  - An event, identified by a unique code, that occurs every time a customer completes the purchase of one or more items. A sale records a list of items with associated quantities, the date and the total amount spent.
- **Refund**
  - An event, identified by a unique code, that occurs every time a customer successfully returns one or more items. A refund records a list of items with associated quantities, the date and the total amount owed.
- **Catalogue**
  - The catalog is the complete collection of all product available in the shop's system. The catalog serves as a reference for managing inventory, creating orders, and displaying available products to the user.
- **Inventory**
  - The collection of all batches (identified by code) and the name of the product they belong to. It represents the shop's overall product availability and supports search, filtering, expiration tracking and quantity monitoring.
- **Supplier**

  - A business entity providing batches to the shop. Each supplier may have identifying details such as name and P.IVA.
- **Invoice**
  - A formal document that records a financial transaction between the shop and a supplier or customer, serving as proof of purchase or sale.
- **Order**
  - The purchase of a collection of batches from a supplier. Orders have a defined structure and status, typically one of: processing, shipped, in transit, delivered and cancelled.
- **Income**
  - The amount of money earned from completed sales.
- **Expenses**
  - The amount of money spent by the shop to maintain it operation such as electrical bills, renting fees, wages, and products purchases.
- **Balance**
  - The overall financial position of the shop, calculated as the difference between total incomes and total expenses within a given period.
- **Owner**
  - The shop holder or manager who uses the EzShop software to control sales, inventory, orders, and accounting. The real end user of the software.
- **Cash Register**
  - A physical or digital terminal that records sales data and transmits it to the EzShop application via API.
- **Shipping company**
  - A third-party service that delivers batches from suppliers to the shop. Some shipping company offer APIs to track delivery and shipment status.
- **Notification**
  - A system-generated message or alert that informs the shop owner about important events or conditions, such as changes in order status, low product quantity, batch expiration, or connectivity issues.


Balance
totalIncome
totalExpenses
based on
based on
Notification
id
message
date
status
tag
Owner
id
password
interacts with
Expense
id
amount
date
source
Income
id
amount
date
source
manages
Shop
name
address
owner
Invoice
id
date
amount
has
associated with
Order
id
status
orderDate
deliveryDate
Inventory
Refund
sent by
CashRegister
id
location
provider
accessToken
refreshToken
sends
Sale
contains
of
made to
delivered by
has
receives
has
Batch
code
expirationDate
productionDate
Supplier
id
name
pIva
contactInfo
address
ShippingCompany
id
name
apiAvailable
contactInfo
RefundOf
quantity
Catalogue
SaleOf
quantity
has
related to
provides
has
Item
id
status
Product
id
name
description
price
category
brand
discount
IVA

<use UML class diagram to define important terms, or concepts in the domain of the application, and their relationships>

<concepts must be used consistently all over the document, ex in use cases, requirements etc>

System Design

<describe here system design>

<must be consistent with Context diagram>

Hardware Software architecture

<describe here the hardware software architecture using UML deployment diagram >