Universidad Autónoma de Ciudad Juárez

Campus CU-IIT

Subject: Software Requirements Development

Team: #2

Waste To Treasure

Professor: Benito Alán Ponce Rodriguez

Student Names:
Oscar Alonso Nava Rivera - 214729
Arturo Perez Gonzales - 215510
Alejandro Campa Alonso - 215833
Gabriel Florentino Reyes - 223154

Deadline: 18/11/2025

SRS Document

1. Introduction

In Ciudad Juárez, the rapid growth of the maquiladora and manufacturing sector has led to a constant increase in industrial waste. According to studies by the Center for Research in Advanced Materials (CIMAV), the city produces a high amount of non-hazardous solid waste from the automotive, woodworking, and electronics industries, many of which have potential for energy recovery or reuse (Luna Velasco, A., Lozoya Márquez, L. A., & González Sánchez, G., 2019). However, the infrastructure for their utilization remains limited, as less than 6% of the waste generated in the city is recycled annually (Ahumada, D., 2025).

At the same time, small producers, workshops, and local artisans face difficulties in accessing affordable or sustainable materials for their production, limiting the growth of the circular economy. Faced with the issues of industrial waste surplus and scarcity of reusable raw materials, Waste-To-Treasure emerges as a technological proposal aimed at transforming industrial waste into valuable resources by connecting companies, recyclers, artisans, and consumers. Thus, the proposal seeks not only to address environmental needs but also to promote a circular economy model that fosters sustainability and local economic development.

This document defines the requirements for the “Waste-To-Treasure” platform, a technological response designed to capitalize on this opportunity by creating a collaborative network where industrial waste becomes a resource.

1.1 Purpose

The purpose of Waste-To-Treasure is to develop a digital platform under the SaaS model, designed to promote the circular economy in industrial areas (such as Ciudad Juárez), by facilitating the connection between industrial waste generators (companies, manufacturing, and recycling) and local transformers (artisans, workshops, and entrepreneurs). It allows users to manage the publication, acquisition, and sale of recyclable materials and manufactured products.

In this way, the platform aims to:

- **Optimize waste management** through local reuse and redistribution.
- **Reduce operational costs** for companies by minimizing material disposal.
- Boost the economic development of small producers by providing access to low-cost raw materials.

- **Promote sustainable practices** aligned with the Sustainable Development Goals (SDGs), specifically SDG 9 (Industry, Innovation and Infrastructure), SDG 11 (Sustainable Cities), and SDG 12 (Responsible Consumption and Production).

1.2 Scope

The scope of the Waste-To-Treasure platform focuses on the integration of two interconnected marketplaces:

- **Materials Marketplace (B2B)**
  - The system will allow registered companies and recyclers to publish inventories of reusable or surplus industrial waste.
  - The managed materials include, among others: wood, metal, textiles, plastics, electronics, glass, and cardboard.
  - Through this, the system will enable artisans and workshops to search for, filter, and acquire these materials.
- **Products Marketplace (B2C):**
  - The system will allow local artisans and workshops to sell the products they have created. whether using the recycled materials they have acquired or through other means.
  - The system will also allow end consumers to purchase these products.

Support Functionality (SaaS): the system will have fixed subscription plans for companies and artisans, and variable transaction commissions (estimated at 10%). In turn, a free plan will be offered, aimed at end consumers of the Products Marketplace.

In summary, the scope of the system includes the design, development, and implementation of an integrated platform that transforms industrial waste into economic and environmental opportunities, contributing to the sustainability of various communities.

1.3 Definitions, acronyms, and abbreviations

Terms	Definition
W2T	Acronym associated with the proposed platform.
Artisans	Specialized workers who create handmade products or products made with traditional tools, using technical skills developed through experience and learning.
Circular economy	It is an economic model that seeks to eliminate waste and maximize the use of

	resources through continuous cycles of use.
API	Application Programming Interface.
EC2	Amazon Elastic Computer Cloud.
RDS	Amazon Relational Data Service.
S3	Amazon Simple Storage Service.
EGS	Environment, social and government.

1.4 References

Ahumada, D. (October 21, 2025). *Ciudad Juárez recicla menos del 6% de basura que produce anualmente.* Radio Juarense. Ciudad Juárez recicla menos del 6% de basura que produce anualmente

Luna Velasco, A., Lozoya Márquez, L. A., & González Sánchez, G. (2019). *Potencial de residuos industriales generados en Ciudad Juárez, Chihuahua, México, como combustibles alternos en un horno cementero*. Revista internacional de contaminación ambiental, 35(3), 713-722. 0188-4999-rica-35-03-713.pdf

ONU (s.f.). *Objetivos de Desarrollo Sostenible. Objetivo 9: Construir infraestructura resiliente, promover la industrialización sostenible y fomentar la innovación.* Infraestructura - Desarrollo Sostenible

ONU (s.f.). *Objetivos de Desarrollo Sostenible. Objetivo 11: Lograr que las ciudades sean más inclusivas, seguras, resilientes y sostenibles.* Ciudades - Desarrollo Sostenible

ONU (s.f.). *Objetivos de Desarrollo Sostenible. Objetivo 12: Garantizar modalidades de consumo y producción sostenibles.* Consumo y producción sostenibles - Desarrollo Sostenible

A.Terry Bahill, Steven J. Henderson: Requirements development, verification, and validation exhibited in famous failures, Systems Engineering. 8. 1 - 14. 10.1002/sys.20017 (2005).

1.5 Overview.

This Software Requirement Specification (SRS) document is divided into four sections. The first section establishes the problem context, project purpose, and system scope. The second section provides a general description of the platform, including the product perspective, main functions of the two integrated marketplaces, user characteristics, technical constraints, and system assumptions. The third section details the functional and non-functional requirements, covering user management, transactions, performance standards, security, and scalability. Finally, the fourth section includes supporting information with technical appendices, glossary of terms, and documentary references that support the system's design and implementation decisions.

1.6 Stakeholder Analysis

The purpose of stakeholder analysis is to understand who the relevant actors are, what is expected from the system, and how their participation affects software requirements (A.Terry Bahill, Steven J. Henderson, 2005). Therefore, the purpose of this section is to identify and analyze the stakeholders of the Waste-To-Treasure system, determining their roles, interests, expectations, and influence within the development and operation of the platform.

Stakeholder	Role	Interest	Level of influence
Companies and general public	Material providers.	Reduce waste disposal cost, comply with environment regulations, improve sustainable image, obtain additional income from surplus sales.	High
Artisans/Local producers and general public	Material buyers and recycled product sellers.	Access to raw materials as well as market visibility.	High
End consumers	Buyers of offered products.	Acquire ecological and responsibly sourced products.	High
Administrators	Responsible for	Ensure stable	High

	platform design, development, and maintenance.	operation, security, and system scalability.	

Key expectations expected by stakeholders:

- Companies and general public: Seek a profitable and practical solution to dispose of their waste surplus.
- Artisans/local producers and general public: Desire a reliable channel to acquire material and sell products.
- End consumers: Expect a simple and reliable purchase experience.
- Technical team/developers: Need tools that facilitate management within the platform.

In this way, participation and interaction during system development will be:

- Companies, artisans/local producers and the general public participate in defining functional requirements.
- Artisans/local producers and end consumers will provide feedback on user experience.
- Technical team/developers define technical, security, and maintenance requirements.

In conclusion, stakeholders analysis allows understanding the motivations and expectations of each actor involved in the Waste-To-Treasure system. In this way, this analysis will serve as a basis to prioritize software requirements, ensure usability, and guarantee that the solution provides value.

2. Overall description

2.1 Product perspective

The product of this SRS is a digital platform that facilitates the circular economy by connecting industrial companies that generate surplus materials with local artisans/producers who transform them into new valuable products.

For companies and maquiladoras, the platform provides an interface where they can publish their usable industrial waste (wood, metal, among others), specifying material type, available quantity, condition, and location. The system automatically generates environmental impact metrics that document the amount of CO2 avoided and kilograms of waste diverted from landfills, allowing companies to report their contribution to ESG objectives and, therefore, reduce waste disposal costs. Similarly, companies have the option to access payment plans that offer greater visibility to their publications, personalized impact reports, and advanced performance metrics.

In the case of artisans and/or local producers, the system offers a marketplace where they can search for and acquire materials at competitive prices or at no cost, depending on the person offering said materials. Additionally, they have access to an integrated second marketplace where they can commercialize products made from these recycled materials, expanding their market reach. The system includes inventory management tools, transaction history, and a rating system that builds trust between buyers and sellers.

The platform's approach is completely digital, eliminating traditional intermediaries and centralizing all transactions in one place. It also includes support tools such as a content moderation system to ensure publication quality, a reporting system for the same objective, material categorization, and notifications when materials of interest are offered.

2.2 Products functions

The Waste-To-Treasure platform is built around two integrated yet functionally independent marketplaces: a Materials Marketplace (for surplus and waste materials) and a Products Marketplace (for upcycled and repurposed items).

The main system functions for each user role are as follows:

- **Material Sellers (General Public, Businesses, etc.):**
  The system allows any registered user, individuals, workshops, or companies to list their available surplus or reusable waste materials (such as wood, textiles, metal, and others). Sellers can manage their listings and define the type of transaction they wish to carry out.
- **Material Buyers (General Public, Artisans, etc.):**
  The system enables users to search, filter, and purchase the materials listed on the Materials Marketplace. The purchase of materials is a standalone action; buyers are not required to perform any further activity on the platform afterward.
- **Product Sellers (Artisans, Creators):**
  The system allows any user to publish and sell finished products in the Products Marketplace. The only requirement is that these products must be made from recycled or reused materials, whether or not those materials were acquired through the W2T Materials Marketplace. Sellers can manage their inventory and describe the sustainable origin of their creations.
- **Product Buyers (End Consumers):**
  The system enables users to browse, search, and purchase sustainable and unique products available on the Products Marketplace, thereby supporting local creators and promoting circular economy practices.
- **W2T Administrators:**
  Through an Admin Console, the technical team will manage the platform, including

user administration, moderation of listings across both marketplaces, and transaction oversight.

2.2.1 Modules

The platform will be organized into the following core modules:

- User management and authentication
- Materials marketplace
- Products marketplace
- Transactions and payment processing
- Communication and ratings
- System management
- Reports and support
- Plans management and metrics

2.2.2 Use cases Diagrams

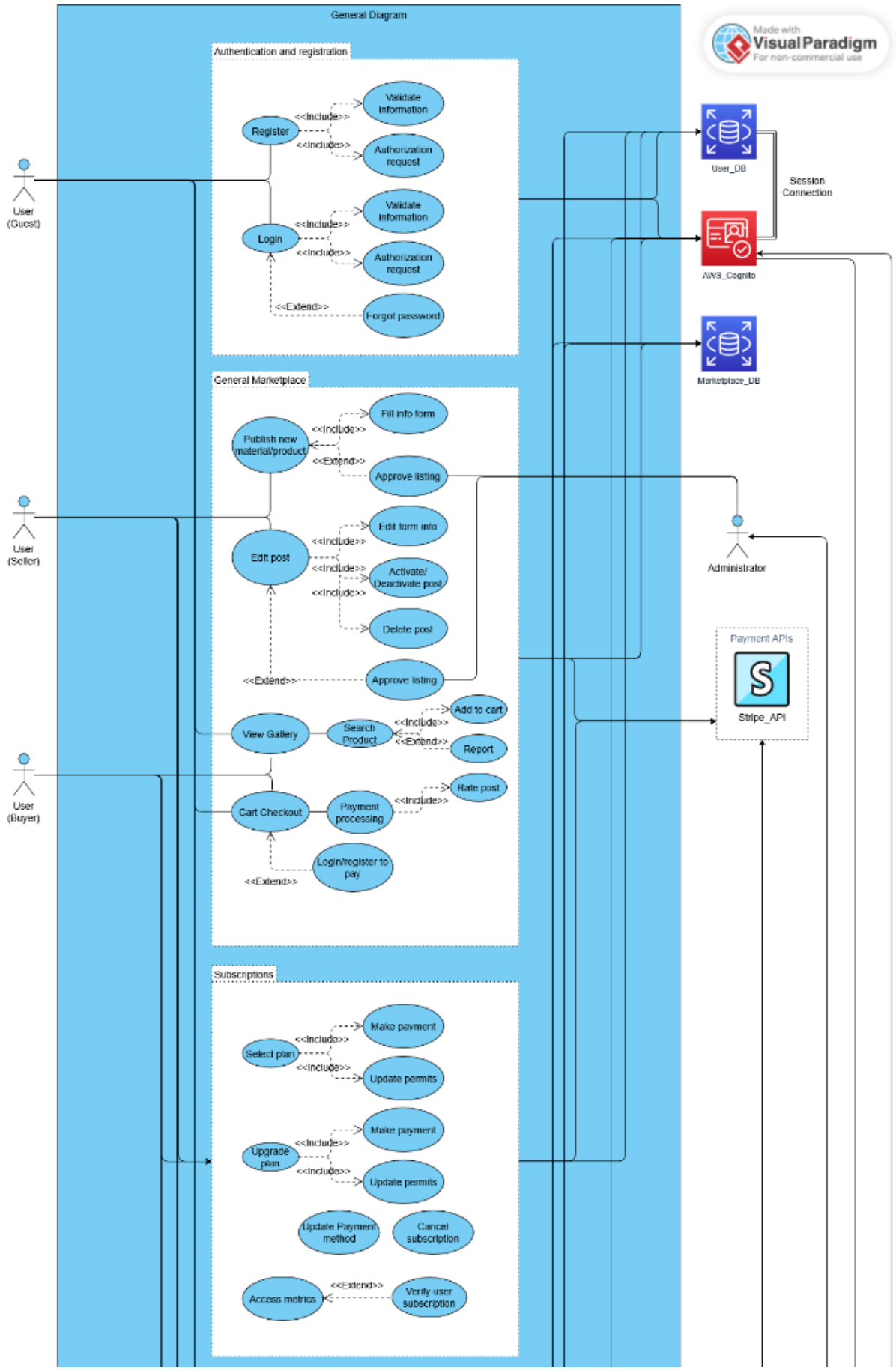
2.2.2.1 General Use Case

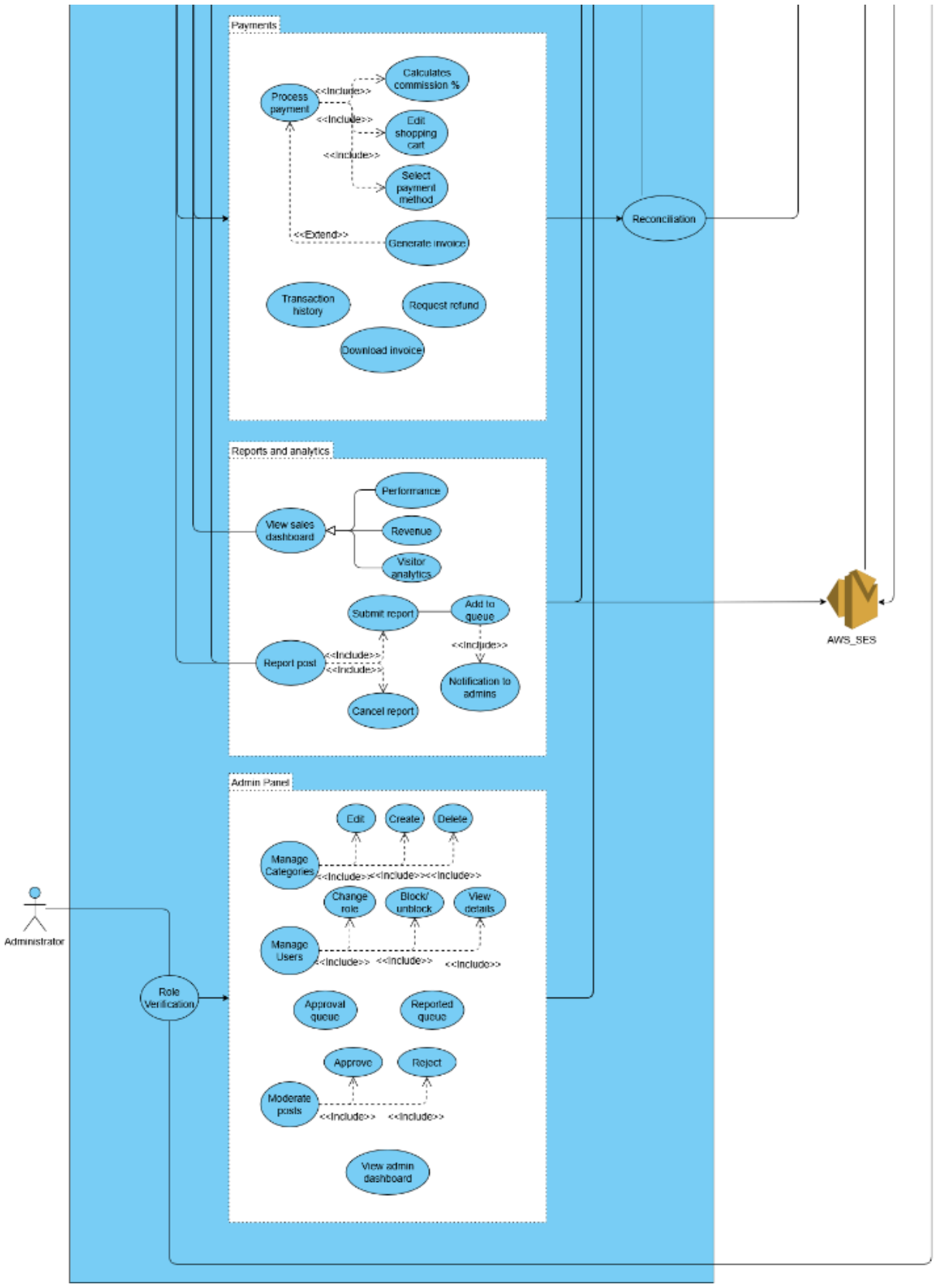
Table 1. General diagram

Name	General Use Case Diagram
Objective	Facilitate the registration of users, allowing interaction with the system and data validation.
Actor	Unauthenticated User (Guest), Buyer, Seller, Administrator
Pre-condition	The system must be operational and accessible via AWS infrastructure. AWS Amplify (frontend hosting) must be deployed and running. Amazon Cognito must be operational for authentication. Amazon EC2 backend (FastAPI) must be running. Amazon RDS (PostgreSQL) must be accessible. Amazon S3 must be available for file storage. External payment APIs must be operational. Internet connectivity is required for all operations.
Main scenario	*User Access and Authentication module:* 1. The guest user accesses the platform via web browser. 2. The system presents login/register options. 3. The user can register via local account (email/password) or OAuth providers (Google/Facebook). 4. Amazon Cognito handles authentication and issues JWT tokens. 5. Authenticated users gain access to role-specific features. *Marketplace module (Material & Product):*

	1. The user browses the materials/products marketplace. 2. The system displays listings with search and filter capabilities. 3. The sellers can publish new materials/products via publication forms. 4. The system validates and sends publications to the admin approval queue. 5. The buyers can view detailed information and add items to cart. 6. The system stores publication data in the marketplace database. *Transaction and Payment module:* 1. The buyer proceeds to checkout with cart items. 2. The system calculates totals and commission. 3. The user enters the billing information. 4. The system processes payment via Stripe. 5. The system performs reconciliation and registers payment in the database. 6. The system generates an invoice (PDF). 8. The seller receives payout notification (48-72hr hold). *Subscription module:* 1. The seller accesses the subscription management. 2. The system displays available plans (Free, Pro, Enterprise). 3. The user selects a plan and processes payment. 4. The system updates user permissions in the subscriptions database. 5. Advanced features unlocked (environmental metrics, increased visibility, advanced analytics…). 6. The system validates the subscription status via Stripe. *Analytics and reporting module:* 1. Sellers access the sales dashboard. 2. The system displays performance metrics (revenue, views conversions, top products). 3. The system queries Material/Products from the database and aggregates data. 4. The users can report inappropriate content. *Administration module:* 1. An administrator accesses the admin console. 2. The system verifies the role via Amazon Cognito. 3. The admin can: View system dashboard (users, transactions, revenue); Manage categories (CRUD operations); Manage users (block/unblock, change roles); Moderate content (approve/reject posts); Review and resolve reports. 4. The system updates the database and logs all admin actions.
Alternative scenario	*Guest user attempts restricted action:* 1. The guest tries to buy items from the cart or publish a listing.

	2. The system redirects to the login/register page. 3. After authentication, the user is returned to the original action. *Payment failure:* 1. Payment gateway declines transaction. 2. The system displays a specific error message. 3. The cart is reserved for retry. 4. The user can select a different payment method. *Subscription update:* 1. The user upgrades from Free to Pro or Pro to Enterprise. 2. The system calculates pro-rated charges. 3. Features immediately unlock. 4. The billing cycle is adjusted. *Service unavailability:* 1. AWS service temporarily unavailable. 2. The system displays an error message. 3. Cached data is shown as soon as possible. 4. The user can retry the operation. *"Multi-role" user flow:* 1. The user acts as both buyer and seller. 2. The system maintains separate contexts. 3. The dashboard shows a combined view. 4. Transaction properly attributed.
Post-conditions	The user is successfully authenticated with a valid session (JWT token). The user role and permissions are properly assigned. Publications are visible in the appropriate marketplace (if approved). The transactions are recorded in the database with the correct commission deduction. The payments are processed and confirmed. Subscription status is updated in the subscriptions database. User permissions reflect current subscription level. Analytics data displayed accurately. The reports are submitted and queued for admin review. Admin actions are logged in the admin logs table. The system stays consistent across all databases. All file assets are stored in Amazon S3.

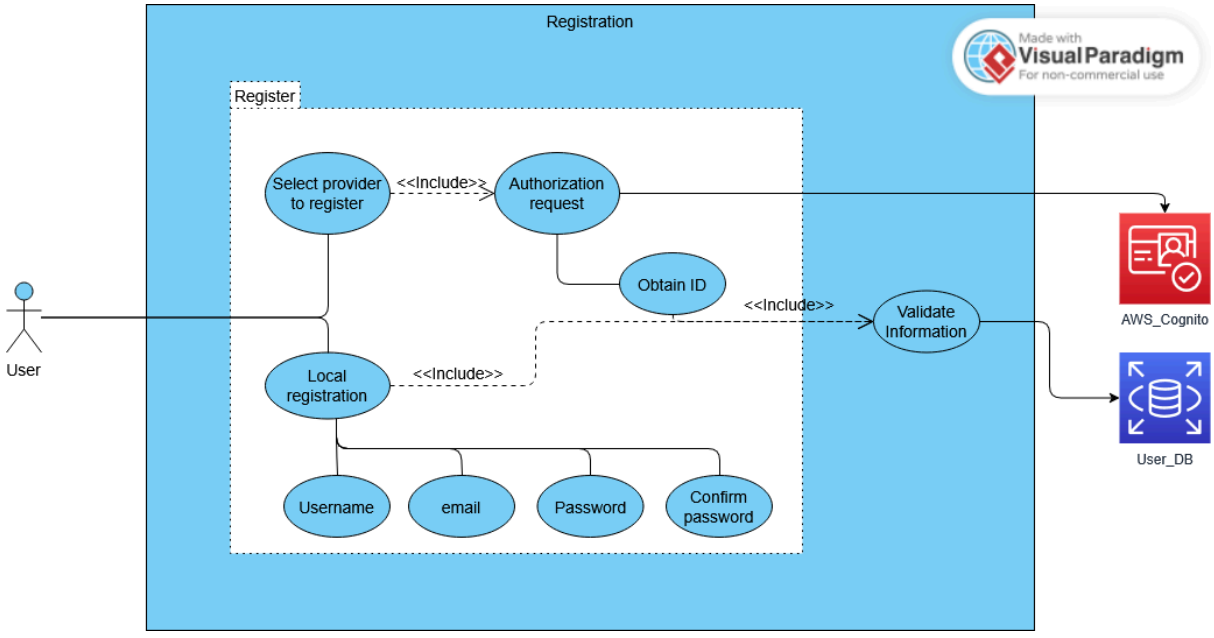

General Diagram
Made with Visual Paradigm
For non-commercial use
Authentication and registration
Register
<<Include>>
Validate information
<<Include>>
Authorization request
Login
<<Include>>
Validate information
<<Include>>
Authorization request
<<Extend>>
Forgot password
User (Guest)
User_DB
Session Connection
AWS_Cognito
Marketplace_DB
General Marketplace
Publish new material/product
<<Include>>
Fill info form
<<Extend>>
Approve listing
Edit post
<<Include>>
Edit form info
<<Include>>
Activate/ Deactivate post
<<Include>>
Delete post
<<Extend>>
Approve listing
User (Seller)
Administrator
View Gallery
Search Product
<<Include>>
Add to cart
<<Extend>>
Report
Payment APIs
Stripe_API
Cart Checkout
Payment processing
<<Include>>
Rate post
<<Extend>>
Login/register to pay
User (Buyer)
Subscriptions
Select plan
<<Include>>
Make payment
<<Include>>
Update permits
Upgrade plan
<<Include>>
Make payment
<<Include>>
Update permits
Update Payment method
Cancel subscription
Access metrics
<<Extend>>
Verify user subscription


Payments
Process payment
<<Include>>
Calculates commission %
<<Include>>
Edit shopping cart
<<Include>>
Select payment method
<<Extend>>
Generate invoice
Transaction history
Request refund
Download invoice
Reconciliation
Reports and analytics
View sales dashboard
Performance
Revenue
Visitor analytics
Submit report
Add to queue
<<Include>>
Report post
<<Include>>
<<Include>>
Notification to admins
Cancel report
AWS_SES
Admin Panel
Edit
Create
Delete
Manage Categories
<<Include>><<Include>><<Include>>
Change role
Block/ unblock
View details
Manage Users
<<Include>>
<<Include>>
<<Include>>
Approval queue
Reported queue
Approve
Reject
Moderate posts
<<Include>>
<<Include>>
View admin dashboard
Administrator
Role Verification

2.2.2.2 Register User

Table 2. Registration

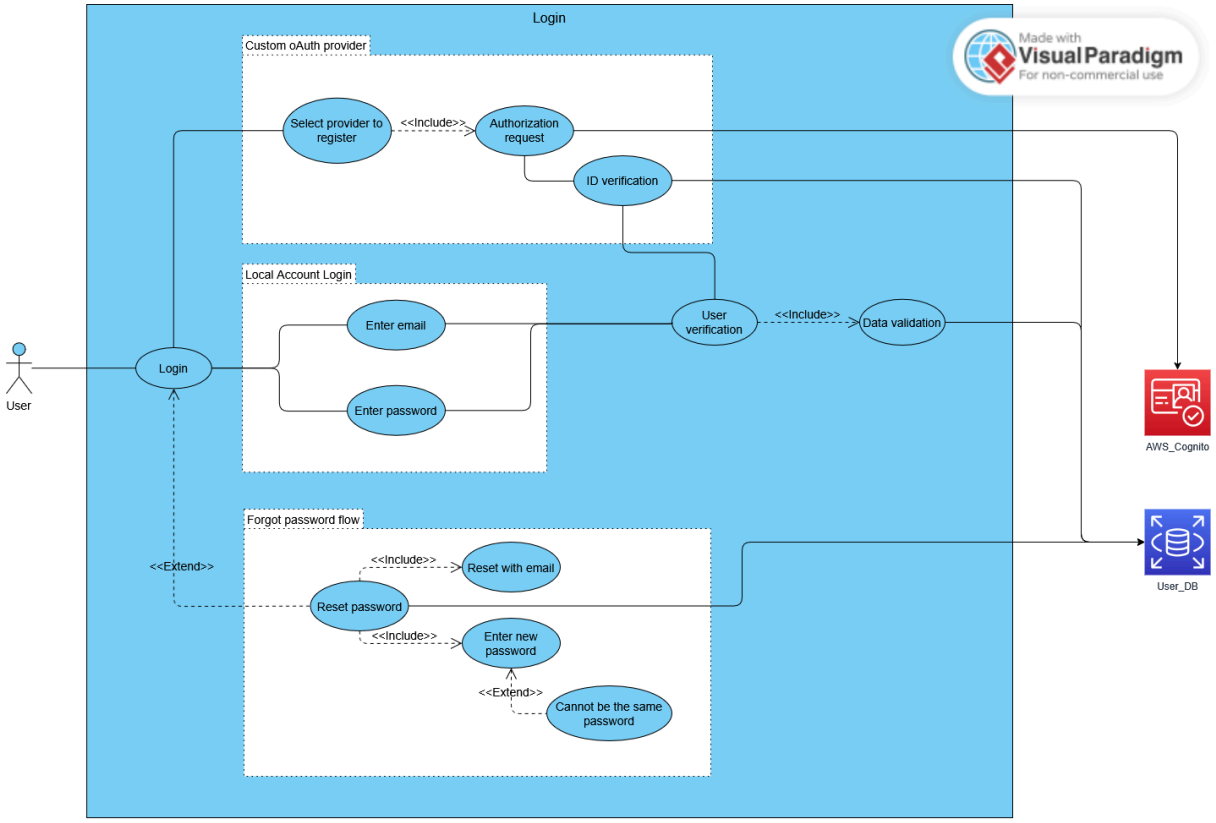
Name	Register Users
Objective	Facilitate the registration of users, allowing interaction with the system and data validation.
Actor	Unauthenticated User (Guest)
Pre-condition	The system must be available, and the user must have access to the required fields.
Main scenario	1. The user will register either locally (Name/Username, Email, Password, and Password confirmation) or by an external provider (Google, Facebook, etc.). 2. If the user registers by an external provider, the system request authorization to Cognito (acting as OAuth 2.0 provider) and, after authorization is granted, the system obtains the user ID to be used in the database. 3. The system validates the data. 4. The system stores the user data in the database.
Alternative scenario	NA
Post-conditions	If all the information is valid, the registration is completed, and the data is stored in the database. If any field is invalid, an error message is displayed. The user will be registered correctly and sent to log in.



2.2.2.3 Login User

Table 3. Login

Name	Register Users
Objective	Allow users to authenticate to the system using their email and password credentials
Actor	Unauthenticated User (Guest)
Pre-condition	The system must be available, and the user must have access to the required fields.
Main scenario	*Cognito login:* 1. The user chooses which provider wants to log with (Google, Facebook, etc). 2. The system makes a request to Cognito for authorization. 3. The system verifies if the user ID exists in the database. 4. The system validates the credentials and authenticates the user. 5. The user is redirected to the home page and displays on screen their user information (if required). *Local account login:* 1. The user enters the registered email address. 2. The user enters the registered password. 3. The system validates the credentials and authenticates the user. 4. The user is redirected to the home page and displays on screen their user information (if required).
Alternative scenario	*Reset Password:* 1. If the user forgets the password, they can select the “Forgot password” option. 2. The user enters the registered email. 3. The system sends an email with a link to reset the password. 4. The user creates a new password, which can’t be the old one. *User ID doesn’t exists:* 1. If the user ID does not exist in the database, the system redirects the user to the register view.
Post-conditions	The user is authenticated and redirected to the home page if the credentials are valid. If authentication and validation fails, an error message is displayed.

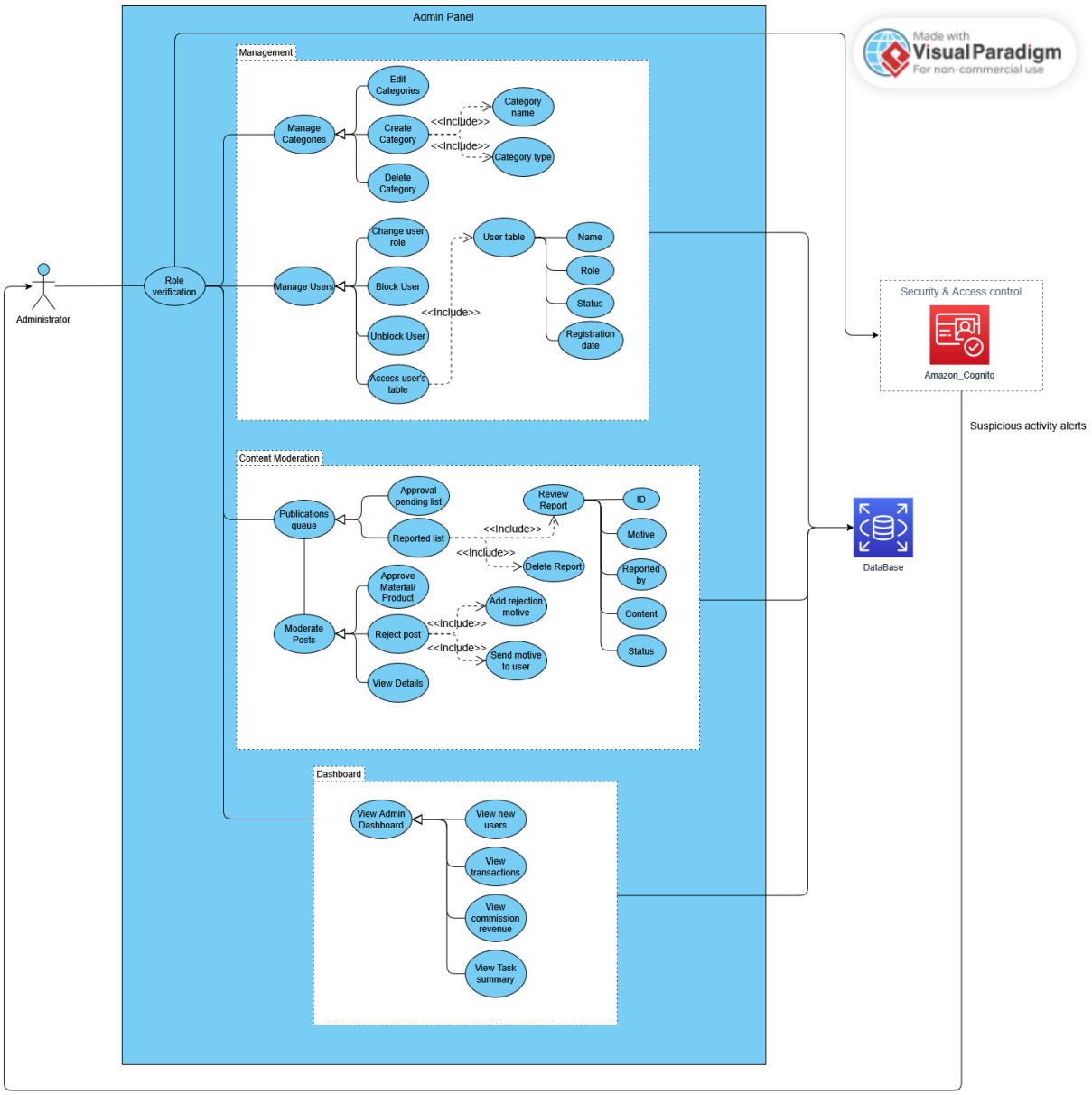


2.2.2.4 Administrator Panel

Table 4. Admin Panel

Name	Admin Panel
Objective	Allows administrators to manage the categories and the users in the system, moderate the existing content, and access some general data about the system.
Actor	Administrator
Pre-condition	Administrator authenticated. Role verified via Amazon Cognito. The database is accessible.
Main scenario	1. The admin accesses the admin panel and the system performs a role/account verification via Amazon Cognito. 2. Once the user is verified as an admin, he has access to three mayor options: *Management:* 1. The system allows an admin user to manage the categories (create, edit, and delete) and then update the database after a modification.

	2. The system also allows an admin user to manage the users (change user role, block/unblock), and access a detailed information window (name, role, status, registration date) of any user in existence. The system updates after any modification. *Content Moderation:* 1. The system shows the publications that are in the approval queue (new posts waiting for approval) and report queue (waiting for review). Any rejection is notified to the seller with an admin motive (reason for rejection). The system updates. *Dashboard:* 1. The system shows a dashboard with general information about the system that exists in the database (new users, transactions, commissions revenue, task summary).
Alternative scenario	NA
Post-conditions	Category created/updated/deleted successfully. User role changed, blocked/unblocked updates successfully along with a successful notification sent to the user. New posts (if approved) are added to their respective marketplace. Admin actions are logged with a timestamp.



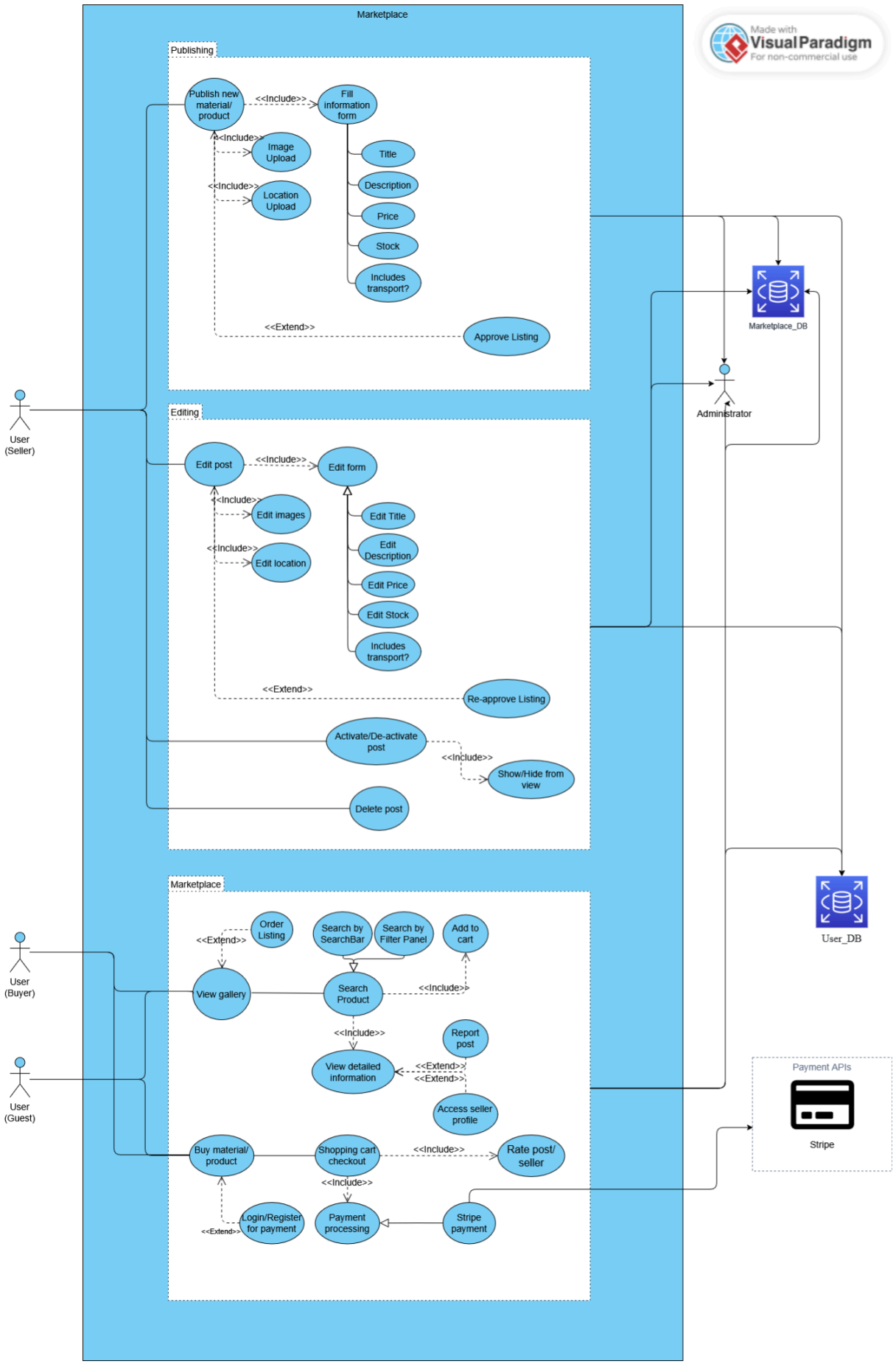
2.2.2.5 General Marketplace (Materials/Products)

Table 5. Marketplace

Name	General Marketplace
Objective	Allows a user (seller) to publish any material/product that may be interested in selling to the platform, as well as an option to edit said post if a mistake would be made. Any authenticated/unauthenticated user that enters the platform can view all marketplace's products. If necessary a report system is available.
Actor	Authenticated User (Seller/Buyer), Unauthenticated User (Guest)

Pre-condition	The system database is available and the system isn't out of service. For certain options, the user must be registered.
Main scenario	*Marketplace:* 1. The user enters the platform domain. 2. The user accesses any marketplace (material/product). 3. The user is allowed to navigate said marketplace with a search bar for easier filtering, a search panel for quick filtering, and an option to order the list by date, price, or rating. 4. The system allows the user to open a detailed view of any product found in the listing (a report system is available, as well as a link to the seller profile if needed). 5. The system redirects the user to the payment process (shopping cart checkout, payment processing) with Stripe integrations. *Publishing:* 1. The seller logs in to the platform. 2. The seller publishes a new material/product by filling a form (title, description, price, stock, includes transport, images, location). 3. The post loads into the system and is redirected to the approve queue waiting for admin approval. *Editing:* 1. The seller is logged in the platform and enters the "edit post" of one of their posts. 2. The system allows the seller to edit any of the form spaces filled in the publishing section. 3. Once editing ends, the post is updated in the system and is redirected, again, for admin approval. 4. The seller can deactivate/activate a post (if deactivated the post is marked as hidden and cannot be viewed by any other user, if activated the post is marked as shown and any user can see it). 5. The seller can delete a post at any moment. Once deleted the system deletes the information in the database and then updates.
Alternative scenario	*The user is not registered/logged in:* 1. If any user wants to buy a product but is not authenticated. 2. The user is redirected to the login/register page. *The user bought and received a product:* 1. The system sends a notification to the user (rate the product/seller). 2. A link is provided to the post for quick access.
Post-conditions	A publication is approved and posted to the platform. An existing post that has been edited, is approved and updated to the platform. The marketplace shows all products that exist in the database.

	Each post connects to their respective seller.

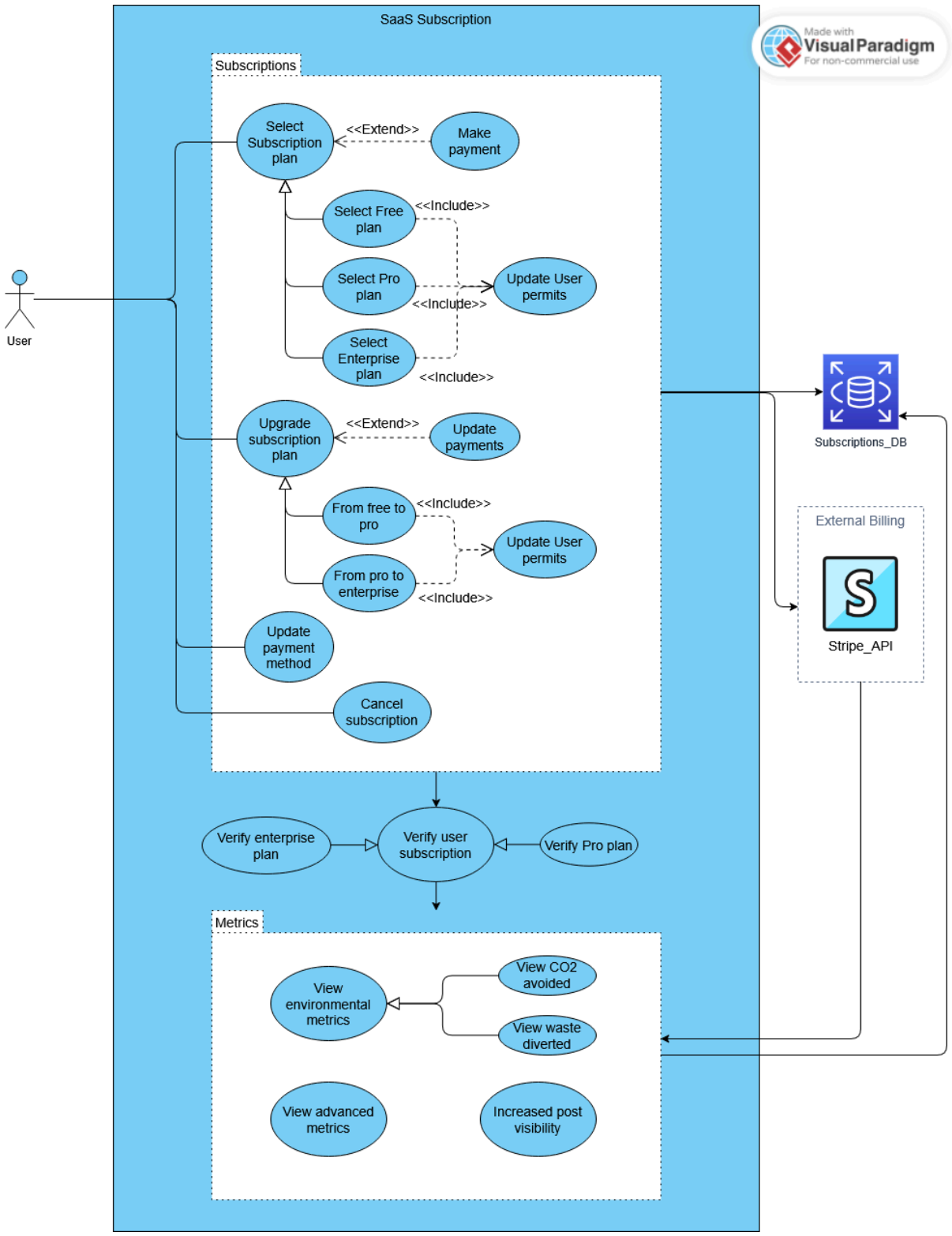

Marketplace
Publishing
Publish new material/ product
<<Include>>
Fill information form
<<Include>>
Image Upload
<<Include>>
Location Upload
Title
Description
Price
Stock
Includes transport?
<<Extend>>
Approve Listing
Editing
Edit post
<<Include>>
Edit form
<<Include>>
Edit images
<<Include>>
Edit location
Edit Title
Edit Description
Edit Price
Edit Stock
Includes transport?
<<Extend>>
Re-approve Listing
Activate/De-activate post
<<Include>>
Show/Hide from view
Delete post
Marketplace
<<Extend>>
Order Listing
Search by SearchBar
Search by Filter Panel
Add to cart
View gallery
Search Product
<<Include>>
<<Include>>
Report post
View detailed information
<<Extend>>
<<Extend>>
Access seller profile
Buy material/ product
Shopping cart checkout
<<Include>>
Rate post/ seller
<<Include>>
<<Extend>>
Login/Register for payment
Payment processing
Stripe payment
User (Seller)
User (Buyer)
User (Guest)
Administrator
Marketplace_DB
User_DB
Payment APIs
Stripe
Made with VisualParadigm
For non-commercial use

2.2.2.6 Subscriptions plans

Table 6. Subscriptions

Name	Subscriptions
Objective	Allows a user to unlock new benefits to their account by paying a monthly subscription to the platform or just limits the publishing interactions with the free access tier. For any paying user, the metrics section is habilitated.
Actor	Authenticated user
Pre-condition	The user must have an existing account in the platform. The platform must be available. The payment connections must be available to the platform.
Main scenario	1. The user logs into the platform. 2. The user accesses the “subscription” window. 3. The user has the option to stay in the free tier (limited publications), the pro or enterprise plan (access to metrics and other benefits). 4. The user selects a payment subscription and is redirected to the payment process to complete it. 5. The system validates the payment and the user permits/benefits are updated. *Metrics:* 1. The system validates the user subscription plan. 2. After the validation is completed, the user has access to the metrics section and the benefits (environmental metrics, advanced metrics, and increased post visibility).
Alternative scenario	*User wants to upgrade:* 1. The user accesses the “subscription” window and selects the next plan. 2. The system redirects the user to the payment process. 3. The user pays the rest of the amount needed. 4. If payment is processed and validated, the system updates the account and the system (the user has access to greater benefits). *User wants to update payment method:* 1. The system redirects the user to a payment information window. 2. The user updates the payment information and saves. 3. The system connects to the payment APIs for confirmation. 4. The system verifies with payment APIs, and updates the information in the database. *User cancels a subscription:*

	1. The user accesses the account information, in the plan/subscription section. 2. The user cancels the subscription. 3. The system cancels any further billing and updates the user's plan to the free tier. 4. The system deactivates the benefits for the user until the last day that was paid.
Post-conditions	The user has added benefits connected to their account. Publishing limitations are removed (5 active posts allowed). The system automatically bills the user every month.

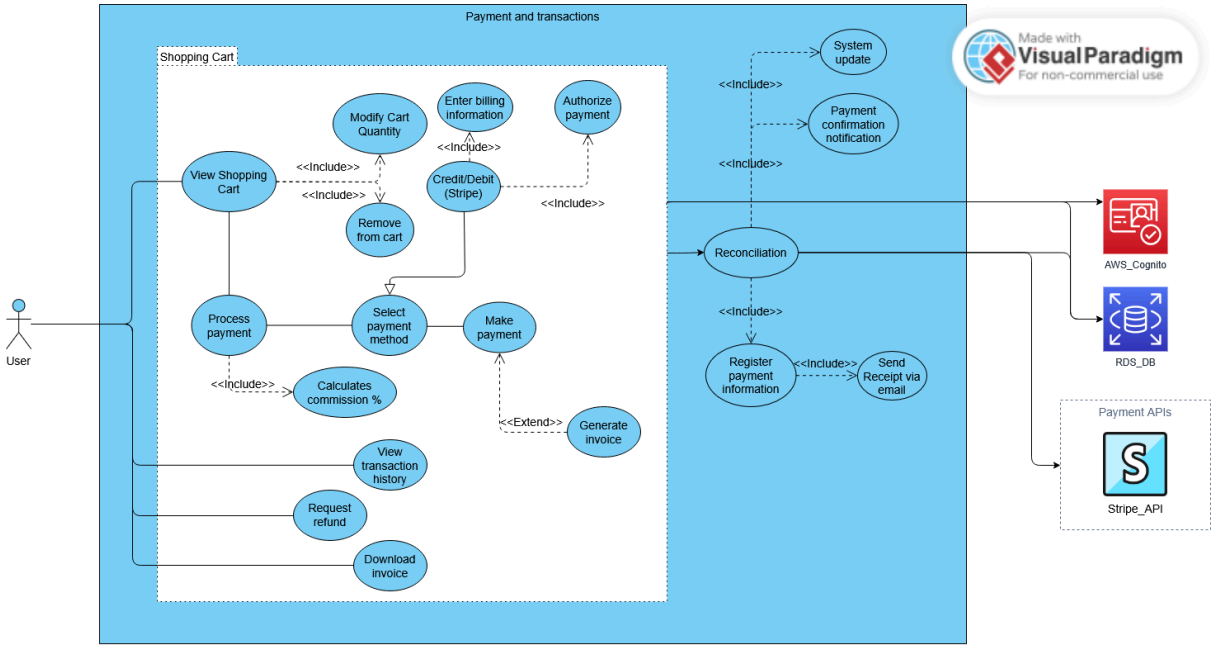

SaaS Subscription
Subscriptions
Select Subscription plan
<<Extend>>
Make payment
Select Free plan
<<Include>>
Select Pro plan
Update User permits
<<Include>>
Select Enterprise plan
<<Include>>
Upgrade subscription plan
<<Extend>>
Update payments
From free to pro
<<Include>>
Update User permits
From pro to enterprise
<<Include>>
Update payment method
Cancel subscription
Verify enterprise plan
Verify user subscription
Verify Pro plan
Metrics
View environmental metrics
View CO2 avoided
View waste diverted
View advanced metrics
Increased post visibility
User
Subscriptions_DB
External Billing
Stripe_API
Made with VisualParadigm
For non-commercial use

2.2.2.7 Transactions and payments

Table 7. Payments and transactions

Name	Payments and transactions
Objective	Allows the users to complete the payments of any of the required paid services in the platform. Generates an invoice and is sent to the user for transparency.
Actor	Authenticated user
Pre-condition	The user must be logged in to the platform. The user must have a shopping cart with various items to proceed with the payment.
Main scenario	1. The user enters the shopping cart view, here there are the options of modify the cart quantity (per item), and remove an item. Any modification is updated in the system. 2. The user proceeds to the payment processing, where all items in the current shopping cart are extracted and loaded to calculate the costs. The system validates the user credentials via Amazon Cognito. 3. The system calculates a commission percentage per transaction and updates the payment (what the platform earns). 4. The system redirects to the selection of the payment method (Stripe). 5. Once a method is selected, the selected method connects to the corresponding payment API for reconciliation, and the payment information is uploaded to the database. 6. The system sends a notification to the user, along with the receipt via email. 7. The system updates and sends the seller the corresponding payment and the billing information.
Alternative scenario	*View transaction history:* 1. The user can access their transaction history. 2. The system shows all of the transactions and the status. *Request refund:* 1. The user can solicit a refund if the product isn't up to the agreed standards. 2. The system begins the refund process. *Invoices:* 1. The system allows the user to download the invoice by retrieving the PDF from the database.

Post-conditions	The payment is processed and confirmed. The transaction is registered in the database. Inventory updates and the seller payout is queued, a notification is sent as well. The invoice is generated and emailed. The session persists the cart data when processing.

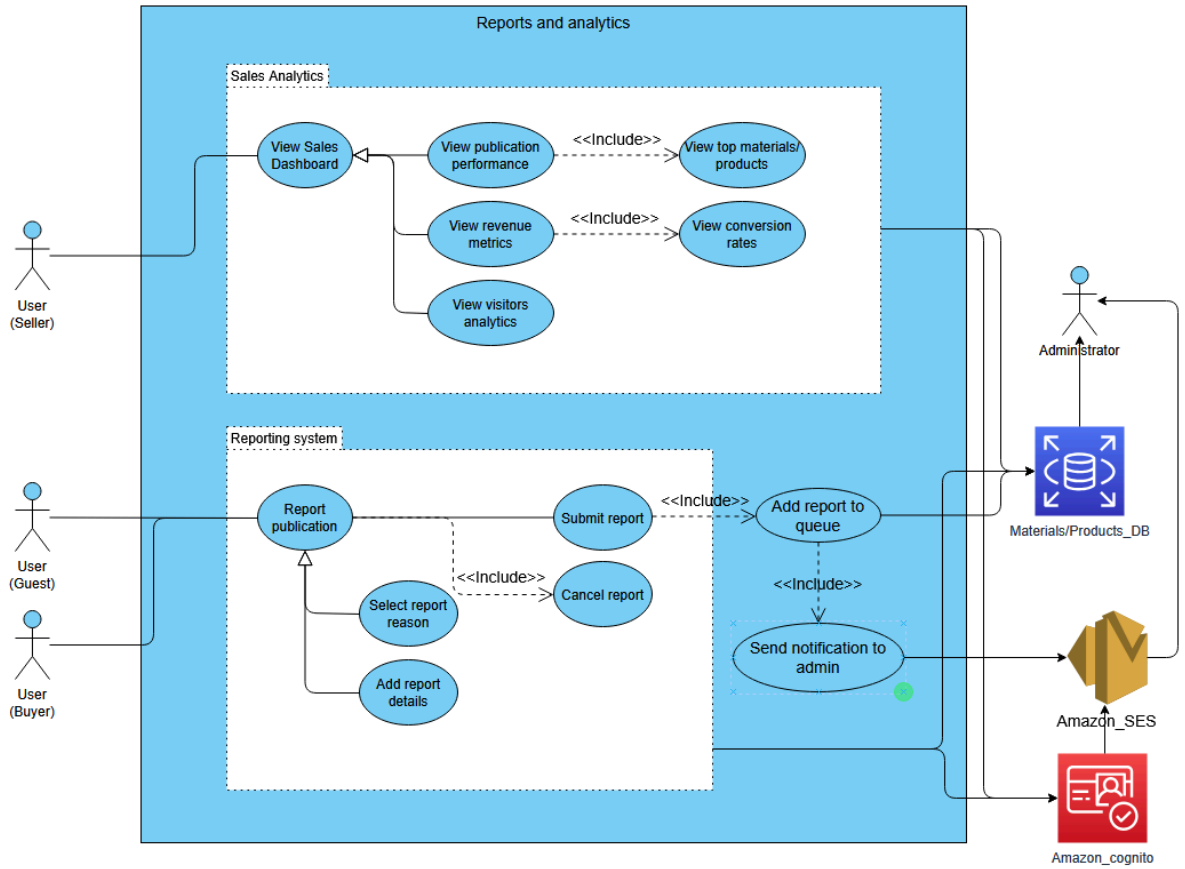


2.2.2.8 Reports and analytics

Table 8. Reports and analytics

Name	Reports and analytics
Objective	Provides sellers with comprehensive analytics dashboard overview showing sales performance, revenue, visitors, and top products. Allows users to report inappropriate, fraudulent, or policy-violating content to admins for review.
Actor	Seller, Buyer, Guest
Pre-condition	The platform must be available. *Sales analytics:* The user must be authenticated. The user must have at least one material/product sold. *Reporting:* The user can be authenticated or not. The user must be in the detailed view of a product/material.
Main scenario	*Sales analytics:* 1. The seller enters the sales dashboard where it is possible to see

	the performance (top material/product), revenue metrics (conversion rates), and visitor analytics of a post. 2. The system connects the database information to the sales dashboard for accuracy. 3. The system uses Amazon Cognito to ensure user credentials verification, so that the seller only sees analytics from his posts. 4. The system updates the dashboard in real-time. *Reporting system:* 1. The user can select the option “report” on any post. 2. The system redirects to the report form (reason selection, report details). 3. The user can submit the report when they fill the form. 4. The system is updated and the data is stored in the database.
Alternative scenario	*Reporting:* 1. The user cancels a report. 2. The system deletes the stored data from the report form and stops the report process.
Post-conditions	The sales dashboard is updated in real-time correctly. The report is successfully added to the admin queue for review. A notification is sent to the admins for the new reports, and successfully received.



2.3 User Characteristics

The system is designed for a broad audience, allowing users to assume one or more of the following roles. User characteristics vary widely and are not limited by technical proficiency.

Material Seller (General Public and Businesses):

- **Role:** Supplier of materials in the Materials Marketplace.
- **Description:** Any individual or entity from someone with leftover materials from a personal project to a manufacturing company who wishes to sell surplus materials that remain usable and valuable.

Material Buyer (Artisans, Hobbyists, General Public):

- **Role:** Purchaser of materials in the Materials Marketplace.
- **Description:** Any user, artisan, workshop, student, or individual, seeking to acquire waste or surplus materials for various purposes such as professional transformation, personal projects, or DIY initiatives.

Product Seller (Artisans, Creators):

- **Role:** Seller in the Products Marketplace.

- **Description:** Any user who creates products made from recycled or repurposed materials. These users are not required to source their materials from the W2T platform, but must be able to certify the sustainable origin of their products.

Product Buyer (General Public):

- **Role:** Buyer in the Products Marketplace.
- **Description:** Members of the general public interested in purchasing sustainable, unique, and meaningful products. These users may also choose to participate as Material Buyers or Sellers if they wish.

Administrator (Admin Users):

- **Role:** Waste-To-Treasure technical and support staff.
- **Description:** Users with elevated privileges and technical knowledge responsible for system administration, user management, transaction monitoring, and overall platform maintenance.

2.4 Constraints

The development and operation of the **Waste-To-Treasure** platform are subject to the following constraints:

- **Cloud Infrastructure Dependency:**
  The system is designed to operate entirely on Amazon Web Services (AWS). Its performance and availability depend on services such as AWS Amplify, Amazon EC2, Amazon RDS (PostgreSQL), and Amazon S3.
- **Third-Party API Dependency:**
  Payment processing and commission handling rely on the integration and continuous availability of external payment gateway APIs, such as Stripe.
- **Browser-Based Platform:**
  The frontend is a Next.js Web Application, requiring users to access the platform through a modern web browser that supports current web standards.
- **Internet Connectivity:**
  As a cloud-hosted SaaS platform, all user operations depend on a stable internet connection for proper functionality across all roles.
- **Marketplace Critical Mass:**
  The platform's long-term success depends on achieving a critical mass of users on both sides of the marketplace:
  (1) those who regularly list reusable materials, and
  (2) those who purchase materials and/or upcycled products.
- **Adoption Curve:**
  Adoption by large-scale material suppliers or industrial companies may progress

slowly, as these users often require additional validation, logistics, or internal approvals before adopting new digital platforms.

- **External Logistics:**
  The platform facilitates the connection between buyers and sellers but does not handle the specialized logistics or transportation of materials. These processes may be costly or complex for users and remain outside the platform's operational scope.
- **Regulatory Compliance:**
  The system must comply with applicable environmental regulations and data protection laws, ensuring that user information and business transactions adhere to legal and ethical standards.

2.5 Assumptions and dependencies

2.5.1 Assumptions

The following assumptions are fundamental to the design, development, and operation of the **Waste-To-Treasure (W2T)** platform. Any deviation from these assumptions could impact the project's schedule, cost, or functionality.

- **Internet Access:**
  It is assumed that users have reliable internet access from their mobile devices or computers, as all platform functionalities are delivered online.
- **Basic Web Platform Literacy:**
  It is assumed that users possess basic knowledge of how to navigate and interact with modern web applications.
- **Visual Content Quality:**
  It is assumed that users are capable of capturing images of sufficient quality to properly document the materials and products they publish.
- **Legal Authorization for Materials:**
  It is assumed that companies and organizations have the legal right to sell or donate their surplus materials, without requiring special permits during the initial operational phase.
- **Material Nature:**
  It is assumed that all listed materials are reusable and do not constitute hazardous waste requiring specialized handling or disposal procedures.

2.5.2 Dependencies

The continuous operation and success of **Waste-To-Treasure** rely on the following external services and regulatory frameworks:

- **Amazon Web Services (AWS):**
  The platform fully depends on **AWS Cloud Services** for hosting, data storage, processing, and scalability.
- **External Payment Gateways:**
  The platform relies on the integration and availability of third-party payment APIs such as **Stripe**, which must support the **Mexican banking system** for payment processing and commission management.
- **Database Performance (PostgreSQL on RDS):**
  The system depends on PostgreSQL hosted on Amazon RDS to efficiently handle complex search and filtering queries without performance degradation.
- **Regulatory and Fiscal Compliance:**
- **Integration with the Mexican Tax Administration Service (SAT):** Required for implementing electronic invoicing (facturación electrónica) once transaction volumes reach a defined threshold, ensuring tax compliance.
- **Continuous Compliance with the Federal Law on Protection of Personal Data (LFPDPPP):** The platform must remain in full compliance with Mexico's Federal Law on the Protection of Personal Data Held by Private Parties (LFPDPPP) to ensure the lawful handling of user data.

2.6 Business Rules

This section defines the business policies, operational constraints, and logical rules that the Waste to Treasure (W2T) platform must enforce to ensure proper alignment with the intended business model.

3.6.1 General and Role Rules

- The system shall operate under a Software as a Service (SaaS) model, enabling user connections for the buying and selling of material and products.
- The system must allow any registered user to assume one or more marketplace roles based on their actions "Material Seller", "Material Buyer", "Product Seller", and "Product Buyer".
- There must be an "Administrator" role with elevated privileges for complete platform, user, and transcription management.
- The registration and authentication of all roles must be managed through Amazon Cognito.

2.6.2 Marketplace and Publication Rules

- The system must manage two integrated markets: a "Material Marketplace" (for surplus and waste materials) and a "Products Marketplace" (for finished goods).
- Any registered user (individual, workshop, or company) may create Material listings in the Material Marketplace.

- Any registered user may create Product listings in the products Marketplace.
- All listings in the Product marketplace must include a description certifying their recycled or reused origin, regardless of whether the material was obtained through the platform.
- All new listings (both materials and products) must be approved by an Administrator before becoming publicly visible in the catalog. The default status must be "Pending Approval".
- The system must prevent the purchase of any product or material if its inventory (stock) is zero.

2.6.3 Transaction and Monetization Rules

- A variable commission (estimated at 10%) must be applied to the final sale price of every completed transaction in both marketplaces.
- All monetary transactions purchased and subscriptions must be processed through the integrated payment gateways (Stripe).
- An order is considered "Paid" only after a successful confirmation is received from the payment gateway API.
- users must have access to a detailed history of all their transactions (both purchase and sales).

2.6.4 SaaS Subscription Rules

- The system must offer a "Free Plan" to all users, allowing basic buying and selling operations.
- The system must offer Premium Subscription Plans (SaaS) that enable advanced functionalities.
- Advanced features such as "increased visibility", "custom impact reports", and "advance metrics" must only be available to users with an active premium subscription.

2.6.5 Administration and Moderation Rules

- The administrator is the only role authorized to approve, reject or remove any user's publication
- The administrator is the only role authorized to manage marketplace categories (CRUD operations).
- The administrator is the only role authorized to manage users (block, change, role, delete).
- Reported posts or users must appear in a priority moderation queue for the *Administrator*.

2.6.6 Reputation and Compliance Rules

- A user (buyer) may leave a rating and comment (review) about a seller only if a transaction with that seller has been successfully completed and received.
- Any authenticated user may report publication or user profile.
- The system must store and manage personal data in compliance with the Federal Law on the Protection of Personal Data (LFPDPPP)
- The system must be designed for potential integrations with the SAT and to report environmental compliance metrics.

3 Specific requirements

3.1 External interfaces

3.1.1 User Interface

Waste-To-Treasure (W2T) is a web-based system. The user interfaces are divided into two main web applications:

Marketplace Portal

This is the primary interface for all public users of the platform, including Material Sellers, Material Buyers, Product Sellers, and Product Buyers.

- Developed as a Next.js Web Application.
- Allows users to register, manage their profiles, publish materials or products, browse catalogs, and manage their transactions.
- The interface must be responsive and fully compatible with modern web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge.

Administration Console

This is the main web interface for **Administrator Users**.

- Enables user management, content moderation across both marketplaces, transaction monitoring, subscription management, and platform analytics.
- Designed with an intuitive dashboard layout to facilitate administrative oversight and operational efficiency.

3.1.1.1 Visual Design and Style Guidelines

To ensure a **consistent brand identity** and a **pleasant user experience**, the following visual style elements have been defined:

Color Palette

Primary Colors

- **#396539 (**Dark Green**):** Represents sustainability, nature, and growth. Used for key brand elements and primary call-to-action buttons.
- **#294730 (**Forest Green**):** A complementary shade of green that adds depth and balance to the interface.

Secondary Colors

- **#69391E (**Earth Brown):**):** Suggests organic materials, authenticity, and warmth.
- **#A2704F (**Copper/Terracotta**):** A vibrant color used to highlight important information or interactive elements.

Neutral Colors

- **#262C32 (Dark Gray):** Used for primary text and backgrounds requiring contrast.
- **#FCFCFC (Off White):** Provides clean, open backgrounds for light UI components.
- **#F3F3F3 (Light Gray):** Used to create a significant contrast with the visual elements without falling into darkness.
- **#000000 (Pure Black):** Used for high-contrast text and specific graphic accents.

Typography

Font selection is focused on legibility, accessibility, and a professional appearance across all devices.

- **Poppins (Headings and Titles):** A geometric and modern font offering excellent readability at large sizes, contributing to a contemporary and clean aesthetic.
- **Roboto (Subheadings and Highlighted Text):** A highly versatile and legible font, ideal for guiding the reader through key interface sections.
- **Inter (Body Text):** Designed specifically for user interfaces, Inter provides superior flexibility at small sizes and ensures comfortable reading in long text blocks.

3.1.2 Hardware Interfaces

No specific hardware interfaces are required beyond standard user devices such as desktop computers, laptops, tablets, or smartphones equipped with a modern web browser to access the web applications.

3.1.3 Software Interfaces and Architecture

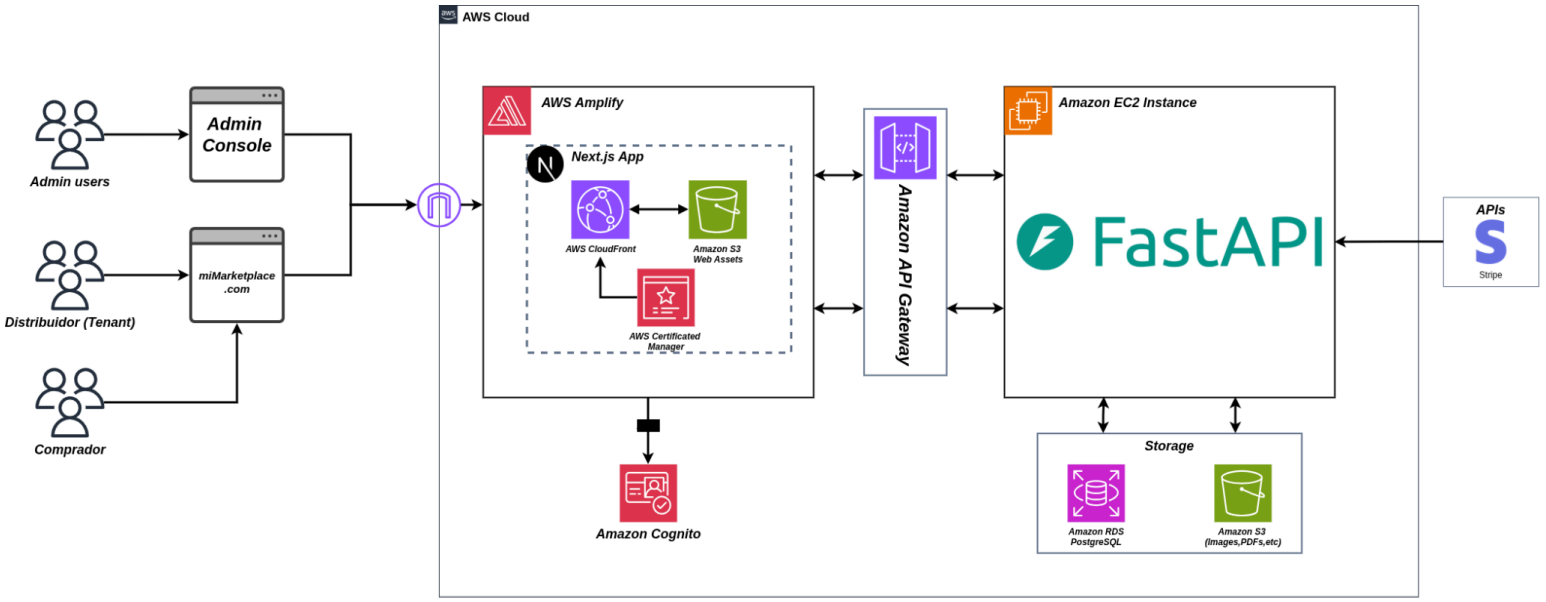
The Waste-To-Treasure platform integrates with multiple third-party APIs and services to support critical functionalities and is built on a cloud-native architecture hosted on Amazon Web Services (AWS).

External APIs and Services

- **Payment Processing:**
  - **Stripe:** Used for processing credit and debit card payments, as well as managing transaction commissions.

Infrastructure: AWS-Based Architecture

The system architecture is designed to be decoupled, scalable, and secure, leveraging managed services within the AWS ecosystem, as illustrated in the Architecture Diagram (Figure 1.1).



The main services and components of this architecture include:

- **AWS Amplify:**
  The frontend is a Next.js Web Application hosted and deployed through AWS Amplify. This service manages continuous integration and deployment (CI/CD) and

content distribution via AWS CloudFront, using Amazon S3 for static web assets and AWS Certificate Manager for SSL/TLS security.

- **Amazon Cognito:**
  Handles user management, registration, and authentication for all user roles. It provides OAuth 2.0 tokens and AWS credentials for secure access, authenticating users through an integrated user directory managed within Cognito.
- **Amazon EC2 Instance:**
  Hosts the backend application, which is developed using FastAPI. This virtual machine (VM) environment executes the business logic and server operations, ensuring high-performance and efficient processing.
- **Amazon API Gateway:**
  Serves as the secure and scalable single entry point for all communication between the Next.js frontend and the FastAPI backend.
- **Amazon RDS and Amazon S3:**
  Responsible for data storage and management, divided as follows:
  - Amazon RDS (PostgreSQL): A managed relational database used to store all structured data, including user information, listings, and transactions.
  - Amazon S3 (Object Storage): Used for storing unstructured data such as product images, PDFs, and material photos uploaded by users.

3.1.4 Communication Protocols

The system employs the following communication protocols to ensure secure and efficient data exchange:

- **HTTP/HTTPS:**
  All communication between the client (web browser) and the server (AWS infrastructure) is conducted over HTTPS using TLS encryption, managed through AWS Certificate Manager, to guarantee the confidentiality and integrity of data in transit.
- **RESTful APIs:**
  Interactions between the Next.js frontend and the FastAPI backend are handled through RESTful APIs, securely exposed via Amazon API Gateway to ensure reliable and scalable communication between system components.

3.2 Functional Requirements

3.2.1 Functional requirement 1

Table 1. Functional requirement 1

Field	Description

ID	RF-1
Actor	User (Unauthenticated or Authenticated)
Description	Manages the user's authentication lifecycle, including account creation (Register), session validation (Login), session termination (Logout), and password recovery.
Pre-conditions	The user is accessing the application.
Flow of events	1. The user selects "Login", "Register", or "Forgot Password". 2. System (via Amazon Cognito) presents the corresponding form. 3. The user submits credentials. 4. The system validates the information. 5. The system grants a session (JWT) or sends a recovery link.
Exceptions	1. Invalid credentials (email/password mismatch). 2. User email does not exist (for login). 3. User email already exists (for registration). 4. The account is locked or unconfirmed.
Post-conditions	The user is successfully authenticated and redirected to their dashboard, or an appropriate error message is displayed.

3.2.2 Functional requirement 2

Table 2. Functional requirement 2

Field	Description
ID	RF-2
Actor	User (Authenticated)
Description	Allows an authenticated user to manage their public-facing profile, including personal data (name, bio) and uploading/changing their profile picture.
Pre-conditions	The user is authenticated and has navigated to the "Profile Management" section.
Flow of events	1. The system displays the user's current profile information. 2. The user modifies the text fields (e.g., name, bio). 3. The user uploads a new profile picture (image file). 4. User clicks "Save Changes". 5. The system validates inputs. 6. The system updates the user record in PostgreSQL and the image file in Amazon S3.

Exceptions	1. Invalid data format (e.g., name is too long). 2. Image upload fails (e.g., file too large, invalid file type).
Post-conditions	The user's profile information is updated and visible on their public profile.

3.2.3 Functional requirement 3

Table 3. Functional requirement 3

Field	Description
ID	RF-3
Actor	Supplier
Description	Provides a dedicated form for a Supplier to create and publish a new listing for industrial materials (B2B Marketplace).
Pre-conditions	The user is authenticated and has a "Supplier" role.
Flow of events	1. Supplier navigates to "Publish New Material". 2. The system displays a form with all required fields (title, description, category, quantity, price, location, images). 3. Supplier fills out the form and submits. 4. The system validates all data fields.
Exceptions	1. Supplier submits the form with missing required fields. 2. Invalid data types are entered (e.g., text in a price field).
Post-conditions	A new material listing is created in the database, potentially with a "Pending Approval" status.

3.2.4 Functional requirement 4

Table 4. Functional requirement 4

Field	Description
ID	RF-4
Actor	Supplier
Description	Provides a dashboard for Sellers to manage their own published material listings (View, Edit, Deactivate/Activate).
Pre-conditions	The user is authenticated as a "Supplier".

Flow of events	1. Supplier navigates to the "My Materials" dashboard. 2. The system displays a list of all materials published by this seller. 3. Supplier selects an action for a specific listing (e.g., "Edit", "Deactivate"). 4. The system performs the requested operation (e.g., opens the RF-3 form for editing, or updates the listing's "active" status in the database).
Exceptions	1. Supplier attempts to edit a listing that does not belong to them.
Post-conditions	The material listing is updated, or its visibility status is changed.

3.2.5 Functional requirement 5

Table 5. Functional requirement 5

Field	Description
ID	RF-5
Actor	User (Buyer or Guest)
Description	Displays a public catalog of all active materials (B2B Marketplace), allowing users to perform text-based searches and apply advanced filters (e.g., category, location, price).
Pre-conditions	Users access the "Materials Marketplace".
Flow of events	1. The system displays a grid/list of materials, paginated. 2. The user enters text in the search bar or selects filter options. 3. The frontend sends an API request to FastAPI. 4. The backend queries PostgreSQL and returns the filtered list of results.
Exceptions	1. No results match the filter criteria.
Post-conditions	The material catalog view is updated to show only the matching results.

3.2.6 Functional requirement 6

Table 6. Functional requirement 6

Field	Description

ID	RF-6
Actor	User (Buyer or Guest)
Description	Displays a detailed page with all information for a specific material listing, including description, all images, price, seller information, and an "Add to Cart" button.
Pre-conditions	The user clicks on a material listing from the catalog (RF-5).
Flow of events	1. The system retrieves the unique ID of the selected material. 2. The system fetches all details for that material from the API. 3. The system renders the complete information on the page.
Exceptions	1. Listing not found (e.g., deactivated or deleted by the seller).
Post-conditions	Users can view all details of the material.

3.2.7 Functional requirement 7

Table 7. Functional requirement 7

Field	Description
ID	RF-7
Actor	Buyer (Authenticated User)
Description	Facilitates the transaction for a Buyer to acquire materials. This requirement specifically covers adding the material to the cart.
Pre-conditions	The user is authenticated and viewing a material's detail page (RF-6).
Flow of events	1. The user selects a quantity and clicks "Add to Cart". 2. The system verifies the item is in stock. 3. The system adds the item to the user's persistent shopping cart.
Exceptions	1. Items are out of stock or requested quantity exceeds stock. 2. The user is not authenticated.
Post-conditions	The item is added to the user's cart, and the cart icon updates. (Note: The full checkout is handled by RF-11).

3.2.8 Functional requirement 8

Table 8. Functional requirement 8

Field	Description
ID	RF-8
Actor	Vendor
Description	Provides a form to vendors so they can fill with product information (standardized data) and sell on the platform.
Pre-conditions	The user is authenticated as a "Vendor".
Flow of events	1. Vendor navigates to "Publish New Product". 2. The system displays a form with all required fields (title, description, category, stock, price, location, images). 3. Vendor fills out the form and submits. 4. The system validates all data fields.
Exceptions	1. Vendor submits the form with missing required fields. 2. Invalid data types are entered (e.g., text in a price field).
Post-conditions	A new product listing is created in the database, potentially with a "Pending Approval" status.

3.2.9 Functional requirement 9

Table 9. Functional requirement 9

Field	Description
ID	RF-9
Actor	Vendor
Description	Provides a dashboard for Vendors to manage their own published product listings (View, Edit, Deactivate/Activate).
Pre-conditions	The user is authenticated as a "Vendor".
Flow of events	1. Vendor navigates to the "My Products" dashboard. 2. The system displays a list of all materials published by this vendor. 3. Vendor selects an action for a specific listing (e.g., "Edit", "Deactivate").

	4. The system performs the requested operation (e.g., opens the RF-8 form for editing, or updates the listing's "active" status in the database).
Exceptions	1. Vendor attempts to edit a listing that does not belong to them.
Post-conditions	The product listing is updated, or its visibility status is changed.

3.2.10 Functional requirement 10

Table 10. Functional requirement 10

Field	Description
ID	RF-10
Actor	User (Buyer or Guest)
Description	The user can view a product catalog with search bar and filter options for better navigation.
Pre-conditions	The user is authenticated or unauthenticated.
Flow of events	1. The user enters the “Product Catalog” section. 2. The system displays a list of all products published by all vendors, from newest to oldest. 3. The user can use the search bar to filter the products in the system (e.g., search chairs, tables, etc.) 4. The user can select a product to see the detailed information for that specific product.
Exceptions	1. The user searches for a product which isn't published on the system.
Post-conditions	1. Unfiltered: The products list from newest to oldest. 2. Filtered: The products list matches the active filter from newest to oldest.

3.2.11 Functional requirement 11

Table 11. Functional requirement 11

Field	Description
ID	RF-11

Actor	User (Buyer or Guest)
Description	The user can access the shopping cart to pay for the products that were added to the checkout.
Pre-conditions	The user is authenticated or unauthenticated and they have products/materials added to the shopping cart.
Flow of events	1. The user adds some products/materials to the shopping cart. 2. The user accesses the shopping cart page. 3. The user can edit the products/materials in the shopping cart (delete or change quantity). 4. The user can proceed to the payment page to complete the transactions for the products/materials.
Exceptions	1. The user tries to add products out of stock. 2. The user doesn't have any product/material added to the shopping cart.
Post-conditions	1. Products added successfully to the shopping cart. 2. The shopping cart allows the user to proceed with the payment.

3.2.12 Functional requirement 12

Table 12. Functional requirement 12

Field	Description
ID	RF-12
Actor	User (Buyer or Guest)
Description	The user can view the Vendor's profile from the product's information page.
Pre-conditions	The user is looking through the product's marketplace and accessed the "More Information" section of a product.
Flow of events	1. The user browses the product's marketplace. 2. The user finds a product that he is interested in, and opens the "More Information" page for that product. 3. The product's information page shows the public profile of the Vendor.
Exceptions	1. The user is browsing on the material's marketplace.
Post-conditions	The user can see the Vendor's public profile and related data (Name, profile picture).

3.2.13 Functional requirement 13

Table 13. Functional requirement 13

Field	Description
ID	RF-13
Actor	Buyer
Description	The buyer can leave a rating to the material/product they bought as well as a comment to go along.
Pre-conditions	The user is authenticated as a “Buyer” and has purchased, and received a product/material.
Flow of events	1. The buyer has finalized the process of purchase, and received the product/material. 2. The system sends an email to the buyer so they can leave a review of the purchased goods. 3. The buyer writes a comment and rates the product/material, and publishes to the platform. 4. The system updates the platform with the new review for that product so it’s visible to all users.
Exceptions	1. The user is not authenticated. 2. The buyer didn’t buy anything on the platform.
Post-conditions	The buyer can leave a review with a rating of the purchased goods and publish it to the platform.

3.2.14 Functional requirement 14

Table 14. Functional requirement 14

Field	Description
ID	FR-14
Actor	User (Authenticated)
Description	Allow the user to submit reports about inappropriate content (posts) or users.
Pre-conditions	- The user is authenticated in the system. - The user is viewing a post or a user's profile.

Flow of events	1. The user selects the "Report" option on a post or profile. 2. The system displays a form requesting the reason for the report. 3. The user selects a reason and (optionally) adds details. 4. The user submits the report. 5. The system logs the report and queues it for administrative review.
Exceptions	1. The user tries to submit a report without selecting a reason.
Post-conditions	The report is submitted and is pending review by an Administrator.

3.2.15 Functional requirement 15

Table 15. Functional requirement 15

Field	Description
ID	FR-15
Actor	Administrator
Description	Provide an administrative dashboard with key platform statistics.
Pre-conditions	- The user is authenticated and has "Administrator" privileges. - The user has accessed the Admin Console.
Flow of events	1. The Administrator enters the main section of the Admin Console. 2. The system retrieves and displays key metrics (e.g., new users, recent transactions, pending reports, new posts).
Exceptions	1. Error loading the data for the statistics.
Post-conditions	The Administrator views the general status and key metrics of the platform.

3.2.16 Functional requirement 16

Table 16. Functional requirement 16

Field	Description
ID	FR-15
Actor	Administrator

Description	Allow administrators to moderate posts (approve, reject, or delete) in both marketplaces.
Pre-conditions	- The user is authenticated as an Administrator. - There are posts pending approval or that have been reported (FR-15).
Flow of events	1. The Administrator navigates to the "Content Moderation" queue. 2. The system displays the list of pending posts. 3. The Administrator selects a post to review. 4. The Administrator applies an action: "Approve" (makes it public), "Reject" (hides it and notifies the seller), or "Delete". 5. The system updates the post's status.
Exceptions	1. The post was deleted by the seller before being moderated.
Post-conditions	The post's visibility status is updated in the database.

3.2.17 Functional requirement 17

Table 17. Functional requirement 17

Field	Description
ID	FR-17
Actor	Administrator
Description	Allow administrators to manage (CRUD actions) the categories for both marketplaces.
Pre-conditions	- The user is authenticated as an Administrator.
Flow of events	1. The Administrator navigates to the "Category" section. 2. The system displays the list of existing categories. 3. The Administrator selects "Create" and fills out the form, or selects an existing category to "Update" or "Delete". 4. The system validates the data. 5. The system applies the changes to the database.
Exceptions	1. The Administrator attempts to delete a category that is being used by active posts. 2. The Administrator attempts to create a category that already exists.
Post-conditions	The list of available categories for posts is updated.

3.2.18 Functional requirement 18

Table 18. Functional requirement 18

Field	Description
ID	FR-18
Actor	Administrator
Description	Allow administrators to manage user profiles (view, block, change role).
Pre-conditions	- The user is authenticated as an Administrator.
Flow of events	1. 1. The Administrator navigates to the "User Management" section. 2. The system displays a searchable list of all registered users. 3. The Administrator searches for and selects a user. 4. The system displays the user's profile and available actions (e.g., "Block", "Unblock", "Assign Role"). 5. The Administrator selects an action and confirms. 6. The system updates the user's status or role.
Exceptions	1. El administrador intenta bloquearse a sí mismo o algún otro administrador del sistema. 2. The Administrator tries to block himself or any other system’s Administrator.
Post-conditions	The status or permissions of the managed user are updated.

3.2.19 Functional requirement 19

Table 19. Functional requirement 19

Field	Description
ID	FR-19
Actor	Administrator
Description	Provide a queue to review and manage reports submitted by users (RF-15).
Pre-conditions	- The user is authenticated as an Administrator. - There are pending reports to be reviewed.

Flow of events	1. The Administrator navigates to the "Reports". 2. The system displays the list of unresolved reports. 3. The Administrator selects a report. 4. The system displays the report details, the reason, and the link to the reported content or user. 5. The Administrator takes an action (e.g., "Dismiss Report", "Proceed to Moderate Content (RF-17)", "Proceed to Manage User (RF-19)").
Exceptions	1. The reported content or user no longer exists.
Post-conditions	The report is marked as "Resolved" and the corresponding action (if any) is executed.

3.2.20 Functional requirement 20

Table 20. Functional requirement 20

Field	Description
ID	FR-20
Actor	Administrator
Description	Allow administrators to supervise transactions and mediate disputes between buyers and sellers.
Pre-conditions	- The user is authenticated as an Administrator. - A buyer or seller has escalated a transaction to "Dispute".
Flow of events	1. The Administrator navigates to the "Dispute Management" queue. 2. The system displays active disputes. 3. The Administrator selects a dispute to review the transaction details, communication between parties, and the reason for the dispute. 4. The Administrator issues a ruling (e.g., "Refund Buyer", "Release payment to Seller", "Reject Dispute"). 5. The system processes the ruling.
Exceptions	1. Error processing a refund via the payment API.
Post-conditions	The dispute is marked as "Closed" and the transaction status is updated.

3.2.21 Functional requirement 21

Table 21. Functional requirement 21

Field	Description
ID	RF-21
Actor	Distributor / Seller
Description	The system will allow sellers to access a free plan with basic platform functionalities.
Pre-conditions	The registered user has the role of "distributor" or “seller”.
Flow of events	1. The user selects the free plan option during registration or in their account settings. 2. The system activates the free plan for the user. 3. The user can publish materials or products with the limitations of the free plants.
Exceptions	None.
Post-conditions	The user has an active free plan and can use the associated functionalities.

3.2.22 Functional requirement 22

Table 22. Functional requirement 22

Field	Description
ID	RF-22
Actor	Distributor / Seller
Description	The system will offer premium plans (Pro and business) with advanced functionalities, preferably for sellers.
Pre-conditions	The registered user has the role of “distributor” or “seller”
Flow of events	1. The user navigates to the premium plans section. 2. The system displays the different premium plans available, their features, and prices. 3. The user selects a premium plan and proceeds to payment. 4. The system processes the payment and activates the advanced functionalities for the user.

Exceptions	Payment processing failure.
Post-conditions	The user has an active premium plan and access to advanced functionalities.

3.2.23 Functional requirement 23

Table 23. Functional requirement 23

Field	Description
ID	R-23
Actor	System, Distributor / Seller and Buyer
Description	The system will calculate and process variable commissions (10%) for each completed transaction.
Pre-conditions	A sales transaction has been confirmed between a buyer and a seller.
Flow of events	1. A buyer makes a purchase. 2. The seller confirms the transactions. 3. The system calculates a 10% commission on the total transaction value. 4. The system retains the commission from the seller's payment before settlement.
Exceptions	Error in commission calculation. Failure to retain commission.
Post-conditions	The commission has been calculated and processed correctly, and the remaining amount is prepared for settlement to the seller.

3.2.24 Functional requirement 24

Table 24. Functional requirement 24

Field	Description
ID	R-24
Actor	System
Description	The system will integrate with external payment gateways (Stripe) to process transactions.
Pre-conditions	Payment gateways are configured and available.

Flow of events	1. A user initiates a payment (e.g., product purchase, premium subscription). 2. The system redirects to the selected payment gateway (Stripe). 3. The payment gateway processes the transaction. 4. The payment gateway notifies the system of the transaction status. 5. The system updates the transaction status internally.
Exceptions	Connection failure with the payment gateway. Transaction rejected by the payment gateway.
Post-conditions	The payment has been processed successfully or the transaction failure has been recorded.

3.2.25 Functional requirement 25

Table 25. Functional requirement 25

Field	Description
ID	R-25
Actor	Any User (has made at least one transaction.)
Description	The system will provide a detailed history of all transactions made by the user.
Pre-conditions	The user has logged into the system and has made at least one transaction.
Flow of events	1. The user navigates to their “Transaction History” section. 2. The system retrieves and displays a list of all transactions associated with the user, including details such as date, amount, items, and status. 3. The user can filter or search for specific transactions.
Exceptions	None.
Post-conditions	The user can view their transaction history clearly and in detail.

3.2.26 Functional requirement 26

Table 26. Functional requirement 26

Field	Description
ID	R-26
Actor	Distributor / Seller and Buyer

Description	The system will include a Frequently Asked Questions (FAQ) section for self-help and resolution of common queries.
Pre-conditions	The user has access to the platform.
Flow of events	1. The user navigates to the “Frequently Asked Questions” section. 2. The system displays a list of questions and answers organized by categories. 3. The user can search for specific questions or browse through the categories.
Exceptions	None
Post-conditions	The user can find answers to their common questions without needing to contact support.

3.3 Non-functional requirements

3.3.1 Non-functional requirement 1

Table 27. Non-functional requirement 1

Field	Description
ID	NFR-1
Actor	End-User
Description	(Performance) Ensures a fast page load experience. The Largest Contentful Paint (LCP) for main pages must be under 3.0 seconds.
Pre-conditions	The user initiates navigation to a main page (e.g., catalog, detail page).
Main scenario	1. The user's browser requests the page. 2. The system serves all necessary assets (HTML, CSS, JS, images). 3. The largest content element on the user's screen becomes visible. 4. This entire process (from step 1 to 3) completes in under 3.0 seconds.
Exceptions	1. The user's network connection is exceptionally slow.
Post-conditions	The page is fully interactive, and the user perceives the load time as fast.

3.3.2 Non-functional requirement 2

Table 28. Non-functional requirement 2

Field	Description
ID	NFR-2
Actor	End-User / System
Description	(Performance) Ensures a responsive API for searching and filtering. 95% of API requests for catalog searches must complete in under 800 milliseconds.
Pre-conditions	The user applies a filter or submits a search query on a catalog page (triggers RF-5).
Main scenario	1. The frontend client sends an API request to the FastAPI backend. 2. The backend queries the PostgreSQL database. 3. The backend formats the JSON response. 4. The backend sends the response back to the client. 5. This server-side process (steps 2-4) completes in < 800ms.
Exceptions	1. The query is abnormally complex (e.g., searching across all possible fields with wildcards). 2. The database is under heavy load from other operations.
Post-conditions	The frontend receives the data quickly and can render the filtered results for the user without perceptible lag.

3.3.3 Non-functional requirement 3

Table 29. Non-functional requirement 3

Field	Description
ID	NFR-3
Actor	System
Description	(Scalability) The platform must handle the defined concurrent user load, supporting 100 concurrent users in read operations (browsing) and 10 concurrent users in write operations (buying, selling).
Pre-conditions	The system is operational and experiencing a peak load of 100 read users and 10 write users simultaneously.

Main scenario	1. All 110 concurrent users submit requests. 2. The system (EC2, RDS, API Gateway) processes all requests without crashing or timing out. 3. All read and write operations complete successfully. 4. The system continues to meet performance metrics defined in RNF-1 and RNF-2.
Exceptions	1. A sudden load spike exceeds this limit (e.g., 1000 users). 2. A specific write operation causes a database lock, slowing other operations.
Post-conditions	The system remains stable, available, and performant under the expected peak load.

3.3.4 Non-functional requirement 4

Table 30. Non-functional requirement 4

Field	Description
ID	NFR-4
Actor	System / End-User
Description	(Performance/Architecture) All static assets (images, CSS, JS) must be optimized and delivered via a Content Delivery Network (CDN), specifically AWS CloudFront.
Pre-conditions	A user (especially a geographically distant one) requests a page, which in turn requests static assets (e.g., logo.png, styles.css).
Main scenario	1. The user's browser requests logo.png. 2. The request is routed to the nearest AWS CloudFront edge location. 3. If the asset is cached, CloudFront serves it immediately. 4. If not cached, CloudFront retrieves it from the origin (S3 bucket), caches it, and then serves it. 5. The user receives the asset from the fast edge location, not the distant S3 origin.
Exceptions	1. The CDN cache is invalidated (e.g., new file deployment). The first request will be slower as it retrieves from the origin.
Post-conditions	The asset is delivered to the user quickly, reducing overall page load time. The load on the origin (S3) is minimized.

3.3.5 Non-functional requirement 5

Table 31. Non-functional requirement 5

Field	Description
ID	NFR-5
Actor	System
Description	The system must implement an integral security architecture that guarantees confidentiality, integrity, and data availability.
Pre-conditions	All communications between the client and the server must be done through HTTPS. Users must authenticate through Amazon Cognito to access all of the platform services. The Backend API must be accessible only with Amazon API Gateway.
Main scenario	1. The user tries to access the platform or interact with the API. 2. All requests are rerouted through HTTPS, ensuring data encryption during transit. 3. Amazon Cognito handles the user's authentication and authorization, emitting OAuth 2.0 tokens and AWS credentials. 4. Amazon API Gateway acts as an unique and safe entry point, validating the requests and rerouting them to the FastAPI backend allocated in Amazon EC2. 5. The system securely processes the request, making sure that only the authorized users can access the resources.
Exceptions	1. Amazon Cognito fails the credentials validation. 2. Unauthorized access attempts or malicious requests blocked by Amazon API Gateway. 3. SSL/TLS configuration problems that impede safe communication.
Post-conditions	1. All user interactions with the platform are protected by HTTPS. 2. Robust user authentication and managed by Amazon Cognito. 3. Backend API access is managed and secured by Amazon API Gateway.

3.3.6 Non-functional requirement 6

Table 32. Non-functional requirement 6

Field	Description
ID	NFR-6
Actor	System / End-User
Description	The platform must offer an intuitive and accessible user experience, ensuring that all users, regardless of their technical skill level, can navigate, interact, and complete their tasks efficiently and satisfactorily.
Pre-conditions	The user interface must be consistent in design and functionality across all sections of the platform. Interactive elements must be clearly identifiable and their purpose obvious. The platform must be compatible with modern web browsers and common devices (desktop, laptop, tablet, smartphone).
Main scenario	1. The user accesses the platform from any compatible device and browser. 2. The interface responsively adapts to the screen size, maintaining readability and functionality. 3. The user navigates through different sections (Materials, Products, Profile, etc.) using a clear and consistent menu structure. 4. Data input forms are easy to understand, with clear labels and real-time validation. 5. Key actions (publish, buy, edit) are completed with a minimum of steps and visual feedback. 6. The system provides clear and helpful error and confirmation messages.
Exceptions	1. Compatibility issues with very old or non-standard browsers. 2. Navigation difficulties due to an unstable internet connection. 3. User input errors that are not clearly communicated.
Post-conditions	1. Users can perform all main tasks (publish materials/products, buy, manage profile) autonomously and without frustration. 2. The platform receives positive ratings for ease of use and design. 3. The need for technical support related to interface usability is minimized.

3.3.7 Non-functional requirement 7

Table 33. Non-functional requirement 7

Field	Description
ID	NFR-7
Actor	System / End-User
Description	The platform must feature a responsive design that ensures an optimal viewing and interaction experience across a wide range of devices, including mobile phones, tablets, and desktop computers. The interface should automatically adapt its layout, images, and text to provide usability and accessibility regardless of screen size or orientation.
Pre-conditions	The platform's frontend is developed as a Next.js Web Application. Users access the platform through modern web browsers that support current web standards. All UI components are designed with flexibility in mind to accommodate different screen dimensions.
Main scenario	1. A user accesses the Waste-To-Treasure platform from any device (mobile, tablet, or desktop). 2. The system detects the screen size and orientation of the user's device. 3. The Next.js Web Application dynamically adjusts its layout, navigation menus, image sizes, and text formatting to fit the available screen space. 4. Interactive elements (buttons, forms) remain fully functional and easily accessible, optimizing for touch input on mobile and tablet devices, and mouse/keyboard input on desktops. 5. The user experiences a consistent and intuitive interface, regardless of the device used, without the need for horizontal scrolling or zooming.
Exceptions	1. Performance degradation on very old or low-power devices due to complex responsive rendering. 2. Minor layout discrepancies on obscure or non-standard browser/device combinations. 3. Slow loading times on poor internet connections, affecting the initial rendering of the responsive layout.
Post-conditions	1. The platform provides an excellent user experience on mobile, tablet, and desktop devices. 2. User engagement and satisfaction are high across all device types.

	3. The platform maintains a professional and modern appearance on any screen.

3.3.8 Non-functional requirement 8

Table 34. Non-functional requirement 8

Field	Description
ID	NFR-8
Actor	System
Description	The platform must adhere to a defined visual style guide that ensures a consistent brand identity and a pleasant user experience across all interfaces. This includes strict adherence to a specified color palette and typography rules for headings, subheadings, and body text.
Pre-conditions	All design and development teams have access to the official visual style guide documentation. UI/UX mockups and prototypes are approved according to the established style guidelines. The frontend development environment supports the implementation of custom fonts and color schemes.
Main scenario	1. During the design phase, all new UI components are created using the specified color palette and font families. 2. During development, CSS styles and other styling mechanisms are implemented to precisely match the defined colors (Primary, Secondary, Neutral) and typography (Poppins for Headings, Roboto for Subheadings, Inter for Body Text). 3. Automated visual regression tests are conducted to ensure consistency across different browsers and devices. 4. Any new content or features introduced to the platform automatically inherit the established visual styles.
Exceptions	1. Inconsistencies in visual rendering due to browser-specific CSS interpretations. 2. Failure to load custom fonts, resulting in fallback to default system fonts. 3. Introduction of new UI elements that deviate from the established style guide without prior approval.
Post-conditions	1. The platform presents a cohesive and professional visual identity to all users. 2. User recognition of the Waste-To-Treasure brand is enhanced through consistent visual elements.

	3. The user interface is aesthetically pleasing and contributes to a positive user experience.

3.3.9 Non-functional requirement 9

Table 35. Non-functional requirement 21

Field	Description
ID	NFR-9
Actor	System / Administrator
Description	The system must ensure data durability and recovery through daily automatic backups of the main database (Amazon RDS).
Pre-conditions	- The Amazon RDS (PostgreSQL) instance is operational. - Automatic backups are configured in AWS.
Flow of events	1. The system (Amazon RDS) automatically initiates a database snapshot daily. 2. The backup is stored securely. 3. The system retains this backup for a 7-day period. 4. In the event of a data loss event (corruption, failure), an Administrator can initiate a restoration from the last available backup.
Exceptions	1. The AWS backup service fails. 2. The database restoration process fails or takes longer than expected.
Post-conditions	The platform data is securely backed up, and a restoration point is available within the 7-day retention window.

3.3.10 Non-functional requirement 10

Table 36. Non-functional requirement 10

Field	Description
ID	NFR-10
Actor	System / Developer

Description	The system frontend (main portal and admin console) must be developed using the Next.js framework and hosted on the AWS Amplify platform.
Pre-conditions	- The AWS Amplify project is configured and linked to the code repository.
Flow of events	1. Developers build the User Interface (UI) components using React. 2. The code is pushed to the repository. 3. AWS Amplify automatically builds, tests, and deploys the Next.js web application. 4. Amplify manages the hosting and distribution of static assets (CSS, JS) via AWS CloudFront.
Exceptions	1. Failure in the build process in the Amplify pipeline. 2. A React dependency causes conflicts or becomes obsolete.
Post-conditions	The frontend is deployed and served to users.

3.3.11 Non-functional requirement 11

Table 37. Non-functional requirement 11

Field	Description
ID	NFR-11
Actor	System / Developer
Description	The backend business logic must be developed as an API using the FastAPI (Python) framework and hosted on an Amazon EC2 instance.
Pre-conditions	- The Amazon EC2 instance is provisioned and secured. - The Python and FastAPI environment is installed on the server.
Flow of events	1. Developers create the API endpoints (business logic) using FastAPI. 2. The FastAPI application runs on the Amazon EC2 server. 3. Amazon API Gateway acts as the entry point, securely routing HTTPS requests from the frontend (Next.js) to the FastAPI application on EC2. 4. The API processes the request, interacts with the database (RDS), and returns the response (JSON)
Exceptions	1. The EC2 instance fails or stops.

	2. The FastAPI application crashes due to an unhandled error. 3. Communication failure between API Gateway and the EC2 instance.
Post-conditions	The backend business logic is served securely.

3.3.12 Non-functional requirement 12

Table 38. Non-functional requirement 12

Field	Description
ID	NFR-12
Actor	System
Description	The system will use storage technologies such as RDS PostgreSQL for the relational databases and S3 for file and object storage.
Pre-conditions	Cloud infrastructure is configured, and RDS PostgreSQL and S3 services are provisioned.
Main scenario	1. The system stores structured data in the RDS PostgreSQL database. 2. The system stores files and objects (e.g. product images) in S3.
Exceptions	Connection failure with RDS PostgreSQL or S3 error in data writing or reading.
Post-conditions	Data is persistently and securely stored in the designated storage technologies.

3.3.13 Non-functional requirement 13

Table 39. Non-functional requirement 13

Field	Description
ID	NFR-13
Actor	System and Administrator

Field	Description
ID	NFR-13
Description	The system will comply with the Federal Law on Protection of Personal Data Held by Private Parties(LFPDPPP) to ensure the privacy and protection of user data.
Pre-conditions	Privacy policies and data handling procedures defined in accordance with LFPDPPP.
Main scenario	1. The system collects personal data from users. 2. The system applies technical and organizational security measures to protect the data. 3. The system allows users to exercise their ARCO rights (Access, Reactification, Cancellation, Opposition).
Exceptions	Security breaches. Non-compliance with ARCO rights requests.
Post-conditions	User personal data is protected, and the system operates in compliance with LFPDPPP.

3.3.14 Non-functional requirement 14

Table 40. Non-functional requirement 14

Field	Description
ID	RNF-14
Actor	System and Administrator
Description	The system will ensure compliance with relevant environment regulations for waste and recycled material management.
Pre-conditions	Applicable environmental regulational identified and documented.
Main scenario	1. The system records and classifies materials according to their type and recycling potential. 2. The system generates reports that facilitate traceability and compliance with regulations. 3. The system informs users about applicable regulations.

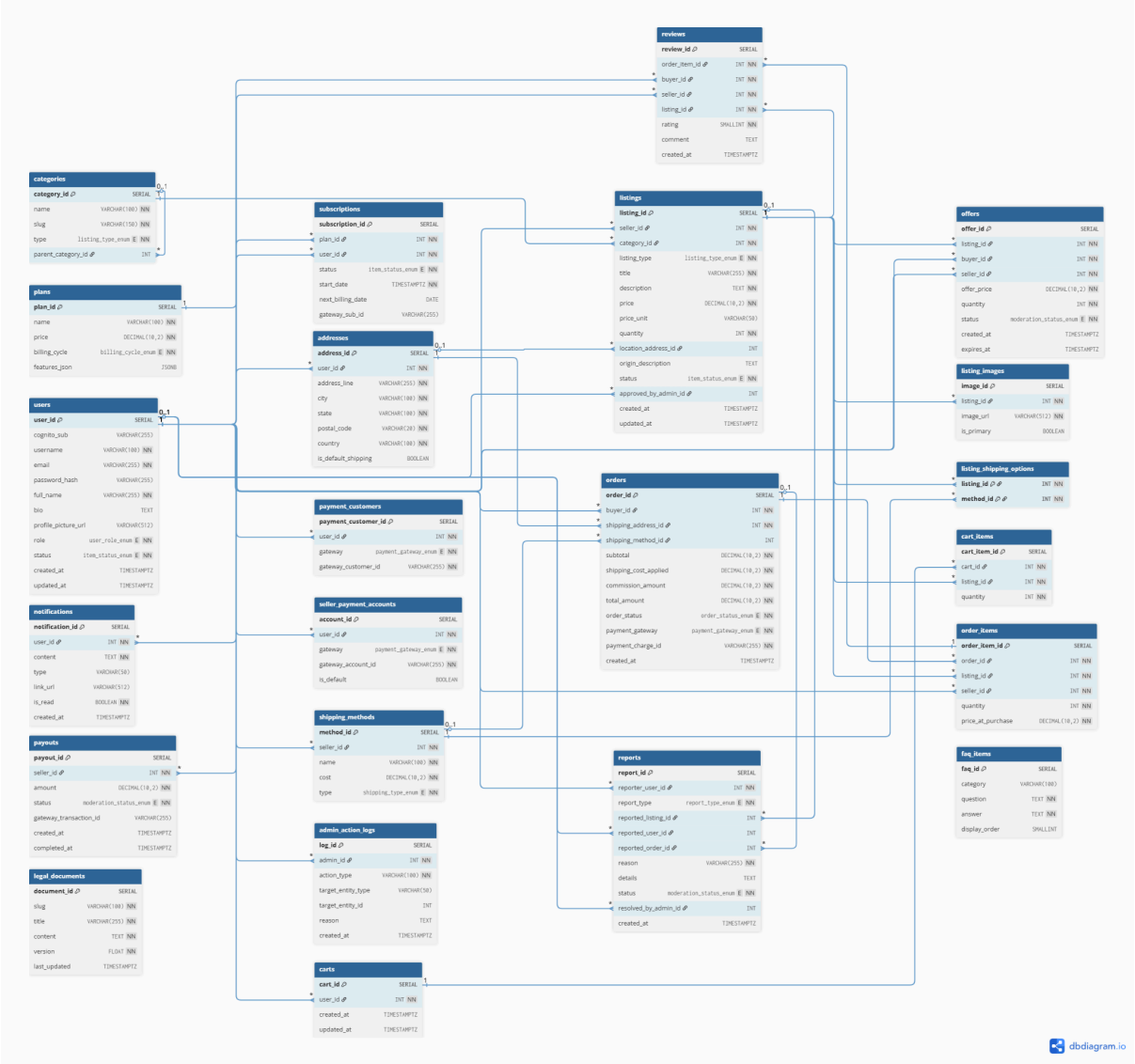
Field	Description
ID	RNF-14
Exceptions	Error in material classification. Outdated information or regulations.
Post-conditions	The system supports compliance with environment regulations and promotes sustainable practices.

3.3.15 Non-functional requirement 15

Table 41. Non-functional requirement 15

Field	Description
ID	RFN-15
Actor	System, Administrator and Seller
Description	The system will integrate with the Servicio de Administración Tributaria(SAT) for compliance with tax obligations related to transactions.
Pre-conditions	SAT API or defined available tax integration mechanisms.
Main scenario	1. The system records all sales transactions. 2. The system generates electronic invoices(CFDI) according to SAT requirements. 3. The system allows consultation and submission of required tax information to the SAT.
Exceptions	Failure to generate CDFI. Errors in communication with SAT.
Post-conditions	Tax obligations are met automatically and accurately through integration with SAT.

3.4 Logical database requirements



entity-relatioship-diagram.png

This database is designed for the Waste-To-Treasure platform using PostgreSQL. It follows a normalized relation model to manage the two marketplaces, the SaaS subscription model, payment transactions (Stripe), and moderation and support functions.

3.4.1 Table users

It stores information for all platform participants, whether they are buyers, sellers (companies, artisans), or system administrators. It manages authentication credentials and basic profile information.

Column	Data type	Restrictions	Description

user_id	SERIAL	PRIMARY KEY	Unique numeric identifier for each user
cognito_sub	VARCHAR (255)	UNIQUE, NULL	Unique user ID from AWS Cognito. Used to link the account if the user registers/logs in with an external provider (e.g., Google).
email	VARCHAR (255)	UNIQUE, NOT NULL	Unique email address. It's the primary means of communication and account recovery.
password_hash	VARCHAR (255)	NOT NULL	Hashed password for local login.
full_name	VARCHAR (255)	NOT NULL	The real name, company name, or workshop name of the user.
bio	TEXT	NULL	Optional public profile description that the user can write about themselves or their business.
profile_picture_url	VARCHAR (255)	NULL	URL link to the user's profile picture file, stored in Amazon SR.
role	user_role_enum	NOT NULL, DEFAULT 'USER'	Defines the user's permission level in the system ('USER' or 'ADMIN').
status	item_status_enum	NOT NULL, DEFAULT 'PENDING'	Moderates the user's status (e.g., 'PENDING' for new users, 'ACTIVE' after confirming email, 'BLOCKED' by an admin).
created_at	TIMESTAMP TZ	DEFAULT NOW()	Timestamp of when the account was created.
updated_at	TIMESTAMP TZ	DEFAULT NOW()	Timestamp that updates automatically each time the user modifies their profile.

3.4.2 Table legal_documents

Serves as a catalog for the different versions of the platform's legal documents, such as "Terms and Conditions" or "Privacy Policies". It allows the administrator to update the texts and maintain a version history.

Column	**Data type**	**Restrictions**	**Description**
document_id	SERIAL	PRIMARY KEY	Unique numeric identifier for the document
slug	VARCHAR (100)	UNIQUE, NOT NULL	Unique text identifier for the URL (e.g., 'terms-of-service')
title	VARCHAR	NOT NULL	The official title of the document (e.g.,

	(255)		“Terms and Conditions of Use")
content	TEXT	NOT NULL	Field that stores the full content of the legal document
version	FLOAT	NOT NULL, DEFAULT 1.0	Version number (e.g., 1.0, 1.1) to track changes
last_updated	TIMESTAMP TZ	DEFAULT NOW()	Timestamp of when this document version was updated.

3.4.3 Table plans

This table is a catalog that defines the different SaaS subscription levels (e.g., Free, Pro, Business) that the platform offers to sellers. It stores the price, billing cycle, and features of each plan.

Column	**Data type**	**Restrictions**	**Description**
plan_id	SERIAL	PRIMARY KEY	Unique numeric identifier for the plan.
name	VARCHAR (100)	UNIQUE, NOT NULL	Commercial name of the plan (e.g., ‘Pro Plan’, ‘Business Plan’).
price	DECIMAL (10, 2)	NOT NULL, DEFAULT 0.0	Monetary cost of the plan per billing cycle
billing_cycle	billing_cycle _enum	NOT NULL, DEFAULT ‘MONTHLY’	The frequency with which the plan is charged (‘MONTHLY’, ‘ANNUAL’)
feature_json	JSONB	NULL	A JSONB field to store a list of features (e.g., [‘increased_visibility’]).

3.4.4 Table subscriptions

It records which plan the user has active, their payment status, and the next billing date.

Column	**Data type**	**Restrictions**	**Description**
subscription_id	SERIAL	PRIMARY KEY	Unique identifier for the subscription
user_id	INTEGER	FK to users (*user_id*), UNIQUE	The user (seller) who purchased the plan. It’s UNIQUE to ensure only one active subscription per user.
plan_id	INTEGER	FK to plans (plan_id)	The specific plan the user has subscribed to.

status	item_status_enum	NOT NULL, DEFAULT 'ACTIVE'	Current status of the subscription (e.g., 'ACTIVE', 'INACTIVE').
start_date	TIMESTAMPTZ	NOT NULL	Date and time the subscription was activated
next_billing_date	DATE	NULL	The day the next recurring payment will be attempted
gateway_sub_id	VARCHAR (255)	NULL	Reference ID of the 'Subscription' object in Stripe to manage recurring payments.

3.4.5 Table payment_customers

It stores the reference (Customer ID) of a user (buyer) in the Stripe systems. This allows the platform to securely save payment methods (e.g., "Remember this card") for future purchases.

Column	**Data type**	**Restrictions**	**Description**
payment_customer_id	SERIAL	PRIMARY KEY	Unique identifier for the customer record
user_id	INTEGER	FK to users (*user_id*)	The user on our platform
gateway	payment_gateway_enum	NOT NULL	The payment gateway ('STRIPE') this customer ID belongs to.
gateway_customer_id	VARCHAR (255)	NOT NULL, UNIQUE	The unique alphanumeric ID generated by the gateway (e.g., cus_… in Stripe) that identifies our user in their system.

3.4.6 Table seller_payment_accounts

It stores the payout information for sellers. It securely saves their Stripe Connect account ID email so the platform can transfer them the money from their sales (minus, commissions).

Column	**Data type**	**Restrictions**	**Description**
account_id	SERIAL	PRIMARY KEY	Unique identifier for the payment account
user_id	INTEGER	FK to users (*user_id*), UNIQUE	The user (seller) who will be paid
gateway	payment_gateway_enum	NOT NULL	The gateway the seller has configured to receive money ('STRIPE')

gateway_account_id	VARCHAR (255)	NOT NULL	The account identifier in the gateway (e.g., the Stripe account ID acct_…).
is_default	BOOLEAN	DEFAULT true	Boolean to mark the primary withdrawal account (in case multiple are allowed)

3.4.7 Table categories

Stores the hierarchy of product and material categories. This table is managed by the Administrator and is used to organize and filter listings in both marketplaces.

Column	**Data type**	**Restrictions**	**Description**
category_id	SERIAL	PRIMARY KEY	Unique numeric identifier for the category
name	VARCHAR (100)	NOT NULL	Visible name of the category (e.g., 'Wood', 'Textiles').
slug	VARCHAR (150)	UNIQUE, NOT NULL	URL version of the name (e.g., 'recycled-wood').
type	listing_type_enum	NOT NULL	Indicates which marketplace this category belongs to ('MATERIAL' or 'PRODUCT')
parent_category_id	INTERGER	FK to categories (*category_id*)	Recursive foreign key (points to this same table) that allows for nested categories (e.g., 'Textiles' > 'Cotton').

3.4.8 Table listings

This table stores every post, whether it's a 'MATERIAL' or a 'PRODUCT'. It contains all descriptive information, price, stock, and moderation status.

Column	**Data type**	**Restrictions**	**Description**
listings_id	SERIAL	PRIMARY KEY	Unique numeric identifier for the post
seller_id	INTEGER	FK to users (*user_id*)	The user (seller) who created and owns this post
category_id	INTEGER	FK to categories (*category_id*)	Link to the category it belongs to.
listing_type	listing_type_enum	NOT NULL	Key field that defines if it's 'MATERIAL' or 'PRODUCT' post
title	listing_type_e	NOT NULL	The main visible name of the post

	num		
description	VARCHAR (255)	NOT NULL	Detailed text description
price	TEXT	NOT NULL	Monetary price of the item
price_unit	DECIMAL (10, 2)	NULL	Optional. Clarifies the price (e.g., ‘Kg’, ‘Lot’, ‘Unit’).
quantity	VARCHAR (50)	NOT NULL. DEFAULT 1	The number of items available in inventory (stock)
location_address_id	INTEGER	FK to addresses (*address_id*)	Links to an address to indicate where the item is physically located.
origin_description	TEXT	NULL	Text field where the seller certifies the recycled or reused origin
status	item_status_enum	NOT NULL, DEFAULT ‘PENDING’	Moderation status ‘PENDING’ is the default on creation ‘ACTIVE’ is visible to the public, ‘REJECTED’ was denied
approved_by_admin_id	INTEGER	FK to users (*user_id*)	Reference to the administrator who approved the change from ‘PENDING’ to ‘ACTIVE’
created_at	TIMESTAMPTZ	DEFAULT NOW()	Timestamp of when the post was created
updated_at	TIMESTAMPTZ	DEFAULT NOW()	Timestamp of the last edit

3.4.9 Table listing_images

Pivot table that associates one or more images with a listing (post). It only stores the image URL; the actual file lives in Amazon S3.

Column	**Data type**	**Restrictions**	**Description**
image_id	SERIAL	PRIMARY KEY	Unique identifier for the image
listing_id	INTEGER	FK to listings (*listing_id*)	The listing this image belongs to
image_url	VARCHAR (512)	NOT NULL	The full, public URL of the object in the S3 bucket
is_primary	BOOLEAN	DEFAULT false	Boolean (True/False) to mark which photo is the cover image.

3.4.10 Table offers

Enable the negotiation functionality of the marketplace. A buyer can send an 'offer' (proposing a price and quantity) for a material listing, and the seller can accept, reject, or counter it.

Column	Data type	Restrictions	Description
offer_id	SERIAL	PRIMARY KEY	Unique identifier for the offer
listing_id	INTEGER	FK to listings *(listing_id*)	The listing (type 'MATERIAL') that the offer is for
buyer_id	INTEGER	FK to users (*user_id*)	The user (buyer) sending the offer
seller_id	INTEGER	FK to users (*user_id*)	The user (seller) receiving the offer
offer_price	DECIMAL (10, 2)	NOT NULL	The unit price the buyer is willing to pay
quantity	INTEGER	NOT NULL	The number of items the buyer wants at that price
status	modration_status_enum	NOT NULL, DEFAULT 'PENDING'	Status of the negotiation cycle ('PENDING', 'ACCEPTED', 'REJECTED', 'COUNTERRED')
created_at	TIMESTAMPTZ	DEFAULT NOW()	Timestamp of when the offer was sent
expires_at	TIMESTAMPTZ	NULL	Optional timestamp for the offer to automatically expire if the seller doesn't respond

3.4.11 Table shipping_methods

This table allows sellers to define their own shipping option and costs.

Column	Data type	Restrictions	Description
method_id	SERIAL	PRIMARY KEY	Unique identifier for the shipping method
seller_id	INTEGER	FK to users (*user_id)*	The user (seller) who offers this method
name	VARCHAR (100)	NOT NULL	Descriptive name (e.g., 'Workshop Pickup').
cost	DECIMAL (10, 2)	NOT NULL, DEFAULT 0.0	The price that will be added to the order if this method is chosen

type	shipping_type _enum	NOT NULL, DEFAULT 'DELIVERY'	Defines if it's a 'PICKUP' or a home 'DELIVERY'

3.4.12 Table listing_shipping_options

Pivot table that connects listings (posts) with shipping_method. It allows the seller to specify which shipping methods (that they previously created) apply to each of their products.

Column	**Data type**	**Restrictions**	**Description**
listing_id	INTEGER	FK to listings (*listing_id*)	The listing to which this option applies
method_id	INTEGER	FK to shipping_methods (*method_id*)	The shipping method available for that listing
CONSTRAINT	PRIMARY KEY	(*listing_id, method_id*)	Composite primary key. Prevents duplicates and defines the relationship

3.4.13 Table cart and cart item

Manage the user's temporary cart before purchase. cart identifies the owner, and cart_item holds the product.

Column	Data type	Restrictions	Description
cart_id	SERIAL	PRIMARY KEY	Unique ID for the cart
user_id	INTEGER	FK to users (*user_id*), UNIQUE	Owner of the cart. UNIQUE FK to ensure one cart per user.

Column	**Data type**	**Restrictions**	**Description**
cart_item_id	SERIAL	PRIMARY KEY	Unique ID for the cart item
cart_id	INTEGER	FK to carts (*cart_id*)	The cart this item belongs to
listing_id	INTEGER	FK to listing (*listing_i*d)	The product/material being purchased
quantity	INTEGER	NOT NULL	The quantity of units

3.4.14 Table orders

This table records a completed sales transaction (Checkout). It stores who bought, where it's being shipped, how much was paid, the commission retained, and the transaction ID from the payment gateway.

Column	Data type	Restrictions	Description
order_id	SERIAL	PRIMARY KEY	Unique numeric identifier for the order
buyer_id	INTEGER	FK to users (*user_id*)	The user (buyer) who paid for the order
shipping_address_id	INTEGER	FK to addresses (*address_id)*	The address the buyer selected for delivery
shipping_method_id	DECIMAL (10, 2)	FK to shippipng_methods (*method_id*)	The shipping_method the buyer chose (which defines the shipping cost)
subtotal	DECIMAL (10, 2)	NOT NULL	The total cost of the products (order_items) before adding shipping
shipping_costo_applied	DECIMAL (10, 2)	NOT NULL, DEFAULT 0.00	The shipping amount, based on the shipping_method_id
commission_amount	DECIMAL (10, 2)	NOT NULL	The commission amount (e.g., 10%) that the platform retained (calculated from the subtotal)
total_amount	DECIMAL (10, 2)	NOT NULL	The final amount charged to the buyer (Subtotal + Shipping)
order_status	oder_status_enum	NOT NULL, DEFAULT 'PAID'	Logistic status of the order ('PAID', 'SHIPPED', 'DELIVERED', 'CANCELED')
payment_gateway	payment_gateway_enum	NOT NULL	Indicates which gateway processed the payment ('STRIPE')
payment_charge_id	VARCHAR (255)	NOT NULL, UNIQUE	The charge reference ID (e.g., pi_… from Stripe). This is the proof of payment.

3.4.15 Table oder_items

Stores a "snapshot" of the specific products/materials that were part of an order.

Column	Data type	Restrictions	Description
order_item_id	SERIAL	PRIMARY KEY	Unique identifier for the order item
order_id	INTEGER	FK to orders	The parent order this item belongs to

		(*order_id*)	
listing_id	INTEGER	FK to listings (listing_id)	The listing that was purchased. Uses ON DELETE RESTRICT to prevent deleting history if the listing is removed
seller_id	INTEGER	FK to users (*user_id*)	Denormalized (copied) to easily query 'who sold what' in an order, and for reviews
quantity	INTEGER	NOT NULL	Quantity of units of this item that were purchased
price_at_purchase	DECIMAL (10, 2)	NOT NULL	Stores the price of the listings at the time of purchase, so the record is correct even if the seller changes the listing price later

3.4.16 Table reviews

Stores the ratings and comments (reviews) that a buyer leaves about an item after a purchase. It is linked directly to order_item to ensure reviews can only be left after a verified purchase.

Column	**Data type**	**Restrictions**	**Description**
review_id	SERIAL	PRIMARY KEY	Unique identifier for the review
order_item_id	INTEGER	FK to order_items (*order_item_id*), UNIQUE	Link to verify the purchase. UNIQUE ensures only one review per item purchased.
buyer_id	INTEGER	FK to users (*user_id*)	The author of the review
seller_id	INTEGER	FK to user (user_id)	The seller being reviewed
listing_id	INTGER	FK to listings (*listing_id*)	The product/material being reviewed
rating	SMALLINT	NOT NULL, CHECK (1-5)	Numeric rating (1 to 5)
comment	TEXT	NULL	Optional text comment accompanying the rating
created_at	TIMESTAMP TZ	DEFAULT NOW ()	Timestamp of when the review was published

3.4.17 Table notifications

Store the “bell” alerts within the application (implied in Views). When an event occurs (e.g., “new offer”, “sale completed”, “report resolved”), a record is created here for the user to see upon login.

Column	Data type	Restrictions	Description
notification_id	SERIAL	PRIMARY KEY	Unique identifier for the notification
user_id	INTEGER	FK to users (*user_id*)	The user who receives the notification
content	TEXT	NOT NULL	The text of the notification (e.g., ‘You have received a new offer for your ’Pine Wood”)
type	VARCHAR (50)	NULL	Category (e.g., ‘ORDER’, ‘REPORT_RESOLVED’, ‘OFFER’ to group or display icons)
link_url	VARCHAR (512)	NULL	The relative link (e.g., ‘/my-offers/123’) the user should go to when clicking
is_read	BOOLEAN	NOT NULL, DEFAULT false	Boolean indicating if the user has already seen this notification
created_at	TIMESTAMP TZ	DEFAULT NOW()	Timestamp of when the notification was generated

3.4.18 Table reports

Unified table that handles both content reports and claims on orders. It allows a user to report a listing, another user, or a problem with an order (“dispute”).

Column	Data type	Restrictions	Description
report_id	SERIAL	PRIMARY KEY	Unique identifier for the report/claim
reporter_user_id	INTEGER	FK to users (*user_id*)	The user initiating the report/claim
report_type	report_type_enum	NOT NULL	Defines what is being reported (‘LISTING’, ‘USER’, or ‘ORDER_ISSUE’)
reported_listing_id	INTEGER	FK to listings (*listing_id*), NULL	(Optional) The listing being reported. Used if report_type is ‘LISTING’
reported_user_id	INTEGER	FK to users (user_id), NULL	(Option) The user being reported. Used if report_type
reported_order_id		FK to order	(Optional) The order being

		(*order_id*), NULL	claimed.Used if reported_type is 'ORDER_ISSUE'
reason	VARCHAR (255)	NOT NULL	Reason selected from a list (e.g., 'Fraud', 'Not received', 'Inappropriate content')
details	TEXT	NULL	Optional text field for the user to explain the problem in detail
status	moderation_status_enum	NOT NULL, DEFAULT 'PENDING'	Moderation status of the report ('PENDING', 'SOLVED', 'DISMISSED')
resolved_by_admin	INTEGER	FK to users (*user_id*)	Reference to the administrator who made a decision on this report

3.4.19 Table admin_action_logs

Records sensitive actions by administrators for internal auditing, such as rejecting a listing, resolving a report, or blocking a user.

Column	**Data type**	**Restrictions**	**Description**
log_id	SERIAL	PRIMARY KEY	Unique identifier for the log entry
admin_id	INTEGER	FK to users (*user_id*)	The administrator (user with 'ADMIN' role) who performed the action
action_type	VARCHAR (100)	NOT NULL	A text code defining the action taken (e.g., 'LISTING_REJECTED', 'REPORT_RESOLVED', 'USER_BLOCKED')
target_enity_type	VARCHAR (50)	NULL	(Optional) The type of object that was affected (e.g., 'LISTING', 'USER', 'REPORTED')
target_entity_id	INTEGER	NULL	(Optional) The ID of the affected object (e.g., the listing_id that was rejected).
reason	TEXT	NULL	(Crucial) The "why" a text field where the admin explains why they rejected something (e.g., "Poor quality photos").
created_at	TIMESTAMPTZ	DEFAULT NOW()	Timestamp of when the action occurred

3.4.20 Table faq_items

Stores the content (questions and answers) displayed on the Help or FAQ page. It is managed by Administrators to resolve common user doubts.

Column	Data type	Restrictions	Description
faq_id	SERIAL	PRIMARY KEY	Unique identifier for the question
category	VARCHAR (100)	DEFAULT 'General'	Thematic grouper (e.g., 'Payments'. 'Shipping', 'General') to organize question
question	TEXT	NOT NULL	The full text of the frequently asked question
answer	TEXT	NOT NULL	The full text of the provided answer

3.4.21 Table payouts

A financial audit table that records money transfers from the platform to the sellers. It stores the requested amount, the status of the transfer, and the gateway's transaction ID.

Column	Data type	Restrictions	Description
payout_id	SERIAL	PRIMARY KEY	Unique identifier for the payout transaction.
seller_id	INTEGER	FK to users (*user_id*)	The user (seller) receives the money.
amount	DECIMAL (10, 2)	NOT NULL	The net amount transferred (Total Sales - Commissions).
status	moderation_status_enum	NOT NULL, DEFAULT 'PENDING'	Status of the transfer ('PENDING', 'RESOLVED', 'COMPLETED', 'REJECTED')
gateway_transaction_id	VARCHAR (255)	NULL	The withdrawal transaction ID (e.g. tr_… in Stripe) as proof.
created_at	TIMESTAMP TZ	DEFAULT NOW()	Timestamp of when the withdrawal was requested.
completed_at	TIMESTAMP TZ	NULL	Timestamp of when the money was confirmed as sent.

3.5 Design Constraints

This section details the non-functional restrictions and pre-determined design decisions that limit the options for the development team. These constraints are mandatory for the project.

3.5.1 Technology Stack Constraints

The development of the system is strictly limited to the following approved technology stack:

- **Front-End:** The client-side application must be a Next.js Web Application.
- **Back-End:** All server-side business logic, APIs, and orchestration be implemented using Python, specifically with the FastAPI framework.
- **Database:** The primary relational database for the project must be PostgreSQL.

3.5.2 Infrastructure Constraints

The system must be designed as a cloud-native application, restricted to the following provides and services:

- **Cloud Provider:** The entire infrastructure must be deployed on Amazon Web Services (AWS).
- **Mandatory AWS Services:** The system architecture must utilize, at a minimum, the following AWS Service:
    - **AWS Amplify:** For hosting, CI/CD pipeline, and deployment of the Next.js front-end.
    - **Amazon EC2:** For hosting the FastAPI back-end application.
    - **Amazon RDS:** For the managed PostgreSQL database.
    - **Amazon S3:** For all static asset storage (e.g., user-uploaded images, documents).
    - **Amazon Cognito:** For all user identity management, authentication, and authorization.
    - **Amazon API Gateway:** To act as the secure, single entry point for all API requests to the back-end.
    - **AWS Certificate Manager (ACM):** For managing and deploying SSL/TLS certificates.
    - **AWS CloudFront:** To serve as the Content Delivery Network (CDN) for the front-end.

5.5.3 Integration and Deployment Constraints

- **Payment Gateways:** The system must exclusively integrate with the Stripe API for all payment and subscription processing. No other payment gateways are to be considered.

- **Mandatory Deployment:** The final project must be fully deployed online on the AWS infrastructure and accessible via public URL for final review and grading.

3.5.4 Project Management and Tooling Constraints

Hard Deadline: The project must be fully implemented, tested, and deployed no later than **November 18, 2025**.

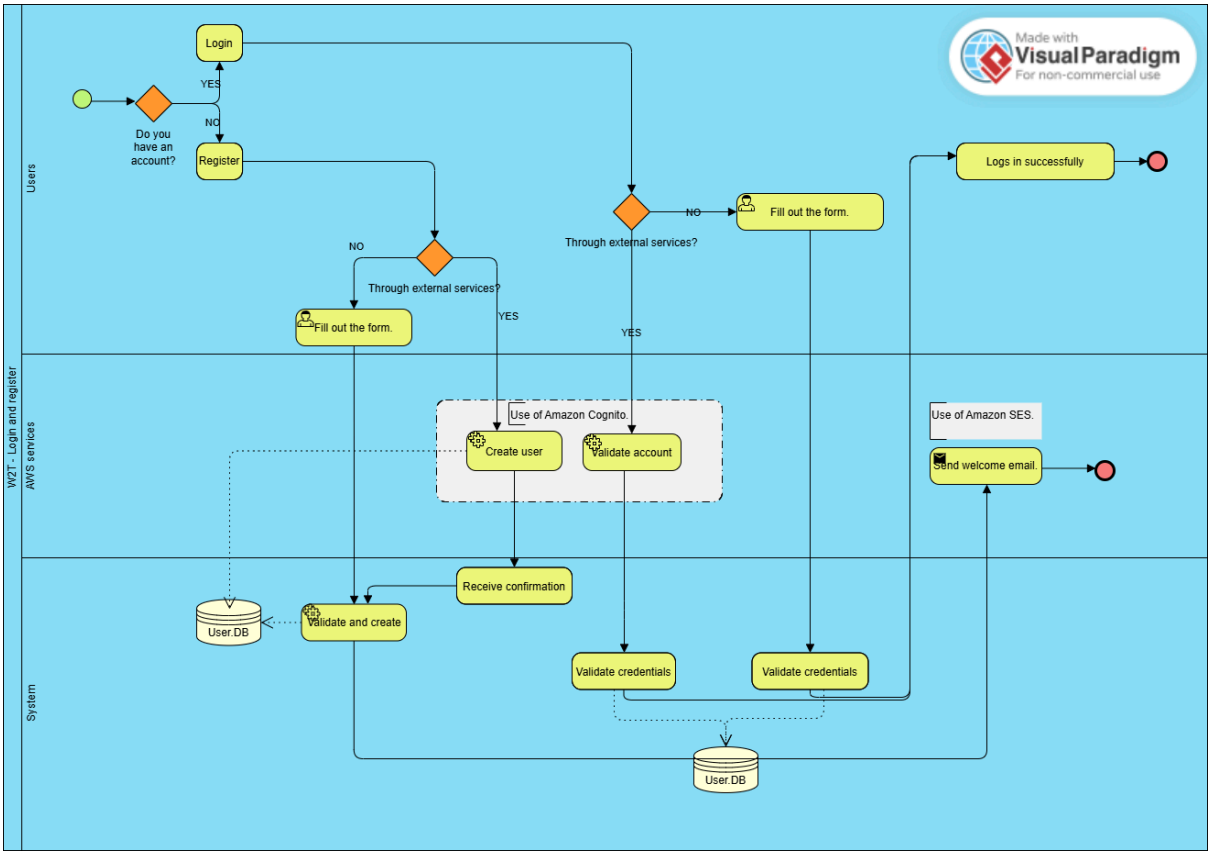
Mandatory Tooling: The team must use the course-defined toolset for all project management and documentation:

- **Version Control:** GitHub.
- **Documentation:** Google Docs / Google Slides.
- **Prototyping:** Figma.
- **Agile Management:** ClickUp.

3.6 BPMN Process Diagram

This section introduces the notation and methodology for modeling the operational processes of the W2T system, utilizing the BPMN. The design focuses on the clarity and standardization of the workflows

3.6.1 Login and register - Process Diagram

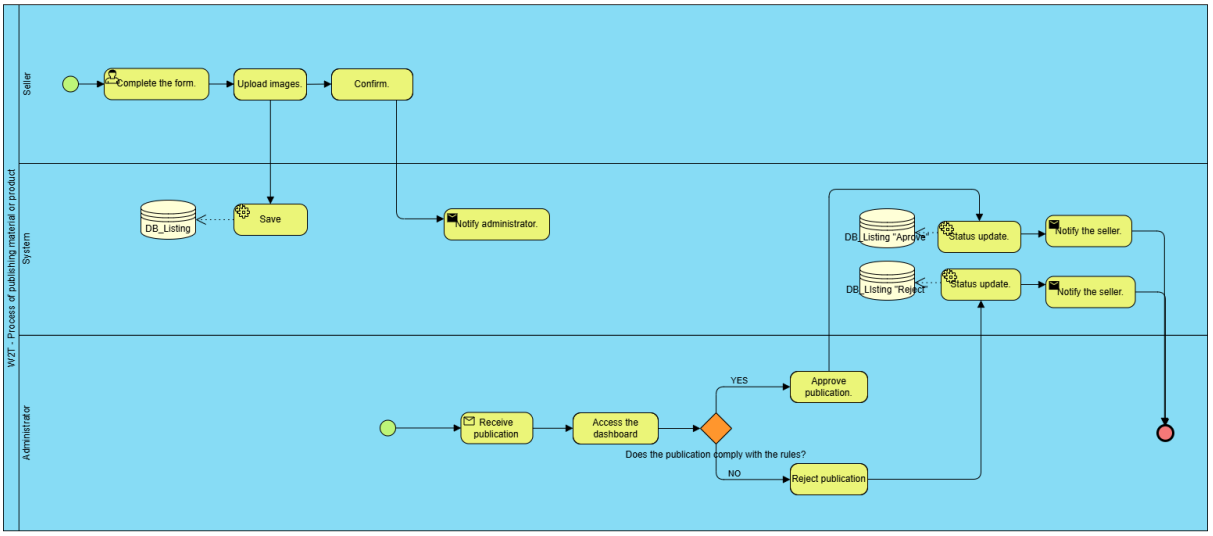


This process exemplifies how users (Administrator, Buyer and Seller), external services and the system operate when user wants to register through different methods such as Google, Facebook, etc., using the Amazon Cognito service, it also shows how users can log in using alternatively choose the traditional login method.

The flow begins with the user taking two possible paths depending on whether they already have an account or not. If the user does not have an account, they follow the registration path, where they can choose to register using one of the previously mentioned methods or opt for a traditional registration. The user registration is then stored in the database, and afterward, the user receives a welcome email.

The second path is for users who already have an account. Here the user can also choose one of the previously mentioned login methods or the traditional login option. The credentials are validated in the databases, and finally, the user gains access to the platform.

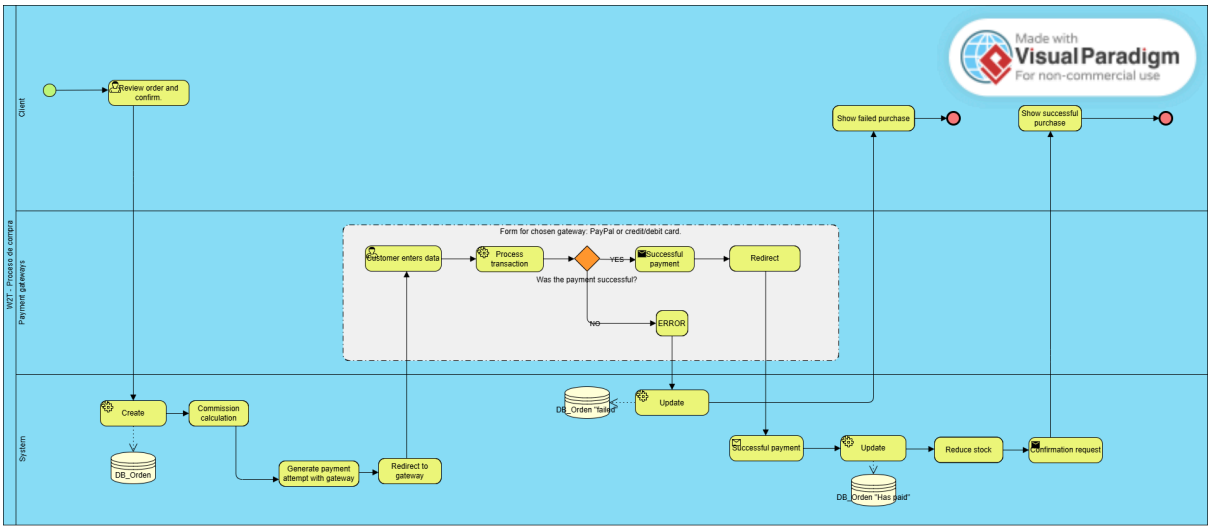
3.6.2 Publishing material or product - Process Diagram



This process shows how product or material sellers (who have already logged in), the system, and administrators interact when a seller attempts to publish a product or material on the platform. The process flow begins when the user decides to publish something, which requires filling out the publication form with details such as name, price, publication. A notification is then sent to the administrator to approve or reject the publication.

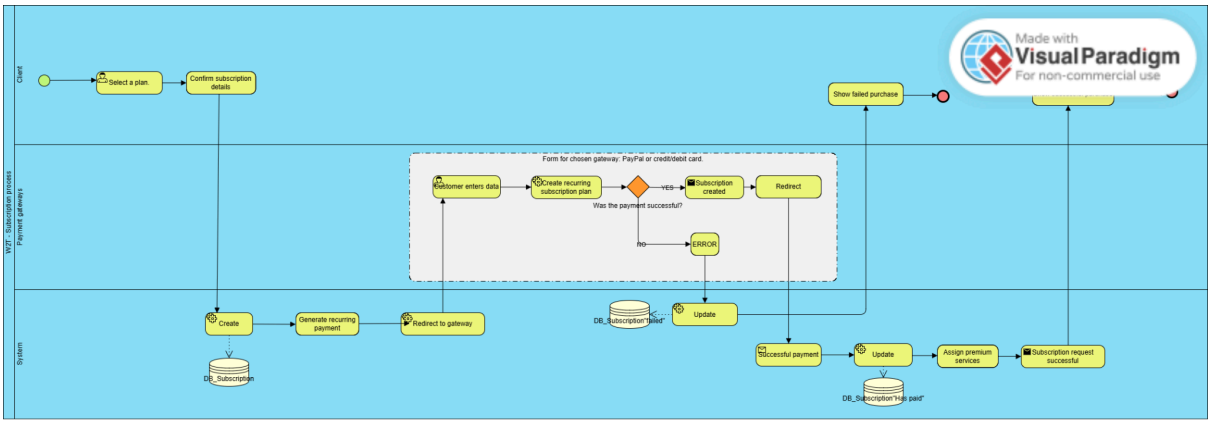
The administrator receives the notification and accesses the dashboard. The administrator follows the path or either approves or rejects the publication. The status is saved in the database, and finally, the seller is notified of the publication status.

3.6.3 Purchase - Process Diagram



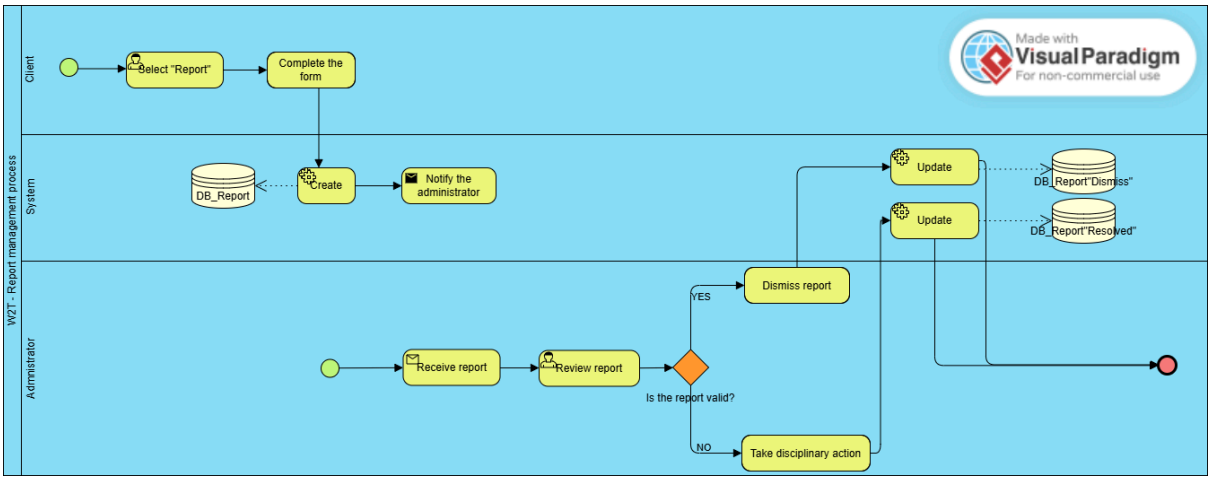
The flow begins when the customer chooses to review their cart, confirms the order and details, and the system proceeds to process the purchase order and update the stock status in the database as the process continues. Meanwhile, the payment gateway displays a form for the customer to fill out according to the selected payment method. Depending on whether the process is completed successfully or an error occurs, the user is shown the outcome of the process, and the corresponding subsequent tasks are carried out accordingly.

3.6.4 Subscription - Process Diagram



The flow shows the process that takes place through the sellers, the system, and the payment gateways when a customer decides to upgrade to a premium plan, as well as the payment process offered by the platform. The flow begins when the customer selects a plan and confirms the details. The system then proceeds with updating the subscription, while the payment gateway displays a form for the customer to fill out according to the selected method. Depending on whether the process is completed successfully or an error occurs, the user is shown the outcome of the process.

3.6.5 Report management - Process Diagram

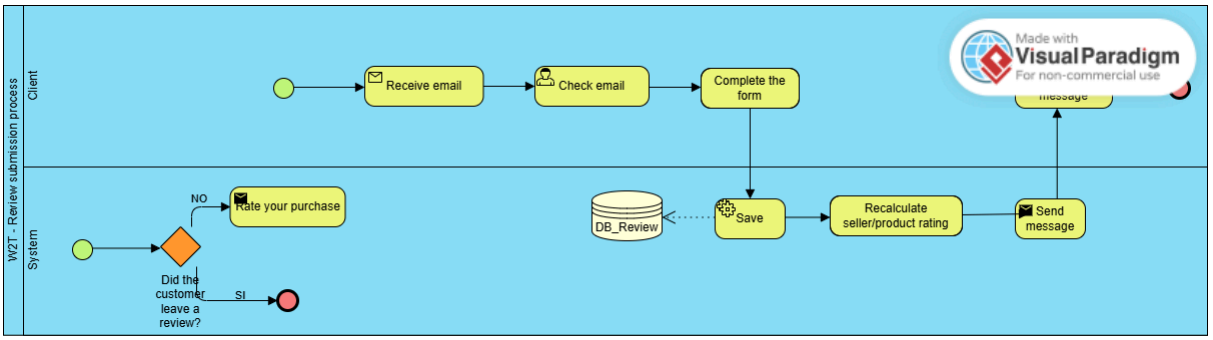


This process exemplifies how customers, the system, and the administrator interact in the creation and management of a report from the user's submission of the report, to its flow

within the system, and how the administrator decides whether to approve or dismiss it. The flow begins when the customer selects the “Reports” option within a seller’s publication, completes the form, and the report’s status is stored in the database while a notification is sent to the administrator.

The second flow continues when the administrator receives the report, reviews it, and decides whether to dismiss it or take disciplinary action. The final status is then updated in the database depending on the chosen course of action.

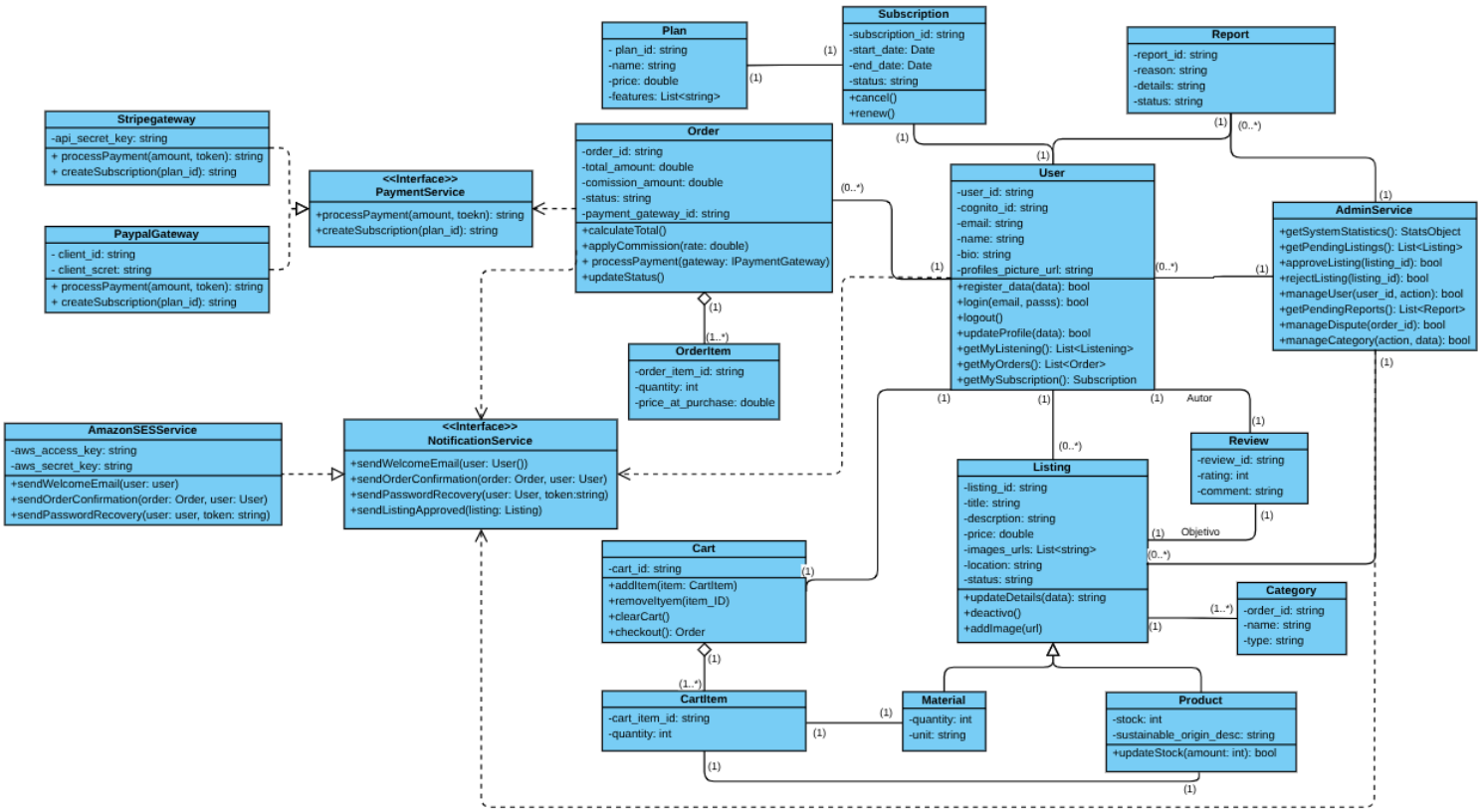
3.6.6 Review of Rating - Process Diagram



The flowchart describes the process of leaving a review by a customer who purchased a product or material within the platform. The first flow begins when the system checks whether the user has already left a review for the purchased item. If the decision is “YES” the process ends; otherwise, a message is sent to the customer.

The second flow begins when the customer receives the email, reviews the review form, and submits it. The information is then stored in the database, the seller’s rating is recalculated, a “review registered” message is generated, and finally, the customer receives that confirmation message.

3.7 Class Diagram



This section defines the software architecture for the W2T system using UML Class Model. The design adheres to the principle of Encapsulation (private attributes, public methods) and utilizes Interfaces for external services, ensuring a modular, secure, and maintainable system.

3.6.1 Entity and model Classes

These classes represent the core data entities of the system.

Class	Type	Attributes (Private -)	Methods (Public +)
User	Concrete	- user_id: string - cognito_id: string - email: string - name: string - bio: string - profile_picture_url: string - roles: List<string>	+ register(data): bool + login(email, pass): bool + logout() + updateProfile(data): bool + getMyListings(): List<Listing> + getMyOrders(): List<Order> + getMySubscription(): Subscription

Address	Concrete	- street: string - city: string - state: string - postal_code: string - country: string - notes: string	+ getFullAddress(): string + validate(): bool
Listing	Abstract	- listing_id: string - title: string - description: string - price: double - images_urls: List<string> - location: string - status: string	+ updateDetails(data): bool + deactivate() + addImage(url)
Material	Concrete (Inherits from Listing)	- quantity: int - unit: string	
Product	Concrete (Inherits from Listing)	- stock: int -sustainable_origin_desc: string	+ updateStock(amount: int): bool
Category	Concrete	- category_id: string - name: string - type: string	
Review	Concrete	- review_id: string - rating: int - comment: string	
Report	Concrete	- report_id: string - reason: string - details: string - status: string	

3.6.2 Transaction and Subscription Classes

Class	Type	Attributes (Private -)	Methods (Public +)
Cart	Concrete	- cart_id: string	+ addItem(item: CartItem) + removeItem(item_id) + clearCart() + checkout(): Order
CartItem	Concrete	- cart_item_id: string - quantity: int	
Order	Concrete	- order_id: string - total_amount: double - commission_amount: double - status: string - payment_gateway_id: string	+ calculateTotal() + applyCommission(rate: double) + processPayment(gateway: PaymentService) + updateStatus(newStatus)
OrderItem	Concrete	- order_item_id: string - quantity: int - price_at_purchase: double	
Plan	Concrete	- plan_id: string - name: string - price: double - features: List<string>	
Subscription	Concrete	- subscription_id: string - start_date: Date - end_date: Date - status: string	+ cancel() + renew()

3.6.3 Service and Interface Classes

Class	Type	Attributes (Private -)	Methods (Public +)
AdminService	Concrete	(None)	+getSystemStatistics(): StatsObject +getPendingListings(): List<Listing> +approveListing(listing_id): bool +rejectListing(listing_id): bool +manageUser(user_id, action): bool +getPendingReports(): List<Report> +manageCategory(action, data): bool
PaymentService	<<Interface>>	(None)	+processPayment(amount, token): string +createSubscription(plan_id): string
StripeGateway	Concrete (Implements PaymentServic)	-api_secret_key: string	+processPayment(amount, token): string +createSubscription(plan_id): string
NotificationService	<<Interface>>	(None)	+ sendWelcomeEmail(user: User) +sendOrderConfirmation(order: Order, user: User) +sendPasswordRecovery(user: User, token: string) +sendListingApproved(listing: Listing)
AmazonSESService	Concrete (Implements NotificationService)	-aws_access_key: string -aws_secret_key: string	(...same methods as interface)

3.6.4 Class Relationship Summary

Relationship	Origin (Multiplicity)	Destination (Multiplicity)	Description
Inheritance	Material	Listing	Material is a type of Listing.
Inheritance	Product	Listing	Product is a type of Listing.
Implementation	StripeGateway	PaymentService	StripeGateway fulfills the PaymentService contract.
Implementation	AmazonSESService	NotificationService	AmazonSESService fulfills the NotificationService contract.
Composition	Order (1)	OrderItem (1..*)	An Order is composed of one or more OrderItems.
Composition	Cart (1)	CartItem (0..*)	A Cart is composed of zero or more CartItems.
Association	User (1)	Cart (1)	A User owns exactly one Cart.
Association	User (1) C	Address (0..*)	A User can save multiple Addresses (Address Book).
Association	User (1)	Order (0..*)	A User places zero or more Orders.
Association	User (1)	Listing (0..*)	A User (Seller) publishes zero or more Listings.
Association	User (1)	Subscription (0..*)	A User can have a history of Subscriptions.
Association	User (1)	Review (0..*)	A User (Author) writes zero or more Reviews.
Association	User (1)	Report (0..*)	A User (Author) submits zero or more Reports.
Association	Listing (0..*)	Category (1)	A Listing must belong to exactly one Category.

Association	CartItem (1)	Listing (1)	A CartItem represents a single Listing.
Association	OrderItem (1)	Listing (1)	An OrderItem represents a single Listing.
Association	Listing (1)	Review (0..*)	A Listing can have zero or more Reviews.
Association	Listing (0..1)	Report (0..*)	A Listing can have zero or more Reports filed against it.
Association	User (0..1)	Report (0..*)	A User can have zero or more Reports filed against them.
Association	Subscription (1)	Plan (1)	A Subscription is for one specific Plan.
Association	Order (1)	Address (1)	An Order has one "shipping_address".
Association	Order (1)	Address (1)	An Order has one "billing_address".
Association	AdminService (1)	Listing (0..*)	The AdminService "controls" or "manages" Listings.
Association	AdminService (1)	User (0..*)	The AdminService "controls" or "manages" Users.
Association	AdminService (1)	Report (0..*)	The AdminService "controls" or "manages" Reports.
Association	AdminService (1)	Category (0..*)	The AdminService "controls" or "manages" Categories.
Dependency	Order	PaymentService	The Order class uses a PaymentService to process payment.

Dependency	Order	NotificationService	The Order class uses a NotificationService to send confirmation.
Dependency	User	NotificationService	The User class (in its register method) uses a NotificationService.
Dependency	AdminService	NotificationService	The AdminService (in approve/reject methods) uses a NotificationService.

4 Supporting information

4.1 Table of contents and index

4.2 Appendixes

4.2.1 Risk Matrix

This section establishes the framework for risk assessment and management associated with the W2T system, using the Risk Matrix as a central tool. The objective is to quantify the probability and impact of potential events.

		[Name risk]				
		Impact				
		Insignificant	Minor	Moderate	Major	Critical
Probability		1	2	3	4	5
76-100%	5	5	10	15	20	25
51-75%	4	4	8	12	16	20
31-50%	3	3	6	9	12	15

11-30%	2	2	4	6	8	10
1-10%	1	1	2	3	4	5

Exposure calculation: Probability X impact

Color Code

- **Green:** Represents low-impact and low-probability risks. Although they do not require urgent actions, it is important to monitor them regularly to prevent them from becoming threats.
- **Yellow:** Indicates moderate risks. These have a moderate probability of occurring and an impact that, although not critical, could affect operations if not managed properly.
- **Orange:** Refers to high-impact risks, but with a low probability of occurring although less frequent, if they materialize, the impact will be considerable, so they must be closely monitored.
- **Red:** These are critical risks. These have a high probability of occurring and, if they happen, can have a devastating impact on the company. They require immediate action.

Types of risks

- Technical Risks - RT.
- Financial Risks - RF.
- Operational Risks - RO.
- Business Risks - RN.
- Team Risks - RE.
- Legal and regulatory Risks - RLR.

Attachments:

Matriz de riesgo.xlsx

Risk Management

Risk table

Risk	Probability	Impact	Exposure	Mitigation	Responsible
Technical Risk					
RT-01 **Deficient User Experience UX/UI:** A poorly	3	4	12	Apply a user-centered design process: Conduct prototyping and usability tests	Alejandro

intuitive design that makes it difficult for companies to publish materials or for artisans to find them.				with real companies and artisans. Iterate the design based on direct feedback to ensure the platform is easy and efficient to use.	
RT-02 **Data loss or database corruption:** Partial or total loss of user, product or transaction information.	3	5	15	Implement a multi-level backup strategy: Active daily automatic backups in Amazon RDS with 7-day retention.	Alejandro - Gabriel
RT-03 **Performance scalability problems:** The platform becomes slow or fails if the number of users or data grows rapidly	4	3	12	Design a scalable architecture from the start: Use self-scaling AWS services (like EC2 Auto Scaling and RDS), optimize database queries, and perform load tests to identify and eliminate bottlenecks before launch.	Alejandro - Gabriel
RT-04 **Lack of technical documentation:** Absence of clear documentation of the code and architecture.	1	1	1	Implement a “documentation-as-code” culture from the project’s inception.	Team

Financial Risks					
RF-01 **Failure to reach the break-even point:** Operating costs consistently exceed income from commissions and subscriptions in a sustained manner.	5	4	20	Strictly control costs and diversify income: Optimize the use of AWS services to minimize expenses. Focus the initial strategy on validating the subscription model, which provides more predictable income than commissions.	Arturo
RF-02 **Failures in revenue model projections:** The conversion rate to premium plans or the transaction volume are much lower than expected.	5	4	20	Validate the pricing model with the real market: Offer free trials or a freemium model to engage users.	Arturo
RF-03 **High subscription cancellation rate:** Users who contract premium plans cancel after 1-2 months, due to unmet expectations or better	4	3	12	Establish engagement metrics (login/week), active publications, report usage, and create alerts for “at-risk” users (no activity for 10 days).	Team

alternatives.					
Operational Risks					
RO-01 **Fraud or disputes between users:** Conflicts over the quality, quantity, or payments of materials that the platform must mediate.	4	3	12	Implement a reputation and dispute resolution system: create a rating and comment system (like in Mercado Libre). Establish clear policies and mediation processes to handle conflicts fairly and transparently.	Arturo - Development team
RO-02 **Insufficient user support capacity:** The low budget ($500) prevents effective problem solving, generating frustration.	4	3	12	Optimize support with self-help tools: Create a robust Frequently Asked Questions (FAQ) section.	Arturo - Development team
RO-03 **Lack of material quality verification:** Impossibility of guarantee that materials meet standards.	3	4	12	Create a structure publication form obligating sellers to specify: type of material condition (new, used, with defects), exact quantity, multiple photographs, material origin and availability	Arturo - Development team

				date.	
RO-04 **Low interaction of registered users:** Users register but do not publish or purchase materials.	3	2	6	Implement smash push and email notifications: Alert when material of user interest is published.	Arturo - Development team.
RO-05 **Problems with content moderation:** Inappropriate publications, spam, or false information on the platform.	4	3	12	Implement automatic content filters with predefined rules: block external URLs in descriptions, detect prohibited words, limit excessive special characters that indicate spam. Implement a reporting system.	Team
Business Risks					
RN-01 **Low adoption by the supply side (companies):** Lack of interest from companies in cataloging and selling their surpluses.	5	4	20	Focus the value proposition on the ROI (Return on investment) for companies: Create case studies and calculators that demonstrate savings in disposal costs and additional income. Offer an “onboarding” service to help the first companies.	Arturo

RN-02 **Low adoption by the demand side (artisans):** Artisans do not find value, variety, or competitive prices on the platform.	5	4	20	Create a community and ensure an initial critical mass: Conduct a launch focused on a specific artisan niche. Create alliances with local associations and ensure that there is an attractive inventory from day one.	Arturo
Team Risks					
RE-01 **Critical dependence on key members:** The departure of a person with vital project knowledge paralyzes or delays development.	4	3	12	Foster a culture of self-learning and proactive communication: Each member must formally allocate time for research and autonomous study. Use open communication channels so that doubts and solutions are visible to all creating an organic knowledge base and reducing information silos.	Oscar
RE-02 **Technical skills gap in the team:** The team lacks the necessary experience to	4	2	8	Promote continuous training: Each member should encourage their training in the different technologies that	Oscar

solve complex technical challenges.				will be used during the project.	
RE-03 **Deficient communication and lack of commitment:** Misalignment in objectives and low motivation that affect the quality and pace of work.	3	3	9	Establish an agile and transparent work methodology: Implement ceremonies, daily meetings, and retrospectives to ensure constant communication. Use project management tools so that everyone knows the priorities.	Oscar
RE-04 **Lack of dedication due to academic commitments:** Prioritization of other subjects or projects.	5	3	15	Create a project roadmap considering the full academic calendar: examination periods, high-workload weeks, vacations, and major project submission dates.	Oscar
Legal and regulatory Risks					
RLR-01 **Required environmental certifications:** Need to certify recycling or traceability processes.	4	3	12	Initially investigate which certifications are mandatory and optional for a specific business model by consulting with certifying bodies.	Arturo

RLR-02 **Required environmental certifications :** Need to certify recycling or traceability processes.	2	4	8	Initially investigate which certifications are mandatory and optional for a specific business model by consulting with certifying bodies.	Arturo

4.3 MoSCow Priorization

This section defines the priority for all system requirements to guide the implementation of the Minimum Viable Product (MVP).

- **M (Must):** Critical for launch. The system is non-functional or illegal without this feature. All **M** features represent the MVP.
- **S (Should):** Important, but not vital for launch. Adds significant value and should be included if time/resources permit after all **M** features.
- **C (Could):** Desirable ("nice-to-have"). A feature with a small impact that can be easily left for a future release.
- **W (Won't):** Will not be implemented in this project version. This is explicitly out of scope.

ID	Requirement	Priority	Justification
RF-1	User Authentication (Login, Register, Logout)	Must	Core system function. No user can interact with the platform without it.
RF-2	User Profile Management	Must	Users must be able to manage their basic information and identity.
RF-3	Publish Material Listing	Must	Core feature of the B2B marketplace. No "Waste" to sell without this.
RF-4	Manage Material Listings	Must	Sellers must be able to edit or deactivate their own listings.
RF-5	View Material Catalog (Filter/Search)	Must	Core feature. Buyers must be able to find materials.

RF-6	View Material Detail Page	Must	Required for a buyer to make a purchase decision.
RF-7	Address Management (New)	Must	Critical for the checkout process (RF-11). The system must know where to ship orders.
RF-8	Publish Product Listing	Must	Core feature of the B2C marketplace. No "Treasure" to sell without this.
RF-9	Manage Product Listings	Must	Sellers must be able to edit or deactivate their own listings.
RF-10	View Product Catalog (Filter/Search)	Must	Core feature. Buyers must be able to find products.
RF-11	View Product Detail Page	Must	Required for a buyer to make a purchase decision.
RF-12	Shopping Cart & Checkout Flow	Must	Critical. Without this, no transactions can occur.
RF-13	Reputation System (Reviews)	Should	Important for platform trust, but transactions can be completed without it.
RF-16	Report System (Content/User)	Should	Important for platform health, but not a blocker for the MVP launch.
RF-17	Admin Dashboard (View Stats)	Must	Required for the Admin to access the moderation queue (RF-18).
RF-18	Admin Moderation (Approve/Reject)	Must	Critical business rule (2.6.2). Ensures quality control on all listings.
RF-19	Admin Category Management	Should	The MVP can launch with "hard-coded" categories. This adds flexibility later.
RF-20	Admin User Management	Must	Critical. The Admin must be able to assign roles (like "Admin") and block malicious users.

RF-21	Admin Report Management	Should	Linked to RF-16. Not critical if RF-16 is not implemented for the MVP.
RF-22	SaaS Subscription Plans	Should	Core to the business model, but the MVP can function solely on commissions (RF-25).
RF-23	SaaS Metrics & Reports	Should	This is the main feature of the SaaS plans (RF-22).
RF-24	Payment Gateway Integration	Must	Critical. The system must be able to process payments (Stripe).
RF-25	Commission Calculation	Must	The primary revenue stream for the MVP.
RF-26	Order History	Must	Users must be able to see what they bought.
RF-27	Notification System (Email)	Must	Required for user registration confirmation, purchase receipts, and password resets.
RF-28	Traceability (Link Material to Product)	Could	Desirable for the "circular economy" concept, but complex. MVP can rely on description fields.
RF-30	FAQ Section	Could	A static page that can be added at any time. Not a core function.
RF-DIS	Admin Dispute Management	Won't	Out of scope for this release. All disputes will be handled offline via email.

4.4 Legal and Compliance Documentation

This section outlines the legal and compliance documents that the system must support and present to users to ensure transparency and regulatory adherence.

- **Privacy Policy:** The system must display a clear and accessible Privacy Policy before the user completes registration. This document must detail hoe personal information is

collected, stored, and used, in strict compliance with Mexico LFPDPPP (Ley Federal de Protección de Datos Personales en Posesión de los Particulares).

- **Terms of Service:** the user must accept the Terms of Service to create an account. This document governs the use of the platform and includes:
    - The nature of the service (dual marketplace).
    - User account responsibilities.
    - Acceptable use policies and prohibited content.
    - Limitation of liability for the platform.
- **Payment and Commission Policy:** A clear document or section must be available to explain the monetization model to sellers, including:
    - The transaction commission percentage (10%).
    - The costs and features of the SaaS subscription plans.
    - The payment process through the integrated gateways (Stripe).
- **Publication and Moderation Policy:** This defines what content is permitted or prohibited on the platform. This public-facing policy supports the actions of the AdminService (RF-18) and provides justification for content moderation.

4.5 Glossary

Term	Definition
Abstract Class	(UML) A class (like Listing) that cannot be instantiated on its own. It acts as a template for other classes (like Product and Material) to inherit from.
Address	(UML) An entity class that stores structured location data (street, city, state, postal_code) for a user or an order.
AdminService	(UML) A software class that encapsulates all business logic and methods available only to the "Administrator" role (e.g., approveListing, manageUser).
Amplify (AWS)	The AWS service used for hosting, CI/CD pipeline, and deployment of the Next.js front-end application.
API (Application Programming Interface)	The set of rules and protocols that defines how the front-end (Next.js) communicates with the back-end (FastAPI).
API Gateway (AWS)	The AWS service that acts as a secure, single entry point for all API requests to the back-end (FastAPI on EC2).
Association	(UML) A relationship between two classes indicating that objects of one class relate to objects of another (e.g., a User has an Order).
B2B (Business-to-Business)	A business model describing commerce transactions between two businesses. In this project, the Material Marketplace.

B2C (Business-to-Consumer)	A business model describing commerce transactions from a business to an end consumer. In this project, the Product Marketplace.
Back-End	The server-side of the application (FastAPI) that handles business logic, database interactions, and authentication.
BPMN (Business Process Model and Notation)	A standard graphical notation for modeling business processes in a workflow, showing participants and tasks.
Business Rules	A set of policies, constraints, or calculations (like the 10% commission) that the software must enforce.
CI/CD (Continuous Integration / Continuous Deployment)	An automated practice of merging and deploying code, managed in this project by AWS Amplify.
Cognito (AWS)	The AWS service is used for managing user identity, authentication (login/register), and authorization (roles).
Commission	The percentage of a sale (10%) that the platform takes as its primary revenue stream.
Composition	(UML) A strong form of association (indicated by a solid diamond) where a "child" object (OrderItem) cannot exist without its "parent" (Order).
Dependency	(UML) A relationship (indicated by a dashed arrow) where one class (Order) uses another (PaymentService) temporarily, without storing it as an attribute.
Deployment	The process of moving the finished code from a development environment to the live production environment on AWS.
EC2 (AWS)	(Elastic Compute Cloud) The AWS service that provides the virtual server (instance) to host the back-end FastAPI application.
Encapsulation	(UML) The design principle of keeping class attributes private (-) and exposing them only through public (+) methods (getters/setters).
FastAPI	The high-performance Python framework used to build the back-end API.
Front-End	The client-side of the application that the user interacts with in their browser (the React application).
Inheritance	(UML) A relationship (indicated by a solid arrow with a hollow triangle) where a "child" class (Product) gains all the

	attributes and methods of a "parent" class (Listing).
Interface	(UML) A class "contract" (like PaymentService) that defines what methods must exist, but not how they work.
LFPDPPP	(Federal Law on Protection of Personal Data Held by Private Parties) Mexico's data privacy and protection law that the system must comply with.
Listing	(UML) The abstract "parent" class representing a generic item for sale. It is inherited by Material and Product.
MoSCoW	A prioritization technique used to define the MVP, standing for Must have, Should have, Could have, and Won't have.
MVP (Minimum Viable Product)	The version of the project with the fewest features necessary to be functional and satisfy the core business goal (register, publish, buy).
NotificationService	(UML) The interface responsible for sending all system notifications (e.g., welcome email, order confirmation).
Payment Service	(UML) The interface responsible for processing payments, implemented by StripeGateway.
PostgreSQL	The specific type of open-source relational database (SQL) used for this project.
React	The JavaScript library used to build the front-end user interface.
RDS (AWS)	(Relational Database Service) The AWS service that hosts and manages the PostgreSQL database.
S3 (AWS)	(Simple Storage Service) The AWS service used for storing all static files, primarily user-uploaded images for listings.
SaaS (Software as a Service)	A software licensing and delivery model in which software is licensed on a subscription basis. In this project, the "Pro" and "Enterprise" plans.
Sprint	A short, time-boxed period (e.g., one week) in which the team works to complete a set amount of tasks.
SRS (Software Requirements Specification)	This document. It formally defines what the system must do (functionally and non-functionally).
UML (Unified Modeling Language)	A standardized graphical language used to visualize, specify, construct, and document the design of a software system.

4.6 Appendix

Testing System Overview

The Waste to Treasure project has a comprehensive automated testing suite that validates all critical system components. Tests are organized into two main categories:

- **Model Tests**: Validate business logic and database relationships
- **API Tests**: Verify correct functionality of REST endpoints

General Statistics

Total Tests: 224 tests

Status: 100% passing

Execution Time: ~3:42 minutes

Code Coverage: 73% average

Testing Architecture

Technologies Used

Component	Technology	Purpose
Testing Framework	**pytest**	Test execution and organization
Database	**PostgreSQL**	Tests against real DB
Async Support	**pytest-asyncio**	Asynchronous endpoint tests
HTTP Client	**httpx**	Client for API testing
Coverage	**pytest-cov**	Code coverage analysis

Model Tests (204 tests)

3.1 Coverage by Model

Model	Tests	Coverage	Status
User	15	97%	Excellent
Order	12	83%	Good
Category	15	79%	Good
Listing	21	73%	Acceptable
Cart	31	46%	Needs Improvement
Address	20	72%	Good
Payment Customer	11	-	Complete
Payment Transaction	12	-	Complete
Auxiliary Models	40	-	Complete
ListingImage	13	88%	Good
Review	7	95%	Excellent
Report/Offer/Notification	17	-	Complete

3.2 Types of Model Tests

Unit Tests

- Instance creation with required fields

- Default value validation
- Enumeration verification (enums)
- Database constraints (UNIQUE, CHECK, NOT NULL)

Integration Tests

- Relationships between models (Foreign Keys)
- CASCADE behavior on deletions
- Referential integrity constraints
- Complete CRUD operations

Business Logic Tests

- Calculation methods (totals, subtotals, commissions)
- Custom validations
- State transitions
- Specific business rules

Featured Test Cases

- User System
- AWS Cognito Authentication
- python
- Cognito JWT token simulation
- Just-In-Time (JIT) user creation
- Role validation (USER, ADMIN)
- User states (PENDING, ACTIVE, BLOCKED)

Validated Features:

- Unique email per user
- User ID as UUID (simulates Cognito sub)
- Relationships: orders, listings, cart, addresses
- Role-based permissions

API Tests (20 tests)

Public Endpoints

Endpoint	Method	Description	Status

/	GET	Root endpoint	Pass
/api/v1/categories/	GET	List categories	Pass
/api/v1/listings/	GET	List products	Pass
/api/v1/faq/	GET	Frequently asked questions	Pass
/api/v1/legal/	GET	Legal documents	Pass
/api/v1/plans/	GET	Subscription plans	Pass

5.2 Protected Endpoints (Require Authentication)

Endpoint	Method	Description	Status
/api/v1/users/me	GET	User profile	Pass
/api/v1/cart/	GET	View cart	Pass
/api/v1/orders/	GET	My orders	Pass
/api/v1/addresses/	GET	My addresses	Pass
/api/v1/notifications/	GET	Notifications	Pass
/api/v1/offers/	GET	Offers	Pass
/api/v1/subscriptions/	GET	Subscriptions	Pass

5.3 Administrative Endpoints

Endpoint	Method	Description	Status

/api/v1/admin/users	GET	User management	Pass

Appendix A: Schedule of activities

Cronograma de Actividades

Appendix B: Change Log and Technical Decisions

Payment Gateway Scope Update

This section documents a significant change to the project's technical scope and design for the **Waste-To-Treasure (W2T)** platform.

Change: Removal of PayPal Integration.

Rationale: To maintain focus, streamline the development process, and standardize compliance efforts, the project team has decided to remove the implementation of PayPal as a payment gateway from the system's scope.

Current Status: All payment and transaction processing will be exclusively managed through the **Stripe** platform. Stripe remains the sole integrated payment solution for both the Materials Marketplace (B2B) and the Products Marketplace (B2C). This decision ensures a consistent, secure, and unified approach to all financial transactions within W2T.