Web application design

Evidence 1

Professor: Carlos Ignacio Hernandez Nolasco

Raquel Esparza – AL02986561

Perform the analysis and interpretation of the problem and start the project; to do this you must do the following:

- Create the Github repository of the project.
- Choose and justify a work methodology that is ideal for the planning and execution of the project.
- Design the diagrams that will help you to describe graphically how the web application will behave. Among the diagrams that you should design, consider the following:
    - BPMN diagram.
    - Class diagrams
    - Activity diagrams
    - Use case diagrams
- Choose the database and its respective ER diagram containing all the entities with their attributes, and the way they are related.
- Personal reflection.

Analysis and Interpretation of the Problem

1. Understanding the Requirements

The web application for Halcon has two main components:

1. Customer Portal – Allows customers to check order status using their customer number and invoice number.
2. Admin Dashboard – Manages order lifecycle, user roles, and internal workflows.

Key Functional Requirements:

- Customer View:
    - Search order status by customer and invoice number.
    - Display order status ("Ordered," "In Process," "In Route," "Delivered").
    - Show delivery evidence (photo) if status is "Delivered."
- Admin Dashboard:
    - User Management:
        - Default admin user can register new users and assign roles (Sales, Purchasing, Warehouse, Route).
    - Order Management:
        - Sales creates orders (with invoice, customer details, delivery address, notes).
        - Warehouse updates status ("In Process" → "In Route").
        - Route personnel upload photos (loaded & delivered) and update status to "Delivered."
    - Order Listing & Search:
        - Filter by invoice number, customer number, date, or status.
        - Logical deletion (soft delete) with restore option.

Non-Functional Requirements:

- Security: Role-based access control (RBAC) for different departments.
- Usability: Simple UI for customers and structured workflows for employees.
- Scalability: Should handle multiple orders and users efficiently.
- Data Integrity: Orders should not be permanently deleted (soft delete).

2. Choosing the Ideal Methodology

Given the structured requirements and need for iterative development with client feedback, Agile (Scrum) is the best-suited methodology.

Why Agile (Scrum)?

1. Iterative Development – The project can be broken into sprints (e.g., customer portal first, then admin dashboard).
2. Client Feedback Integration – Halcon may refine requirements as they see progress.
3. Flexibility for Changes – New features (e.g., additional order statuses) can be added in future sprints.
4. Clear Role Assignments – Scrum roles (Product Owner, Scrum Master, Dev Team) align well with the structured workflow.

---

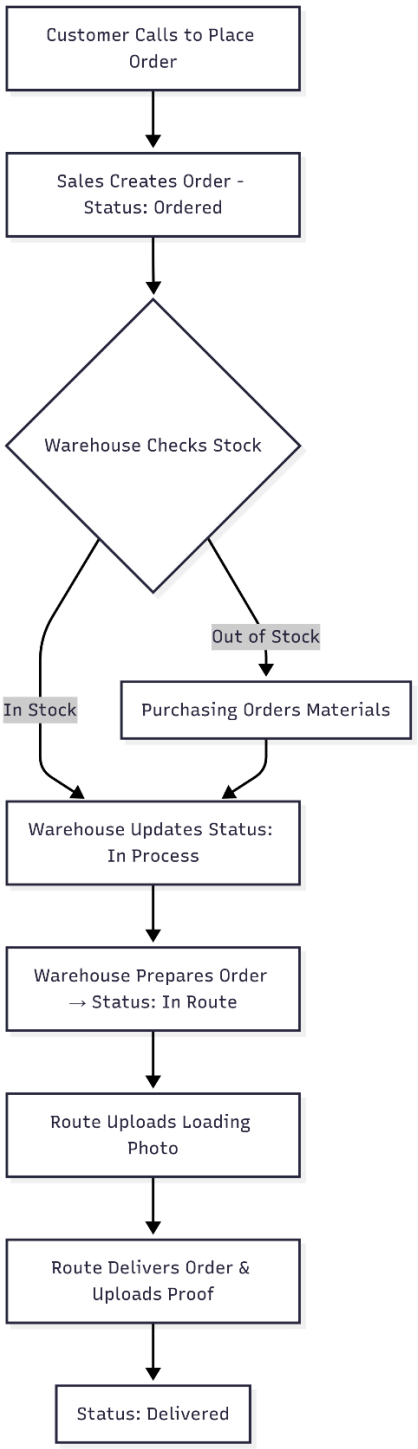
3. Project Execution Plan (Agile-Scrum Approach)

Sprint Breakdown (Example)

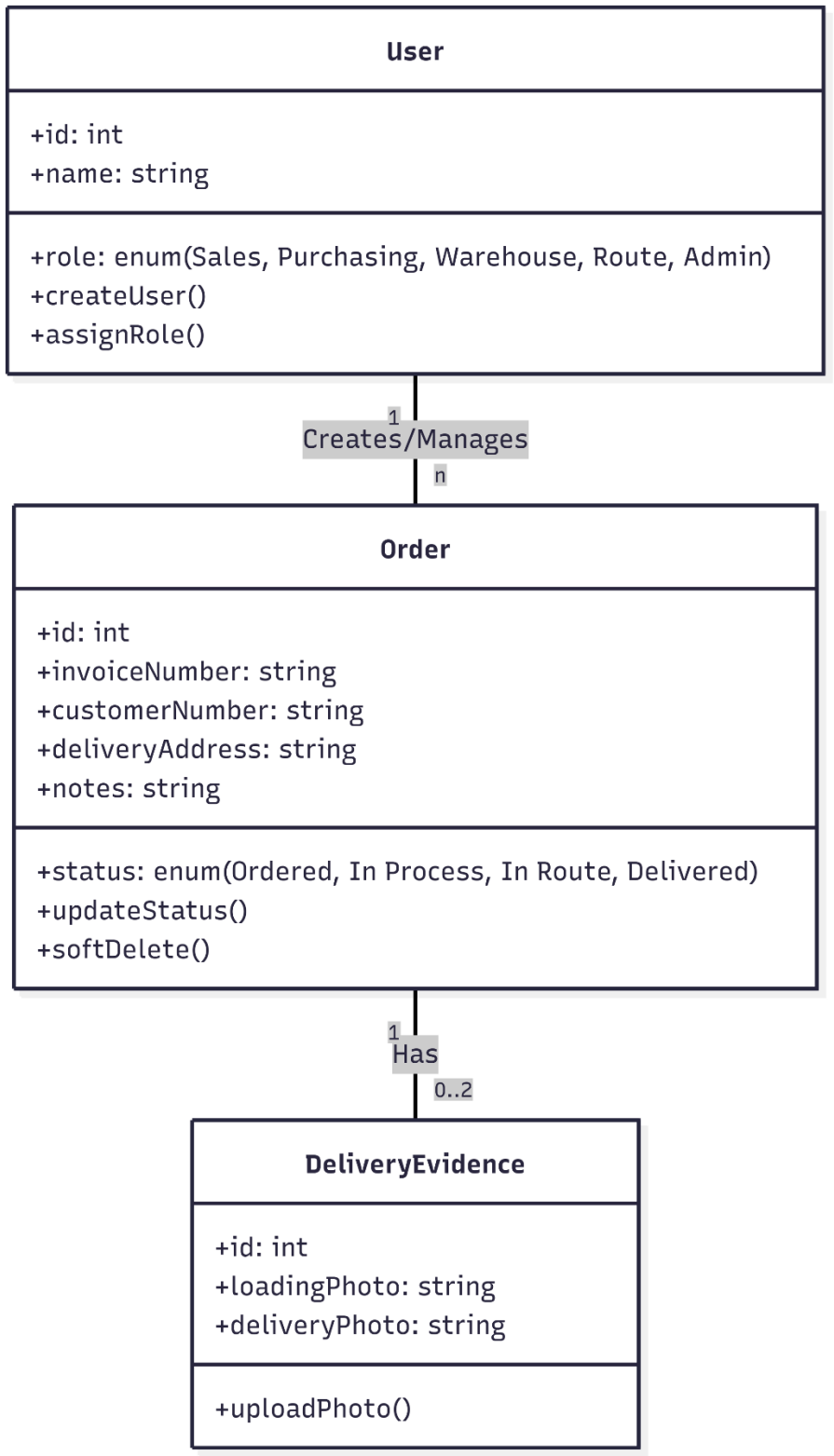
Sprint	Focus	Deliverables
Sprint 1	Authentication & User Roles	Admin user setup, role-based access
Sprint 2	Customer Portal	Order search, status display, photo evidence
Sprint 3	Order Creation (Sales)	Form for new orders, default "Ordered" status
Sprint 4	Warehouse & Route Workflow	Status updates, photo uploads
Sprint 5	Order Management	Search/filter, soft delete, restore
Sprint 6	Testing & Deployment	Bug fixes, UAT, launch

Design the diagrams that will help you to describe graphically how the web application will behave. Among the diagrams that you should design, consider the following:

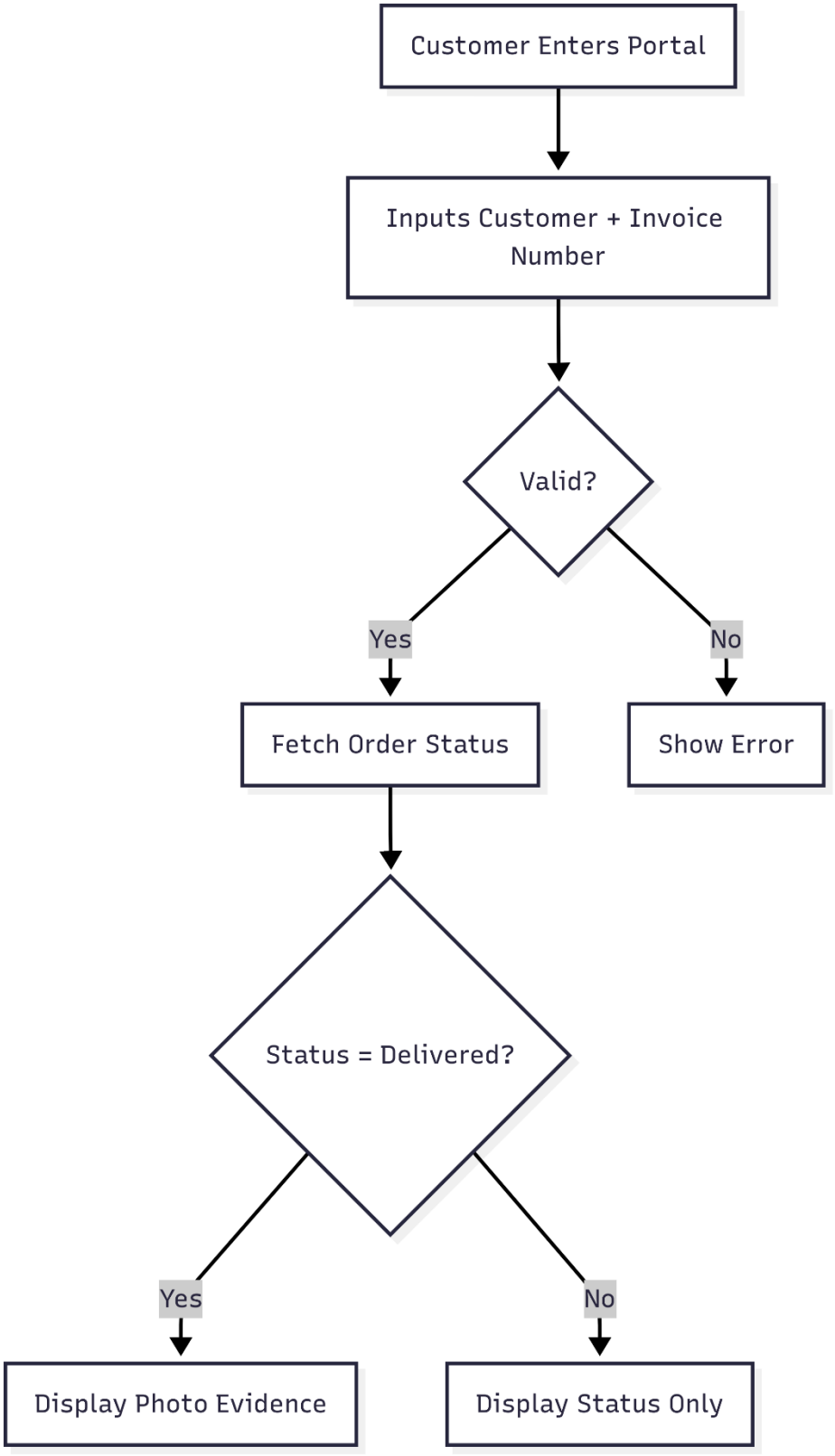
BPMN diagram



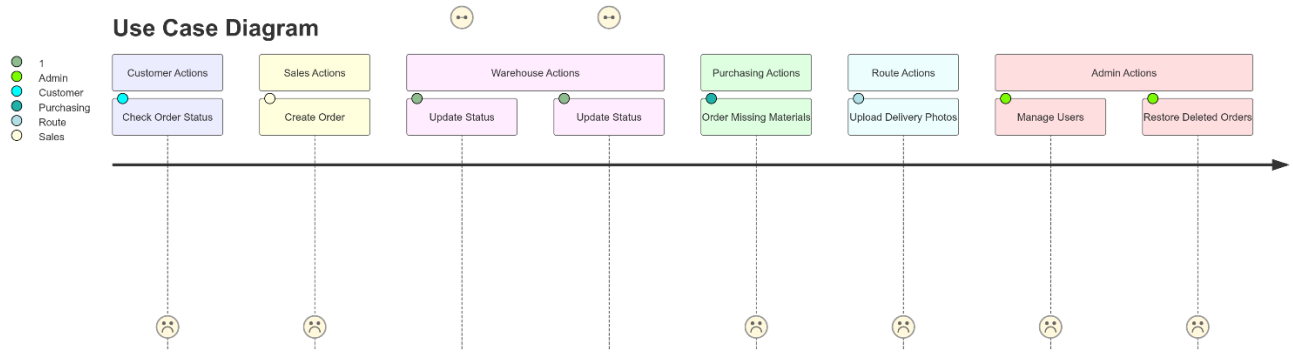
Class diagram



Activity diagram



Use case diagram



Choose the database and its respective ER diagram containing all the entities with their attributes, and the way they are related.

Chosen Database: PostgreSQL

Justification:

- Relational Structure – Fits Halcon's structured data (orders, users, delivery evidence).
- ACID Compliance – Ensures data integrity for order status transitions.
- Scalability – Handles growing order volumes efficiently.
- Role-Based Security – Native support for user permissions.

Key Entities & Relationships

USER

- Attributes: id, role (Admin/Sales/Purchasing/Warehouse/Route), credentials.
- Relationships: 1-to-Many with ORDER (Sales creates orders).

ORDER

- Attributes: status, invoice_number, customer_details, is_deleted (soft delete).
- Relationships: Many-to-1 with USER (creator). 1-to-Many with DELIVERY_EVIDENCE.

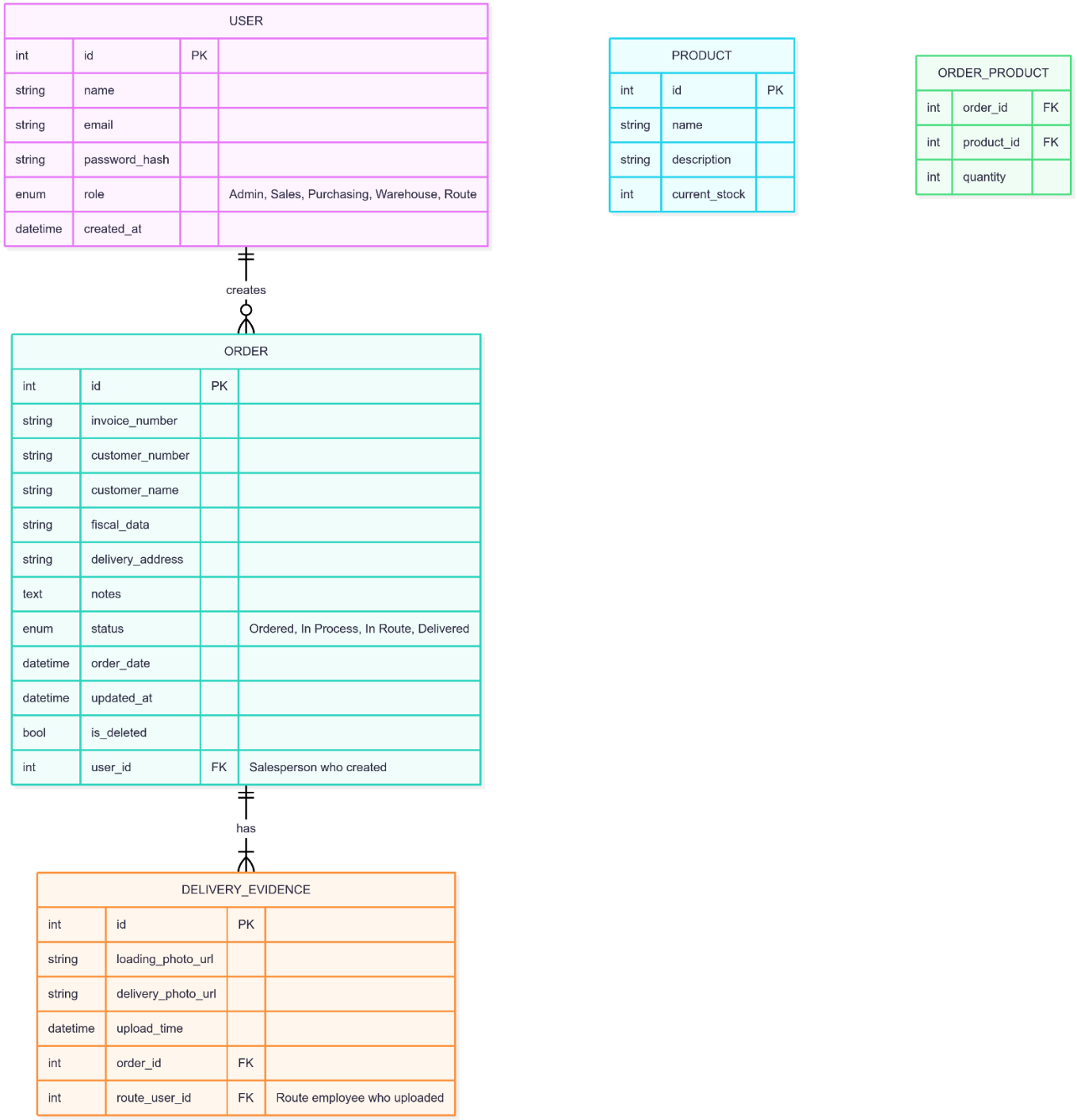
DELIVERY_EVIDENCE

- Attributes: Photo URLs, timestamps.

- Relationships: Many-to-1 with ORDER and USER (Route employee uploader).

PRODUCT (Optional for inventory extension)

- Tracks stock levels for warehouse/purchasing teams.
- Many-to-Many with ORDER via ORDER_PRODUCT junction table.



Personal reflection:

This project has been a great opportunity to design a web application that solves real-world problems for Halcon, a construction material distributor. By analyzing their needs, we identified two main parts:

- A customer portal – Where clients can check their order status easily.
- An admin dashboard – Where employees manage orders, track deliveries, and control user access.

Key Takeaways

- Agile Methodology Works Best – Instead of planning everything at once, we broke the project into smaller steps (sprints). This allows for flexibility and quick adjustments based on feedback.
- Clear Workflows Matter – The BPMN and State Machine diagrams helped visualize how orders move from "Ordered" to "Delivered," ensuring no steps are missed.
- Security & Roles Are Essential – Not everyone should have the same access. Sales, Warehouse, Route, and Admin each have specific permissions to prevent errors and misuse.
- Data Must Be Protected & Organized – Using PostgreSQL, we designed a database that keeps orders, users, and delivery evidence connected while allowing for future growth.
- User Experience Should Be Simple – Customers only need to enter two details (customer number + invoice) to track orders, while employees get a structured system to manage workflows.

Final Thoughts

This project taught me how important it is to listen to the client's needs and turn them into a functional, easy-to-use system. By using diagrams (BPMN, ER, Use Cases), we will ensure everyone—developers, managers, and Halcon's team—understands how the system works before coding even begins.

Lesson learned: A well-structured plan saves time, reduces confusion, and leads to a better final product.