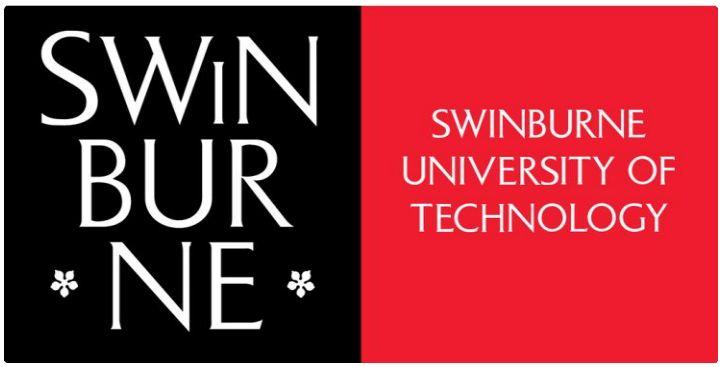


COS20031

Computing Technology Design Project

Progress Report

Instructor: Dr. Hai Sam Nang

Project Team Member:

- Nguyen Thuan Khang (104171078)
- Doan Nhat Long (104167705)
- Le Ba Tung (104175915)
- Lau Ngoc Quyen (104198996)

Team Progress Report Overview

Purpose: This progress report evaluates the team's performance over the past three week, focusing on key aspects such as the use of project management and collaboration tools, progress on deliverables, and the team's overall health. The purpose is to reflect on the team's effectiveness, ensure tasks and objectives are being met, and identify areas for improvement.

Key Areas of Evaluation:

1. **Management and Collaboration Tools:**

   This assesses the team's efficiency in using tools like Jira for task management and Confluence for collaboration. A good rating reflects comprehensive organization and forward planning, utilizing advanced Confluence features, while a satisfactory rating focuses on basic usage such as task creation, assignment, and collaboration evidence.

2. **Progress on Deliverables:**

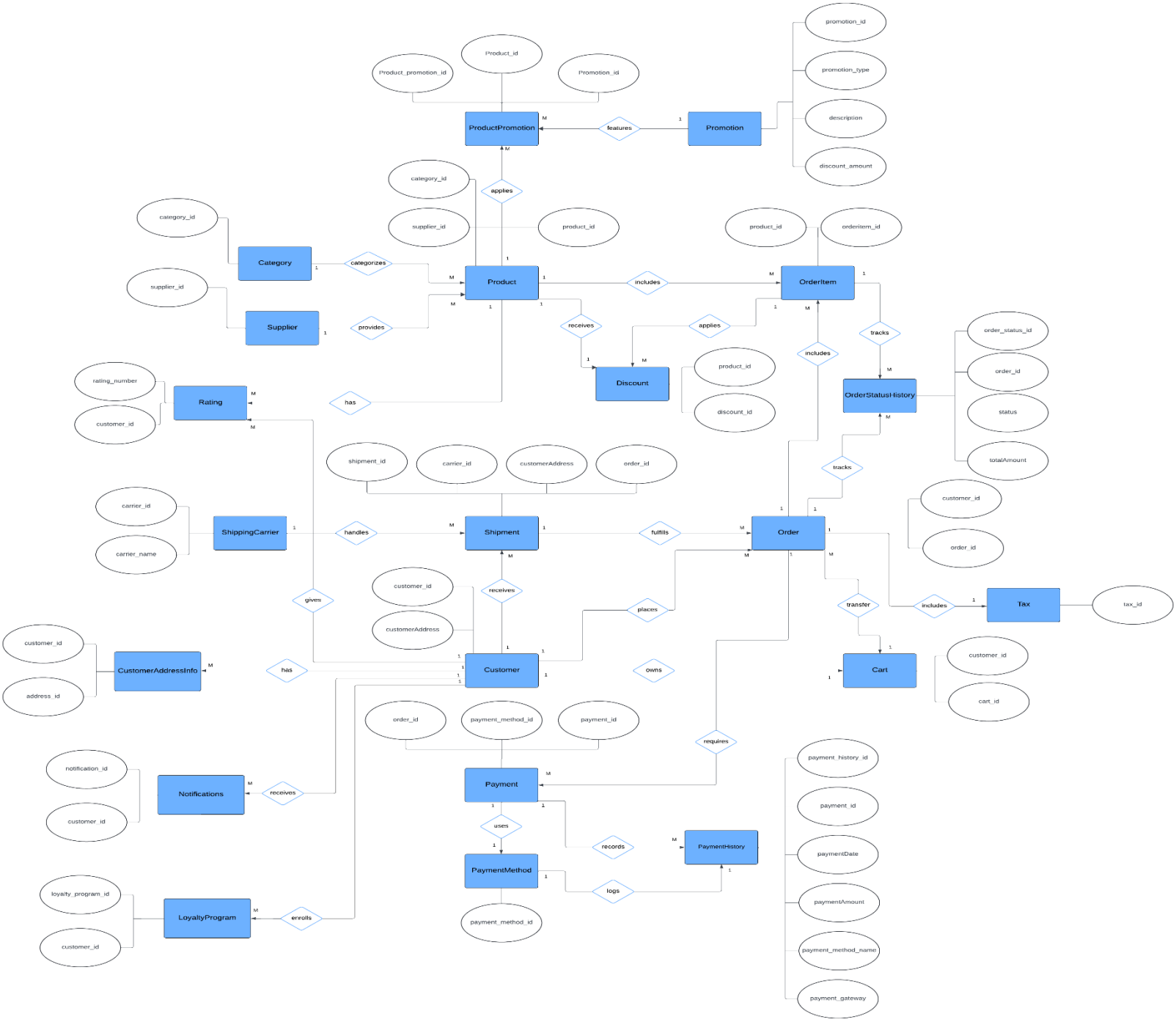
   This evaluates the team's progress in researching, planning, and completing project deliverables. Teams that have moved beyond planning and started creating delivera bles will be rated higher, while teams still in the early planning stages will receive satisfactory marks.

3. **Team Health Check:**

   This reflects on how the team functions together, identifying strengths and weaknesses in the team's process. A good rating requires the team to not only conduct a health check but also take action on identified weaknesses to improve workflow, while satisfactory ratings are given to teams that complete the health check but have not yet taken steps to address any identified issues.

Initial ER Diagram

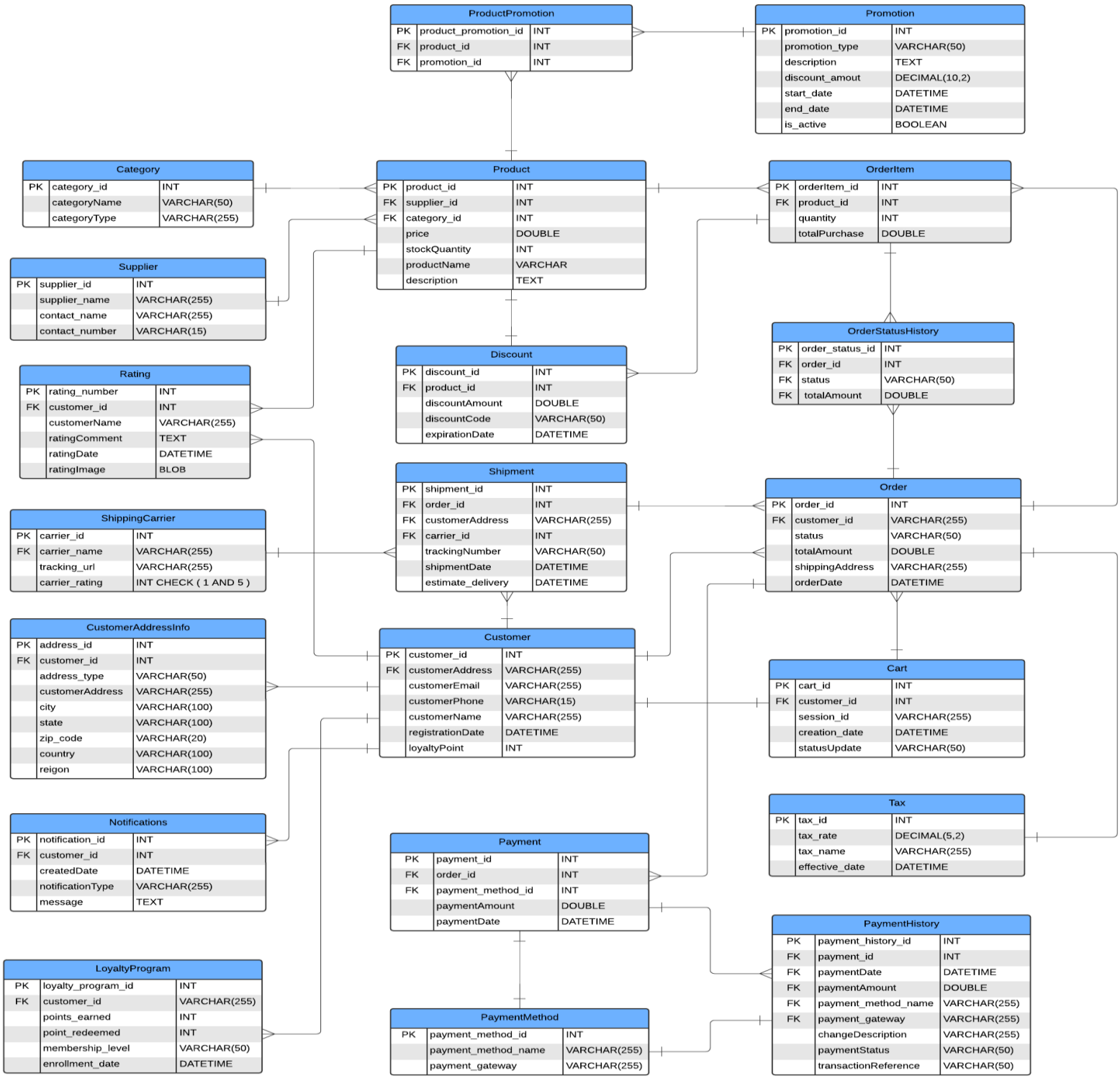
Conceptual Relationship Diagram



Conceptual Relationship Diagram

Entity Relationship Diagram

Note: Foreign Key (FK), Primary Key (PK)



ERD Explaination and Relationship

ERD Explaination :

1. Supplier

- **Purpose**: Represents suppliers who provide products to the business.
- **Attributes**:
    - **supplier_id**: Unique identifier for each supplier (PK).
    - **supplier_name**: Name of the supplier.
    - **contact_name**: Primary contact person at the supplier.
    - **contact_number**: Contact number for the supplier.

2. Category

- **Purpose**: Categorizes products to make it easier for customers to browse.
- **Attributes**:
    - **category_id**: Unique identifier for each category (PK).
    - **category_name**: Name of the category (e.g., electronics, clothing).
    - **category_type**: Additional type information about the category.

3. Product

- **Purpose**: Represents products that can be sold.
- **Attributes**:
    - **product_id**: Unique identifier for each product (PK).
    - **supplier_id**: Identifier for the supplier of the product (FK).
    - **category_id**: Identifier for the category the product belongs to (FK).
    - **price**: Selling price of the product.
    - **stock_quantity**: Number of items available in stock.
    - **product_name**: Name of the product.
    - **description**: Description of the product features.

4. ShippingCarrier

- **Purpose**: Represents shipping carriers used for order deliveries.
- **Attributes**:
    - **carrier_id**: Unique identifier for each carrier (PK).
    - **carrier_name**: Name of the shipping carrier.
    - **tracking_url**: URL for tracking shipments.
    - **carrier_rating**: Rating of the carrier (1-5).

5. Shipment

- **Purpose**: Records details of shipments for orders.
- **Attributes**:
    - **shipment_id**: Unique identifier for each shipment (PK).
    - **order_id**: Identifier for the order being shipped (FK).
    - **customer_address**: Address where the order is shipped.
    - **carrier_id**: Identifier for the shipping carrier (FK).
    - **tracking_number**: Unique number for tracking the shipment.
    - **shipment_date**: Date the shipment was made.
    - **estimate_delivery**: Estimated delivery date.

6. OrderStatusHistory

- **Purpose**: Tracks the status changes of an order.
- **Attributes**:
    - **order_status_id**: Unique identifier for each order status entry (PK).
    - **order_id**: Identifier for the associated order (FK).
    - **status**: Current status of the order (e.g., processing, shipped).
    - **total_amount**: Total amount for the order at that status.

7. Customer

- **Purpose**: Represents customers who place orders.

- **Attributes**:
    - **customer_id**: Unique identifier for each customer (PK).
    - **customer_address**: Address of the customer.
    - **customer_email**: Email address of the customer.
    - **customer_phone**: Phone number of the customer.
    - **customer_name**: Name of the customer.
    - **registration_date**: Date when the customer registered.
    - **loyalty_points**: Points accumulated by the customer in the loyalty program.

8. Order

- **Purpose**: Represents customer orders.
- **Attributes**:
    - **order_id**: Unique identifier for each order (PK).
    - **customer_id**: Identifier for the customer who placed the order (FK).
    - **status**: Current status of the order (e.g., completed, canceled).
    - **total_amount**: Total amount of the order.
    - **shipping_address**: Address where the order will be shipped.
    - **order_date**: Date the order was placed.

9. OrderItem

- **Purpose**: Represents individual items in an order.
- **Attributes**:
    - **order_item_id**: Unique identifier for each order item (PK).
    - **order_id**: Identifier for the associated order (FK).
    - **product_id**: Identifier for the product being ordered (FK).
    - **quantity**: Quantity of the product ordered.
    - **total_purchase**: Total cost for this item (quantity x price).

10. Discount

- **Purpose**: Represents discounts applicable to products.
- **Attributes**:
    - **discount_id**: Unique identifier for each discount (PK).
    - **product_id**: Identifier for the product to which the discount applies (FK).
    - **discount_amount**: Amount or percentage of the discount.
    - **discount_code**: Code required to apply the discount.
    - **expiration_date**: Date when the discount expires.

11. CustomerAddressInfo

- **Purpose**: Stores multiple addresses for customers.
- **Attributes**:
    - **address_id**: Unique identifier for each address entry (PK).
    - **customer_id**: Identifier for the customer (FK).
    - **address_type**: Type of address (e.g., billing, shipping).
    - **customer_address**: Street address.
    - **city**: City of the address.
    - **state**: State of the address.
    - **zip_code**: Postal code.
    - **country**: Country of the address.
    - **region**: Region information.

12. Rating

- **Purpose**: Captures customer ratings and feedback on products.
- **Attributes**:
    - **rating_id**: Unique identifier for each rating (PK).
    - **customer_id**: Identifier for the customer who made the rating (FK).
    - **rating_comment**: Customer's comments about the product.
    - **rating_date**: Date when the rating was submitted.

- **rating_image**: Optional image uploaded with the rating.

13. Tax

- **Purpose**: Represents tax rates applicable to purchases.
- **Attributes**:
  - **tax_id**: Unique identifier for each tax entry (PK).
  - **tax_rate**: Tax percentage.
  - **tax_name**: Name of the tax (e.g., VAT, sales tax).
  - **effective_date**: Date when the tax rate becomes effective.

14. PaymentMethod

- **Purpose**: Stores payment methods available to customers.
- **Attributes**:
  - **payment_method_id**: Unique identifier for each payment method (PK).
  - **payment_method_name**: Name of the payment method (e.g., credit card, PayPal).
  - **payment_gateway**: Gateway used for processing the payment.

15. Cart

- **Purpose**: Represents the shopping cart of a customer.
- **Attributes**:
  - **cart_id**: Unique identifier for each cart (PK).
  - **customer_id**: Identifier for the customer (FK).
  - **session_id**: Identifier for the customer's session.
  - **creation_date**: Date when the cart was created.
  - **status_update**: Current status of the cart (e.g., active, abandoned).

16. Payment

- **Purpose**: Records payments made for orders.
- **Attributes**:

- **payment_id**: Unique identifier for each payment (PK).
- **order_id**: Identifier for the associated order (FK).
- **payment_method_id**: Identifier for the payment method used (FK).
- **payment_amount**: Amount paid.
- **payment_date**: Date when the payment was made.

17. PaymentHistory

- **Purpose**: Tracks changes and history of payments.
- **Attributes**:
  - **payment_history_id**: Unique identifier for each payment history entry (PK).
  - **payment_id**: Identifier for the associated payment (FK).
  - **payment_date**: Date of the payment.
  - **payment_amount**: Amount of the payment.
  - **payment_method_name**: Name of the payment method used.
  - **payment_gateway**: Gateway used for the transaction.
  - **change_description**: Description of any changes to the payment.
  - **payment_status**: Current status of the payment (e.g., completed, pending).
  - **transaction_reference**: Reference number for the transaction.

18. Notifications

- **Purpose**: Sends notifications to customers about their orders or promotions.
- **Attributes**:
  - **notification_id**: Unique identifier for each notification (PK).
  - **customer_id**: Identifier for the customer receiving the notification (FK).
  - **created_date**: Date when the notification was created.
  - **notification_type**: Type of notification (e.g., order update, promotion).
  - **message**: Message content of the notification.

19. LoyaltyProgram

- **Purpose**: Tracks customer participation in loyalty programs.
- **Attributes**:
    - **loyalty_program_id**: Unique identifier for each loyalty program entry (PK).
    - **customer_id**: Identifier for the customer in the loyalty program (FK).
    - **points_earned**: Total points earned by the customer.
    - **points_redeemed**: Total points redeemed by the customer.
    - **membership_level**: Level of membership in the loyalty program (e.g., silver, gold).
    - **enrollment_date**: Date when the customer enrolled in the program.

20. Promotion

- **Purpose**: Represents special promotions available to customers.
- **Attributes**:
    - **promotion_id**: Unique identifier for each promotion (PK).
    - **promotion_type**: Type of promotion (e.g., seasonal sale, clearance).
    - **description**: Description of the promotion.
    - **discount_amount**: Amount of discount offered.
    - **start_date**: Date when the promotion starts.
    - **end_date**: Date when the promotion ends.
    - **is_active**: Indicates whether the promotion is currently active.

21. ProductPromotion

- **Purpose**: Links products to specific promotions.
- **Attributes**:
    - **product_promotion_id**: Unique identifier for each product promotion (PK).
    - **product_id**: Identifier for the associated product (FK).
    - **promotion_id**: Identifier for the associated promotion (FK).

Relationship Between ERD entities:

1. Supplier and Product:

- **Relationship**: One-to-Many.
- **Explanation**: Each supplier can provide multiple products, but each product is supplied by one specific supplier.
- **Foreign Key**: supplier_id in the Product table references supplier_id in the Supplier table.

Example: A coffee supplier (from Đắk Lắk, Lâm Đồng) might offer various types of coffee beans, and all those products will have the same supplier_id.

---

2. Category and Product:

- **Relationship**: One-to-Many.
- **Explanation**: A category can include multiple products (e.g., "Coffee Beans"), but each product belongs to one category.
- **Foreign Key**: category_id in the Product table references category_id in the Category table.

Example: The category "Coffee Machines" will include various products like espresso machines, grinders, etc., but each machine belongs to that single category.

---

3. Customer and Order:

- **Relationship**: One-to-Many.
- **Explanation**: A customer can place multiple orders over time, but each order belongs to only one customer.
- **Foreign Key**: customer_id in the Order table references customer_id in the Customer table.

Example: A customer named Long can have multiple orders across different dates, but each order belongs to Long only.

---

4. Order and OrderItem:

- **Relationship**: One-to-Many.
- **Explanation**: An order can have multiple items (products), but each item belongs to a single order.
- **Foreign Key**: order_id in the OrderItem table references order_id in the Order table.

Example: An order placed by a customer may include 3 products (like coffee beans, a grinder, and filters), which would each have an entry in OrderItem linked to the same order_id.

---

5. Product and OrderItem:

- **Relationship**: One-to-Many.
- **Explanation**: A product can appear in multiple order items (since many customers can order the same product), but each order item contains only one product.
- **Foreign Key**: product_id in the OrderItem table references product_id in the Product table.

Example: The product "Arabica, Robusta Coffee Beans" can appear in many different orders placed by various customers.

---

6. Shipment and Order:

- **Relationship**: One-to-Many.
- **Explanation**: Each order can have only one shipment associated with it, and each shipment refers to one specific order.
- **Foreign Key**: order_id in the Shipment table references order_id in the Order table.

Example: Once an order is processed, it is shipped out as a single shipment (E.g:Shopee, Lazada).

---

7. ShippingCarrier and Shipment:

- **Relationship**: One-to-Many.
- **Explanation**: A shipping carrier can handle multiple shipments, but each shipment is handled by a single carrier.
- **Foreign Key**: carrier_id in the Shipment table references carrier_id in the ShippingCarrier table.

Example: A shipment might be delivered via Lazada, Shopee, or XanhSM, and the carrier_id would identify which one.

---

8. Customer and CustomerAddressInfo:

- **Relationship**: One-to-Many.
- **Explanation**: A customer can have multiple addresses (e.g., home, work), but each address belongs to one customer.
- **Foreign Key**: customer_id in the CustomerAddressInfo table references customer_id in the Customer table.

Example: Khang can have multiple shipping addresses like his home and office, but each address is tied only to Khang.

---

9. Customer and Rating:

- **Relationship**: One-to-Many.
- **Explanation**: A customer can leave multiple ratings (for different products), but each rating belongs to one customer.
- **Foreign Key**: customer_id in the Rating table references customer_id in the Customer table.

Example: Tung might leave a rating for "Espresso Machine" and another for "Coffee Beans," but both ratings belong to Tung.

---

10. Product and Discount:

- **Relationship**: One-to-One.
- **Explanation**: A product can have multiple discounts applied over time, but each discount applies to one specific product.
- **Foreign Key**: product_id in the Discount table references product_id in the Product table.

Example: A product might have a 10% discount one week, followed by a 5%-off discount the next week.

---

11. Product and ProductPromotion:

- **Relationship**: One-to-Many.
- **Explanation**: A product can be part of multiple promotions, and a promotion can apply to multiple products. The ProductPromotion table resolves this many-to-many relationship.
- **Foreign Keys**:
    - product_id in ProductPromotion references product_id in Product.
    - promotion_id in ProductPromotion references promotion_id in Promotion.

Example: A promotion for "Free Shipping" (E.g: Grab, Shopee) could apply to several products, while a single product could be part of a "Buy One, Get One" and a "10% off" promotion simultaneously.

---

12. Customer and Cart:

- **Relationship**: One-to-One.
- **Explanation**: A customer can have multiple shopping sessions (carts), but each cart belongs to one customer.
- **Foreign Key**: customer_id in the Cart table references customer_id in the Customer table.

Example: A customer might browse and create a cart today and then start a new shopping session next week, creating a new cart.

---

13. Order and Payment:

- **Relationship**: One-to-Many.
- **Explanation**: An order can have multiple payments (e.g., partial payments, advance payment), but each payment is associated with a specific order.
- **Foreign Key**: order_id in the Payment table references order_id in the Order table.

Example: An order might be paid for in two installments, or one full payment.

---

14. Payment and PaymentHistory:

- **Relationship**: One-to-Many.
- **Explanation**: A payment can have multiple records of changes or statuses, but each change or status is tied to one payment.
- **Foreign Key**: payment_id in the PaymentHistory table references payment_id in the Payment table.

Example: A payment (E.g: Zalopay, Momo) might first be marked as "Pending" and later updated to "Completed" in the PaymentHistory table.

---

15. Customer and Notifications:

- **Relationship**: One-to-Many.
- **Explanation**: A customer can receive multiple notifications (e.g., for order updates, promotional messages), but each notification is tied to one customer.
- **Foreign Key**: customer_id in the Notifications table references customer_id in the Customer table.

Example: A customer might receive a notification when their order is shipped and another when a promotion is live.

---

16. Customer and LoyaltyProgram:

- **Relationship**: One-to-Many.
- **Explanation**: A customer has only one loyalty program membership, and each loyalty program membership belongs to one customer.
- **Foreign Key**: customer_id in the LoyaltyProgram table references customer_id in the Customer table.

Example: A customer can earn points from purchases and then redeem those points based on their loyalty program level (e.g., "Vip1" or "Vip2").

---

17. Promotion and ProductPromotion:

- **Relationship**: One-to-Many.
- **Explanation**: A promotion can apply to many products, and a product can be part of many promotions. This is managed by the ProductPromotion table.
- **Foreign Keys**:
    - promotion_id in ProductPromotion references promotion_id in Promotion.
    - product_id in ProductPromotion references product_id in Product.

Example: A "Black Friday, Mega Live" promotion could apply to various products, and a single product could be eligible for multiple promotions.

---

18. Order and Tax:

- **Relationship:** One-to-One.
- **Explanation:** Each order can have exactly one associated invoice, and each invoice corresponds to exactly one order. This relationship ensures that every order generates a unique invoice.
- **Foreign Key:** order_id in the Tax table references order_id in the Order table.

Example: Each order on Trung Nguyen Coffee Corp's e-commerce platform will have a unique associated tax record, ensuring accurate tax calculation and compliance with regional regulations.

---

19. Product and Rating:

- **Relationship:** One-to-Many.
- **Explanation:** Each product can have multiple ratings given by different customers, while each rating corresponds to exactly one product. This allows for a product to receive feedback from many users, providing a comprehensive view of its quality and performance.
- **Foreign Key:** product_id in Rating references product_id in Product

Example: Each “coffee sữa, coffee trứng” can receive multiple ratings from different customers, allowing for comprehensive feedback on quality and performance through user reviews.

---

20. PaymentMethod and PaymentHistory:

- **Relationship:** One-to-One.
- **Explanation:** Each payment method can have one corresponding entry in the payment history, which details the status and specifics of a transaction made using that payment method. This ensures that every payment method is associated with a unique historical record.
- **Foreign Key:** payment_method_id in the PaymentHistory table references payment_method_id in the PaymentMethod table.

Example: Each payment method is linked to a unique payment history entry, providing specific transaction details and ensuring accurate record-keeping for transactions.

---

21. Cart and Order:

- **Relationship:** One-to-Many.
- **Explanation:** Each cart can be converted into a single order upon checkout, and each order originates from one specific cart.
- **Foreign Key:** cart_id in the Order table references cart_id in the Cart table.

Example: Each cart can generate multiple orders over time, while every order is derived from a single cart, ensuring clear tracking of purchases.

---

22. Payment and PaymentMethod:

- **Relationship:** One-to-One.
- **Explanation:** A payment can be associated with a specific payment method, but each payment method can be used for multiple payments.
- **Foreign Key:** payment_method_id in the Payment table references payment_method_id in the PaymentMethod table.

Example: Each payment is linked to a specific payment method, while that payment method can facilitate multiple payments (E.g: cash, bank), allowing for streamlined transaction processing.

---

23. OrderItem and Discount:

- **Relationship:** One-to-Many.
- **Explanation:** Each order item can have multiple discounts applied to it, while each discount can be associated with many order items. This allows flexibility in managing how discounts are applied to specific items within an order.
- **Foreign Key:** order_item_id in the Discount table references order_item_id in the OrderItem table.

Example: Each order item can receive various discounts, and each discount can be applied to multiple order items, enabling dynamic pricing strategies and promotions.

---

24. OrderItem and OrderStatusHistory:

- **Relationship:** One-to-Many.
- **Explanation:** Each order item can have multiple status history entries associated with it, tracking its status changes throughout the order process (e.g., processing, shipped, delivered, canceled). Conversely, each status entry pertains to exactly one order item.
- **Foreign Key:** order_item_id in the OrderStatusHistory table references order_item_id in the OrderItem table.

Example: Each order item can have numerous status history entries, detailing its journey through various stages, while each status entry corresponds to a single order item.

---

25. Order and OrderStatusHistory:

- **Relationship:** One-to-Many.
- **Explanation:** Each order can have multiple status history entries associated with it, tracking changes to the order's status throughout its lifecycle (e.g., placed, processing, shipped, delivered, canceled). Conversely, each status entry pertains to exactly one order.
- **Foreign Key:** order_id in the OrderStatusHistory table references order_id in the OrderItem table.

Example: Each order can have multiple status history entries that document its lifecycle, with each entry related to a specific order, ensuring transparent tracking of order progress.

---

26. Customer and Shipment:

- **Relationship:** One-to-Many.
- **Explanation:** Each customer can have multiple shipments associated with their orders, while each shipment corresponds to exactly one customer. This relationship allows for the tracking of all shipments that a customer has received over time.
- **Foreign Key:** customer_id in the Shipment table references customer_id in the Customer table.

Example: Each customer can have multiple shipments (E.g: Bee, XanhSM) linked to their orders, with each shipment associated with a single customer, enabling effective tracking of all received orders.

SQL Queries:

```
CREATE TABLE Supplier (
supplier_id INT AUTO_INCREMENT,
supplier_name VARCHAR(255) NOT NULL,
contact_name VARCHAR(255) NOT NULL,
contact_number VARCHAR(15) NOT NULL,
PRIMARY KEY (supplier_id)
);

CREATE TABLE Category (
category_id INT AUTO_INCREMENT,
category_name VARCHAR(50) NOT NULL,
category_type VARCHAR(255) NOT NULL,
PRIMARY KEY (category_id)
);

CREATE TABLE Product (
product_id INT AUTO_INCREMENT,
supplier_id INT NOT NULL,
category_id INT NOT NULL,
price DECIMAL(10, 2) NOT NULL,
stock_quantity INT NOT NULL,
product_name VARCHAR(255) NOT NULL,
description TEXT NOT NULL,
PRIMARY KEY (product_id),
FOREIGN KEY (supplier_id) REFERENCES Supplier(supplier_id),
FOREIGN KEY (category_id) REFERENCES Category(category_id)
);

CREATE TABLE ShippingCarrier (
carrier_id INT AUTO_INCREMENT,
carrier_name VARCHAR(255) NOT NULL,
tracking_url VARCHAR(255) NOT NULL,
carrier_rating INT CHECK (carrier_rating BETWEEN 1 AND 5) NOT NULL,
PRIMARY KEY (carrier_id)
);
```

```
CREATE TABLE Shipment (
shipment_id INT AUTO_INCREMENT,
order_id INT NOT NULL,
customer_address VARCHAR(255) NOT NULL,
carrier_id INT NOT NULL,
tracking_number VARCHAR(50) NOT NULL,
shipment_date DATETIME NOT NULL,
estimate_delivery DATETIME NOT NULL,
PRIMARY KEY (shipment_id),
FOREIGN KEY (carrier_id) REFERENCES ShippingCarrier(carrier_id)
);

CREATE TABLE OrderStatusHistory (
order_status_id INT AUTO_INCREMENT,
order_id INT NOT NULL,
status VARCHAR(50) NOT NULL,
total_amount DECIMAL(10, 2) NOT NULL,
PRIMARY KEY (order_status_id)
);

CREATE TABLE Customer (
customer_id INT AUTO_INCREMENT,
customer_address VARCHAR(255) NOT NULL,
customer_email VARCHAR(255) NOT NULL,
customer_phone VARCHAR(15) NOT NULL,
customer_name VARCHAR(255) NOT NULL,
registration_date DATETIME NOT NULL,
loyalty_points INT NOT NULL DEFAULT 0,
PRIMARY KEY (customer_id)
);

CREATE TABLE Order (
order_id INT AUTO_INCREMENT,
customer_id INT NOT NULL,
status VARCHAR(50) NOT NULL,
total_amount DECIMAL(10, 2) NOT NULL,
shipping_address VARCHAR(255) NOT NULL,
order_date DATETIME NOT NULL,
PRIMARY KEY (order_id),
```

```
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE OrderItem (
order_item_id INT AUTO_INCREMENT,
order_id INT NOT NULL,
product_id INT NOT NULL,
quantity INT NOT NULL,
total_purchase DECIMAL(10, 2) NOT NULL,
PRIMARY KEY (order_item_id),
FOREIGN KEY (order_id) REFERENCES Order(order_id),
FOREIGN KEY (product_id) REFERENCES Product(product_id)
);

CREATE TABLE Discount (
discount_id INT AUTO_INCREMENT,
product_id INT NOT NULL,
discount_amount DECIMAL(10, 2) NOT NULL,
discount_code VARCHAR(50) NOT NULL,
expiration_date DATETIME NOT NULL,
PRIMARY KEY (discount_id),
FOREIGN KEY (product_id) REFERENCES Product(product_id)
);

CREATE TABLE CustomerAddressInfo (
address_id INT AUTO_INCREMENT,
customer_id INT NOT NULL,
address_type VARCHAR(50) NOT NULL,
customer_address VARCHAR(255) NOT NULL,
city VARCHAR(100) NOT NULL,
state VARCHAR(100) NOT NULL,
zip_code VARCHAR(20) NOT NULL,
country VARCHAR(100) NOT NULL,
region VARCHAR(100) NOT NULL,
PRIMARY KEY (address_id),
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE Rating (
rating_id INT AUTO_INCREMENT,
```

```
customer_id INT NOT NULL,
rating_comment TEXT NOT NULL,
rating_date DATETIME NOT NULL,
rating_image BLOB,
PRIMARY KEY (rating_id),
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE Tax (
tax_id INT AUTO_INCREMENT,
tax_rate DECIMAL(5, 2) NOT NULL,
tax_name VARCHAR(255) NOT NULL,
effective_date DATETIME NOT NULL,
PRIMARY KEY (tax_id)
);

CREATE TABLE PaymentMethod (
payment_method_id INT AUTO_INCREMENT,
payment_method_name VARCHAR(255) NOT NULL,
payment_gateway VARCHAR(255) NOT NULL,
PRIMARY KEY (payment_method_id)
);

CREATE TABLE Cart (
cart_id INT AUTO_INCREMENT,
customer_id INT NOT NULL,
session_id VARCHAR(255) NOT NULL,
creation_date DATETIME NOT NULL,
status_update VARCHAR(50) NOT NULL,
PRIMARY KEY (cart_id),
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE Payment (
payment_id INT AUTO_INCREMENT,
order_id INT NOT NULL,
payment_method_id INT NOT NULL,
payment_amount DECIMAL(10, 2) NOT NULL,
payment_date DATETIME NOT NULL,
PRIMARY KEY (payment_id),
```

```
FOREIGN KEY (order_id) REFERENCES Order(order_id),
FOREIGN KEY (payment_method_id) REFERENCES PaymentMethod(payment_method_id)
);

CREATE TABLE PaymentHistory (
payment_history_id INT AUTO_INCREMENT,
payment_id INT NOT NULL,
payment_date DATETIME NOT NULL,
payment_amount DECIMAL(10, 2) NOT NULL,
payment_method_name VARCHAR(255) NOT NULL,
payment_gateway VARCHAR(255) NOT NULL,
change_description VARCHAR(255),
payment_status VARCHAR(50) NOT NULL,
transaction_reference VARCHAR(50) NOT NULL,
PRIMARY KEY (payment_history_id),
FOREIGN KEY (payment_id) REFERENCES Payment(payment_id)
);

CREATE TABLE Notifications (
notification_id INT AUTO_INCREMENT,
customer_id INT NOT NULL,
created_date DATETIME NOT NULL,
notification_type VARCHAR(255) NOT NULL,
message TEXT NOT NULL,
PRIMARY KEY (notification_id),
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE LoyaltyProgram (
loyalty_program_id INT AUTO_INCREMENT,
customer_id INT NOT NULL,
points_earned INT NOT NULL DEFAULT 0,
points_redeemed INT NOT NULL DEFAULT 0,
membership_level VARCHAR(50) NOT NULL,
enrollment_date DATETIME NOT NULL,
PRIMARY KEY (loyalty_program_id),
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);
```

```
CREATE TABLE Promotion (
promotion_id INT AUTO_INCREMENT,
promotion_type VARCHAR(50) NOT NULL,
description TEXT NOT NULL,
discount_amount DECIMAL(10, 2) NOT NULL,
start_date DATETIME NOT NULL,
end_date DATETIME NOT NULL,
is_active BOOLEAN NOT NULL DEFAULT TRUE,
PRIMARY KEY (promotion_id)
);

CREATE TABLE ProductPromotion (
product_promotion_id INT AUTO_INCREMENT,
product_id INT NOT NULL,
promotion_id INT NOT NULL,
PRIMARY KEY (product_promotion_id),
FOREIGN KEY (product_id) REFERENCES Product(product_id),
FOREIGN KEY (promotion_id) REFERENCES Promotion(promotion_id)
);
```

Our team has integrated Jira with GitHub to streamline code storage and enhance collaboration. We can now track commits directly from Jira issues, making the workflow smoother and more efficient.

Link Github: https://github.com/quynezz/COS20031

Team Health Check

Team name	**QLTK - COS20031**
Sponsor	Swinburne University of Technology
Health monitor cadence	**Monthly check-in:** This method focusing on workload balance, collaboration, and any immediate challenges. The goal is to identify small issues and improve team efficiency. **Quarterly review:** This is a more thorough examination of your team's dynamics, project alignment, and individual improvement. It addresses and highlights strategies to improve teamwork, productivity, and professional growth. **Annual review:** This is an opportunity to discuss the team's yearly achievements, challenges, and alignment with the corporate aims. A discussion workshop and in-depth feedback sessions are part of the process to design a roadmap for the following year, emphasizing long-term development and improvements.

👨‍👨‍👦 Team health assessment

With your team, read the definition of each attribute of healthy, high-performing teams out loud. On the count of three have each person rate how they feel the team is doing compared to each definition (thumbs-up/green, thumbs-sideways/yellow, thumbs-down/red). Record the results of each attribute rating in the table. Highlight each cell using this color code: HEALTHY = "We're strong here", BIT SICK = "We're ok... but a little shaky", SICK = "We're not healthy".

🚩 Full-time owner

There is **one lead who is accountable** for the result of this project. This needs to be someone whose time is at least 80% dedicated to it, and who can champion the mission inside and outside of the team.

Sep 9, 2024	**Thuan Khang**, the project's full-time proprietor, is currently unable to dedicate 80% of his time due to personal concerns. We are actively working to allocate tasks and resources to compensate for his reduced availability. While we are concerned about this situation, we believe there are opportunities for improvement as we move forward.
Sep 15-16, 2024	**Thuan Khang** has not yet completed his personal commitments, which limits his ability to fully focus on the project. While he cannot dedicate 80% of his time to this endeavor, I have confidence in his past management abilities and the team's automation skills. We believe that, despite these challenges, our project has strong potential for success.
Sep 23, 2024	He has willingly extended his working hours to ensure he can commit at least 80% of his time to this initiative, promoting its growth and success. **Thuan Khang's** passion and focus on the project have contributed to a strong and productive working environment.
Sep 30, 2024	**Thuan Khang** has made this project his top priority and has developed a work schedule that enables him to dedicate at least 80% of his time to it. The project's progress has been greatly accelerated thanks to **Thuan Khang's** excellent time management skills and clear vision.

Oct 9, 2024	**Thuan Khang** has taken significant steps to ensure effective time and resource management. Each time he dedicates more than 80% of his time to the project, noticeable improvements are made.
Oct 16, 2024	During this time, **Thuan Khang** ran into several project management issues. We were a little concerned at this point because of a few issues that were impeding the project's progress.

⚖️ Balanced team

Roles and responsibilities are clear and agreed upon. The project has people with the right blend of skill set. Acknowledge that team members can change by stage.

Sep 9, 2024	In the early stages, we assigned tasks to each team member appropriately. However, we were caught off guard when unforeseen issues arose, leading to several complications that negatively impacted our project.
Sep 15-16, 2024	We already understand how to organize and assign responsibilities efficiently. However, as we continue to acclimate to our new roles, it takes time for us to adapt and identify who should take charge when unusual tasks arise. This can extend the time required to complete tasks and potentially lead to complications.
Sep 23, 2024	We are accustomed to taking breaks from our duties, which is why we have policies and plans in place to address any unusual situations that may arise. By doing this, we have been able to ensure consistent project progress and productivity.
Sep 30, 2024	We have combined our knowledge and skills to enable everyone to complete the project efficiently and as soon as it is practical for them to do so.
Oct 9, 2024	We have established clear guidelines regarding responsibilities and have implemented plans and processes to address any unusual situations that may arise. We found this approach to be highly beneficial while executing our project.

Oct 16, 2024	Team members have a solid understanding of their assigned tasks, and we have divided responsibilities in a clear and systematic manner. The project progresses on schedule due to the team's consensus and mutual willingness to support one another. Our readiness to respond and find quick solutions makes it easier for us to manage unforeseen challenges.

💞 Shared understanding

The team has a **common understanding of why they're here**, the problem/need, are convinced about the idea, confident they have what they need, and trust each other.

Sep 9, 2024	We have established clear guidelines regarding responsibilities and have implemented plans and processes to address any unusual situations that may arise. We found this approach to be highly beneficial while executing our project.
Sep 15-16, 2024	To promote a spirit of change, it is important to enhance team members' trust and support for one another. Mutual trust and understanding are essential for building high collective effectiveness.
Sep 23, 2024	The team leader creates project diagrams to communicate objectives and tasks, allowing everyone to see how the components fit together.
Sep 30, 2024	Invite team members to propose recommendations for project fixes or adjust strategies as needed. The team feels more confident in responding to changes when everyone is informed about the objectives.
Oct 9, 2024	Hold regular meetings to ensure the team stays focused on the decisions and goals of the project.
Oct 16, 2024	Invite team members to propose recommendations for project fixes or adjust strategies as needed. The team feels more confident in responding to changes when everyone is informed about the objectives.

💍 Value and metrics

It's clear **what success means** from a business and user's perspective, and there is a unique value proposition in place for the target users and to the business. Success is defined, with a goal, and how it will be measured.

Sep 9, 2024	To ensure project success, establish specific targets. The development and construction of goals should be guided by user and business requirements.
Sep 15-16, 2024	Identify precise metrics to measure success, which may encompass revenue, conversion rates, user counts, or other indicators pertinent to the project.
Sep 23, 2024	We create and cultivate distinctive values for both businesses and users. This project has made significant achievements in the B2B2C sector, drawing attention and enticing users.
Sep 30, 2024	We compile feedback and reviews from businesses and clients to expand and enhance the project. This approach includes leveraging social networks, reviews, and direct customer surveys. Additionally, associations contribute opinion pieces or engage in discussions to share their perspectives.
Oct 9, 2024	We effectively and transparently communicate the values of our goods and services to our clientele, utilizing social media and marketing campaigns to achieve this.
Oct 16, 2024	We conduct assessments to monitor the project's progress and make necessary adjustments, ensuring that everything stays on track and meets objectives.

🥽 Proof of concept

Some sort of demonstration has been created and tested, that demonstrates why this problem needs to be solved, and demonstrates its value.

Sep 9, 2024	To save time and money, a proof of concept must be produced early in the development process.
Sep 15-16, 2024	A cross-functional team must develop the proof of concept to ensure it satisfies the requirements of all parties involved.
Sep 23, 2024	To ensure the proof of concept functions well and meets all requirements, our team needs to complete a thorough testing procedure.
Sep 30, 2024	Stakeholders must review the proof of concept to provide their input.
Oct 9, 2024	A procedure must be established for regularly reviewing the proof of concept to accommodate any modifications to the solution.
Oct 16, 2024	"The proof of concept is used to create a detailed development plan.

One-pager

The **project is summarized** in a one-pager and shared with anyone so that they understand the purpose of the project, and its value.

Sep 9, 2024	Our group must establish a consistent procedure for updating the one-pager, such as revising it before every stakeholder meeting and after any significant project modification.
Sep 15-16, 2024	We must be mindful of one-pager presentations by employing clear, concise language, using illustrative graphics and charts, and dividing content with headings and subheadings.
Sep 23, 2024	Sharing a one-pager with the appropriate audience is essential.

Sep 30, 2024	To effectively convey information about the project, one-pagers must be used. Consequently, our team should employ one-pagers to spread knowledge about the project at meetings, conferences, and other gatherings.
Oct 9, 2024	One-pagers need to be reviewed periodically to determine whether updates are necessary to reflect shifts in the team's perspective on the project.
Oct 16, 2024	"One-pagers must be used to monitor our progress toward the project's objectives.

🤰 Managed dependencies

Clear understanding of complexity, infrastructure involved, risks, resources, effort, and timeline. Clear understanding of **who we depend on, and who depends on us.**

Sep 9, 2024	For managing dependencies, our team needs a well-defined procedure tailored to the project's size and type. This process must clearly identify the stages to be followed, the parties involved, and each party's responsibilities.
Sep 15-16, 2024	Technology and tools are essential for facilitating dependency management. Therefore, to support dependency management, the team must acquire and apply the necessary tools and technologies. These tools will enable our team to monitor the status of interdependent tasks, receive updates, and identify potential issues.
Sep 23, 2024	Risks associated with dependencies must be recognized and assessed. Steps must be taken to mitigate these risks, such as diversifying sources of supply, creating backup plans, and enhancing adaptability.
Sep 30, 2024	The team must develop a strategy for testing and assessing dependency management. This strategy should specify evaluation criteria and frequency. With this support, the team will ensure effective dependency management.

Oct 9, 2024	Handling dependencies is crucial for teams. Team members need training and development in dependency management. Therefore, the group must create a curriculum covering fundamentals, tools and techniques, and best practices.
Oct 16, 2024	Dependency management is an essential component of project management. It must be integrated into the project management process. Identifying project dependencies and creating a suitable management strategy should be incorporated into the team's project management procedure.

🏇 Velocity

The team is making **incremental progress** by shipping concrete iterations to stakeholders (and, even better, to production), learning along the way, and **implementing lessons learned**, resulting in greater success.

Sep 9, 2024	Although the team is growing rapidly, there is definitely room for development. To ensure steady and sustainable growth, the team requires a comprehensive and extended development plan.
Sep 15-16, 2024	For assessing development speed, our team needs precise measurements. This will allow us to track and enhance progress more effectively. Specific metrics, such as the number of features released and the time it takes to complete projects, must be chosen to assess development speed.
Sep 23, 2024	Close collaboration between team members is essential for accelerated progress. Therefore, the team needs to engage in activities such as regular meetings, knowledgesharing sessions, and training sessions to improve coordination.
Sep 30, 2024	We must seek assistance from other sources, including communities, professionals, and consulting firms. This will facilitate the team's rapid development.
Oct 9, 2024	The team must routinely assess elements such as member knowledge and expertise, the technology and tools used, and the working environment. Strategies to enhance these areas should then be devised.

Oct 16, 2024	The development of the team and the quality of the product or service must be balanced. The team must have a realistic plan to ensure both speed and quality in the development of the product or service.

🎯 Focus areas

Ask your team to collectively come up with one attribute you want to focus on. Then, call out ways to move the **SICK** or **BIT SICK** toward **HEALTHY** . Make sure they are actionable, specific, and measurable.

Date	Focus areas and action items
Sep 10,2024	☒ Establish limitations and evaluate the current scalability of the system.
Sep 10,2024	☒ Assign manageable responsibilities to team members so they can collaborate and solve issues when the full-time owner is unavailable.
Sep 19,2024	☒ Create backup plans in advance to swiftly and efficiently address unexpected problems
Sep 25,2024	☒ Focus on implementation and prioritize activities to avoid atypical tasks from impacting the project.
Oct 10,2024	☒ Conduct vulnerability analyses and security tests on the system.
Oct 18,2024	☒ Use performance monitoring tools for routine system performance analysis.

🌹 Next steps

1. Important steps in the process of constructing an information system include creating databases with MySQL, creating ER diagrams, database normalization, and installing and configuring databases.
2. The data requirements of the system are defined using ER diagrams. Entities, properties, and relationships between entities in a system can all be described using ER diagrams.
3. The ER diagram serves as the basis for the database diagram. The database's structure is shown in the database diagram, which includes primary keys, foreign keys, and tables.
4. The practice of structuring data in a database to reduce redundancy and enhance integrity is known as database normalization. It involves five stages, from level 1 to level 5.
5. The final step in developing a database is installing MySQL and building the database. MySQL, a free relational database management system (RDBMS), is widely used in web and mobile applications.