BIL372 Database Systems - Final Report

Physical Game and Console Sales Website

Project Team Members

Name	Student ID
Muhammed Yusuf Kartal	231401012
Yağız Can Akay	231401013
Ali Türkücü	231401014

Table of Contents

Github repository link: https://github.com/eyay-org/gameD0NTst0p/tree/main
You can find the reports (interim and final report), screenshots and diagrams used in the final report under reports/ folder.

1. Real-World Problem Definition

1.1 Problem Statement

This project aims to develop a comprehensive B2C (Business-to-Consumer) e-commerce platform for selling physical video games and gaming consoles. Today, the gaming industry is rapidly growing, and consumers demand access to both new and classic games/consoles.

1.2 Solution Approach

The developed system offers the following features:

- **Product Catalog Management**: Detailed information storage for games and consoles
- **Customer Interaction**: Registration, login, cart management, order tracking
- **Multi-Branch Support**: Inventory management across different physical stores
- **Supplier Relations**: Product procurement and purchasing processes
- **After-Sales Services**: Return and review systems
- **Admin Panel**: Inventory, order, analytics, and branch management

1.3 Target Users

User Type	Description
Customers	End users who purchase games and consoles
Administrators (Admin)	System management, inventory control, order processing
Branch Managers	Physical store operations

2. Requirements Analysis

2.1 Functional Requirements

Customer Functions

#	Function	Description
F1	Product Search	Filtering by platform, genre, price range, ESRB rating
F2	Cart Management	Add, remove products, update quantity
F3	Order Creation	Place orders with payment and delivery information
F4	Order Tracking	View order status and tracking number
F5	Write Reviews	Add 1-5 star ratings and comments for purchased products
F6	Profile Management	Add/delete addresses, change password
F7	Return Request	Submit return applications for delivered orders

Admin Functions

#	Function	Description
A1	Dashboard	Total sales, order count, low stock alerts
A2	Inventory Management	View and update stock levels
A3	Order Management	Update order statuses (pending, shipped, delivered)
A4	Stock Transfer	Product transfer between branches
A5	Restock	Purchase products from suppliers
A6	In-Store Sales	Record sales at physical stores
A7	Return Processing	Approve/reject return requests
A8	Analytics	Revenue, profit, branch performance reports

2.2 Non-Functional Requirements

Requirement	Description
Security	Password hashing (SHA256), session management
Performance	Fast queries with database indexes
Scalability	Multi-branch support
Usability	Modern and responsive web interface
Data Integrity	Foreign key, CHECK, UNIQUE constraints

2.3 Business Rules

9. **Price Control**: Product price must be greater than 0 (`CHECK (price > 0)`)
10. **Stock Control**: Inventory quantity cannot be negative (`CHECK (quantity >= 0)`)
11. **Review Rating**: Rating must be between 1-5 (`CHECK (rating >= 1 AND rating <= 5)`)
12. **Unique Email**: Each customer must have a unique email address
13. **Product Type**: Products can only be ‘game’ or ‘console’ type
14. **Sale Type**: Sales can only be ‘online’ or ‘in-store’ type

3. Conceptual Design (EER Diagrams)

3.1 Entities

The system contains a total of **19 entities**:

Base Entities

Entity	Description	Key
CUSTOMER	Customer information	customer_id (PK)
PRODUCT	Product superclass	product_id (PK)
SUPPLIER	Supplier information	supplier_id (PK)
GENRE	Game genres	genre_id (PK)
BRANCH	Branch information	branch_id (PK)

Subclasses

Entity	Superclass	Description
GAME	PRODUCT	Game details (platform, developer, ESRB)
CONSOLE	PRODUCT	Console details (manufacturer, storage)

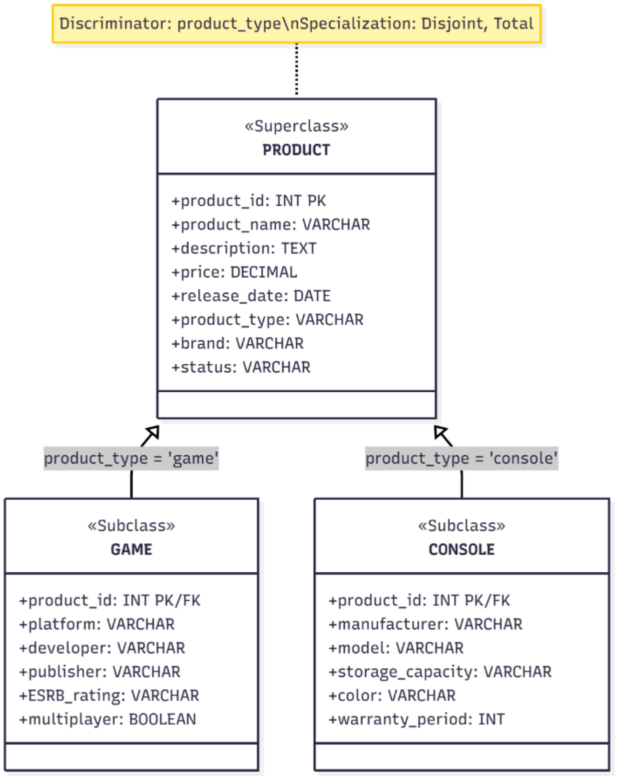
Transaction Entities

Entity	Description
ORDER	Customer orders
ORDER_DETAIL	Order line items (Weak Entity)
SALE	Sales records
PURCHASE	Supplier purchases
RETURN	Return transactions

Supporting Entities

Entity	Description
INVENTORY	Branch-based stock information
ADDRESS	Customer addresses
REVIEW	Product reviews
CART	Shopping cart
PRODUCT_MEDIA	Product images/videos
GAME_GENRE	Game-Genre relationship (Associative)
STOCK_LOG	Stock change logs

3.2 Superclass-Subclass Relationship



Properties:

- **Specialization Type**: Disjoint - A product can be either a game or a console
- **Completeness**: Total - Every product must belong to a subclass
- **Discriminator**: `product_type` field (‘game’ or ‘console’)

3.3 Weak Entity

ORDER_DETAIL is a **Weak Entity** dependent on the ORDER entity:

- **Composite Key**: (order_id, line_no)
- **Identifying Relationship**: ORDER → ORDER_DETAIL
- When an order is deleted, related details are also deleted (CASCADE)

3.4 Relationships

One-to-Many (1:N) Relationships

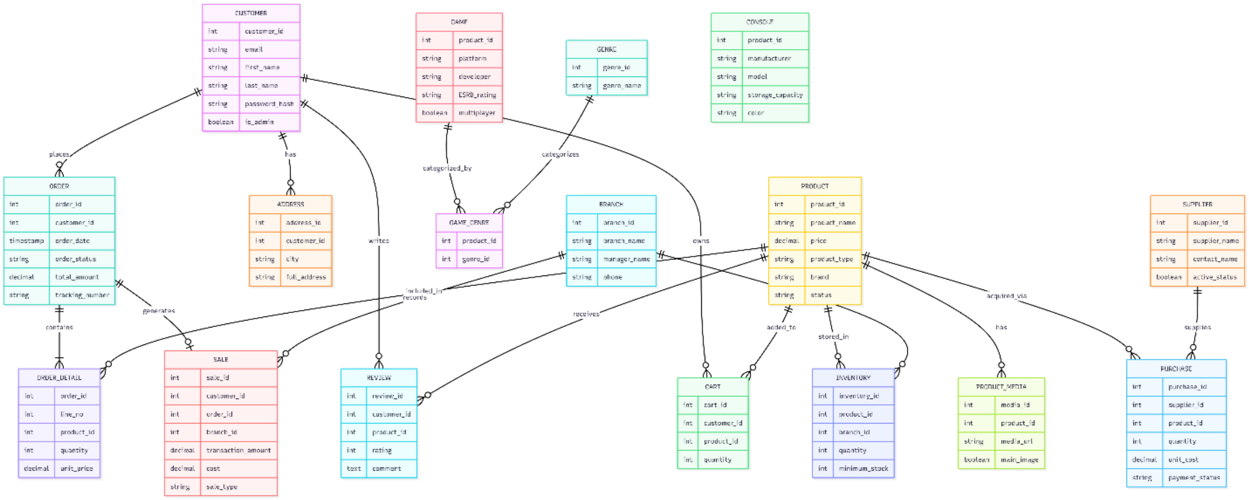
Relationship	Description

Relationship	Description
CUSTOMER → ORDER	A customer can place multiple orders
CUSTOMER → ADDRESS	A customer can have multiple addresses
CUSTOMER → REVIEW	A customer can write multiple reviews
PRODUCT → REVIEW	A product can receive multiple reviews
PRODUCT → INVENTORY	A product can exist in multiple branches
BRANCH → INVENTORY	A branch can stock multiple products
ORDER → ORDER_DETAIL	An order can contain multiple line items
SUPPLIER → PURCHASE	Multiple purchases can be made from a supplier

Many-to-Many (M:N) Relationships

Relationship	Associative Entity	Description
GAME ↔ GENRE	GAME_GENRE	A game can belong to multiple genres
CUSTOMER ↔ PRODUCT	CART	A customer can add multiple products to cart

3.5 EER Diagram



4. Logical Design and Schema Diagrams

4.1 EER to Relational Model Transformation

Applied Transformation Rules

EER Concept	Applied Transformation
Superclass/Subclass	Separate table for each, subclass PK = FK (PRODUCT →

EER Concept	Applied Transformation
	GAME, CONSOLE)
Weak Entity	Contains composite PK (ORDER_DETAIL)
M:N Relationship	Associative entity (GAME_GENRE, CART)
Composite Attribute	Separate columns (delivery_full_address, delivery_city)
Multivalued Attribute	Separate table (PRODUCT_MEDIA)

4.2 Table Schemas

CUSTOMER Table

```
CUSTOMER (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    phone VARCHAR(20),
    registration_date DATE,
    last_login_date TIMESTAMP,
    active_status BOOLEAN DEFAULT TRUE,
    is_admin BOOLEAN DEFAULT FALSE
)
```

PRODUCT Table (Superclass)

```
PRODUCT (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    product_name VARCHAR(200) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL CHECK (price > 0),
    release_date DATE,
    product_type VARCHAR(20) CHECK (product_type IN ('game', 'console')),
    brand VARCHAR(100),
    status VARCHAR(20),
    weight DECIMAL(6,2),
    dimensions VARCHAR(50),
    stock_alert_level INT DEFAULT 10
)
```

GAME Table (Subclass)

```
GAME (
    product_id INT PRIMARY KEY REFERENCES PRODUCT(product_id) ON DELETE
CASCADE,
    platform VARCHAR(255),
    developer VARCHAR(100),
    publisher VARCHAR(100),
    ESRB_rating VARCHAR(10),
    multiplayer BOOLEAN,
    language_support TEXT,
    subtitle_languages TEXT
)
```

CONSOLE Table (Subclass)

```
CONSOLE (
    product_id INT PRIMARY KEY REFERENCES PRODUCT(product_id) ON DELETE CASCADE,
    manufacturer VARCHAR(100),
    model VARCHAR(100),
    storage_capacity VARCHAR(20),
    color VARCHAR(30),
    included_accessories TEXT,
    warranty_period INT
)
```

ORDER Table

```
ORDER (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT REFERENCES CUSTOMER(customer_id) ON DELETE SET NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    order_status VARCHAR(20),
    total_amount DECIMAL(10,2),
    shipping_fee DECIMAL(6,2),
    payment_method VARCHAR(30),
    payment_status VARCHAR(20),
    tracking_number VARCHAR(50),
    estimated_delivery_date DATE,
    actual_delivery_date DATE,
    delivery_full_address TEXT,
    delivery_city VARCHAR(100),
    billing_full_address TEXT,
    billing_city VARCHAR(100)
)
```

ORDER_DETAIL Table (Weak Entity)

```
ORDER_DETAIL (
    order_id INT REFERENCES ORDER(order_id) ON DELETE CASCADE,
    line_no INT,
    product_id INT REFERENCES PRODUCT(product_id) ON DELETE SET NULL,
    quantity INT CHECK (quantity > 0),
    unit_price DECIMAL(10,2),
    PRIMARY KEY (order_id, line_no)
)
```

INVENTORY Table

```
INVENTORY (
    inventory_id INT PRIMARY KEY AUTO_INCREMENT,
    product_id INT NOT NULL REFERENCES PRODUCT(product_id) ON DELETE CASCADE,
    branch_id INT NOT NULL REFERENCES BRANCH(branch_id) ON DELETE CASCADE,
    quantity INT NOT NULL CHECK (quantity >= 0),
    minimum_stock INT DEFAULT 10,
    maximum_stock INT DEFAULT 100,
    shelf_location VARCHAR(50),
    last_update_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    UNIQUE (product_id, branch_id)
)
```

PURCHASE Table (BCNF Compliant)

```
PURCHASE (
    purchase_id INT PRIMARY KEY AUTO_INCREMENT,
    supplier_id INT REFERENCES SUPPLIER(supplier_id) ON DELETE SET NULL,
    product_id INT REFERENCES PRODUCT(product_id) ON DELETE SET NULL,
    transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    quantity INT CHECK (quantity > 0),
    unit_cost DECIMAL(10,2) CHECK (unit_cost > 0),
    payment_status VARCHAR(20),
    payment_date DATE,
    invoice_no VARCHAR(50)
)
-- NOTE: total_cost = quantity × unit_cost, calculated via VIEW_PURCHASE_WITH_TOTAL
```

SALE Table (BCNF Compliant)

```
SALE (
    sale_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT REFERENCES CUSTOMER(customer_id) ON DELETE SET NULL,
    order_id INT REFERENCES ORDER(order_id) ON DELETE SET NULL,
    branch_id INT REFERENCES BRANCH(branch_id) ON DELETE SET NULL,
    transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    transaction_amount DECIMAL(10,2),
    cost DECIMAL(10,2),
    sale_type VARCHAR(20) CHECK (sale_type IN ('online', 'in-store')),
    sale_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
-- NOTE: profit = transaction_amount - cost, calculated via
VIEW_SALE_WITH_PROFIT
```

4.3 Normalization Analysis

All 19 tables comply with 1NF, 2NF, 3NF, and BCNF.

1NF (First Normal Form) ✓

- Atomic values are used in all tables
- Repeating groups have been eliminated (e.g., GAME_GENRE table for genres)

2NF (Second Normal Form) ✓

- Full functional dependency is ensured in all tables
- In ORDER_DETAIL table: (order_id, line_no) → {product_id, quantity, unit_price}

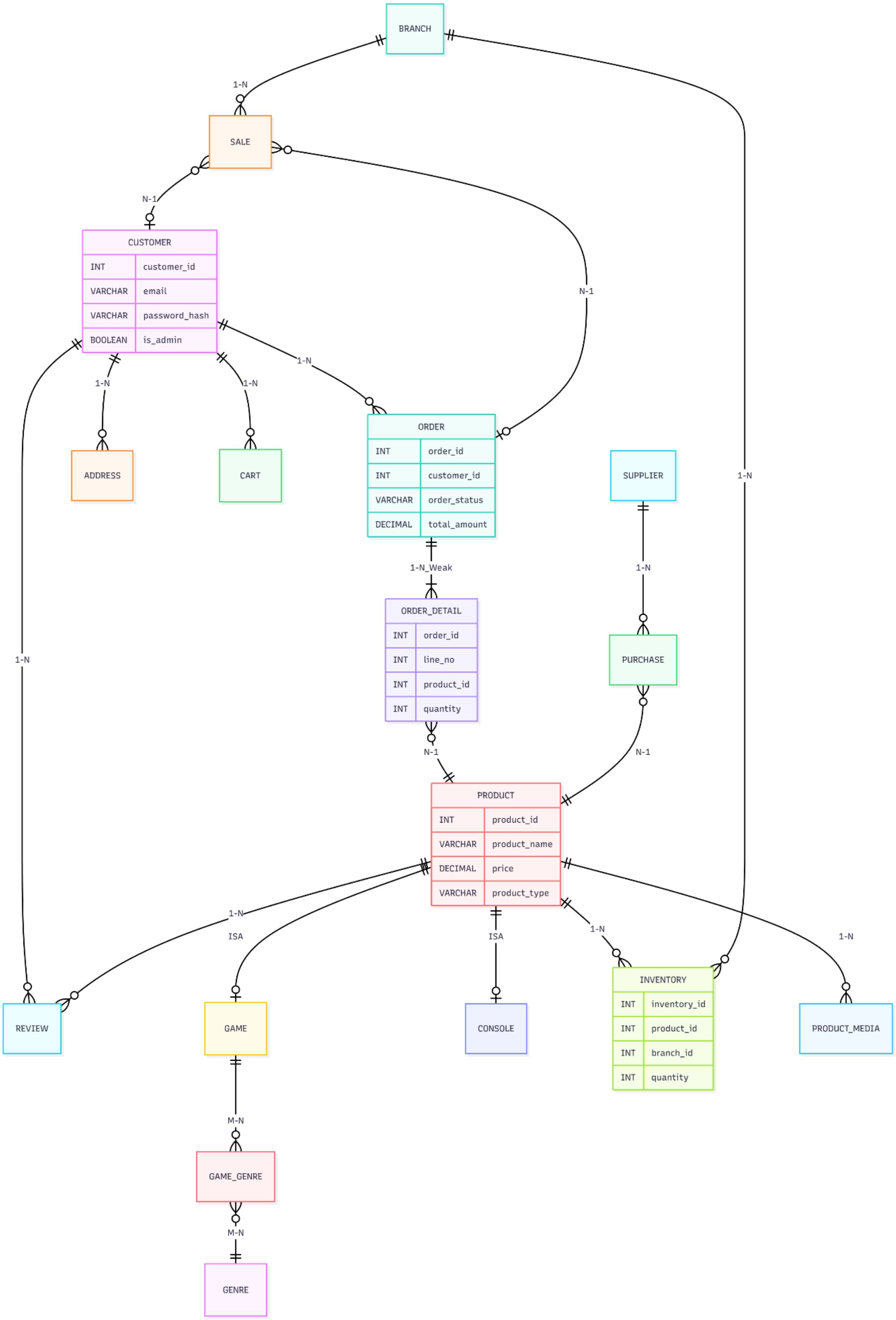
3NF (Third Normal Form) ✓

- Transitive dependencies have been eliminated
- Derived columns (total_cost, profit) in PURCHASE and SALE tables have been removed and are calculated via VIEWs

BCNF (Boyce-Codd Normal Form) ✓

- In every functional dependency, the determinant is a candidate key
- Derived columns (total_cost, profit) were removed to prevent BCNF violation
- These values are calculated through VIEWs:
  - `VIEW_PURCHASE_WITH_TOTAL`: total_cost = quantity × unit_cost
  - `VIEW_SALE_WITH_PROFIT`: profit = transaction_amount - cost
- All 19 tables are in BCNF

4.4 Relational Schema Diagram



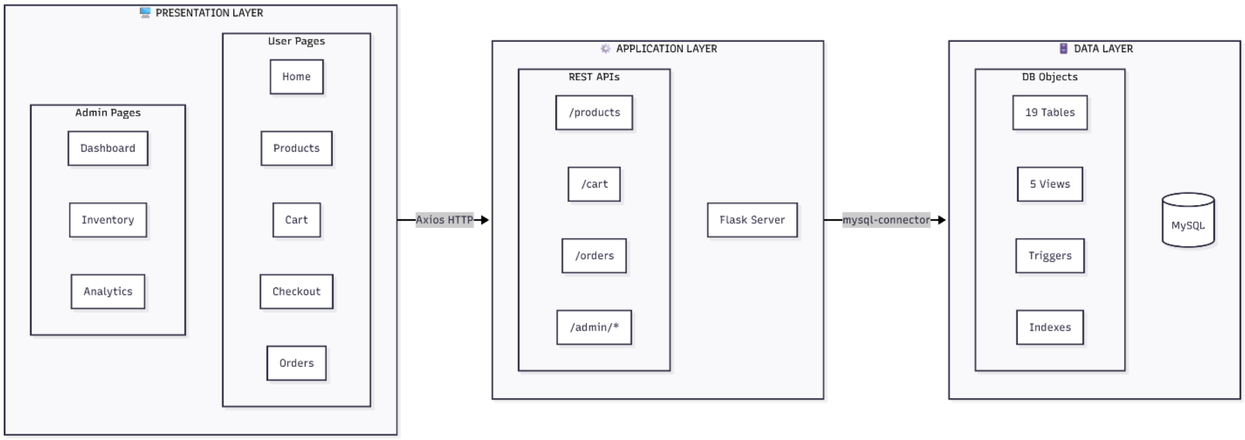
5. Design Implementation

5.1 Software/Hardware Environment and DBMS Information

Selected Technologies

Layer	Technology	Version	Description
Database (DBMS)	MySQL	8.0+	Relational database management system
Backend	Python Flask	3.1.2	RESTful API server
Frontend	React.js	18.x	Single page web application
DB Connector	mysql-connector-python	9.5.0	Python MySQL connector (ODBC/JDBC alternative)
HTTP Client	Axios	-	Frontend-Backend communication

Architecture (3-Tier Architecture)



5.2 Table Creation

All tables are defined in the `database/dbsetup.sql` file. Below are the main DDL commands:

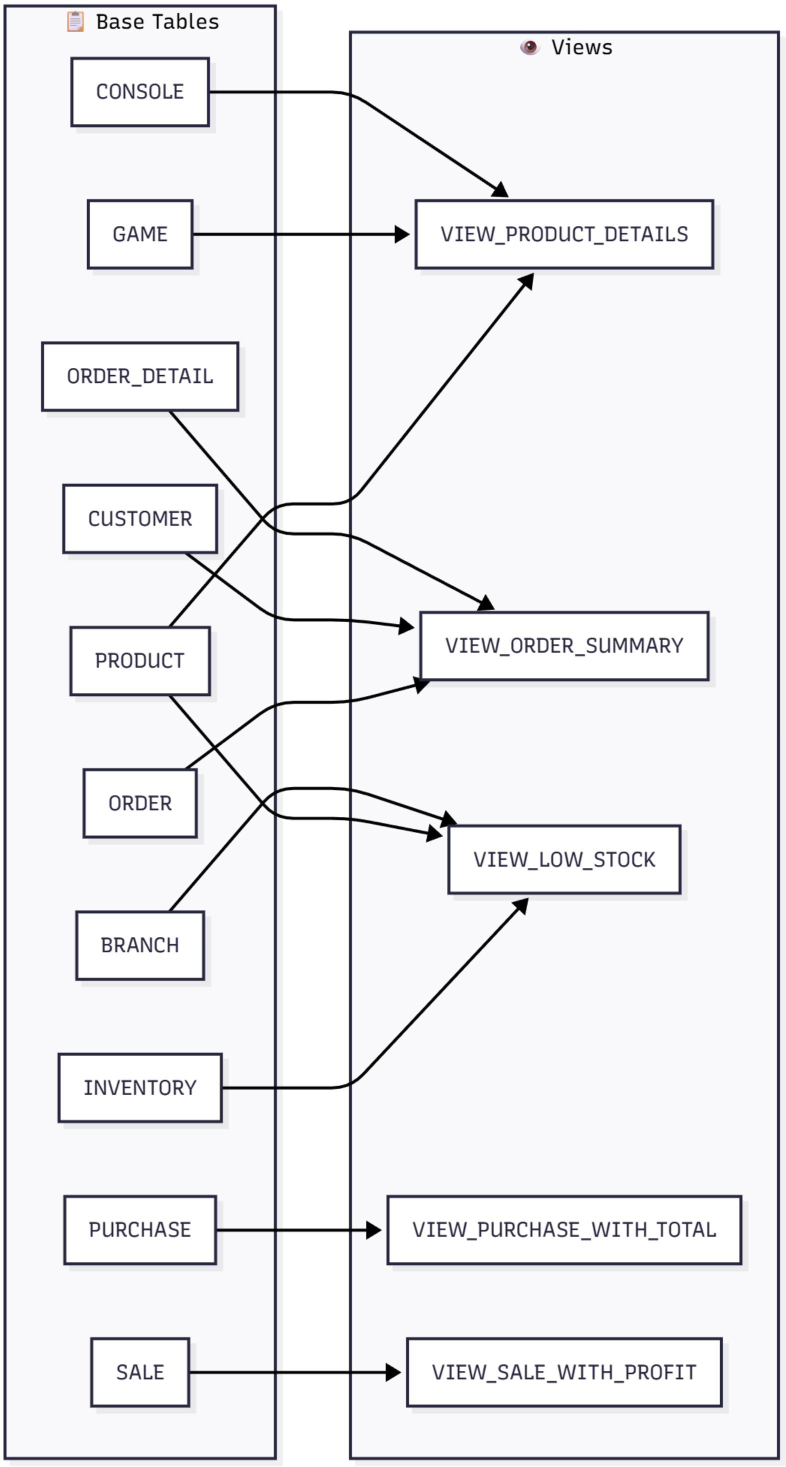
```sql
-- Database Creation
CREATE DATABASE IF NOT EXISTS oyun_satis_db DEFAULT CHARACTER SET 'utf8mb4';
USE oyun_satis_db;
```

```
-- CUSTOMER Table
CREATE TABLE IF NOT EXISTS `CUSTOMER` (
  `customer_id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(50) NOT NULL,
  `last_name` VARCHAR(50) NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  `password_hash` VARCHAR(255) NOT NULL,
  `phone` VARCHAR(20),
  `registration_date` DATE,
  `last_login_date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `active_status` BOOLEAN DEFAULT TRUE,
  `is_admin` BOOLEAN DEFAULT FALSE,
  PRIMARY KEY (`customer_id`),
  UNIQUE KEY `uk_email` (`email`)
);

-- PRODUCT Table (Superclass)
CREATE TABLE IF NOT EXISTS `PRODUCT` (
  `product_id` INT NOT NULL AUTO_INCREMENT,
  `product_name` VARCHAR(200) NOT NULL,
  `description` TEXT,
  `price` DECIMAL(10, 2) NOT NULL,
  `release_date` DATE,
  `product_type` VARCHAR(20),
  `brand` VARCHAR(100),
  `status` VARCHAR(20),
  `weight` DECIMAL(6, 2),
  `dimensions` VARCHAR(50),
  `stock_alert_level` INT DEFAULT 10,
  PRIMARY KEY (`product_id`),
  CONSTRAINT `chk_price` CHECK (`price` > 0),
  CONSTRAINT `chk_product_type` CHECK (`product_type` IN ('game', 'console'))
);

-- GAME Table (Subclass)
CREATE TABLE IF NOT EXISTS `GAME` (
  `product_id` INT NOT NULL,
  `platform` VARCHAR(255),
  `developer` VARCHAR(100),
  `publisher` VARCHAR(100),
  `ESRB_rating` VARCHAR(10),
  `multiplayer` BOOLEAN,
  `language_support` TEXT,
  `subtitle_languages` TEXT,
  PRIMARY KEY (`product_id`),
  CONSTRAINT `fk_game_product`
    FOREIGN KEY (`product_id`) REFERENCES `PRODUCT` (`product_id`)
    ON DELETE CASCADE
);
```

```sql
-- CONSOLE Table (Subclass)
CREATE TABLE IF NOT EXISTS `CONSOLE` (
  `product_id` INT NOT NULL,
  `manufacturer` VARCHAR(100),
  `model` VARCHAR(100),
  `storage_capacity` VARCHAR(20),
  `color` VARCHAR(30),
  `included_accessories` TEXT,
  `warranty_period` INT,
  PRIMARY KEY (`product_id`),
  CONSTRAINT `fk_console_product`
    FOREIGN KEY (`product_id`) REFERENCES `PRODUCT` (`product_id`)
    ON DELETE CASCADE
);

-- ORDER Table
CREATE TABLE IF NOT EXISTS `ORDER` (
  `order_id` INT NOT NULL AUTO_INCREMENT,
  `customer_id` INT,
  `order_date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  `order_status` VARCHAR(20),
  `total_amount` DECIMAL(10, 2),
  `shipping_fee` DECIMAL(6, 2),
  `payment_method` VARCHAR(30),
  `payment_status` VARCHAR(20),
  `tracking_number` VARCHAR(50),
  `estimated_delivery_date` DATE,
  `actual_delivery_date` DATE,
  `delivery_full_address` TEXT,
  `delivery_city` VARCHAR(100),
  `billing_full_address` TEXT,
  `billing_city` VARCHAR(100),
  PRIMARY KEY (`order_id`),
  CONSTRAINT `fk_order_customer`
    FOREIGN KEY (`customer_id`) REFERENCES `CUSTOMER` (`customer_id`)
    ON DELETE SET NULL
);
```

```sql
-- ORDER_DETAIL Table (Weak Entity)
CREATE TABLE IF NOT EXISTS `ORDER_DETAIL` (
  `order_id` INT NOT NULL,
  `line_no` INT NOT NULL,
  `product_id` INT,
  `quantity` INT,
  `unit_price` DECIMAL(10, 2),
  PRIMARY KEY (`order_id`, `line_no`),
  CONSTRAINT `fk_od_order`
    FOREIGN KEY (`order_id`) REFERENCES `ORDER` (`order_id`)
    ON DELETE CASCADE,
  CONSTRAINT `fk_od_product`
    FOREIGN KEY (`product_id`) REFERENCES `PRODUCT` (`product_id`)
    ON DELETE SET NULL,
  CONSTRAINT `chk_od_quantity` CHECK (`quantity` > 0)
);

-- INVENTORY Table
CREATE TABLE IF NOT EXISTS `INVENTORY` (
  `inventory_id` INT NOT NULL AUTO_INCREMENT,
  `product_id` INT NOT NULL,
  `branch_id` INT NOT NULL,
  `quantity` INT NOT NULL,
  `minimum_stock` INT DEFAULT 10,
  `maximum_stock` INT DEFAULT 100,
  `shelf_location` VARCHAR(50),
  `last_update_date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (`inventory_id`),
  UNIQUE KEY `uk_product_branch` (`product_id`, `branch_id`),
  CONSTRAINT `fk_inv_product`
    FOREIGN KEY (`product_id`) REFERENCES `PRODUCT` (`product_id`)
    ON DELETE CASCADE,
  CONSTRAINT `fk_inv_branch`
    FOREIGN KEY (`branch_id`) REFERENCES `BRANCH` (`branch_id`)
    ON DELETE CASCADE,
  CONSTRAINT `chk_quantity` CHECK (`quantity` >= 0)
);
```

5.3 Views



5 VIEWs are defined in the system (3 for reporting + 2 for BCNF compliance):

VIEW 1: VIEW_PRODUCT_DETAILS

Combines all product details (Game and Console):

```
CREATE OR REPLACE VIEW VIEW_PRODUCT_DETAILS AS
SELECT
    p.product_id,
    p.product_name,
    p.price,
    p.product_type,
    p.brand,
    p.status,
    g.platform,
    g.developer,
    g.ESRB_rating,
    c.manufacturer,
    c.storage_capacity,
    c.color
FROM PRODUCT p
LEFT JOIN GAME g ON p.product_id = g.product_id
LEFT JOIN CONSOLE c ON p.product_id = c.product_id;
```

VIEW 2: VIEW_ORDER_SUMMARY

Order summary for admin panel:

```
CREATE OR REPLACE VIEW VIEW_ORDER_SUMMARY AS
SELECT
    o.order_id,
    o.order_date,
    o.order_status,
    o.total_amount,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    c.email,
    COUNT(od.line_no) as item_count
FROM `ORDER` o
JOIN CUSTOMER c ON o.customer_id = c.customer_id
LEFT JOIN ORDER_DETAIL od ON o.order_id = od.order_id
GROUP BY o.order_id;
```

VIEW 3: VIEW_LOW_STOCK

Products at critical stock levels:

```
CREATE OR REPLACE VIEW VIEW_LOW_STOCK AS
SELECT
    i.inventory_id,
    p.product_name,
    b.branch_name,
    i.quantity,
    i.minimum_stock
FROM INVENTORY i
JOIN PRODUCT p ON i.product_id = p.product_id
JOIN BRANCH b ON i.branch_id = b.branch_id
WHERE i.quantity <= i.minimum_stock;
```

VIEW 4: VIEW_PURCHASE_WITH_TOTAL (BCNF Compliance)

Calculated total_cost column for PURCHASE table:

```
CREATE OR REPLACE VIEW VIEW_PURCHASE_WITH_TOTAL AS
SELECT
    purchase_id,
    supplier_id,
    product_id,
    transaction_date,
    quantity,
    unit_cost,
    (quantity * unit_cost) AS total_cost,
    payment_status,
    payment_date,
    invoice_no
FROM PURCHASE;
```

VIEW 5: VIEW_SALE_WITH_PROFIT (BCNF Compliance)

Calculated profit column for SALE table:

```
CREATE OR REPLACE VIEW VIEW_SALE_WITH_PROFIT AS
SELECT
    sale_id,
    customer_id,
    order_id,
    branch_id,
    transaction_date,
    transaction_amount,
    cost,
    (transaction_amount - cost) AS profit,
    sale_type,
    sale_date
FROM SALE;
```

5.4 Indexes

Indexes created for query performance:

```
-- For product searches
CREATE INDEX idx_product_name ON PRODUCT(product_name);
CREATE INDEX idx_product_price ON PRODUCT(price);
CREATE INDEX idx_product_type ON PRODUCT(product_type);

-- For order queries
CREATE INDEX idx_order_date ON `ORDER`(order_date);
CREATE INDEX idx_order_customer ON `ORDER`(customer_id);

-- For game filtering
CREATE INDEX idx_game_rating ON GAME(ESRB_rating);
```

5.5 Triggers

Stock Change Log Trigger

```
-- STOCK_LOG Table
CREATE TABLE IF NOT EXISTS `STOCK_LOG` (
  `log_id` INT NOT NULL AUTO_INCREMENT,
  `product_id` INT NOT NULL,
  `branch_id` INT NOT NULL,
  `old_quantity` INT,
  `new_quantity` INT,
  `change_date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`log_id`),
  CONSTRAINT `fk_log_product`
    FOREIGN KEY (`product_id`) REFERENCES `PRODUCT` (`product_id`)
    ON DELETE CASCADE,
  CONSTRAINT `fk_log_branch`
    FOREIGN KEY (`branch_id`) REFERENCES `BRANCH` (`branch_id`)
    ON DELETE CASCADE
);

-- Trigger: Automatic logging on stock changes
DELIMITER //
CREATE TRIGGER after_inventory_update
AFTER UPDATE ON INVENTORY
FOR EACH ROW
BEGIN
    IF OLD.quantity != NEW.quantity THEN
        INSERT INTO STOCK_LOG (product_id, branch_id, old_quantity,
new_quantity, change_date)
        VALUES (NEW.product_id, NEW.branch_id, OLD.quantity, NEW.quantity,
NOW());
    END IF;
END//
DELIMITER ;
```

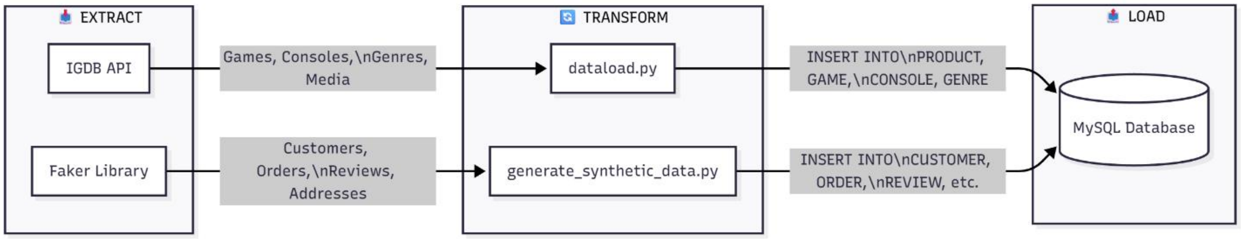
5.6 Data Loading

Data Sources

Data Type	Source	Description
Games	IGDB API	Real game data (name, description, platform, images)
Consoles	IGDB API + Manual	Console information and images (Wikimedia)
Genres	IGDB API	Game genres
Synthetic Data	Faker (Python)	Customers, orders, reviews, addresses, etc.

ETL Process

1. **Extract**: Game and console data is fetched from IGDB API (`dataload.py`)
2. **Transform**: Data is converted to database schema format
3. **Load**: Loaded into MySQL database


EXTRACT
IGDB API
Faker Library
Games, Consoles,\nGenres, Media
Customers, Orders,\nReviews, Addresses
TRANSFORM
dataload.py
generate_synthetic_data.py
INSERT INTO\nPRODUCT, GAME,\nCONSOLE, GENRE
INSERT INTO\nCUSTOMER, ORDER,\nREVIEW, etc.
LOAD
MySQL Database

```python
# dataload.py - Sample ETL Code
def load_games(cnx, cursor, igdb_genre_map):
    """Fetches games from IGDB and loads into database"""

    api_query = (
        "fields name, summary, first_release_date, "
        "platforms.name, genres, "
        "involved_companies.company.name, "
        "cover.url, screenshots.url; "
        "where platforms = (48, 49, 130, 6); "
        "limit 50;"
    )

    byte_array = wrapper.api_request("games", api_query)
    games_list = json.loads(byte_array)

    for game in games_list:
        # Add to PRODUCT table
        cursor.execute(query_product, (game_name, description, ...))
        product_id = cursor.lastrowid

        # Add to GAME table
        cursor.execute(query_game, (product_id, platform, developer, ...))
        # Add GAME_GENRE relationships
        for genre_id in game["genres"]:
            cursor.execute(query_game_genre, (product_id, genre_id))
```

Synthetic Data Generation

```
# generate_synthetic_data.py - Configuration
NUM_CUSTOMERS = 200
NUM_SUPPLIERS = 15
NUM_BRANCHES = 5
NUM_ORDERS = 250
NUM_REVIEWS = 300
NUM_PURCHASES = 100
NUM_RETURNS = 30
```

5.7 Query Designs

Query 1: Product Listing (Filtered + Paginated)

```
SELECT p.product_id, p.product_name, p.price, p.product_type, p.release_date,
       MAX(pm.media_url) as main_image,
       COALESCE(AVG(r.rating), 0) as avg_rating
FROM PRODUCT p
LEFT JOIN PRODUCT_MEDIA pm ON p.product_id = pm.product_id AND pm.main_image
= TRUE
LEFT JOIN REVIEW r ON p.product_id = r.product_id
LEFT JOIN GAME gm ON p.product_id = gm.product_id
WHERE p.product_type = 'game'
  AND gm.platform REGEXP '(^|, )PlayStation 5($|,)'
  AND p.price BETWEEN 20 AND 100
GROUP BY p.product_id
HAVING avg_rating >= 4.0
ORDER BY p.release_date DESC
LIMIT 24 OFFSET 0;
```

Query 2: Customer Orders with Order Details

```
SELECT
    o.order_id,
    o.order_date,
    o.order_status,
    o.total_amount,
    od.product_id,
    od.quantity,
    od.unit_price,
    p.product_name,
    pm.media_url as image_url
FROM `ORDER` o
JOIN ORDER_DETAIL od ON o.order_id = od.order_id
JOIN PRODUCT p ON od.product_id = p.product_id
LEFT JOIN PRODUCT_MEDIA pm ON p.product_id = pm.product_id AND pm.main_image
= TRUE
```

```sql
WHERE o.customer_id = ?
ORDER BY o.order_date DESC;
```

Query 3: Analytics - Net Revenue and Profit Calculation

```sql
SELECT
    COALESCE(SUM(s.transaction_amount), 0) -
    COALESCE((
        SELECT SUM(r.refund_amount)
        FROM `RETURN` r
        WHERE r.return_status = 'completed'
    ), 0) as total_revenue,
    COALESCE(SUM(s.profit), 0) -
    COALESCE((
        SELECT SUM(r.refund_amount)
        FROM `RETURN` r
        WHERE r.return_status = 'completed'
    ), 0) as total_profit,
    COUNT(s.sale_id) as total_transactions
FROM SALE s
JOIN `ORDER` o ON s.order_id = o.order_id
WHERE o.order_status != 'cancelled';
```

Query 4: Top Selling Products

```sql
SELECT
    p.product_name,
    SUM(od.quantity) - COALESCE((
        SELECT SUM(r.quantity)
        FROM `RETURN` r
        WHERE r.product_id = p.product_id
        AND r.return_status = 'completed'
    ), 0) as total_sold,
    SUM(od.quantity * od.unit_price) as revenue
FROM ORDER_DETAIL od
JOIN PRODUCT p ON od.product_id = p.product_id
JOIN `ORDER` o ON od.order_id = o.order_id
WHERE o.order_status != 'cancelled'
GROUP BY p.product_id, p.product_name
ORDER BY total_sold DESC
LIMIT 5;
```

Query 5: Branch Performance Comparison

```sql
SELECT
    b.branch_name,
    COUNT(s.sale_id) as transaction_count,
    COALESCE(SUM(s.transaction_amount), 0) as revenue,
    COALESCE(SUM(s.profit), 0) as profit
FROM BRANCH b
LEFT JOIN SALE s ON b.branch_id = s.branch_id
JOIN `ORDER` o ON s.order_id = o.order_id
WHERE o.order_status != 'cancelled'
```

```sql
GROUP BY b.branch_id, b.branch_name
ORDER BY revenue DESC;
```

6. Application Program Introduction and Sample Usage

6.1 Installation and Running

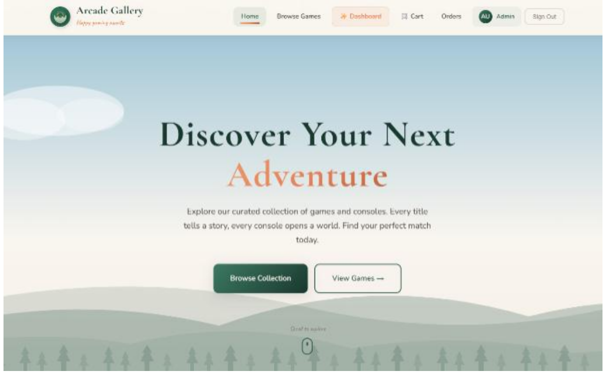
Requirements

- Python 3.x
- Node.js and npm
- MySQL 8.0+

Installation Steps

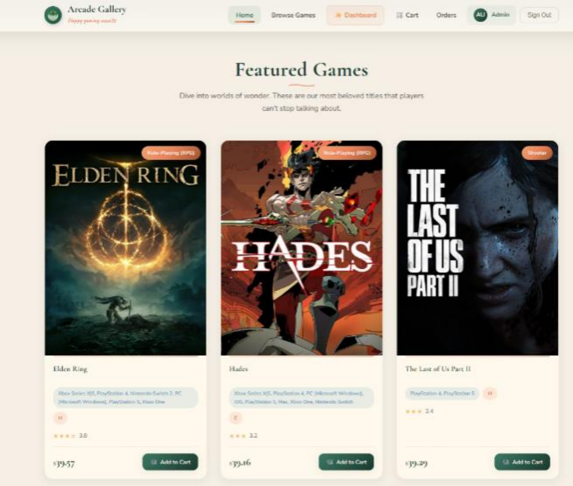
```bash
# 1. Clone the repository
git clone <repository-url>
cd gameD0NTst0p

# 2. Install Python dependencies
pip install -r requirements.txt

# 3. Create .env file
cp .env_example .env
# Edit .env file and enter database credentials

# 4. Create the database
mysql -u root -p < database/dbsetup.sql

# 5. Load data
cd database
python dataload.py
python generate_synthetic_data.py
cd ..

# 6. Start the backend
python app.py

# 7. Open new terminal and start frontend
cd frontend
npm install
npm start
```

6.2 Application Screenshots and Features



Discover Your Next Adventure

Explore our curated collection of games and consoles. Every title tells a story, every console opens a world. Find your perfect match today.



Featured Games

Dive into worlds of wonder. These are our most beloved titles that players can't stop talking about.

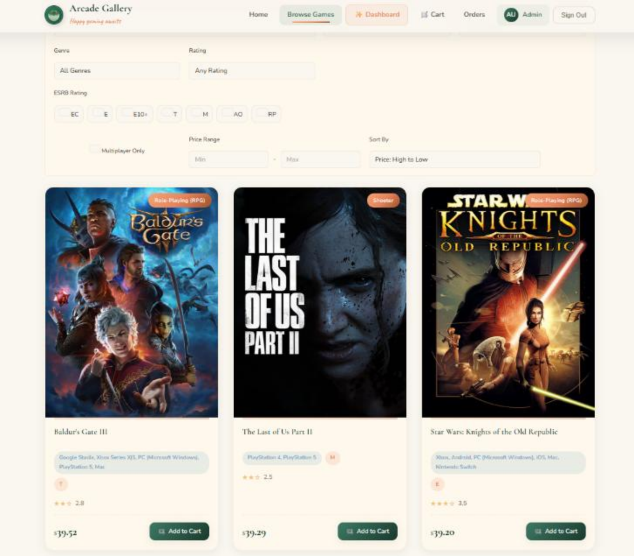
Elden Ring — Xbox Series X|S, PlayStation 4, Nintendo Switch 2, PC (Microsoft Windows), PlayStation 5, Xbox One — ★★★★ 3.8 — $39.57

Hades — Xbox Series X|S, PlayStation 4, PC (Microsoft Windows), iOS, PlayStation 5, Mac, Xbox One, Nintendo Switch — ★★★ 3.2 — $39.16

The Last of Us Part II — PlayStation 4, PlayStation 5 — ★★★ 3.4 — $39.29

Home Page

- Featured games and consoles
- Quick access buttons
- Modern design



Genre: All Genres

Rating: Any Rating

ESRB Rating

Multiplayer Only

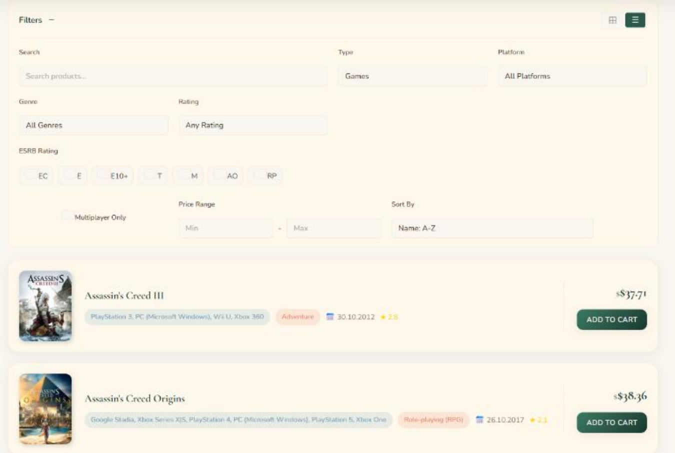
Price Range: Min - Max

Sort By: Price: High to Low

Baldur's Gate III — Google Stadia, Xbox Series X|S, PC (Microsoft Windows), PlayStation 5, Mac — ★★★ 2.8 — $39.52

The Last of Us Part II — PlayStation 4, PlayStation 5 — ★★★ 2.5 — $39.29

Star Wars: Knights of the Old Republic — Xbox, Android, PC (Microsoft Windows), iOS, Mac, Nintendo Switch — ★★★★ 3.5 — $39.20



Search: Search products...

Type: Games

Platform: All Platforms

Genre: All Genres

Rating: Any Rating

ESRB Rating

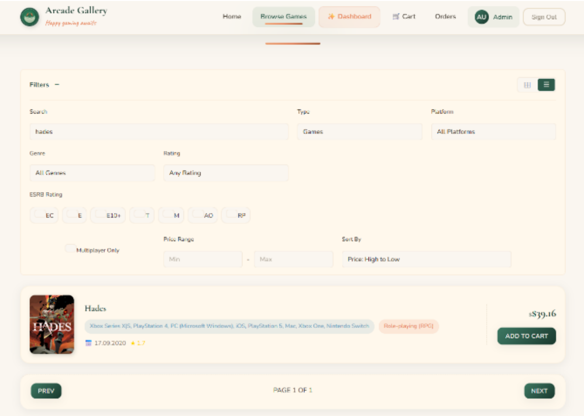
Multiplayer Only

Price Range: Min - Max

Sort By: Name: A-Z

Assassin's Creed III — PlayStation 3, PC (Microsoft Windows), Wii U, Xbox 360 — 30.10.2012 ★ 2.8 — $$37.71

Assassin's Creed Origins — Google Stadia, Xbox Series X|S, PlayStation 4, PC (Microsoft Windows), PlayStation 5, Xbox One — 26.10.2017 ★ 2.1 — $$38.36



Search: hades

Type: Games

Platform: All Platforms

Genre: All Genres

Rating: Any Rating

ESRB Rating

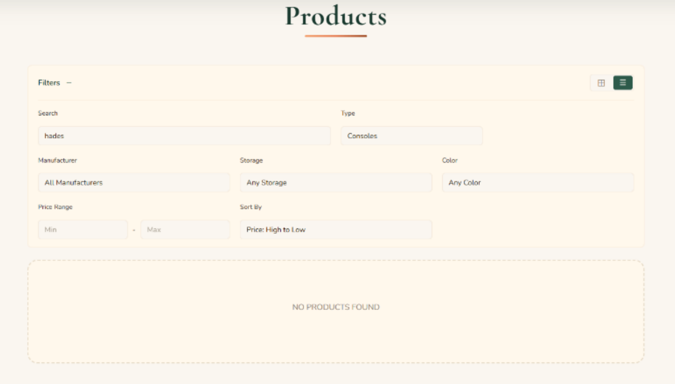
Multiplayer Only

Price Range: Min - Max

Sort By: Price: High to Low

Hades — Xbox Series X|S, PlayStation 4, PC (Microsoft Windows), iOS, PlayStation 5, Mac, Xbox One, Nintendo Switch — 17.09.2020 ★ 1.7 — $$39.16

PAGE 1 OF 1



Products

Search: hades

Type: Consoles

Manufacturer: All Manufacturers

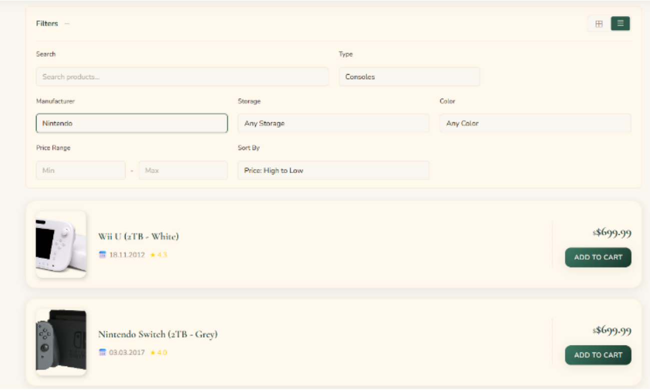
Storage: Any Storage

Color: Any Color

Price Range: Min - Max

Sort By: Price: High to Low

NO PRODUCTS FOUND



Filters –

Search: Search products...

Type: Consoles

Manufacturer: Nintendo

Storage: Any Storage

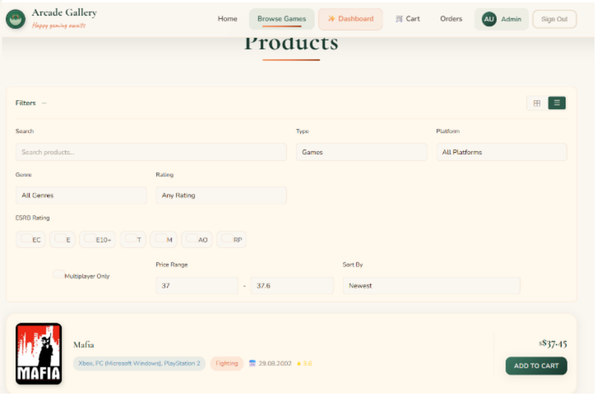
Color: Any Color

Price Range: Min - Max

Sort By: Price: High to Low

Wii U (2TB - White)
18.11.2012 ★ 4.3
s$699.99
ADD TO CART

Nintendo Switch (2TB - Grey)
03.03.2017 ★ 4.0
s$699.99
ADD TO CART



Arcade Gallery
Happy gaming awaits

Products

Filters –

Search: Search products...

Type: Games

Platform: All Platforms

Genre: All Genres

Rating: Any Rating

ESRB Rating: □ EC □ E □ E10+ □ T □ M □ AO □ RP

□ Multiplayer Only

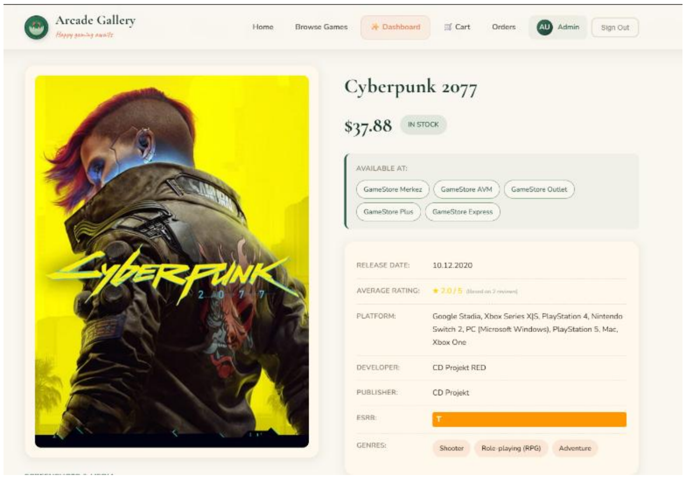
Price Range: 37 - 37.6

Sort By: Newest

Mafia
Xbox, PC (Microsoft Windows), PlayStation 2 · Fighting · 29.08.2002 ★ 3.6
s$37.45
ADD TO CART

Product Listing (Products)

- Product cards in grid view
- Filtering options:
  - Product type (Game/Console)
  - Genre
  - Platform
  - Price range
  - ESRB rating
  - Multiplayer feature
- Sorting options (Price, Date, Rating)
- Pagination



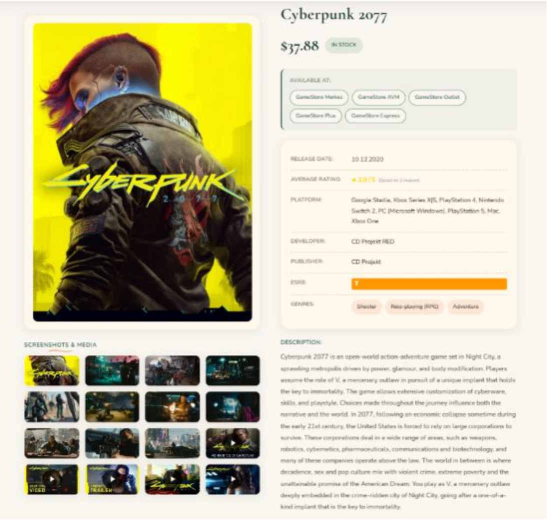
Arcade Gallery
Happy gaming awaits

Cyberpunk 2077

$37.88 IN STOCK

AVAILABLE AT:
GameStore Merkez · GameStore AVM · GameStore Outlet · GameStore Plus · GameStore Express

RELEASE DATE:	10.12.2020
AVERAGE RATING:	★ 2.0 / 5 (Based on 2 reviews)
PLATFORM:	Google Stadia, Xbox Series X|S, PlayStation 4, Nintendo Switch 2, PC (Microsoft Windows), PlayStation 5, Mac, Xbox One
DEVELOPER:	CD Projekt RED
PUBLISHER:	CD Projekt
ESRB:	T
GENRES:	Shooter · Role-playing (RPG) · Adventure



Cyberpunk 2077

$37.88 IN STOCK

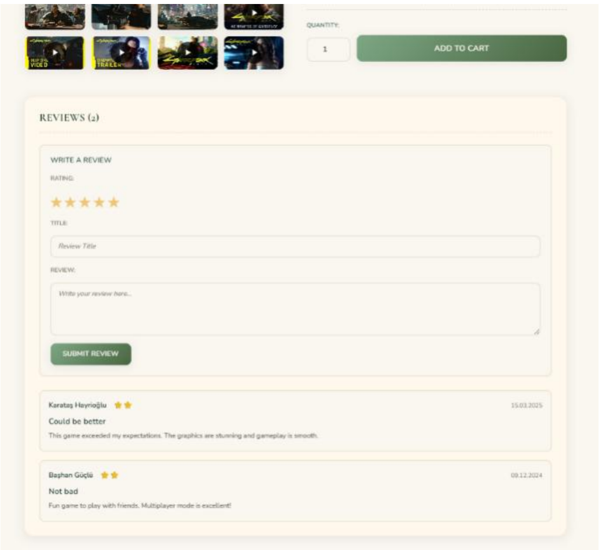
AVAILABLE AT:
GameStore Merkez · GameStore AVM · GameStore Outlet · GameStore Plus · GameStore Express

RELEASE DATE:	10.12.2020
AVERAGE RATING:	★ 2.0 / 5 (Based on 2 reviews)
PLATFORM:	Google Stadia, Xbox Series X|S, PlayStation 4, Nintendo Switch 2, PC (Microsoft Windows), PlayStation 5, Mac, Xbox One
DEVELOPER:	CD Projekt RED
PUBLISHER:	CD Projekt
ESRB:	T
GENRES:	Shooter · Role-playing (RPG) · Adventure

SCREENSHOTS & MEDIA

DESCRIPTION:

Cyberpunk 2077 is an open-world action-adventure game set in Night City, a sprawling metropolis driven by power, glamour, and body modification. Players assume the role of V, a mercenary outlaw in pursuit of a unique implant that holds the key to immortality. The game allows extensive customization of cyberware, skills, and playstyle. Choices made throughout the journey influence both the narrative and the world. In 2077, following an economic collapse sometime during the early 21st century, the United States is forced to rely on large corporations to survive. These corporations deal in a wide range of areas, such as weapons, robotics, cybernetics, pharmaceuticals, communications and biotechnology, and many of these companies operate above the law. The world in between is where decadence, sex and pop culture mix with violent crime, extreme poverty and the unattainable promise of the American Dream. You play as V, a mercenary outlaw deeply embedded in the crime-ridden city of Night City, going after a one-of-a-kind implant that is the key to immortality.



QUANTITY:

1

ADD TO CART

REVIEWS (2)

WRITE A REVIEW

RATING:

★★★★★

TITLE:

Review Title

REVIEW:

Write your review here...

SUBMIT REVIEW

Karataş Hayrioğlu ★★ 15.03.2025

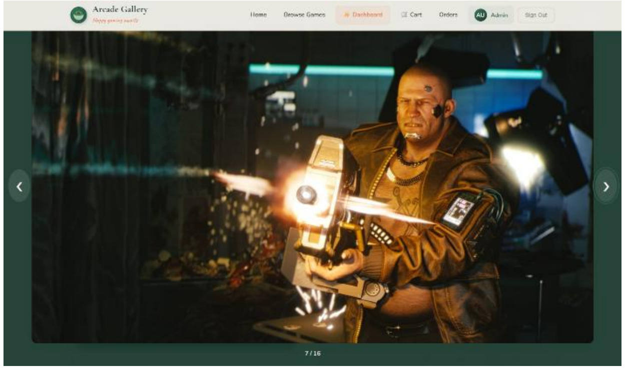
Could be better

This game exceeded my expectations. The graphics are stunning and gameplay is smooth.

Başhan Güçlü ★★ 09.12.2024

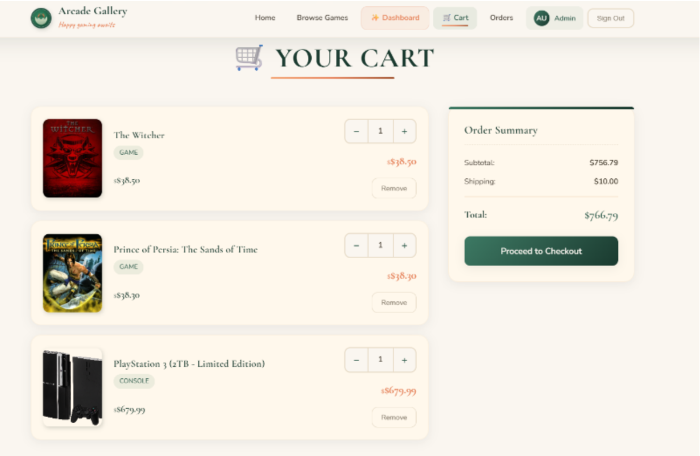
Not bad

Fun game to play with friends. Multiplayer mode is excellent!



Product Detail

- Product images gallery
- Detailed information (platform, developer, ESRB, etc.)
- Stock status and available branches
- Customer reviews and ratings
- Add to cart button



YOUR CART

The Witcher

GAME

s$38.50

− 1 +

s$38.50

Remove

Prince of Persia: The Sands of Time

GAME

s$38.30

− 1 +

s$38.30

Remove

PlayStation 3 (2TB - Limited Edition)

CONSOLE

s$679.99

− 1 +

s$679.99

Remove

Order Summary

Subtotal: $756.79

Shipping: $10.00

Total: $766.79

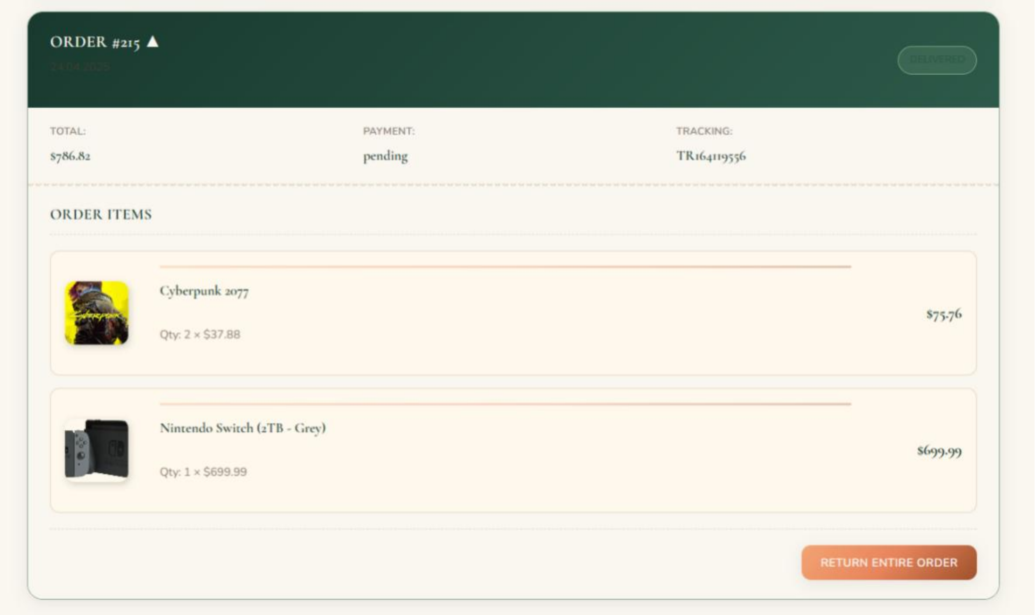
Proceed to Checkout

Shopping Cart

- List of items in cart
- Quantity update
- Remove product
- Total price calculation
- Proceed to checkout

Checkout

- Delivery address selection
- Billing address
- Payment method selection
- Order summary
- Order confirmation



ORDER #215 ▲

24.04.2025

DELIVERED

TOTAL: $786.82

PAYMENT: pending

TRACKING: TR164119556

ORDER ITEMS

Cyberpunk 2077

Qty: 2 × $37.88

$75.76

Nintendo Switch (2TB - Grey)

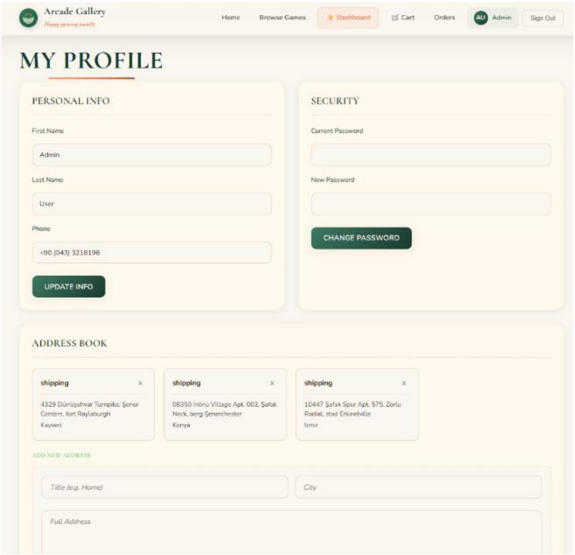
Qty: 1 × $699.99

$699.99

RETURN ENTIRE ORDER

My Orders

- Order history
- Order status tracking
- Tracking number
- Return request creation



Arcade Gallery

Happy gaming awaits

Home Browse Games Dashboard Cart Orders AU Admin Sign Out

MY PROFILE

PERSONAL INFO

First Name: Admin

Last Name: User

Phone: +90 (043) 3218196

UPDATE INFO

SECURITY

Current Password

New Password

CHANGE PASSWORD

ADDRESS BOOK

shipping ×
4329 Dürrüşehvar Turnpike, Şener Centers, fort Raylaburgh
Kayseri

shipping ×
08350 İnönü Village Apt. 003, Şafak Neck, berg Şenerchester
Konya

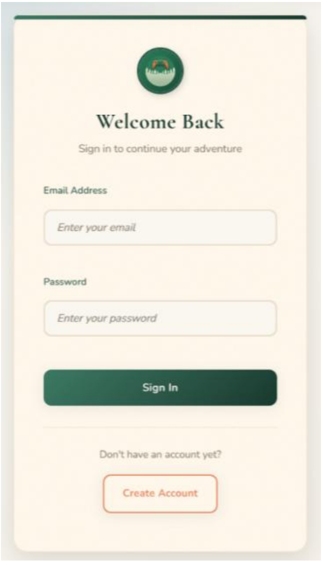
shipping ×
10447 Şafak Spur Apt. 575, Zorlu Radial, stad Erkinelville
İzmir

ADD NEW ADDRESS

Title (e.g. Home)

City

Full Address



Welcome Back

Sign in to continue your adventure

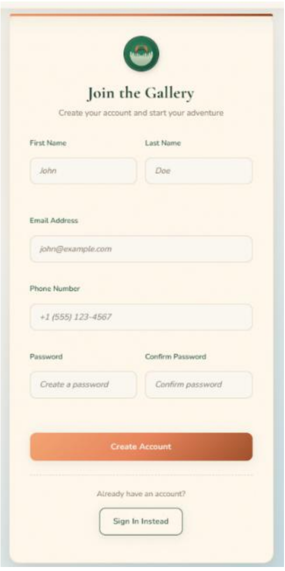
Email Address: Enter your email

Password: Enter your password

Sign In

Don't have an account yet?

Create Account



Join the Gallery

Create your account and start your adventure

First Name: John

Last Name: Doe

Email Address: john@example.com

Phone Number: +1 (555) 123-4567

Password: Create a password

Confirm Password: Confirm password

Create Account

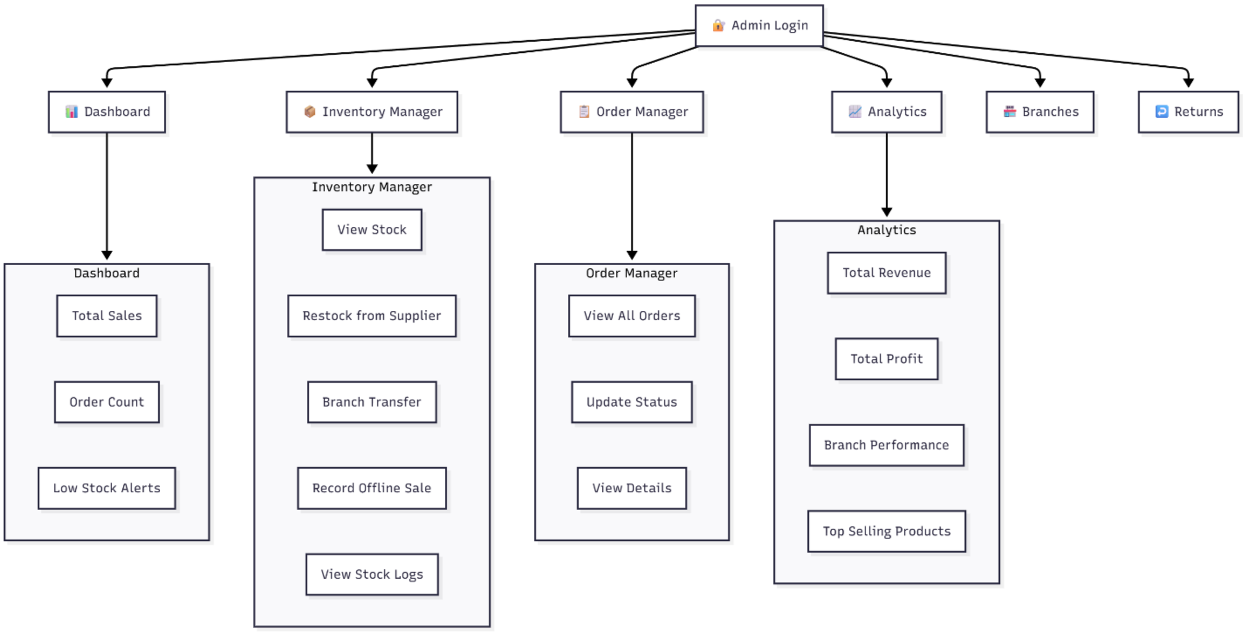
Already have an account?

Sign In Instead

Profile

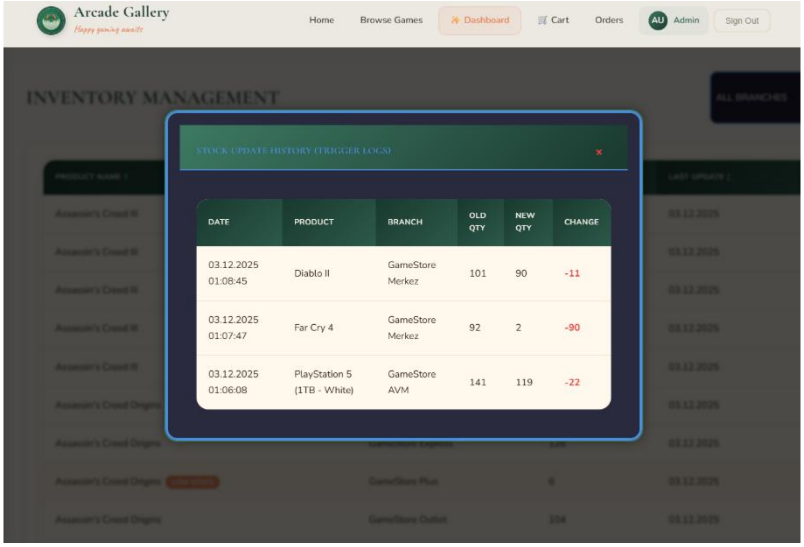
- Edit personal information
- Change password
- Address management

6.3 Admin Panel

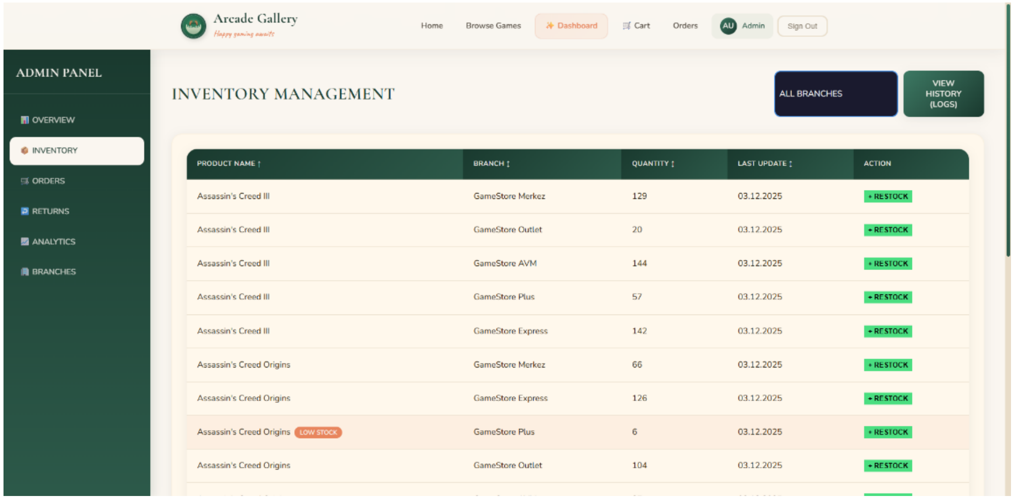


Dashboard

- Total sales amount
- Order count
- Product count
- Low stock alerts



DATE	PRODUCT	BRANCH	OLD QTY	NEW QTY	CHANGE
03.12.2025 01:08:45	Diablo II	GameStore Merkez	101	90	-11
03.12.2025 01:07:47	Far Cry 4	GameStore Merkez	92	2	-90
03.12.2025 01:06:08	PlayStation 5 (1TB - White)	GameStore AVM	141	119	-22

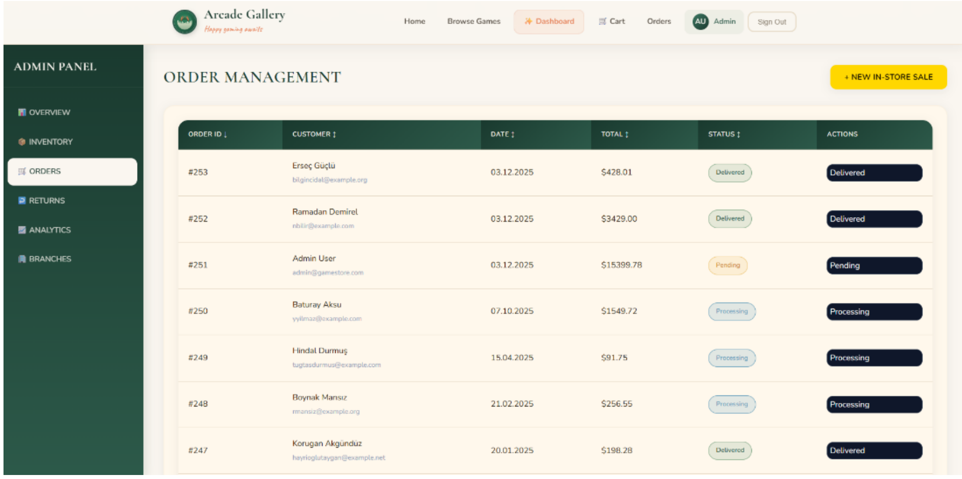


INVENTORY MANAGEMENT

PRODUCT NAME ↑	BRANCH ↕	QUANTITY ↕	LAST UPDATE ↕	ACTION
Assassin's Creed III	GameStore Merkez	129	03.12.2025	+ RESTOCK
Assassin's Creed III	GameStore Outlet	20	03.12.2025	+ RESTOCK
Assassin's Creed III	GameStore AVM	144	03.12.2025	+ RESTOCK
Assassin's Creed III	GameStore Plus	57	03.12.2025	+ RESTOCK
Assassin's Creed III	GameStore Express	142	03.12.2025	+ RESTOCK
Assassin's Creed Origins	GameStore Merkez	66	03.12.2025	+ RESTOCK
Assassin's Creed Origins	GameStore Express	126	03.12.2025	+ RESTOCK
Assassin's Creed Origins LOW STOCK	GameStore Plus	6	03.12.2025	+ RESTOCK
Assassin's Creed Origins	GameStore Outlet	104	03.12.2025	+ RESTOCK

Inventory Manager

- Stock status of all products
- Branch-based filtering
- Restock from supplier
- Inter-branch transfer
- Record in-store sales
- Stock change logs

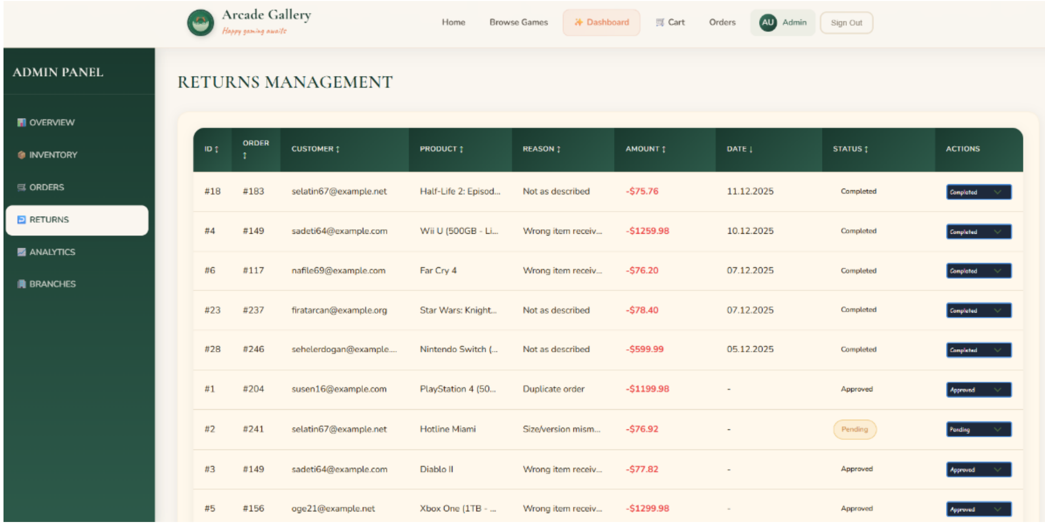


ORDER MANAGEMENT

ORDER ID ↓	CUSTOMER ↕	DATE ↕	TOTAL ↕	STATUS ↕	ACTIONS
#253	Erseç Güçlü bilgincidal@example.org	03.12.2025	$428.01	Delivered	Delivered
#252	Ramadan Demirel nbilir@example.com	03.12.2025	$3429.00	Delivered	Delivered
#251	Admin User admin@gamestore.com	03.12.2025	$15399.78	Pending	Pending
#250	Baturay Aksu yyilmaz@example.com	07.10.2025	$1549.72	Processing	Processing
#249	Hindal Durmuş tugtasdurmus@example.com	15.04.2025	$91.75	Processing	Processing
#248	Boynak Mansız rmansiz@example.org	21.02.2025	$256.55	Processing	Processing
#247	Korugan Akgündüz hayrioglutaygan@example.net	20.01.2025	$198.28	Delivered	Delivered

Order Manager

- List of all orders
- Status update (pending → shipped → delivered)
- View order details
- Cancel and return operations

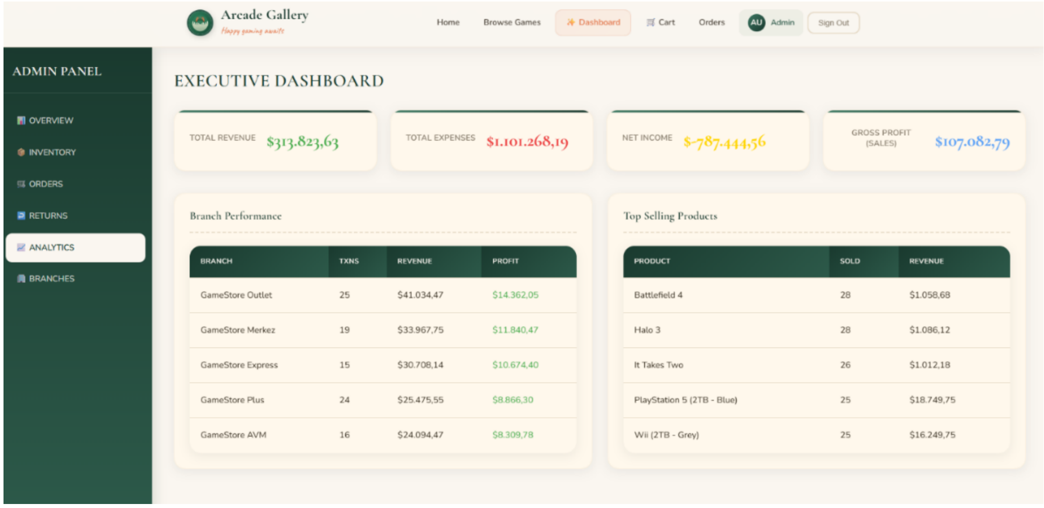


RETURNS MANAGEMENT

ID ↕	ORDER ↕	CUSTOMER ↕	PRODUCT ↕	REASON ↕	AMOUNT ↕	DATE ↓	STATUS ↕	ACTIONS
#18	#183	selatin67@example.net	Half-Life 2: Episod...	Not as described	-$75.76	11.12.2025	Completed	Completed
#4	#149	sadeti64@example.com	Wii U (500GB - Li...	Wrong item receiv...	-$1259.98	10.12.2025	Completed	Completed
#6	#117	nafile69@example.com	Far Cry 4	Wrong item receiv...	-$76.20	07.12.2025	Completed	Completed
#23	#237	firatarcan@example.org	Star Wars: Knight...	Not as described	-$78.40	07.12.2025	Completed	Completed
#28	#246	sehelerdogan@example....	Nintendo Switch (...	Not as described	-$599.99	05.12.2025	Completed	Completed
#1	#204	susen16@example.com	PlayStation 4 (50...	Duplicate order	-$1199.98	-	Approved	Approved
#2	#241	selatin67@example.net	Hotline Miami	Size/version mism...	-$76.92	-	Pending	Pending
#3	#149	sadeti64@example.com	Diablo II	Wrong item receiv...	-$77.82	-	Approved	Approved
#5	#156	oge21@example.net	Xbox One (1TB - ...	Wrong item receiv...	-$1299.98	-	Approved	Approved

Returns Management

- Return requests list
- Approve/Reject
- Complete return



EXECUTIVE DASHBOARD

TOTAL REVENUE $313.823,63

TOTAL EXPENSES $1.101.268,19

NET INCOME $-787.444,56

GROSS PROFIT (SALES) $107.082,79

Branch Performance

BRANCH	TXNS	REVENUE	PROFIT
GameStore Outlet	25	$41.034,47	$14.362,05
GameStore Merkez	19	$33.967,75	$11.840,47
GameStore Express	15	$30.708,14	$10.674,40
GameStore Plus	24	$25.475,55	$8.866,30
GameStore AVM	16	$24.094,47	$8.309,78

Top Selling Products

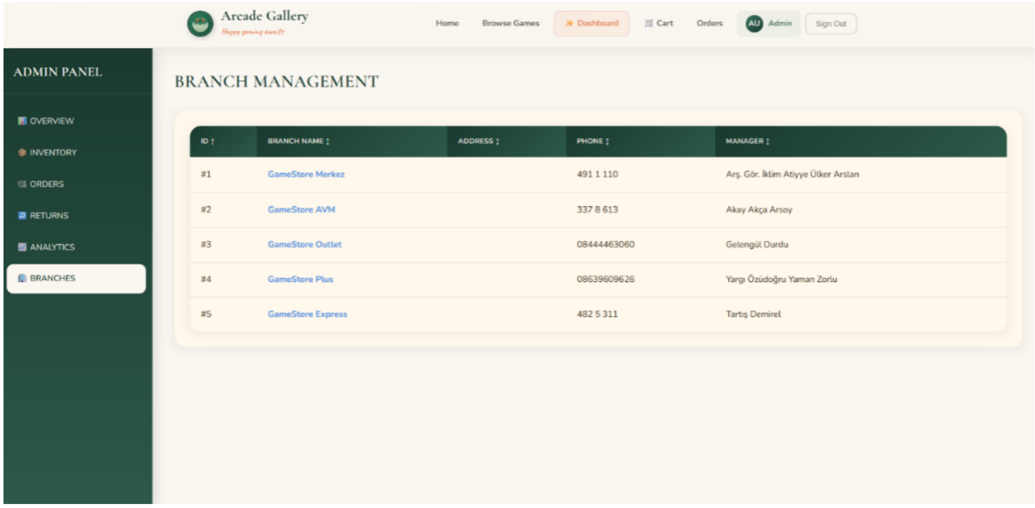
PRODUCT	SOLD	REVENUE
Battlefield 4	28	$1.058,68
Halo 3	28	$1.086,12
It Takes Two	26	$1.012,18
PlayStation 5 (2TB - Blue)	25	$18.749,75
Wii (2TB - Grey)	25	$16.249,75

Analytics

- Total revenue and profit
- Branch performance comparison
- Top selling products

Branches

- Branch information list
- Branch-based inventory view



BRANCH MANAGEMENT

ID ↑	BRANCH NAME ↕	ADDRESS ↕	PHONE ↕	MANAGER ↕
#1	GameStore Merkez		491 1 110	Arş. Gör. İklim Atiyye Ülker Arslan
#2	GameStore AVM		337 8 613	Akay Akça Arsoy
#3	GameStore Outlet		08444463060	Gelengül Durdu
#4	GameStore Plus		08639609626	Yargı Özüdoğru Yaman Zorlu
#5	GameStore Express		482 5 311	Tartış Demirel

6.4 API Endpoints

Product APIs

Method	Endpoint	Description
GET	/api/products	Product list (filtered + paginated)
GET	/api/products/:id	Product details
GET	/api/genres	Genre list
GET	/api/platforms	Platform list

Customer APIs

Method	Endpoint	Description
POST	/api/customers/register	Registration
POST	/api/customers/login	Login
GET	/api/profile	Profile information
PUT	/api/profile/update	Update profile
PUT	/api/profile/password	Change password

Cart APIs

Method	Endpoint	Description
GET	/api/cart/:customer_id	Cart contents
POST	/api/cart	Add to cart
DELETE	/api/cart/:customer_id/:product_id	Remove from cart

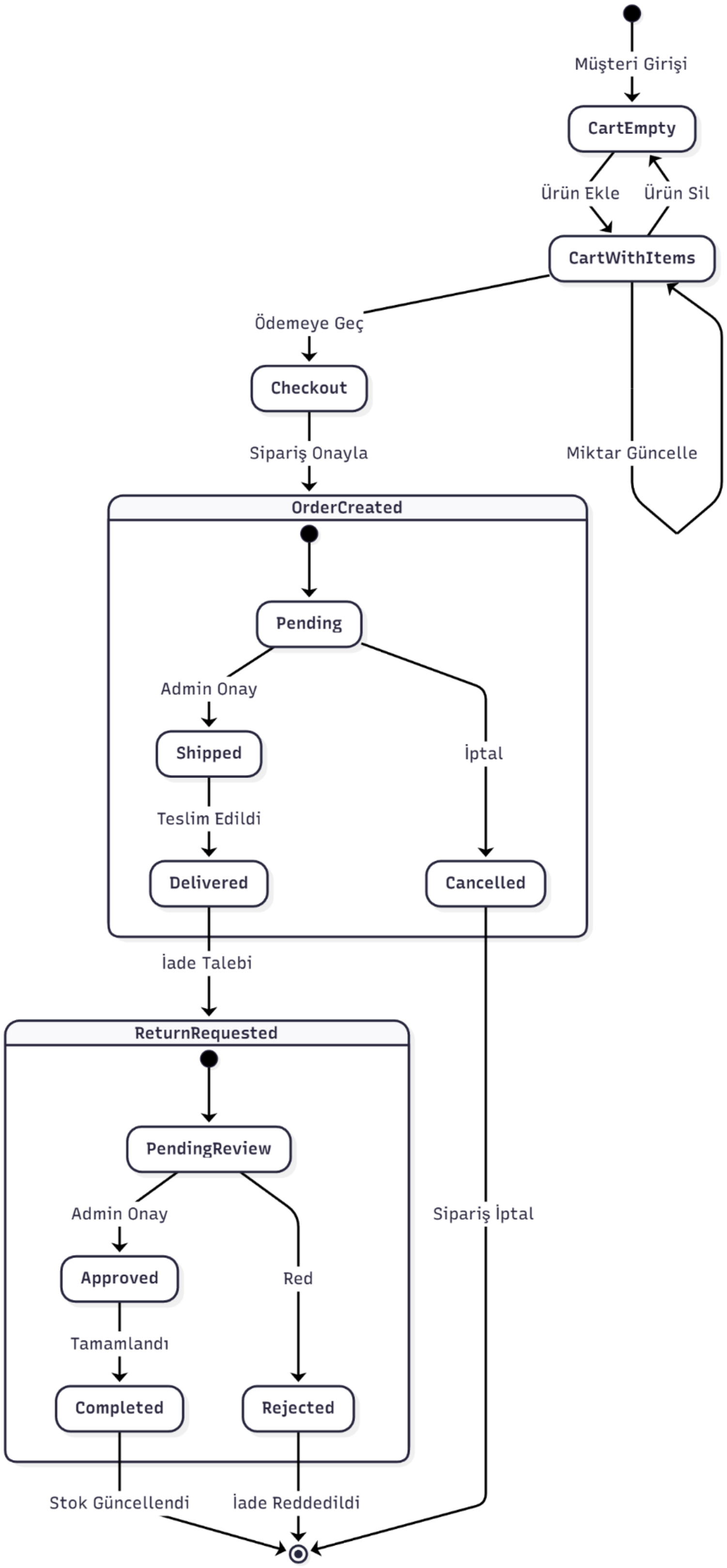
Order APIs

Method	Endpoint	Description
POST	/api/orders	Create order
GET	/api/orders/:customer_id	Customer orders
PUT	/api/orders/:order_id/status	Update status

Admin APIs

Method	Endpoint	Description
GET	/api/admin/stats	Dashboard statistics
GET	/api/admin/inventory	Inventory list
GET	/api/admin/orders	All orders (uses VIEW)
GET	/api/admin/analytics	Analytics data
POST	/api/admin/restock	Restock inventory
POST	/api/admin/inventory/transfer	Stock transfer
POST	/api/admin/sales/offline	In-store sale
GET	/api/admin/returns	Returns list
PUT	/api/admin/returns/:id/status	Update return status

6.5 Sample Usage Scenarios



Scenario 1: Customer Purchasing a Game

15. Customer logs into the site
16. Goes to "Products" page
17. Filters: Type=Game, Platform=PlayStation 5
18. Selects desired game and views detail page
19. Clicks "Add to Cart" button
20. Goes to cart page
21. Clicks "Checkout" button
22. Selects delivery address
23. Selects payment method
24. Confirms order
25. Tracks order status from "My Orders" page

Scenario 2: Admin Stock Management

26. Logs in with admin account
27. Accesses admin panel
28. Goes to "Inventory" page
29. Finds low stock product
30. Clicks "Restock" button
31. Enters supplier, quantity, and unit cost
32. Stock is replenished and logged in STOCK_LOG

Scenario 3: Return Process

33. Customer goes to "My Orders" page
34. Clicks "Request Return" on delivered order
35. Enters return reason
36. Admin goes to "Returns" page in admin panel
37. Reviews return request
38. Selects "Approve" or "Reject"
39. If approved, completes with "Complete"
40. Stock is automatically restored

7. Conclusion

7.1 Project Summary

In this project, a comprehensive B2C e-commerce platform was developed as part of the BIL372 Database Systems course. The project simulates an online store selling physical video games and consoles.

7.2 Applied Database Concepts

Concept	Implementation
Superclass/Subclass	PRODUCT → GAME, CONSOLE
Weak Entity	ORDER_DETAIL (dependent on ORDER)
Associative Entity	GAME_GENRE, CART
Composite Attributes	Delivery and billing addresses
Foreign Keys	15+ relationships
CHECK Constraints	price > 0, quantity >= 0, rating 1-5
UNIQUE Constraints	email, (product_id, branch_id)
Views	5 views (3 reporting + 2 BCNF compliance)
Indexes	6 performance indexes
Triggers	Stock change log
Transactions	Order creation, stock update
Normalization	All tables in 1NF, 2NF, 3NF, and BCNF

7.3 Technical Achievements

41. **3-Tier Architecture**: Separated Presentation, Application, Data layers
42. **RESTful API**: Full CRUD operations with 30+ endpoints
43. **IGDB Integration**: Enriched catalog with real game data
44. **Synthetic Data**: 1000+ records with Faker library
45. **Admin Panel**: Comprehensive management tools
46. **Responsive Design**: Modern and user-friendly interface

7.4 Challenges and Solutions

Challenge	Solution
IGDB API data inconsistency	Supplemented with synthetic data
Stock management complexity	Used Transactions and Triggers
M:N relationship management	Associative entity tables
Performance optimization	Indexes and Views

7.5 Future Improvements

- ☐ Real payment integration (Stripe, PayPal)
- ☐ Email notifications
- ☐ Wishlist feature
- ☐ Product comparison
- ☐ Advanced search (full-text search)
- ☐ Mobile application
- ☐ Chatbot support

8. References

1. **IGDB API Documentation** - https://api-docs.igdb.com/
2. **MySQL 8.0 Reference Manual** - https://dev.mysql.com/doc/refman/8.0/en/
3. **Flask Documentation** - https://flask.palletsprojects.com/
4. **React Documentation** - https://react.dev/
5. **Faker Python Library** - https://faker.readthedocs.io/
6. **mysql-connector-python** - https://dev.mysql.com/doc/connector-python/en/
7. **Axios HTTP Client** - https://axios-http.com/
8. **Elmasri & Navathe, "Fundamentals of Database Systems"** - 7th Edition

Appendices

Appendix A: Differences from Interim Report

During implementation, some differences from the interim report occurred:

Interim Report	Implementation	Explanation
CUSTOMER table	is_admin added	For admin authorization
ORDER address	Inline columns	Separate columns instead of composite
BRANCH address_id	Preserved	FK linked to ADDRESS
-	STOCK_LOG table	Log table for trigger
PURCHASE.total_cost	Removed	BCNF compliance - calculated via VIEW
SALE.profit	Removed	BCNF compliance - calculated via VIEW
-	VIEW_PURCHASE_WITH_TOTAL	total_cost = quantity × unit_cost
-	VIEW_SALE_WITH_PROFIT	profit = transaction_amount - cost

BCNF Compliance Note: The derived columns (total_cost, profit) in PURCHASE and SALE tables in the interim report were causing 3NF/BCNF violations. These columns were removed and replaced with VIEWs. Thus, all tables are now fully compliant with 1NF, 2NF, 3NF, and BCNF.

These differences arose from practical needs during application development and normalization requirements.

Appendix B: File Structure

```
gameD0NTst0p/
├── app.py                      # Flask Backend (2100+ lines)
├── requirements.txt            # Python dependencies
├── .env_example                # Sample environment variables
├── run_project.bat             # Auto-start script
├── README.md                   # Project description
│
├── database/
│   ├── dbsetup.sql             # Table creation DDL
│   ├── dataload.py             # ETL from IGDB
│   ├── generate_synthetic_data.py  # Data generation with Faker
│   ├── igdb_service.py         # IGDB API wrapper
│   └── init_db.py              # Database initialization
│
├── frontend/
│   ├── package.json            # React dependencies
│   ├── public/                 # Static files
│   └── src/
│       ├── App.js              # Main application
│       ├── context/            # Auth context
│       ├── components/         # UI components
│       ├── pages/              # Page components
│       │   ├── Home.js
│       │   ├── Products.js
│       │   ├── ProductDetail.js
│       │   ├── Cart.js
│       │   ├── Checkout.js
│       │   ├── Orders.js
│       │   ├── Profile.js
│       │   └── admin/
│       │       ├── AdminDashboard.js
│       │       ├── InventoryManager.js
│       │       ├── OrderManager.js
│       │       ├── Analytics.js
│       │       ├── Branches.js
│       │       └── Returns.js
│       └── services/
│           └── api.js          # API calls
│
└── reports/
    ├── *_SonRapor.pdf         # Final report
    ├── *_AraRapor.pdf         # Interim report
    ├── screenshots/           # Screenshots used in the final report
    └── diagrams/              # Diagrams used in the final report
```