

**Billiards Management**

**Program**

**Documentation**

|  |  |
| --- | --- |
| Authors | C2312L – Group 1 |
| Hieu Nguyen Van |
| Quan Luu Minh |
| Manh Nguyen Duc |
| Long Nguyen Viet |
| Date | 29/03/2025 |
| Instructor | Tuan Dang Minh |

Hanoi, 03/2025

**INDEX**

[**1.** **Problem research:** 3](#_Toc193838018)

[1.1. General introduction 3](#_Toc193838019)

[1.2. Current situation 3](#_Toc193838020)

[1.3. Proposed system functions 4](#_Toc193838021)

[1.4. Boundaries of the program 5](#_Toc193838028)

[1.5. Hardware and software requirements 9](#_Toc193838029)

[**2.** **Requirements analysis** 9](#_Toc193838032)

[2.1. Users of the system 9](#_Toc193838033)

[2.2. System Functions 9](#_Toc193838034)

[**3.** **System database design** 13](#_Toc193838035)

[3.1. ERD 13](#_Toc193838036)

[3.2. List of tables 14](#_Toc193838037)

[3.3. Relationship Diagram 18](#_Toc193838038)

[3.4. Sitemap 20](#_Toc193838039)

[**4.** **System functions design** 21](#_Toc193838040)

[4.1. Pool tables management functionality 21](#_Toc193838041)

[4.2. Orders management functionality 23](#_Toc193838042)

[4.3. Products management functionality 25](#_Toc193838043)

[4.4. Users management functionality 28](#_Toc193838044)

[4.5. Customers management functionality 29](#_Toc193838045)

[4.6. Report management functionality 30](#_Toc193838046)

[**5.** **Validation checklists** 33](#_Toc193838047)

[**Appendices: ACTIVITY DIAGRAM OF FUNCTIONALITIES** 38](#_Toc193838048)

1. **Problem research:**
   1. **General introduction**

**Program origin:**

The Billiard Management Program was developed with the goal of providing a comprehensive solution for managing billiard clubs, helping owners optimize operations, track revenue, and enhance customer experience. This software is designed based on the real needs of billiard businesses, from small independent establishments to large professional chains. With a user-friendly interface and a wide range of smart features, the Billiard Management Program simplifies management processes, ensuring accuracy and efficiency.

**Why is this program needed?**

Billiards is a popular sport enjoyed by people of all ages. As the demand for this sport increases, more and more billiard clubs are opening, offering attractive promotions and services. However, traditional management methods are no longer sufficient to meet the growing needs of club owners. Issues such as monitoring table usage, tracking revenue, managing additional services, and supervising users require a professional system to minimize errors and improve operational efficiency.

The *Billiard Management Program* is not just a management tool but also allows club owners to monitor operations remotely, improve service quality, and enhance customer satisfaction. Implementing this software helps businesses operate more efficiently, increase profits, and gain a competitive advantage in the market.

* 1. **Current situation**

Currently, in Vietnam, several companies provide billiard club and store management solutions, such as **POS365, TPOS, and Suno**.

**Competitor strengths:**

* Intuitive and easy-to-use interface.
* Features for table management and order tracking.
* Online payment integration with digital wallets.

**Competitor weaknesses:**

* Lacks essential functions such as inventory and users management.
* High cost compared to the features and service quality provided.
* The interface is not fully optimized for all devices.
* Minimal customer interaction and no online community support.
* Restrictive table booking system: When a table is in an "Ordered" status, other customers cannot use it, even if the playing session ends earlier than expected.
  1. **Proposed system functions**

### **Pool Tables Management**

#### a. Description:

* Display a list of available and in-use billiards tables, including table name, price, and status.
* Show the quantity of each table type, allowing users to add new table types and adjust prices.
* Display table status list along with the number of tables currently set for each status.
* Detailed information for each table includes its orders and additional table details.
* User can access order details for better management, streamlining workflow, and improving statistics.
* A search bar enables quick and easy table lookup.

#### b. Features:

* Table list display.
* "Create New Table" button.
* "Order" button to view and update table orders directly without switching to the order page.
* "Info" button to display table name and type, with options to update or delete the table.
* "Search" bar for quick table lookup.
* "New Table Category" button to add new table types.

### **Orders management**

#### Description:

* Users can view details of each order associated with a booking.
* The order table displays key details, including: Bill number, Player name, Phone number, Table number, Price, Status, Date
* A print invoice button for convenient customer billing.
* Search functionality to find orders by name or phone number.
* Ability to create new orders directly.
* Double-clicking an order opens a detailed view where users can:
  + Update player name and phone number.
  + Add a new table to the order.
  + Add new order items.
  + Complete the order directly from this page.

#### Features:

* Create, read, update operations for orders.
* Update customer name and phone number.
* Create, read, update operations for booking associated with each order.
* Create, read, update, remove operations for order items.
* Print invoice button for each order.
* Order completion functionality.

### **Products management**

#### Description:

* User can view a list of products, for example: cues for rent, food, drinks, and board games.
* Features for adding, viewing, updating products.
* Ability to add new product categories, as well as update existing products.
* Product search functionality.
* A well-organized interface ensures easy navigation and usability.

#### Features:

* Display product list categorized by type.
* "Add New Product" button.
* Create, Read, Update functionality for products with corresponding buttons.

### **Users management**

#### Description:

* User can manage user's details, including: Avatar, Username, Role, Full Name, Phone Numbers, Birthday, Address, Hire Date
* Easily add new user members.
* Modify individual user roles.
* Update user conveniently without affecting others.

#### Features:

* Create, Read, Update functionality for users management.
* Intuitively labeled buttons for easy navigation and usage.

### **Customers management**

#### Description:

* Display a list of customers who have created accounts, along with a detailed customer information table containing: Customer name, Phone number, Address, Date of birth, Total playtime, Search functionality to find customers by name or phone number.
* Add, update customer records as needed.

1. **Features:**

* User-friendly interface for easy customer lookup.
* Create, read, update functionality for customer management.

### **Report management**

* 1. **Description:**
* The system provides statistical reports and data analysis for business operations.
* These reports help management make data-driven decisions.
* Report types include:
  + Revenue reports by day, month, and year.
  + Customer count reports over time.
  + Most popular products or services report.
  + Role-based access control for viewing reports.

#### Features:

* Create and manage report templates.
* Display reports in various formats (tables, charts, graphs).
* Apply filters and time range settings for reports.
  1. **Boundaries of the program**

#### Lack of detailed inventory management and stock history tracking:

The system does not record detailed information during stock in or out operations, such as supplier details, product series, timestamps, responsible personnel, and stock quantity changes.

There is no functionality to track, analyze, or generate reports on stock history, making inventory management difficult and hindering supply-demand analysis.

#### No integration of multiple payment methods:

The system does not support any online or offline payment methods, such as bank transfers, crupdate or debit cards, e-wallets (Momo, ZaloPay), or QR code payments, leading to inconvenience for customers during transactions.

#### No users attendance tracking system:

The system lacks features to monitor users check-in or check-out times, overtime, and leave management.

It does not provide daily or monthly attendance reports, making HR management and payroll calculations inefficient.

#### Limited to a single location, no multi-branch support:

The system does not support centralized management for multiple branches, making it impossible to synchronize data between stores, which hinders business expansion.

#### No session persistence, no saved login credentials:

The system does not maintain login sessions, requiring users to log in again for each access.

There is no functionality to save passwords for convenient future logins.

#### Passwords are stored as hashed values, cannot be recovered:

Passwords are stored as hashed values and cannot be recovered to their original form, ensuring security but sometimes causing inconvenience for users.

The system does not support user-initiated password recovery or reset. Only the administrator has the authority to change or reset passwords.

#### No user activity logging:

The system does not record user actions, such as adding, modifying, or deleting data, making it difficult to track, monitor, and handle security issues or management errors.

#### No performance testing with large data volumes:

The system has not been tested for performance and processing speed beyond one million records, posing potential risks of system slowdowns as business scales.

#### No multi-language support:

The system supports only a single default language and does not allow switching or expanding to other languages, limiting its ability to serve international customers.

#### No advanced security features (2FA, Data Encryption):

The system lacks advanced security measures such as Two-Factor Authentication (2FA) and encryption for personal and transaction data, increasing the risk of data breaches.

#### No promotion and discount code system:

The system does not provide features to create, manage, and apply promotions, discount codes, or customer loyalty programs, limiting sales growth opportunities.

#### Remove Pool Table & Remove Product Functions cannot be used:

These functions are not yet fully implemented and cannot be used in the current version. Issues include data integrity risks and missing validation checks. They will be updated in future versions to ensure safe and proper functionality.

#### No error logging and handling system:

The system does not log or manage system errors, making troubleshooting and maintenance inefficient.

#### No multitasking capabilities:

Users can only perform one task at a time, as the system does not support opening multiple functional windows simultaneously, decreasing workflow efficiency.

#### No inventory threshold and low-stock alerts:

The system does not support setting minimum stock levels or generating alerts for low-stock products, increasing the risk of stock shortages.

#### No invoice splitting or merging functionality:

The system does not support splitting an invoice among multiple customers or merging separate invoices, causing inconvenience during payment processing.

#### Cannot modify product prices in orders:

The system enforces fixed listed prices and does not allow real-time price adjustments within an invoice.

#### No customer classification system:

The system does not support customer segmentation (e.g., VIP, regular members), making it difficult to apply targeted promotions and personalized customer service.

#### No account lock mechanism for failed login attempts:

The system lacks an automatic account lockout mechanism after multiple failed login attempts, reducing security measures.

#### No export reports in various formats:

Reports cannot be exported in commonly used formats such as PDF or Excel, making data sharing and analysis inconvenient.

#### Supports only one active user login:

The system currently allows only one user to be logged in at a time, limiting operational efficiency.

#### No automatic data backup:

The system does not perform automatic data backups, posing a risk of data loss in case of system failure.

#### No web or mobile platform support:

The system is not accessible via web or mobile applications, limiting its flexibility in usage.

#### No payroll and employment contract management:

The system does not support payroll management, employment contracts, or users benefits, making HR management challenging.

#### No internal notification system:

The system lacks an internal messaging or notification system for user communication, reducing teamwork and coordination efficiency.

Unable to set auto-cancellation time for bookings (handled only in backend)

The system does not provide a frontend option to configure an auto-cancellation time for bookings.

Instead, the cancellation logic must be managed through backend processing, limiting flexibility for users who want to customize booking expiration rules dynamically.

**New bookings cannot be created for "ordered" pool tables, even if playtime ends earlier**

When a pool table has the status "Ordered", the system does not allow new bookings for that table, regardless of whether the actual playtime would conclude before the "Ordered" period begins.

This restriction can lead to inefficient table utilization as potential free time slots remain unavailable for new reservations.

**Only administrators can modify user information**

Regular users are not allowed to update their own profile information; only administrators have the authority to modify user details.

This limitation prevents users from making changes such as updating their names, contact details, or other profile settings.

While this enhances security and administrative control, it may also cause inconvenience for users who need to update their information without administrator intervention.

* 1. **Hardware and software requirements**

### **Hardware**

* Intel Core i3 or i5 Processor or higher
* 8 GB RAM or more
* Color SVGA Monitor
* 500 GB Hard Disk
* Mouse
* Keyboard

### **Software**

* Operating System: Windows
* Software Requirements:
  + Software: IntelliJ Idea
  + Framework: JavaFX, JDK 23
  + Configurations Macros:
  + VM args: --add-modules javafx.controls,javafx.fxml,javafx.base --add-exports=javafx.base or com.sun.javafx.event=org.controlsfx.controls -ea -Xmx1024m
  + Maven : mvn clean install

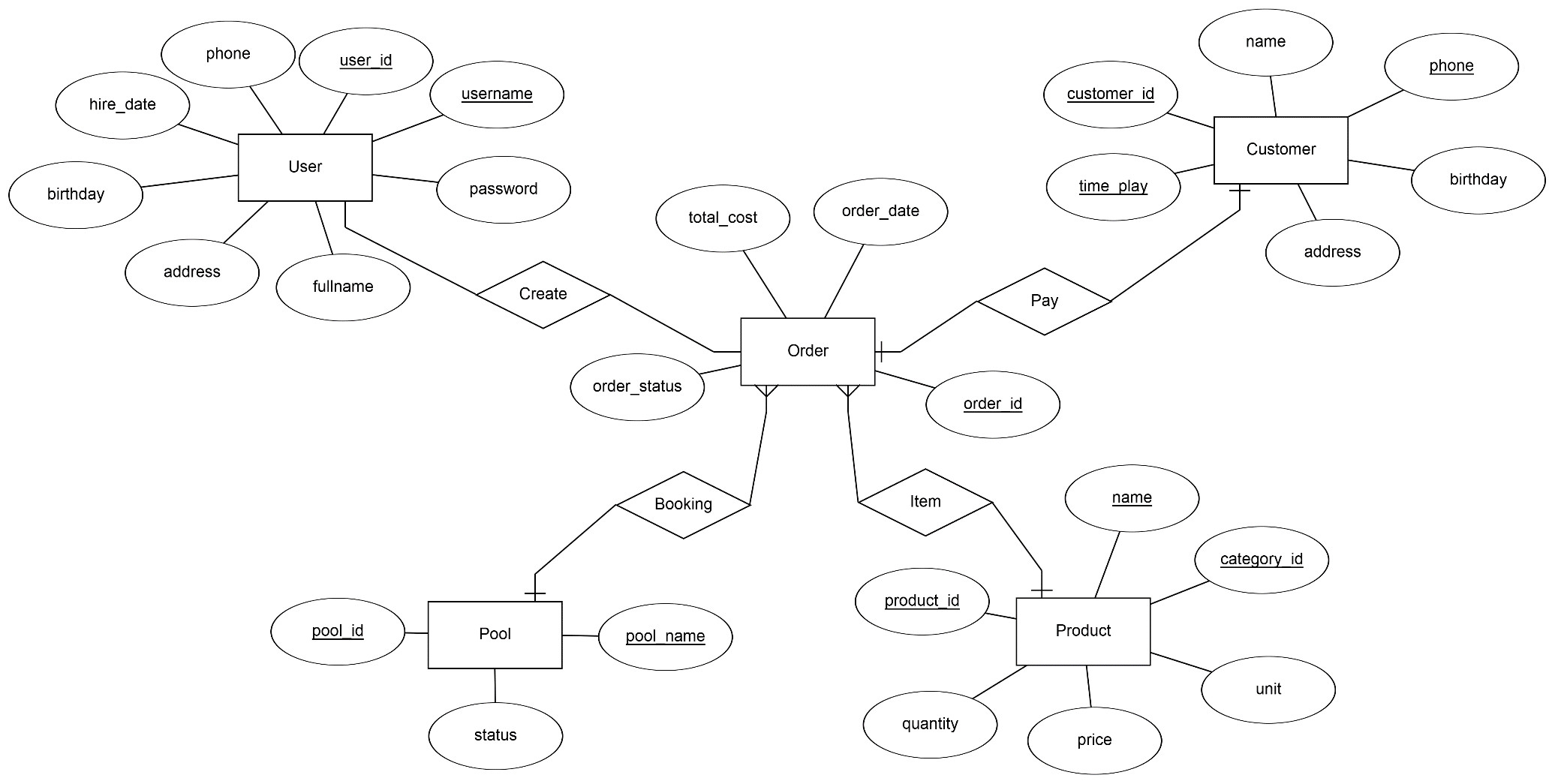
1. **Requirements analysis**
   1. **Users of the system**

Individuals who manage the program, add and update pool tables, pool category, processing orders, manage list of order, manage products list by each product categories, manager list of user and user’s role and permissions, show the report

* 1. **System Functions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Input Information** | **Output Information** | **Processing Method** | **Data to be stored** |
| **Pool Tables Management** | Pool Table Information Includes:  • Pool table name.  • Hourly rate for table usage.  • Pool table status (Available, Ordered, Playing).  • Total number of pool tables in the system.  • Detailed status of each table, including usage and reservation information. | • A list of all active pool tables.  • Detailed information for each specific table.  • The current status of each table. | • Display the pool table list directly from the database for easy monitoring.  • Allow adding new tables, update table information.  • Automatically update table status in real time based on usage. | • Pooltable: Stores billiard table's information.  • Fields: cate\_id, table\_id, name, status |
| **Orders Management** | Order Information Includes:  • Bill No (Invoice Number).  • Player or Customer Name.  • Customer's Phone Number.  • Table(s) Used.  • Total Cost.  • Order Status (Playing, Ordered, Finished, Paid, Canceled)  • Order Date & Time.  • Actions: Create, Read, Update | • A detailed list of all current orders.  • Invoice details, including total amount and payment status.  • Search orders by Customer Name or Customer Phone.  • Print detailed invoice information. | • Display a complete list of invoices from the database.  • Allow printing invoices directly from the system in PDF format.  • Quickly search for orders based on search field. | • Orders Table: Stores order information.  • Fields: order\_id, customer\_id, user\_id, total\_cost, order\_date, order\_status |
| **Products Management** | • Product categories.  • Functions to add new products, update prices.  • Detailed information, including price, category, and stock quantity for each product. | • A complete list of all currently available products.  • Detailed information for each specific product.  • Search results for products by name or category. | • Visually display products from the database for easy tracking and management.  • Perform add, update product operations easily and intuitively.  • Provide a quick search feature for products by name or category. | • Products Table: Stores product information.  • Fields: product\_id, name, category\_id, price, unit, quantity |
| **Users Management** | • Avatar (Profile picture).  • Username (Login name).  • Role (User’s role in the system).  • User's Permissions  • Full Name.  • Contact Number.  • Address.  • Date of Birth.  • Hire Date.  • Functions to add, update users.  • Information related to user's role and permissions. | • A complete list of all current users.  • Detailed information for each user.  • Display specific system access permissions for each user. | • Display the users list from the database with full details.  • Functions to add, update user from the system.  • Efficiently and securely update and manage users roles and permissions with full CRUD functions. | • Table: users – Stores users information.  • Fields: user\_id, username, password, fullname, phone, address, hire\_date, birthday, role\_id, image\_path |
| **Customers Management** | • Detailed customer information includes:  • Customer name.  • Customer phone number.  • Contact address.  • Customer's date of birth.  • Total accumulated playtime.  • Functions to add, update, update customer from the system. | • A list of all customers currently in the database.  • Detailed information for each customer.  • Search results for customers based on criteria such as name or phone number. | • Display the customer list from the database for easy management.  • Allow adding new customers, updating customer's information  directly from interface.  • Provide a flexible search function based on criteria such as customer name and phone number, ensuring quick and convenient data retrieval. | • Table: customers – stores customer information.  • Fields: customer\_id, name, phone, total\_playtime, birthday, address |
| **Report Management** | • The date, month, and year to view the report.  • The time period to be analyzed. | • Total revenue by: day or month or year.  • List of invoices within the filtered time period – Revenue chart over time.  • Total number of tables used – Revenue by each billiards table – Chart displaying table usage frequency.  • Total number of products sold – List of best-selling and slowest-selling products – Chart displaying product consumption. | • Retrieve data from database to calculate total revenue – • Use JFreeChart to draw the revenue chart – • Display detailed data for each order.  • Filter booking list by time  • Summarize the number of times each table was used • Display table usage frequency chart.  • Query the number of products sold from orders\_items.  • Draw a best-selling products chart using JFreeChart –  • Display detailed product list. | • Tables and Fields  orders: id, customer\_name,  table\_id, total\_cost, order\_status, order\_date  bookings: id, table\_id. start\_time, end\_time, total\_time  pooltables: id, name, status  cate\_pooltables: id, name, price  orders\_items: order\_id, product\_id, quantity, price |

1. **System database design**
   1. **ERD**



* 1. **List of tables**
     1. **users**

**Description**: This table stores user information, including authentication details and user roles.

**Fields**:

user\_id: Primary Key, type INT (AUTO\_INCREMENT).

username: User's username, type VARCHAR(100), must be unique.

password: User's password, type VARCHAR(100).

fullname: User's full name, type VARCHAR(50).

phone: User's phone number, type VARCHAR(15), must be unique.

address: User's address, type VARCHAR(100), can be NULL.

hire\_date: Date the user was hired, type DATE.

birthday: User's date of birth, type DATE, can be NULL.

role\_id: Foreign Key referencing the roles table, type INT.

image\_path: Path to the user's profile image, type VARCHAR(255), can be NULL.

**3NF Normalization Description:**

**Primary Key**: user\_id

The remaining attributes (username, password, fullname, phone, address, hire\_date, birthday, role\_id, image\_path) all directly depend on the primary key (user\_id).

The table follows 3NF as there are no transitive dependencies or attributes that do not depend on the primary key

* + 1. **roles**

**Description**: This table stores information about user roles in the system.

**Fields**:

role\_id: Primary Key, type INT (AUTO\_INCREMENT).

role\_name: Name of the role, type VARCHAR(50), must be unique.

**3NF Normalization Description:**

**Primary Key**: role\_id

The role\_name attribute directly depends on the primary key role\_id.

This table follows 3NF as there are no transitive dependencies.

* + 1. **permissions**

**Description**: This table stores information about system permissions.

**Fields**:

permission\_id: Primary Key, type INT (AUTO\_INCREMENT).

permission\_name: Name of the permission, type VARCHAR(100), must be unique.

description: Description of the permission, type TEXT, can be NULL.

**3NF Normalization Description:**

**Primary Key**: permission\_id

The remaining attributes (permission\_name, description) all directly depend on the primary key (permission\_id).

The table follows 3NF as there are no transitive dependencies.

* + 1. **role\_permission**

**Description**: This table manages the relationship between roles and permissions, ensuring role-based access control.

**Fields**:

role\_id: Primary Key, type INT, references the roles table.

permission\_id: Primary Key, type INT, references the permissions table.

**3NF Normalization Description:**

**Primary Key**: Composite key (role\_id, permission\_id).

Each attribute in this table directly depends on the composite primary key.

This table is in 3NF as it contains only relationships and does not introduce transitive dependencies.

* + 1. **products**

**Description**: This table stores information about products available in the system.

**Fields**:

product\_id: Primary Key, type INT (AUTO\_INCREMENT).

name: Name of the product, type VARCHAR(255), must be unique.

category\_id: Foreign Key referencing the categories table, type INT.

price: Price of the product, type DOUBLE, can be NULL.

unit: Measurement unit of the product, type VARCHAR(255).

quantity: Available stock quantity, type INT.

**3NF Normalization Description:**

**Primary Key**: product\_id

The remaining attributes (name, category\_id, price, unit, quantity) all directly depend on the primary key (product\_id).

The table follows 3NF as there are no transitive dependencies.

* + 1. **cate\_products**

**Description**: This table stores information about product categories.

**Fields**:

category\_id: Primary Key, type INT (AUTO\_INCREMENT).

category\_name: Name of the category, type VARCHAR(255), must not be NULL.

**3NF Normalization Description:**

**Primary Key**: category\_id

The remaining attribute (category\_name) directly depends on the primary key (category\_id).

The table follows 3NF as there are no transitive dependencies.

* + 1. **orders**

**Description**: This table stores information about customer orders.

**Fields**:

order\_id: Primary Key, type INT (AUTO\_INCREMENT).

customer\_id: Foreign Key referencing the customers table, type INT.

user\_id: Foreign Key referencing the users table (employee handling the order), type INT.

total\_cost: Total cost of the order, type DOUBLE, can be NULL.

order\_date: Date and time when the order was placed, type DATETIME.

order\_status: Status of the order (e.g., Pending, Completed, Canceled), type ENUM.

**3NF Normalization Description:**

**Primary Key**: order\_id

The remaining attributes (customer\_id, user\_id, total\_cost, order\_date, order\_status) all directly depend on the primary key (order\_id).

The table follows 3NF as there are no transitive dependencies.

* + 1. **bookings**

**Description**: This table stores information about table reservations in the system.

**Fields**:

booking\_id: Primary Key, type INT (AUTO\_INCREMENT).

order\_id: Foreign Key referencing the orders table, type INT.

table\_id: Foreign Key referencing the pool tables table, type INT.

start\_time: Timestamp indicating the start of the booking, default is current\_timestamp().

end\_time: Timestamp indicating the end of the booking, can be NULL.

timeplay: Total time played, type DOUBLE, can be NULL.

total: Total cost for the booking, type DOUBLE, can be NULL.

**3NF Normalization Description:**

**Primary Key**: booking\_id

The remaining attributes (order\_id, table\_id, start\_time, end\_time, timeplay, total) all directly depend on the primary key (booking\_id).

The table follows 3NF as there are no transitive dependencies.

* + 1. **orders\_items**

**Description**: This table stores details about the products included in customer orders.

**Fields**:

order\_item\_id: Primary Key, type INT (AUTO\_INCREMENT).

order\_id: Foreign Key referencing the orders table, type INT.

product\_id: Foreign Key referencing the products table, type INT.

quantity: Number of products ordered, type INT.

total: Total cost for this order item, type DOUBLE.

**3NF Normalization Description:**

**Primary Key**: order\_item\_id

The remaining attributes (order\_id, product\_id, quantity, total) all directly depend on the primary key (order\_item\_id).

The table follows 3NF as there are no transitive dependencies.

* + 1. **pooltables**

**Description**: This table stores information about pool tables available in the system.

**Fields**:

table\_id: Primary Key, type INT (AUTO\_INCREMENT).

cate\_id: Foreign Key referencing the categories table, type INT.

name: Name of the table, type VARCHAR(255), must be unique.

status: Status of the table, type ENUM ('Available', 'Ordered', 'Playing'), default is 'Available'.

**3NF Normalization Description:**

**Primary Key**: table\_id

The remaining attributes (cate\_id, name, status) all directly depend on the primary key (table\_id).

The table follows 3NF as there are no transitive dependencies.

* + 1. **cate\_pooltables**

**Description**: This table stores information about product categories.

**Fields**:

id: Primary Key, type INT (AUTO\_INCREMENT).

name: Name of the category, type VARCHAR(255), must not be NULL.

shortName: Abbreviated name of the category, type VARCHAR(10), must not be NULL.

price: Default price associated with the category, type DOUBLE, can be NULL.

**3NF Normalization Description:**

**Primary Key**: id

The remaining attributes (name, shortName, price) all directly depend on the primary key (id).

The table follows 3NF as there are no transitive dependencies.

* + 1. **customers**

**Description**: This table stores information about customers.

**Fields**:

customer\_id: Primary Key, type INT (AUTO\_INCREMENT).

name: Name of the customer, type VARCHAR(50), must not be NULL.

phone: Customer's phone number, type VARCHAR(15), must be unique.

total\_playtime: Total time the customer has played, type DOUBLE, can be NULL.

birthday: Customer's date of birth, type DATE, default is '1990-01-01'.

address: Address of the customer, type VARCHAR(255), can be NULL.

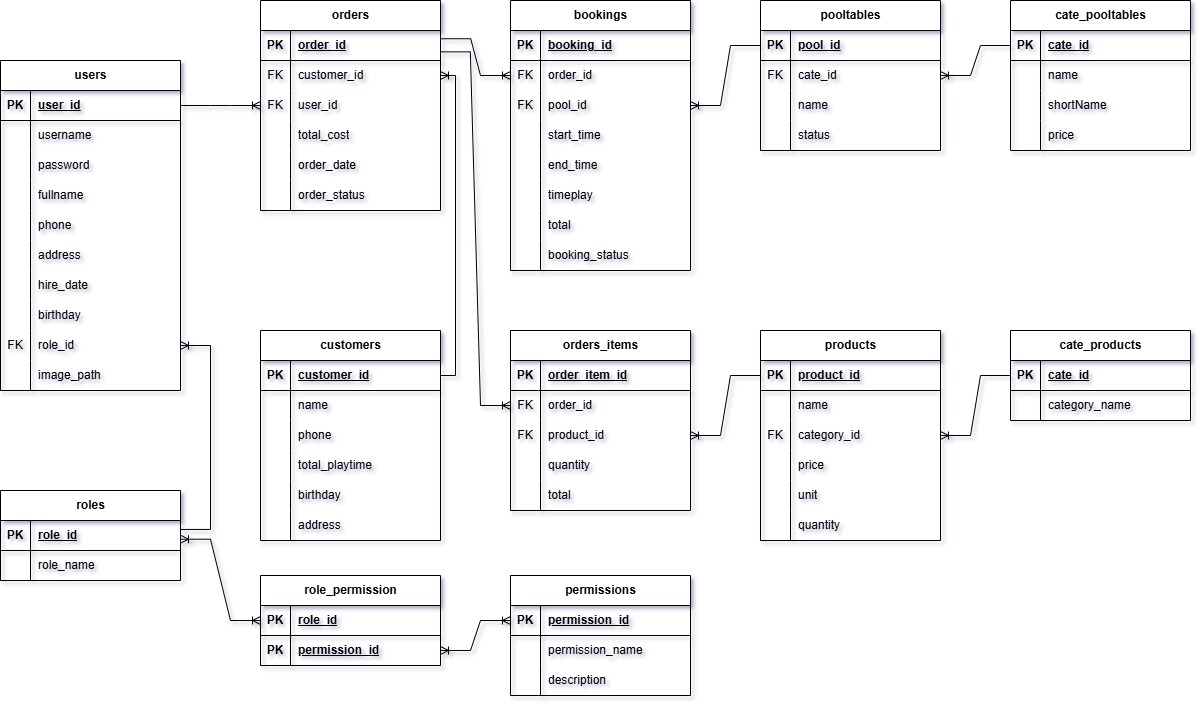
**3NF Normalization Description:**

**Primary Key**: customer\_id

The remaining attributes (name, phone, total\_playtime, birthday, address) all directly depend on the primary key (customer\_id).

The table follows 3NF as there are no transitive dependencies.

* 1. **Relationship Diagram**



**Relationships:**

* users ↔ orders: Relationship between users and orders. Each order is created by one user (1:N).
* customers ↔ orders: Relationship between customers and orders. Each order belongs to one customer (1:N).
* orders ↔ order\_items: Each order can contain multiple items (1:N).
* products ↔ order\_items: Each product can appear in multiple order items (1:N).
* bookings ↔ orders: Each booking is linked to one order (1:1).
* bookings ↔ pooltables: Each booking is assigned to one table (1:1).
* roles ↔ users: Each user is assigned one role (1:N).
* roles ↔ permissions: Each role can have multiple permissions (1:N).

* 1. **Sitemap**

****

1. **System functions design**
   1. **Pool tables management functionality**
      1. **Detailed functionality**
      2. **Input information**

* List of billiard tables:
* Table Name: Name of each billiard table in the system (e.g., "STD 1", "DLX 10").
* Table Status: "Available" (Free), "Playing" (In use), "Ordered" (Reserved).
* Table Type: Classification of tables into groups like STD, DLX, VIP.
* Related Order Information: If a pool table is occupied, the system will load the order associated with this pool table.
* Category Pooltable:
* Category Name: Name of the Pooltable Category
* Category Price: Price per Hour of each Pooltable Category
  + 1. **Output information**
* List of Active Billiard Tables: Visually display the status of tables (colors indicating status, as shown in the interface image: Playing is Red, Ordered is Orange, and Available is Green).
* When hovering over each table, show “Order Info” and “Pool Info” button.
* The “Order Info” button will be used to show the Order associated with this pool table (if any), or show a pane which display status information of this pool table, and allow user to Create new Order with this pool table, or Add this Pool table to an existed Order.
* The “Pool Info” button will be used to show this pool table’s detailed information, including this pool table’s name and category.
  + 1. **Processing method**
* Retrieve Data from Database to a Billiard Table List:
* Data Retrieval:
* The system queries the database to fetch all pool tables, including their current status (Available, Ordered, Playing).
* Each table’s associated details, such as name, category, and any current orders, are also retrieved.
* Display on Billiard Table Page:
* The retrieved pool tables are displayed on the Billiard Table page.
* Each table is visually represented with a color-coded status indicator:
* Green: Available
* Orange: Ordered
* Red: Playing
* Display Billiard Table Information on the Screen:
* Visual Icons for Table Status:
* Each table will have an icon that reflects its current status, providing a quick visual reference for Users.
* Hover Interaction:
* When Users hover over a table icon, two buttons appear: the "Order Info" button and the "Pool Info" button.
* *Order Info Button:*
* When clicked, this button will show the order associated with the pool table, if any exists.
* If there is no existing order, it will display a pane with status information about the pool table.
* This pane will also provide options for the User to create a new order with this pool table or to add this pool table to an existing order.
* *Pool Info Button:*
* When clicked, this button will display detailed information about the selected pool table.
* The information includes the pool table’s name, category, and any other relevant details.
* Table Order Processing:
* Creating or Updating Pool Table Status:
* User Actions:
* Users can create new pool tables or update the status of existing tables through the management interface.
* Updating Table Status:
* When a Customer Orders a Table:
* The system updates the table status from "Available" to "Ordered."
* If an associated order exists, it is linked to the table; otherwise, a new order can be created.
* When a Customer Starts Playing:
* The system changes the status from "Ordered" to "Playing," indicating active use.
* When a Customer Finishes Playing:
* The status is updated from "Playing" back to "Available."
* A billiard bill is generated based on the duration of play and any applicable charges, ensuring that all transactions are recorded.
  + 1. **Data to be stored**
* table\_id: ID of each pool table
* cate\_id: pool table category which pool table is belong to
* name: Name of the table
* status: status of pool table
  + 1. **UI/UX design**
    - Billiard Table List Page:
* Display a list of billiard tables with different statuses (Available, Ordered, Playing).
* Each table is represented as a visual billiard table icon.
* When hovering over a pool table, show detailed information about the table and the order associated with this table (if any).
  + - Booking Pool Table Page:
* Show a form to enter customer information, phone number, and user details.
* Show the list of table that customer order / playing in / finish playing in or cancel.
* Allow adding new table to order / playing or updating information.
* A "Finish Order" button for customer payment.
  + - Reservation Form:
* Allow user to select a pool table from the list.
* Enter customer information and any special notes (if applicable).
* "Confirm" button to save reservation details into the system.
  + 1. **Activity diagram (Flowchart)**
    2. **Overview flowchart**

*Flowchart 01: Pool tables management flowchart (page 38)*

* + 1. **Detailed steps**

1. User Accesses Billiard Table List:

The system retrieves the list of tables from the database and displays it on the interface.

1. Display Table Details:

When the user clicks on a table, a detail page is shown with complete information about that table.

1. Table Reservation:

Users can select a table, enter customer information, the system checks the input information and saves it to the bookings table.

1. Process Table Reservation:

The system displays a success message and saves the reservation order.

* 1. **Orders management functionality** 
     1. **Detailed functionality**
     2. **Input information**
* Customer Name: Name of the customer placing the order (e.g., "Nguyen Van A").
* Phone Number: Customer's contact phone number.
* Table Name: Billiard table(s) that are being occupied by this customer.
* Start Time: Time when the customer start playing or ordering on a table.
* End Time: Time when the playing section finish
* Table Rental Price: Table price, calculated per hour.
* Products: List of ordered items (drinks, snacks, etc.).
* User Name: Name of the serving user.
* Total Cost: Total cost calculated based on pool tables occupied, total time-play, pool table's price per hour and items or products that customer ordered
  + 1. **Output information**
* Order List: Display a list of orders including customer name, phone number, occupied billiard table(s), status.
* Order Details: Show detailed information when requested by the user.
* Result of Adding, Updating Orders: Handle add, update, request, save data to the database, and show a success or error (if input data is not valid or if operation failed) notification on the screen.
  + 1. **Processing method**
* Add New Order:
* User click on "Add New Order" button and the system will show this Add Order page.
* Enter customer information, occupied pool table, playing time, and ordered products.
* The system then validate the input data; if valid, the order will be saved to the database. If no customer information specified, a profile "Guest" will be added by default.
* Display a success message for updated information.
* Update Order:
* Admin selects an order to update.
* The system displays that order's current information.
* Admin updates new information and submit the update request.
* The system validate the input data; if valid, new data will be updated and save to the database.
* Display a success message for updating.
  + 1. **Data to be stored**
* order\_id: ID of each order
* customer\_id: ID of customer
* user\_id: ID of user who create order
* total\_cost: Total cost of the order
* order\_date: Date and time when the order was created
* order\_status: Status of order
  + 1. **UI/UX design**
* Order Interface:
* Retrieve all orders in the database and placed inside a table.
* Show a "Filter Type" button, which user can choose:
* Filter by Date Range: User can choose start date - end date to show order occupied between this date range.
* Filter by Pool Table's Category: User can choose which pool table's category to filter.
* Filter by Order's Status: User can choose order status to filter like Playing, Finished, Paid or Canceled
* Show an "Search by Name / Phone" text field to search order by customer name or phone.
* Show an "Add New Order" button to add new order.
* Show an "Refresh" button to refresh the order table.
* On each order, there will be a button Print Bill placed. This button can only be accessed when that order's status is Finished or Paid. When this button is clicked, the program will show a Bill form. Order information, customer information, pool table(s) occupied and ordered products will be show in this Bill form. User can click "Print Bill" button to create a PDF bill for later usage (the system can use this PDF bill to print from the printer and give it to the customer). Or, user can click "Pay Order" button in case customer pay for this order. This action will make this order's status change to Paid.
* Add New Order Interface:
* This will be displayed when user click on "Add New Order" button on Order interface.
* Show a Form to enter customer information (name and phone number), add billiard table(s) and order products.
* If a Customer information has already existed in the system, then this customer information will be saved automatically on chosen. If not existed, user can ask for customer information and click button "Add New Customer".
* If customer don't want to reveal their identity, then a default profile "Guest" will be added by default.
* Order Details Interface:
* This will be displayed when user double-clicks on an order or clicks once and then presses 'Enter' on the selected order, lay inside Order interface.
* Display the order's current information, including pool table(s) occupied, ordered products, customer information and serving user information
* Show an "Add New Booking" button, which is used to add new pool table to order or play on
* Show an "Add New Item" button, which is used to order product on user request, including: foods, drinks, rent cues, etc…
* Show an "Finish this Order" button to finish playing / finish ordering on all tables that the customer has occupied, and then finish the order.
* On each pool table occupied (shown in the Booking table), buttons will be display based on that pool table's status. If "Playing", there will be a finish button beside. If "Ordered", there will be two buttons shown: a "Start Playing" button to start playing on this pool table, and a "Cancel" button to cancel order.
* On each ordered products (shown in Order Items table), there will be two buttons shown: a "Update" button to update that ordered product's information, including ordered product's name and ordered quantity, and a "Remove" button to remove ordered product from order.
  + 1. **Activity diagram (Flowchart)**
    2. **Overview flowchart**

*Flowchart 02: Orders management flowchart (page 39)*

* + 1. **Detailed steps**
* Adding a New Order:
* Admin enters order information into the new order form.
* The system validate the data; if valid, the program will save data to the database and display a success notification on the screen.
* Updating an Order:
* Admin selects an existing order, then the program loads that order's current data.
* Admin updates new information, and the program checks for validity before saving the new information.
  1. **Products management functionality** 
     1. **Detailed functionality**
     2. **Input information**
* Product Name: The name of the product or service offered in the system (e.g., Billiard Cues, Drinks, Snacks, Board Games, etc.).
* Quantity: The number of products currently available in stock.
* Price: The selling or rental price of the product.
* Unit: The measurement unit for the product (e.g., Piece, Glass, Pack, etc.).
* Product Category: The category to which the product belongs (e.g., Cues for Rent, Drinks, Food, Board Games, etc.).
* User Request: User's request / action to add new / update Product Category, or add new / stock up / update product
  + 1. **Output information**
* Product List: Displays all products available in the system, categorized accordingly.
* Product Details: Shows detailed information for each product upon user request.
* Result of Adding, Updating, or Deleting Products: Notifies the admin about the outcome of the operations.
  + 1. **Processing method**
* Adding New Product:
* User accesses the product management page.
* User enter product information, including name, quantity, price, unit, category, and image.
* The system validates the data; if valid, data will be saved to the database.
* Displays a success message for the product addition.
* Updating Product:
* User selects a product to update.
* The system displays the current information of that product.
* User updates the necessary information and submits the request.
* The system validates and updates the data in the database.
* Displays a success message for the update.
  + 1. **Data to be stored**
* id: Unique ID of the product.
* name: Name of the product.
* quantity: Stock quantity.
* price: Product price.
* unit: Measurement unit.
* category\_id: ID of the product category.
  + 1. **UI/UX design**
* Product List Interface:
* An "Add Category" button to create a new product category.
* A "Search by name" text field to search product by product name.
* Display tables of products categorized by Product Category. Each table will contain only the products that belong to its corresponding category. For example, the Foods table will exclusively list Food products.
* Information columns include: Product Name, Quantity, Price, Unit.
* Show action buttons:
* For Product Categories: we have "Add New Category" button, which show an Add New Category page on click. This will be used to add new Product Category and a new table corresponding to it. We also have "Edit Category" button to edit / update category information.
* For each Product: we have "Stock Up" button which stock up the quantity of a product, and "Edit Product" button to update that product's information.
* Add Product Interface:
* Show a form to enter product information: Name, Quantity, Price, Unit, Category.
* A "Save Product" button to add the product to the tables and save new data to the database.
* Edit Product Interface:
* The program will load the product's current information from database into a form.
* User can enter new information / update existed field in the form
* An "Update" button to submit new changes.
  + 1. **Activity diagram (Flowchart)**
    2. **Overview flowchart**

*Flowchart 03: Products management flowchart (page 40)*

* + 1. **Detailed steps**
* Display the Products:
* Accessing the Products Page: User navigates to the Products page, triggering the system to load product categories.
* Retrieving Categories: The system queries the database for all Product Categories and displays them as separate tables.
* Loading Products: For each category, the system fetches the corresponding product data from the products table, displaying details such as Product Name, Quantity, Price, and Unit.
* User Interaction: Users can easily navigate between products, view details, or edit information as needed. A search function allows Users to find specific products quickly.
* Handle Add New Product Category / Update Product Category:
* Accessing Category Management: User clicks the "Add Category" button to create a new product category.
* Entering Category Information: User provides necessary details and submits the form.
* Validation and Saving: The system validates the input; if valid, it saves the new category to the database and confirms the addition.
* Handle Add New Product / Update Product:
* Adding a New Product:
* Accessing the Form: User clicks "Add Product" to open the input form.
* Entering Product Details: User fills in fields like Product Name, Quantity, Price, and Category.
* Saving the Product: The system validates the data and saves it to the products table, displaying a success message.
* Editing an Existing Product:
* Selecting the Product: User chooses a product to edit, prompting the system to load its current information.
* Updating Information: User modifies the necessary fields and submits the changes.
* Validation and Update: The system checks the updated data; if valid, it updates the products table and confirms the edit.
  1. **Users management functionality** 
     1. **Detailed functionality**
     2. **Input information**
* Avatar: Profile image of user.
* Username: user account login name.
* Role: Position in the system (Admin, Manager, Receptionist, Warehouse, etc.).
* Full Name: Full name of user.
* Phone: Contact number.
* Birthday: Date of birth.
* Address: Current address.
* Hire Date: Date of employment start.
  + 1. **Output information**
* Users List: Displays all users in the system, including avatar, username, role, full name, phone, birthday, address, hire date.
* User Details: Shows detailed information when a user is selected.
* Operation Results: Shows success or error messages for add, update actions.
  + 1. **Processing method**
* Add User:
* Admin accesses the user list page.
* Fills in user info (avatar, username, role, full name, phone, birthday, address, hire date).
* System validates the input and saves to database if valid.
* Displays a success message.
* Edit User:
* Admin selects a user to edit.
* System loads current information.
* Admin updates necessary fields and submits.
* System validates and saves the updated info.
* Displays a success message.
  + 1. **Data to be stored**
* id: Unique user ID
* avatar: Profile image
* username: Login name
* role: Role/position
* full\_name: Full name
* phone: Phone number
* birthday: Date of birth
* address: Address
* hire\_date: Hire date
  + 1. **UI/UX design**
    2. **Activity diagram (Flowchart)**
    3. **Overview flowchart**

*Flowchart 04: Users management flowchart (page 41)*

* + 1. **Detailed steps**
* Add new user
* Step 1: Admin accesses the user list page.
* Step 2: Admin fills out the form with information (avatar, username, role, full name, phone, birthday, address, hire date).
* Step 3: The system validates the input:
* If any field is missing or has an invalid format → show error message.
* If all data is valid → proceed to the next step.
* Step 4: Save the data into the users table.
* Step 5: Show the message: "User added successfully."
* Edit user
* Step 1: Admin selects user to edit.
* Step 2: The system displays the current information.
* Step 3: Admin modifies the information and submits the update request.
* Step 4: The system validates the data and updates it in the database.
* Step 5: Show the message: "User information updated successfully."
  1. **Customers management functionality** 
     1. **Detailed functionality**
     2. **Input information**
* Customer Name: Full name of the customer (e.g., “Nguyen Van A”).
* Phone Number: Contact number of the customer.
* Total Playtime: Total number of hours the customer has played billiards in the system.
  + 1. **Output information**
* Customer List: Displays a list of all customers, including ID, name, phone number, and total playtime.
* Customer Details: Shows detailed information when a customer is selected.
* Add, Update Results: Shows success or error messages after each operation.
  + 1. **Processing method**
* Add New Customer:
* Admin accesses the customer list page.
* Fills in the customer name and phone number.
* The system validates the data and, if valid, saves it to the database.
* A success message is displayed.
  + 1. **Data to be stored**
* id: Unique customer ID
* name: Customer name
* phone\_number: Phone number
* total\_playtime: Total billiard playtime
* birthday: date of birth of customer
* address: address of customer
  + 1. **UI/UX design**
* Customer List Interface
* Displays the list of customers including: ID, Name, Phone Number, and Total Playtime.
* Search bar to filter customers by name.
* "Add Customer" button to create a new entry.
* Add/Edit Customer Interface
* Form to input name and phone number.
* Display of total playtime.
* "Save" button to confirm adding or editing.
* "Cancel" button to exit the form.
  + 1. **Activity diagram (Flowchart)**
    2. **Overview flowchart**

*Flowchart 05: Customers management flowchart (page 42)*

* + 1. **Detailed steps**
* Add New Customer
* Step 1: Admin accesses the customer list page.
* Step 2: Admin fills in the form with customer name and phone number.
* Step 3: The system validates the input:
* If any field is missing or invalid → show error message.
* If valid → proceed.
* Step 4: Save the data into the customers table.
* Step 5: Show message: "Customer added successfully."
* Edit Customer
* Step 1: Admin selects the customer to edit.
* Step 2: The system displays the current information.
* Step 3: Admin updates the necessary fields.
* Step 4: The system validates the input and updates the database.
* Step 5: Show message: "Customer information updated successfully."
  1. **Report management functionality** 
     1. **Detailed functionality**
     2. **Input information**
* Date to view the report
* Time period for statistics
  + 1. **Output information**
* Total revenue by day/month/year
* List of invoices within the selected time frame
* Revenue chart over time
* Total number of tables used
* Revenue per billiard table
* Chart displaying table usage frequency
* Total number of products sold
* List of best-selling and slowest-selling products
* Chart visualizing product consumption
  + 1. **Processing method**
* Query MySQL to calculate total revenue and use JFreeChart to generate revenue charts
* Display detailed order data
* Filter the booking list based on the selected time range
* Aggregate the number of times each table has been used
* Display a chart showing table usage frequency
* Query the number of products sold from order\_items
* Generate best-selling product charts using JFreeChart
* Display a detailed product sales list
  + 1. **Required Data Storage**
* Order List (Orders):
* Order ID: (id from the orders table)
* Customer Name: (customer\_name from orders)
* Table ID: (table\_id from orders)
* Total Cost: (total\_cost from orders)
* Order Status: (order\_status from orders)
* Order Date: (order\_date from orders)
* Order Items List:
* Product ID: (product\_id from orders\_items)
* Quantity: (quantity from orders\_items)
* Price: (price from orders\_items)
* Booking List (Bookings):
* Booking ID: (id from bookings)
* Table ID: (table\_id from bookings)
* Start Time: (start\_time from bookings)
* End Time: (end\_time from bookings)
* Total Booking Cost: (total\_cost from bookings)
* 3. Pool Table List (Pool Tables):
* Pool Table ID: (id from pooltables)
* Pool Table Name: (name from pooltables)
* Pool Table Status: (status from pooltables)
* 4. Pool Table Category List (Cate Pool Tables):
* Pool Table Category ID: (id from cate\_pooltables)
* Pool Table Category Name: (name from cate\_pooltables)
  + 1. **UI/UX design**
* Navigation Bar:
* Tabs: Customers, Orders, Products for quick access to different sections
* Filter by: (Date/Month/Year) using DatePicker/ComboBox to select the time range
* Apply Filter button to apply selected filters
* Summary Information:
* Total Revenue: Displays total revenue for the selected time period
* Total Orders: Displays total number of orders within the selected time frame
* Revenue Data Table:
* Display revenue by table category (e.g., Deluxe Pool, Standard Pool)
* Date column showing revenue per day/month
* Charts:
* Bar Chart: Displays revenue over time (day/month/year)
* Pie Charts:
* Revenue by Table Group: Shows revenue distribution across different table groups
* Revenue by Product Group: Displays revenue breakdown by product categories
  + 1. **Activity diagram (Flowchart)**
    2. **Overview flowchart**

*Flowchart 06: Report management flowchart (page 43)*

* + 1. **Detailed steps**
* Step 1: User selects the time period
* The user selects a date, month, or year on the interface using a DatePicker or ComboBox.
* Step 2: User clicks "Apply Filter"
* The system captures the event and retrieves the selected date/month/year.
* Step 3: Query data from MySQL
* The system executes SQL queries to fetch data from the following tables:
* Orders: Retrieve total revenue and the number of orders.
* Bookings: Check table reservations and usage time.
* PoolTables: Check table groups and usage status.
* Products: Get sales statistics for each product.
* Step 4: Calculate key statistics
* Total revenue = Sum of all order values within the selected period.
* Total orders = Total number of recorded orders.
* Total table usage time = Sum of playtime from bookings for each table.
* Total product sales = Sum of sold quantities from order\_items.
* Step 5: Generate charts using JFreeChart
* Bar chart: Displays revenue by day/week/month/year.
* Pie chart:
* Revenue by table group.
* Revenue by product category.
* Line chart: Revenue trends over time.
* Step 6: Display summary report
* Total revenue
* Total number of orders
* Total table usage time
* Step 7: Display detailed data tables
* Revenue by table group
* Invoice list (invoice number, customer, total amount)
* Product sales list
* Step 8: Show report charts
* Revenue over time
* Table usage by group
* Product sales quantity

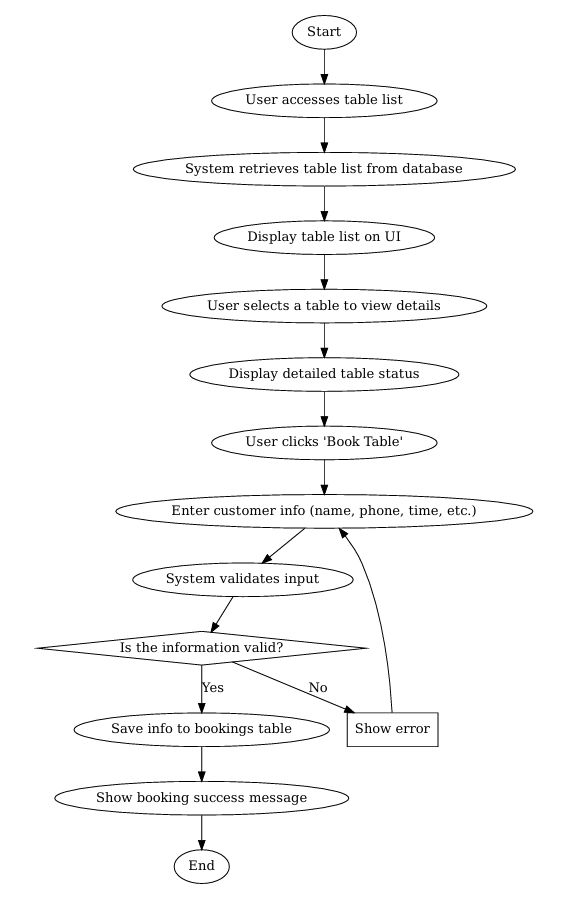
1. **Validation checklists**

|  |  |  |
| --- | --- | --- |
| **Table** | **Column** | **Validation** |
| **users** | username | Required, Unique, String, max length 100 characters |
| password | Required, String, max length 100 characters |
| fullname | Required, String, max length 50 characters |
| phone | Required, String, valid phone number format |
| address | Nullable, String, max length 100 characters |
| hire\_date | Required, Valid date format |
| birthday | Nullable, Valid date format |
| role\_id | Required, Integer, foreign key references roles |
| image\_path | Required, String, max length 100 characters |
| **customers** | name | Required, String, max length 50 characters |
| phone | Required, String, valid phone number format |
| total\_playtime | Nullable, Double, value ≥ 0 |
| birthday | Nullable, Valid date format |
| address | Nullable, String, max length 100 characters |
| **products** | name | Required, String, max length 255 characters |
| category\_id | Required, Integer, foreign key references category |
| price | Required, Double, value > 0 |
| unit | Required, String, max length 255 characters |
| quantity | Nullable, Integer, value ≥ 0 |
| **orders** | customer\_id | Required, Integer, foreign key references customers |
| user\_id | Required, Integer, foreign key references users |
| total\_cost | Nullable, Double, value ≥ 0 |
| order\_date | Required, Valid datetime format |
| order\_status | Required, Enum ('Order','Playing','Finished','Paid','Canceled') |
| **orders\_items** | order\_id | Required, Integer, foreign key references orders |
| product\_id | Required, Integer, foreign key references products |
| quantity | Required, Integer, value > 0 |
| total | Nullable, Double, auto-calculated from quantity and product price |
| **bookings** | order\_id | Required, Integer, foreign key references orders |
| table\_id | Required, Integer, foreign key references pooltables |
| start\_time | Required, Valid timestamp |
| end\_time | Nullable, Valid timestamp |
| timeplay | Nullable, Double, value ≥ 0 |
| total | Nullable, Double, value ≥ 0 |
| booking\_status | Required, Enum ('Order','Playing','Finish','Canceled') |
| **pooltables** | cate\_id | Required, Integer, foreign key references cate\_pooltables |
| name | Required, String, max length 255 characters |
| status | Required, Enum ('Available','Ordered','Playing') |
| **category** | category\_name | Required, String, max length 255 characters |
| **cate\_pooltables** | name | Required, Double, value > 0 |
| shortName | Required, String, max length 10 characters |
| price | Required, Double, value > 0 |
| **roles** | role\_name | Required, String, max length 50 characters |
| **permissions** | permission\_name | Required, String, max length 100 characters |
| description | Required, String, max length 100 characters |

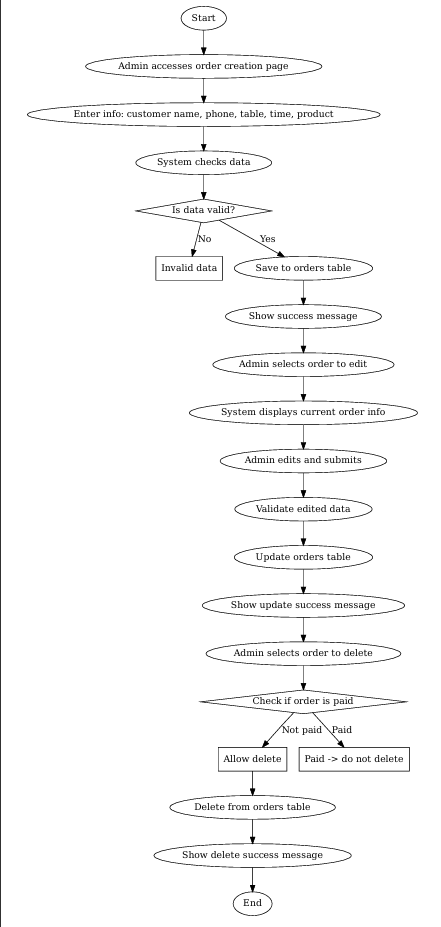
**TASK SHEET**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Start time** | **Finish time** | **Implemented by** | **Evaluation (%)** |
| **Feasibility study phase** |  |  |  |  |
| Determine the overall objective of the project. | 10/01/2025 | 13/01/2025 | Hieu Nguyen Van | 25% |
| Determine the overall objective of the project. | 10/01/2025 | 13/01/2025 | Quan Luu Minh | 25% |
| Determine the overall objective of the project. | 10/01/2025 | 13/01/2025 | Manh Nguyen Duc | 25% |
| Determine the overall objective of the project. | 10/01/2025 | 13/01/2025 | Long Nguyen Viet | 25% |
| Evaluate human resources. | 10/01/2025 | 13/01/2025 | Hieu Nguyen Van | 100% |
| Identify technologies and frameworks that can be used. | 10/01/2025 | 13/01/2025 | Hieu Nguyen Van | 25% |
| Identify technologies and frameworks that can be used. | 10/01/2025 | 13/01/2025 | Quan Luu Minh | 25% |
| Identify technologies and frameworks that can be used. | 10/01/2025 | 13/01/2025 | Manh Nguyen Duc | 25% |
| Identify technologies and frameworks that can be used. | 10/01/2025 | 13/01/2025 | Long Nguyen Viet | 25% |
| Analyze the integration with external technologies. | 10/01/2025 | 13/01/2025 | Hieu Nguyen Van | 25% |
| Analyze the integration with external technologies. | 10/01/2025 | 13/01/2025 | Quan Luu Minh | 25% |
| Analyze the integration with external technologies. | 10/01/2025 | 13/01/2025 | Manh Nguyen Duc | 25% |
| Analyze the integration with external technologies. | 10/01/2025 | 13/01/2025 | Long Nguyen Viet | 25% |
| **Requirement Analysis phase** |  |  |  |  |
| Clearly define the functions and features needed in the system. | 10/01/2025 | 13/01/2025 | Hieu Nguyen Van | 50% |
| Clearly define the functions and features needed in the system. | 10/01/2025 | 13/01/2025 | Long Nguyen Viet | 50% |
| Analyze database requirements and necessary tables. | 10/01/2025 | 13/01/2025 | Hieu Nguyen Van | 50% |
| Analyze database requirements and necessary tables. | 10/01/2025 | 13/01/2025 | Long Nguyen Viet | 50% |
| **Design phase** |  |  |  |  |
| Design the database architecture. | 14/01/2025 | 17/01/2025 | Hieu Nguyen Van | 50% |
| Design the database architecture. | 14/01/2025 | 17/01/2025 | Long Nguyen Viet | 50% |
| Design the system architecture. | 14/01/2025 | 17/01/2025 | Quan Luu Minh | 50% |
| Design the system architecture. | 14/01/2025 | 17/01/2025 | Manh Nguyen Duc | 50% |
| Create the ERD (Entity-Relationship Diagram). | 14/01/2025 | 17/01/2025 | Long Nguyen Viet | 100% |
| Design activity diagrams for the main functions. | 14/01/2025 | 17/01/2025 | Hieu Nguyen Van | 50% |
| Design activity diagrams for the main functions. | 14/01/2025 | 17/01/2025 | Long Nguyen Viet | 50% |
| Design UI/UX mockups for the main pages. | 14/01/2025 | 10/02/2025 | Quan Luu Minh | 50% |
| Design UI/UX mockups for the main pages. | 14/01/2025 | 10/02/2025 | Manh Nguyen Duc | 50% |
| Design the API if integrating with external services. | 14/01/2025 | 10/02/2025 | Hieu Nguyen Van | 100% |
| **Development phase** |  |  |  |  |
| Build the database and tables based on the designed architecture. | 18/01/2025 | 10/02/2025 | Hieu Nguyen Van | 25% |
| Build the database and tables based on the designed architecture. | 18/01/2025 | 10/02/2025 | Quan Luu Minh | 25% |
| Build the database and tables based on the designed architecture. | 18/01/2025 | 10/02/2025 | Manh Nguyen Duc | 25% |
| Build the database and tables based on the designed architecture. | 18/01/2025 | 10/02/2025 | Long Nguyen Viet | 25% |
| Develop the users management functions. | 11/02/2025 | 10/03/2025 | Hieu Nguyen Van | 100% |
| Develop the roles and permissions management functions. | 11/02/2025 | 10/03/2025 | Hieu Nguyen Van | 100% |
| Develop the pool tables, pool categories management functions. | 11/02/2025 | 12/03/2025 | Quan Luu Minh | 30% |
| Develop the pool tables, pool categories management functions. | 11/02/2025 | 12/03/2025 | Manh Nguyen Duc | 70% |
| Develop the orders management functions. | 11/02/2025 | 10/03/2025 | Manh Nguyen Duc | 55% |
| Develop the orders management functions. | 11/02/2025 | 10/03/2025 | Long Nguyen Viet | 45% |
| Develop the products, product categories management functions. | 11/02/2025 | 10/03/2025 | Hieu Nguyen Van | 100% |
| Develop the customers management functions. | 11/02/2025 | 12/03/2025 | Quan Luu Minh | 40% |
| Develop the customers management functions. | 11/02/2025 | 12/03/2025 | Manh Nguyen Duc | 60% |
| Develop the report, statistic management functions. | 11/02/2025 | 12/03/2025 | Long Nguyen Viet | 100% |
| Write seeders for the database to add sample data. | 20/01/2025 | 10/03/2025 | Hieu Nguyen Van | 25% |
| Write seeders for the database to add sample data. | 20/01/2025 | 10/03/2025 | Quan Luu Minh | 25% |
| Write seeders for the database to add sample data. | 20/01/2025 | 10/03/2025 | Manh Nguyen Duc | 25% |
| Write seeders for the database to add sample data. | 20/01/2025 | 10/03/2025 | Long Nguyen Viet | 25% |
| Optimize the frontend with CSS for the UI/UX. | 11/02/2025 | 12/03/2025 | Hieu Nguyen Van | 25% |
| Optimize the frontend with CSS for the UI/UX. | 11/02/2025 | 12/03/2025 | Quan Luu Minh | 25% |
| Optimize the frontend with CSS for the UI/UX. | 11/02/2025 | 12/03/2025 | Manh Nguyen Duc | 25% |
| Optimize the frontend with CSS for the UI/UX. | 11/02/2025 | 12/03/2025 | Long Nguyen Viet | 25% |
| **Testing phase** |  |  |  |  |
| Test the functionality for the main features | 11/02/2025 | 12/03/2025 | Hieu Nguyen Van | 25% |
| Test the functionality for the main features | 11/02/2025 | 12/03/2025 | Quan Luu Minh | 25% |
| Test the functionality for the main features | 11/02/2025 | 12/03/2025 | Manh Nguyen Duc | 25% |
| Test the functionality for the main features | 11/02/2025 | 12/03/2025 | Long Nguyen Viet | 25% |
| Fix the issues discovered during testing. | 11/02/2025 | 12/03/2025 | Hieu Nguyen Van | 25% |
| Fix the issues discovered during testing. | 11/02/2025 | 12/03/2025 | Quan Luu Minh | 25% |
| Fix the issues discovered during testing. | 11/02/2025 | 12/03/2025 | Manh Nguyen Duc | 25% |
| Fix the issues discovered during testing. | 11/02/2025 | 12/03/2025 | Long Nguyen Viet | 25% |

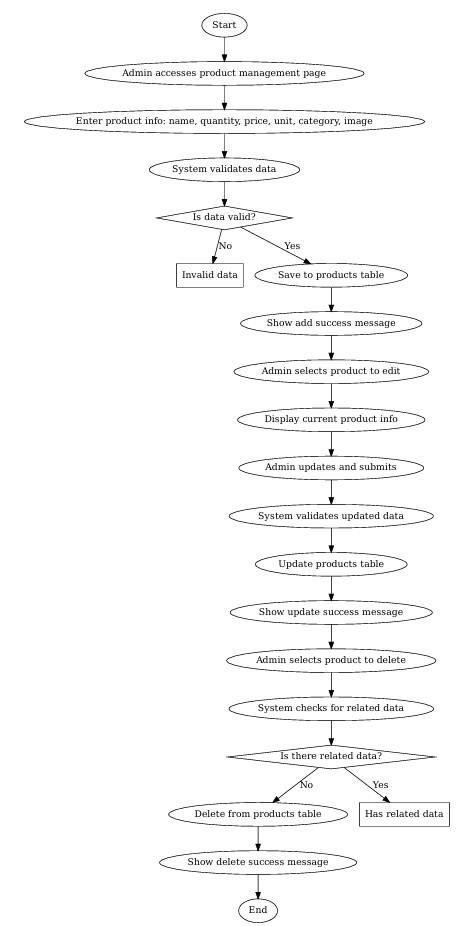
**Appendices: ACTIVITY DIAGRAM OF FUNCTIONALITIES**



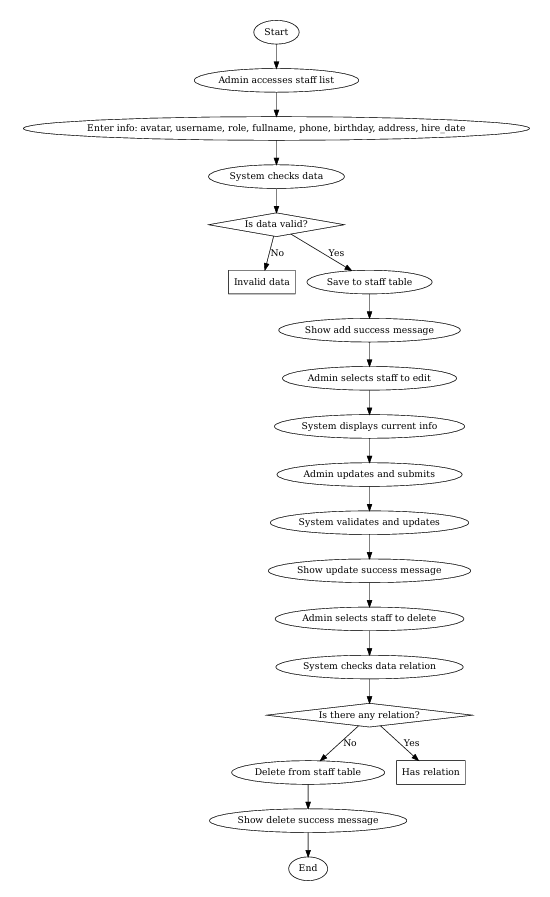
*Flowchart 01: Pool tables management flowchart*



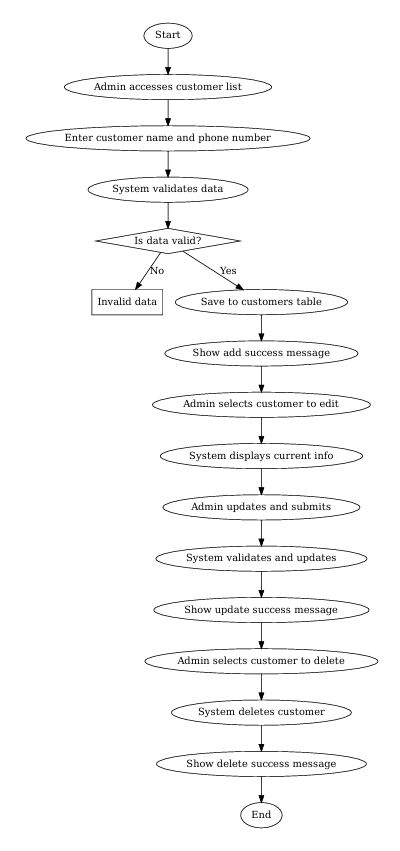
*Flowchart 02: Orders management flowchart*



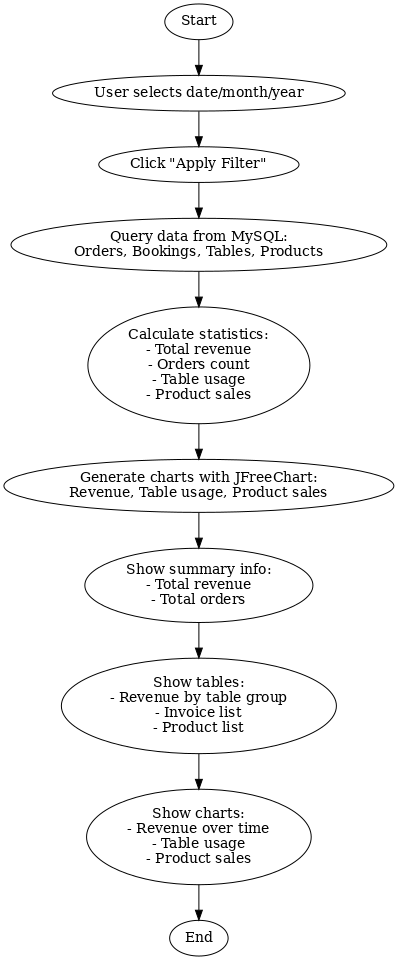
*Flowchart 03: Products management flowchart*



*Flowchart 04: Users management flowchart*



*Flowchart 05: Customers management flowchart*

**

*Flowchart 06: Report management flowchart*