# Design Document: Basic Rental System

---

## 1. Introduction

This document describes the design for the Basic Rental System extension to the inventory management application. The system enables admin/staff users to manage rentals of inventory items to customers, ensure item availability, and maintain accurate records of rental transactions and customer information.

---

## 2. System Architecture

- \*\*Architecture Style:\*\* Web-based, client-server (PHP backend, HTML/JavaScript frontend)

- \*\*Database:\*\* Relational (MySQL or compatible)

- \*\*Users:\*\* Admin/Staff (authenticated)

---

## 3. Data Model

### 3.1 Entity Relationship Diagram (ERD) \*(text description)\*

- \*\*Customer\*\* (1) — (M) \*\*Rental\*\* (M) — (1) \*\*Item\*\*

### 3.2 Tables

#### `customers`

- customer\_id (PK, INT, AUTO\_INCREMENT)

- name (VARCHAR)

- contact (VARCHAR)

#### `items`

- item\_id (PK, INT, AUTO\_INCREMENT)

- name (VARCHAR)

- status (ENUM: 'available', 'rented')

- ... (other existing fields)

#### `rentals`

- rental\_id (PK, INT, AUTO\_INCREMENT)

- item\_id (FK → items.item\_id)

- customer\_id (FK → customers.customer\_id)

- start\_date (DATE)

- expected\_return\_date (DATE)

- actual\_return\_date (DATE, NULLABLE)

- status (ENUM: 'active', 'returned', 'overdue')

---

## 4. Main Components

### 4.1 Backend (PHP)

- \*\*Rental Controller\*\*

- Create rental (with validation)

- Return rental (with date/status handling)

- List/filter rentals

- \*\*Customer Controller\*\*

- CRUD operations for customers

- \*\*Item Controller\*\*

- Update item status on rental/return

### 4.2 Frontend (HTML/JavaScript)

- \*\*Rental Management Pages\*\*

- Create Rental form (dropdowns for item and customer, date pickers)

- Active Rentals list (with "Return" action)

- Rental History (with filters)

- \*\*Customer Management\*\*

- Add/Edit/View/Delete customer forms

- Customer search/select in rental form

- \*\*Feedback UI\*\*

- Inline validation, error displays, and confirmation messages

---

## 5. Core Flows

### 5.1 Rent Out an Item

1. User navigates to "Create Rental"

2. System displays available items and customers

3. User selects item, customer, start and expected return dates

4. Form submitted to backend

5. Backend validates:

- Item is available

- Dates are valid

6. On success:

- Rental created, item status set to "rented"

- Confirmation shown

### 5.2 Return an Item

1. User goes to "Active Rentals"

2. User selects a rental to return

3. Backend updates rental (sets status to "returned", records actual return date)

4. Item status reset to "available"

5. Confirmation shown

### 5.3 Customer Management

- User adds/edits/deletes customers via forms

- Prevent delete if customer has active rentals

---

## 6. Validation & Business Logic

- \*\*Prevent Double-Booking:\*\* Only items with status "available" can be rented.

- \*\*Date Validation:\*\* Rental start date ≤ expected return date.

- \*\*Customer Linkage:\*\* Rentals must reference an existing customer.

- \*\*Required Fields:\*\* All necessary fields checked both client- and server-side.

---

## 7. Security

- \*\*Authentication:\*\* Only admin/staff can access rental/customer features.

- \*\*SQL Injection Protection:\*\* Use prepared statements for all database queries.

- \*\*Input Validation:\*\* Sanitize and validate all user inputs.

---

## 8. Error Handling

- User-friendly error messages on the frontend (e.g., "Item already rented", "Missing required information").

- Backend logs all exceptions and validation errors for maintenance.

---

## 9. Extensibility Considerations

- \*\*Modular Controllers:\*\* Separate logic for rentals, customers, and items to allow future expansion (e.g., payments, notifications).

- \*\*Database Fields:\*\* Status fields and foreign keys allow for easier extension to multi-item rentals or advanced reporting.

---

## 10. Non-Functional Requirements

- \*\*Responsiveness:\*\* UI adapts to desktop and mobile.

- \*\*Performance:\*\* All rental and customer queries optimized for real-time operation.

- \*\*Documentation:\*\* Inline code comments and user-facing documentation for flows.

---

## 11. Out of Scope

- Self-service customer portal

- Payment processing, deposits, and fees

- Automated notifications and reminders

- Advanced analytics or reporting dashboards

---

## 12. Appendix

- See Basic Rental System Requirements Updated Version Basic\_Rental\_System\_Requirements\_v2.md for functional and technical requirements.