# Newsagent Management System Documentation

This document provides information about the Newsagent Management System. The system manages customers, areas, orders, invoices, publications, delivery dockets, and warning letters. It leverages a MySQL database for storage and have validation and CRUD (Create, Read, Update, Delete) operations for each entity.

## Customer

Purpose: Manages customer information.

### Fields:

- customer\_id: Unique ID for the customer.  
- name: Customer's name (2-50 characters).  
- address: Customer's address (5-60 characters).  
- phoneNumber: Phone number (7-15 characters).  
- area\_id: ID of the area the customer belongs to.  
- email: Customer's email address (optional).  
- last\_payment\_date: Last payment date (YYYY-MM-DD format).  
- status: Status of the customer (e.g., active, inactive, suspended).

### Validation:

- Name, address, and phone must meet length constraints.  
- Email and last payment date must be valid formats.  
- Area ID must be a positive number.

### Example Usage:

const customer = new Customer(null, "John Doe", "123 Main St", "5551234", 1, "john@example.com", "2024-01-01", "active");

## Area

Purpose: Represents a geographical area where services are provided.

### Fields:

- area\_id: Unique ID for the area.  
- name: Name of the area (2-50 characters).

### Validation:

- Area name must be between 2 and 50 characters.

### Example Usage:

const area = new Area(null, "Downtown");

## Order

Purpose: Tracks orders placed by customers.

### Fields:

- order\_id: Unique ID for the order.  
- customer\_id: ID of the customer placing the order.  
- area\_id: ID of the area for delivery.  
- newspaper\_id: ID of the newspaper being ordered.  
- delivery\_date: Date of delivery (YYYY-MM-DD format).  
- status: Status of the order (e.g., pending, delivered, missed, canceled).

### Validation:

- All IDs must be positive numbers.  
- Delivery date must be valid.  
- Status must be one of the allowed values.

### Example Usage:

const order = new Order(null, 1, 1, 1, "2024-12-01", "pending");

## Invoice

Purpose: Manages customer invoices.

### Fields:

- invoice\_id: Unique ID for the invoice.  
- customer\_id: ID of the customer.  
- invoice\_date: Invoice generation date.  
- due\_date: Due date for payment.  
- total\_amount: Total amount of the invoice.  
- payment\_status: Payment status (e.g., paid, unpaid, late).  
- details: List of details (array of objects).

### Validation:

- Dates must be valid.  
- Total amount must be a positive number.  
- Payment status must be one of the allowed values.

### Example Usage:

const invoice = new Invoice(null, 1, "2024-11-01", "2024-12-01", 100.50, "unpaid", [{ item: "Newspaper", amount: 5 }]);

## Publication

Purpose: Manages newspaper publications.

### Fields:

- publication\_id: Unique ID for the publication.  
- name: Name of the publication (2-50 characters).  
- type: Type of publication (daily, weekly, monthly).  
- price: Price of the publication.

### Validation:

- Name must meet length requirements.  
- Type must be one of the allowed values.  
- Price must be a number between 0 and 1000.

### Example Usage:

const publication = new Publication("The Daily News", "daily", 5.00);

## Delivery Docket

Purpose: Tracks delivery details.

### Fields:

- docket\_id: Unique ID for the docket.  
- area\_id: ID of the delivery area.  
- delivery\_person: Name of the delivery person.  
- orders: List of orders.

### Validation:

- Area ID must be positive.  
- Delivery person must be a non-empty string.

### Example Usage:

const docket = new DeliveryDocket(null, 1, "John Smith", [{ order\_id: 1, status: "pending" }]);

## Warning Letter

Purpose: Issues warning letters to customers.

### Fields:

- letter\_id: Unique ID for the letter.  
- customer\_id: ID of the customer.  
- warning\_date: Date of the warning (YYYY-MM-DD format).  
- status: Status of the warning (warning, suspension).  
- message: Warning message.

### Validation:

- Customer ID must be positive.  
- Warning date must be valid.  
- Status must be one of the allowed values.  
- Message must be a non-empty string.

### Example Usage:

const warningLetter = new WarningLetter(1, "2024-11-07", "warning", "Late payment warning");

## Database Access

Handles CRUD operations for all entities. Provides flexible methods to manage data in the database while ensuring data integrity.

### Key Methods:

- createRecord(table, columns, values): Inserts a new record into a table.  
- readRecordById(table, idField, idValue): Reads a record by ID.  
- updateRecord(table, fields, values, idField, idValue): Updates a record.  
- deleteRecord(table, idField, idValue): Deletes a record.

### Example Usage:

const db = new MySQLAccess();  
await db.connectToDatabase();  
await db.createRecord("customers", ["name", "address"], ["John Doe", "123 Main St"]);