Chapter Three

3.1 Low-level (Detail) Design (class design)

Low-level design (LLD), also known as detail design or class design is the phase in software development where the system's components and modules are described in more detail. This phase follows high-level design and involves specifying the details of each class, including their methods, attributes, relationships, and interactions

3.2 Components of Class Diagram

● **User**:

- Attributes: UserId (string), Username (string), Userpassword (string), Userid (string).
- Methods: Login(), Logout().
- Role: Base class for system users, handling authentication and basic user operations.

Admin (inherits from User):

- Attributes: Admin Name (string), Admin phone (int).
- Methods: registerUser(), trackInventory(), monitorExpired(), viewSalesReport(), viewActivityPage(), viewWeeklyReports(), manageMachines().
- Role: Manages user registrations, monitors inventory, checks expired medicines, views reports, and oversees pharmacy equipment.

Pharmacist (inherits from User):

- Attributes: Phamid (string), pharmname (string), pharmphone (string).
- Methods: viewmedicine(), receiveStockAlert(), sellmedicine(), generateWeeklyReports(), manageMedicine().
- Role: Handles medicine sales, receives stock alerts, manages medicine details, and generates reports.

Inventory:

- Attributes: inventoryId (string), medicines (list of Medicine objects).

- Methods: trackStock(), updateStock().
- Role: Tracks real-time stock levels and updates inventory after sales or restocking.

Medicine:

- Attributes: medicineId (string), name (string), type (string), quantity (int), exprimDate (Date), price (double).
- Methods: Expired() (checks if the medicine is expired).
- Role: Represents individual medicines and their properties, including expiration status.

Alert:

- Attributes: alertId (string), message (string), date (Date).
- Methods: sendAlert().
- Role: Generates notifications for low stock, expired medicines, or payment failures.

Report:

- Attributes: reportId (string), reportType (string), generatedDate (Date), content (string).
- Methods: generateReport().
- Role: Creates structured reports (e.g., weekly sales, inventory summaries) for analysis.

Key Relationships:

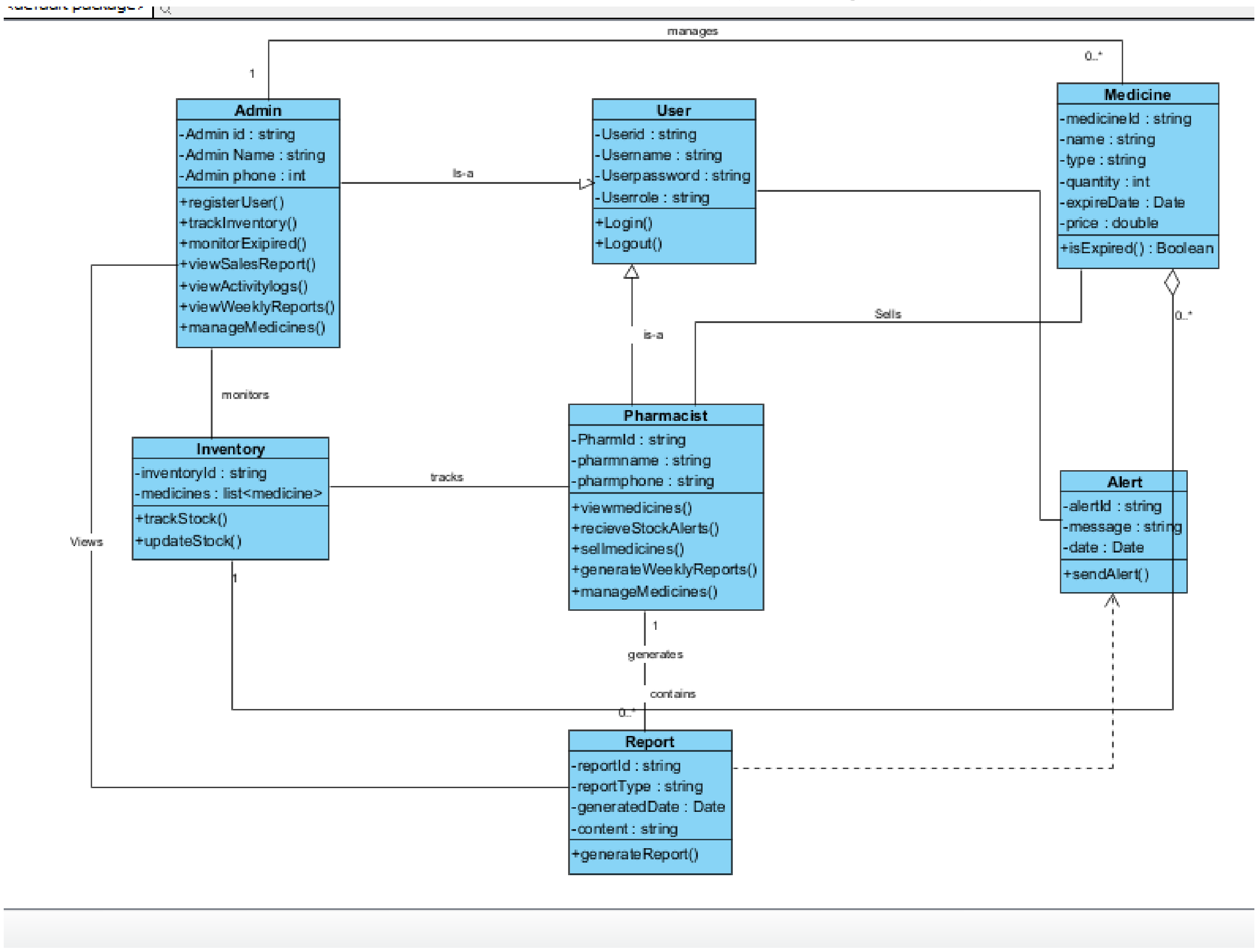
- Admin and Pharmacist inherit from User, enabling role-based access control.
- Inventory contains a list of Medicine objects, updated via trackStock() and updateStock().
- Pharmacist interacts with Report to generate weekly reports and with Alert to receive stock notifications.
- Medicine's Expired() method triggers Alert when expiry dates are near.
- Admin uses trackInventory() and monitorExpired() to manage Inventory and Alert systems.

Key Relationships:

- **Admin** and **Pharmacist** inherit from **User**, enabling role-based access.
- **Inventory** contains a list of **Medicine** objects, updated via sales or restocking.
- **Alert** is triggered by **Inventory** checks (e.g., expiry or low stock) or payment system failures.
- **Pharmacist** interacts with **Report** to generate and view summaries.

3.3 Example of Class Diagram

The following class diagram, created using Visual Paradigm, represents the detailed object-oriented structure of the pharmacy management systems



3.4 Tools and steps to draw Class Diagram

Step 1: Open Visual Paradigm

- Launch Visual Paradigm → Create a **New Project** → Select **UML Diagram** → **Class Diagram**.

Step 2: Add Classes

1. Drag the **Class** icon from the toolbar onto the canvas.
2. Name each class (e.g., user, Admin, pharmacist ,medicine, Inventory, Alert, Report)

Step 3: Define Attributes and Methods

- Double-click a class to add:
  - **Attributes** (e.g. MedicineId: string, quantity :int)
  - **Methods** (e.g., Expired ( ): Boolean, Updatestock( )
- Example for the Medicine class:

Step 4: Add Relationships

Use the **Relationship Toolbar** to connect classes:

1. **Inheritance (Generalization)**:
   - Example: admin and pharmacist inherit from user
   - Use a hollow arrow (▷) pointing to the parent class.
2. **Composition/Aggregation**:
   - Example Inventory "contains" a list of medicine objects.
   - Use a diamond arrow (◆) for composition.
3. **Associations**:
   - Example pharmacists: interacts with report.

  - Use a simple line with an optional label (e.g., "generates").

Step 5: Add Notes or Constraints (Optional)

- Use the **Note Tool** to clarify complex relationships.
  - Example: Add a note to Alert class: *"Triggers when stock < 10 or medicine expires."*

Step 6: Validate and Refine

- Ensure all attributes/methods align with your system’s requirements.
- Use **Auto-Layout** to organize classes neatly.

Step 7: Export/Share

- Save the diagram → Export as PNG/PDF via **File > Export**.