### 🔷 ****System Design (Using ATM System Example)****

System design defines **how** the system will be structured at a high level. Here's how each step applies to the ATM system:

1. **Organize the system into subsystems**  
   ➤ Divide the ATM system into logical parts like:
   * **User Interface Subsystem** (screen, buttons)
   * **Transaction Processing Subsystem**
   * **Authentication Subsystem**
   * **Bank Database Interface Subsystem**
2. **Identify concurrency inherent in the problem**  
   ➤ ATMs can operate concurrently at different physical locations. So, multiple transactions (withdrawals, balance checks) may happen at the same time on different ATMs.  
   ➤ Need to manage **shared resources** like central database carefully.
3. **Allocate subsystems to processors and tasks**  
   ➤ For example:
   * User Interface runs on the ATM machine
   * Transaction processing may involve server-side communication
   * Authentication may call a secure server for PIN validation
4. **Choose the basic strategy for implementing data stores**  
   ➤ Bank data (accounts, balances) stored in a **relational database**  
   ➤ Use **tables** for Users, Accounts, Transactions  
   ➤ Use **encryption** for sensitive data like PIN
5. **Identify global resources and control access**  
   ➤ Global resource = Central Bank Database  
   ➤ Only authenticated users should access their account data  
   ➤ Use session tokens, access control mechanisms
6. **Choose approach to implement software control**  
   ➤ ATM operations follow **state transitions** (e.g., Idle → Card Inserted → PIN Verified → Transaction)  
   ➤ So, use a **state machine** to manage the control flow
7. **Consider boundary conditions**  
   ➤ What happens if:
   * Card is ejected midway?
   * User enters wrong PIN 3 times?
   * Network/server fails?  
     ➤ Handle such edge cases with fallback and alerts
8. **Establish trade-off priorities**  
   ➤ Example trade-offs:
   * **Security** vs **Speed**
   * **Cost** of high-end machines vs **Performance**
   * **Offline support** vs **Real-time access**

➡️ **System Design Document for ATM**  
= Describes architecture (UI, Processing, DB), data strategies, concurrency, control approach, error handling, etc.

### 🔷 ****Object Design (Using ATM System Example)****

Object design defines the **internal structure** using **classes and objects** to implement the system. Here's the breakdown:

1. **Obtain operations for the object model**  
   ➤ Identify key operations:
   * Authenticate user
   * Check balance
   * Withdraw money
   * Transfer funds  
     ➤ Each operation links to a class: ATM, Account, Transaction
2. **Design algorithms to implement operations**  
   ➤ Example:
   * **Authenticate user:** match entered PIN with stored PIN using hashing
   * **Withdraw:** check balance → deduct amount → record transaction
3. **Optimize access paths to data**  
   ➤ Store recent transactions in memory for fast display  
   ➤ Use efficient data structures like hash maps for account lookup  
   ➤ Cache frequent operations where secure
4. **Implement software control**  
   ➤ Use **state machine pattern**
   * Idle → CardInserted → PINVerified → TransactionSelected → Completed  
     ➤ Define transitions and validations in code
5. **Adjust class structure to increase inheritance**  
   ➤ Create general class: Transaction  
   ➤ Subclasses: WithdrawTransaction, DepositTransaction, TransferTransaction  
   ➤ Use common behavior in parent class, like timestamp or validation
6. **Design implementation of associations**  
   ➤ ATM is associated with many Accounts  
   ➤ Account has many Transactions  
   ➤ Model this using references and associations in code
7. **Determine representation of object attributes**  
   ➤ Account: accountNumber, balance, PINHash  
   ➤ ATM: id, location  
   ➤ Transaction: amount, time, type
8. **Package classes and associations into modules**  
   ➤ Divide into packages like:
   * com.atm.auth (for login/authentication classes)
   * com.atm.ui (for screen and button handling)
   * com.atm.transactions (for different transaction types)
   * com.atm.db (for database interaction)

➡️ **Design Document for ATM**  
= Includes class diagrams, attributes, method details, associations, and interaction sequences