1. Write Software Requirement Specification for ATM System.  
Here is an example of a Software Requirements Specification (SRS) document for an ATM System:

### **1. Introduction**

#### **1.1 Purpose**

The purpose of this SRS document is to describe the functional and non-functional requirements for an Automated Teller Machine (ATM) System. This system will enable users to perform banking transactions like withdrawing cash, checking account balances, transferring funds, and more in a secure and convenient manner.

#### **1.2 Scope**

The ATM System is a standalone machine connected to a banking network. It will support user authentication, cash withdrawal, fund transfers, balance inquiries, and transaction history viewing. The system will ensure secure and efficient transactions, providing 24/7 banking services to users.

#### **1.3 Definitions, Acronyms, and Abbreviations**

* **ATM:** Automated Teller Machine.
* **Cardholder:** A bank account holder with an issued ATM or debit card.
* **PIN:** Personal Identification Number used for user authentication.
* **Admin:** Bank personnel responsible for maintaining and managing the ATM.
* **Transaction:** Any operation performed by a user, such as withdrawal or fund transfer.

#### **1.4 References**

* Banking Network Security Standards
* Financial Data Protection Regulations
* ISO 8583 Standard for Financial Transaction Messaging

#### **1.5 Document Overview**

This document outlines functional requirements, external interface requirements, and system specifications for the ATM System. It also describes assumptions, dependencies, and constraints for its operation and implementation.

### **2. General Description**

#### **2.1 Product Perspective**

The ATM System will act as an interface between customers and the bank's central system. It will consist of modules for user authentication, account management, transaction processing, and reporting.

#### **2.2 Product Functions**

The main functions of the ATM System include:

* **User Authentication:** Authenticate users via card and PIN.
* **Cash Withdrawal:** Allow users to withdraw available funds.
* **Balance Inquiry:** Display the current account balance.
* **Fund Transfer:** Enable money transfers between accounts.
* **Transaction History:** Show recent transaction details.

#### **2.3 User Characteristics**

* **Cardholders:** Users who have a bank-issued ATM card and perform transactions.
* **Admin:** Bank personnel responsible for cash replenishment and machine maintenance.

#### **2.4 General Constraints**

* Must comply with banking regulations and security protocols.
* The system must be operational 24/7 except during scheduled maintenance.
* Limitations on transaction amounts as per banking policies.

#### **2.5 Assumptions and Dependencies**

* Users have valid ATM cards and PINs.
* The ATM is connected to the bank's network for real-time processing.
* Sufficient cash is available in the ATM for withdrawals.

### **3. Specific Requirements**

#### **3.1 Functional Requirements**

**3.1.1 User Authentication**

* **Introduction:** Users authenticate themselves using a card and PIN.
* **Input:** ATM card and PIN.
* **Processing:** Validate credentials against bank records.
* **Output:** Grant or deny access to the ATM functions.

**3.1.2 Cash Withdrawal**

* **Introduction:** Users can withdraw cash from their accounts.
* **Input:** Withdrawal amount.
* **Processing:** Verify account balance and dispense cash.
* **Output:** Cash dispensed and receipt generated.

**3.1.3 Balance Inquiry**

* **Introduction:** Users can check their account balance.
* **Input:** Account details.
* **Processing:** Fetch the current balance from the bank's database.
* **Output:** Display balance on the screen.

**3.1.4 Fund Transfer**

* **Introduction:** Users can transfer funds between accounts.
* **Input:** Beneficiary account details and transfer amount.
* **Processing:** Validate beneficiary account and deduct funds.
* **Output:** Confirmation of successful transfer.

**3.1.5 Transaction History**

* **Introduction:** Displays the user's recent transactions.
* **Input:** User request for transaction history.
* **Processing:** Retrieve and format transaction data.
* **Output:** Display recent transactions.

#### **3.2 External Interface Requirements**

* **User Interface:** Touchscreen interface with clear and simple navigation.
* **Hardware Interface:** Card reader, PIN pad, cash dispenser, and receipt printer.
* **Network Interface:** Connection to the bank's central server for real-time updates.

#### **3.3 Performance Requirements**

* Must process transactions within 5 seconds.
* Should handle concurrent transactions across multiple ATMs in the network.

#### **3.4 Design Constraints**

* Developed as a standalone machine with robust security.
* Supports multiple languages for diverse users.

#### **3.5 Security Requirements**

* Encrypt all sensitive data during transmission and storage.
* PIN validation with a maximum of three incorrect attempts before card lockout.
* Secure hardware with tamper-proof mechanisms.

#### **3.6 Maintainability Requirements**

* Modular design for ease of updates and troubleshooting.
* Regular logs and diagnostics for monitoring performance.

#### **3.7 Reliability Requirements**

* Must operate with a 99.99% uptime.
* Redundant systems to handle power and network outages.

#### **3.8 Availability Requirements**

* Available 24/7 except for scheduled maintenance.

#### **3.9 Database Requirements**

* Store transaction data securely with backup systems.
* Ensure compliance with financial data protection regulations.

#### **3.10 Documentation Requirements**

* User manuals for cardholders.
* Maintenance manuals for administrators.

#### **3.11 Safety Requirements**

* Must comply with safety standards for electronic and mechanical components.
* Include emergency stop functionality for hardware malfunctions.

#### **3.12 Operational Requirements**

* Compatible with various types of ATM cards (chip, magnetic strip).
* User-friendly interface for customers of all age groups.

#### **3.13 Site Adaptation**

* Installed in secure and accessible locations.
* Adaptable to different banking requirements and policies.

This document can be further customized based on specific banking requirements or project details.

2. Draw any one UML Diagram for ATM System.

3. Create Project Management Process for ATM System using Agile tools and techniques.  
**Agile Methodology Case Study — ATM System**

**Inception of Agile Methodology:** Waterfall projects often failed due to rigid processes and the inability to adapt to evolving requirements. Agile methodology was developed to address these shortcomings by fostering flexibility and collaboration in software development. It combines iterative and incremental approaches to accommodate changing customer needs.

* **Iterative Development:** Progress is made through repeated refinement of the product, incorporating user feedback during each iteration.
* **Incremental Development:** The system is built and delivered in stages, with each increment representing a fully functional subset of features that are coded, tested, and potentially deployable.

**Iterative and Incremental Development Model** The Agile Manifesto emphasizes:

* Individuals and interactions over processes and tools
* Working software over comprehensive documentation
* Customer collaboration over contract negotiation
* Responding to change over following a plan

This philosophy introduced a transformative approach to project management and software development.

### **About the System — ATM System**

An ATM (Automated Teller Machine) system allows bank customers to perform financial transactions such as cash withdrawals, deposits, balance inquiries, and fund transfers. The product vision is to provide secure, efficient, and user-friendly services to customers.

### **Implementation of Agile Methodology in ATM System**

**Framework:** The **Scrum Framework** can be utilized for developing the ATM system. Scrum is a lightweight Agile framework that organizes work into sprints, allowing teams to deliver potentially shippable increments at the end of each sprint.

#### **Scrum Team for ATM System**

1. **Product Owner:**
   * Defines and prioritizes the product backlog.
   * Ensures the functionality aligns with customer needs.
   * Maximizes ROI by focusing on business goals.
2. **Scrum Master:**
   * Coaches the team and facilitates Scrum events like stand-ups, sprint reviews, and retrospectives.
   * Addresses impediments that may block the team's progress.
3. **Development Team:**
   * Self-organizing team responsible for delivering sprint goals.
   * Includes developers, testers, and designers.

#### **Typical Scrum Implementation**

* **Team Size:** 5 to 9 members
* **Sprint Duration:** 2 to 4 weeks
* **Sprint Backlog:** Prioritized user stories from the product backlog, divided into tasks.

### **Feature List**

The following features will be implemented in the ATM System:

1. **User Authentication**
2. **Cash Withdrawal and Deposit**
3. **Balance Inquiry**
4. **Fund Transfer**
5. **Transaction History and Printing**

### **Feature Implementation Phases**

#### **Feature 1: User Authentication**

* **Ideation:** Users authenticate using a card or credentials (PIN, biometrics, etc.). Ensure robust security to prevent unauthorized access.
* **Design:**
  + Card and PIN input screens
  + OTP-based or biometric authentication for enhanced security
* **Implementation & Testing:**
  + Backend logic for validating credentials and managing session tokens.
  + QA testing for secure authentication and multi-factor methods.
* **Deployment:** The feature is deployed, allowing users to log in securely.

#### **Feature 2: Cash Withdrawal and Deposit**

* **Ideation:**
  + Withdrawal: Users can withdraw cash based on their account balance and daily limit.
  + Deposit: Users can deposit cash or checks with validation for correctness.
* **Design:**
  + Withdrawal screen with keypad for amount entry.
  + Deposit interface for cash/check entry.
* **Implementation & Testing:**
  + Develop logic for validating amounts and updating the account balance.
  + Test edge cases such as overdrafts and incorrect deposits.
* **Deployment:** The feature is made available for secure withdrawals and deposits.

#### **Feature 3: Balance Inquiry**

* **Ideation:** Users should be able to view their current account balance.
* **Design:**
  + Display balance on the ATM screen or print receipt.
* **Implementation & Testing:**
  + Develop the functionality to fetch and display balance securely.
  + Ensure correct handling of concurrent transactions.
* **Deployment:** The feature is deployed, allowing real-time balance inquiries.

#### **Feature 4: Fund Transfer**

* **Ideation:** Users can transfer funds between accounts, both within the same bank and to other banks.
* **Design:**
  + Input interface for recipient account details and amount.
  + Confirmation screen before processing the transaction.
* **Implementation & Testing:**
  + Develop API integration for interbank fund transfers.
  + Test for compliance with banking protocols and transaction limits.
* **Deployment:** The feature is rolled out, enabling seamless fund transfers.

#### **Feature 5: Transaction History and Printing**

* **Ideation:** Users can view recent transaction history and print receipts for selected transactions.
* **Design:**
  + Transaction history screen with pagination.
  + Option for receipt printing.
* **Implementation & Testing:**
  + Implement backend to fetch transaction data securely.
  + Test the functionality with varied transaction loads.
* **Deployment:** The feature is deployed, providing users with transaction records.

### **Summary of Agile Implementation**

Each feature of the ATM system is iteratively designed, implemented, tested, and deployed. Agile methodology ensures flexibility, collaboration, and continuous improvement to meet user needs effectively.

4. Estimate the cost for implementation of ATM System  
**Project Scope — ATM System**

The ATM system involves the following components:

1. **User Interface:**
   * For customers to perform transactions (withdrawals, deposits, balance inquiry, etc.)
   * Admin interface for ATM monitoring and maintenance.
2. **Transaction Processing:**
   * Secure handling of financial transactions.
   * Integration with banking core systems for real-time updates.
3. **Backend Database:**
   * Stores user data, account details, transaction logs, and machine statuses.
4. **Security Features:**
   * Authentication using PINs, biometrics, or OTP.
   * Encryption for sensitive data.
5. **APIs for Integration:**
   * To connect with third-party systems, such as banking networks and fraud detection systems.

The system is categorized as a **semi-detached project** due to its moderate complexity, involving reusable components and custom functionalities.

### **COCOMO Basic Effort Estimation for ATM System**

**1. Basic Effort Equation** The COCOMO effort equation is:  
 E = at{KLOC}^b  
 For a semi-detached project:

* a=3.0
* b=1.12

**2. Estimate KLOC (Thousands of Lines of Code)** Estimated size of an ATM system:

* User Interface: 10 KLOC
* Backend Processing: 15 KLOC
* Security and Encryption Features: 5 KLOC
* API Integration: 10 KLOC
* Additional Features (e.g., fraud detection, reporting): 10 KLOC  
   **Total Estimated Size:** 50 KLOC

**3. Calculate Effort (E)** Using the equation:  
E = 3.0{KLOC}^{1.12}  
 Substitute the values:  
 E = 3.0(50)^{1.12}  
 **Effort (E):** Approximately **154 person-months (PM)**

**4. Estimate Development Time (T)** The development time equation is:  
T = c{Effort}^d  
 For semi-detached projects:

* c=2.5
* d=0.35

Substitute the values:  
 T = 2.5 \* (154)^{0.35}  
 **Development Time (T):** Approximately **16.5 months**

### **Cost Estimation for ATM System**

**1. Average Monthly Cost of a Developer** Average Monthly Cost=$10,000×₹83=₹8,30,000

**2. Total Development Cost** Cost=Effort×Monthly Cost

Cost=154×₹8,30,000

**Total Development Cost:** ₹12,78,20,000/-

### **Additional Considerations**

1. **Hardware and Deployment Costs:**
   * ATMs, servers, and networking equipment.
   * Estimated cost: ₹1.5 to ₹6 crore, depending on the number of ATMs deployed.
2. **Operational Costs:**
   * Annual maintenance, monitoring, and updates: 10-20% of development cost (~₹1.27 to ₹2.55 crore/year).

### **Final Estimate for ATM System**

* **Development Cost (COCOMO):** ₹12.78 crore
* **Hardware and Deployment:** ₹1.5 crore to ₹6 crore
* **Total Initial Cost:** ₹14.28 crore to ₹18.78 crore

5. Write Software Requirement Specification for Library Management System.  
Here’s an SRS document for a Library Management System (LMS) following the provided structure:

### **1. Introduction**

#### **1.1 Purpose**

The purpose of this SRS document is to describe the functional and non-functional requirements for a Library Management System (LMS). This system will allow users to manage book inventory, track borrowing and returns, and streamline library operations for administrators.

#### **1.2 Scope**

The Library Management System (LMS) will be a web-based platform accessible to librarians, library members, and administrators. It will support book cataloging, user registration, book borrowing and returning, fine calculation, and report generation.

#### **1.3 Definitions, Acronyms, and Abbreviations**

* **LMS:** Library Management System
* **Member:** A user who registers to borrow or return books.
* **Librarian:** A user responsible for managing the library's inventory and transactions.
* **Admin:** An authorized user who manages system configurations and access.
* **ISBN:** International Standard Book Number, used to identify books.

#### **1.4 References**

* IEEE SRS Standards Document
* Library Management Guidelines
* Data Protection Regulations and Privacy Laws

#### **1.5 Document Overview**

This document includes functional requirements, external interface requirements, and other specifications necessary to build and maintain the Library Management System. It also outlines assumptions, dependencies, and constraints.

### **2. General Description**

#### **2.1 Product Perspective**

The Library Management System will replace traditional paper-based library operations with a digital solution accessible via a web browser. The system will include modules for user management, book inventory, borrowing, returning, and reporting.

#### **2.2 Product Functions**

The main functions of the Library Management System include:

* **User Registration and Login:** Allow members and librarians to create accounts.
* **Book Inventory Management:** Enable librarians to add, edit, and delete book records.
* **Borrowing and Returning Books:** Track the status of borrowed and returned books.
* **Fine Calculation:** Automatically calculate and track overdue fines.
* **Report Generation:** Provide detailed reports on library usage and inventory.

#### **2.3 User Characteristics**

* **Members:** Users who borrow books and manage their accounts.
* **Librarians:** Users who manage books and transactions.
* **Admins:** Users who manage system configurations and user access.

#### **2.4 General Constraints**

* The system must be accessible during library hours.
* It must handle a high volume of transactions, especially during peak hours.
* All operations must comply with institutional library policies.

#### **2.5 Assumptions and Dependencies**

* Users have reliable internet access and compatible browsers.
* Librarians provide accurate book information during cataloging.
* The system integrates with institutional databases for user information.

### **3. Specific Requirements**

#### **3.1 Functional Requirements**

**3.1.1 User Registration and Authentication**

* **Introduction:** Users must register and authenticate to access the system.
* **Input:** User details such as name, email, and password.
* **Processing:** Verify user details and create profiles.
* **Output:** Confirmation of registration and access to the dashboard.

**3.1.2 Book Inventory Management**

* **Introduction:** Librarians can manage the library's book collection.
* **Input:** Book details (title, author, ISBN, etc.).
* **Processing:** Add, edit, or remove book records.
* **Output:** Confirmation of inventory updates.

**3.1.3 Borrowing and Returning Books**

* **Introduction:** Members can borrow and return books within specified rules.
* **Input:** Book ID, user ID, and transaction type.
* **Processing:** Update book status and transaction records.
* **Output:** Transaction confirmation.

**3.1.4 Fine Calculation**

* **Introduction:** Automatically calculates fines for overdue books.
* **Input:** Due date and return date.
* **Processing:** Calculate overdue days and fine amount.
* **Output:** Display fine amount to the user.

**3.1.5 Report Generation**

* **Introduction:** Generate reports on library activity.
* **Input:** Date range or specific criteria (e.g., most borrowed books).
* **Processing:** Retrieve and process data.
* **Output:** Detailed report in a specified format.

#### **3.2 External Interface Requirements**

* **User Interface:** Web-based interface accessible via modern browsers.
* **API Integration:** Interfaces for integrating with other library or institutional systems.

#### **3.3 Performance Requirements**

* Must handle multiple concurrent users without downtime.
* Response times for user actions (e.g., searching books, issuing books) should be under 2 seconds.

#### **3.4 Design Constraints**

* Developed as a web-based application with responsive design.
* Data encryption for secure communication.

#### **3.5 Security Requirements**

* Role-based access control to restrict access by role.
* Multi-factor authentication (MFA) for admin accounts.
* Encrypt sensitive data, such as user details and transaction history.

#### **3.6 Maintainability Requirements**

* Modular code structure for future enhancements.
* Documentation for code, setup, and maintenance procedures.

#### **3.7 Reliability Requirements**

* Ensure 99.9% uptime during library hours.
* Redundant storage for transaction and inventory data.

#### **3.8 Availability Requirements**

* Available to users during library hours.
* Maintenance scheduled outside of peak usage times.

#### **3.9 Database Requirements**

* Securely store user, book, and transaction data.
* Regular data backups in compliance with privacy regulations.

#### **3.10 Documentation Requirements**

* User manuals for members and librarians.
* Technical documentation for developers and administrators.

#### **3.11 Safety Requirements**

* Comply with data privacy laws for all user data.
* Validate data inputs to prevent errors and unauthorized access.

#### **3.12 Operational Requirements**

* Compatible with commonly used web browsers.
* Simple and intuitive navigation for all user roles.

#### **3.13 Site Adaptation**

* Accessible via web browsers on various devices.
* Adaptable to different library policies and regulations.

This document can be tailored further to meet specific institutional needs or project details.

6. Draw any one UML Diagram for Library Management System.

7. Create Project Management Process for Library Management System using Agile tools and techniques.  
**Agile Methodology Case Study — Library Management System**

**Inception of Agile Methodology:** The limitations of the Waterfall model, such as rigidity and inability to accommodate evolving requirements, highlighted the need for Agile methodology. Agile combines iterative and incremental approaches to deliver functionality in smaller, manageable portions while adapting to changes in requirements.

* **Iterative Development:** Progress is achieved through repeated refinement, incorporating feedback during each iteration to align with user needs.
* **Incremental Development:** The system is built and delivered in stages, with each increment delivering a functional subset of features that are coded, tested, and ready for deployment.

**Iterative and Incremental Development Model** The Agile Manifesto emphasizes:

* Individuals and interactions over processes and tools
* Working software over comprehensive documentation
* Customer collaboration over contract negotiation
* Responding to change over following a plan

This approach transformed software project management, emphasizing collaboration, adaptability, and customer satisfaction.

### **About the System — Library Management System**

A Library Management System (LMS) helps libraries manage resources, track borrowing and returns, and maintain an organized catalog of books and media. The product vision is to create a user-friendly, efficient system for librarians, members, and administrators.

### **Implementation of Agile Methodology in Library Management System**

**Framework:** The **Scrum Framework** is well-suited for developing the LMS. Scrum organizes development into sprints, delivering potentially shippable increments at the end of each sprint.

#### **Scrum Team for Library Management System**

1. **Product Owner:**
   * Defines and prioritizes the product backlog.
   * Aligns development with business goals and user needs.
   * Maximizes ROI through iterative delivery of valuable features.
2. **Scrum Master:**
   * Facilitates Scrum events like daily stand-ups, sprint reviews, and retrospectives.
   * Resolves impediments and ensures smooth team collaboration.
3. **Development Team:**
   * Self-organizing team responsible for delivering sprint goals.
   * Includes developers, testers, and designers.

#### **Typical Scrum Implementation**

* **Team Size:** 5 to 9 members
* **Sprint Duration:** 2 to 4 weeks
* **Sprint Backlog:** Prioritized user stories divided into tasks for the sprint.

### **Feature List**

The following features will be implemented in the Library Management System:

1. **User Registration and Login**
2. **Catalog Management**
3. **Borrowing and Returns**
4. **Search and Filtering**
5. **Reports and Analytics**

### **Feature Implementation Phases**

#### **Feature 1: User Registration and Login**

* **Ideation:** Users register as librarians or members. Login credentials are verified, and access is role-based (admin, librarian, member). Password recovery functionality is also required.
* **Design:**
  + Registration form with role selection
  + Login page with secure authentication
  + Admin panel for user management
* **Implementation & Testing:**
  + Backend functionality for user management and role-based authentication.
  + Testing for secure registration, login, and password recovery.
* **Deployment:** This feature is deployed, enabling secure user registration and login.

#### **Feature 2: Catalog Management**

* **Ideation:** Librarians can add, update, or remove books and media. Each item includes metadata like title, author, category, and availability status.
* **Design:**
  + User-friendly interface for catalog management
  + Option for bulk import/export of records
* **Implementation & Testing:**
  + Backend development for managing catalog data.
  + Testing ensures accurate addition, update, and deletion of records.
* **Deployment:** The catalog management feature is deployed, enabling librarians to manage resources effectively.

#### **Feature 3: Borrowing and Returns**

* **Ideation:** Members can borrow items, and librarians can record transactions. Automatic notifications for due dates and overdue items are included.
* **Design:**
  + Borrowing and return interface for librarians
  + Member dashboard for transaction history and notifications
* **Implementation & Testing:**
  + Logic for borrowing limits, due dates, and overdue penalties.
  + Testing ensures accurate record-keeping and notifications.
* **Deployment:** The borrowing and return feature is deployed, streamlining transaction management.

#### **Feature 4: Search and Filtering**

* **Ideation:** Members and librarians can search for books by title, author, category, or availability. Advanced filters for specific queries enhance the experience.
* **Design:**
  + Search bar with filter options
  + Pagination for displaying search results
* **Implementation & Testing:**
  + Development of search algorithms and filtering logic.
  + Testing for search accuracy and performance.
* **Deployment:** The search and filtering feature is deployed, improving resource discovery.

#### **Feature 5: Reports and Analytics**

* **Ideation:** Librarians and admins can generate reports on borrowing trends, overdue items, and resource usage analytics.
* **Design:**
  + Dashboard for visualizing trends and generating reports
  + Options to export reports in various formats
* **Implementation & Testing:**
  + Develop analytics and reporting tools.
  + Testing ensures accuracy and usability of reports.
* **Deployment:** This feature is deployed, enabling insights into library operations.

### **Summary of Agile Implementation**

The Library Management System is developed using Agile methodology, ensuring adaptability, collaboration, and continuous improvement. Each feature is iteratively designed, implemented, tested, and deployed, delivering a user-centric solution for managing library resources efficiently.

8. Estimate the cost for implementation of Library Management System.  
  
**COCOMO Effort Estimation for Library Management System**

### **1. Define the Project Scope**

The Library Management System involves the following components:

1. **User Interface:**
   * Interfaces for users (students, library members) to search and borrow books.
   * Admin interface to manage books, members, and transactions.
2. **Book Management System:**
   * Functions to add, remove, and update book details.
   * Categorization by author, genre, publication, and availability.
3. **Membership and Borrowing Management:**
   * User registration, borrowing, and return tracking.
   * Automated reminders for due dates and fines calculation.
4. **Backend Database:**
   * Stores data on books, members, transactions, and overdue records.
5. **Security Features:**
   * Role-based access for users and administrators.
6. **APIs for Integration:**
   * Integration with external systems for e-books, payment systems, and publishers.

This system is categorized as a **semi-detached project** due to moderate complexity and its mix of reusable and custom functionalities.

### **2. COCOMO Basic Effort Equation**

The basic effort equation is:  
 E=a×(KLOC)bE = a \times (\text{KLOC})^b  
 For a semi-detached project:

* a=3.0a = 3.0
* b=1.12b = 1.12

### **3. Estimate KLOC (Thousands of Lines of Code)**

Estimated size of the Library Management System:

* User Interface and Logic: 10 KLOC
* Backend Integration and APIs: 15 KLOC
* Security and Role Management: 5 KLOC
* Membership and Borrowing Features: 10 KLOC  
   **Total Estimated Size:** 40 KLOC

### **4. Calculate Effort (E)**

Using the equation:  
 E=3.0×(KLOC)1.12E = 3.0 \times (\text{KLOC})^{1.12}  
 Substitute the values:  
 E=3.0×(40)1.12E = 3.0 \times (40)^{1.12}  
 **Effort (E):** Approximately **123 person-months (PM)**

### **5. Estimate Development Time (T)**

The development time equation is:  
 T=c×(Effort)dT = c \times (\text{Effort})^d  
 For semi-detached projects:

* c=2.5c = 2.5
* d=0.35d = 0.35

Substitute the values:  
 T=2.5×(123)0.35T = 2.5 \times (123)^{0.35}  
 **Development Time (T):** Approximately **15.8 months**

### **6. Estimate Cost**

**Average Monthly Cost of a Software Developer:** Monthly Cost=$10,000×₹83=₹8,30,000\text{Monthly Cost} = \$10,000 \times ₹83 = ₹8,30,000

**Total Development Cost:** Cost=Effort×Monthly Cost\text{Cost} = \text{Effort} \times \text{Monthly Cost}  
 Cost=123×₹8,30,000\text{Cost} = 123 \times ₹8,30,000  
 **Total Development Cost:** ₹10,20,90,000/-

### **7. Additional Considerations**

1. **Hardware and Deployment Costs:**
   * Server and networking hardware for the system.
   * Estimated cost: ₹50 lakh to ₹2 crore, depending on deployment scale.
2. **Operational Costs:**
   * Annual maintenance and updates: 10-20% of development cost (~₹1.02 to ₹2.04 crore/year).

### **Final Estimate for Library Management System**

* **Development Cost (COCOMO):** ₹10.21 crore
* **Hardware and Deployment:** ₹50 lakh to ₹2 crore
* **Total Initial Cost:** ₹10.71 crore to ₹12.21 crore

9. Write Software Requirement Specification for Hotel Management System.  
Here's an SRS document for a **Hotel Management System (HMS)** based on your provided structure:

### **1. Introduction**

#### **1.1 Purpose**

The purpose of this SRS document is to describe the functional and non-functional requirements for a Hotel Management System (HMS). This system will enable efficient management of hotel operations, including room bookings, guest management, and staff coordination.

#### **1.2 Scope**

The Hotel Management System (HMS) will be a web-based platform accessible to hotel staff, administrators, and customers. It will support room reservation, user authentication, billing, and reporting, streamlining daily hotel operations.

#### **1.3 Definitions, Acronyms, and Abbreviations**

* **HMS:** Hotel Management System
* **Guest:** A customer who books rooms and avails hotel services.
* **Staff:** Hotel employees responsible for day-to-day operations.
* **Admin:** An authorized user who manages system settings, user accounts, and hotel operations.
* **POS:** Point of Sale for hotel services like dining and spa.

#### **1.4 References**

* IEEE SRS Standards Document
* Hospitality Industry Guidelines
* Data Protection Regulations and Privacy Laws

#### **1.5 Document Overview**

This document includes functional requirements, external interface requirements, and other specifications necessary to build and maintain the Hotel Management System. It also outlines assumptions, dependencies, and constraints.

### **2. General Description**

#### **2.1 Product Perspective**

The Hotel Management System will replace traditional manual operations with a digital solution accessible via a web browser. It will include modules for guest registration, room management, billing, and reporting.

#### **2.2 Product Functions**

The main functions of the Hotel Management System include:

* **Room Booking and Reservations:** Allow guests to book rooms online.
* **Guest Management:** Maintain guest profiles and preferences.
* **Billing and Invoicing:** Generate invoices for stays and additional services.
* **Staff Management:** Enable admins to manage staff roles and schedules.
* **Reporting:** Provide detailed reports on bookings, revenue, and guest feedback.

#### **2.3 User Characteristics**

* **Guests:** Users who book rooms and avail hotel services.
* **Staff:** Employees responsible for managing guest check-ins, check-outs, and services.
* **Admins:** Users who oversee hotel operations and manage the system.

#### **2.4 General Constraints**

* The system must be operational 24/7 to accommodate global time zones.
* The system must handle peak booking periods without delays.
* All operations must comply with hospitality and privacy regulations.

#### **2.5 Assumptions and Dependencies**

* Users have reliable internet access and compatible devices.
* Staff and admins are trained to use the system effectively.
* The system will integrate with third-party payment gateways and customer feedback systems.

### **3. Specific Requirements**

#### **3.1 Functional Requirements**

**3.1.1 User Registration and Authentication**

* **Introduction:** Users must register and authenticate to access the system.
* **Input:** User details such as name, email, and password.
* **Processing:** Verify user details and create user profiles.
* **Output:** Confirmation of registration and access to the main dashboard.

**3.1.2 Room Booking and Reservations**

* **Introduction:** Guests can book rooms online based on availability.
* **Input:** Check-in/out dates, room type, and guest details.
* **Processing:** Check room availability, calculate total charges, and reserve the room.
* **Output:** Booking confirmation with details.

**3.1.3 Guest Management**

* **Introduction:** Maintain guest profiles, including preferences and stay history.
* **Input:** Guest information and service preferences.
* **Processing:** Save and update guest profiles.
* **Output:** Updated guest profile accessible to staff.

**3.1.4 Billing and Invoicing**

* **Introduction:** Generate invoices for room bookings and additional services.
* **Input:** Room charges, service charges, and taxes.
* **Processing:** Calculate total bill and generate an invoice.
* **Output:** Printable or email-ready invoice.

**3.1.5 Staff Management**

* **Introduction:** Admins can manage staff roles and schedules.
* **Input:** Staff details, roles, and work schedules.
* **Processing:** Save and update staff information.
* **Output:** Updated staff schedule and role assignment.

**3.1.6 Reporting**

* **Introduction:** Generate reports on bookings, revenue, and guest feedback.
* **Input:** Date range or specific criteria (e.g., room occupancy).
* **Processing:** Retrieve and process data.
* **Output:** Report in a specified format.

#### **3.2 External Interface Requirements**

* **User Interface:** Web-based interface accessible through modern browsers.
* **API Integration:** Interfaces for connecting to payment gateways and CRM systems.

#### **3.3 Performance Requirements**

* Must handle multiple concurrent users without downtime.
* Response times for user actions (e.g., booking a room) should be under 2 seconds.

#### **3.4 Design Constraints**

* Developed as a web-based application with mobile-friendly design.
* Data encryption to ensure secure communication.

#### **3.5 Security Requirements**

* Role-based access control to restrict features by role.
* User authentication with multi-factor authentication (MFA).
* Encryption of sensitive data, such as payment details and guest profiles.

#### **3.6 Maintainability Requirements**

* Modular code structure for ease of future enhancements.
* Comprehensive documentation for setup, usage, and maintenance.

#### **3.7 Reliability Requirements**

* Ensure 99.9% uptime during peak booking seasons.
* Implement redundant storage for guest and booking data.

#### **3.8 Availability Requirements**

* Available 24/7 for users globally.
* Scheduled maintenance during off-peak hours.

#### **3.9 Database Requirements**

* Store user, booking, and financial data in a secure database.
* Perform regular backups to prevent data loss.

#### **3.10 Documentation Requirements**

* User manuals for staff and guests.
* Technical documentation for developers and admins.

#### **3.11 Safety Requirements**

* Ensure compliance with data privacy laws.
* Validate data inputs to prevent errors and unauthorized access.

#### **3.12 Operational Requirements**

* Compatible with commonly used web browsers.
* Provide a simple and intuitive interface for all users.

#### **3.13 Site Adaptation**

* Accessible via web browsers on various devices.
* Adaptable to new hotel services and operations.

This structure can be further refined to match specific requirements or project scope.

10. Draw any one UML Diagram for Hotel Management System.

11. Create Project Management Process for Hotel Management System using Agile tools and techniques.  
**Agile Methodology Case Study — Hotel Management System**

**Inception of Agile Methodology:** Traditional Waterfall projects often faced challenges due to rigid requirements and lack of adaptability. Agile methodology emerged as a solution to address these challenges by incorporating iterative and incremental approaches, fostering a flexible and adaptive environment.

* **Iterative Development:** Progress is made through successive refinements, incorporating feedback at various stages to align with customer requirements.
* **Incremental Development:** The system is built and delivered in functional increments, with each stage being fully coded, tested, and deployable.

**Iterative and Incremental Development Model** The Agile Manifesto promotes a new way of managing software projects by prioritizing:

* Individuals and interactions over processes and tools
* Working software over comprehensive documentation
* Customer collaboration over contract negotiation
* Responding to change over following a plan

This approach transformed the management and execution of software development projects.

### **About the System — Hotel Management System**

A **Hotel Management System (HMS)** streamlines hotel operations by managing room bookings, customer records, billing, inventory, and staff schedules. The product vision is to create an efficient, user-friendly platform for hotel administrators, staff, and guests to interact seamlessly.

### **Implementation of Agile Methodology in Hotel Management System**

**Framework:** The **Scrum Framework** is used for the development of the HMS, emphasizing short sprints and iterative delivery. Each sprint delivers a potentially shippable product increment.

#### **Scrum Team for Hotel Management System**

1. **Product Owner:**
   * Defines and prioritizes the product backlog.
   * Ensures development aligns with business goals.
   * Maximizes ROI by delivering high-value features iteratively.
2. **Scrum Master:**
   * Facilitates Scrum events such as daily stand-ups, sprint reviews, and retrospectives.
   * Removes impediments to ensure smooth progress.
3. **Development Team:**
   * Self-organizing team responsible for delivering sprint tasks.
   * Includes developers, designers, and testers.

#### **Typical Scrum Implementation**

* **Team Size:** 5 to 9 members
* **Sprint Duration:** 2 to 4 weeks
* **Sprint Backlog:** Prioritized user stories divided into tasks for the sprint.

### **Feature List**

The following features are identified for the Hotel Management System:

1. **User Registration and Login**
2. **Room Booking and Management**
3. **Inventory and Services Management**
4. **Billing and Payment Processing**
5. **Reports and Analytics**

### **Feature Implementation Phases**

#### **Feature 1: User Registration and Login**

* **Ideation:** Users register as administrators, staff, or guests. Login credentials are verified, and access is role-based. Password recovery functionality is also required.
* **Design:**
  + Registration form with role selection
  + Login page with secure authentication
  + Admin panel for user management
* **Implementation & Testing:**
  + Backend functionality for user management and role-based access control.
  + Testing ensures secure registration, login, and password recovery.
* **Deployment:** The user registration and login feature is deployed, enabling secure user access.

#### **Feature 2: Room Booking and Management**

* **Ideation:** Guests can view room availability and book rooms online. Administrators can manage bookings, cancellations, and availability in real time.
* **Design:**
  + Room availability calendar
  + Booking form with check-in and check-out dates
  + Admin dashboard for booking management
* **Implementation & Testing:**
  + Backend development for managing bookings and real-time availability updates.
  + Testing ensures accurate booking functionality and seamless user experience.
* **Deployment:** The room booking and management feature is deployed, enabling efficient room reservation.

#### **Feature 3: Inventory and Services Management**

* **Ideation:** Administrators can manage inventory such as food supplies, toiletries, and other resources. Guests can request additional services (e.g., housekeeping, dining).
* **Design:**
  + Inventory dashboard for admins
  + Guest interface for requesting services
* **Implementation & Testing:**
  + Logic for managing inventory levels and service requests.
  + Testing ensures accurate updates and service fulfillment.
* **Deployment:** The inventory and services management feature is deployed, streamlining resource management.

#### **Feature 4: Billing and Payment Processing**

* **Ideation:** Guests can view and settle bills for room bookings and additional services. Multiple payment methods (e.g., credit card, digital wallets) are supported.
* **Design:**
  + Invoice generation system
  + Secure payment gateway integration
* **Implementation & Testing:**
  + Development of billing algorithms and payment processing.
  + Testing ensures secure transactions and accurate billing.
* **Deployment:** The billing and payment processing feature is deployed, enabling smooth financial transactions.

#### **Feature 5: Reports and Analytics**

* **Ideation:** Administrators can generate reports on occupancy rates, revenue, and service usage. Analytics provide insights for better decision-making.
* **Design:**
  + Dashboard for visualizing key metrics
  + Exportable reports in various formats
* **Implementation & Testing:**
  + Development of reporting and analytics tools.
  + Testing ensures accuracy and usability of reports.
* **Deployment:** The reports and analytics feature is deployed, enabling data-driven decision-making.

### **Summary of Agile Implementation**

The Hotel Management System is developed using Agile methodology, emphasizing adaptability, collaboration, and iterative delivery. Each feature undergoes ideation, design, implementation, testing, and deployment, ensuring a robust and user-friendly platform for hotel management.

12. Estimate the cost for implementation of Hotel Management System.

### **COCOMO Effort Estimation for Hotel Management System**

### **1. Define the Project Scope**

The Hotel Management System involves the following components:

1. **User Interface:**
   * Interfaces for guests, administrators, and staff.
   * Online booking, room selection, and payment portals.
2. **Booking and Room Management:**
   * Room availability, booking, cancellation, and scheduling.
   * Integration with room cleaning and maintenance schedules.
3. **Billing and Payment System:**
   * Invoicing, refunds, and tracking of payments.
   * Integration with multiple payment gateways.
4. **Backend Database:**
   * Stores data for guests, bookings, payments, and rooms.
5. **Security Features:**
   * Role-based access for admins, receptionists, and housekeeping staff.
   * Data encryption for sensitive information.
6. **APIs for Integration:**
   * Integration with external systems for OTAs (Online Travel Agencies), loyalty programs, and local attractions.

This system is categorized as a **semi-detached project** due to its moderate complexity and combination of custom and reusable components.

### **2. COCOMO Basic Effort Equation**

The basic effort equation is:  
 E=a×(KLOC)bE = a \times (\text{KLOC})^b  
 For a semi-detached project:

* a=3.0a = 3.0
* b=1.12b = 1.12

### **3. Estimate KLOC (Thousands of Lines of Code)**

Estimated size of the Hotel Management System:

* User Interface and Logic: 20 KLOC
* Backend Integration and APIs: 25 KLOC
* Billing and Payment Features: 10 KLOC
* Security and Role Management: 5 KLOC  
   **Total Estimated Size:** 60 KLOC

### **4. Calculate Effort (E)**

Using the equation:  
 E=3.0×(KLOC)1.12E = 3.0 \times (\text{KLOC})^{1.12}  
 Substitute the values:  
 E=3.0×(60)1.12E = 3.0 \times (60)^{1.12}  
 **Effort (E):** Approximately **194 person-months (PM)**

### **5. Estimate Development Time (T)**

The development time equation is:  
 T=c×(Effort)dT = c \times (\text{Effort})^d  
 For semi-detached projects:

* c=2.5c = 2.5
* d=0.35d = 0.35

Substitute the values:  
 T=2.5×(194)0.35T = 2.5 \times (194)^{0.35}  
 **Development Time (T):** Approximately **18.1 months**

### **6. Estimate Cost**

**Average Monthly Cost of a Software Developer:** Monthly Cost=$10,000×₹83=₹8,30,000\text{Monthly Cost} = \$10,000 \times ₹83 = ₹8,30,000

**Total Development Cost:** Cost=Effort×Monthly Cost\text{Cost} = \text{Effort} \times \text{Monthly Cost}  
 Cost=194×₹8,30,000\text{Cost} = 194 \times ₹8,30,000  
 **Total Development Cost:** ₹16,10,20,000/-

### **7. Additional Considerations**

1. **Hardware and Deployment Costs:**
   * Server and networking hardware for hosting the system.
   * Estimated cost: ₹1 crore to ₹5 crore, depending on deployment scale.
2. **Operational Costs:**
   * Annual maintenance and updates: 10-20% of development cost (~₹1.6 to ₹3.2 crore/year).

### **Final Estimate for Hotel Management System**

* **Development Cost (COCOMO):** ₹16.10 crore
* **Hardware and Deployment:** ₹1 crore to ₹5 crore
* **Total Initial Cost:** ₹17.10 crore to ₹21.10 crore