# Use Case Diagram for Library Management System:

A use case diagram in UML helps to show the various ways in which a user could interact with a system. For a Library Management System, the use case diagram helps visualize the interactions between users (actors) and the system’s functionalities (use cases). This diagram provides a clear, simplified way to understand how the system operates and what it offers to its users.

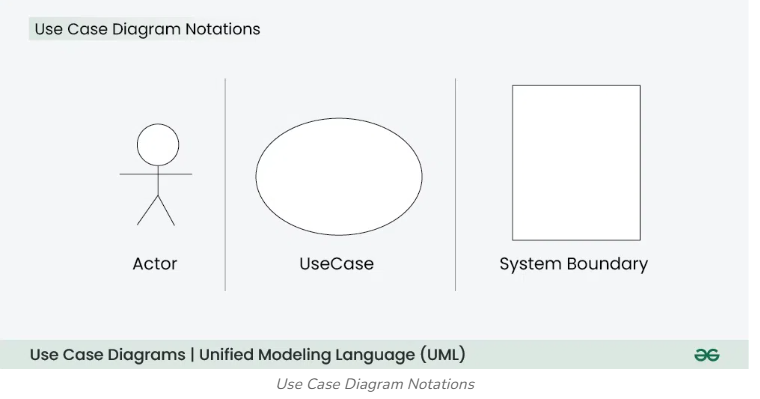
## **What is a Use case diagram?**

Use Case Diagram, referred to as a Behaviour model or diagram. It simply describes and displays the relationship or interaction between the users or customers and providers of the application service or the system. It describes different actions that a system performs in collaboration to achieve something with one or more users of the system. A use-case diagram is used a lot nowadays to manage the system.

## Use Case Diagram Notations

Use Case Diagram consists of the following components:

1. **Actor:**Actors are external entities that interact with the system. These can include users, other systems, or hardware devices. In the context of a Use Case Diagram, actors initiate use cases and receive the outcomes.
2. **Use Case:** Use cases are like scenes in the play. They represent specific things your system can do.
3. **System Boundary:** The system boundary is a visual representation of the scope or limits of the system you are modelling. It defines what is inside the system and what is outside.

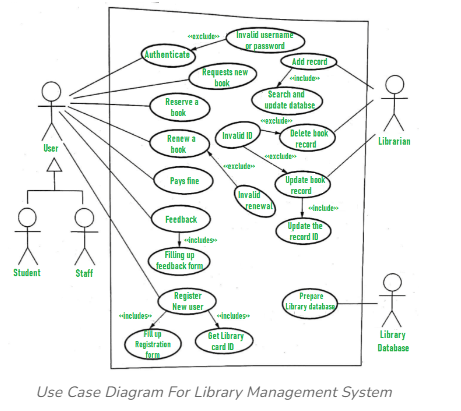


## **Use Case Diagram for Library Management System**

Let’s visually map out the relationships and interactions. Below is the textual description of what the diagram would look like:

1. **Actors:**
   * User (Staff or Student)
   * Librarian
2. **Use Cases:**
   * Register New User
   * Issue Library Card
   * Request New Book
   * Reserve Book
   * Renew Book
   * Pay Fine
   * Fill Feedback Form
   * Manage Records
   * Delete Records
   * Update Database
3. **System Boundary:**
   * The system boundary will encompass all the use cases mentioned above.

### **Below is the use case diagram of a Library Management System**



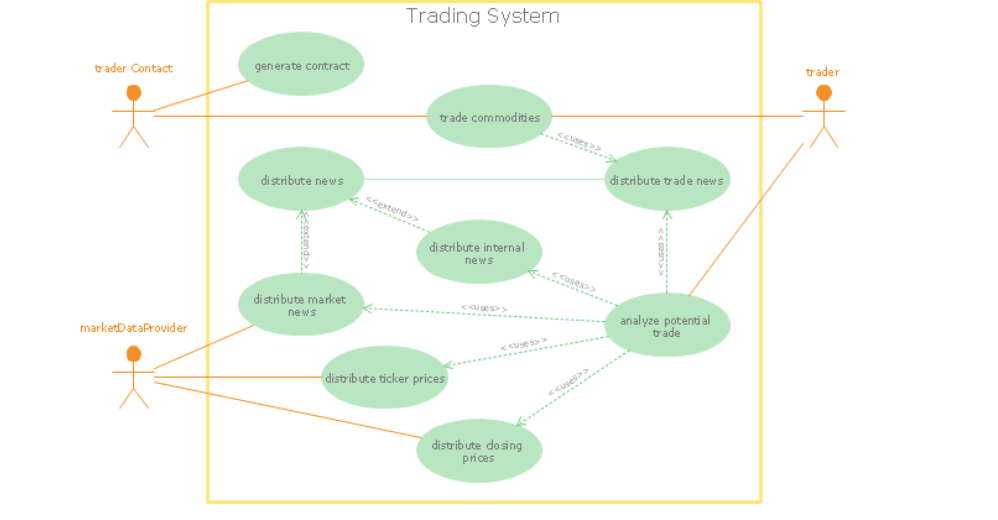
### **Explanation of Use Case Diagram of a Library Management System:**

Here, we will understand the designing use case diagram for the library management system. Some scenarios of the system are as follows:

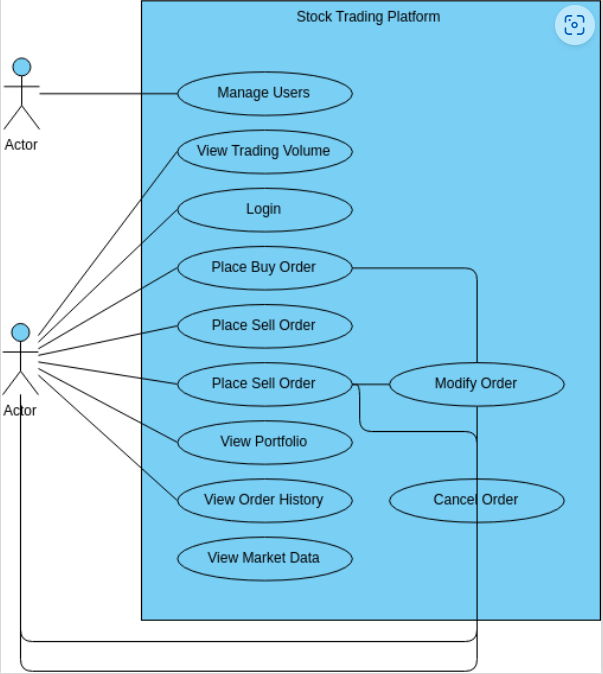
1. User who registers himself as a new user initially is regarded as staff or student for the library system.
   * For the user to get registered as a new user, registration forms are available that is needed to be fulfilled by the user.
   * After registration, a library card is issued to the user by the librarian. On the library card, an ID is assigned to cardholder or user.
2. After getting the library card, a new book is requested by the user as per there requirement.
3. After, requesting, the desired book or the requested book is reserved by the user that means no other user can request for that book.
4. Now, the user can renew a book that means the user can get a new due date for the desired book if the user has renewed them.
5. If the user somehow forgets to return the book before the due date, then the user pays fine. Or if the user forgets to renew the book till the due date, then the book will be overdue and the user pays fine.
6. User can fill the feedback form available if they want to.
7. Librarian has a key role in this system. Librarian adds the records in the library database about each student or user every time issuing the book or returning the book, or paying fine.
8. Librarian also deletes the record of a particular student if the student leaves the college or passed out from the college. If the book no longer exists in the library, then the record of the particular book is also deleted.
9. Updating database is the important role of Librarian.

# Stock Trading Platform:

# [UML use case diagram - Trading system usage scenarios](https://conceptdraw.com/a648c3/preview/640)



"Algorithmic trading, also called automated trading, black-box trading, or algo trading, is the use of electronic platforms for entering trading orders with an algorithm which executes pre-programmed trading instructions whose variables may include timing, price, or quantity of the order, or in many cases initiating the order by a "robot", without human intervention. Algorithmic trading is widely used by investment banks, pension funds, mutual funds, and other buy-side (investor-driven) institutional traders, to divide large trades into several smaller trades to manage market impact and risk. Sell side traders, such as market makers and some hedge funds, provide liquidity to the market, generating and executing orders automatically.  
A special class of algorithmic trading is "high-frequency trading" (HFT), which is often most profitable during periods of high market volatility. During the past years, companies such as Algorates have employed HFT strategies, recording high profits even during periods in which the markets have seen steep declines." [Algorithmic trading. Wikipedia]  
The UML use case diagram example "Trading system usage scenarios" was created using the ConceptDraw PRO diagramming and vector drawing software extended with the Rapid UML solution from the Software Development area of ConceptDraw Solution Park.



**Sure! Here are the detailed use cases for buying a stock using an online brokerage account:**

### Actors

1. **User**: The person who wants to buy stocks.
2. **Online Brokerage System**: The platform that facilitates stock trading.
3. **Stock Exchange**: The market where stocks are bought and sold.
4. **Bank**: The financial institution that handles the user’s funds.

### Use Cases

#### 1. **Create Account**

* **Description**: The user creates an account on the online brokerage platform.
* **Primary Actor**: User
* **Steps**:
  1. User navigates to the brokerage website.
  2. User fills in personal details (name, email, address, etc.).
  3. User sets up login credentials (username and password).
  4. User submits the form.
  5. System verifies the details and creates the account.
  6. System sends a confirmation email to the user.

#### 2. **Deposit Funds**

* **Description**: The user deposits money into their brokerage account.
* **Primary Actor**: User
* **Steps**:
  1. User logs into the brokerage account.
  2. User navigates to the “Deposit Funds” section.
  3. User selects the deposit method (bank transfer, credit card, etc.).
  4. User enters the amount to deposit.
  5. User confirms the transaction.
  6. System processes the deposit and updates the account balance.

#### 3. **Search for Stock**

* **Description**: The user searches for a specific stock to buy.
* **Primary Actor**: User
* **Steps**:
  1. User logs into the brokerage account.
  2. User navigates to the “Search” section.
  3. User enters the stock symbol or company name.
  4. System displays the search results.
  5. User selects the desired stock from the results.

#### 4. **View Stock Details**

* **Description**: The user views detailed information about a selected stock.
* **Primary Actor**: User
* **Steps**:
  1. User selects a stock from the search results.
  2. System displays detailed information (current price, historical data, company info, etc.).

#### 5. **Place Buy Order**

* **Description**: The user places an order to buy a stock.
* **Primary Actor**: User
* **Steps**:
  1. User navigates to the stock’s detail page.
  2. User clicks on the “Buy” button.
  3. User enters the number of shares to buy.
  4. User selects the order type (market, limit, etc.).
  5. User reviews the order details.
  6. User confirms the order.
  7. System sends the order to the stock exchange.
  8. Stock exchange processes the order.
  9. System updates the user’s portfolio and account balance.

#### 6. **Receive Order Confirmation**

* **Description**: The user receives confirmation of the executed order.
* **Primary Actor**: User
* **Steps**:
  1. System sends a notification to the user about the order status.
  2. User receives the notification (email, SMS, app notification).

#### 7. **View Portfolio**

* **Description**: The user views their stock portfolio.
* **Primary Actor**: User
* **Steps**:
  1. User logs into the brokerage account.
  2. User navigates to the “Portfolio” section.
  3. System displays the user’s current holdings and their values.

#### 8. **Withdraw Funds**

* **Description**: The user withdraws money from their brokerage account.
* **Primary Actor**: User
* **Steps**:
  1. User logs into the brokerage account.
  2. User navigates to the “Withdraw Funds” section.
  3. User selects the withdrawal method.
  4. User enters the amount to withdraw.
  5. User confirms the transaction.
  6. System processes the withdrawal and updates the account balance.
  7. Bank processes the withdrawal and transfers the funds to the user’s bank account.

These use cases cover the primary activities involved in buying a stock using an online brokerage account. If you need more details or have any specific questions, feel free to ask!

# Design an Online Stock Brokerage System

### **Let's design an Online Stock Brokerage System**

**We'll cover the following:**

* [System Requirements](https://github.com/wyaadarsh/Grokking-OOD/blob/master/object-oriented-design-case-studies/design-an-online-stock-brokerage-system.md#system-requirements)
* [Use Case Diagram](https://github.com/wyaadarsh/Grokking-OOD/blob/master/object-oriented-design-case-studies/design-an-online-stock-brokerage-system.md#use-case-diagram)
* [Class Diagram](https://github.com/wyaadarsh/Grokking-OOD/blob/master/object-oriented-design-case-studies/design-an-online-stock-brokerage-system.md#class-diagram)
* [Activity Diagrams](https://github.com/wyaadarsh/Grokking-OOD/blob/master/object-oriented-design-case-studies/design-an-online-stock-brokerage-system.md#activity-diagrams)
* [Code](https://github.com/wyaadarsh/Grokking-OOD/blob/master/object-oriented-design-case-studies/design-an-online-stock-brokerage-system.md#code)

An Online Stock Brokerage System facilitates its users the trade (i.e. buying and selling) of stocks online. It allows clients to keep track of and execute their transactions, and shows performance charts of the different stocks in their portfolios. It also provides security for their transactions and alerts them to pre-defined levels of changes in stocks, without the use of any middlemen.

The online stock brokerage system automates traditional stock trading using computers and the internet, making the transaction faster and cheaper. This system also gives speedier access to stock reports, current market trends, and real-time stock prices.

[](https://github.com/wyaadarsh/Grokking-OOD/blob/master/media-files/stock-brokerage-system.png)  
Online Stock Brokerage System

### System Requirements

We will focus on the following set of requirements while designing the online stock brokerage system:

1. Any user of our system should be able to buy and sell stocks.
2. Any user can have multiple watchlists containing multiple stock quotes.
3. Users should be able to place stock trade orders of the following types: 1) market, 2) limit, 3) stop loss and, 4) stop limit.
4. Users can have multiple ‘lots’ of a stock. This means that if a user has bought a stock multiple times, the system should be able to differentiate between different lots of the same stock.
5. The system should be able to generate reports for quarterly updates and yearly tax statements.
6. Users should be able to deposit and withdraw money either via check, wire, or electronic bank transfer.
7. The system should be able to send notifications whenever trade orders are executed.

### Use Case Diagram

We have three main Actors in our system:

* **Admin:** Mainly responsible for administrative functions like blocking or unblocking members.
* **Member:** All members can search the stock inventory, as well as buy and sell stocks. Members can have multiple watchlists containing multiple stock quotes.
* **System:** Mainly responsible for sending notifications for stock orders and periodically fetching stock quotes from the stock exchange.

Here are the top use cases of the Stock Brokerage System:

* **Register new account/Cancel membership:** To add a new member or cancel the membership of an existing member.
* **Add/Remove/Edit watchlist:** To add, remove or modify a watchlist.
* **Search stock inventory:** To search for stocks by their symbols.
* **Place order:** To place a buy or sell order on the stock exchange.
* **Cancel order:** Cancel an already placed order.
* **Deposit/Withdraw money:** Members can deposit or withdraw money via check, wire or electronic bank transfer.

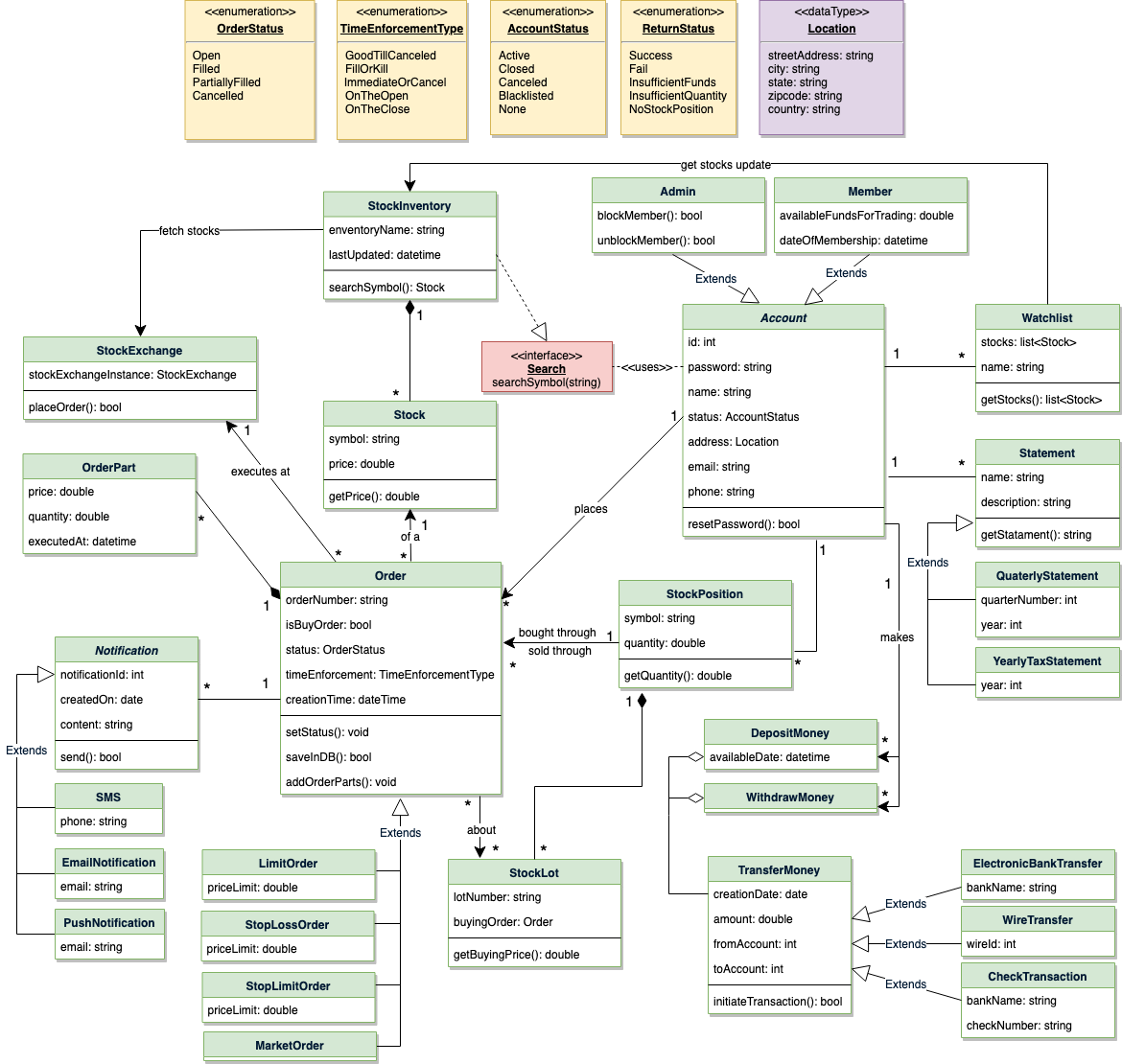
Here is the use case diagram of an Online Stock Brokerage System:

Use Case Diagram for Online Stock Brokerage System

### **Class Diagram**

Here are the main classes of our Online Stock Brokerage System:

* **Account:** Consists of the member’s name, address, e-mail, phone, total funds, funds that are available for trading, etc. We’ll have two types of accounts in the system: one will be a general member, and the other will be an Admin. The Account class will also contain all the stocks the member is holding.
* **Stock Exchange:** The stockbroker system will fetch all stocks and their current prices from the stock exchange. Stock Exchange will be a singleton class encapsulating all interactions with the stock exchange. This class will also be used to place stock trading orders on the stock exchange.
* **Stock:** The basic building block of the system. Every stock will have a symbol, current trading price, etc.
* **Stock Inventory:** This class will fetch and maintain the latest stock prices from the Stock Exchange. All system components will read the most recent stock prices from this class.
* **Watchlist:** A watchlist will contain a list of stocks that the member wants to follow.
* **Order:** Members can place stock trading orders whenever they would like to sell or buy stock positions. The system would support multiple types of orders:
  + **Market Order:** Market order will enable users to buy or sell stocks immediately at the current market price.
  + **Limit Order:** Limit orders will allow a user to set a price at which they want to buy or sell a stock.
  + **Stop Loss Order:** An order to buy or sell once the stock reaches a certain price.
  + **Stop Limit Order:** The stop-limit order will be executed at a specified price or better after a given stop price has been reached. Once the stop price is reached, the stop-limit order becomes a limit order to buy or sell at the limit price or better.
* **Order Part:** An order could be fulfilled in multiple parts. For example, a market order to buy 100 stocks could have one part containing 70 stocks at $10 and another part with 30 stocks at $10.05.
* **Stock Lot:** Any member can buy multiple lots of the same stock at different times. This class will represent these individual lots. For example, the user could have purchased 100 shares of AAPL yesterday and 50 more stocks of AAPL today. While selling, users will be able to select which lot they want to sell first.
* **Stock Position:** This class will contain all the stocks that the user holds.
* **Statement:** All members will have reports for quarterly updates and yearly tax statements.
* **Deposit Money & Withdraw Money:** Members will be able to move money through check, wire or electronic bank transfers.
* **Notification:** Will take care of sending notifications to members.

[](https://github.com/wyaadarsh/Grokking-OOD/blob/master/media-files/sbs-class-diagram.png)  
**Class Diagram for Online Stock Brokerage System**

**UML for Online Stock Brokerage System**

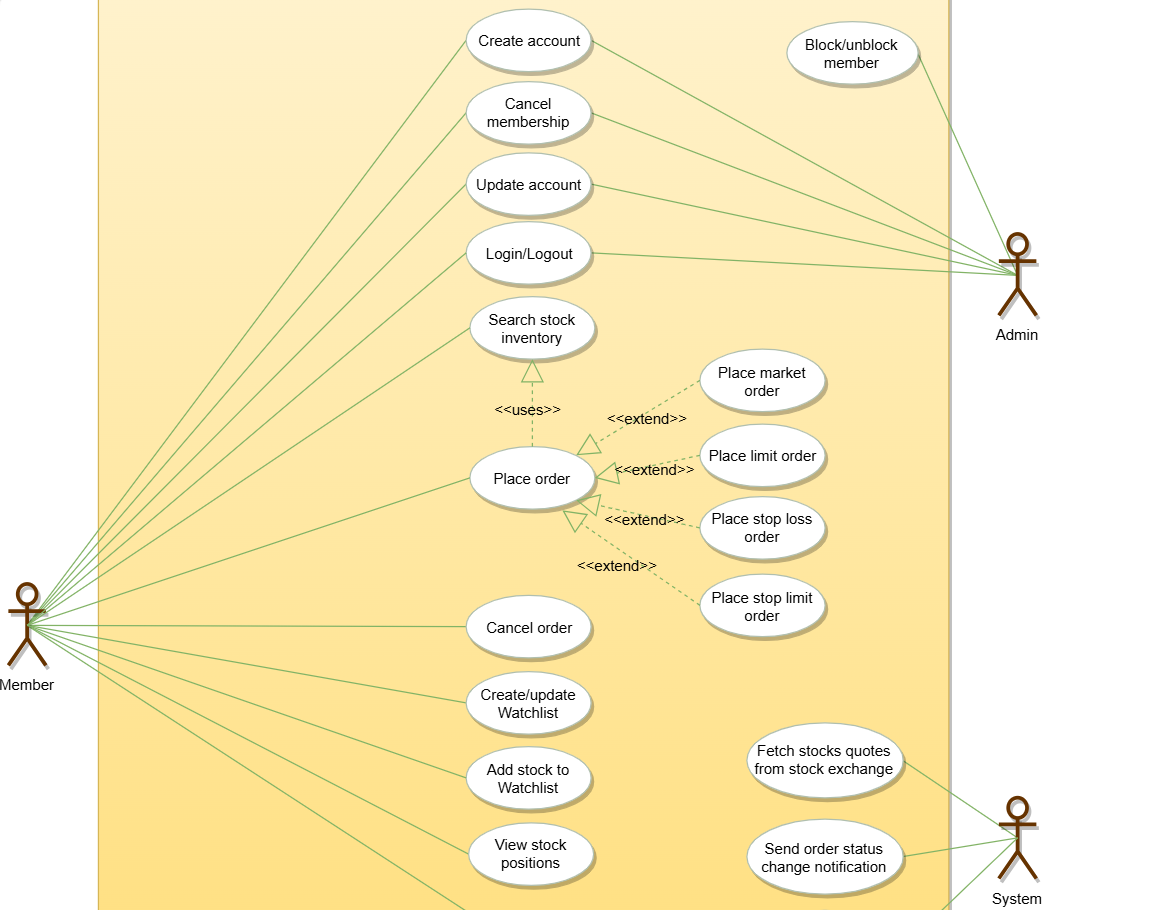
### **Activity Diagrams**

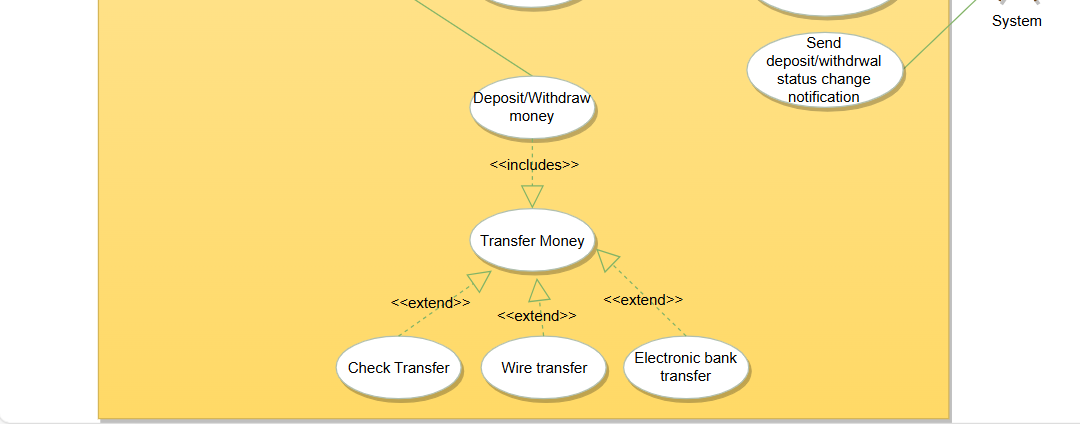
**Place a buy order:** Any system user can perform this activity. Here are the steps to place a buy order:

**Activity Diagram for Online Stock Brokerage System Buy Order**

**Place a sell order:** Any system user can perform this activity. Here are the steps to place a buy order:

Activity Diagram for Online Stock Brokerage System Sell Order





**Using credit card at a Restaurant**

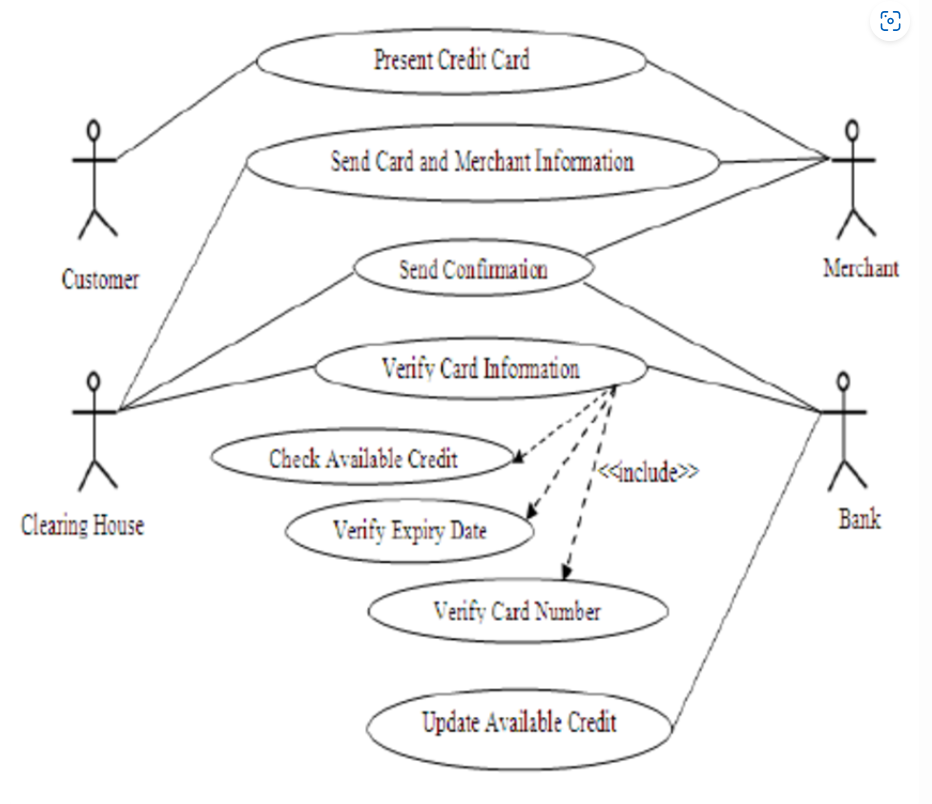
### Use Case 3: Using Credit Card at a Restaurant

**Actors:**

* Customer
* Waiter
* Payment Processing System

**Use Cases:**

1. **Place Order**
   * **Actor:** Customer, Waiter
   * **Precondition:** Customer is seated at the restaurant.
   * **Description:** The customer places an order for food and beverages with the waiter.
   * **Postcondition:** The order is placed and sent to the kitchen.
2. **Request Bill**
   * **Actor:** Customer, Waiter
   * **Precondition:** Customer has finished the meal.
   * **Description:** The customer requests the bill from the waiter.
   * **Postcondition:** The waiter generates the bill and presents it to the customer.
3. **Provide Credit Card**
   * **Actor:** Customer, Waiter
   * **Precondition:** Customer has received the bill.
   * **Description:** The customer provides their credit card to the waiter for payment.
   * **Postcondition:** The waiter takes the credit card to process the payment.
4. **Process Payment**
   * **Actor:** Waiter, Payment Processing System
   * **Precondition:** The waiter has the customer's credit card.
   * **Description:** The waiter swipes or inserts the credit card into the payment terminal, and the payment processing system authorizes the transaction.
   * **Postcondition:** Payment is approved or declined, and a receipt is printed.
5. **Return Credit Card and Receipt**
   * **Actor:** Waiter, Customer
   * **Precondition:** Payment has been processed.
   * **Description:** The waiter returns the credit card and a copy of the receipt to the customer.
   * **Postcondition:** The customer receives their credit card and receipt, completing the payment process.
6. **Sign Receipt**
   * **Actor:** Customer
   * **Precondition:** Payment has been approved.
   * **Description:** The customer signs the receipt if required.
   * **Postcondition:** The signed receipt is returned to the waiter.
7. **Provide Tip**
   * **Actor:** Customer
   * **Precondition:** Customer has received the final bill and credit card.
   * **Description:** The customer adds a tip to the receipt or provides it in cash.
   * **Postcondition:** The tip is given to the waiter.



**Suggest who might be stakeholders in a hospital In a Hospital Management System (HMS), stakeholders include a variety of individuals and groups, each with unique interests and needs. Here are some key stakeholders:**

1. **Patients**: They are the primary beneficiaries of healthcare services and expect efficient, high-quality care.
2. **Healthcare Providers**: This group includes doctors, nurses, and other medical staff who use the system to manage patient care and medical records.
3. **Hospital Administrators**: They oversee hospital operations and require the system for scheduling, resource management, and compliance with regulations.
4. **IT Staff**: Responsible for maintaining the HMS, ensuring it is secure, up-to-date, and functioning correctly.
5. **Insurance Companies**: They need access to patient data for billing and claims processing.
6. **Government and Regulatory Bodies**: They ensure the hospital complies with healthcare regulations and standards.
7. **Suppliers**: They provide medical supplies and equipment, and their interactions with the hospital are often managed through the HMS.
8. [**Financial Stakeholders**: This includes the hospital’s financial department, investors, and donors who are interested in the financial health and efficiency of the hospital](https://www.stakeholdermap.com/stakeholders-in-hospitals.html)[1](https://www.stakeholdermap.com/stakeholders-in-hospitals.html)[2](https://www.healthcareis.com/blog/how-to-identify-key-stakeholders-for-a-healthcare-it-project)[3](https://www.supsalv.org/what-are-stakeholders-in-healthcare/).

### Why Conflicts Are Inevitable

Conflicts among stakeholders’ requirements are almost inevitable due to the following reasons:

1. **Diverse Objectives**: Each stakeholder group has different goals. For example, patients want quick and effective treatment, while administrators focus on cost efficiency and regulatory compliance.
2. **Resource Allocation**: Limited resources such as time, money, and personnel can lead to conflicts. For instance, IT staff may prioritize system security, which might slow down system performance, affecting healthcare providers’ efficiency.
3. **Data Privacy vs. Accessibility**: Patients and regulatory bodies emphasize data privacy, while healthcare providers and insurance companies need easy access to patient data for effective treatment and billing.
4. **Change Resistance**: Different stakeholders may resist changes for various reasons. Healthcare providers might resist new workflows introduced by the HMS, while IT staff might push for updates to improve system security.
5. **Communication Gaps**: Misunderstandings and lack of communication between stakeholders can lead to conflicts. [For example, if IT staff are not fully aware of the healthcare providers’ needs, they might implement features that are not user-friendly](https://www.stakeholdermap.com/stakeholders-in-hospitals.html)[4](https://www.med.unc.edu/ihqi/resources/stakeholder-analysis/)[5](https://www.instahms.com/blog/hospital-management-system-benefits-for-its-stakeholders).

Balancing these diverse requirements requires careful planning, effective communication, and ongoing stakeholder engagement to ensure the HMS meets the needs of all parties involved.

management system. Explain why it is almost inevitable that the requirements of different stakeholders will conflict in some way.