```
import getpass
import os

GROQ_API_KEY = "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"
```

```
%pip install -qU langchain-groq
```

```
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.9/121.9 kB 4.0 MB/s eta 0:00:00
```

```
from langchain_groq import ChatGroq

llm = ChatGroq(
    model="deepseek-r1-distill-qwen-32b",
    temperature=0,
    max_tokens=None,
    timeout=None,
    max_retries=2,
    # other params...
)
```

```
messages = [
    (
        "system",
        "You are a helpful assistant that translates English to French.
        Translate the user sentence.",
    ),
    ("human", "I love programming."),
]
ai_msg = llm.invoke(messages)
ai_msg
```

```
AIMessage(content='<think>\nAlright, the user wants me to translate "I love
programming." into French. Let me break this down.\n\nFirst, "I love" in French is
"J\'aime." That\'s straightforward.\n\nNext, "programming." The French word for that
is "la programmation." So, putting it together, it should be "J\'aime la
programmation."\n\nI should make sure the sentence structure is correct. In French,
the definite article "la" is used before "programmation," so it\'s important to
include that.\n\nAlso, considering the context, the user is likely expressing a
passion for programming, so the translation should convey that enthusiasm
accurately.\n\nI don\'t think there are any regional variations or slang terms
needed here. Keeping it formal and clear is best.\n\nDouble-checking the
translation: "J\'aime la programmation." Yep, that sounds right. No mistakes
there.\n\nSo, the final translation is "J\'aime la
programmation."\n</think>\n\nJ\'aime la programmation.', additional_kwargs={},
response_metadata={'token_usage': {'completion_tokens': 206, 'prompt_tokens': 22,
'total_tokens': 228, 'completion_time': 0.53052133, 'prompt_time': 0.003526423,
'queue_time': 0.020665375, 'total_time': 0.534047753}, 'model_name': 'deepseek-r1-
distill-qwen-32b', 'system_fingerprint': 'fp_8cd3bdd003', 'finish_reason': 'stop',
'logprobs': None}, id='run-f82d5787-796b-434b-b995-db1e405f7c97-0', usage_metadata=
{'input_tokens': 22, 'output_tokens': 206, 'total_tokens': 228})
```

Identifying the stakeholders and end-users from the problem statement

```
problem_statement = f"""NewYork Metro station wants to establish a TicketDistributor mach
travelling in metro rails. Travelers have options of selecting a ticket for a single trip
multiple trips. They can also issue a metro pass for regular passengers or a time card fo
a month according to their requirements. The discounts on tickets will be provided to fre
passengers. The machine is also supposed to read the metro pass and time cards issued by
counters or machine. The ticket rates differ based on whether the traveler is a child or
machine is also required to recognize original as well as fake currency notes. The typica
payment method of either cash, credit/debit card or smartcard. The ticket or tickets are
dispensed to the user. Also the messaging facilities after every transaction are required
number. The system can also be operated comfortably by a touch-screen. A large number of
components are to be used. We do not want our system to slow down, and also usability of
The TicketDistributor must be able to handle several exceptions, such as aborting the tra
incomplete transactions, insufficient amount given by the travelers to the machine, money
of aborted transaction, change return after successful transaction, showing insufficient
card, updated information printed on the tickets e.g. departure time, date, time, price,
till, validity duration, ticket issued from and destination station. In case of exception
to be displayed. We do not want user feedback after every development stage but after eve
to save time. The machine is required to work in a heavy load environment such that at th
evening time in weekdays, and in weekends performance and efficiency would not get affect

co_analyst_context = """ You are a requirements analyst powered by AI along with a human

prompt = """Assume you are a requirement analyst responsible for developing requirements
system.
{co_analyst_context}
Using the problem statement below, generate the key stakeholders and end-users.

Problem Statement:
{problem_statement}
 """

stakeholders = llm.invoke(prompt)
print(stakeholders.content)
```

```
<think>
Okay, so I need to figure out the key stakeholders and end-users for the MetroTicketD

The problem statement mentions that the MetroTicketDistributor system is designed to

Now, I need to identify the stakeholders and end-users. Stakeholders are individuals

Starting with stakeholders, I can think of the Metro Authority as a primary stakeholc

The ticket distributor is another stakeholder. They are responsible for selling the t

Metro passengers are the end-users, but they are also a stakeholder because the syste

Metro staff, such as station attendants and customer service representatives, are als

The IT department of the metro system is another stakeholder. They are responsible fc

The ticketing system vendor, if the system is being developed by an external company,

Now, moving on to end-users. The primary end-users are the metro passengers who purch
```

```
Metro staff, such as station attendants, are also end-users. They use the system to a

Customer service representatives are another group of end-users. They use the system

Additionally, system administrators are end-users who manage the backend of the syste

I should also consider if there are any other potential stakeholders or end-users. Fo

To summarize, the key stakeholders are the Metro Authority, ticket distributor, metro

I should make sure that each stakeholder and end-user is clearly defined and that the
</think>

**Key Stakeholders and End-Users for the MetroTicketDistributor System**

**Stakeholders:**
1. **Metro Authority:** Manages the metro system and ensures the ticket distribution
2. **Ticket Distributor:** Responsible for ticket sales, concerned with user-friendli
3. **Metro Passengers:** End-users who purchase tickets, seeking an efficient, reliab
4. **Metro Staff:** Includes station attendants and customer service representatives
5. **IT Department:** Ensures the system is secure, scalable, and well-integrated wit
6. **Ticketing System Vendor:** Involved in development, testing, and maintenance, en

**End-Users:**
1. **Metro Passengers:** Directly interact with the user interface to purchase ticket
2. **Metro Staff:** Utilize the system to assist passengers and manage operations, ne
3. **Customer Service Representatives:** Address inquiries and resolve issues, requir
4. **System Administrators:** Manage the backend, handling user accounts, monitoring

This breakdown ensures that the MetroTicketDistributor system is developed to meet th
```

Eliciting requirements from the stakeholders using different elicitation techniques

```python
from langchain_core.prompts import ChatPromptTemplate

prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses elicitation
            techniques to get requirements from {stakeholders} and list those
            elicitation techniques and justify why and how the chosen
            elicitation techniques are helpful in gathering requirements.
            Do it in the following format :
            Elicitation technique name -->
            Why :
            How :
            """
        ),
        ("human", "{input}"),
    ]
)

chain = prompt | llm
```

```
elicitation_tech = chain.invoke(
    {
     "stakeholders" : stakeholders,
     "input" : problem_statement,
    }
)

print(elicitation_tech.content)
```

```
<think>
Okay, so I need to figure out the elicitation techniques for gathering requirements f

First, I should think about who the stakeholders are. There are the Metro Authority,

Now, what elicitation techniques can I use? I remember that there are several methods

Interviews seem useful because they allow direct communication with stakeholders. I c

Workshops could be good for bringing together different stakeholders to discuss requi

Questionnaires might be useful for gathering quantitative data, especially from passe

Document analysis could help by reviewing existing Metro policies, ticketing systems,

Prototyping could be effective here. Creating a prototype of the ticket machine inter

Considering the project's need for efficiency and user-friendliness, prototyping seem

I should also think about the system's technical requirements, like handling high tra

In summary, using a combination of interviews, workshops, document analysis, and prot
</think>

**Elicitation Techniques for MetroTicketDistributor System Requirements**

1. **Interviews**
   - **Why:** Direct communication with stakeholders like the Metro Authority and pas
   - **How:** Conduct one-on-one interviews with key stakeholders to gather detailed

2. **Workshops**
   - **Why:** Facilitates collaborative discussions among diverse stakeholders, helpi
   - **How:** Organize workshops with representatives from the Metro Authority, IT de

3. **Questionnaires**
   - **Why:** Useful for gathering quantitative data, especially from passengers, to
   - **How:** Distribute questionnaires to passengers to collect feedback on usabilit

4. **Document Analysis**
   - **Why:** Provides a foundation by reviewing existing policies and systems, ensur
   - **How:** Analyze current Metro policies and ticketing systems to identify succes

5. **Prototyping**
   - **Why:** Allows stakeholders to visualize and interact with the system, providir
   - **How:** Develop a prototype of the ticket machine interface and gather feedback

**Conclusion:**
By combining interviews, workshops, document analysis, and prototyping, the project c
```

Generate User Stories

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses elicitation techniques {elicita
            The front of the card will have the following format :
            As a [user role], I want to [goal] so that I can [reason].
            The back of the card will contain Success and Failure scenario.
            Use {example} for format.
             """
        ),
        ("human", "{input}"),
    ]
)

example = """Front of the Card :
As a registered user I want to log in so that I can access subscriber-only content
Back of the Card :
1. Success - valid user logged in and referred to home page
(a) 'Remember me' ticked-store cookie/ automatic login next
time.
(b) 'Remember me' not ticked - force login next time.
2.   Failure-display message
(a) "Email address in wrong format"
(b) "Unrecognized user name, please try again"
(c) "Incorrect password, please try again"
(d) "Service unavailable, please try again"
(e) Account has expired - refer to account renewal sales page.
"""

chain = prompt | llm

generated_user_stories = chain.invoke(
    {
     "elicitation_tech" : elicitation_tech,
     "stakeholders" : stakeholders,
     "example" : example,
     "input" : problem_statement,
    }
)

print(generated_user_stories.content)
```

```
As a passenger, I want to view available payment methods so that I can choose the

**Back of the Card:**
1. Success Scenario:
   - Passenger selects payment method, system processes payment successfully.

2. Failure Scenarios:
   - Invalid payment method: Display error message and prompt re-entry of payment
   - System error: Display error message and offer alternative payment options.

**Front of the Card:**
As a system administrator, I want to manage system logs so that I can monitor and

**Back of the Card:**
1. Success Scenario:
   - Administrator accesses logs, reviews, and troubleshoots issues.

2. Failure Scenarios:
   - Log access denied: Display error message and prompt re-entry of credentials.
   - Corrupted logs: Display error message and prompt log restoration or re-entry.

**Front of the Card:**
As a passenger, I want to view available ticket types so that I can choose the mos

**Back of the Card:**
1. Success Scenario:
   - Passenger selects ticket type, system processes purchase successfully.

2. Failure Scenarios:
   - Invalid ticket type: Display error message and prompt re-entry of details or
   - System error: Display error message and offer alternative solutions.

**Front of the Card:**
As a system administrator, I want to manage system updates so that I can ensure th

**Back of the Card:**
1. Success Scenario:
   - Administrator schedules and performs updates without downtime.

2. Failure Scenarios:
   - Update failure: Display error message and prompt re-entry of update or rollba
   - System crash during update: Display error message and prompt restart or maint

**Front of the Card:**
As a passenger, I want to view available discounts so that I can benefit from prom

**Back of the Card:**
1. Success Scenario:
   - Passenger views available discounts and applies them during purchase
```

Validate User Stories (according to INVEST framework)

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses the INVEST framework to validat
            The INVEST framework rules are :
```

```
1. All the stories that have been broken down are mostly independent    , hence making su
2. Not every minute detail has been included in the cards, encouraging developers to seek
3. Stories have been written in regular business language that is equally valuable to bot
4. It is assumed that all the user stories mentioned below can be converted into working
5. It is also written in a way where the completion of the story and its working is quant
Give the validated user stories with this format :
The front of the card will have the following format :
            As a [user role], I want to [goal] so that I can [reason].
            The back of the card will contain Success and Failure scenario.
            Refer to {example}.
             """
        ),
        ("human", "{input}"),
    ]
)

example = """Front of the Card :
As a registered user I want to log in so that I can access subscriber-only content
Back of the Card :
1. Success - valid user logged in and referred to home page
(a) 'Remember me' ticked-store cookie/ automatic login next
time.
(b) 'Remember me' not ticked - force login next time.
2.   Failure-display message
(a) "Email address in wrong format"
(b) "Unrecognized user name, please try again"
(c) "Incorrect password, please try again"
(d) "Service unavailable, please try again"
(e) Account has expired - refer to account renewal sales page.
"""

chain = prompt | llm

validated_user_stories = chain.invoke(
    {
     "stakeholders" : stakeholders,
     "example" : example,
     "input" : problem_statement,
    }
)

print(validated_user_stories.content)
```

```
- **Failure:** System slows down, crashes, or fails under load.

#### 10. Touch-Screen Interface
**Front:**
As a passenger, I want an intuitive touch-screen interface so that I can easily na

**Back:**
- **Success:** Interface is user-friendly, responsive, and easy to navigate.
- **Failure:** Interface is unresponsive, confusing, or slow.

#### 11. Reading Existing Passes/Time Cards
**Front:**
As a passenger, I want the machine to read my existing pass or time card so that I

**Back:**
- **Success:** Pass/card is read, updated, and transaction is completed.
- **Failure:** System fails to read, update, or process the pass/card.

#### 12. Providing Ticket Information
**Front:**
As a passenger, I want the ticket to display all necessary information so that I k

**Back:**
- **Success:** Ticket includes departure time, date, validity, etc.
- **Failure:** Ticket lacks necessary information or is unclear.

#### 13. Applying Discounts for Frequent Users
**Front:**
As a frequent passenger, I want the machine to apply discounts so that I save mone

**Back:**
- **Success:** System identifies frequent user, applies discount, and displays con
- **Failure:** Discount not applied, incorrect amount charged, or no confirmation.

#### 14. Displaying Error Messages
**Front:**
As a passenger, I want clear error messages so that I understand any issues during

**Back:**
- **Success:** Clear, helpful error messages are displayed.
- **Failure:** Messages are unclear, missing, or incorrect.

Each story is designed to be independent, testable, and follows the INVEST framewo
```

Prioritizing User Stories using MoSCoW

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses the MoSCoW technique (that is c
            Display the user stories along with their MoSCoW counterparts, specify how, a
             """
        ),
        ("human", "{input}"),
    ]
)



chain = prompt | llm

prioritized_user_stories = chain.invoke(
    {
     "validated_user_stories" : validated_user_stories,
     "input" : problem_statement,
    }
)

print(prioritized_user_stories.content)
```

```
- **Should have:**
  1. Payment via Credit/Debit Card
  2. Payment via Smartcard
  3. Reading Existing Passes/Time Cards

- **Could have:**
  1. Applying Discounts for Frequent Users

- **Won't have:**
  - No features are identified as out of scope at this stage.

This classification ensures that the most critical features are developed first, e
```

Identifying EPICs

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """Identify three different EPICs (or collection of user stories)
            from {validated_user_stories}, where conflicts between the
            requirements occur. These EPICs will have user stories that have
            conflicting interests. Describe these conflicts. If the conflicts
            can be resolved, state solutions.
             """
        ),
        ("human", "{input}"),
    ]
)



chain = prompt | llm

epics = chain.invoke(
    {
     "validated_user_stories" : validated_user_stories,
     "input" : problem_statement,
    }
)

print(epics.content)
```

```
<think>
Okay, so I need to figure out the conflicts between the user stories for the Ticke

First, I'll look at the user stories listed. There are 14 user stories covering ev

Let me start by listing the user stories again to get a clear picture:

1. User Authentication and Ticket Selection
2. Payment via Cash
3. Payment via Credit/Debit Card
4. Payment via Smartcard
```

```
5. Exception Handling for Incomplete Transactions
6. Handling Insufficient Funds
7. Dispensing Tickets and Sending Messages
8. Recognizing Currency
9. Efficient Operation Under Heavy Load
10. Touch-Screen Interface
11. Reading Existing Passes/Time Cards
12. Providing Ticket Information
13. Applying Discounts for Frequent Users
14. Displaying Error Messages

Now, I need to think about how these stories might conflict. Conflicts can arise w

Let me go through each story and see where overlaps or conflicts might occur.

1. **User Authentication and Ticket Selection** - This is about selecting the type

2. **Payment via Cash** - This involves handling cash, giving change, etc. Conflic

3. **Payment via Credit/Debit Card** - Similar to cash, but with card payments. Co

4. **Payment via Smartcard** - This is about using a smartcard for payment. Confli

5. **Exception Handling for Incomplete Transactions** - This is about handling cas

6. **Handling Insufficient Funds** - This is about informing users when they don't

7. **Dispensing Tickets and Sending Messages** - This involves dispensing tickets

8. **Recognizing Currency** - This is about recognizing genuine and fake currency.

9. **Efficient Operation Under Heavy Load** - This is about the system handling mu

10. **Touch-Screen Interface** - This is about the user interface. Conflicts might

11. **Reading Existing Passes/Time Cards** - This involves reading existing passes

12. **Providing Ticket Information** - This is about the information printed on ti

13. **Applying Discounts for Frequent Users** - This is about applying discounts.

14. **Displaying Error Messages** - This is about showing clear error messages. Co

Now, I'll look for specific areas where these stories might conflict.

**Conflict 1: Payment Methods and System Performance (Stories 2, 3, 4, 9)**
```

```
pip install -qU langchain-google-genai
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.7/41.7 kB 1.7 MB/s eta 0:00:00
```

```
import getpass
import os

os.environ["GOOGLE_API_KEY"]="AIzaSyAAZBYzj1wYzNH8AhipxLEj5shCIjUKfG4"
```

```
from langchain_google_genai import ChatGoogleGenerativeAI

llm = ChatGoogleGenerativeAI(
    model="gemini-1.5-pro",
    temperature=0,
    max_tokens=None,
    timeout=None,
    max_retries=2,

)
```

```
messages = [
    (
        "system",
        "You are a helpful assistant that translates English to French. Translate the user sentence.",
    ),
    ("human", "I love programming."),
]
ai_msg = llm.invoke(messages)
ai_msg
```

```
AIMessage(content="J'adore programmer.", additional_kwargs={}, response_metadata={'prompt_feedback': {'block_reason': 0,
'safety_ratings': []}, 'finish_reason': 'STOP', 'safety_ratings': []}, id='run-1df157df-87c0-4e45-a7d4-753a886da79f-0',
usage_metadata={'input_tokens': 20, 'output_tokens': 6, 'total_tokens': 26, 'input_token_details': {'cache_read': 0}})
```

Identifying the stakeholders and end-users from the problem statement

```
problem_statement = f"""NewYork Metro station wants to establish a TicketDistributor machine that issues tickets for passengers
travelling in metro rails. Travelers have options of selecting a ticket for a single trip, round trips or for
multiple trips. They can also issue a metro pass for regular passengers or a time card for a day, a week or
a month according to their requirements. The discounts on tickets will be provided to frequent travelling
passengers. The machine is also supposed to read the metro pass and time cards issued by the metro
counters or machine. The ticket rates differ based on whether the traveler is a child or an adult. The
machine is also required to recognize original as well as fake currency notes. The typical transaction consists of a user using the disp
payment method of either cash, credit/debit card or smartcard. The ticket or tickets are printed and
dispensed to the user. Also the messaging facilities after every transaction are required on the registered
number. The system can also be operated comfortably by a touch-screen. A large number of heavy
components are to be used. We do not want our system to slow down, and also usability of the machine.
The TicketDistributor must be able to handle several exceptions, such as aborting the transaction for
incomplete transactions, insufficient amount given by the travelers to the machine, money return in case
of aborted transaction, change return after successful transaction, showing insufficient balance in the
card, updated information printed on the tickets e.g. departure time, date, time, price, valid from, valid
till, validity duration, ticket issued from and destination station. In case of exceptions, an error message is
to be displayed. We do not want user feedback after every development stage but after every two stages
to save time. The machine is required to work in a heavy load environment such that at the morning and
evening time in weekdays, and in weekends performance and efficiency would not get affected."""

co_analyst_context = """ You are a requirements analyst powered by AI along with a human analyst and you need to identify the stakeholde

prompt = """Assume you are a requirement analyst responsible for developing requirements for a MetroTicketDistributor
system.
{co_analyst_context}
Using the problem statement below, generate the key stakeholders and end-users.

Problem Statement:
{problem_statement}
 """

stakeholders = llm.invoke(prompt)
print(stakeholders.content)
```

```
Problem Statement:
The current process of purchasing metro tickets is cumbersome and time-consuming.  Passengers often face long queues at ticket coun

Key Stakeholders and End-Users:

**Stakeholders:**

* **Metro Management:**  They are responsible for the overall operation and profitability of the metro system.  They are interested
* **Finance Department:**  Concerned with revenue collection, accounting, and financial reporting related to ticket sales.
* **IT Department:** Responsible for the development, deployment, and maintenance of the MetroTicketDistributor system.
* **Marketing Department:** Interested in using the system to implement promotions, discounts, and loyalty programs to attract and r
* **Operations Department:**  Responsible for the smooth running of the metro system, including station management and passenger flo
* **Ticket Counter Staff:**  Their current roles may be impacted by the new system.  Training and potential reassignment need to be
* **Maintenance Contractors:** Responsible for the hardware maintenance and upkeep of the ticket distributor machines.
* **System Integrators:**  Responsible for integrating the new system with existing metro systems (e.g., fare gates, backend databas


**End-Users:**

* **Daily Commuters:**  They need a quick and easy way to purchase tickets regularly.
* **Occasional Riders:**  They need a user-friendly system that is easy to understand, even for infrequent use.
* **Tourists:**  They may require multi-lingual support and options for purchasing different types of passes.
* **Passengers with Disabilities:**  The system must be accessible to users with various disabilities, including visual, auditory, a
* **Senior Citizens:**  May require simplified user interfaces and potentially discounted fares.
* **Students:**  May also require discounted fares and specific ticket types.


This list covers the primary stakeholders and end-users.  Further analysis and consultation may reveal additional stakeholders or sp
```

Eliciting requirements from the stakeholders using different elicitation techniques

```python
from langchain_core.prompts import ChatPromptTemplate

prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses elicitation techniques to get requirements from {stakeholders} and list those
            Do it in the following format :
            Elicitation technique name -->
            Why :
            How :
            """
        ),
        ("human", "{input}"),
    ]
)

chain = prompt | llm

elicitation_tech = chain.invoke(
    {
     "stakeholders" : stakeholders,
     "input" : problem_statement,
    }
)

print(elicitation_tech.content)
```

```
Given the complexity and the variety of stakeholders involved in the Metro Ticket Distributor project, several elicitation technique

**1. Interviews:**

* **Why:** Interviews are crucial for understanding the specific needs and pain points of different stakeholder groups.  They allow
* **How:** Structured interviews with prepared questions can be conducted with representatives from each stakeholder group.  Open-en

**2. Workshops:**

* **Why:** Workshops bring key stakeholders together to collaboratively discuss requirements and reach a consensus. This is particul
* **How:**  A facilitated workshop with representatives from Metro Management, Finance, IT, Marketing, and Operations can be held to

**3. Document Analysis:**

* **Why:** Analyzing existing documentation, such as reports on current ticket sales, customer service complaints, and maintenance l
* **How:** Reviewing existing data on ticket sales can inform decisions about ticket types and pricing strategies.  Analyzing custom

**4. Use Cases:**

* **Why:** Use cases provide a detailed description of how different users will interact with the system.  This helps to clarify sys
* **How:** Develop use cases for different scenarios, such as purchasing a single-trip ticket, using a smartcard, or handling a fail

**5. Prototyping:**

* **Why:** Prototypes allow users to interact with a simulated version of the system early in the development process. This helps to
* **How:** Develop low-fidelity prototypes (e.g., paper prototypes or wireframes) to test key user interface elements and workflows

**6. Observations:**

* **Why:** Observing users interacting with the current ticket purchasing system can provide valuable insights into their behavior a
* **How:** Conduct observational studies at different times of day and on different days of the week to understand user behavior dur
```

Generate User Stories

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses elicitation techniques {elicitation_tech} to get all possible functionalities
            The front of the card will have the following format :
            As a [user role], I want to [goal] so that I can [reason].
            The back of the card will contain Success and Failure scenario.
            Use {example} for format.
             """
        ),
        ("human", "{input}"),
    ]
)

example = """Front of the Card :
As a registered user I want to log in so that I can access subscriber-only content
Back of the Card :
1. Success - valid user logged in and referred to home page
(a) 'Remember me' ticked-store cookie/ automatic login next
time.
(b) 'Remember me' not ticked - force login next time.
2.   Failure-display message
(a) "Email address in wrong format"
(b) "Unrecognized user name, please try again"
(c) "Incorrect password, please try again"
(d) "Service unavailable, please try again"
(e) Account has expired - refer to account renewal sales page.
"""

chain = prompt | llm

generated_user_stories = chain.invoke(
    {
     "elicitation_tech" : elicitation_tech,
     "stakeholders" : stakeholders,
     "example" : example,
     "input" : problem_statement,
    }
)

print(generated_user_stories.content)
```

```
## User Stories for Metro Ticket Distributor

Here are some user stories for the Metro Ticket Distributor, following the requested format:

**1. Purchase Single Trip Ticket (Cash)**

* **Front of the Card:** As a daily commuter, I want to purchase a single-trip ticket using cash so that I can quickly and easily

* **Back of the Card:**
    * **Success:**  Ticket printed with correct details (date, time, price, origin, destination) and dispensed. Correct change re
    * **Failure:**
        * Insufficient cash inserted. Error message displayed. Cash returned.
        * Machine out of tickets. Error message displayed. Cash returned.
        * Hardware failure (printer, dispenser). Error message displayed. Cash returned.
        * Fake currency detected. Error message displayed. Currency rejected.


**2. Purchase Round Trip Ticket (Card)**

* **Front of the Card:** As an occasional rider, I want to purchase a round-trip ticket using my credit card so that I can conver

* **Back of the Card:**
    * **Success:** Ticket printed with correct details (date, time, price, origin, destination, return trip validity) and dispens
    * **Failure:**
        * Card declined. Error message displayed.
        * Insufficient funds on card. Error message displayed.
        * Invalid card. Error message displayed.
        * Connection to payment gateway failed. Error message displayed.


**3. Purchase Multi-Trip Ticket (Smartcard)**

* **Front of the Card:** As a frequent traveler, I want to purchase a multi-trip ticket using my smartcard so that I can save tim

* **Back of the Card:**
    * **Success:** Smartcard balance updated. Discount applied (if eligible). Confirmation message sent.
    * **Failure:**
        * Insufficient balance on smartcard. Error message displayed.
        * Invalid smartcard. Error message displayed.
        * Smartcard reader error. Error message displayed.


**4. Use Existing Metro Pass**

* **Front of the Card:** As a regular commuter with a metro pass, I want to scan my pass at the ticket distributor so that I can

* **Back of the Card:**
    * **Success:** Pass validated. Entry granted.
    * **Failure:**
        * Invalid pass. Error message displayed.
        * Expired pass. Error message displayed.
        * Damaged pass. Error message displayed.


**5. Purchase Day/Week/Month Pass (Tourist)**
```

Validate User Stories (according to INVEST framework)

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses the INVEST framework
            to validate the generated user stories {generated_user_stories} i.
            e., make sure that the user stories follow the INVEST framework.
            The INVEST framework rules are :
1. All the stories that have been broken down are mostly independent  , hence
making sure that they can be parallely developed.
2. Not every minute detail has been included in the cards, encouraging
developers to seek further clarification and ensuring negotiability.
3. Stories have been written in regular business language that is equally
valuable to both stakeholders and developers.
4. It is assumed that all the user stories mentioned below can be converted
into working pieces of code within the working period of 1 to 3 weeks and hence
are estimable and small.
5. It is also written in a way where the completion of the story and its
working is quantifiable and can be tested, hence being testable.
Give the validated user stories with this format :
The front of the card will have the following format :
            As a [user role], I want to [goal] so that I can [reason].
            The back of the card will contain Success and Failure scenario.
            Refer to {example}.
             """
        ),
        ("human", "{input}"),
    ]
)

example = """Front of the Card :
As a registered user I want to log in so that I can access subscriber-only
content
Back of the Card :
1. Success - valid user logged in and referred to home page
(a) 'Remember me' ticked-store cookie/ automatic login next
time.
(b) 'Remember me' not ticked - force login next time.
2.   Failure-display message
(a) "Email address in wrong format"
(b) "Unrecognized user name, please try again"
(c) "Incorrect password, please try again"
(d) "Service unavailable, please try again"
(e) Account has expired - refer to account renewal sales page.
"""

chain = prompt | llm

validated_user_stories = chain.invoke(
    {
     "generated_user_stories" : generated_user_stories,
     "stakeholders" : stakeholders,
     "example" : example,
     "input" : problem_statement,
    }
)

print(validated_user_stories.content)
```

```
## Validated User Stories for Metro Ticket Distributor

**1. Purchase Single Trip Ticket (Cash)**
```

```
* **Front of the Card:** As a daily commuter, I want to purchase a single-trip ticket using cash so that I can quickly and easily

* **Back of the Card:**
    * **Success:** Ticket printed with correct details (date, time, price, origin, destination) and dispensed. Correct change ret
    * **Failure:**
        * Insufficient cash inserted. Error message displayed, and inserted cash returned.
        * Machine out of tickets. Error message displayed, and inserted cash returned.
        * Hardware failure (printer, dispenser). Error message displayed, and inserted cash returned.  Technician alerted.
        * Fake currency detected. Error message displayed. Currency rejected.


**2. Purchase Round Trip Ticket (Card)**

* **Front of the Card:** As an occasional rider, I want to purchase a round-trip ticket using my credit/debit card so that I can

* **Back of the Card:**
    * **Success:** Ticket printed with correct details (date, time, price, origin, destination, return trip validity) and dispens
    * **Failure:**
        * Card declined. Error message displayed.
        * Insufficient funds on card. Error message displayed.
        * Invalid card. Error message displayed.
        * Connection to payment gateway failed. Error message displayed.  Technician alerted.


**3. Purchase Multi-Trip Ticket (Smartcard)**

* **Front of the Card:** As a frequent traveler, I want to purchase a multi-trip ticket using my smartcard so that I can save tim

* **Back of the Card:**
    * **Success:** Smartcard balance updated. Discount applied (if eligible). Confirmation message sent. Number of trips added to
    * **Failure:**
        * Insufficient balance on smartcard. Error message displayed.
        * Invalid smartcard. Error message displayed.
        * Smartcard reader error. Error message displayed. Technician alerted.


**4. Use Existing Metro Pass/Time Card**

* **Front of the Card:** As a regular commuter with a metro pass/time card, I want to scan my pass/card at the ticket distributor

* **Back of the Card:**
    * **Success:** Pass/Card validated. Entry granted.  Remaining trips/validity displayed.
    * **Failure:**
        * Invalid pass/card. Error message displayed.
        * Expired pass/card. Error message displayed.
        * Damaged pass/card. Error message displayed.


**5. Purchase Day/Week/Month Pass (Tourist)**

* **Front of the Card:** As a tourist, I want to purchase a multi-day pass using cash or card so that I can easily travel around

* **Back of the Card:**
```

Prioritizing User Stories using MoSCoW

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses the MoSCoW technique to prioritize the validated user stories {validated_user_
            Display the user stories along with their MoSCoW counterparts, specify how, and provide details.
             """
        ),
        ("human", "{input}"),
    ]
)



chain = prompt | llm

prioritized_user_stories = chain.invoke(
    {
     "validated_user_stories" : validated_user_stories,
     "input" : problem_statement,
    }
)

print(prioritized_user_stories.content)
```

```
## MoSCoW Prioritization of Metro Ticket Distributor User Stories

Here's a breakdown of the user stories with MoSCoW prioritization, incorporating non-functional requirements and considerations f
```

```
**Legend:**

* **MUST:**  Essential for launch.
* **SHOULD:**  Important but not critical for launch.
* **COULD:**  Desirable but can be postponed.
* **WON'T:**  Out of scope for this release.


**1. Purchase Single Trip Ticket (Cash)**

* **Priority:** MUST
* **Front of the Card:** As a daily commuter, I want to purchase a single-trip ticket using cash so that I can quickly and easily
* **Back of the Card:**
    * **Success:** Ticket printed with correct details (date, time, price, origin, destination) and dispensed. Correct change ret
    * **Failure:** Insufficient cash inserted. Error message displayed, and inserted cash returned. Machine out of tickets. Error
* **Non-Functional Considerations:**  Performance under heavy load (MUST),  Accurate fare calculation based on age (adult/child)


**2. Purchase Round Trip Ticket (Card)**

* **Priority:** MUST
* **Front of the Card:** As an occasional rider, I want to purchase a round-trip ticket using my credit/debit card so that I can
* **Back of the Card:**  (Same as Single Trip, but with return trip validity)
* **Non-Functional Considerations:** Secure card processing (MUST), Handling various card types (MUST),  Connection to payment ga


**3. Purchase Multi-Trip Ticket (Smartcard)**

* **Priority:** MUST
* **Front of the Card:** As a frequent traveler, I want to purchase a multi-trip ticket using my smartcard so that I can save tim
* **Back of the Card:** (Similar to Single Trip, but with balance updates and discounts)
* **Non-Functional Considerations:**  Discount application (MUST), Smartcard reader reliability (MUST), Handling different smartc


**4. Use Existing Metro Pass/Time Card**

* **Priority:** MUST
* **Front of the Card:** As a regular commuter with a metro pass/time card, I want to scan my pass/card at the ticket distributor
* **Back of the Card:** Pass/Card validated. Entry granted. Remaining trips/validity displayed.
* **Non-Functional Considerations:**  Fast and reliable scanning (MUST), Handling various pass/card formats (MUST).


**5. Purchase Day/Week/Month Pass (Tourist)**

* **Priority:** SHOULD
* **Front of the Card:** As a tourist, I want to purchase a multi-day pass using cash or card so that I can easily travel around
* **Back of the Card:** (Similar to Single/Round Trip, with validity dates and optional start/end stations)
* **Non-Functional Considerations:**  Multiple language support (SHOULD).


**6. System Administrator - Generate Sales Reports**
```

Identifying EPICs

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """Identify three different EPICs (or collection of user stories) from {prioritized_user_stories}, where conflicts between t
             """
        ),
        ("human", "{input}"),
    ]
)



chain = prompt | llm

epics = chain.invoke(
    {
     "prioritized_user_stories" : prioritized_user_stories,
     "input" : problem_statement,
    }
)

print(epics.content)
```

```
WARNING:langchain_google_genai.chat_models:Retrying langchain_google_genai.chat_models._chat_with_retry.<locals>._chat_with_retry in
Here are three EPICs for the Metro Ticket Distributor, derived from the provided description, with a focus on resolving conflicts an

**EPIC 1: Core Ticket Purchase Functionality**
```

```
* **User Story 1:** As a daily commuter, I want to purchase a single-trip ticket using cash so I can quickly and easily travel.
* **User Story 2:** As an occasional rider, I want to purchase a round-trip ticket using my credit/debit card so I can conveniently
* **User Story 3:** As a frequent traveler, I want to purchase a multi-trip ticket using my smartcard so I can save time and benefi
* **User Story 4:** As a system, I need to handle incomplete/aborted transactions, ensuring appropriate cash/card refunds and error
* **User Story 5:** As a system, I need to validate inserted cash for authenticity, rejecting counterfeit currency and displaying ar
* **User Story 6:** As a system, I need to accurately calculate fares based on passenger age (adult/child) and trip type.

**Conflict Resolution and Considerations:**

* **Performance under heavy load:** This is a cross-cutting concern addressed by rigorous load testing and performance optimization
* **Hardware reliability:**  Component selection and modular design are crucial. Redundancies might be considered for critical compo
* **Incomplete/Aborted Transactions:**  Transaction rollback mechanisms are essential.  Logging and monitoring of these events are r
* **Fake Currency Detection:** Integration with reliable currency validation hardware/software is a must.

**EPIC 2:  Pass/Card Management and System Administration**

* **User Story 1:** As a regular commuter with a metro pass/time card, I want to scan my pass/card at the ticket distributor so I ca
* **User Story 2:** As a tourist, I want to purchase a day/week/month pass using cash or card so I can easily travel around the city
* **User Story 3:** As a Metro Management representative, I want to generate reports on ticket sales to analyze revenue and passenge
* **User Story 4:** As a maintenance technician, I want to refill the ticket stock and cash reserves in the machine.

**Conflict Resolution and Considerations:**

* **Various Pass/Card Formats:** The system must be adaptable to handle different formats through modular card reader systems or so
* **Report Generation Performance:**  Database optimization and efficient reporting tools are necessary.  Report generation should r
* **Security:** Secure access control for technicians during refills and data security for sales reports are paramount.  Role-based

**EPIC 3: User Interface and System Qualities**

* **User Story 1:** As a user, I want to interact with a user-friendly touchscreen interface for easy navigation and ticket selectic
* **User Story 2:** As a user with disabilities, I need the system to be accessible.
* **User Story 3:** As a system, I need to provide messaging facilities (e.g., SMS) to registered users after successful transactior
* **User Story 4:** As a system, I need to be highly reliable and maintainable.
* **User Story 5:** As a system, I need to perform efficiently under heavy load conditions, particularly during peak hours and weeke

**Conflict Resolution and Considerations:**

* **Usability Testing:**  Thorough usability testing with diverse user groups, including those with disabilities, is essential.
* **Touchscreen Responsiveness:**  High-quality hardware and optimized software are crucial for a responsive user experience.
* **Accessibility:**  Adherence to accessibility guidelines (e.g., WCAG) is a must.  Features like text-to-speech and adjustable for
* **Maintainability:** Modular design and clear documentation are key for easy maintenance and troubleshooting.
* **"User feedback after every two stages":** This requirement is not practical.  Feedback should be gathered at key milestones and


By organizing the user stories into these EPICs, we can address the interconnectedness of features and ensure that non-functional re
```

```
import getpass
import os

os.environ["GROQ_API_KEY"] =
"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"
```

```
%pip install -qU langchain-groq
```

```
from langchain_groq import ChatGroq

llm = ChatGroq(
    model="llama-3.2-3b-preview",
    temperature=0,
    max_tokens=None,
    timeout=None,
    max_retries=2,
    # other params...
)
```

```
messages = [
    (
        "system",
        "You are a helpful assistant that translates English to French.
        Translate the user sentence.",
    ),
    ("human", "I love programming."),
]
ai_msg = llm.invoke(messages)
ai_msg
```

```
AIMessage(content='Je aime le programmation.', additional_kwargs={}, response_metadata={'token_usage': {'completion_tokens': 7,
'prompt_tokens': 55, 'total_tokens': 62, 'completion_time': 0.004330176, 'prompt_time': 0.009206014, 'queue_time': 0.292379163,
'total_time': 0.01353619}, 'model_name': 'llama-3.2-3b-preview', 'system_fingerprint': 'fp_a926bfdce1', 'finish_reason': 'stop',
'logprobs': None}, id='run-a0d1f835-9430-4d33-b25b-cf29edc2e1a1-0', usage_metadata={'input_tokens': 55, 'output_tokens': 7,
'total_tokens': 62})
```

Identifying the stakeholders and end-users from the problem statement

```
problem_statement = f"""NewYork Metro station wants to establish a TicketDistributor machine that issues tickets for passengers
travelling in metro rails. Travelers have options of selecting a ticket for a single trip, round trips or for
multiple trips. They can also issue a metro pass for regular passengers or a time card for a day, a week or
a month according to their requirements. The discounts on tickets will be provided to frequent travelling
passengers. The machine is also supposed to read the metro pass and time cards issued by the metro
counters or machine. The ticket rates differ based on whether the traveler is a child or an adult. The
machine is also required to recognize original as well as fake currency notes. The typical transaction consists of a user using the disp
payment method of either cash, credit/debit card or smartcard. The ticket or tickets are printed and
dispensed to the user. Also the messaging facilities after every transaction are required on the registered
number. The system can also be operated comfortably by a touch-screen. A large number of heavy
components are to be used. We do not want our system to slow down, and also usability of the machine.
The TicketDistributor must be able to handle several exceptions, such as aborting the transaction for
incomplete transactions, insufficient amount given by the travelers to the machine, money return in case
of aborted transaction, change return after successful transaction, showing insufficient balance in the
card, updated information printed on the tickets e.g. departure time, date, time, price, valid from, valid
till, validity duration, ticket issued from and destination station. In case of exceptions, an error message is
to be displayed. We do not want user feedback after every development stage but after every two stages
to save time. The machine is required to work in a heavy load environment such that at the morning and
evening time in weekdays, and in weekends performance and efficiency would not get affected."""

co_analyst_context = """ You are a requirements analyst powered by AI along with a human analyst and you need to identify the stakehold

prompt = """Assume you are a requirement analyst responsible for developing requirements for a MetroTicketDistributor
system.
{co_analyst_context}
Using the problem statement below, generate the key stakeholders and end-users.

Problem Statement:
{problem_statement}
 """

stakeholders = llm.invoke(prompt)
print(stakeholders.content)
```

```
{co_analyst_context}

As a requirement analyst responsible for developing requirements for a MetroTicketDistributor system, I have identified the key stak

**Key Stakeholders:**
```

```
1. **Metro Officials**: Responsible for managing the metro system, including ticketing, operations, and maintenance.
2. **Transportation Department**: Oversees the overall transportation infrastructure, including the metro system.
3. **Ticketing System Providers**: Suppliers of ticketing systems for the metro, who may be involved in the implementation and maint
4. **System Integrators**: Companies responsible for integrating the new system with existing infrastructure and systems.
5. **Regulatory Bodies**: Government agencies responsible for ensuring compliance with regulations and standards related to ticketi
6. **Customer Service Representatives**: Responsible for providing support and assistance to customers using the new system.

**End-Users:**

1. **Metro Passengers**: Individuals who purchase and use tickets for the metro system.
2. **Ticketing System Administrators**: Responsible for managing and maintaining the ticketing system, including setting up accounts
3. **Metro Staff**: Employees of the metro system, including ticket collectors, customer service representatives, and maintenance pe
4. **Transportation Workers**: Employees of transportation companies, including drivers, conductors, and dispatchers.
5. **System Administrators**: Responsible for managing and maintaining the MetroTicketDistributor system, including setting up accou

These stakeholders and end-users will be involved in the development and implementation of the MetroTicketDistributor system, and th
```

Eliciting requirements from the stakeholders using different elicitation techniques

```python
from langchain_core.prompts import ChatPromptTemplate

prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses elicitation techniques to get requirements from {stakeholders} and list those
            Do it in the following format :
            Elicitation technique name -->
            Why :
            How :
            """
        ),
        ("human", "{input}"),
    ]
)

chain = prompt | llm

elicitation_tech = chain.invoke(
    {
     "stakeholders" : stakeholders,
     "input" : problem_statement,
    }
)

print(elicitation_tech.content)
```

```
Here are the elicitation techniques used to gather requirements for the MetroTicketDistributor system:

**1.** **Interviews -->**
Why : Interviews allow for in-depth discussions with stakeholders and end-users to gather detailed information about their needs and
How : Conduct one-on-one interviews with key stakeholders, including Metro Officials, Transportation Department, Ticketing System Pr

**2.** **Surveys -->**
Why : Surveys help to gather information from a larger group of people, including Metro Passengers, Ticketing System Administrators,
How : Create a survey questionnaire that covers the requirements and needs of the system, such as ticket types, payment methods, mes

**3.** **Use Cases -->**
Why : Use cases help to identify the functional requirements of the system, including the interactions between the system and the us
How : Create a list of use cases that describe the different scenarios in which the system will be used, such as selecting a ticket,

**4.** **User Stories -->**
Why : User stories help to identify the non-functional requirements of the system, including usability, performance, and reliability
How : Create user stories that describe the different scenarios in which the system will be used, such as "As a Metro Passenger, I w

**5.** **Requirements Gathering Workshops -->**
Why : Requirements gathering workshops help to identify the requirements of the system in a collaborative and interactive environmen
How : Organize a workshop with stakeholders and end-users to discuss the requirements of the system. Use techniques such as brainsto

**6.** **System Analysis -->**
Why : System analysis helps to identify the technical requirements of the system, including hardware, software, and networking requi
How : Conduct a system analysis to identify the technical requirements of the system, including the hardware, software, and networki

**7.** **Performance Testing -->**
Why : Performance testing helps to identify the performance requirements of the system, including the system's ability to handle a h
How : Conduct performance testing to identify the performance requirements of the system, including the system's ability to handle a

**8.** **Usability Testing -->**
Why : Usability testing helps to identify the usability requirements of the system, including the system's ability to be used comfor
How : Conduct usability testing to identify the usability requirements of the system, including the system's ability to be used comf

These elicitation techniques are helpful in gathering requirements because they allow for a comprehensive and interactive approach t
```

Generate User Stories

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses elicitation techniques {elicitation_tech} to get all possible functionalities
            The front of the card will have the following format :
            As a [user role], I want to [goal] so that I can [reason].
            The back of the card will contain Success and Failure scenario.
            Use {example} for format.
             """
        ),
        ("human", "{input}"),
    ]
)

example = """Front of the Card :
As a registered user I want to log in so that I can access subscriber-only content
Back of the Card :
1. Success - valid user logged in and referred to home page
(a) 'Remember me' ticked-store cookie/ automatic login next
time.
(b) 'Remember me' not ticked - force login next time.
2.   Failure-display message
(a) "Email address in wrong format"
(b) "Unrecognized user name, please try again"
(c) "Incorrect password, please try again"
(d) "Service unavailable, please try again"
(e) Account has expired - refer to account renewal sales page.
"""

chain = prompt | llm

generated_user_stories = chain.invoke(
    {
     "elicitation_tech" : elicitation_tech,
     "stakeholders" : stakeholders,
     "example" : example,
     "input" : problem_statement,
    }
)

print(generated_user_stories.content)
```

```
Here are the user stories for the TicketDistributor machine:

**Front of the Card**

1. As a registered user, I want to select a ticket type (single trip, round trip, multiple trips) so that I can choose my travel
        * Back of the Card:
                + Success:
                        - (a) Ticket type selected successfully
                        - (b) Ticket type validation (e.g. single trip, round trip, multiple trips)
                + Failure:
                        - (a) Invalid ticket type selected
                        - (b) Ticket type not available
                        - (c) Error message displayed
2. As a registered user, I want to select the quantity of tickets so that I can choose the number of tickets I need.
        * Back of the Card:
                + Success:
                        - (a) Quantity selected successfully
                        - (b) Quantity validation (e.g. positive integer)
                + Failure:
                        - (a) Invalid quantity selected
                        - (b) Quantity not available
                        - (c) Error message displayed
3. As a registered user, I want to select the payment method (cash, credit/debit card, smartcard) so that I can pay for my ticket
        * Back of the Card:
                + Success:
                        - (a) Payment method selected successfully
                        - (b) Payment method validation (e.g. valid payment method)
                + Failure:
                        - (a) Invalid payment method selected
                        - (b) Payment method not available
                        - (c) Error message displayed
4. As a registered user, I want to receive the printed tickets so that I can have a physical copy of my tickets.
        * Back of the Card:
                + Success:
                        - (a) Tickets printed successfully
                        - (b) Tickets dispensed successfully
                + Failure:
                        - (a) Tickets not printed
                        - (b) Tickets not dispensed
```

```
                    - (c) Error message displayed
    5. As a registered user, I want to receive a message after every transaction so that I can confirm my transaction.
            * Back of the Card:
                    + Success:
                            - (a) Message sent successfully
                            - (b) Message received successfully
                    + Failure:
                            - (a) Message not sent
                            - (b) Message not received
                            - (c) Error message displayed

    **User Stories for Exceptions**

    1. As a registered user, I want the system to abort the transaction if the amount is insufficient so that I can avoid losing mone
            * Back of the Card:
                    + Success:
                            - (a) Transaction aborted successfully
                            - (b) Error message displayed
```

Validate User Stories (according to INVEST framework)

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses the INVEST framework to validate the {generated_user_stories} i.e., make sure
            The INVEST framework rules are :
1. All the stories that have been broken down are mostly independent    , hence making sure that they can be parallely developed.
2. Not every minute detail has been included in the cards, encouraging developers to seek further clarification and ensuring negotiabil
3. Stories have been written in regular business language that is equally valuable to both stakeholders and developers.
4. It is assumed that all the user stories mentioned below can be converted into working pieces of code within the working period of 1 
5. It is also written in a way where the completion of the story and its working is quantifiable and can be tested, hence being testabl
Give the validated user stories with this format :
The front of the card will have the following format :
            As a [user role], I want to [goal] so that I can [reason].
            The back of the card will contain Success and Failure scenario.
            Refer to {example}.
             """
        ),
        ("human", "{input}"),
    ]
)

example = """Front of the Card :
As a registered user I want to log in so that I can access subscriber-only content
Back of the Card :
1. Success - valid user logged in and referred to home page
(a) 'Remember me' ticked-store cookie/ automatic login next
time.
(b) 'Remember me' not ticked - force login next time.
2.   Failure-display message
(a) "Email address in wrong format"
(b) "Unrecognized user name, please try again"
(c) "Incorrect password, please try again"
(d) "Service unavailable, please try again"
(e) Account has expired - refer to account renewal sales page.
"""

chain = prompt | llm

validated_user_stories = chain.invoke(
    {
        "generated_user_stories" : generated_user_stories,

      "stakeholders" : stakeholders,
      "example" : example,
      "input" : problem_statement,
    }
)

print(validated_user_stories.content)
```

```
    Here are the validated user stories for the TicketDistributor machine using the INVEST framework:

    **User Stories for Independent Development**

    1. As a registered user, I want to select a ticket type (single trip, round trip, multiple trips) so that I can choose my travel
            * Back of the Card:
                    + Success:
                            - (a) Ticket type selected successfully
                            - (b) Ticket type validation (e.g. single trip, round trip, multiple trips)
                    + Failure:
                            - (a) Invalid ticket type selected
```

```
                                - (b) Ticket type not available
                                - (c) Error message displayed
        2. As a registered user, I want to select the quantity of tickets so that I can choose the number of tickets I need.
                * Back of the Card:
                        + Success:
                                - (a) Quantity selected successfully
                                - (b) Quantity validation (e.g. positive integer)
                        + Failure:
                                - (a) Invalid quantity selected
                                - (b) Quantity not available
                                - (c) Error message displayed
        3. As a registered user, I want to select the payment method (cash, credit/debit card, smartcard) so that I can pay for my ticket
                * Back of the Card:
                        + Success:
                                - (a) Payment method selected successfully
                                - (b) Payment method validation (e.g. valid payment method)
                        + Failure:
                                - (a) Invalid payment method selected
                                - (b) Payment method not available
                                - (c) Error message displayed
        4. As a registered user, I want to receive the printed tickets so that I can have a physical copy of my tickets.
                * Back of the Card:
                        + Success:
                                - (a) Tickets printed successfully
                                - (b) Tickets dispensed successfully
                        + Failure:
                                - (a) Tickets not printed
                                - (b) Tickets not dispensed
                                - (c) Error message displayed
        5. As a registered user, I want to receive a message after every transaction so that I can confirm my transaction.
                * Back of the Card:
                        + Success:
                                - (a) Message sent successfully
                                - (b) Message received successfully
                        + Failure:
                                - (a) Message not sent
                                - (b) Message not received
                                - (c) Error message displayed

        **User Stories for Exceptions**

        1. As a registered user, I want the system to abort the transaction if the amount is insufficient so that I can avoid losing mone
                * Back of the Card:
                        + Success:
                                - (a) Transaction aborted successfully
                                - (b) Error message displayed
```

Prioritizing User Stories using MoSCoW

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses the MoSCoW technique (that is classifying user story into must have, should ha
            Display the user stories along with their MoSCoW counterparts, specify how, and provide details.
             """
        ),
        ("human", "{input}"),
    ]
)



chain = prompt | llm

prioritized_user_stories = chain.invoke(
    {
     "validated_user_stories" : validated_user_stories,
     "input" : problem_statement,
    }
)

print(prioritized_user_stories.content)
```

```
        Based on the provided requirements, I will categorize the user stories into MoSCoW counterparts.

        **Must-Have (M)**

        1. As a registered user, I want to select a ticket type (single trip, round trip, multiple trips) so that I can choose my travel
                * Back of the Card:
                        + Success: (a) Ticket type selected successfully
                        + Failure: (a) Invalid ticket type selected
                        + (b) Ticket type not available
                        + (c) Error message displayed
        2. As a registered user, I want to select the quantity of tickets so that I can choose the number of tickets I need.
```

```
            * Back of the Card:
                    + Success: (a) Quantity selected successfully
                    + Failure: (a) Invalid quantity selected
                    + (b) Quantity not available
                    + (c) Error message displayed
    3. As a registered user, I want to select the payment method (cash, credit/debit card, smartcard) so that I can pay for my ticket
            * Back of the Card:
                    + Success: (a) Payment method selected successfully
                    + Failure: (a) Invalid payment method selected
                    + (b) Payment method not available
                    + (c) Error message displayed
    4. As a registered user, I want to receive the printed tickets so that I can have a physical copy of my tickets.
            * Back of the Card:
                    + Success: (a) Tickets printed successfully
                    + Failure: (a) Tickets not printed
                    + (b) Tickets not dispensed
                    + (c) Error message displayed
    5. As a registered user, I want to receive a message after every transaction so that I can confirm my transaction.
            * Back of the Card:
                    + Success: (a) Message sent successfully
                    + Failure: (a) Message not sent
                    + (b) Message not received
                    + (c) Error message displayed

    **Should-Have (SH)**

    1. As a registered user, I want the system to abort the transaction if the amount is insufficient so that I can avoid losing mone
            * Back of the Card:
                    + Success: (a) Transaction aborted successfully
                    + Failure: (a) Insufficient amount
                    + (b) Error message displayed
    2. As a registered user, I want the system to return money in case of aborted transaction so that I can get my money back.
            * Back of the Card:
                    + Success: (a) Money returned successfully
                    + Failure: (a) Money not returned
                    + (b) Error message displayed
    3. As a registered user, I want the system to display an error message if the card has insufficient balance so that I can avoid c
            * Back of the Card:
                    + Success: (a) Error message displayed
                    + Failure: (a) Insufficient balance
                    + (b) Error message displayed
    4. As a registered user, I want the system to update the information on the ticket (departure time, date, time, price, valid from
            * Back of the Card:
                    + Success: (a) Information updated successfully
                    + Failure: (a) Information not updated
```

Identifying EPICs

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """Identify three different EPICs (or collection of user stories)
            from {validated_user_stories}, where conflicts between the
            requirements occur. These EPICs will have user stories that have
            conflicting interests. Describe these conflicts. If the conflicts
            can be resolved, state solutions.
             """
        ),
        ("human", "{input}"),
    ]
)
```

```
chain = prompt | llm

epics = chain.invoke(
    {
     "validated_user_stories" : validated_user_stories,
     "input" : problem_statement,
    }
)

print(epics.content)
```

```
Based on the provided requirements, I have identified three EPICs (or collection of user stories) that seem to have conflicting inte

**EPIC 1: User Experience and Usability**

* User Stories:
        + As a registered user, I want the system to be operated comfortably by a touch-screen so that I can easily interact with tl
        + As a registered user, I want the system to handle heavy loads without slowing down so that I can use the system efficient]
        + As a registered user, I want the system to be operated comfortably by a touch-screen so that I can easily interact with tl
* Conflicts:
        + The first two user stories require the system to handle heavy loads without slowing down, which may conflict with the requ
        + The system may need to optimize its performance to handle heavy loads while still providing a smooth user experience.

**EPIC 2: Payment and Transaction**

* User Stories:
        + As a registered user, I want to select the payment method (cash, credit/debit card, smartcard) so that I can pay for my ti
        + As a registered user, I want the system to recognize original as well as fake currency notes.
        + As a registered user, I want the system to abort the transaction if the amount is insufficient so that I can avoid losing
* Conflicts:
        + The first two user stories require the system to recognize different payment methods, which may conflict with the requirem
        + The system may need to implement additional checks to prevent fake currency notes from being accepted.

**EPIC 3: Ticketing and Information**

* User Stories:
        + As a registered user, I want to select the type and quantity of tickets so that I can choose my travel option.
        + As a registered user, I want the system to update the information on the ticket (departure time, date, time, price, valid
        + As a registered user, I want the system to display an error message if the card has insufficient balance so that I can avo
* Conflicts:
        + The first two user stories require the system to update information on the ticket, which may conflict with the requirement
        + The system may need to implement additional checks to prevent over-drafting.

To resolve these conflicts, the following solutions can be proposed:

1. **Prioritize user stories**: Prioritize the user stories based on their importance and complexity. For example, the user stories
2. **Implement a modular design**: Implement a modular design for the system, where each module is responsible for a specific functi
3. **Use a separate module for payment processing**: Use a separate module for payment processing, which can handle the payment meth
4. **Use a separate module for ticketing and information**: Use a separate module for ticketing and information, which can handle th
5. **Implement additional checks**: Implement additional checks to prevent fake currency notes from being accepted, over-drafting, a

By implementing these solutions, the conflicts between the user stories can be resolved, and the system can be designed to meet all
```

```
import getpass
import os


os.environ["MISTRAL_API_KEY"] ="v3hj7pDnJ8PqxOejAAYM2imTPIKhueWL"
```

```
%pip install -qU langchain_mistralai
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 413.2/413.2 kB 12.6 MB/s eta 0:00:00
```

```
from langchain_mistralai import ChatMistralAI

llm = ChatMistralAI(
    model="mistral-large-latest",
    temperature=0,
    max_retries=2,
    # other params...
)
```

```
messages = [
    (
        "system",
        "You are a helpful assistant that translates English to French. Translate the user sentence.",
    ),
    ("human", "I love programming."),
]
ai_msg = llm.invoke(messages)
ai_msg
```

```
AIMessage(content="I love programming. = J'aime la programmation.\n\nHere's a breakdown:\n- I love = J'aime\n- programming = la
programmation", additional_kwargs={}, response_metadata={'token_usage': {'prompt_tokens': 27, 'total_tokens': 66,
'completion_tokens': 39}, 'model': 'mistral-large-latest', 'finish_reason': 'stop'}, id='run-aaf37ce5-07b3-4657-953b-5f63cfe41e8e-
0', usage_metadata={'input_tokens': 27, 'output_tokens': 39, 'total_tokens': 66})
```

Identifying stakeholders

```
problem_statement = f"""NewYork Metro station wants to establish a
TicketDistributor machine that issues tickets for passengers
travelling in metro rails. Travelers have options of selecting a ticket for a
single trip, round trips or for
multiple trips. They can also issue a metro pass for regular passengers or a
time card for a day, a week or
a month according to their requirements. The discounts on tickets will be
provided to frequent travelling
passengers. The machine is also supposed to read the metro pass and time cards
issued by the metro
counters or machine. The ticket rates differ based on whether the traveler is a
child or an adult. The
machine is also required to recognize original as well as fake currency notes.
The typical transaction consists of a user using the display interface to
select the type and quantity of tickets and then choosing a
payment method of either cash, credit/debit card or smartcard. The ticket or
tickets are printed and
dispensed to the user. Also the messaging facilities after every transaction
are required on the registered
number. The system can also be operated comfortably by a touch-screen. A large
number of heavy
components are to be used. We do not want our system to slow down, and also
usability of the machine.
The TicketDistributor must be able to handle several exceptions, such as
aborting the transaction for
incomplete transactions, insufficient amount given by the travelers to the
machine, money return in case
of aborted transaction, change return after successful transaction, showing
insufficient balance in the
card, updated information printed on the tickets e.g. departure time, date,
time, price, valid from, valid
till, validity duration, ticket issued from and destination station. In case of
exceptions, an error message is
to be displayed. We do not want user feedback after every development stage but
after every two stages
to save time. The machine is required to work in a heavy load environment such
that at the morning and
evening time in weekdays, and in weekends performance and efficiency would not
get affected."""

co_analyst_context = """ You are a requirements analyst powered by AI along
with a human analyst and you need to identify the stakeholders and end-users"""
```

```
prompt = """Assume you are a requirement analyst responsible for developing
requirements for a MetroTicketDistributor
system.
{co_analyst_context}
Using the problem statement below, generate the key stakeholders and end-users.

Problem Statement:
{problem_statement}
 """

stakeholders = llm.invoke(prompt)
print(stakeholders.content)
```

```
To generate the key stakeholders and end-users for the MetroTicketDistributor system, let's first consider the problem statement. Si

**Problem Statement:**
The MetroTicketDistributor system aims to streamline the process of purchasing and managing metro tickets for commuters. The system

### Key Stakeholders:
1. **Commuters:**
   - **Description:** Individuals who use the metro system for daily travel.
   - **Interests:** Easy ticket purchasing, balance checking, travel history, and notifications.

2. **Metro Authority:**
   - **Description:** The governing body responsible for managing the metro system.
   - **Interests:** Efficient ticket distribution, revenue management, and system performance monitoring.

3. **System Administrators:**
   - **Description:** IT personnel responsible for maintaining the MetroTicketDistributor system.
   - **Interests:** System uptime, security, and performance.

4. **Customer Support:**
   - **Description:** Staff handling customer inquiries and issues related to ticket purchases and usage.
   - **Interests:** Effective tools for resolving customer issues and providing support.

5. **Payment Gateway Providers:**
   - **Description:** Third-party services facilitating online payments.
   - **Interests:** Secure and reliable transaction processing.

6. **Marketing Team:**
   - **Description:** Team responsible for promotions and customer engagement.
   - **Interests:** Tools for managing promotions and customer notifications.

### End-Users:
1. **Commuters:**
   - **Description:** Individuals who use the metro system for daily travel.
   - **Needs:** User-friendly interface for purchasing tickets, checking balance, viewing travel history, and receiving notificatior

2. **System Administrators:**
   - **Description:** IT personnel responsible for maintaining the MetroTicketDistributor system.
   - **Needs:** Tools for monitoring system performance, ensuring security, and managing user accounts.

3. **Customer Support Representatives:**
   - **Description:** Staff handling customer inquiries and issues related to ticket purchases and usage.
   - **Needs:** Access to customer data, tools for resolving issues, and communication channels with commuters.

4. **Marketing Team:**
   - **Description:** Team responsible for promotions and customer engagement.
   - **Needs:** Tools for creating and managing promotions, sending notifications, and analyzing customer engagement.

5. **Metro Authority Officials:**
   - **Description:** Officials overseeing the metro system operations.
   - **Needs:** Reports on ticket sales, revenue, and system performance.

By identifying these key stakeholders and end-users, we can ensure that the requirements for the MetroTicketDistributor system are
```

Give elicitation techniques

```
from langchain_core.prompts import ChatPromptTemplate

prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses elicitation
            techniques to get requirements from {stakeholders} and list those
            elicitation techniques and justify why and how the chosen
            elicitation techniques are helpful in gathering requirements.
            Do it in the following format :
            Elicitation technique name -->
            Why :
            How :
            """
        ),
        ("human", "{input}"),
    ]
)

chain = prompt | llm

elicitation_tech = chain.invoke(
    {
     "stakeholders" : stakeholders,
     "input" : problem_statement,
    }
)

print(elicitation_tech.content)
```

```
- Document the responses and analyze them to identify common themes and unique requirements.

### Elicitation Technique 2: Surveys/Questionnaires
**Why:**
Surveys and questionnaires are useful for collecting quantitative data from a large number of stakeholders. They can help identif

**How:**
- Design a survey with questions related to ticket purchasing preferences, payment methods, usability, and desired features.
- Distribute the survey to a broad audience, including regular commuters, occasional travelers, and Metro Authority staff.
- Analyze the survey results to identify common requirements and areas of improvement.

### Elicitation Technique 3: Workshops
**Why:**
Workshops facilitate collaborative discussions and brainstorming sessions among stakeholders. They help in generating creative sc

**How:**
- Organize workshops with key stakeholders, including representatives from the Metro Authority, system administrators, customer s
- Use brainstorming sessions to discuss the system's features, usability, and exception handling.
- Document the outcomes and prioritize the requirements based on the discussions.

### Elicitation Technique 4: Observation
**Why:**
Observation helps in understanding the current processes and identifying areas for improvement. It provides insights into user be

**How:**
- Observe the current ticket purchasing process at metro stations.
- Note the challenges faced by users and the inefficiencies in the current system.
- Use the observations to identify requirements for improving the user experience and system efficiency.

### Elicitation Technique 5: Document Analysis
**Why:**
Document analysis involves reviewing existing documents, such as problem statements, user manuals, and system specifications, to

**How:**
- Review the problem statement and any existing documentation related to the MetroTicketDistributor system.
- Extract relevant information and identify gaps in the current documentation.
- Use the findings to supplement the requirements gathered through other elicitation techniques.

### Elicitation Technique 6: Prototyping
**Why:**
Prototyping helps in visualizing the system and gathering feedback on the design and usability. It allows stakeholders to interac

**How:**
- Create a prototype of the MetroTicketDistributor system, including the user interface and key features.
- Demonstrate the prototype to stakeholders and gather their feedback.
- Use the feedback to refine the requirements and improve the system design.

### Elicitation Technique 7: Use Cases and Scenarios
**Why:**
Use cases and scenarios help in understanding the system's functionality from the user's perspective. They provide a clear pictur

**How:**
- Develop use cases and scenarios for different user interactions, such as purchasing tickets, checking balances, and handling ex
- Review the use cases with stakeholders to ensure they accurately represent the system's requirements.
- Use the use cases to identify additional requirements and refine the system design.

By employing these elicitation techniques, we can ensure that the requirements for the MetroTicketDistributor system are comprehe
```

Generate user stories

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses elicitation
            techniques {elicitation_tech} to get all possible functionalities
            for each user in {stakeholders} in the form of user stories (both
            front and back of card). Therefore, list all possible user stories
            with the appropriate format.
            The front of the card will have the following format :
```

```
            As a [user role], I want to [goal] so that I can [reason].
            The back of the card will contain Success and Failure scenario.
            Use {example} for format.
             """
        ),
        ("human", "{input}"),
    ]
)

example = """Front of the Card :
As a registered user I want to log in so that I can access subscriber-only
content
Back of the Card :
1. Success - valid user logged in and referred to home page
(a) 'Remember me' ticked-store cookie/ automatic login next
time.
(b) 'Remember me' not ticked - force login next time.
2.   Failure-display message
(a) "Email address in wrong format"
(b) "Unrecognized user name, please try again"
(c) "Incorrect password, please try again"
(d) "Service unavailable, please try again"
(e) Account has expired - refer to account renewal sales page.
"""

chain = prompt | llm

generated_user_stories = chain.invoke(
    {
     "elicitation_tech" : elicitation_tech,
     "stakeholders" : stakeholders,
     "example" : example,
     "input" : problem_statement,
    }
)

print(generated_user_stories.content)
```

```
     Back of the Card:
    1. **Success:**
       - Transaction processed securely and reliably.
       - **Failure:**
         - "Transaction processing failed."
         - "Security issues detected."

    By creating these user stories, we can ensure that the requirements for the MetroTicketDistributor system are comprehensive and a
```

Validate user stories

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses the INVEST framework
            to validate the generated user stories {generated_user_stories} i.
            e., make sure that the user stories follow the INVEST framework.
            The INVEST framework rules are :
1. All the stories that have been broken down are mostly independent  , hence
making sure that they can be parallely developed.
2. Not every minute detail has been included in the cards, encouraging
developers to seek further clarification and ensuring negotiability.
3. Stories have been written in regular business language that is equally
valuable to both stakeholders and developers.
4. It is assumed that all the user stories mentioned below can be converted
into working pieces of code within the working period of 1 to 3 weeks and hence
are estimable and small.
5. It is also written in a way where the completion of the story and its
working is quantifiable and can be tested, hence being testable.
Give the validated user stories with this format :
The front of the card will have the following format :
            As a [user role], I want to [goal] so that I can [reason].
            The back of the card will contain Success and Failure scenario.
            Refer to {example}.
             """
        ),
        ("human", "{input}"),
    ]
)

example = """Front of the Card :
As a registered user I want to log in so that I can access subscriber-only
content
Back of the Card :
1. Success - valid user logged in and referred to home page
(a) 'Remember me' ticked-store cookie/ automatic login next
time.
(b) 'Remember me' not ticked - force login next time.
2.   Failure-display message
(a) "Email address in wrong format"
(b) "Unrecognized user name, please try again"
(c) "Incorrect password, please try again"
(d) "Service unavailable, please try again"
(e) Account has expired - refer to account renewal sales page.
"""

chain = prompt | llm

validated_user_stories = chain.invoke(
    {
     "generated_user_stories" : generated_user_stories,
     "stakeholders" : stakeholders,
     "example" : example,
     "input" : problem_statement,
    }
)

print(validated_user_stories.content)
```

```
           Customer issue resolution failed.
         - "Support tools not functioning."
      3. **Success:**
         - Communication with commuters established.
         - **Failure:**
           - "Communication channels failed."
           - "Commuter contact issues detected."

      ### Marketing Team

      **Front of the Card:**
      1. **As a** marketing team member, **I want to** create and manage promotions **so that I can** engage customers effectively.
      2. **As a** marketing team member, **I want to** send notifications to customers **so that I can** keep them informed about promo
      3. **As a** marketing team member, **I want to** analyze customer engagement **so that I can** improve marketing strategies.

      **Back of the Card:**
      1. **Success:**
         - Promotions created and managed successfully.
         - **Failure:**
           - "Promotion creation failed."
           - "Promotion management tools not functioning."
      2. **Success:**
         - Notifications sent to customers successfully.
         - **Failure:**
           - "Notification sending failed."
           - "Customer contact issues detected."
      3. **Success:**
         - Customer engagement analyzed accurately.
         - **Failure:**
           - "Customer engagement analysis failed."
           - "Data retrieval issues detected."

      ### Payment Gateway Providers

      **Front of the Card:**
      1. **As a** payment gateway provider, **I want to** ensure secure and reliable transaction processing **so that I can** maintain

      **Back of the Card:**
      1. **Success:**
         - Transaction processed securely and reliably.
         - **Failure:**
           - "Transaction processing failed."
           - "Security issues detected."

      By creating these user stories, we can ensure that the requirements for the MetroTicketDistributor system are comprehensive and a
```

Prioritize user stories

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """You are a requirements coanalyst that uses the MoSCoW technique
            to prioritize the validated user stories {validated_user_stories}
            taking into consideration the non-functional aspects as well.
            Display the user stories along with their MoSCoW counterparts,
            specify how, and provide details.
             """
        ),
        ("human", "{input}"),
    ]
)
```

```
chain = prompt | llm

prioritized_user_stories = chain.invoke(
    {
     "validated_user_stories" : validated_user_stories,
     "input" : problem_statement,
    }
)

print(prioritized_user_stories.content)
```

```
      To prioritize the user stories using the MoSCoW technique, we need to categorize each user story into one of the following catego

      - **Must Have (M)**: Critical requirements that must be implemented for the system to function.
      - **Should Have (S)**: Important requirements that should be implemented if possible, but the system can still function without t
      - **Could Have (C)**: Nice-to-have requirements that can be implemented if time and resources allow.
      - **Won't Have (W)**: Requirements that are not necessary for the current release but might be considered for future releases.

      Here are the user stories along with their MoSCoW counterparts:

      ### Commuters
```

```
**1. As a commuter, I want to select the type of ticket (single trip, round trip, multiple trips, metro pass, time card) so that
   - **MoSCoW**: Must Have (M)
   - **How**: This is a core functionality that allows commuters to choose the type of ticket they need.
   - **Details**: This feature is essential for the basic operation of the ticket distributor machine.

**2. As a commuter, I want to choose my payment method (cash, credit/debit card, smartcard) so that I can complete my ticket purc
   - **MoSCoW**: Must Have (M)
   - **How**: This feature enables commuters to pay for their tickets using various methods.
   - **Details**: Essential for completing the transaction process.

**3. As a commuter, I want to receive a message on my registered number after every transaction so that I have a record of my pur
   - **MoSCoW**: Should Have (S)
   - **How**: This feature provides a confirmation and record of the transaction.
   - **Details**: Important for user satisfaction and record-keeping but not critical for basic functionality.

**4. As a commuter, I want the machine to recognize original and fake currency notes so that I can avoid fraudulent transactions.
   - **MoSCoW**: Must Have (M)
   - **How**: This feature ensures the security and integrity of cash transactions.
   - **Details**: Critical for preventing fraud and ensuring accurate payments.

**5. As a commuter, I want to receive change after a successful cash transaction so that I get the correct amount back.**
   - **MoSCoW**: Must Have (M)
   - **How**: This feature ensures that commuters receive the correct change.
   - **Details**: Essential for user satisfaction and accurate financial transactions.

**6. As a commuter, I want to see an error message for any exceptions during the transaction so that I know what went wrong.**
   - **MoSCoW**: Must Have (M)
   - **How**: This feature provides feedback to the user in case of errors.
   - **Details**: Critical for user experience and troubleshooting.

### Administrators

**1. As an administrator, I want to manage user accounts so that I can ensure the system is secure and users are authenticated.**
   - **MoSCoW**: Must Have (M)
   - **How**: This feature ensures the security and authentication of user accounts.
   - **Details**: Critical for system security and user management.

**2. As an administrator, I want to monitor ticket sales so that I can track revenue and usage patterns.**
   - **MoSCoW**: Should Have (S)
   - **How**: This feature provides insights into ticket sales and usage.
   - **Details**: Important for revenue tracking and usage analysis but not critical for basic functionality.

**3. As an administrator, I want to maintain the system so that it runs smoothly and efficiently.**
   - **MoSCoW**: Must Have (M)
   - **How**: This feature ensures the system's performance and reliability.
   - **Details**: Critical for the overall functioning of the system.
```

Identify EPICs

```
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            """Identify three different EPICs (or collection of user stories) from {prioritized_user_stories}, where conflicts between t
             """
        ),
        ("human", "{input}"),
    ]
)



chain = prompt | llm

epics = chain.invoke(
    {
     "prioritized_user_stories" : prioritized_user_stories,
     "input" : problem_statement,
    }
)

print(epics.content)
```

```
Based on the provided requirements and the user stories categorized using the MoSCoW technique, we can identify three different EPIC

### EPIC 1: Core Ticket Purchasing Functionality
**User Stories:**
1. As a commuter, I want to select the type of ticket (single trip, round trip, multiple trips, metro pass, time card) so that I car
2. As a commuter, I want to choose my payment method (cash, credit/debit card, smartcard) so that I can complete my ticket purchase
3. As a commuter, I want the machine to recognize original and fake currency notes so that I can avoid fraudulent transactions. (Mus
4. As a commuter, I want to receive change after a successful cash transaction so that I get the correct amount back. (Must Have)
5. As a commuter, I want to see an error message for any exceptions during the transaction so that I know what went wrong. (Must Hav
6. As a commuter, I want to receive a message on my registered number after every transaction so that I have a record of my purchase

**Conflicts and Solutions:**
```

```
- **Conflict:** The requirement to recognize original and fake currency notes might conflict with the need to handle a large number
  - **Solution:** Implement a fast and efficient currency validation system that does not significantly slow down the transaction pr

### EPIC 2: System Administration and Maintenance
**User Stories:**
1. As an administrator, I want to manage user accounts so that I can ensure the system is secure and users are authenticated. (Must
2. As an administrator, I want to monitor ticket sales so that I can track revenue and usage patterns. (Should Have)
3. As an administrator, I want to maintain the system so that it runs smoothly and efficiently. (Must Have)
4. As a system developer, I want to ensure the system is scalable and maintainable so that it can handle a large number of transacti
5. As a system developer, I want to gather user feedback after every two development stages so that I can improve the system based o

**Conflicts and Solutions:**
- **Conflict:** The need to gather user feedback after every two development stages might conflict with the desire to save time.
  - **Solution:** Streamline the feedback collection process to minimize time spent while still gathering valuable insights.

### EPIC 3: Customer Support and Exception Handling
**User Stories:**
1. As a customer support agent, I want to access user data so that I can assist commuters with their issues. (Must Have)
2. As a customer support agent, I want to have tools for resolving issues efficiently so that I can provide better support to commut
3. As a commuter, I want to see an error message for any exceptions during the transaction so that I know what went wrong. (Must Hav
4. As a commuter, I want the machine to handle exceptions such as aborting the transaction for incomplete transactions, insufficient

**Conflicts and Solutions:**
- **Conflict:** The need to handle a wide range of exceptions might conflict with the goal of maintaining system performance and eff
  - **Solution:** Design robust exception handling mechanisms that are integrated seamlessly into the system to ensure minimal impac

By organizing the user stories into these EPICs, we can prioritize and manage the development process more effectively, ensuring tha
```