Color coding

- Problem Setup:
  - Project context: scope and expectations
  - Course context: definitions and required components
- Instructions and Expected Results:
  - Explicit Requests
  - Expected Content
  - Expected Format: paragraph vs. bullet points, length, etc.
- Contextualization:
  - Personas
  - Examples
- Not considered/counted

> Prompt

Hi ChatGPT! You are a software engineer with experience in requirement engineering. You will aid in defining requirements specifications for a new project. You will need to refine, clarify, and expand upon an initial requirements specifications provided as an input.

> Prompt

For the project, we need to develop a mobile Android application with a NodeJS backend running on a cloud. The development of the application will be carried out by a group of 4 students of a third-year software engineering course, over the duration of 3 months. The application must include the following four components:

1. Use of an external authentication service, e.g., Google or Facebook authentication.
2. Use of at least one additional external service accessible via an API, e.g., Google calendar, Google maps, etc.
3. Real-time updates, e.g., multi-user chats, push notification, etc.
4. Non-trivial logic in either front-end or back-end component (i.e., something that involves an algorithm, not a simple API call or database read/update/delete).

I will provide you with my application idea in the next prompt and then ask a few questions about it.

> Prompt

My application name is <NAME> and its high-level idea is <IDEA>.

The target audience for my application is: <TARGET AUDIENCE >

For the project, we need to define functional and non-functional requirements for this application. Functional requirements define the basic system behaviour, i.e., what the system can do. A non-functional requirement system properties and constraints as a whole. They specify acceptance criteria related to system qualities, such as performance, reliability, and usability.

A functional requirement has a name, short description (1-2 sentences), primary actor(s), success scenario(s), and failure scenario(s). Requirements should be comprehensive, feasible to implement in the scope of this 3-month project, and aligned with the application's goal.

Let's start with functional requirements. I will now provide you my initial specification of functional requirements and will ask you to refine them.

> **Prompt**

Let's start with the main actors and functional requirements. Actors are those individuals or external systems that will interact with the application. Functional requirements define the basic system behaviour, i.e., what the system can do.

The main actors of the application are: <ACTORS>

Main high-level requirement requirements are: <FR>.

Given this high-level description, list 6-7 project functional requirements that are specific, verifiable, and measurable. Recall that a functional requirement has a name (written in active / verb-style), short description (1-2 sentences), primary actor(s), success scenario(s), and failure scenario(s).

For now, for each requirement, list the name, short description (1-2 sentences) and primary actor(s) of the produced requirements.

Feel free to add/remove/update actors and functional requirements, as needed.

Ensure that the project satisfies the following constraints

1. At least one functional requirement that uses an external authentication service
2. At least one functional requirement that uses one additional external service
3. At least one functional requirement that incorporates real-time updates
4. All requirements should not be overly simplistic and should include some non-trivial logic

> Prompt (repeat for each requirement <#i>**)**

Let's consider one functional requirement at a time. For the functional requirement <FR_i_name>

Refine the success scenarios and failure scenarios of this requirement. A success scenario is a numbered sequence of steps in the normal flow of events in the system. A failure scenario describes what can go wrong in each step of the success scenario and how this is handled. A failure scenario should be numbered with the same number as the corresponding success scenario. That is, the list of failure scenarios does not have to start at 1. Ensure the success scenarios and failure scenarios fully capture the functionality of the requirement, and add points that may have been missed in the initial specification.

Here is an example:
Success Scenario:

1. Student selects “Register New Courses” from the menu.
2. System displays list of courses available for registering.
3. Student selects one or more courses he wants to register for.
4. Student clicks “Submit” button.
5. System registers student for the selected courses and displays a confirmation message.

Failure Scenarios:

2a. No courses are available for this student.
— 2a1. System displays error message saying no courses are available, and provides the reason & how to rectify if possible.
— 2a2. Student backs out of this use case and tries again after rectifying the cause.

5a. Some courses could not be registered.
— 5a1. System displays message showing which courses were registered, and which courses were not registered along with a reason for each failure.

5b. None of the courses could be registered.
— 5b1. System displays message saying none of the courses could be registered, along with a reason for each failure.

> Prompt

Let’s put everything together**.** List all functional requirements for the project using the format described below:
- The name of the requirement (written in active / verb-style)
- A short description of the requirement (1-2 sentences)
- Primary actor(s)
- Success scenario(s)
- Failure scenario(s)

> Prompt

The main actors are primary actors in at least one functional requirement. Consider all functional requirements in the text above and generate a consolidated list of all actors used. For each actor, describe in 1-2 sentences its role and list the names of all functional requirements where it is used.

> Prompt

Using the description above, create a UML use-case diagram as a LaTeX graph. It should contain a node for each actor and a node for each functional requirement. Use the name of an actor and the name of a functional requirement as name of the corresponding nodes. For each functional requirement, add a connection between the functional requirement and each of its primary actors. Add include, extend, and generalize relationships of the use case diagram, if necessary.

> Prompt

Let's consider the non-functional requirements next. Non-functional requirements describe system properties and constraints as a whole. They specify acceptance criteria related to system qualities such as performance, reliability, and usability. Examples of non-functional requirements include performance, safety, security, scalability, dependability, reusability, portability

Non-functional requirements should be specific, measurable, verifiable. For example:
- The user should not need more than 5 clicks to perform any action [usability]
- The message should be received within 5 seconds after it was sent [performance]
- The search should terminate within 7 seconds [performance]

Non-functional requirements should also be doable in the scope of the project. A specification of each requirement should include: textual description, justification (why needed, in 1-2 sentences), and validation approach (how to confirm, in 1-2 sentences).

Please produce a list of 2-3 main non-functional requirements for this app in the format below:
- a textual description
- an explanation for why this requirement is needed/relevant in 1-2 sentences
- how the requirement will be validated in 1-2 sentences.