# Inf2- SEPP 2024-25

# Tutorial 2 (Week 3, on SE)

# Notes on Answers

## 1 Introduction

## 2 Task 1: Requirements for an electronic health clinic

(a) For the definitions, check lecture slides and reading for Lecture 4, but also Wikipedia for non-functional requirements.

Examples of functional requirements using the required format:

- The system shall allow clinic administrators to enter patient appointments.
- The system shall allow doctors to enter details of the patient's condition and any treatment prescribed.
- The system shall allow nurses to enter details of the patient's condition and any treatment prescribed. (note: it is best to write the above as two separate requirements for complete clarity)
- The system shall allow patients to provide changes in their health one week after each appointment by using a web form.

Examples of non-functional requirements which are clearly relevant from the system description, also classified into categories:

- Security: The system shall protect the personal information of its users.
- Privacy: The system shall not leak the patients' personal or treatment information to third parties.
- Performance: The system shall respond to patient actions within 5 seconds.
- Usability: The system's patient interfaces shall be easy to use by elderly, ill users, or users with disabilities.

Here, it is important to note the distinction (sometimes blurred) between non-functional and functional requirements (see Lecture 4 slide 6 and associated explanation). In particular, the following can be functional requirements for the first 2 non-functional requirements above:

- The system shall use two-factor authentication (related to security non-functional requirement above).
- The system shall encrypt any data about patients' personal data and treatment information (related to privacy non-functional requirement above)

(b) For the definition, check lecture slides for Lecture 4.

For the above system, stakeholders directly mentioned in the situation description, along with example brief explanations of their interests in the new system, are:

- *Patients*: - wanting system that helps doctors and nurses deliver good health care to them
- *Doctors*: - wanting system that supports them in delivering good health care to patients and reporting outcomes to the country
- *Nurses*: - wanting system that supports them in delivering good health care to patients
- *Administrators*: - wanting system that helps them provide support to patients, doctors and nurses

Then, as the just-elected ReformHealthCare political party is important, we have:

- *Politicians* - wanting to see health outcomes
- *Voters* - wanting to see health outcomes

Further stakeholders include:

- *Tax payers* - wanting system to succeed and value for money
- *Software engineers* - wanting to deliver system on time within budget meeting requirements

For this exercise, we could say "doctors and nurses" together or rename them to e.g. "healthcare professionals", which would be acceptable given the system description, but this is unlikely to be right in real life. Technically, stakeholders in the last 2 groups are acceptable as a solution, though if you list those rather than those that are more explicit you are missing obvious information.

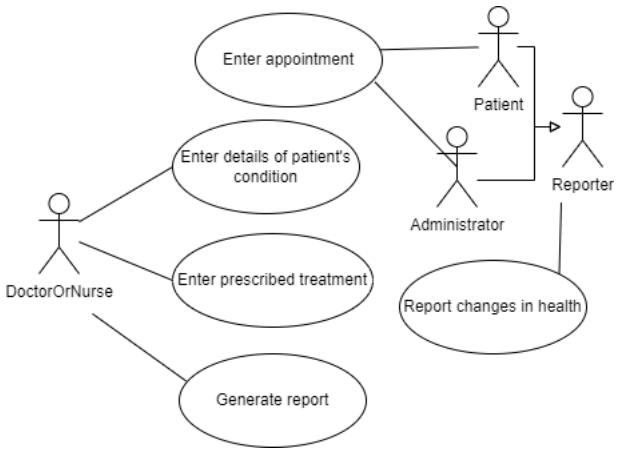
(c) See Lecture 4. Any well justified answer discussing requirements elicitation approaches accepted here. For example, one could say that because doctors and nurses are busy professionals, observation and/or being able to attend team meetings with them may be best (but requires thorough ethical clearance), while interviews and facilitated meetings may not be feasible.

(d) Check lecture slides and reading for Lecture 5, starting from the definitions of use cases. E.g. Advantages: help discussing and eliciting requirements from stakeholders as easy to understand; manage conflicts between requirements of different stakeholders; Disadvantages: only capture requirements that are about interactions, you may fall into the temptation to make them too detailed and thus constrain the design, also see others in last slide of Lecture 5.

(e) Any reasonable solution acceptable. It should include: actors (DoctorOrNurse, Patient, Administrator), use cases (Enter appointment, Enter details of patient's condition, Enter prescribed treatment, Report changes in health, Generate reports), associations, details of notation e.g. capitalisation, comments.

Notes: Actor names should always include capitalisation of each word, and no blanks, like for classes. There should only be one primary actor per use case, representing the role triggering it.

An example is:




The patient and administrator can both report changes in health, and we assume that the administrator would be using the same functionality as patients (i.e. the same web form) to do this. This assumption would need to be checked with the stakeholders. As corresponding Patient and Administrator actors woud both need to play the role of primary actor for the same use case ("Report changes in health") and this is not allowed, generalization is used to identify one superactor with their common role in this context, that would become the single primary actor.

The Patient actor is shown as interacting with the *Enter appointment* use case, as it is reasonable to assume that patients are notified when an appointment is made. Therefore, the Patient actor would be a supporting actor in this context. As with the *Report changes in health* use case, maybe the system could allow patients to make appointments directly. Such assumptions would need to be clarified with the stakeholders, and in such a context the Patient would become a second primary actor alongside the Administrator, so generalization would again be needed like above to end up with a single primary actor.

The specification doesn't make clear who asks for the reports to be generated and who they are intended for. For example, given the outcomes are of wide interest, we might

assume some reports are desired by politicians, some government statistics office, and ultimately the public. The diagram assumes the doctors/ nurses in the first place ask for reports, as presumably these reports might help them monitor and improve the controversial treatments. Such an assumption would need to be clarified with the stakeholders.

All doctors and nurses are represented by a single actor DoctorOrNurse because they have access to the same functionality (i.e. are connected to the same use cases), and for lack of a better term to describe both. Also, separate Doctor and Nurse actors were not needed as the description did not specify any use cases that would be available only for one or the other.

# 3 Task 2: Requirements for a food box system

(a) A sample solution is included below.

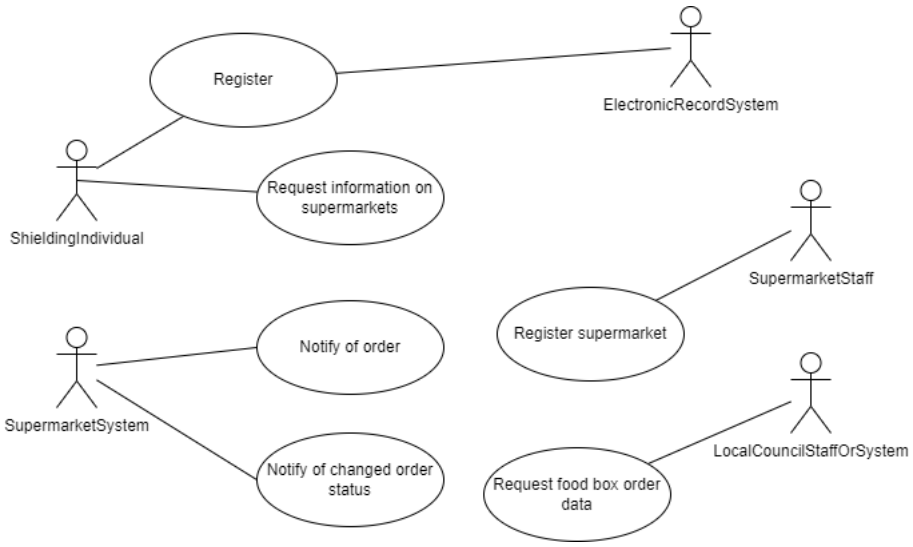
Important notes regarding use cases, in response to "Things to consider" from tutorial:

- In use cases, we only represent units of functionality from our system, and not from external systems (e.g. here, the supermarket system); Also, we do not represent actions from the environment (e.g. delivering the food box).
- Use cases and their scenarios should be all about the interaction actor-system, and not about processing internal to the system (e.g. system calculating distance supermarket-individual's post code); also, the system itself cannot be a use case actor (very frequent mistake in student solutions!)
- Use cases should be high level and represent clearly distinguishable chunks of functionality, but their scope can be reduced as development progresses. If a real-world process involves a mix of interaction with our system- actions outside the system-interaction with our system again, it is clearest if we split the interactive parts into separate use cases. E.g. instead of having one unclear and easy-to-get-wrong "Order from supermarket" use case (the title of which does not even make sense because ordering is not part of our system), it is better to split into parts that do happen on our system only, i.e. "Request information on supermarkets" and "Notify of order", triggered by the correct different primary actors.

In the diagram, actors which are systems are distinguished from actors which are human for clarity. While a new supermarket would need to involve a member of staff registering it onto the system, local councils could either have their system integrated already with ours, or getting order information may instead require their staff to access it (ambiguity). For this reasons, the solution includes an "or" in the local council actor, which would need further clarification from the government and local councils themselves. A more complete solution would also involve all human actors authenticating on our system (an assumption could be made that registering/logging in are involved).

Also, please note that all actors must be roles, e.g. the actor is ElectronicRecordSystem and not PublicHealthScotlandSystem.

Which is the primary and which is the supplementary actor of a use case is not apparent in the diagram, but will be explained in the use case descriptions (see b).




(b) Use case descriptions for 2 important use cases:

**Use case name:** **Register**
**Primary actor:** ShieldingIndividual
**Supporting actors:** ElectronicRecordSystem
**Summary:** The shielding individual registers onto the system.
**Precondition:** None.
**Trigger:** Shielding individual requires to register.
**Success guarantee:** Shielding individual is allowed to do other actions on the system.
**Failure guarantee:** Shielding individual is denied access to other actions on system.
**Main Success Scenario:**
1. System requires individual's CHI number
2. Shielding individual provides CHI number
3. System requests electronic record system to check CHI number
4. Electronic record system confirms correct CHI number to the system
5. System requests individual's details from electronic record system
6. Electronic record system provides individual's details to system
7. System confirms successful registration to individual

**Extensions:**
3a. Individual provided empty or invalid CHI number
.1 Return to step 1
4a. Electronic record system provides error that the individual is not shielding
.1 System gives error message to individual and use case ends
7a. There is an error in getting the individual's details
.1 System gives error message to individual and use case ends

**Use case name:** **Request information on supermarkets**
**Primary actor:** ShieldingIndividual
**Supporting actors:** None

**Summary:** The shielding individual requests information on supermarkets.
**Precondition:** The sheilding individual is registered onto the system.
**Trigger:** Shielding individual chooses to get a food box from a supermarket.
**Success guarantee:** Shielding individual receives the list of supermarkets with delivery times and links.
**Failure guarantee:** Shielding individual does not get the list of supermarkets.
**Main Success Scenario:**
1. System provides a list of supermarkets with delivery times and links.
**Extensions:**
1a. Individual placed a food box order less than a week ago
.1 System gives error message requesting individual to try again after one week has elapsed, use case ends
1b. Individual's postcode is too far for delivery by any of the registered supermarkets
.1 System informs that no supermarkets delivering food boxes available in proximity, use case ends.

(c) Stakeholders who are not use case actors: the Scottish Government, delivery service providers, farmers working with participating supermarkets, the NHS, etc.. Actors that are not stakeholders: any actors which are systems.

Main differences stakeholders - actors: stakeholders are always human while actors can also be systems; actors must interact with the system while stakeholders may not.

(d) Look up 'measurable' in the lecture on Requirements Engineering (lecture 4, week 2). Also, you can find examples in section 4.1.2 (Non-functional requirements) of Chapter 4 of Sommerville SE.

Some examples (be specific about the numbers as otherwise they are not measurable!):

- Order information shall only be retained for 6 months.
- The system shall scale to handle 500 supermarkets.

# 4 Task 3 (Advanced): A lift system

Here we introduce the lift as an actor: it is an agent external to the system whose cooperation in moving (observable by our system) is needed in order to fulfill the use case goals. However, we assume that buttons and lights are part of our system, and observable by the caller.

We imagine some extensions for mechanical failures. These highlight the need for some alarm system, not mentioned in the question system description. We assume there is some mechanical arrangement by which the doors at each floor open together with the doors on the lift itself when the lift is at the floor. For simplicity we assume the lift position displays track the position of the lift. We skip mentioning updating these displays in the use cases.

**Use case name: Call lift**
**Primary actor:** Caller
**Supporting actors:** Lift

**Summary:** Caller requests lift to come to their floor
**Precondition:** Lift is stopped at some floor with doors open
**Trigger:** Caller presses call button at some floor
**Guarantee:** Lift is at floor of call, doors are open
**Main Success Scenario:**
1. System turns on call button light and closes doors
2. System commands lift to move to floor of pressed button
3. Lift arrives at floor of pressed button
4. System turns off light in call button
5. System commands doors to open
**Extensions:**
1a. Lift is already at floor of call
.1 System finishes MSS immediately
3a. Lift motor fails before arrival at destination floor
.1 System sounds alarm
5a. Doors stick closed
.1 System sounds alarm
**Notes:**

- It would be easy to tweak this use-case to relax the *doors open* precondition..
- A possible non-functional requirement associated with this use-case is that the lift shall arrive at the floor of the call within 30 s (be specific as otherwise it's not measurable!).

**Use case name:** **Request destination floor**
**Primary actor:** Caller
**Supporting actors:** Lift
**Precondition:** Lift is stopped at some floor. Caller is in lift.
**Trigger:** Caller presses one of destination floor buttons inside lift
**Guarantee:** Lift is at destination floor selected, doors are open
**Main Success Scenario:**
1. System turns on destination button light and, if doors are open, closes them
2. System commands lift to move to destination floor
3. Lift arrives at destination floor
4. System turns off light in destination floor button
5. System commands doors to open
**Extensions:**
1a. Lift is already at destination floor
.1 System finishes MSS immediately, ensuring doors are open

**Notes:**

- Extensions could be added for mechanical failure conditions as before
- A possible non-functional requirement associated with this use-case is that the lift shall arrive at the destination floor within 30 s (be specific as otherwise it's not measurable!).

Cristina Adriana Alexandru, 22 January 2025.