UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

**Department of Computer Science**
**COS 730**
**Software Engineering 1**

# Individual Project: Assignment 1 Instructions

Stacey Baror


# 1 Introduction

Choose a system of your choice that should contain a minimum of four core components that could be treated as a service. Ensure you choose your system carefully as the system you choose will be used for all assignments throughout the semester.

## 1.1 Administration and Infrastructure

- The deadline for this assignment is **12 April @ 23h30 PM**
- Upload your PDF using ClickUp
- For modelling (UML diagrams), you are required to model the UML diagrams for this assignment and **ALL** future deliverable of COS 730 using any of the tools below.
  - Visual-paradigm `https://online.visual-paradigm.com/`
  - Another option for modelling is `draw.io`
  - Or Magic draw `https://www.nomagic.com/products/magicdraw`
  - You are always advised to use a proper UML standard syntax to avoid losing marks.
  - You should use LATEX to write your documentation. Here is an online LATEX- Overleaf (`https://www.overleaf.com/`)
  - At the completion of your SRS document, Upload **ONE PDF** version of your SRS documentation to ClickUp for grading.
  - Familiarise yourself with latex, as it is only acceptable form of documentation for COS 730. It is advisable (not compulsory) to integrate your git/github/bitbucket with SRS documentation.
  - For grading purposes, ensure you invite me to your git - username: staceybaror, email: stacey.baror@tuks.co.za

# 2 Software Requirements Specification

## 2.1 Functional Requirements Deliverable

The main deliverable for this assignment is the SRS document for your chosen system. Use the given project template as well as the SRS structure to guide you and get you started. Your SRS document should contain the following sections:

- Introduction
- User characteristics
- Functional requirements
  - Use case diagram
  - Traceability matrix
- Domain model
- Non-functional requirements
  - (Quality requirements)

### 2.1.1 Introduction

Describe the purpose and scope of the software system- Explain the vision and objectives. State the business need for the application and summarise the scope of the project.

### 2.1.2 User characteristics

Describe the general characteristics of the intended users. Include assumptions about their expected educational level, experience, expertise, and technical skills. - List all intended users and explain for what purpose each of them would use the system.

### 2.1.3 Functional requirements

Specify functional requirements to satisfy the use cases. Formulate them in terms of requirements and sub-requirements.

Describe the functional requirements in terms of use cases and requirements. Illustrate the use of cases using use case diagrams. Provide a requirement use case with a traceability matrix and ensure that your design with service-oriented in mind.

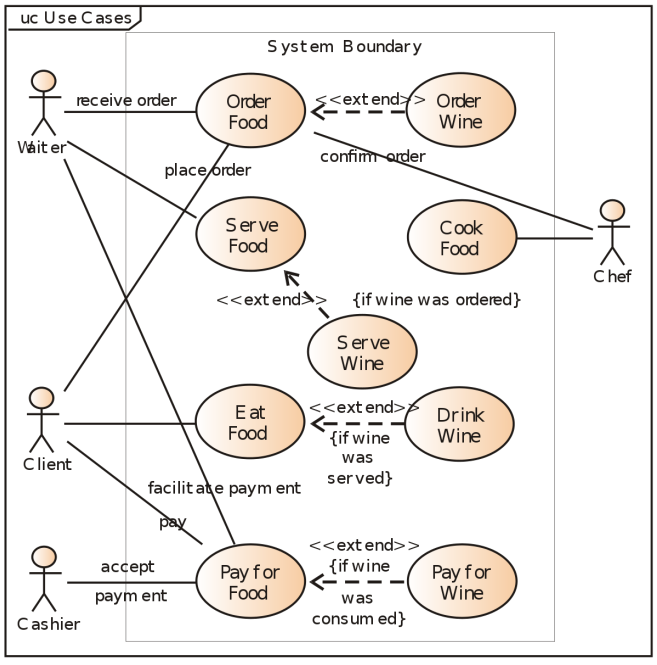
Number the use cases using dotted numbers such as U1, U2.2.1, etc. Number the requirements using dotted numbers such as R1, R1.1, R2.1.3, etc. Use cases are business processes which users of the system should be able to do, while requirements are capabilities that the system must deliver.

Draw high level use case diagrams for the use cases of your system. Number them for traceability purposes.

### 2.1.4 Use Case diagram

Draw high level use case diagrams for the use cases of your system. Number them for traceability purposes.

An example UML use case diagram specifying the functional requirements for a use case is shown in the following image: An example UML use case diagram specifying the functional requirements for a use case is shown in Figure 1.




https://en.wikipedia.org/wiki/Use_case_diagram

Figure 1: An example of a UML use case diagram

For each concrete use case a use case diagram with the required functionality in the form of includes and extends relationships to lower level use cases. Use `<<include>>` to specify sub-functionality for a given functionality and `<<extend>>` if the sub-functionality is optional. Show the required functionality within a system boundary as actions (ovals). The actors (stick figures) are people or other systems requesting or delivering services (labels on the connection line between an actor and an action).

Group the requirements in logical modules that can be implemented as inter-dependent subsystems with low coupling and high cohesion.

#### 2.1.5 Trace-ability matrix - Requirements VS Sub-systems

A matrix with requirements numbers as rows and the sub-systems as columns. Include both functional and quality requirements. Indicate with x in the cells which subsystem will implement the solution to satisfy the requirement.

### 2.2 Non-Functional Requirements(Quality requirements)

Specify each of the quality requirements and constraints which are relevant to the system. Examples of quality requirements include performance, reliability, scalability, security, flexibility, maintainability, auditability/moni- torability, integrability, cost, usability.

#### 2.2.1 Quality requirements

Examples of quality requirements include performance, reliability, scalability, security maintainability, usability.

#### 2.2.2 Specify and quantify

For each of the quality requirements relevant to your system, quantify. For example:

- Availability: Test system is expected to be at least 99.5% up time.
- Scalability: Test system is designed to handle 50millions users request per sec.
- For security: Role-based access control in to used.

Each of these quality requirements need to be either **quantified or at least be specified in a testable way**.

Domain model Show the domain concepts and their relations using UML class diagram syntax.

## 3 Assessment rubric

| **Item** | **Marks** |
|---|---|
| Introduce the system | 5 |
| User characteristics (describe the users) | 10 |
| List the functional requirements | 5 |
| Traceability matrix | 10 |
| Use case diagrams (modelling) *drawing without description ='0'* | 25 |
| Non-Functional (Quality requirements and quantification) | 20 |
| Domain model (modelling) *drawing without description ='0'* | 25 |
| **Total** | **100** |