Systems Design and Databases (CIS1018-N)

Week 2

UML, UML Tool(s), Use Cases & Wireframe

Teaching Team

Module Leader & Lecturer: Dr Yar Muhammad

Email: Yar.Muhammad@tees.ac.uk

Office: G0.39 (Greig Building)

Tutor:

- Dr Mengda He
- Mr Mansha Nawaz
- Mr Vishalkumar Thakor

My Academic Hub Time Slots:
Monday 10:00 - 11:00 in Room IT1.13 (Europa Building)
Tuesday 13:00 - 14:00 in Room IT1.13 (Europa Building)

- See Blackboard Ultra for online materials: https://bb.tees.ac.uk/

Lectures & IT Labs

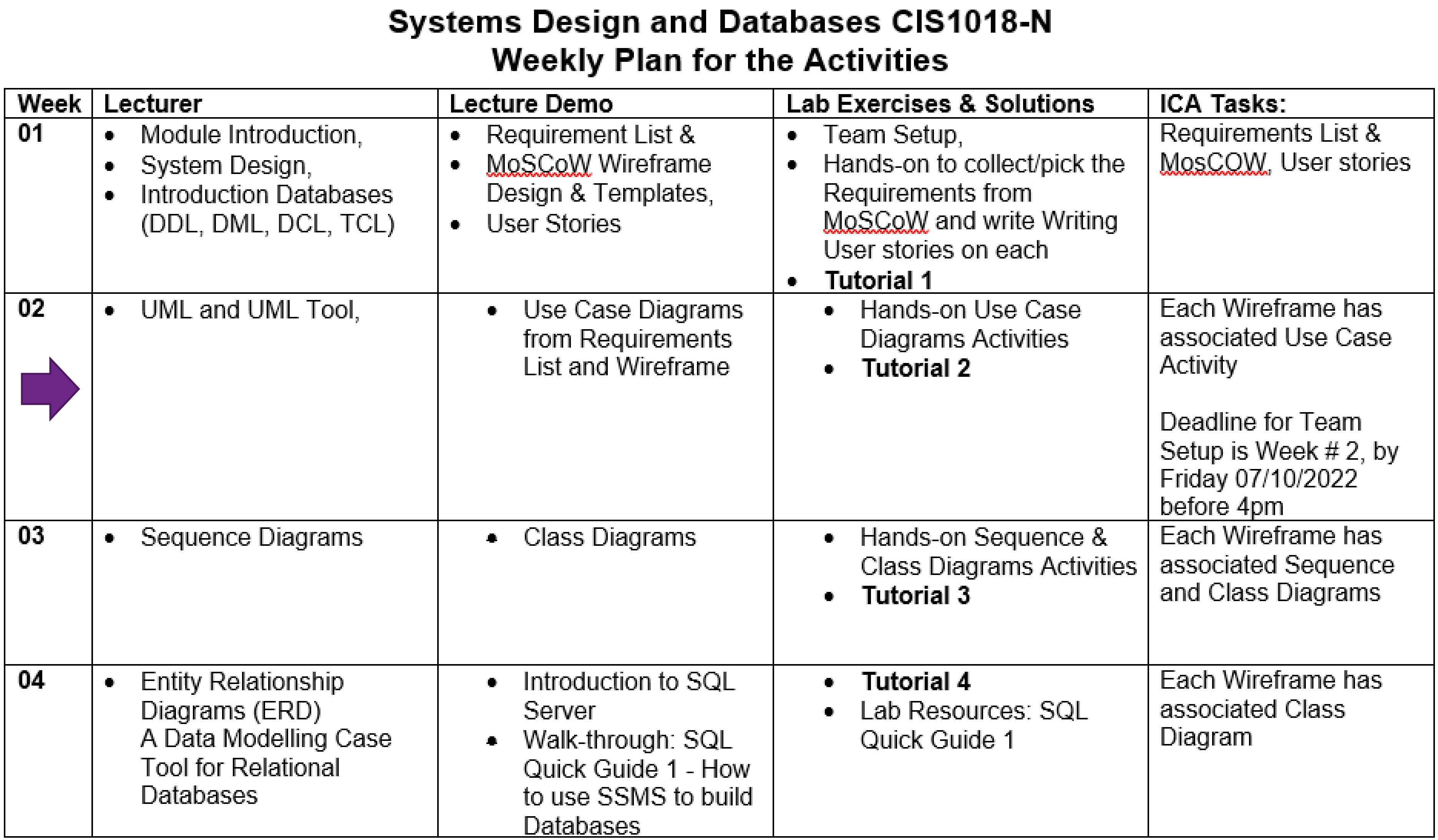
Lectures – Dr Yar Muhammad	Tuesdays @ 2-3 pm	Thursdays @ 1-2 pm
Week 1 – Week 12	CL1.87	

Tutor – Thursday	IT Lab Session Room #: IT2.42
Mr Mansha Nawaz M.Nawaz@tees.ac.uk	Time: 3 – 5 pm

Tutor – Friday	IT Lab Session Room #: OL3
Dr Yar Muhammad Yar.Muhammad@tees.ac.uk	Time: 9 – 11 am & 11 am – 1 pm
Dr Mengda He M.He@tees.ac.uk	Time: 9 – 11 am
Mr Vishalkumar Thakor V.Thakor@tees.ac.uk	Time: 11 am – 1 pm & 1 – 3 pm
Mr Mansha Nawaz M.Nawaz@tees.ac.uk	Time: 1 – 3 pm

Schedule 1/3

Systems Design - UML



Systems Design and Databases CIS1018-N
Weekly Plan for the Activities

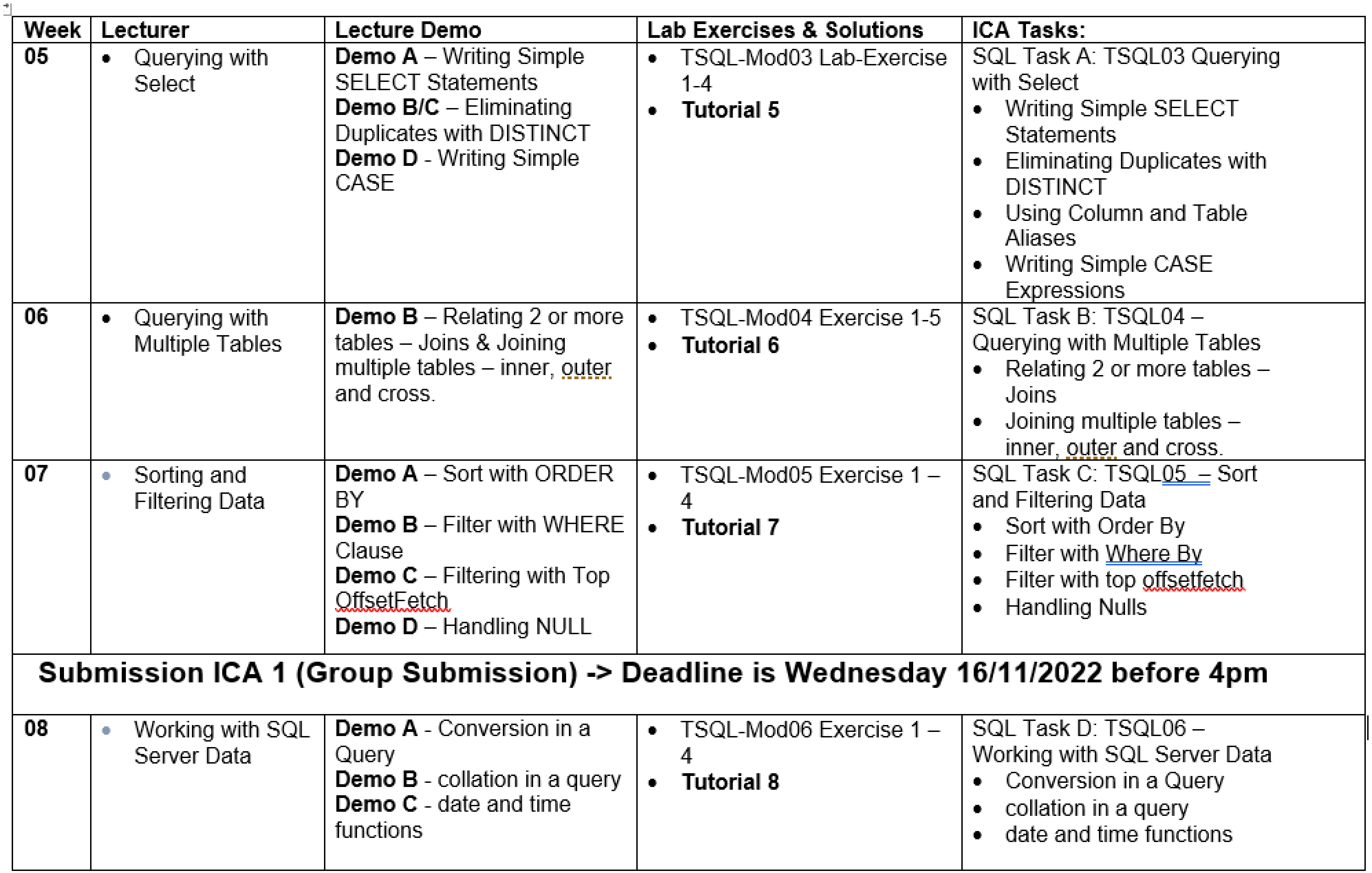
Week	Lecturer	Lecture Demo	Lab Exercises & Solutions	ICA Tasks:
01	• Module Introduction, • System Design, • Introduction Databases (DDL, DML, DCL, TCL)	• Requirement List & • MoSCoW Wireframe Design & Templates, • User Stories	• Team Setup, • Hands-on to collect/pick the Requirements from MoSCoW and write Writing User stories on each • **Tutorial 1**	Requirements List & MosCOW, User stories
02	• UML and UML Tool,	• Use Case Diagrams from Requirements List and Wireframe	• Hands-on Use Case Diagrams Activities • **Tutorial 2**	Each Wireframe has associated Use Case Activity Deadline for Team Setup is Week # 2, by Friday 07/10/2022 before 4pm
03	• Sequence Diagrams	• Class Diagrams	• Hands-on Sequence & Class Diagrams Activities • **Tutorial 3**	Each Wireframe has associated Sequence and Class Diagrams
04	• Entity Relationship Diagrams (ERD) A Data Modelling Case Tool for Relational Databases	• Introduction to SQL Server • Walk-through: SQL Quick Guide 1 - How to use SSMS to build Databases	• **Tutorial 4** • Lab Resources: SQL Quick Guide 1	Each Wireframe has associated Class Diagram

Analysis

Design

Schedule 2/3

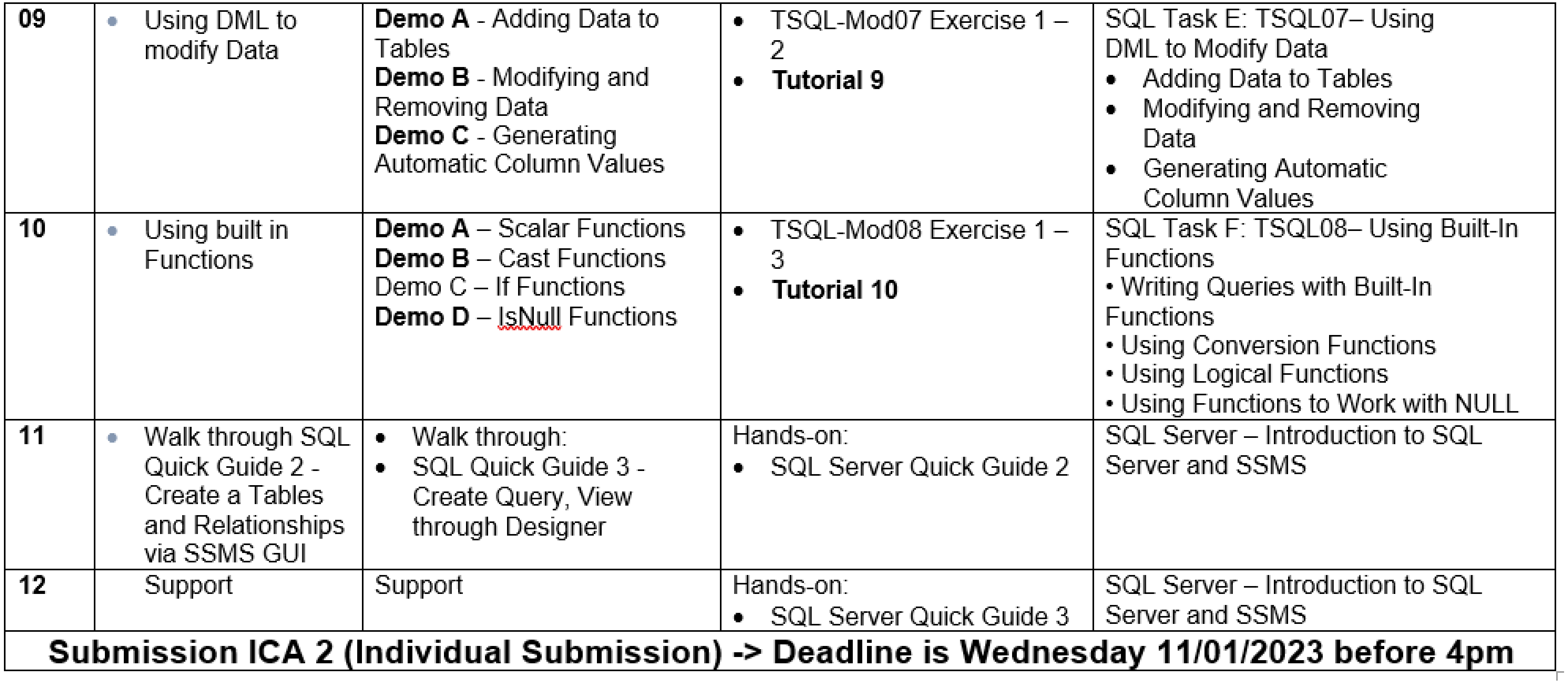
Databases - TSQL



Week	Lecturer	Lecture Demo	Lab Exercises & Solutions	ICA Tasks:
05	• Querying with Select	**Demo A** – Writing Simple SELECT Statements **Demo B/C** – Eliminating Duplicates with DISTINCT **Demo D** - Writing Simple CASE	• TSQL-Mod03 Lab-Exercise 1-4 • **Tutorial 5**	SQL Task A: TSQL03 Querying with Select • Writing Simple SELECT Statements • Eliminating Duplicates with DISTINCT • Using Column and Table Aliases • Writing Simple CASE Expressions
06	• Querying with Multiple Tables	**Demo B** – Relating 2 or more tables – Joins & Joining multiple tables – inner, outer and cross.	• TSQL-Mod04 Exercise 1-5 • **Tutorial 6**	SQL Task B: TSQL04 – Querying with Multiple Tables • Relating 2 or more tables – Joins • Joining multiple tables – inner, outer and cross.
07	• Sorting and Filtering Data	**Demo A** – Sort with ORDER BY **Demo B** – Filter with WHERE Clause **Demo C** – Filtering with Top OffsetFetch **Demo D** – Handling NULL	• TSQL-Mod05 Exercise 1 – 4 • **Tutorial 7**	SQL Task C: TSQL05 – Sort and Filtering Data • Sort with Order By • Filter with Where By • Filter with top offsetfetch • Handling Nulls
Submission ICA 1 (Group Submission) -> Deadline is Wednesday 16/11/2022 before 4pm				
08	• Working with SQL Server Data	**Demo A** - Conversion in a Query **Demo B** - collation in a query **Demo C** - date and time functions	• TSQL-Mod06 Exercise 1 – 4 • **Tutorial 8**	SQL Task D: TSQL06 – Working with SQL Server Data • Conversion in a Query • collation in a query • date and time functions

Schedule 3/3

Databases - TSQL



09	• Using DML to modify Data	**Demo A** - Adding Data to Tables **Demo B** - Modifying and Removing Data **Demo C** - Generating Automatic Column Values	• TSQL-Mod07 Exercise 1 – 2 • **Tutorial 9**	SQL Task E: TSQL07– Using DML to Modify Data • Adding Data to Tables • Modifying and Removing Data • Generating Automatic Column Values
10	• Using built in Functions	**Demo A** – Scalar Functions **Demo B** – Cast Functions Demo C – If Functions **Demo D** – IsNull Functions	• TSQL-Mod08 Exercise 1 – 3 • **Tutorial 10**	SQL Task F: TSQL08– Using Built-In Functions • Writing Queries with Built-In Functions • Using Conversion Functions • Using Logical Functions • Using Functions to Work with NULL
11	• Walk through SQL Quick Guide 2 - Create a Tables and Relationships via SSMS GUI	• Walk through: • SQL Quick Guide 3 - Create Query, View through Designer	Hands-on: • SQL Server Quick Guide 2	SQL Server – Introduction to SQL Server and SSMS
12	Support	Support	Hands-on: • SQL Server Quick Guide 3	SQL Server – Introduction to SQL Server and SSMS
Submission ICA 2 (Individual Submission) -> Deadline is Wednesday 11/01/2023 before 4pm				

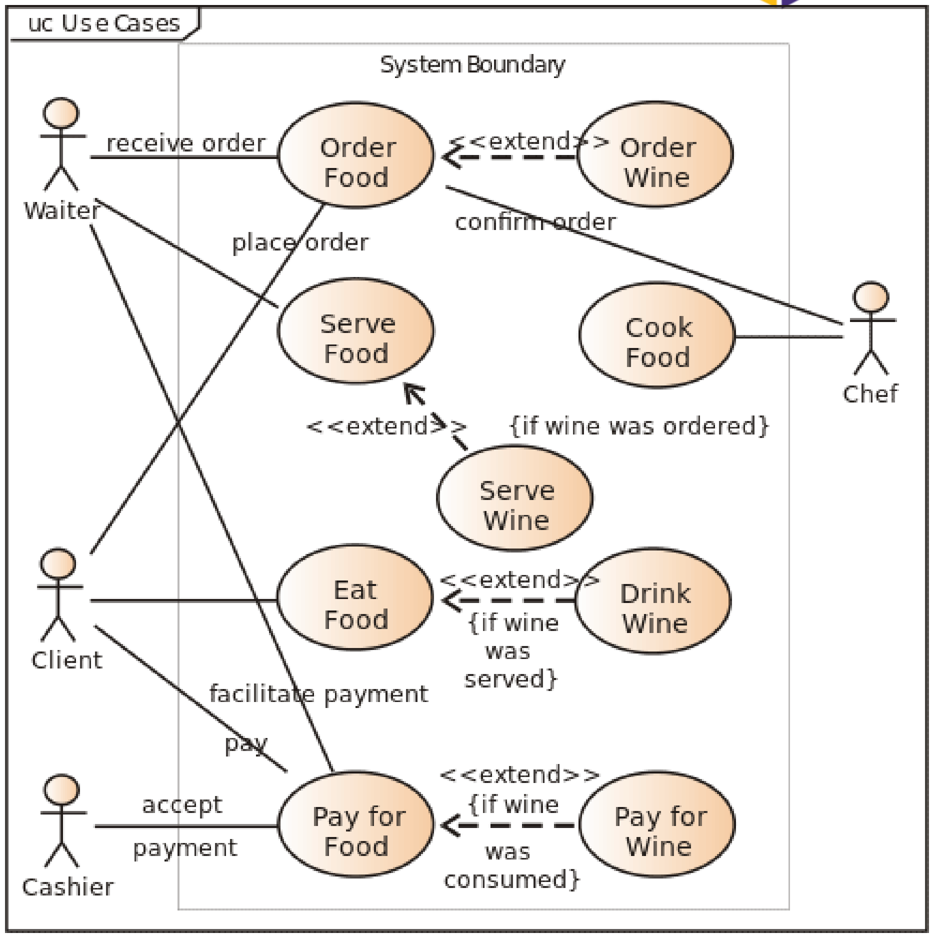
This Week’s Agenda

- UML
- UML Tools
- Wireframe
- Use Cases Diagrams

Unified Modelling Language (UML)

UNIFIED MODELING LANGUAGE™

- UML a visual language for specifying, constructing, and documenting the artifacts of systems.
- The UML is a general-purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.
- UML is not a programming language, it is rather a visual language



UML Tools

- **Lucidchart:** Many developer's go-to diagramming tool, Lucidchart provides a suite of simple tools. ...
- **Gleek.io:** Gleek.io creates several types of UML diagrams: sequence diagrams, class diagrams, and object diagrams. ...
- **Cacoo:** is web-based software for designing and drawing diagrams
- **Gliffy:** is a visual diagram which is web application used to create and edit diagrams and flowcharts.
- **EdrawMax:** Edraw Max is a 2D business technical diagramming.
- **Microsoft Visio Pro:** is software for drawing a variety of diagrams
- **Diagrams.net/Draw.io:** is a free and open source cross-platform graph drawing software.

Why Diagrams.net (Draw.io)?

- Diagrams.net is a free and open source cross-platform graph drawing software.
- Its interface can be used to create diagrams such as flowcharts, wireframes, UML diagrams, organizational charts, and network diagrams.

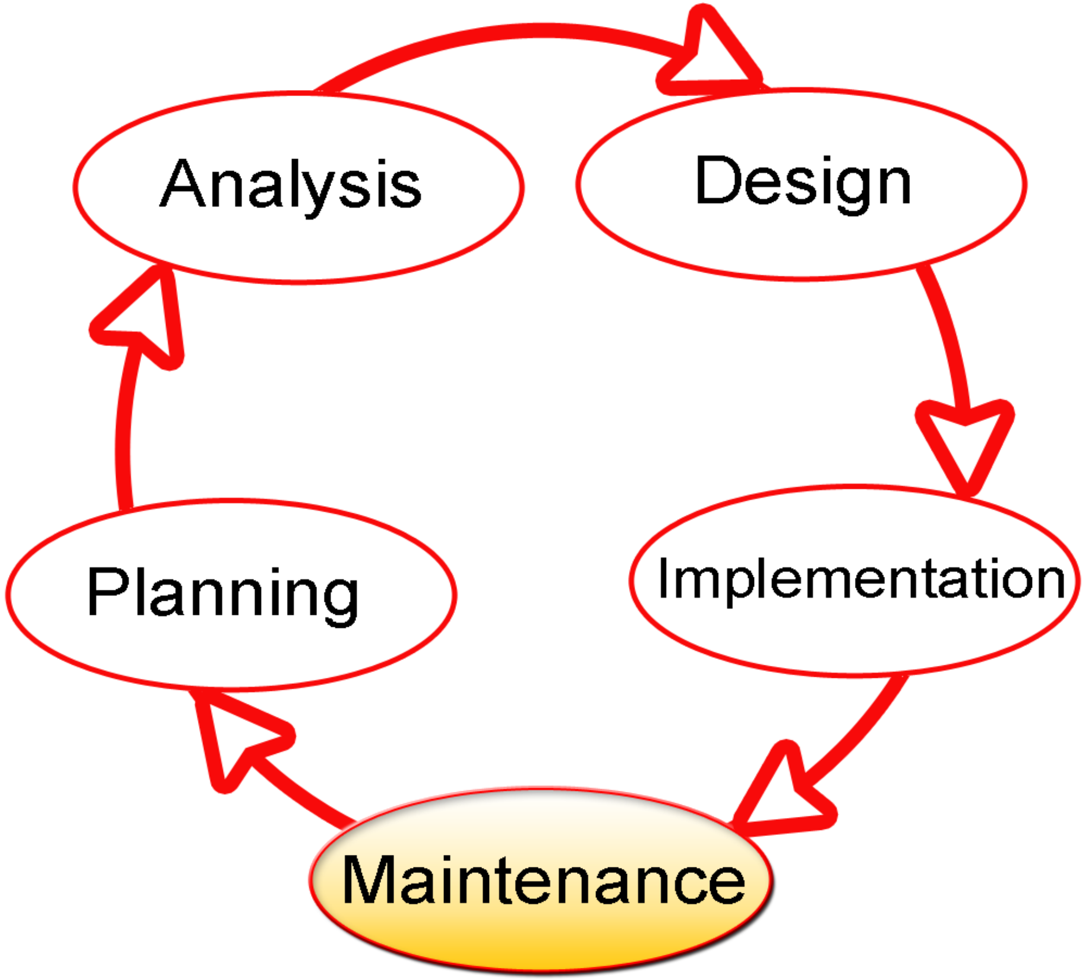
diagrams.net

draw.io

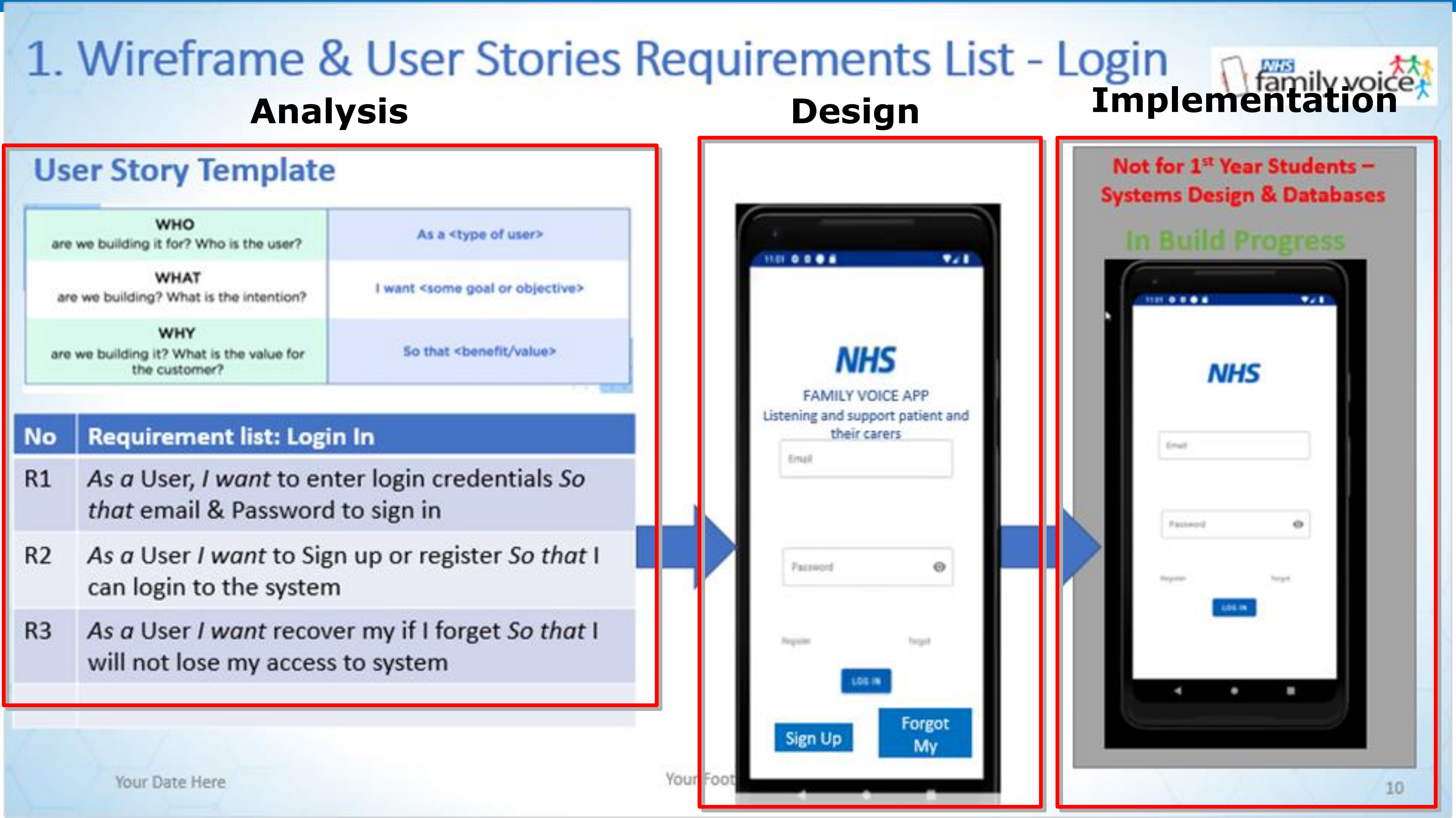
DEMO

Diagrams.net or Draw.io

System Development Life Cycle (SDLC)



From Analysis -> Design - **Wireframe**



1. Wireframe & User Stories Requirements List - Login

Analysis

User Story Template

WHO are we building it for? Who is the user?	As a <type of user>
WHAT are we building? What is the intention?	I want <some goal or objective>
WHY are we building it? What is the value for the customer?	So that <benefit/value>

No	Requirement list: Login In
R1	*As a* User, *I want* to enter login credentials *So that* email & Password to sign in
R2	*As a* User *I want* to Sign up or register *So that* I can login to the system
R3	*As a* User *I want* recover my if I forget *So that* I will not lose my access to system

Design

Implementation

Not for 1st Year Students – Systems Design & Databases

In Build Progress

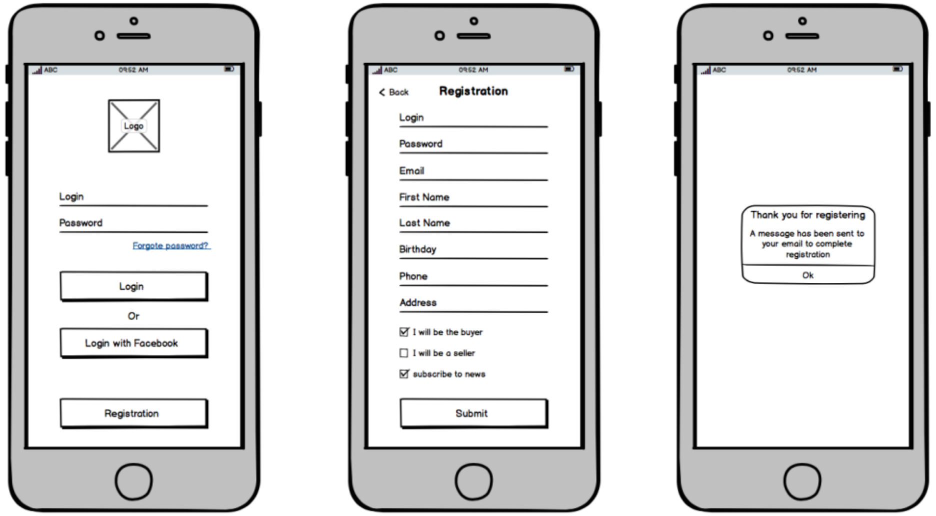
Your Date Here

Your Foot

10

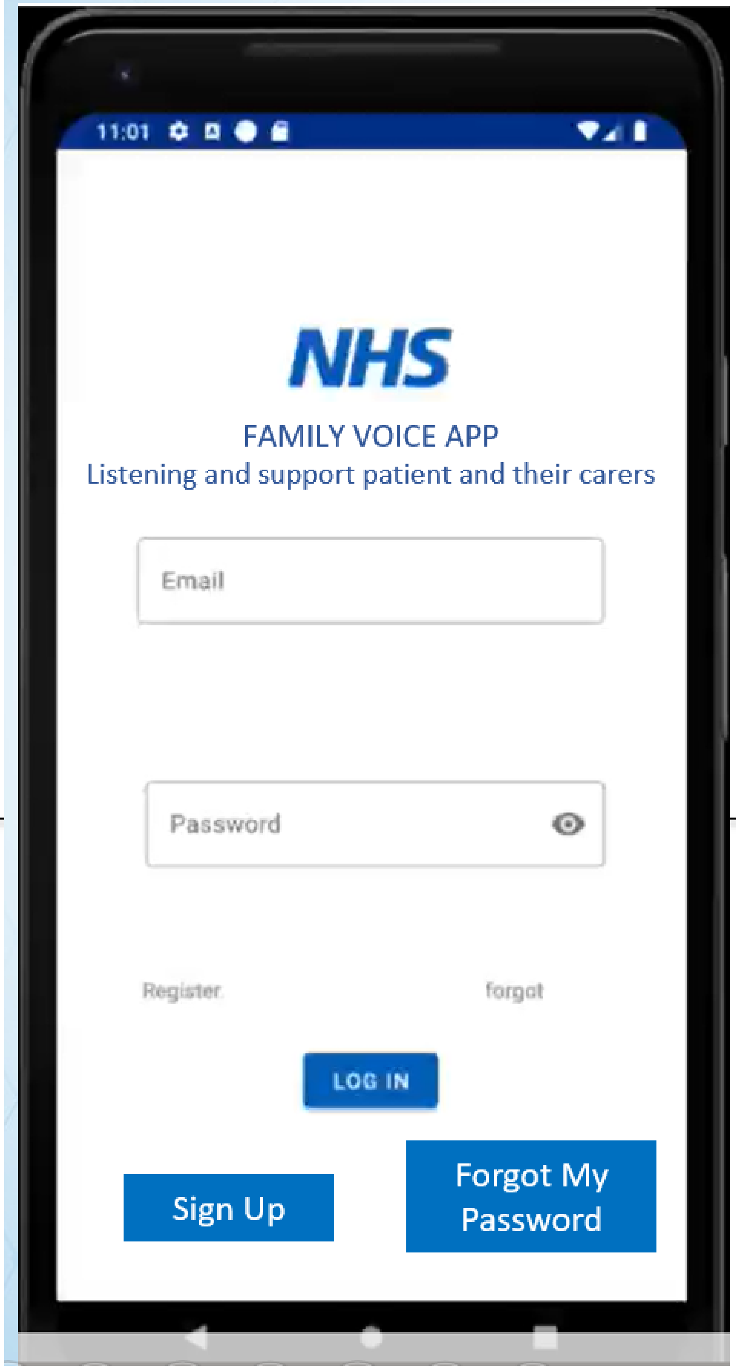
What is Wireframe ?

- A wireframe is a basic, **two-dimensional visual representation** of a web page, app interface, or product layout.
- You can think of it as a low-fidelity**, functional sketch**. **Product designers** and UX (user experience) professionals draw up wireframes to communicate how they plan to arrange and prioritize features, and how they **intend for users to interact** with its product or website/App.
- Wireframes typically **depict only functionality**, not the true style and visual elements of the final product.
- Because of that most wireframes look simple: grayscale instead of colours, placeholders for images, and Lorem Ipsum for text.



DEMO

Demonstration on Wireframe by using the Diagrams.net



NHS

FAMILY VOICE APP

Listening and support patient and their carers

Email

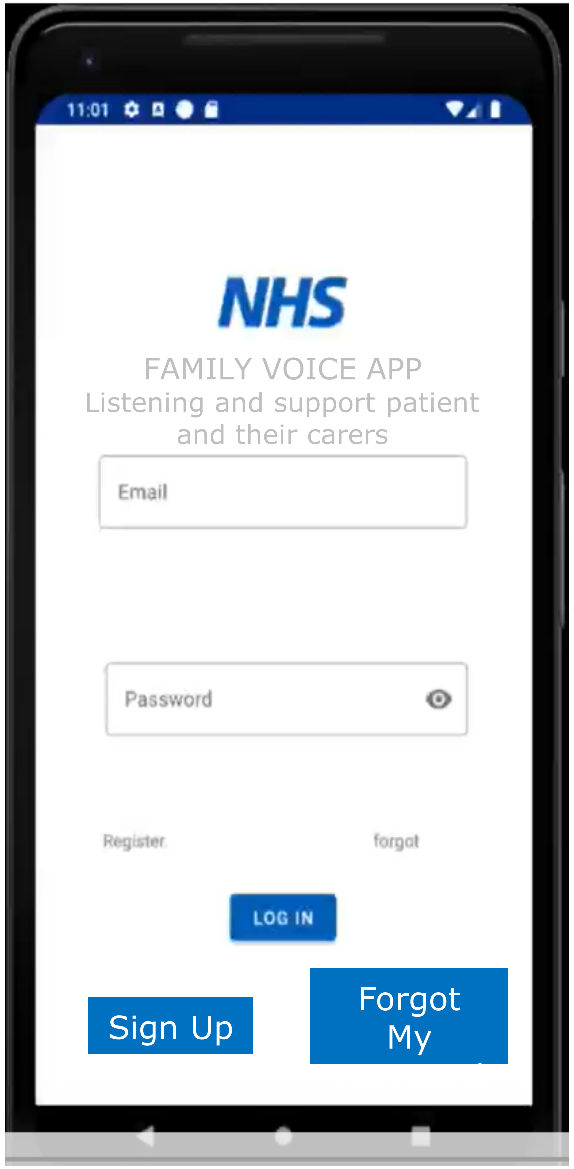
Password

Register

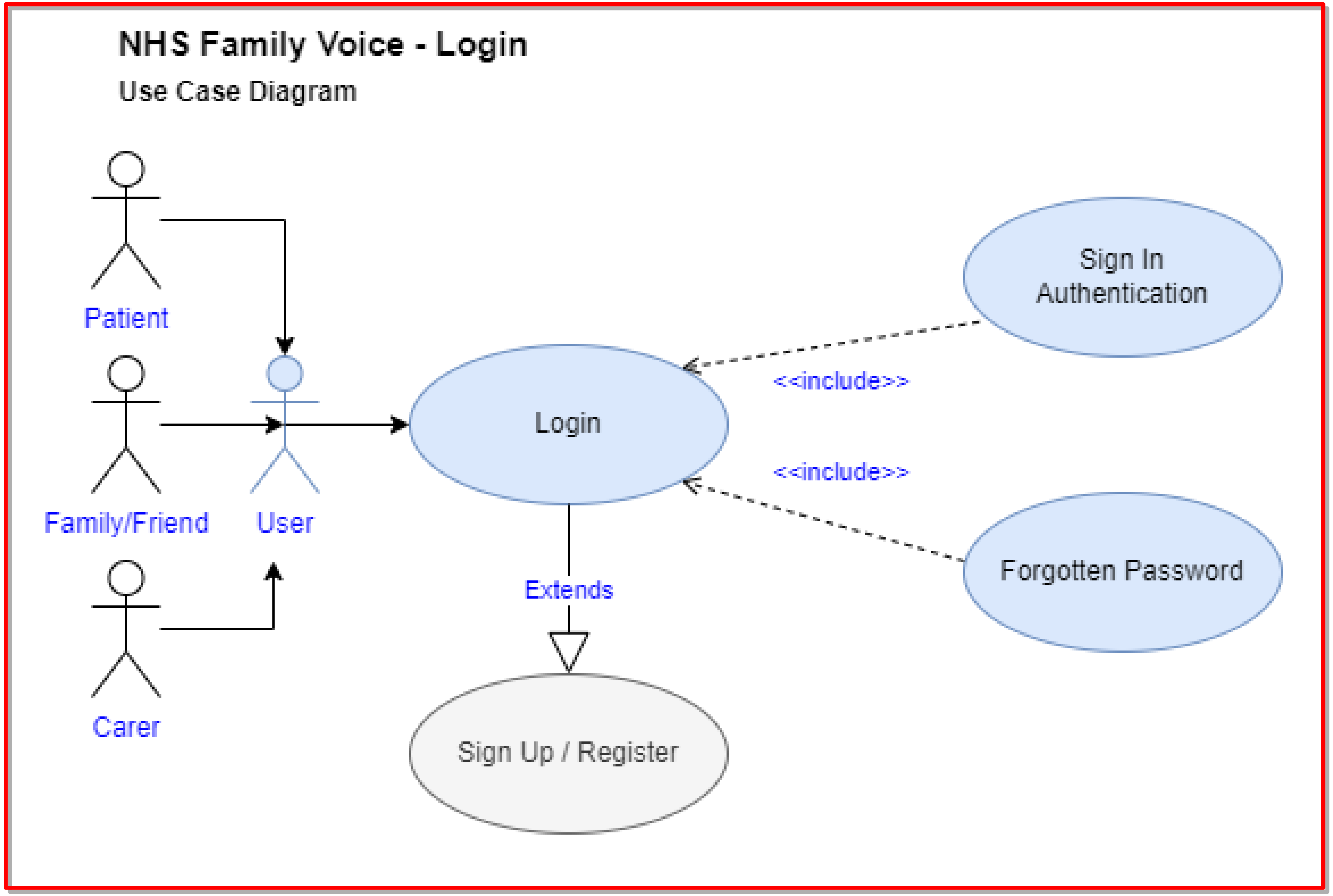
forgot

From Analysis -> Design -> **Use Case**

Wireframe

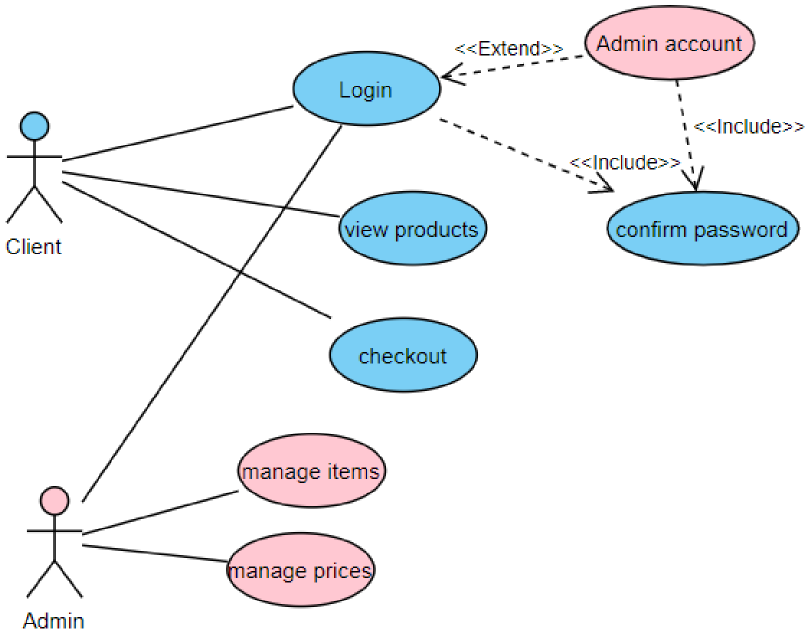


Use Case



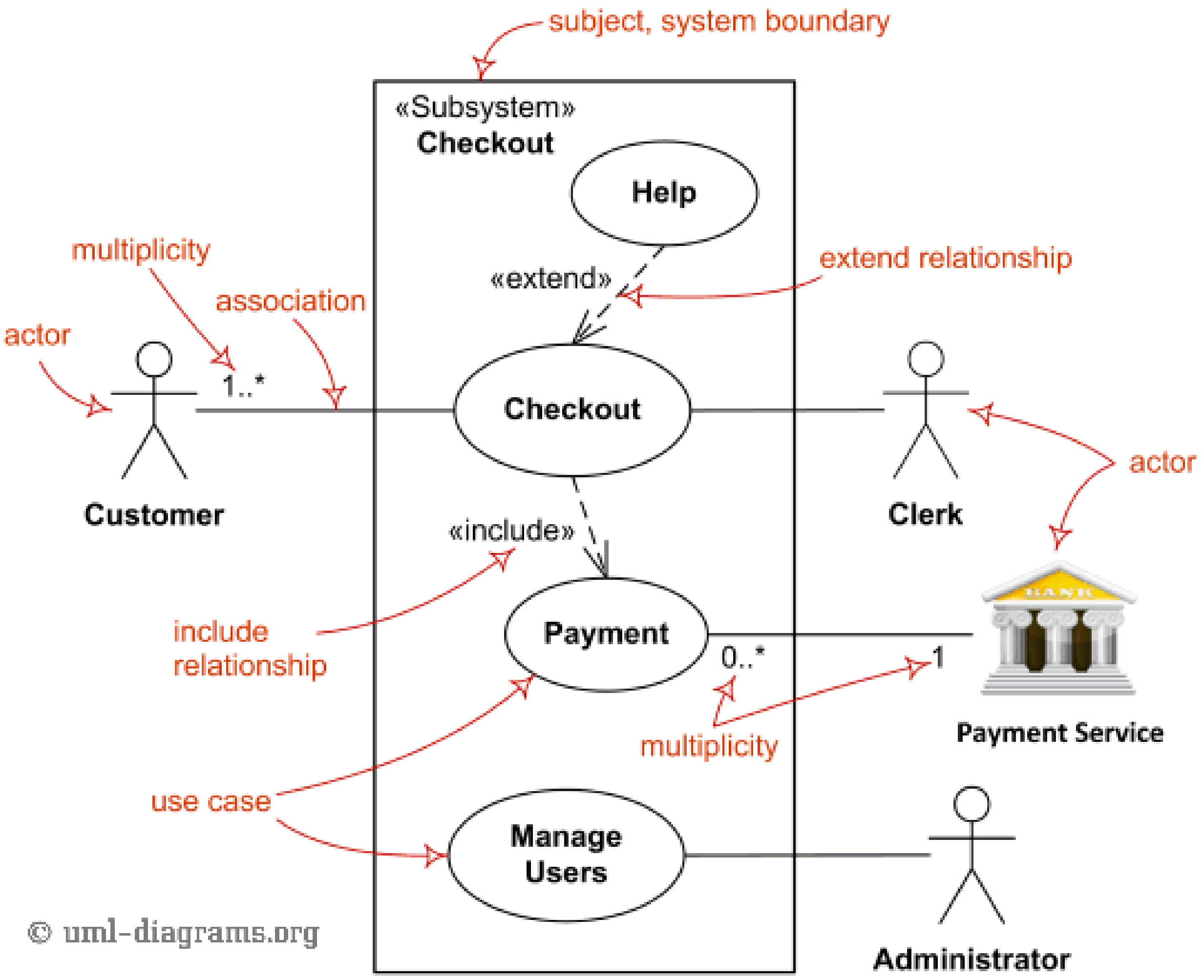
Use Cases Diagrams

- A use case diagram is a **graphical depiction of a user's possible interactions with a system.**
- A use case diagram shows **various use cases and different types of users**
- The **system has and will often be accompanied by other types of diagrams** as well.
- The use cases are **represented by either circles or ellipses**.

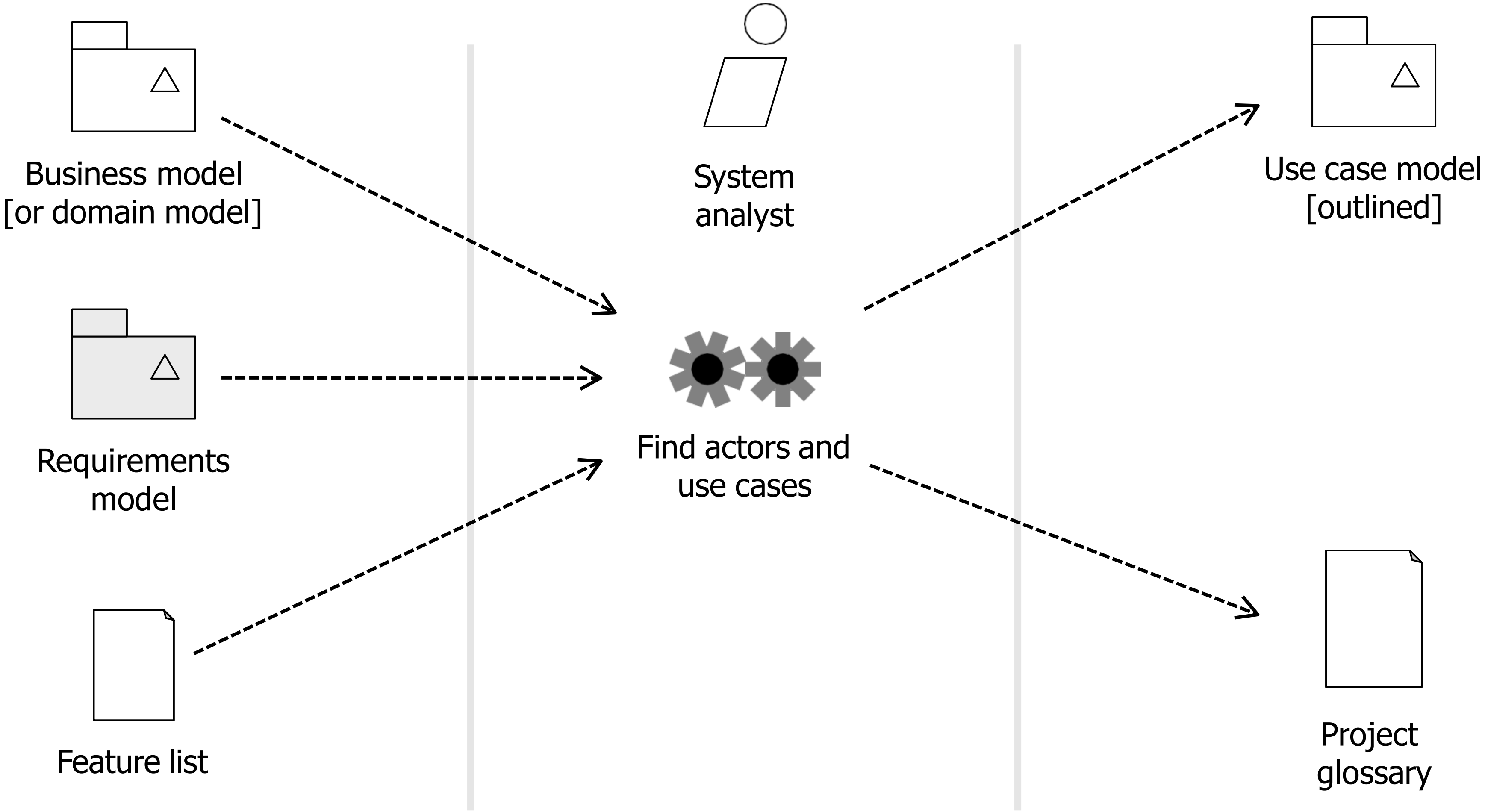


Use case modelling

- Use case modelling is a form of requirements engineering
- Use case modelling proceeds as follows:
  - Find the system boundary
  - Find actors
  - Find use cases
    - Use case specification
    - Scenarios
- It lets us identify the system boundary, who or what uses the system, and what functions the system should offer



Find actors and use cases


Business model
[or domain model]

Requirements
model

Feature list

System
analyst

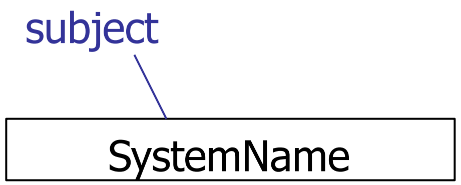
Find actors and
use cases

Use case model
[outlined]

Project
glossary

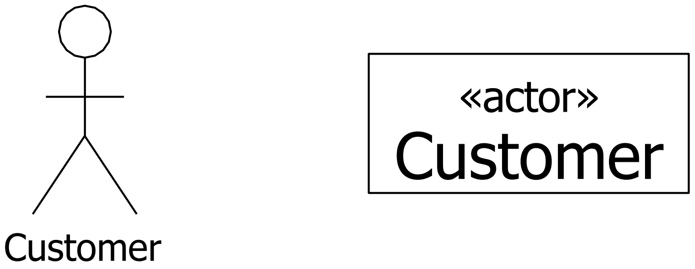
The subject

- Before we can build anything, we need to know:
  - Where the boundary of the system lies
  - Who or what uses the system
  - What functions the system should offer to its users
- We create a Use Case model containing:
  - Subject – the edge of the system
    - also known as the system boundary
  - Actors – who or what uses the system
  - Use Cases – things actors do with the system
  - Relationships - between actors and use cases



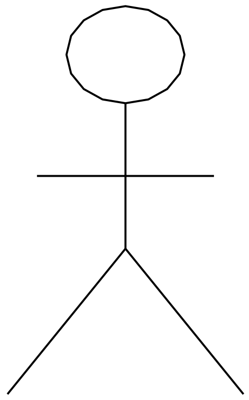
What are actors?

- An actor is anything that interacts *directly* with the system
  - Actors identify who or what uses the system and so indicate where the system boundary lies
- Actors are *external* to the system
- An Actor specifies a *role* that some external entity adopts when interacting with the system



Identifying Actors

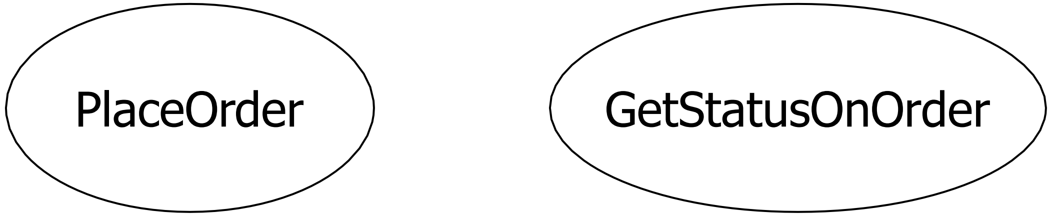
- When identifying actors ask:
- Who or what uses the system?
  - What roles do they play in the interaction?
  - Who installs the system?
  - Who starts and shuts down the system?
  - Who maintains the system?
  - What other systems use this system?
  - Who gets and provides information to the system?
  - Does anything happen at a fixed time?



Time

What are use cases?

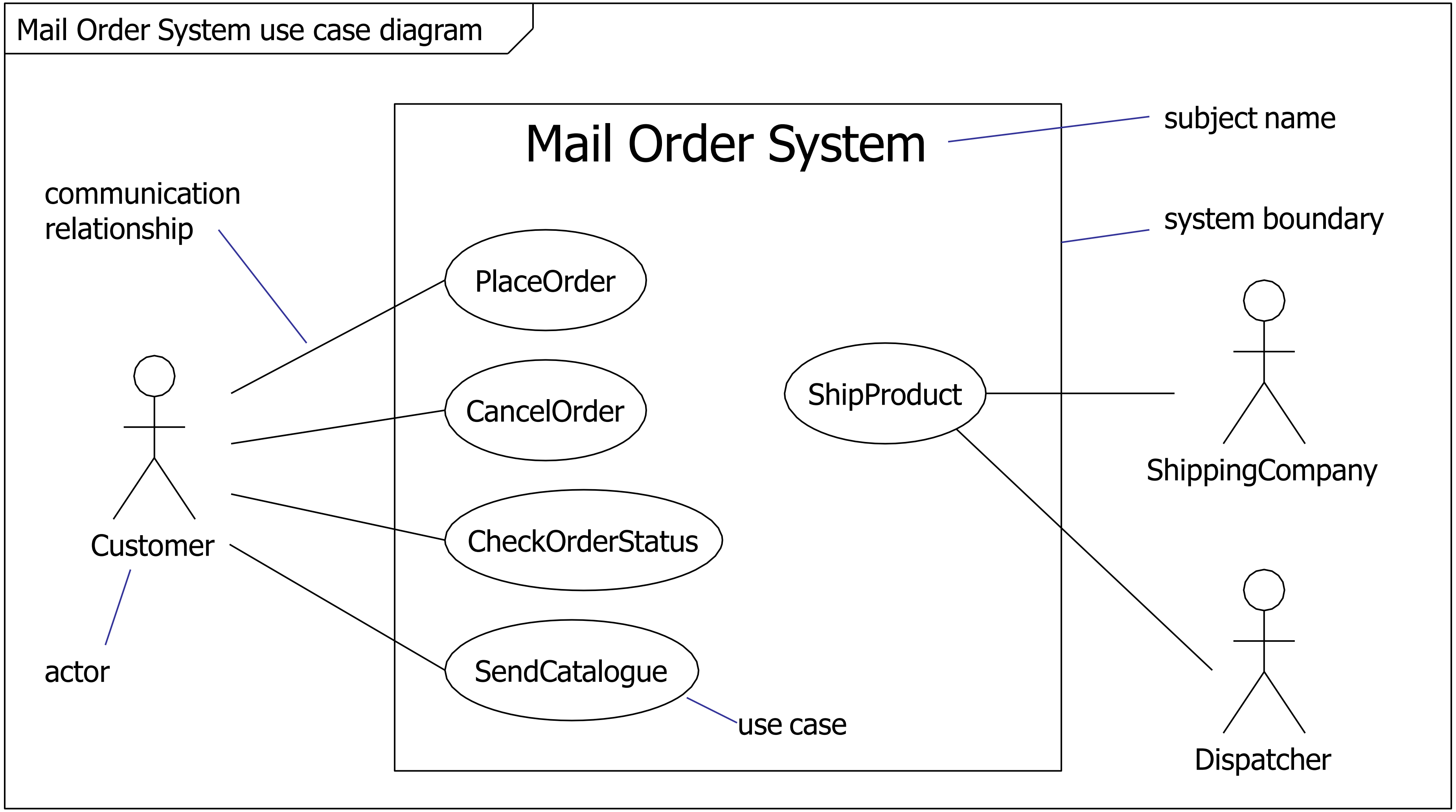
- A use case is **something an actor needs the system to do**. It is a “case of use” of the system by a specific actor
- Use cases are *always* started by an actor
  - The *primary actor* triggers the use case
  - Zero or more *secondary actors* interact with the use case in some way
- Use cases are *always* written from the point of view of the actors



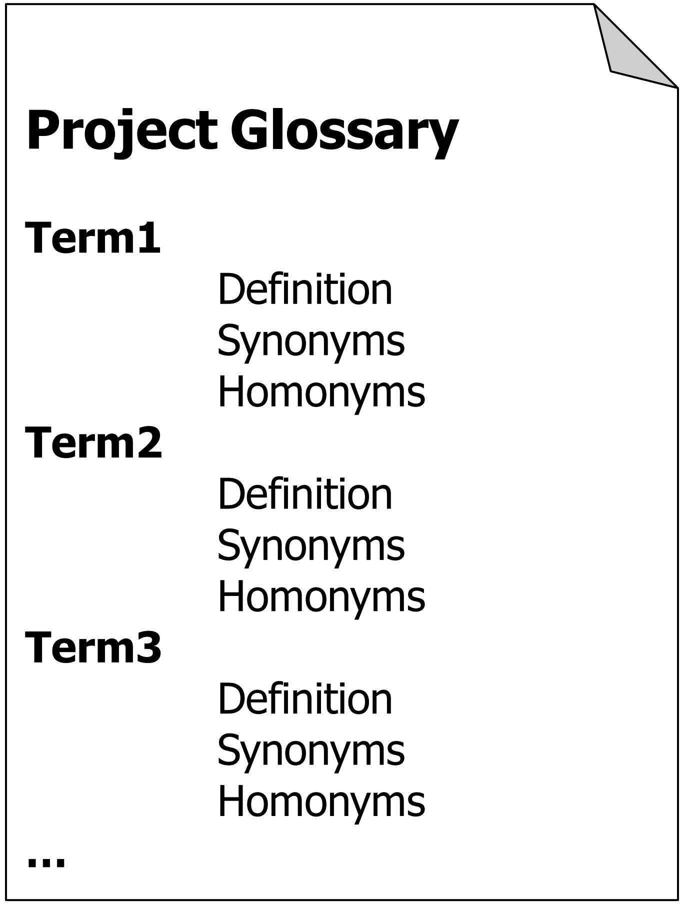
Identifying use cases

- Start with the list of actors that interact with the system
- When identifying use cases ask:
  - What functions will a specific actor want from the system?
  - Does the system store and retrieve information? If so, which actors trigger this behaviour?
  - What happens when the system changes state (e.g. system start and stop)? Are any actors notified?
  - Are there any external events that affect the system? What notifies the system about those events?
  - Does the system interact with any external system?
  - Does the system generate any reports?

The use case diagram



The Project Glossary



Project Glossary

Term1

Definition
Synonyms
Homonyms

Term2

Definition
Synonyms
Homonyms

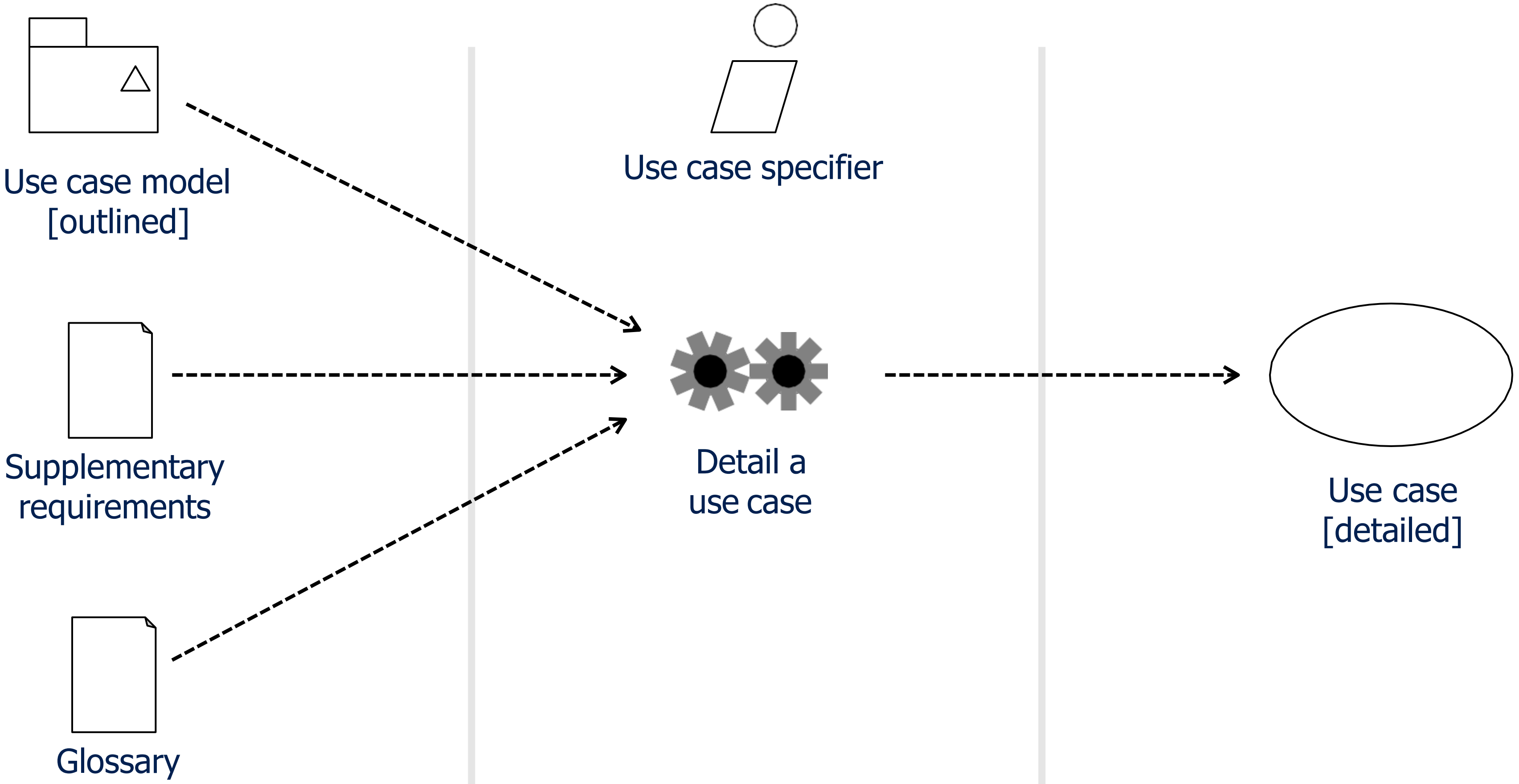
Term3

Definition
Synonyms
Homonyms

…

- In any business domain there is always a certain amount of jargon. It's important to capture the language of the domain in a project glossary
- The aim of the glossary is to define key terms and to resolve synonyms and homonyms
- You are building a vocabulary that you can use to discuss the system with the stakeholders

Detail a use case



Use case specification

use case name	Use case: PaySalesTax
use case identifier	ID: 1
brief description	Brief description: Pay Sales Tax to the Tax Authority at the end of the business quarter.
the actors involved in the use case	Primary actors: Time
	Secondary actors: TaxAuthority
the system state before the use case can begin	Preconditions: 1. It is the end of the business quarter.
the actual steps of the use case	Main flow: implicit time actor 1. The use case starts when it is the end of the business quarter. 2. The system determines the amount of Sales Tax owed to the Tax Authority. 3. The system sends an electronic payment to the Tax Authority.
the system state when the use case has finished	Postconditions: 1. The Tax Authority receives the correct amount of Sales Tax.
alternative flows	Alternative flows: None.

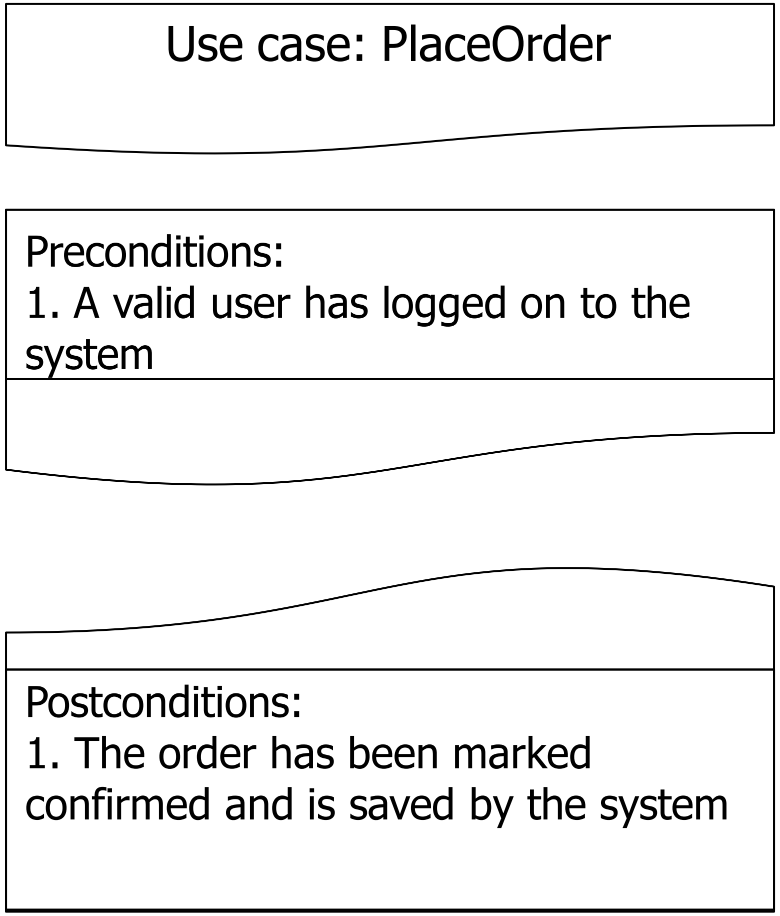
Naming use cases

- Use cases describe something that happens
- They are named using verbs or verb phrases
- Naming standard [1]: use cases are named using UpperCamelCase e.g. PaySalesTax

[1] UML 2 does not specify *any* naming standards.
All naming standards are our own, based on industry best practice.

Pre and postconditions

- Preconditions and postconditions are *constraints*
- Preconditions constrain the state of the system *before* the use case can start
- Postconditions constrain the state of the system *after* the use case has executed
- If there are no preconditions or postconditions write "None" under the heading



Use case: PlaceOrder

Preconditions:
1. A valid user has logged on to the system

Postconditions:
1. The order has been marked confirmed and is saved by the system

Main flow

<number> The <something> <some action>

- The flow of events lists the steps in a use case
- It *always* begins by an actor doing something
  - A good way to start a flow of events is:
    1)The use case starts when an <actor> <function>
- The flow of events should be a sequence of short steps that are:
  - Declarative
  - Numbered,
  - Time ordered
- The main flow is always the *happy day* or *perfect world* scenario
  - Everything goes as expected and desired, and there are no errors, deviations, interrupts, or branches
  - Alternatives can be shown by branching or by listing under
  - Alternative flows

Branching within a flow: If

- Use the keyword if to indicate alternatives within the flow of events
  - There must be a Boolean expression immediately after if
- Use indentation and numbering to indicate the conditional part of the flow
- Use else to indicate what happens if the condition is false (see next slide)

Use case: ManageBasket
ID: 2
Brief description: The Customer changes the quantity of an item in the basket.
Primary actors: Customer
Secondary actors: None.
Preconditions: 1. The shopping basket contents are visible.
Main flow: 1. The use case starts when the Customer selects an item in the basket. 2. If the Customer selects "delete item"   1. The system removes the item from the basket. 3. If the Customer types in a new quantity   1. The system updates the quantity of the item in the basket.
Postconditions: None.
Alternative flows: None.

Repetition within a flow: For

- We can use the keyword For to indicate the start of a repetition within the flow of events
- The iteration expression immediately after the For statement indicates the number of repetitions of the indented text beneath the For statement.

Use case: FindProduct
ID: 3
Brief description: The system finds some products based on Customer search criteria and displays them to the Customer.
Actors: Customer
Preconditions: None.
Main flow: 1. The use case starts when the Customer selects "find product". 2. The system asks the Customer for search criteria. 3. The Customer enters the requested criteria. 4. The system searches for products that match the Customer's criteria. 5. For each product found 1. The system displays a thumbnail sketch of the product. 2. The system displays a summary of the product details. 3. The system displays the product price.
Postconditions: None.
Alternative flows: NoProductsFound

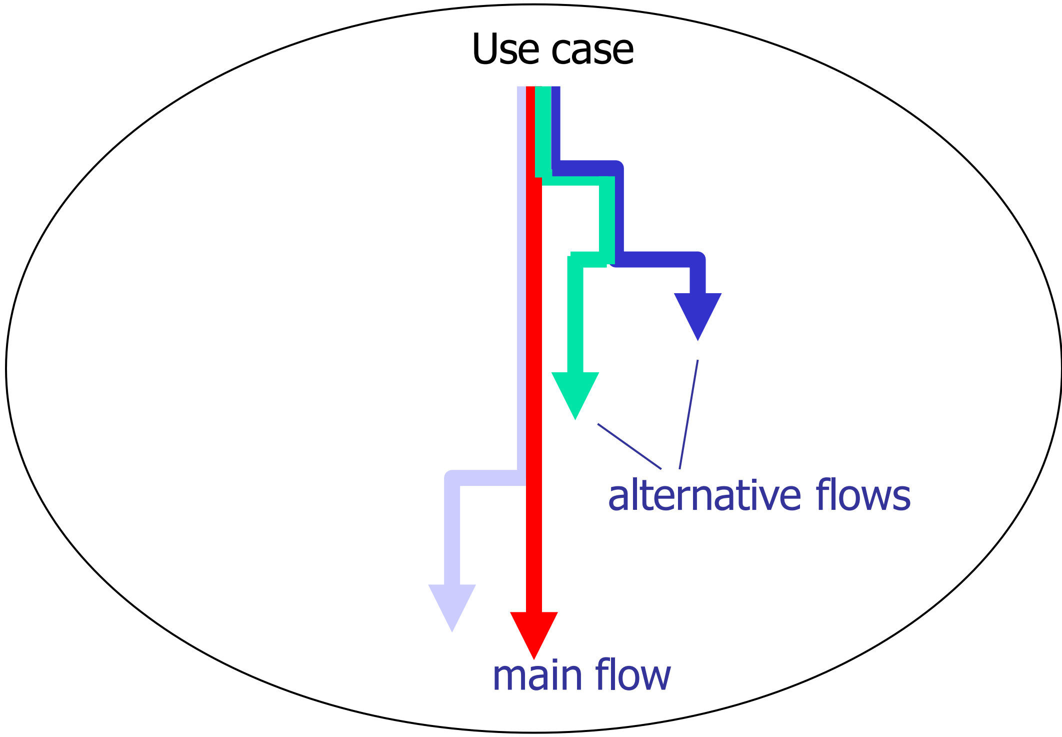
Repetition within a flow: While

- We can use the keyword while to indicate that something repeats while some Boolean condition is true

Use case: ShowCompanyDetails
ID: 4
Brief description: The system displays the company details to the Customer.
Primary actors: Customer
Secondary actors: None
Preconditions: None.
Main flow: 1. The use case starts when the Customer selects "show company details". 2. The system displays a web page showing the company details. 3. While the Customer is browsing the company details 4. The system searches for products that match the Customer's criteria.   1. The system plays some background music.   2. The system displays special offers in a banner ad.
Postconditions: 1. The system has displayed the company details. 2. The system has played some background music. 3. The systems has displayed special offers.
Alternative flows: None.

Branching: Alternative flows

- We may specify one or more alternative flows through the flow of events:
  - Alternative flows capture errors, branches, and interrupts
  - Alternative flows never return to the main flow
- Potentially very many alternative flows! You need to manage this:
  - Pick the most important alternative flows and document those.
  - If there are groups of similar alternative flows - document one member of the group as an exemplar and (if necessary) add notes to this explaining how the others differ from it.



Only document enough alternative flows to clarify the requirements!

Referencing alternative flows

- List the names of the alternative flows at the end of the use case
- Find alternative flows by examining each step in the main flow and looking for:
  - Alternatives
  - Exceptions
  - Interrupts

Use case: CreateNewCustomerAccount
ID: 5
Brief description: The system creates a new account for the Customer.
Primary actors: Customer
Secondary actors: None.
Preconditions: None.
Main flow: 1. The use case begins when the Customer selects "create new customer account". 2. While the Customer details are invalid 1. The system asks the Customer to enter his or her details comprising email address, password and password again for confirmation. 2.2 The system validates the Customer details. 3. The system creates a new account for the Customer.
Postconditions: 1. A new account has been created for the Customer.
Alternative flows: InvalidEmailAddress InvalidPassword Cancel

alternative flows

An alternative flow example

notice how we name and number alternative flows

Alternative flow: CreateNewCustomerAccount:InvalidEmailAddress
ID: 5.1
Brief description: The system informs the Customer that they have entered an invalid email address.
Primary actors: Customer
Secondary actors: None.
Preconditions: 1. The Customer has entered an invalid email address
Alternative flow: 1. The alternative flow begins after step 2.2. of the main flow. 2. The system informs the Customer that he or she entered an invalid email address.
Postconditions: None.

always indicate how the alternative flow begins. In this case it starts after step 2.2 in the main flow

- The alternative flow may be triggered *instead* of the main flow - started by an actor
- The alternative flow may be triggered *after a particular step* in the main flow - after
- The alternative flow may be triggered *at any time* during the main flow - at any time

Requirements tracing

- Given that we can capture functional requirements in a requirements model *and* in a use case model we need some way of relating the two
- There is a many-to-many relationship between requirements and use cases:
  - One use case covers many individual functional requirements
  - One functional requirement may be realised by many use cases
- Hopefully we have CASE support for requirements tracing:
  - With UML tagged values, we can assign numbered requirements to use cases
  - We can capture use case names in our Requirements Database
- If there is no CASE support, we can create a Requirements Traceability matrix

		Use cases			
		U1	U2	U3	U4
Requirements	R1	🟥			
	R2		🟥	🟥	
	R3			🟥	
	R4				🟥
	R5	🟥			

Requirements Traceability Matrix

When to use case analysis

- Use cases describe system behaviour from the point of view of one or more actors. They are the *best* choice when:
  - The system is dominated by functional requirements
  - The system has many types of user to which it delivers different functionality
  - The system has many interfaces
- Use cases are designed to capture *functional* requirements. They are a *poor* choice when:
  - The system is dominated by non-functional requirements
  - The system has few users
  - The system has few interfaces

Summary

- We have seen how to capture functional requirements with use cases
- We have looked at:
  - Use cases
  - Actors
  - Branching with if
  - Repetition with for and while
  - Alternative flows
  - Requirements tracing

Requirements – advanced use case modelling

Aim and Objective is to produce evidence of an extra layer of design details into our draft UML Use Cases from the previous section:

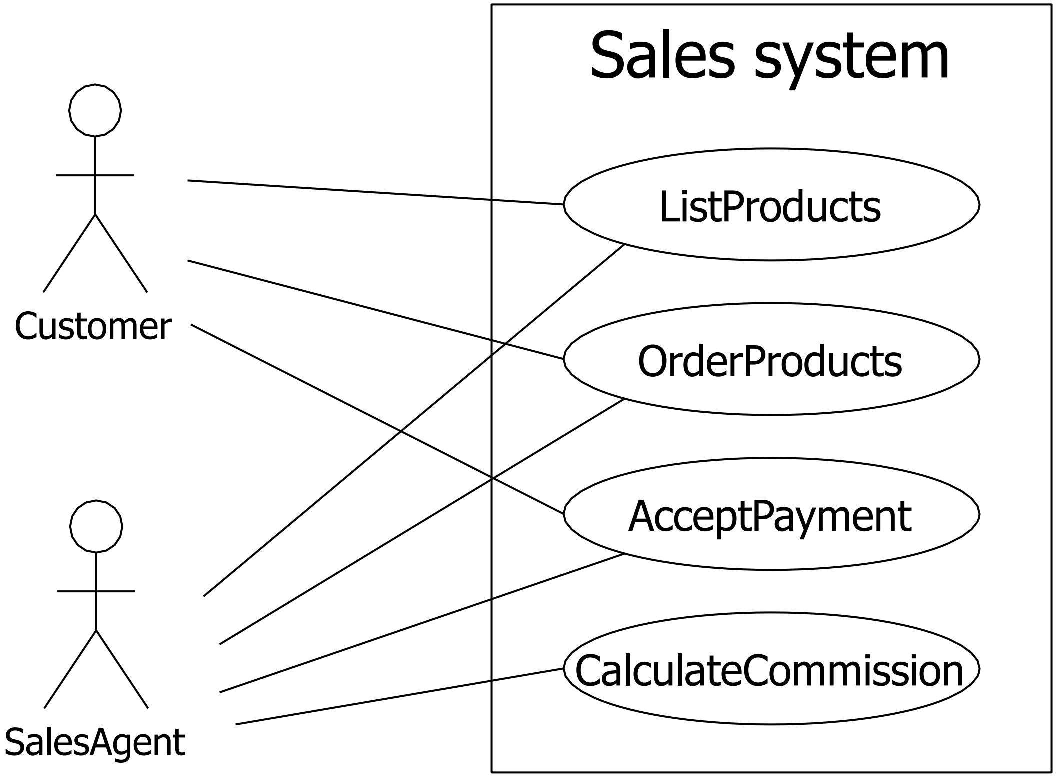
- UML Use Case Diagrams

More relationships…

- We have studied basic use case analysis, but there are relationships that we have still to explore:
  - Actor generalization
  - Use case generalization
  - «include» – between use cases
  - «extend» – between use cases

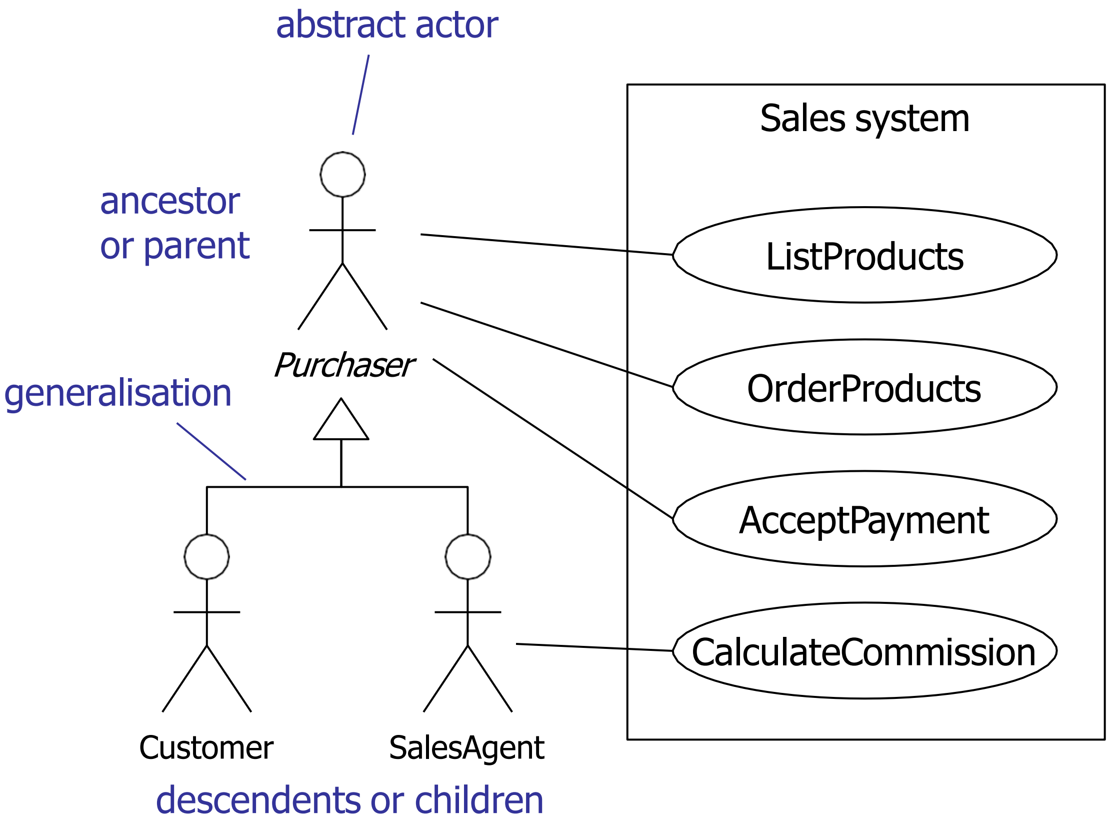
Actor generalization - example

- The Customer and the Sales Agent actors are *very* similar
- They both interact with List products, Order products, Accept payment
- Additionally, the Sales Agent interacts with Calculate commission
- Our diagram is a *mess* – can we simplify it?



Actor generalization

- If two actors communicate with the same set of use cases in the same way, then we can express this as a generalisation to another (possibly abstract) actor
- The descendent actors inherit the roles and relationships to use cases held by the ancestor actor
- We can substitute a descendent actor anywhere the ancestor actor is expected. This is the *substitutability principle*

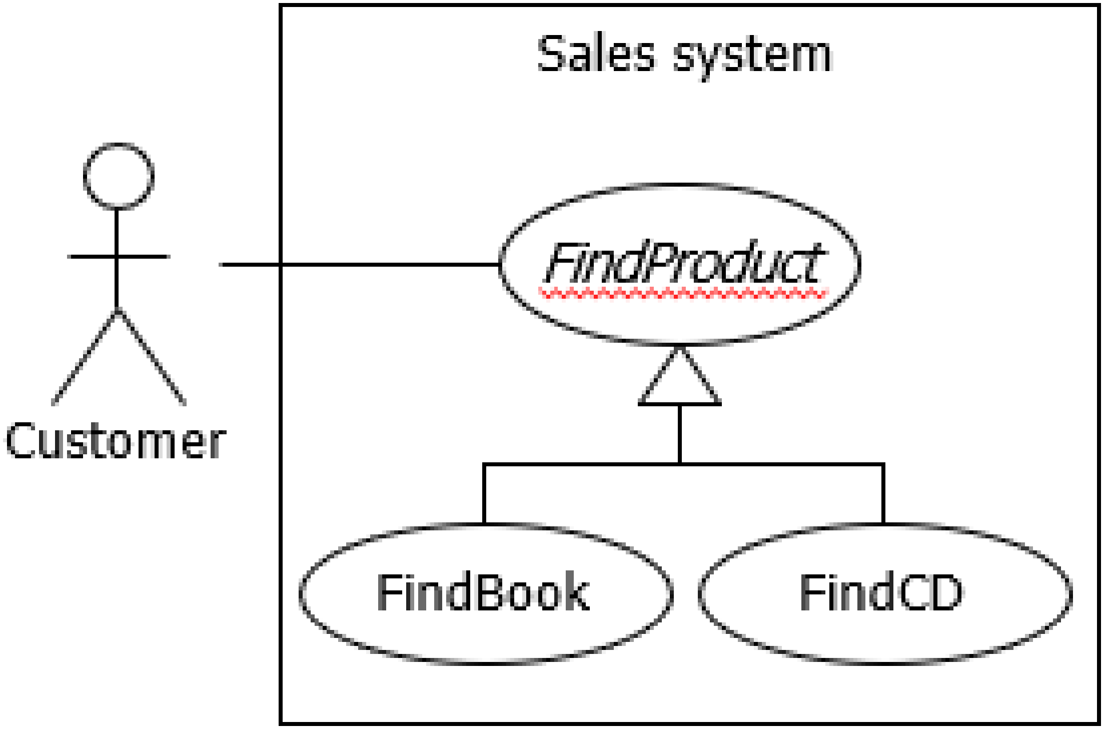


Use actor generalization when it simplifies the model

Use case generalisation

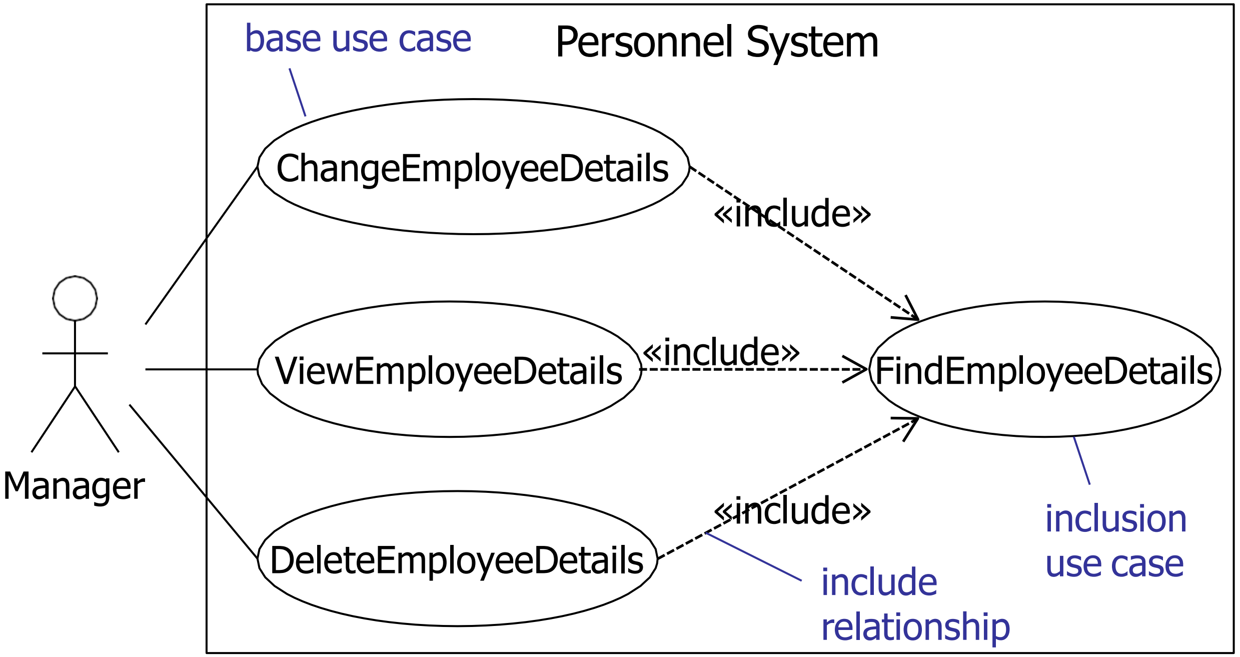
- The ancestor use case must be a more general case of one or more descendant use cases
- Child use cases are more specific forms of their parent
- They can inherit, add and override features of their parent

Use case generalization semantics			
Use case element	Inherit	Add	Override
Relationship	Yes	Yes	No
Extension point	Yes	Yes	No
Precondition	Yes	Yes	Yes
Postcondition	Yes	Yes	Yes
Step in main flow	Yes	Yes	Yes
Alternative flow	Yes	Yes	Yes



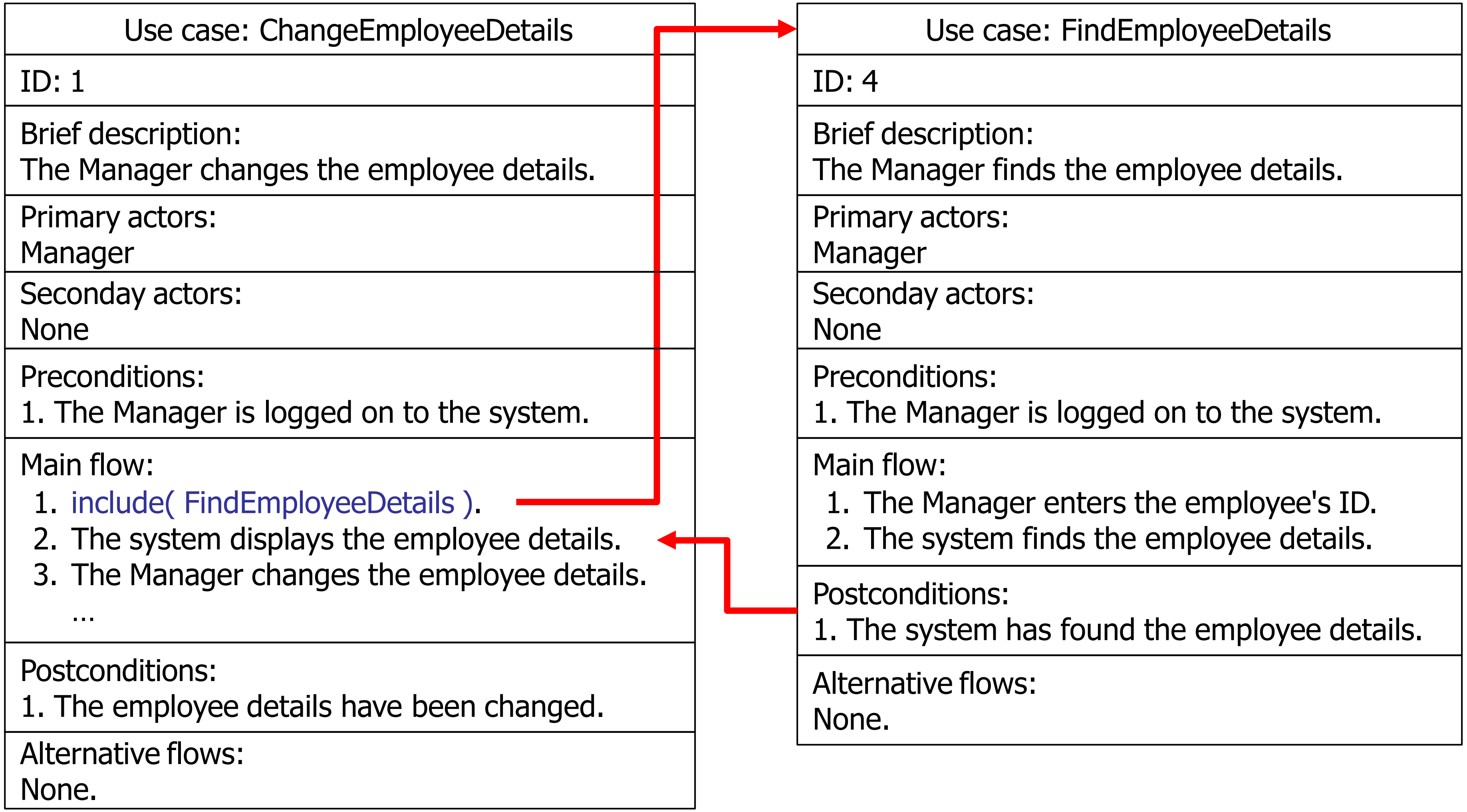
«include»

- The base use case executes until the point of inclusion: include(InclusionUseCase)
  - Control passes to the inclusion use case which executes
  - When the inclusion use case is finished, control passes back to the base use case which finishes execution
- Note:
  - Base use cases are *not complete* without the included use cases
  - Inclusion use cases may be complete use cases, or they may just specify a fragment of behaviour for inclusion elsewhere



When use cases share common behaviour we can factor this out into a separate inclusion use case and «include» it in base use cases

«include» example

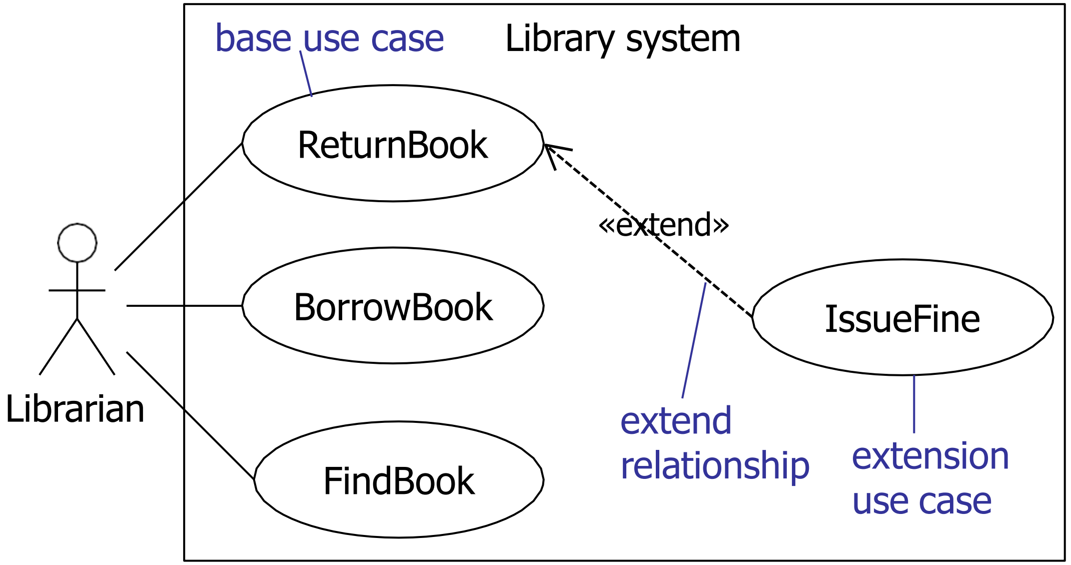


Use case: ChangeEmployeeDetails
ID: 1
Brief description: The Manager changes the employee details.
Primary actors: Manager
Seconday actors: None
Preconditions: 1. The Manager is logged on to the system.
Main flow: 1. include(FindEmployeeDetails). 2. The system displays the employee details. 3. The Manager changes the employee details. …
Postconditions: 1. The employee details have been changed.
Alternative flows: None.

Use case: FindEmployeeDetails
ID: 4
Brief description: The Manager finds the employee details.
Primary actors: Manager
Seconday actors: None
Preconditions: 1. The Manager is logged on to the system.
Main flow: 1. The Manager enters the employee's ID. 2. The system finds the employee details.
Postconditions: 1. The system has found the employee details.
Alternative flows: None.

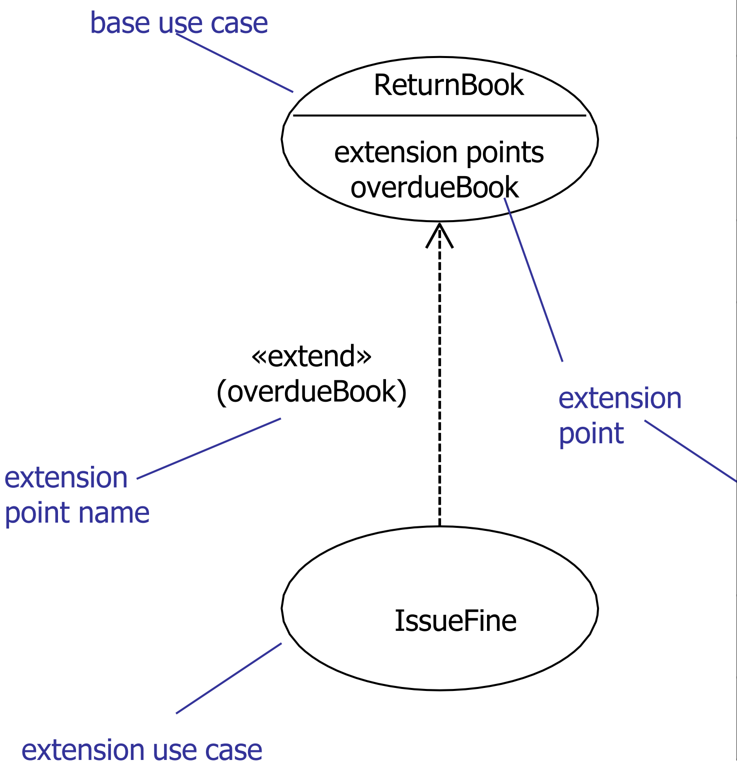
«extend»

- «extend» is a way of adding new behaviour into the base use case by inserting behaviour from one or more extension use cases
  - The base use case specifies one or more extension points in its flow of events
- The extension use case may contain several insertion segments
- The «extend» relationship may specify *which* of the base use case extension points it is extending



The extension use case inserts behaviour into the base use case.
The base use case provides extension points, but *does not know* about the extensions.

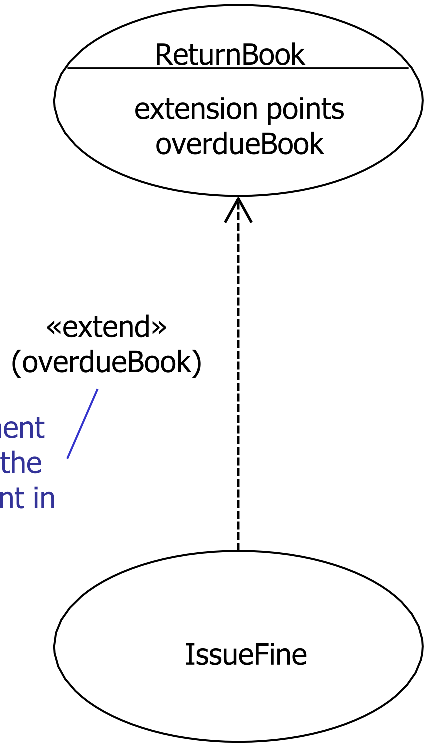
Base use case



Use case: ReturnBook
ID: 9
Brief description: The Librarian returns a borrowed book.
Primary actors: Librarian
Secondary actors: None.
Preconditions: 1. The Librarian is logged on to the system.
Main flow: 1. The Librarian enters the borrower's ID number. 2. The system displays the borrower's details including the list of borrowed books. 3.The Librarian finds the book to be returned in the list of books. extension point: overdueBook 4. The Librarian returns the book. …
Postconditions: 1. The book has been returned.
Alternative flows: None.

- There is an extension point overdueBook just before step 4 of the flow of events
- Extension points are *not* numbered, as they are *not* part of the flow

Extension use case

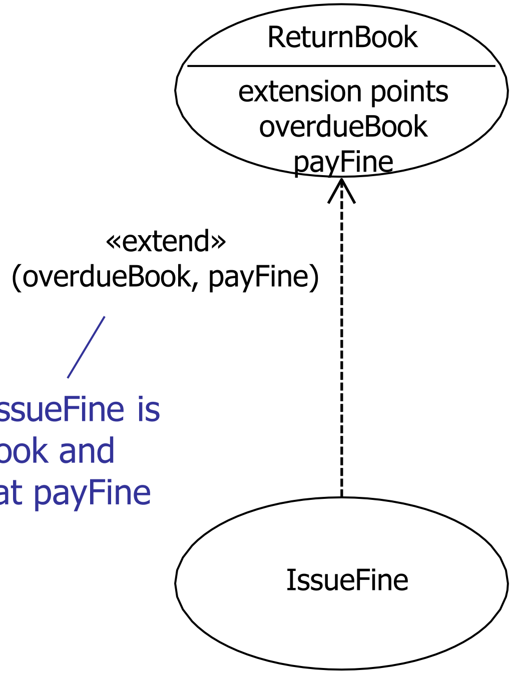


the single insertion segment in IssueFine is inserted at the overdueBook insertion point in the ReturnBook use case

Extension Use case: IssueFine
ID: 10
Brief description: Segment 1: The Librarian records and prints out a fine.
Primary actors: Librarian
Secondary actors: None.
Segment 1 preconditions: 1. The returned book is overdue.
Segment 1 flow: 1. The Librarian enters details of the fine into the system. 2. The system prints out the fine.
Segment 1 postconditions: 1. The fine has been recorded in the system. 2. The system has printed out the fine.

- Extension use cases have one or more *insertion segments* which are behaviour fragments that will be inserted at the specified extension points in the base use case

Multiple insertion points

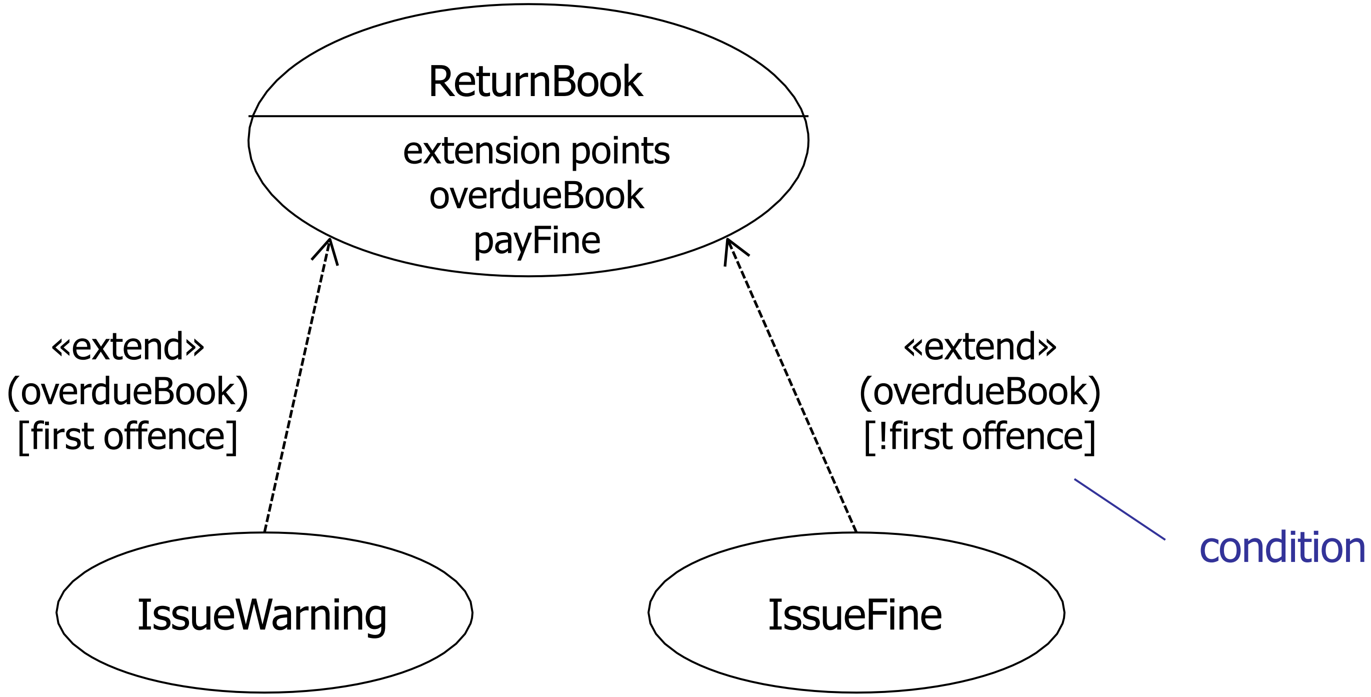


the first segment in IssueFine is inserted at overdueBook and the second segment at payFine

- If more than one extension point is specified in the «extend» relationship then the extension use case must have the *same number* of insertion segments

Extension Use case: IssueFine
ID: 10
Brief description: Segment 1: The Librarian records and prints out a fine. Segment 2: The Librarian accepts payment for a fine.
Primary actors: Librarian
Secondary actors: None.
Segment 1 preconditions: 1. The returned book is overdue.
Segment 1 flow: 1. The Librarian enters details of the fine into the system. 2. The system prints out the fine.
Segment 1 postconditions: 1. The fine has been recorded in the system. 2. The system has printed out the fine.
Segment 2 preconditions: 1. A fine is due from the borrower.
Segment 2 flow: 1. The Librarian accepts payment for the fine from the borrower. 2. The Librarian enters the paid fine in the system. 3. The system prints out a receipt for the paid fine.
Segment 2 postconditions: 1. The fine is recorded as paid. 2. The system has printed a receipt for the fine.

Conditional extensions



- We can specify conditions on «extend» relationships
  - Conditions are Boolean expressions
  - The insertion is made if and only if the condition evaluates to true

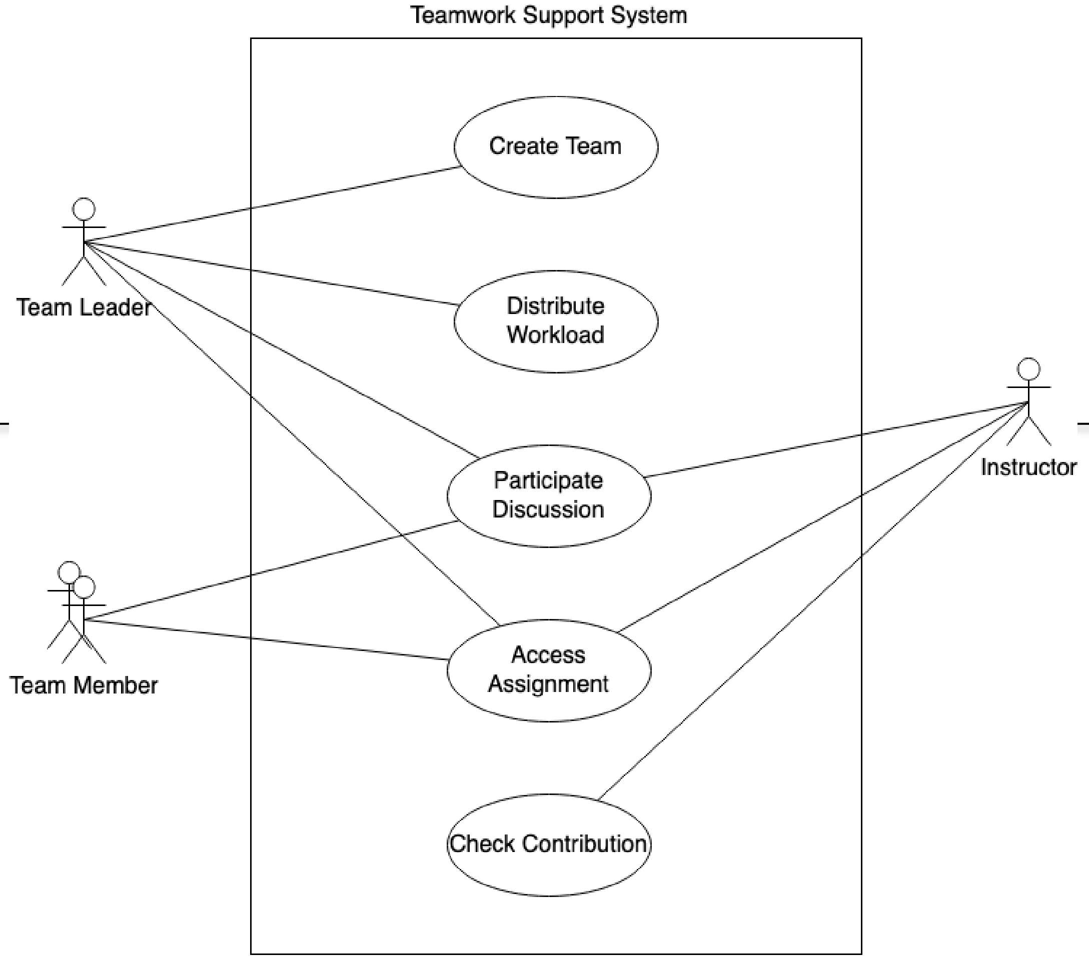
Summary

- We have learned about techniques for advanced use case modelling:
  - Actor generalization
  - Use case generalization
  - «include»
  - «extend»
- Use advanced features with discretion only where they simplify the model!

ID	Details	Priority
R1	Team leader shall be able to create a team.	MustHave
R2	Team leader shall distribute workload.	MustHave
R3	Everyone shall be able to participate discussion of the teamwork.	MustHave
R4	Everyone shall be able to access the work file, aka the Assignment.	MustHave
R5	Instructor shall be able to check students’ contribution scores.	MustHave

DEMO

Demonstration on Use Cases by using the Diagrams.net



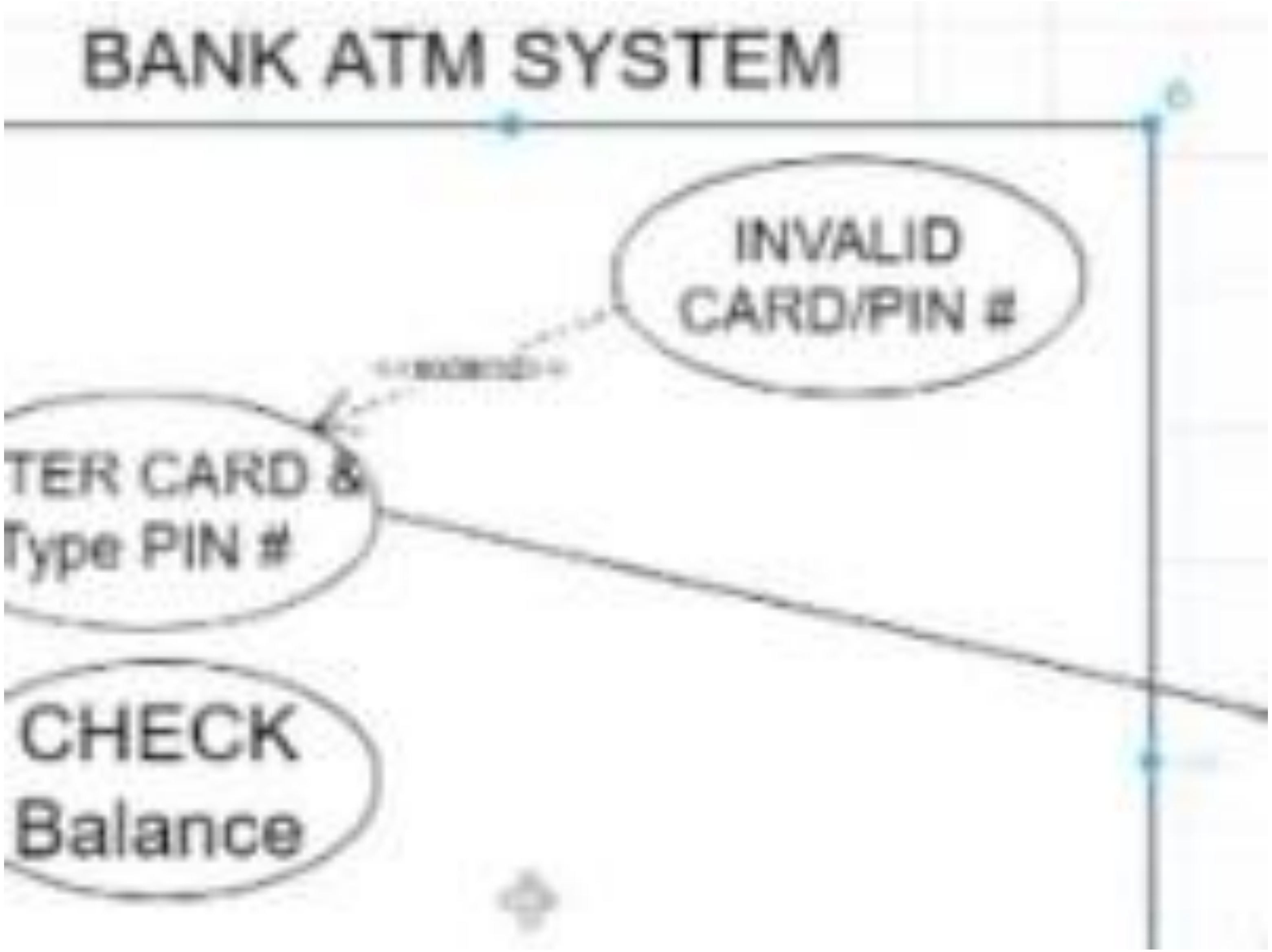
Supporting Material

- Highly recommended to walk through in your own time

Creating App Wireframes in Draw.io (Diagrams.net)

https://youtu.be/8eGaSmVFYhI

How to create USE CASE diagram using Draw.IO.



https://youtu.be/A1o4FNmLuw4

Draw.io (aka diagrams.net) Basics Tutorial with Example of Class Diagram, Sequence Diagram and Mockup/Wireframe

https://youtu.be/WlCKv49Pkvg

Use Case Diagrams Tutorial for Business Analysts

https://youtu.be/0TJ4GoSFbhs

How to create an Interactive Prototype in Draw.io (Diagrams.net)

https://youtu.be/CyA2bqQzKmE