# COMP2511

## Tute08

# Agenda

- Software Architecture 101
- Sequence Diagrams
- C4 Model
- Assignment II!!!

# Software Architecture?

Software Architecture vs Design Principles?

Consequences of Poor Architecture?

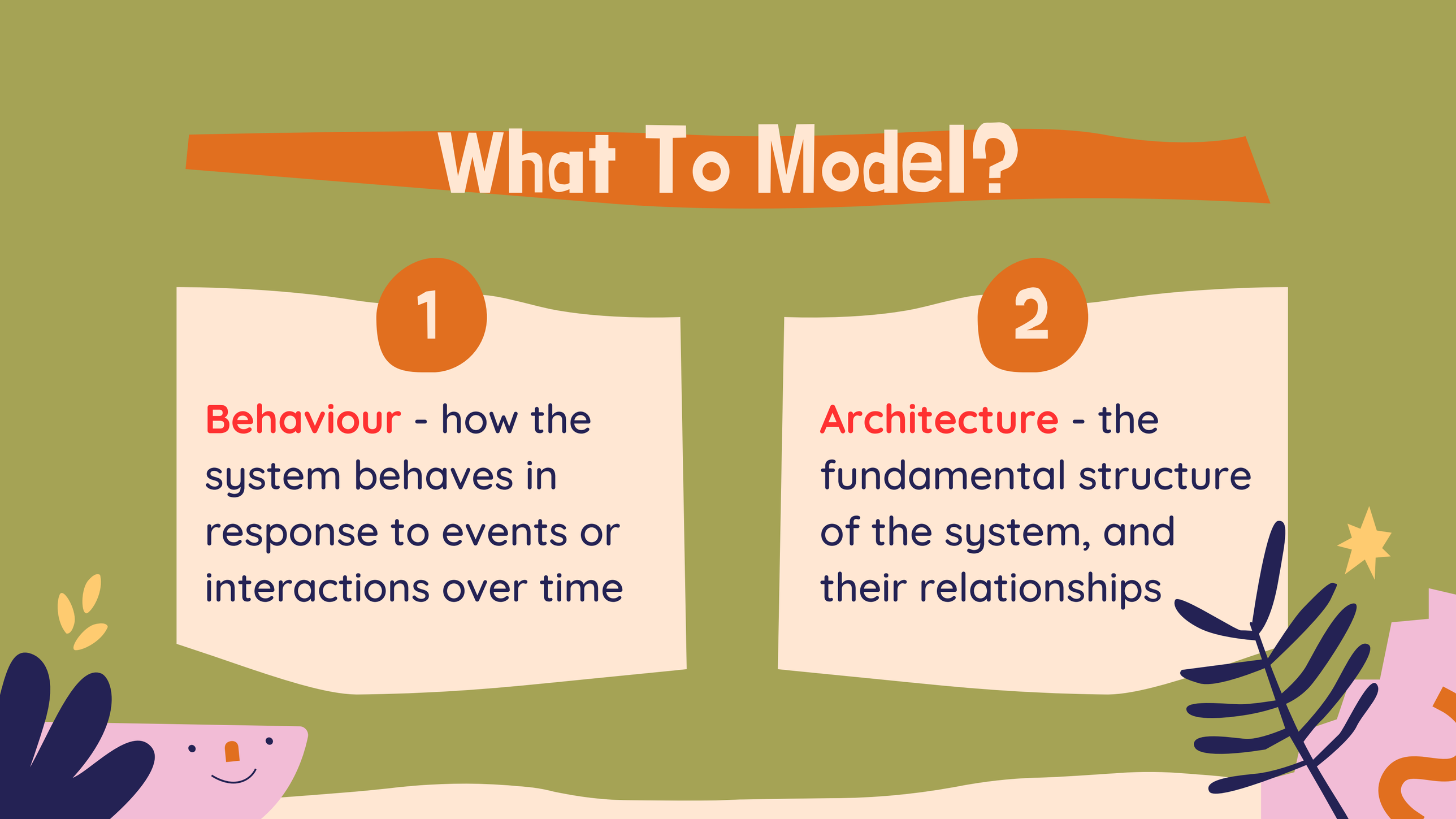
# What To Model?

1

**Behaviour** - how the system behaves in response to events or interactions over time

2

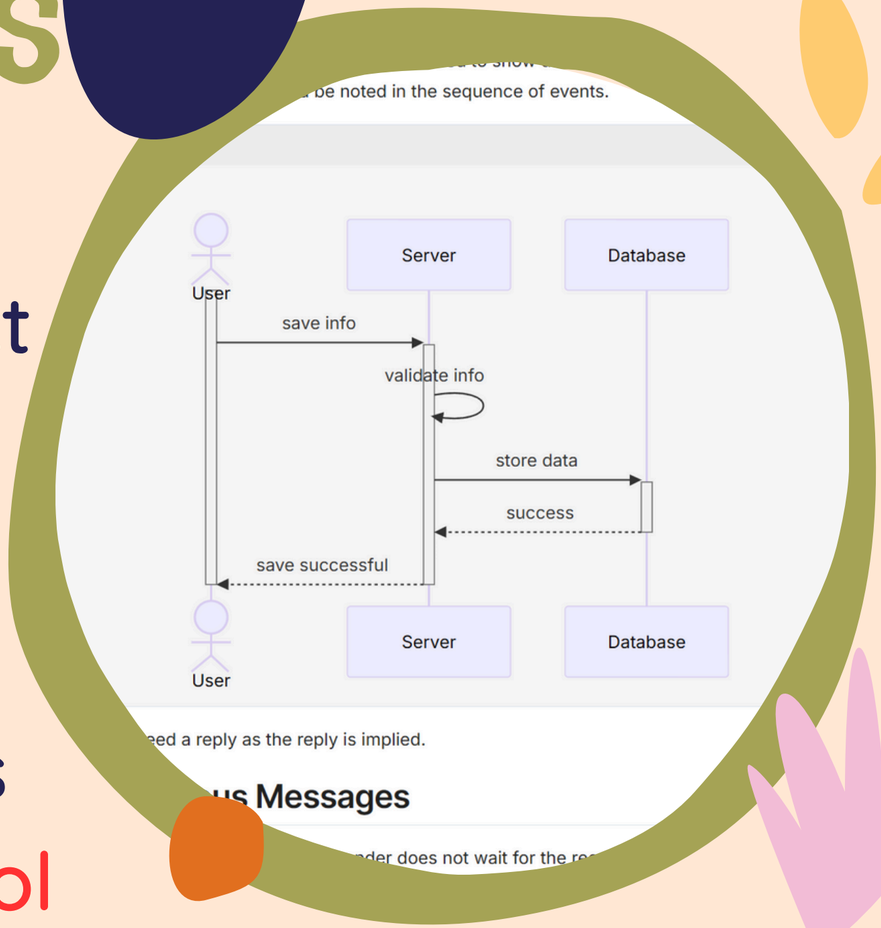
**Architecture** - the fundamental structure of the system, and their relationships



# Sequence Diagrams

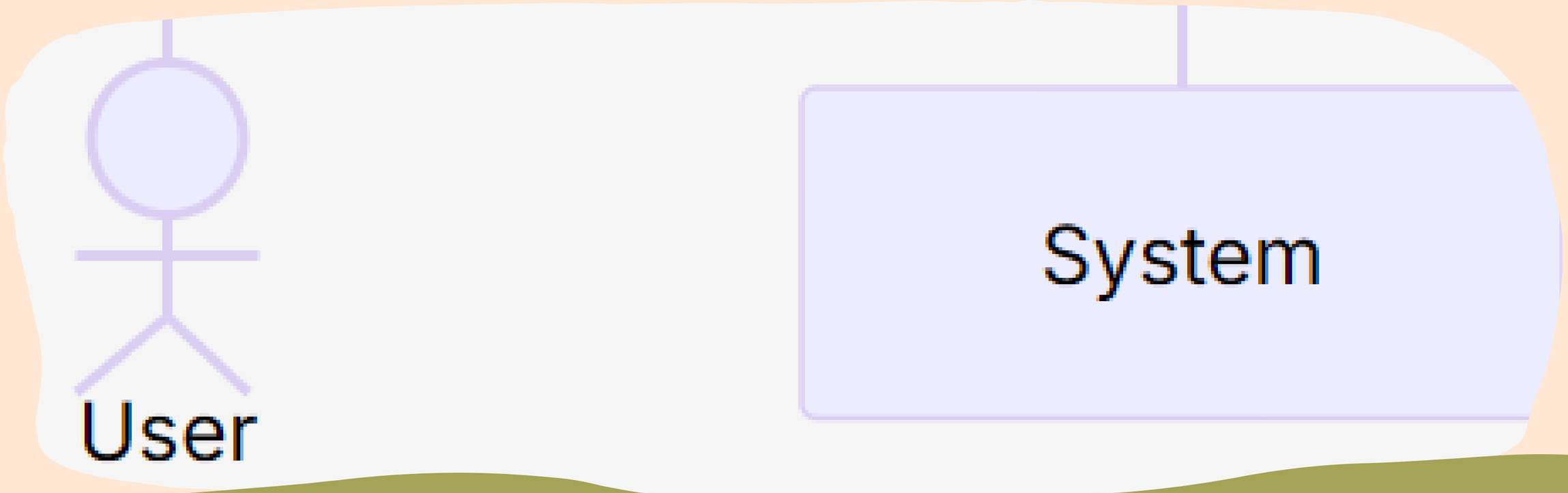
A sequence diagram is an interaction diagram showing how objects interact in a time sequenced manner.

- Used to model behaviour
- Illustrate the temporal order of interactions between components
- Visualisation for the flow of control and messages over time



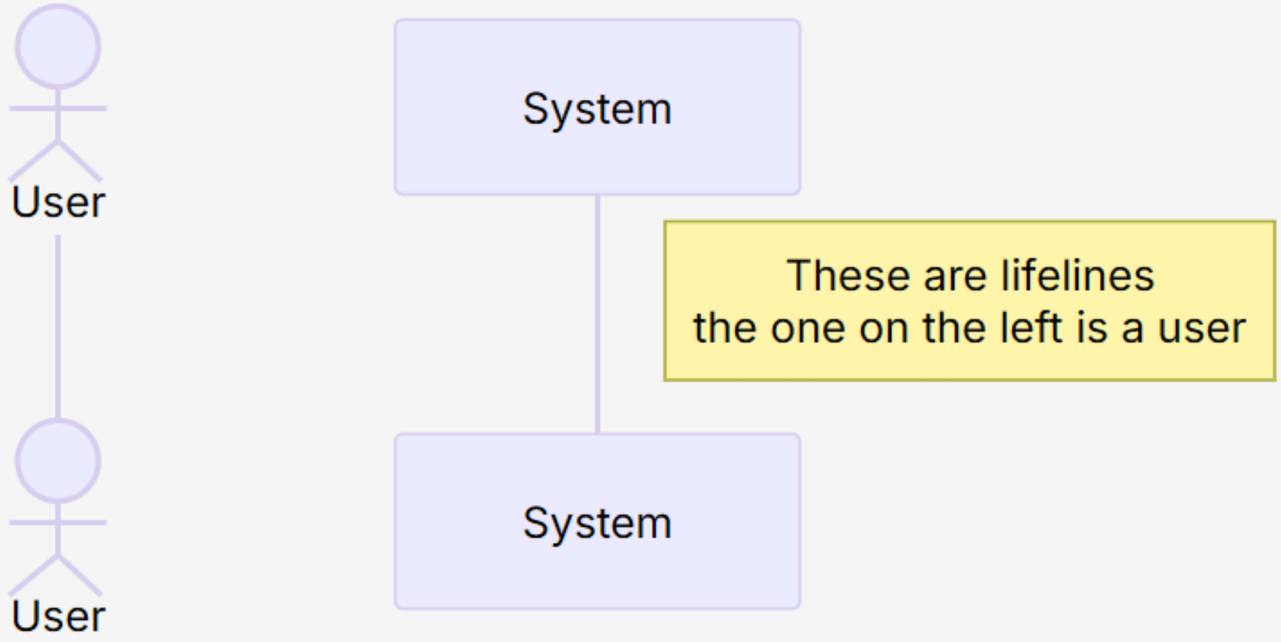
# Entities

- Users of the system
- External entities
- Containers within the system



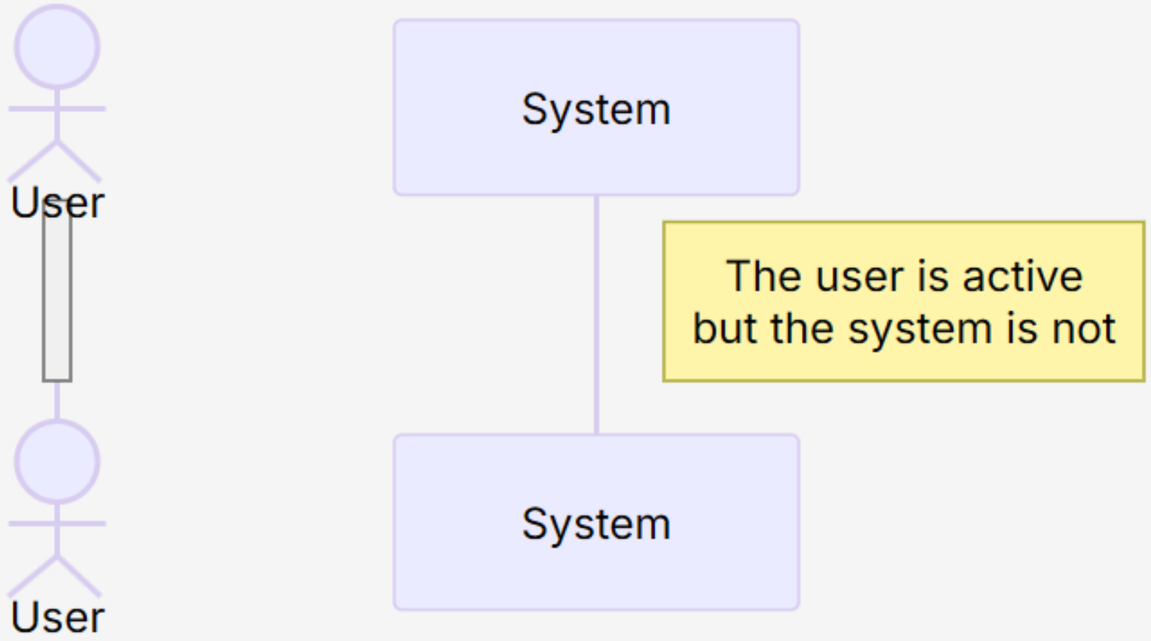
# Lifelines

- Vertical lines that represent the timeline of their attached entity
- One lifeline / entity



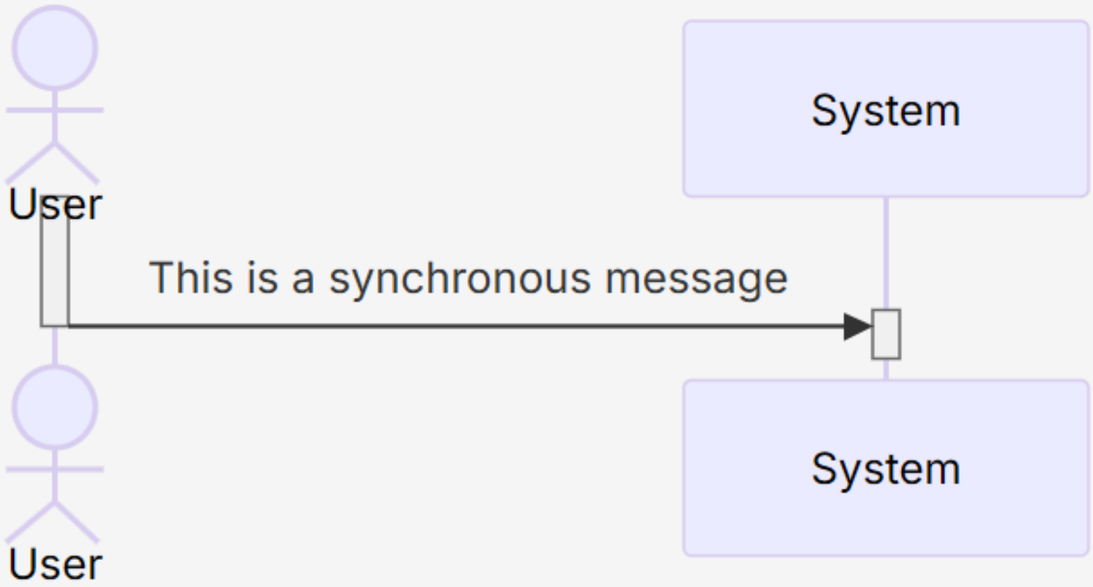
# Activation Boxes

- Rectangular boxes on lifelines
- Indicate when entity is active (sending/recieving/awaiting response)



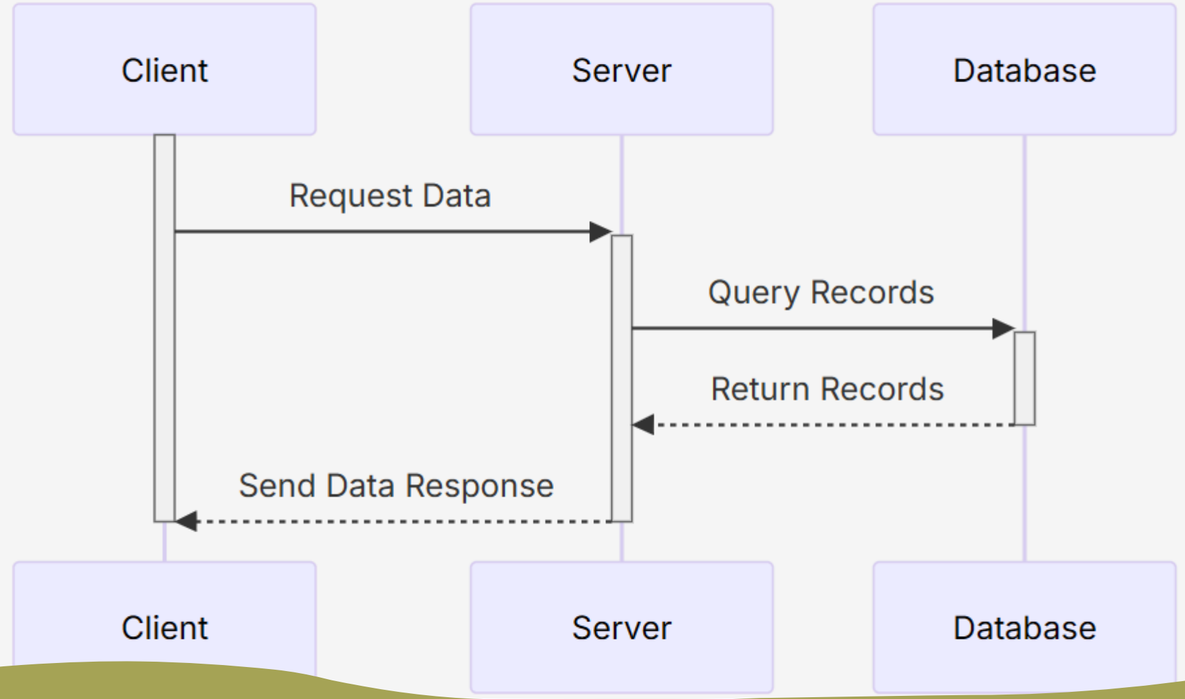
# Synchronous Messages

- Solid line with filled arrowhead
- One entity makes a request from another entity and waits for the response



# Synchronous Replies

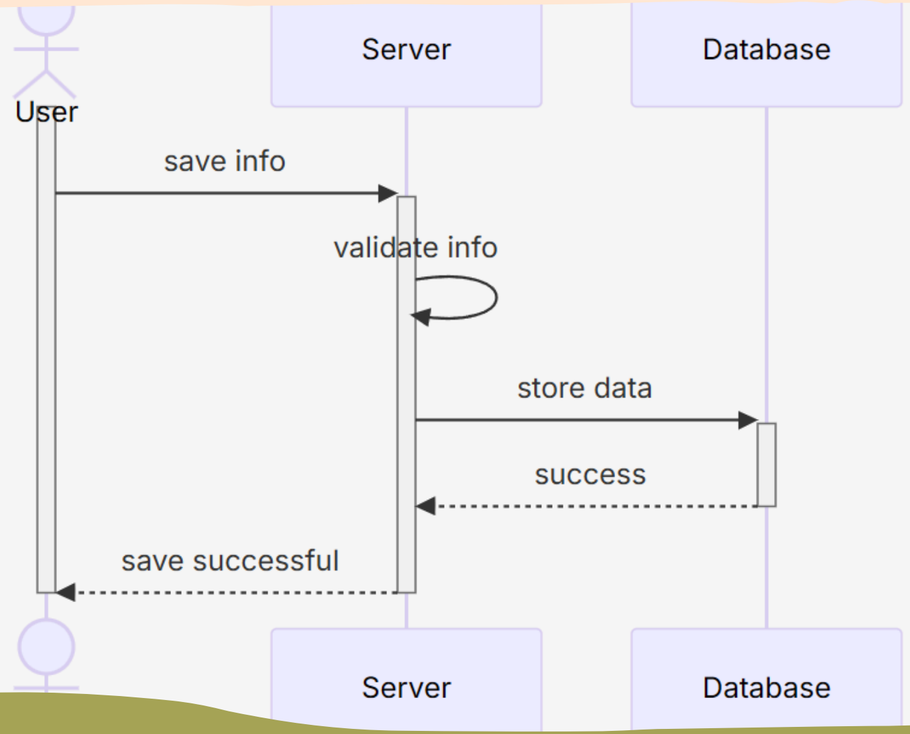
- Dashed line with filled arrowhead
- Response to a synchronous request, containing data or confirmation of completion




3.1.1.
3.1.2. Syn
Example
3.1.3. Self L
3.2. Asynchro
3.3. Message
Parameters
4. Guards
5. Combined
5.1. Option

# Self Loops

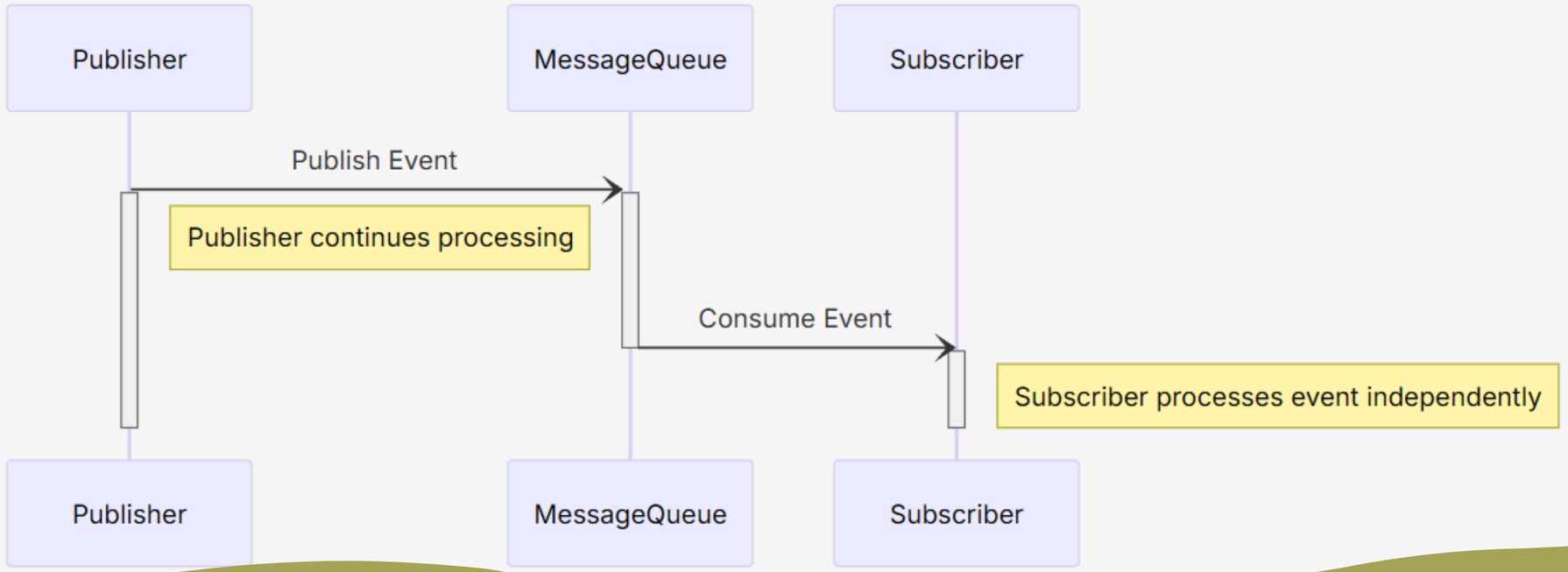
- Solid line with filled arrowhead looping back
- Entity sending a message to itself, usually to perform some additional processing




3. Messages
3.1. Synchronous Messag
3.1.1. Synchronous Replies
3.1.2. Synchronous Commun
Example
3.1.3. Self Loops
3.2. Asynchronous Messages
3.3. Message Labels and
Parameters
4. Guards
5. Combined Fragments
5.1. Option

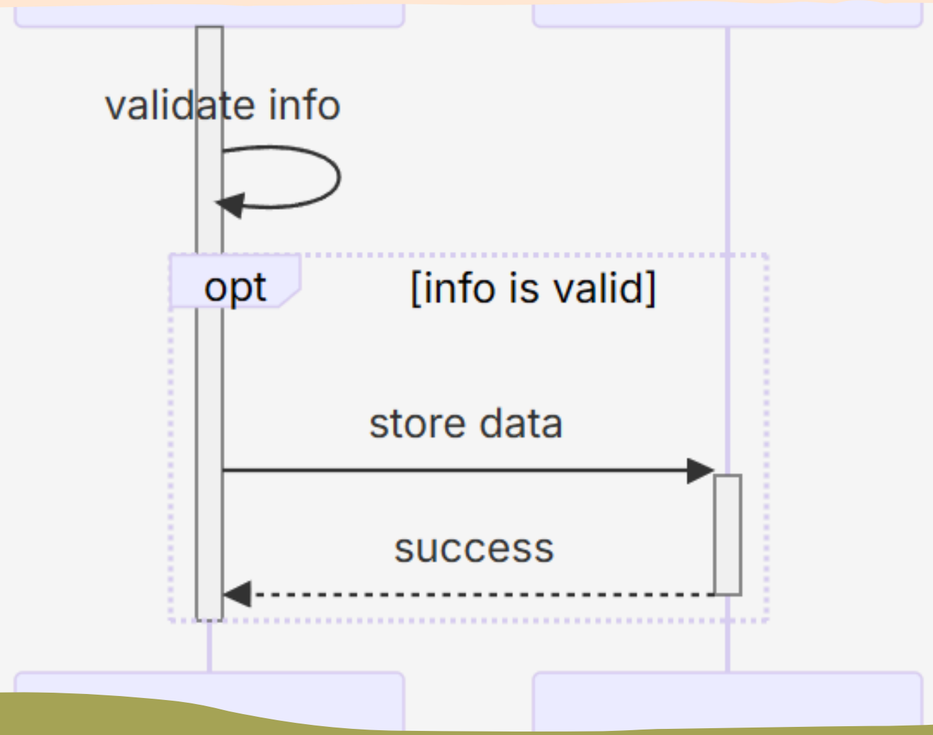
# Asynchronous Messages

- Solid line with open arrowhead
- One entity makes a request from another entity and continues executing



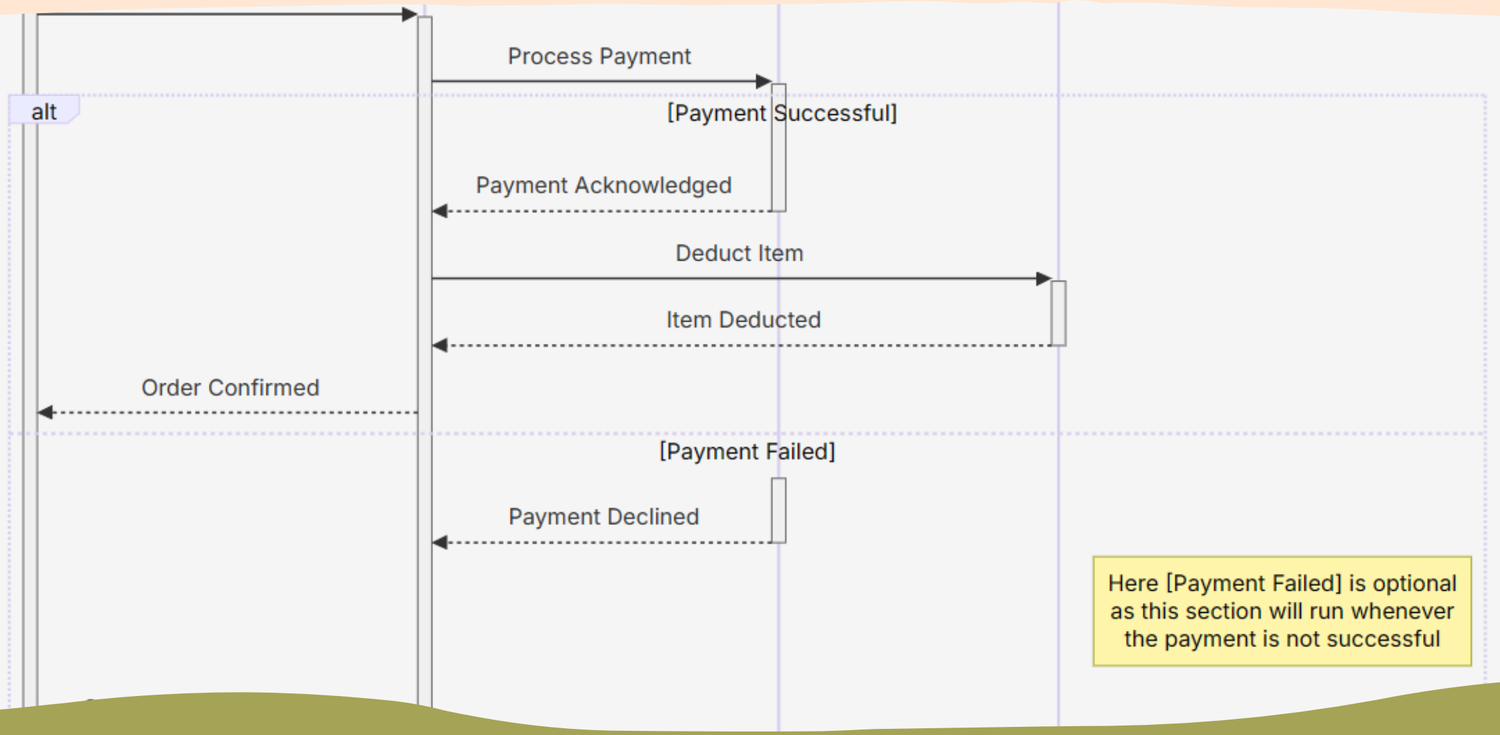
# Option

- Represented with “opt”
- Model a single optional flow
- Equivalent to “if [guard is true]” statement



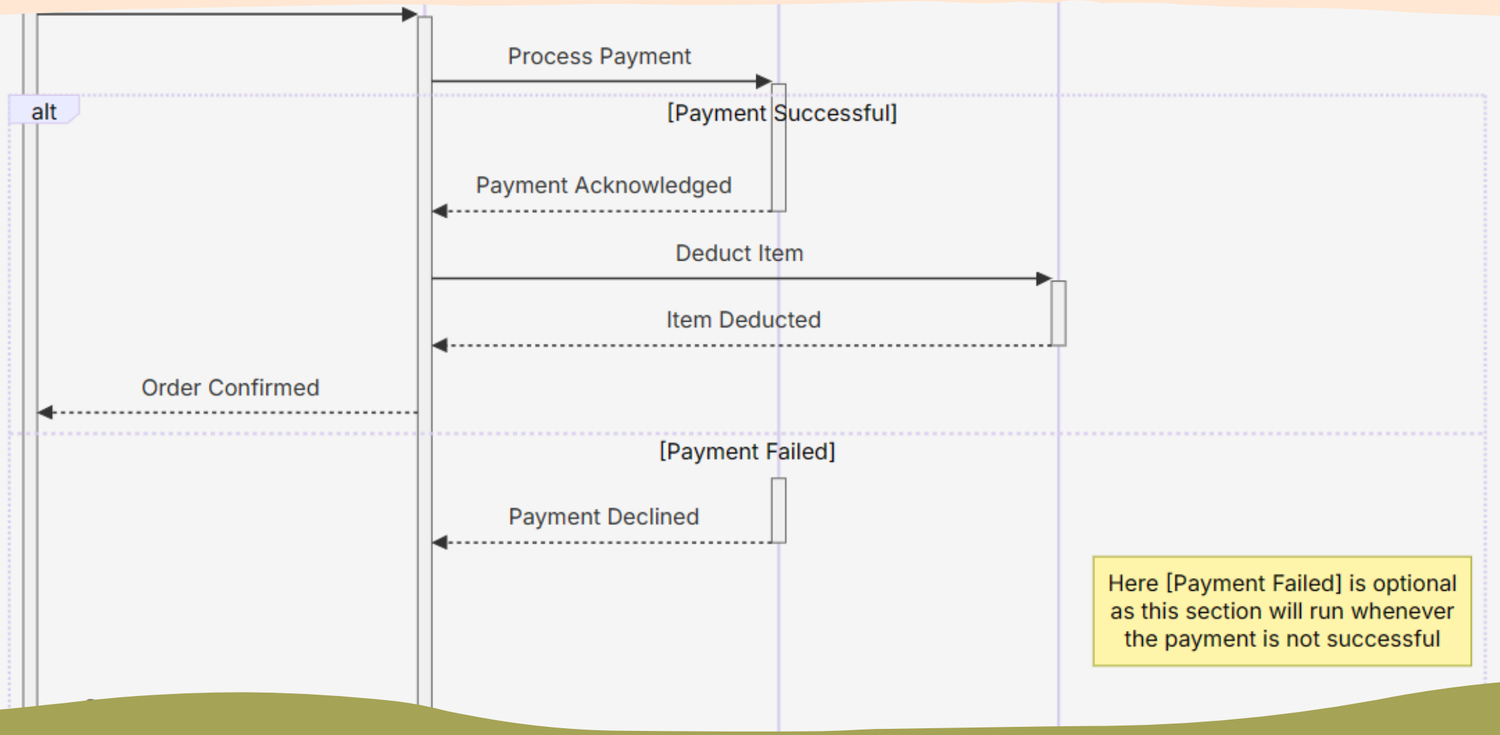
# Alternatives

- Represented with “alt”
- Model conditional statements with 2+ options
- Equivalent to “if….else if…else” statement



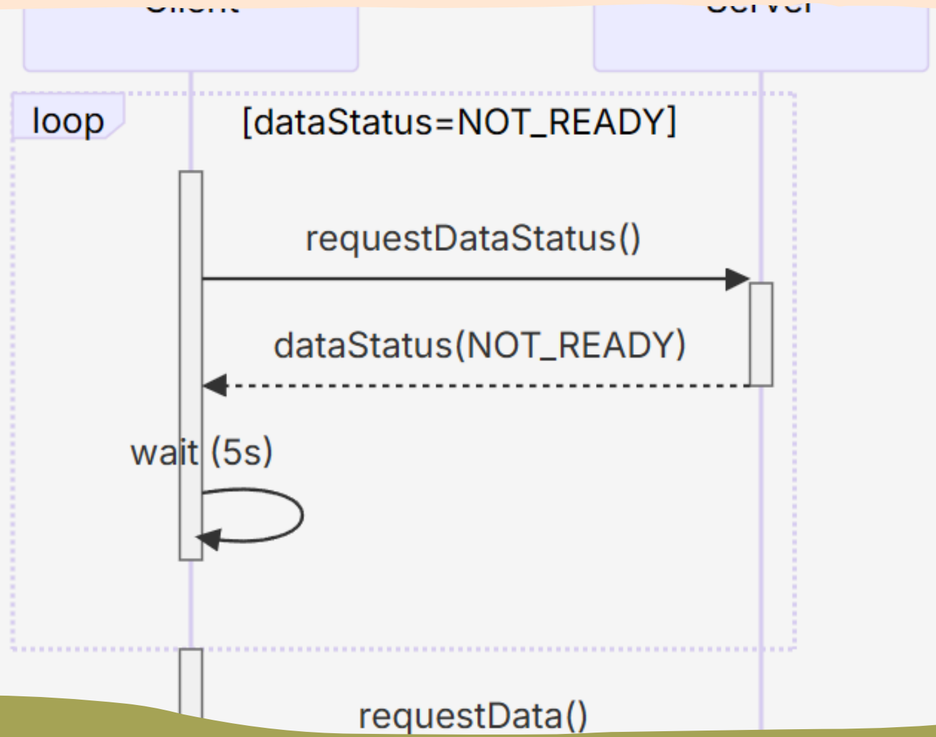
# Alternatives

- Represented with “alt”
- Model conditional statements with 2+ options
- Equivalent to “if….else if…else” statement



# Loop

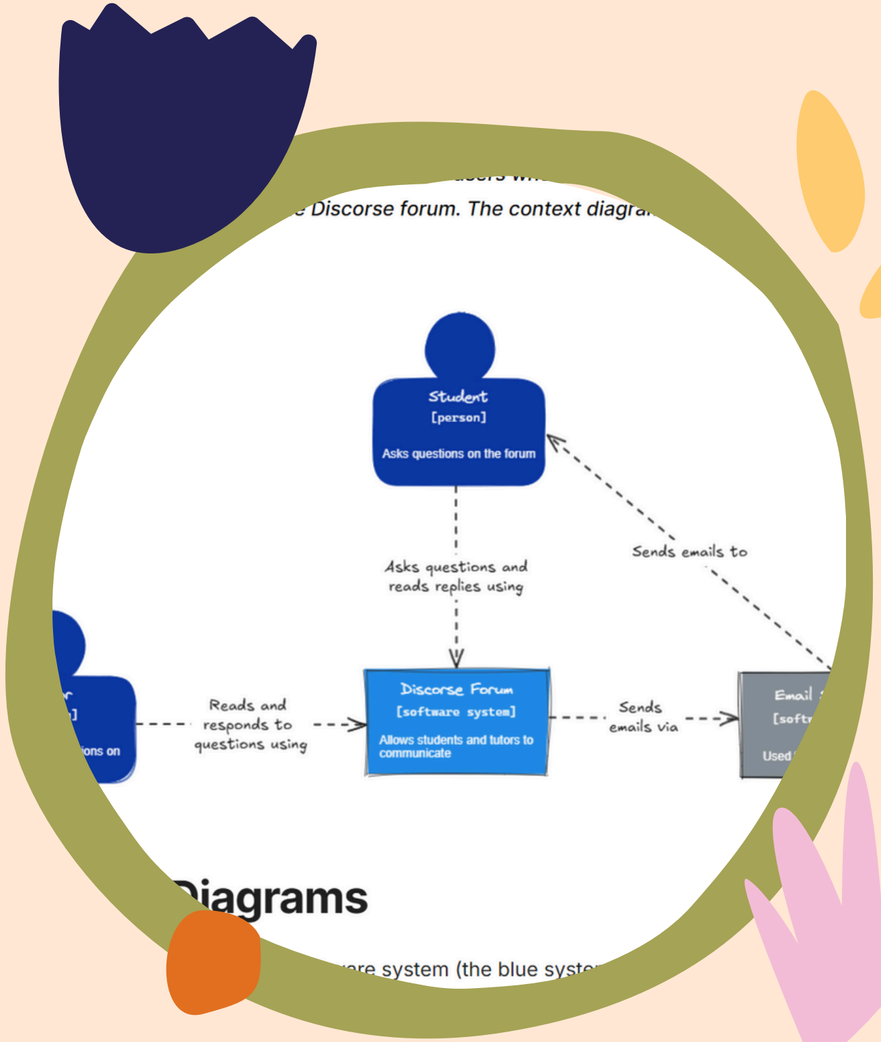
- Represented with “loop”
- Model repeated actions
- Equivalent to “while [guard is true]” statement



# C4 Model

The C4 Model focuses on the intuitive visual representation of the software architecture, defined by a set of hierarchical diagrams arranged by levels.

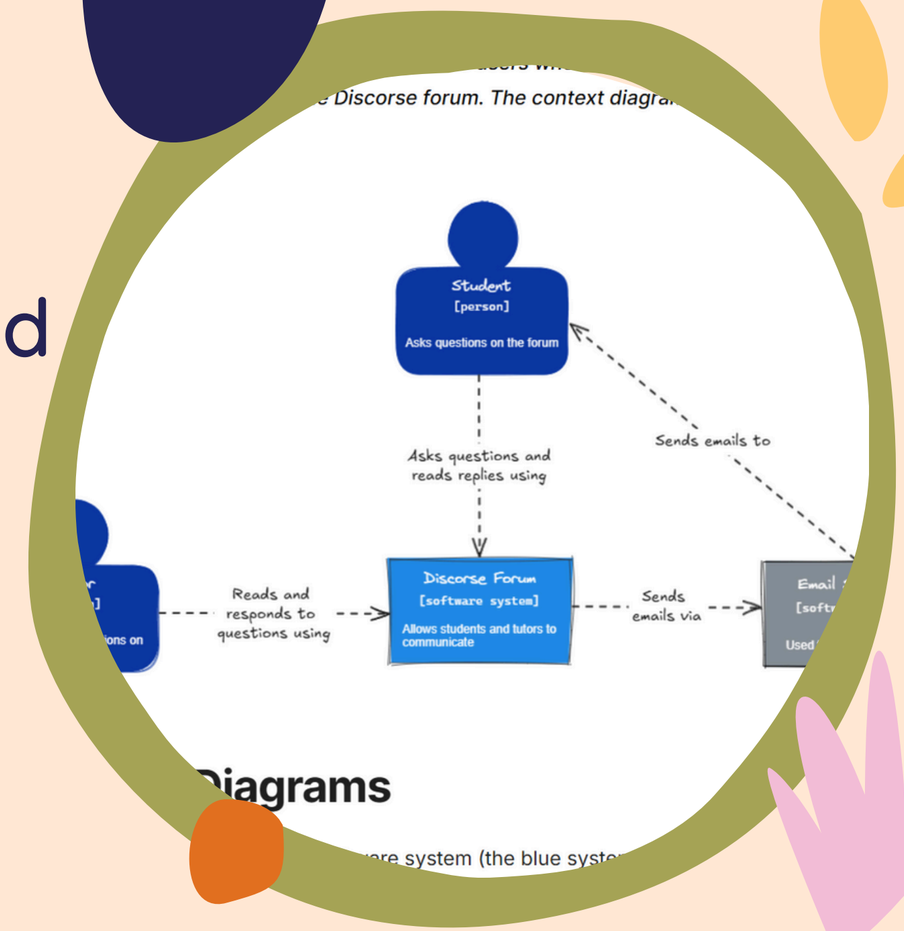
- Used to model architecture
- How individual components within the system interact



# 4 Core Diagrams

In order of least to most detailed:

- Context Diagram - system as a “box” + interactions with users and external systems.
- Container Diagram - system as containers + their interactions
- Component Diagram - zoom into specific container to show components and relationships
- Code (Class) Diagram - like UML

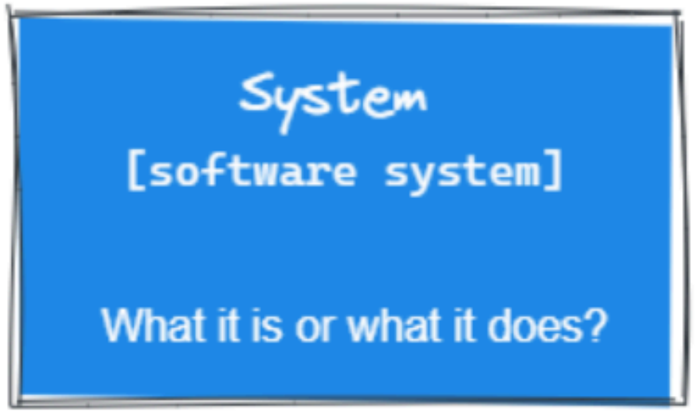


# Person

- Any individual that interacts with system
- Include name, subtitle (always [person]), short description

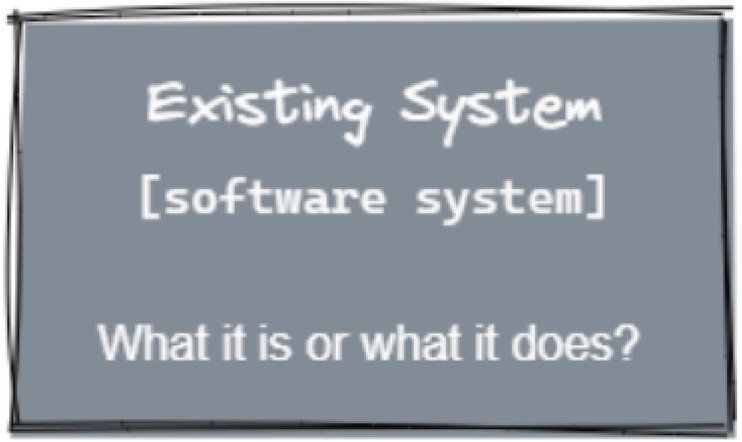
# Software System

- System planned to be implemented (ONLY 1 IN DIAGRAM!!!)
- Include name, subtitle (always [software system]), short description



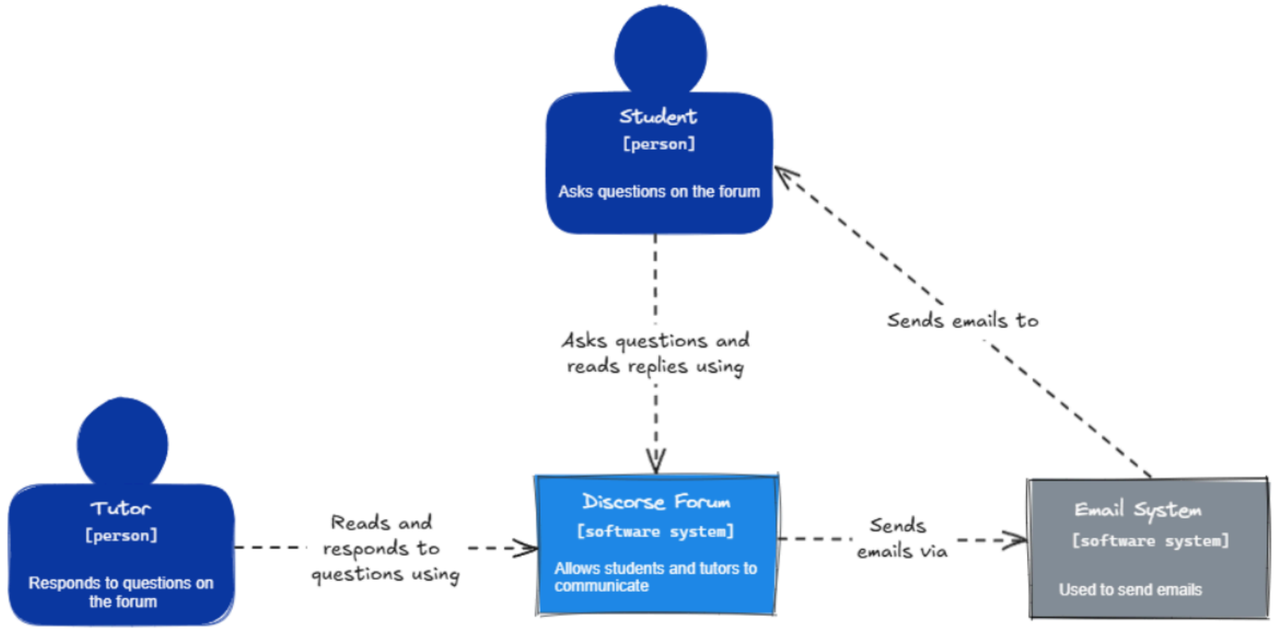
# Existing Systems

- Existing systems that your software interacts with
- Include name, subtitle (always [software system]), short description



# Relatonships

- Arrows between entities and containers to show interaction
- Include a short description of the interaction




3.3. Existing Sy…
3.4. Relationships
4. Container Diagrams
4.1. Converting Context Diagrams … Container Diagrams
4.2. Containers
4.2.1. Frontend Containers
4.2.2. Database Containers
4.2.3. Regular Containers
4.3. Relationships

# Containers

- Include name, subtitle (always [container]), description
- Specifically, Frontend, Database, and Regular
- Regular describes containers that aren’t Frontend or Database

# Sequence Diagrams

When would you use a Sequence Diagram instead of a Class Diagram (similar to UML)?

# Sequence Diagrams

Describe and provide an example for a Sequence Diagram actor, lifeline, activation bar, message.

# Sequence Diagrams

How does Sequence Diagram represent synchronous vs asynchronous?

# Sequence Diagrams

How would we model conditional behaviour and looping?

# Sequence Diagrams

How can exception handling be represented?

# Sequence Diagrams

What can we do when Sequence Diagrams become too complex to read?

# C4 Model

What is the purpose of the C4 Model? Name the 4 Core Diagrams in the C4 Model.

# C4 Model

Who is the intended audience for the Context Diagram (simplest)?

# C4 Model

What is the purpose of the C4 Model? Name the 4 Core Diagrams in the C4 Model.

# C4 Model

How does a Container Diagram differ from a Component Diagram?

# C4 Model

In a Container Diagram, how would a shared database used by 2 services be represented?

# C4 Model

Why split frontend & backend into two different containers?

# C4 Model

Why is it important to describe responsibilities of each container in the diagram?

# ASSIGMENT ii!!!

- Assignment ii IS OUT! - so take a look :D
- Enormous codebase - UNDERSTAND WHAT THE CODE IS DOING BEFORE ATTEMPTING TO REFACTOR IT
- Part 1 is very similar to the Labs 4-7, applying design patterns to refactor and extend the codebase.
- Must pass regression tests + dryrun.
- Get started early!!! - So many half-completed assignment i’s ;(