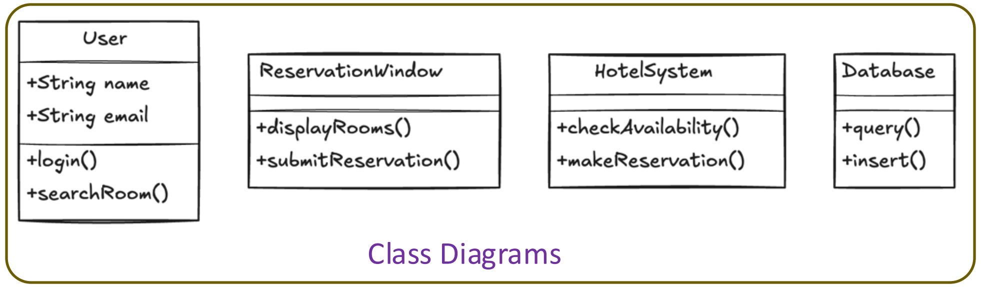
Behavioural Modelling

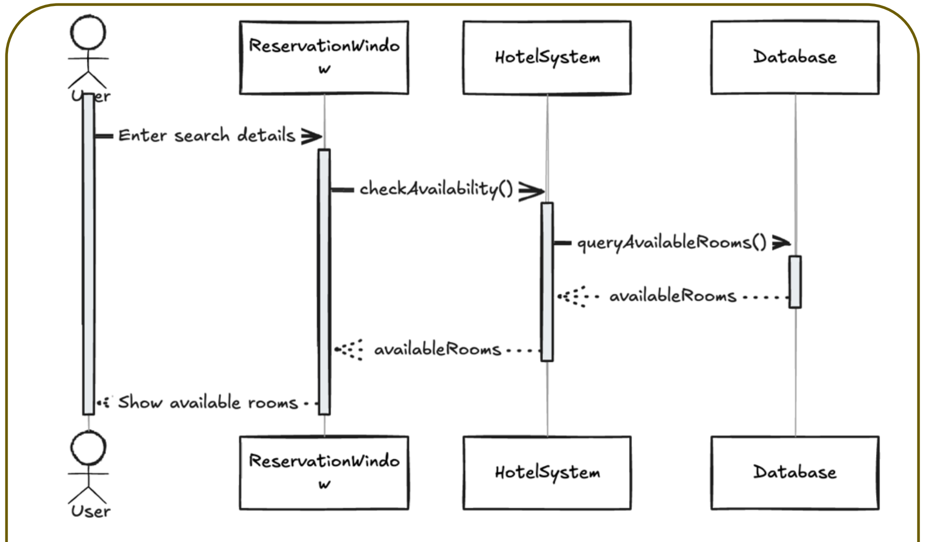
COMP2511, CSE, UNSW

What is Behavioural Modelling

- Behavioural modelling captures *how* the system behaves in response to events or interactions over time.
- Software Design and Architecture **do not** tell us how components behave or interact over time.
- Different notations for expressing behaviour:
  - Sequence diagrams
  - Activity diagrams
  - State charts



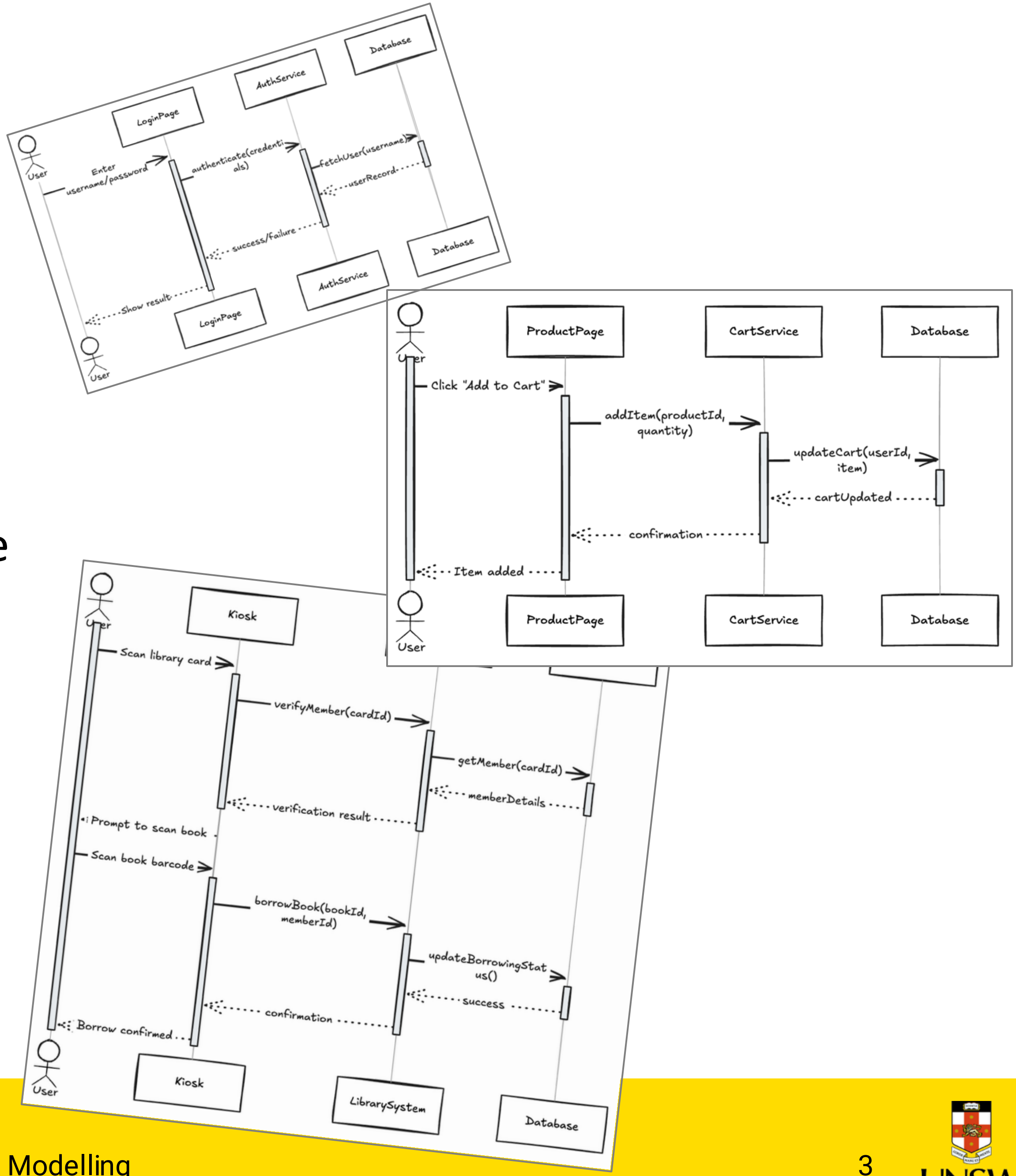
Class Diagrams



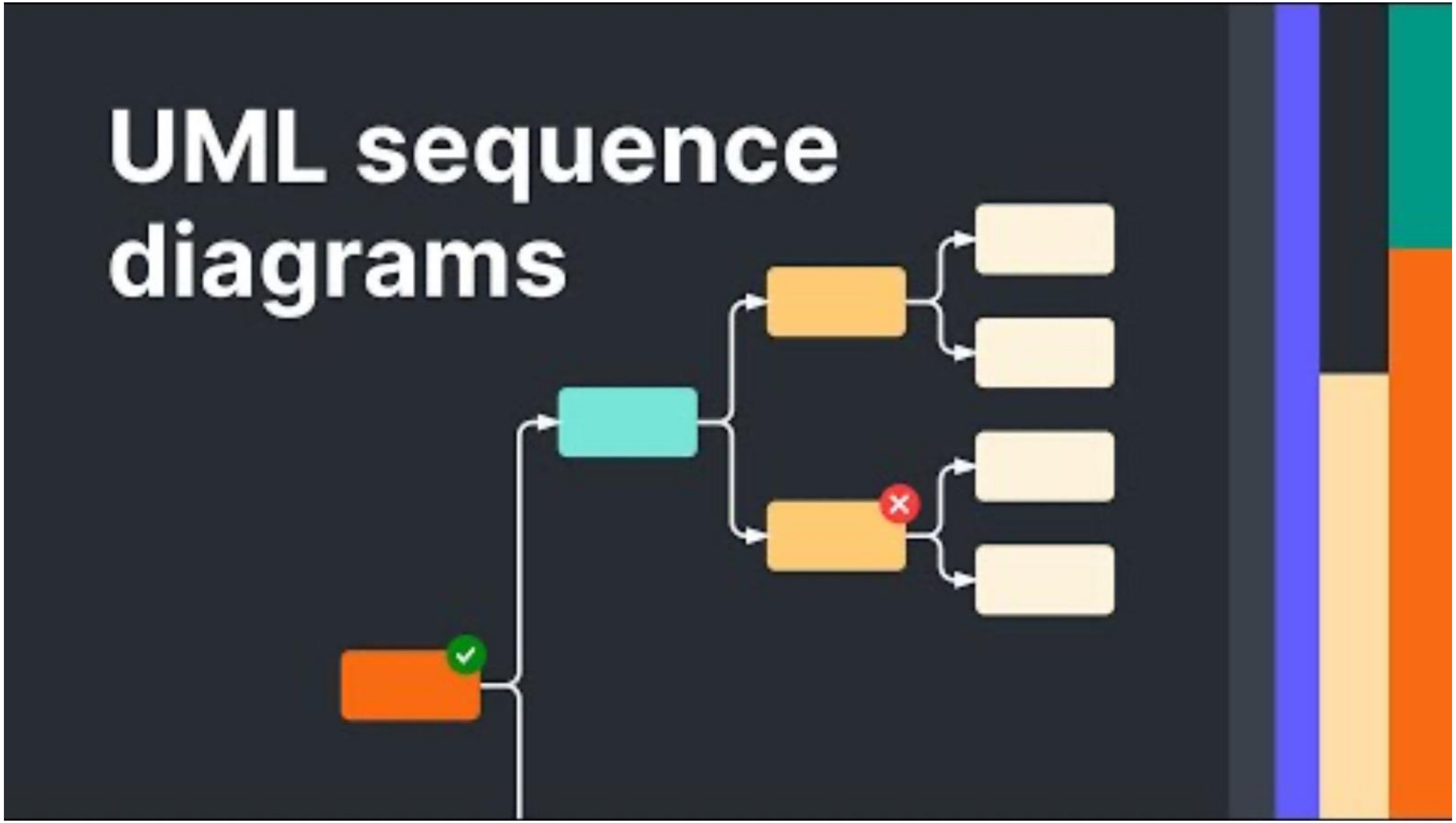
Sequence Diagram

Sequence Diagrams

- A sequence diagram is an interaction diagram showing how objects interact in a time-sequenced manner.
- Clarify interactions among objects and improve system behaviour understanding.
- Show how operations are carried out through message exchanges.
- Emphasize the temporal order of interactions.

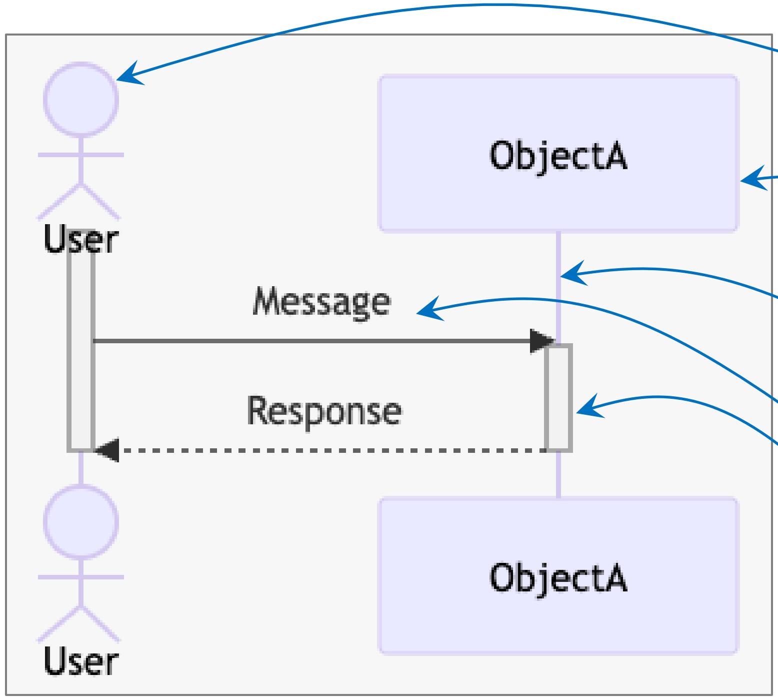


More on sequence diagrams



https://www.youtube.com/watch?v=pCK6prSq8aw (8 mins)

Key Components of a Sequence Diagram



- User sends a message to ObjectA, activating ObjectA's processing.
- ObjectA responds, deactivating afterwards.

Actor: External user or system

Objects: Entities involved, represented by rectangles.

Lifelines: Lines (or dashed lines) showing object existence during interactions.

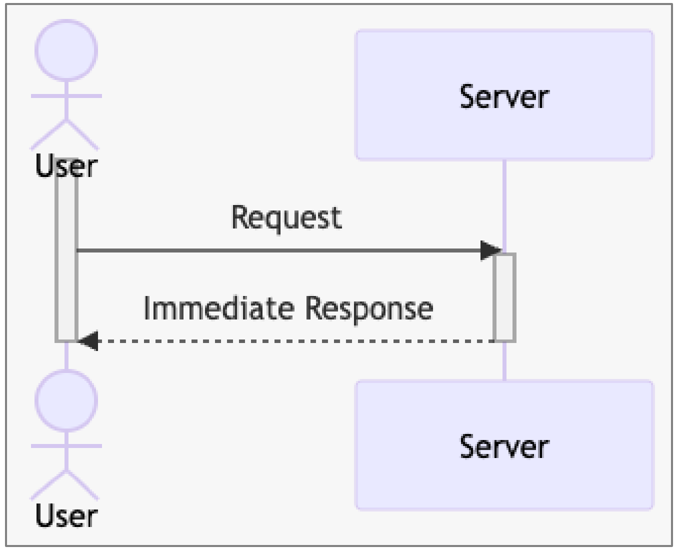
Messages: Communication between objects.

Activation Boxes: Indicate active processing of messages.

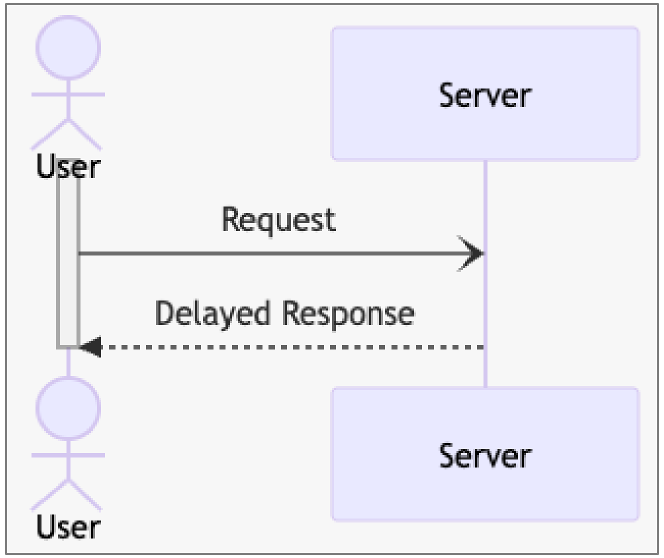
Types of Messages

Synchronous: Sender waits for a response.

Asynchronous: Sender does not wait for an immediate response.



Synchronous: User waits for Server to complete processing before proceeding.



Asynchronous: User continues immediately without waiting for Server's response.

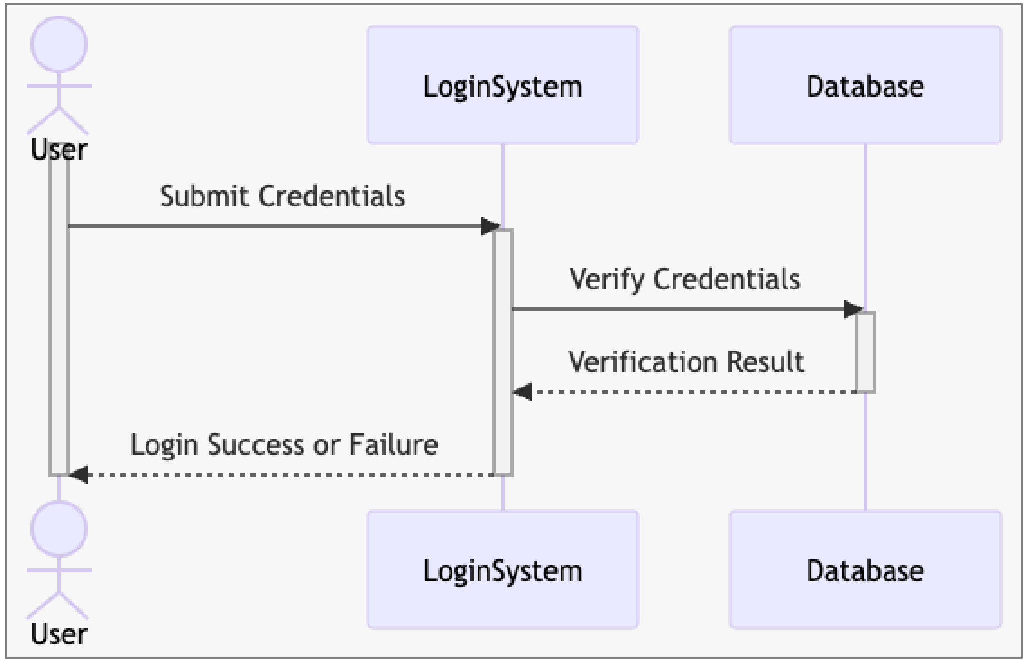
Sequence Diagram Overview

- Horizontal axis captures participating objects.
  - Objects are placed from left to right.
  - Order reflects participation in message sequence.
  - Horizontal layout is flexible but typically chronological.

- Vertical axis represents time (top to bottom).
  - Time flows downward.
  - Sequence diagrams focus on *order*, not *duration*.
  - Vertical spacing is not indicative of actual time intervals.

- Messages are shown as horizontal arrows. Messages can be calls/invocations for some methods in a component, or results given by that component.
- Execution shown using rectangles (activations boxes).

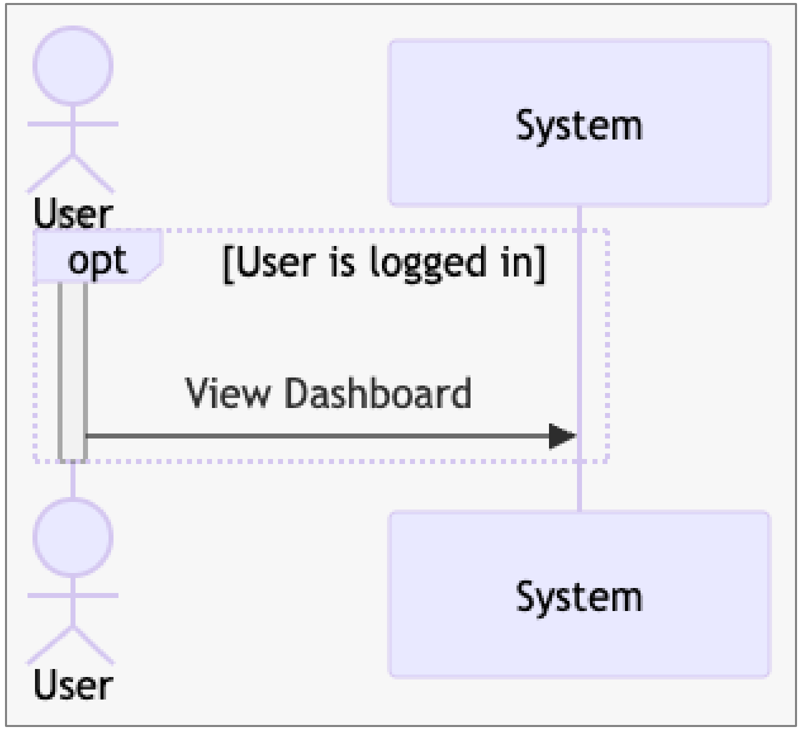
Example: User Login



User's credentials are checked against a database, resulting in either login success or failure.

Optional Interaction

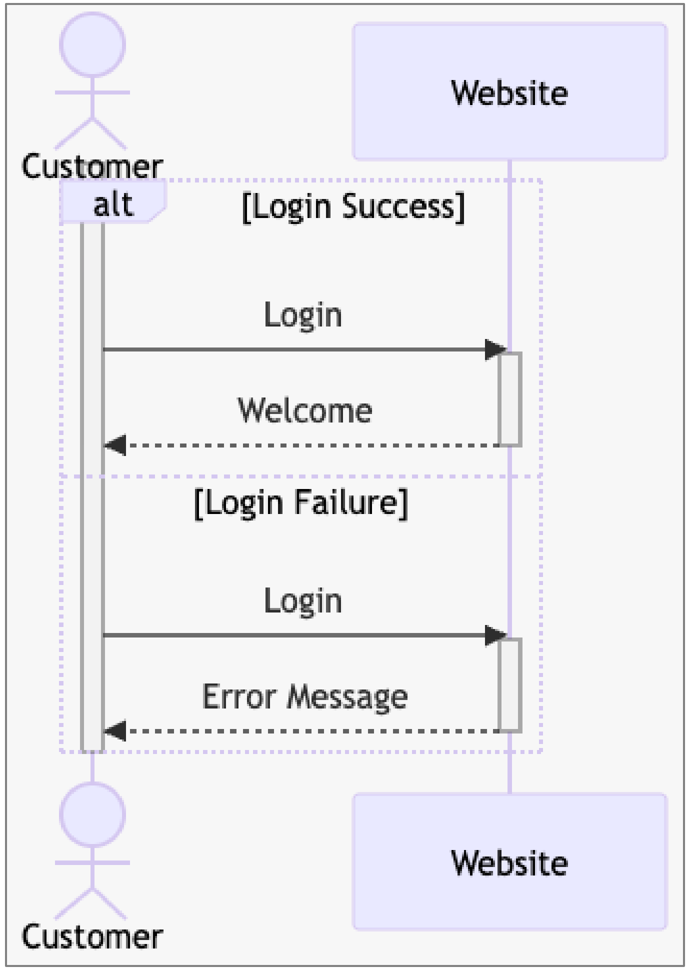
- **opt** represents **optional** scenarios.



Illustrates **optional logic** based on condition result (success).

Conditional Interaction

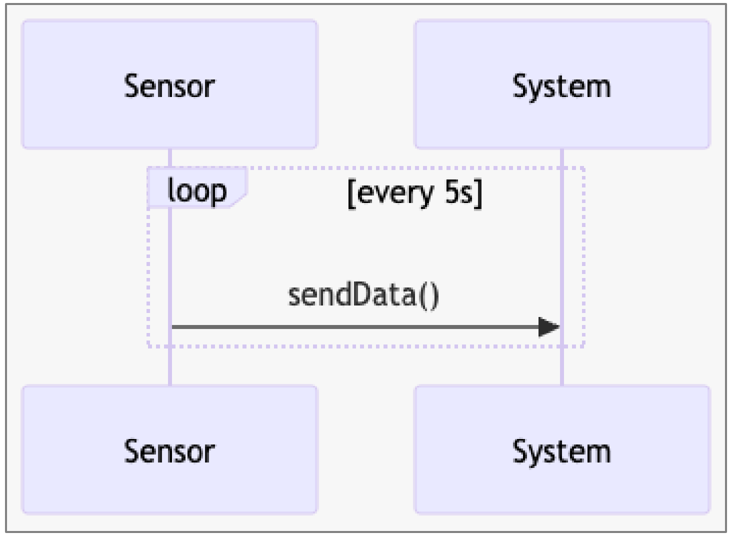
- **alt** represents **alternate** scenarios.

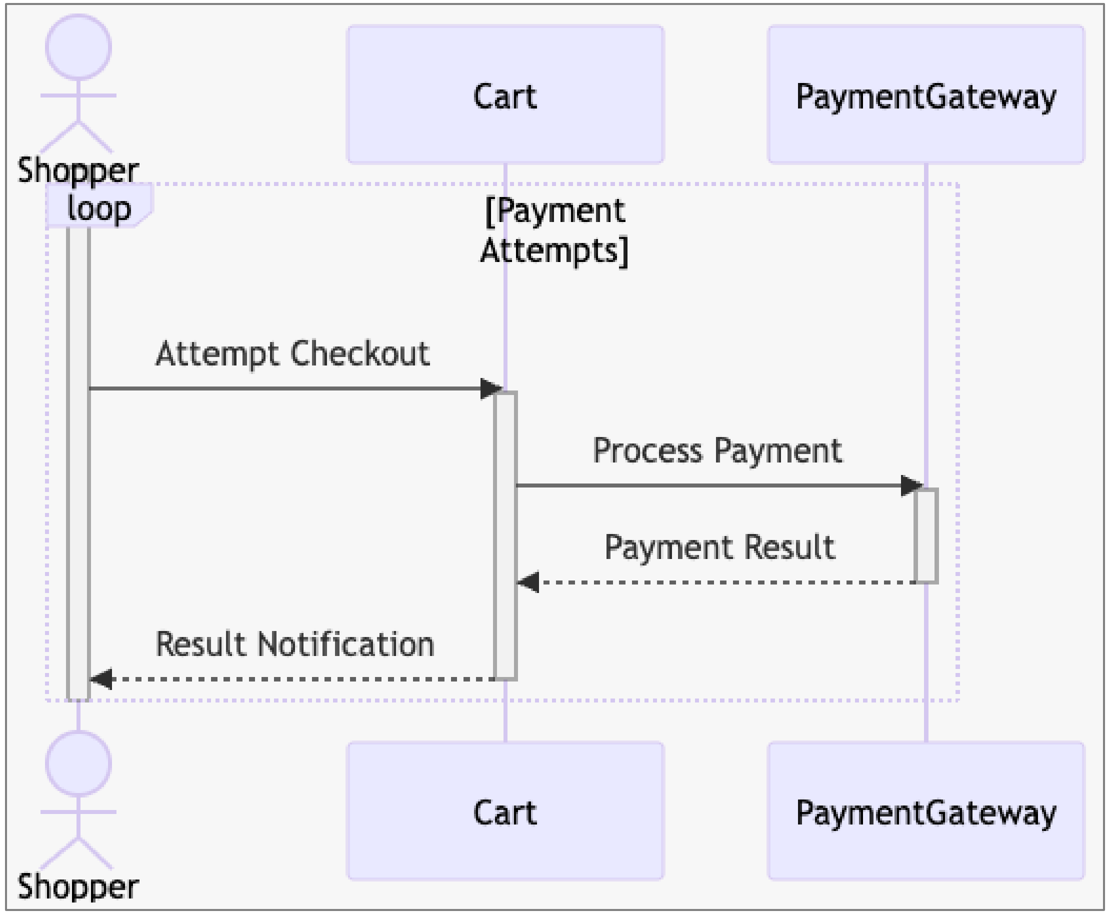


Illustrates **branching logic** based on condition results (success or failure).

Looping Interaction

- **loop** represent **repeated** actions.

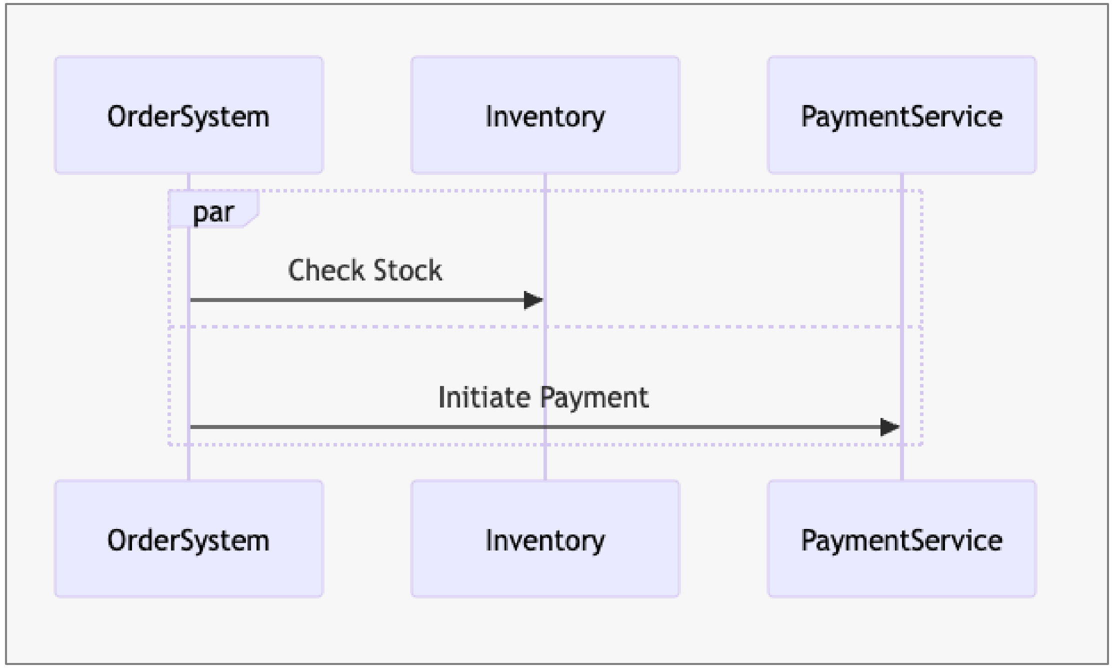




The loop continues until payment is successful, emphasizing **iterative** process.

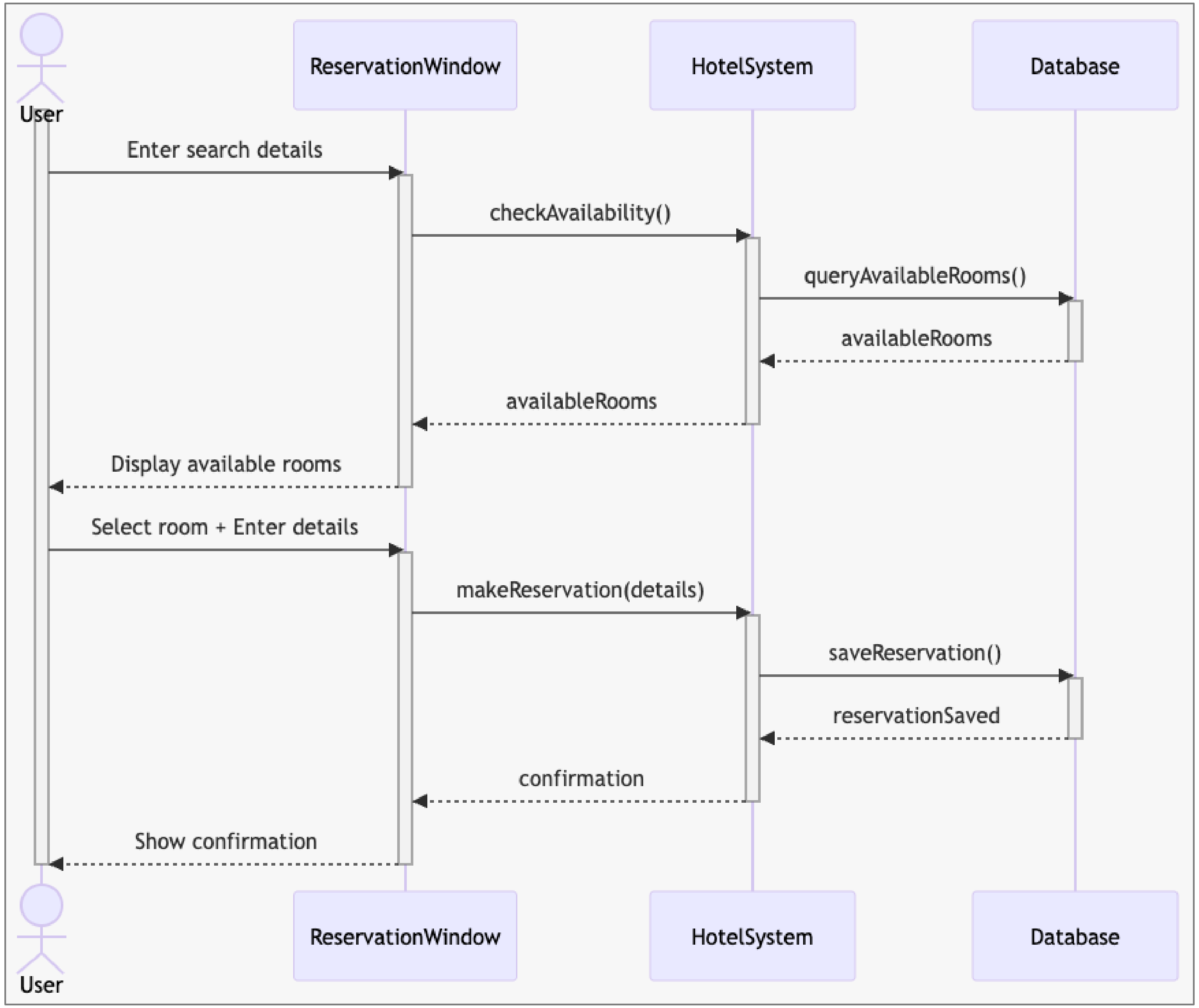
Parallel Processes

par represents **concurrent** processes.

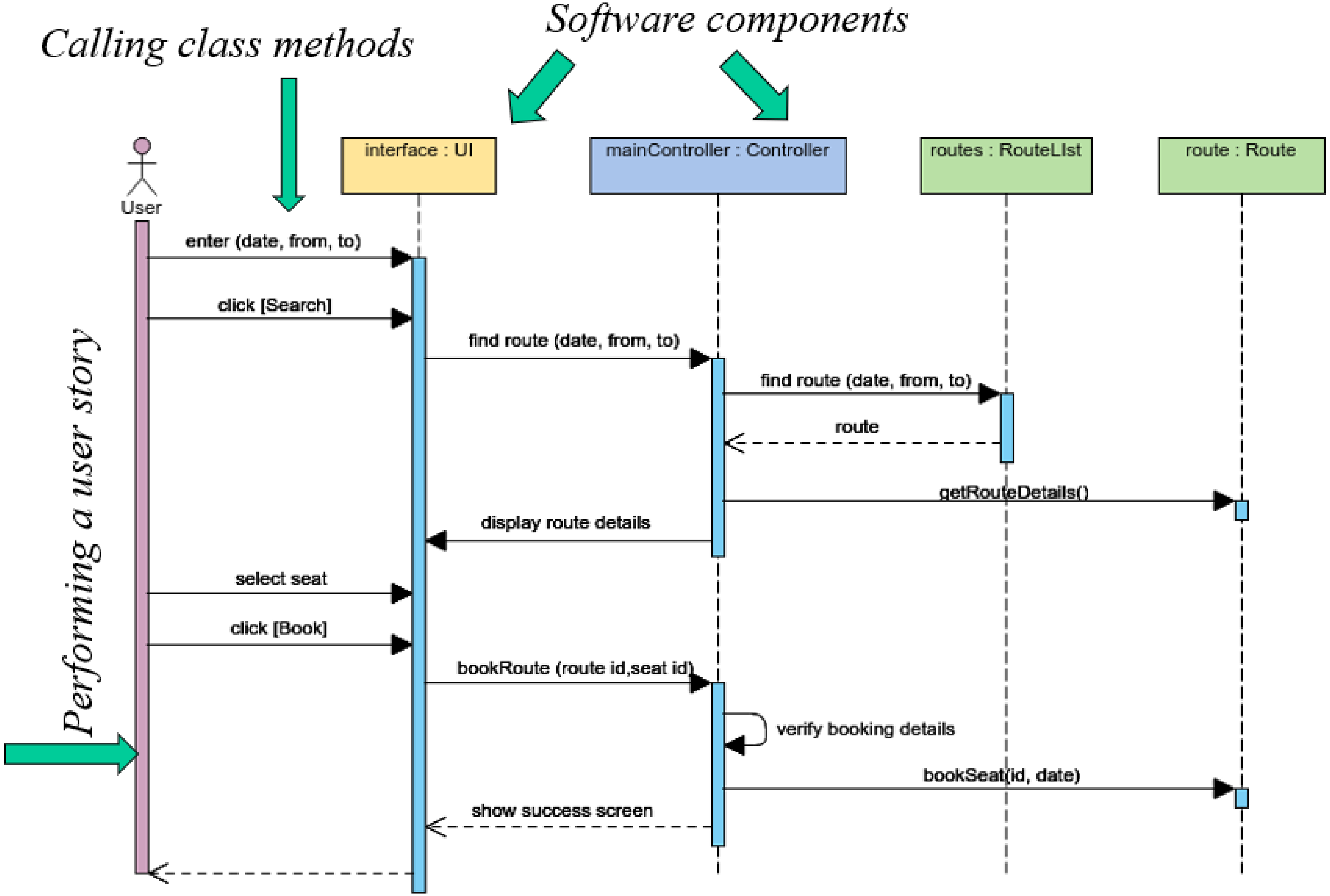


Multiple processes, such as inventory checking and payment, occur **simultaneously**.

Example:
Hotel Reservation



Example: Airline Booking



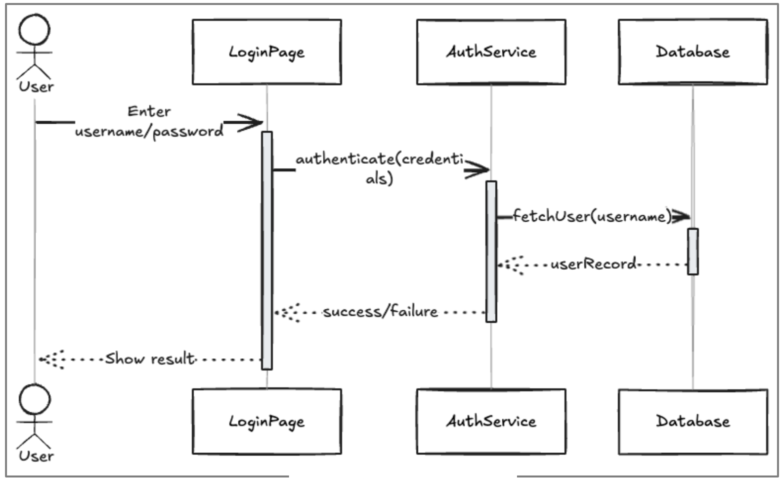
Calling class methods

Software components

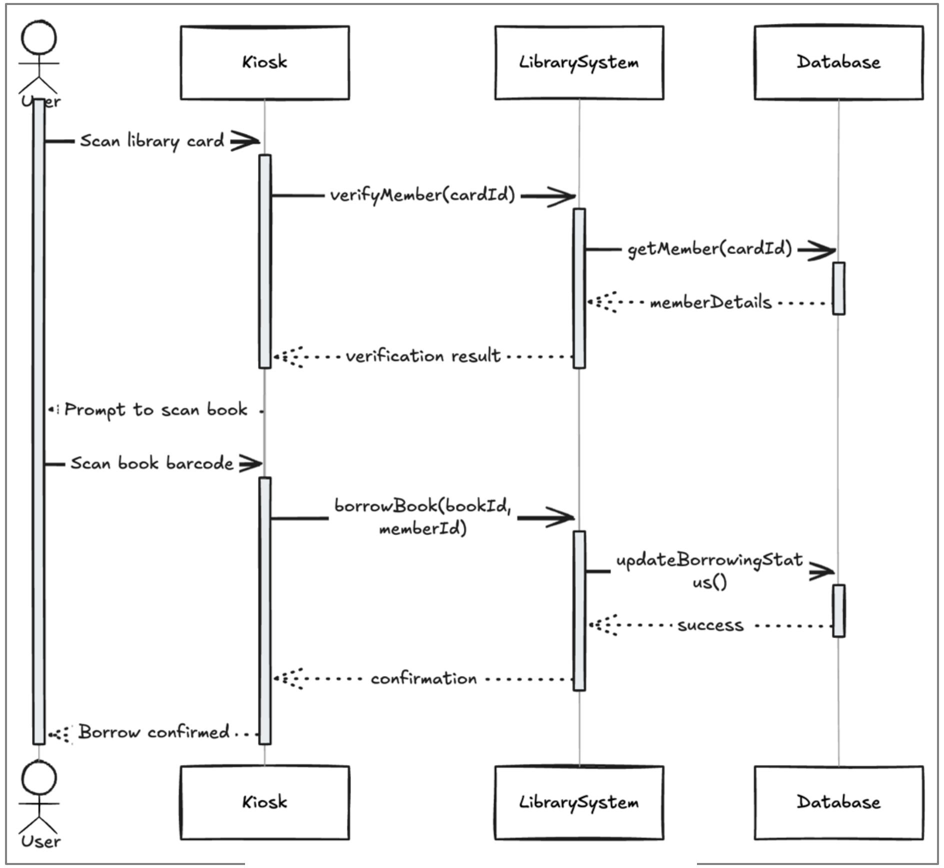
Performing a user story

❖ More information in The sequence diagram – IBM Developer

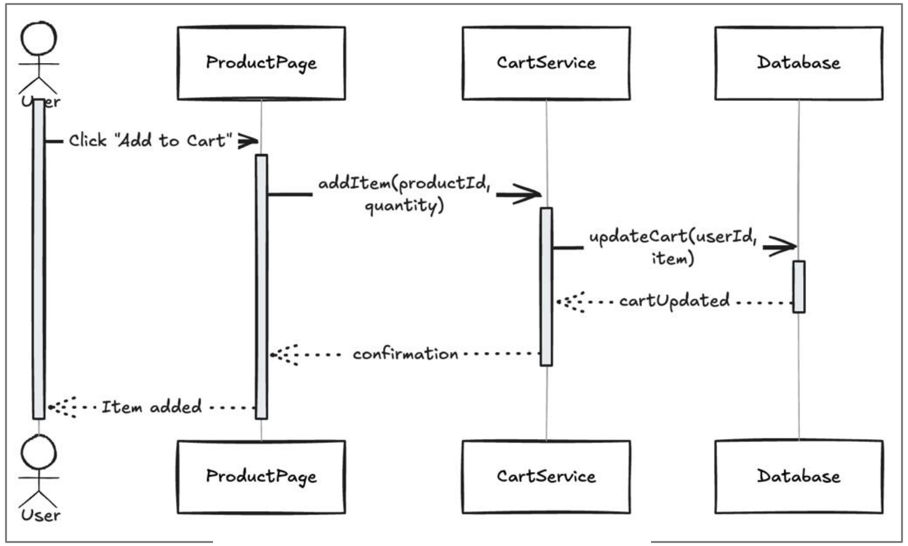
Examples



Login Process



Library System – Borrow Book



Online Shopping – Add to Cart

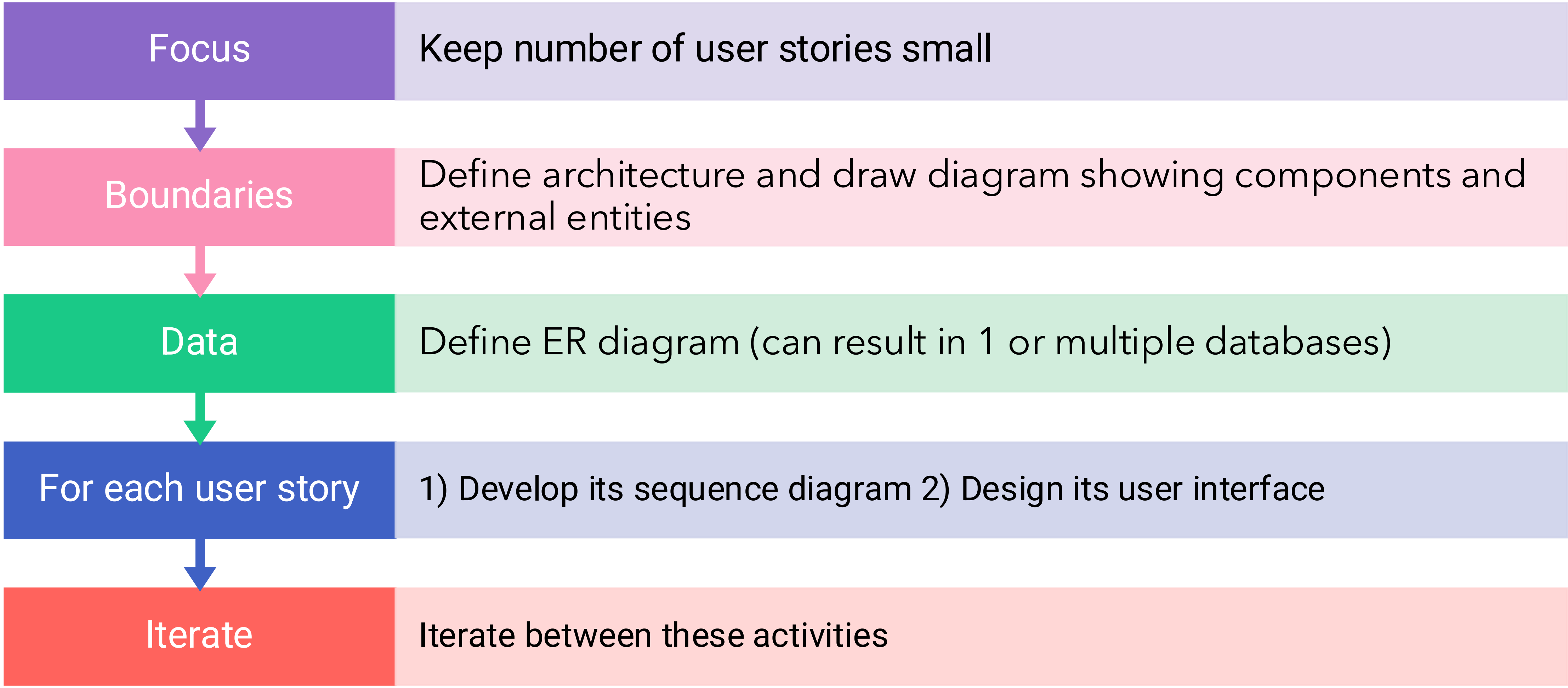
Benefits of Sequence Diagrams

- Clarifies interaction order and logic.
- Identifies inefficiencies and redundancies.
- Enhances team communication.
- Aids debugging and improves process clarity.
- Improves collaboration and understanding.

Common Mistakes

- Overcomplicating diagrams.
- Undefined roles and interactions.
- Incorrect message ordering.

Suggested Design Process in Software Engineering



Focus: Keep number of user stories small

Boundaries: Define architecture and draw diagram showing components and external entities

Data: Define ER diagram (can result in 1 or multiple databases)

For each user story: 1) Develop its sequence diagram 2) Design its user interface

Iterate: Iterate between these activities

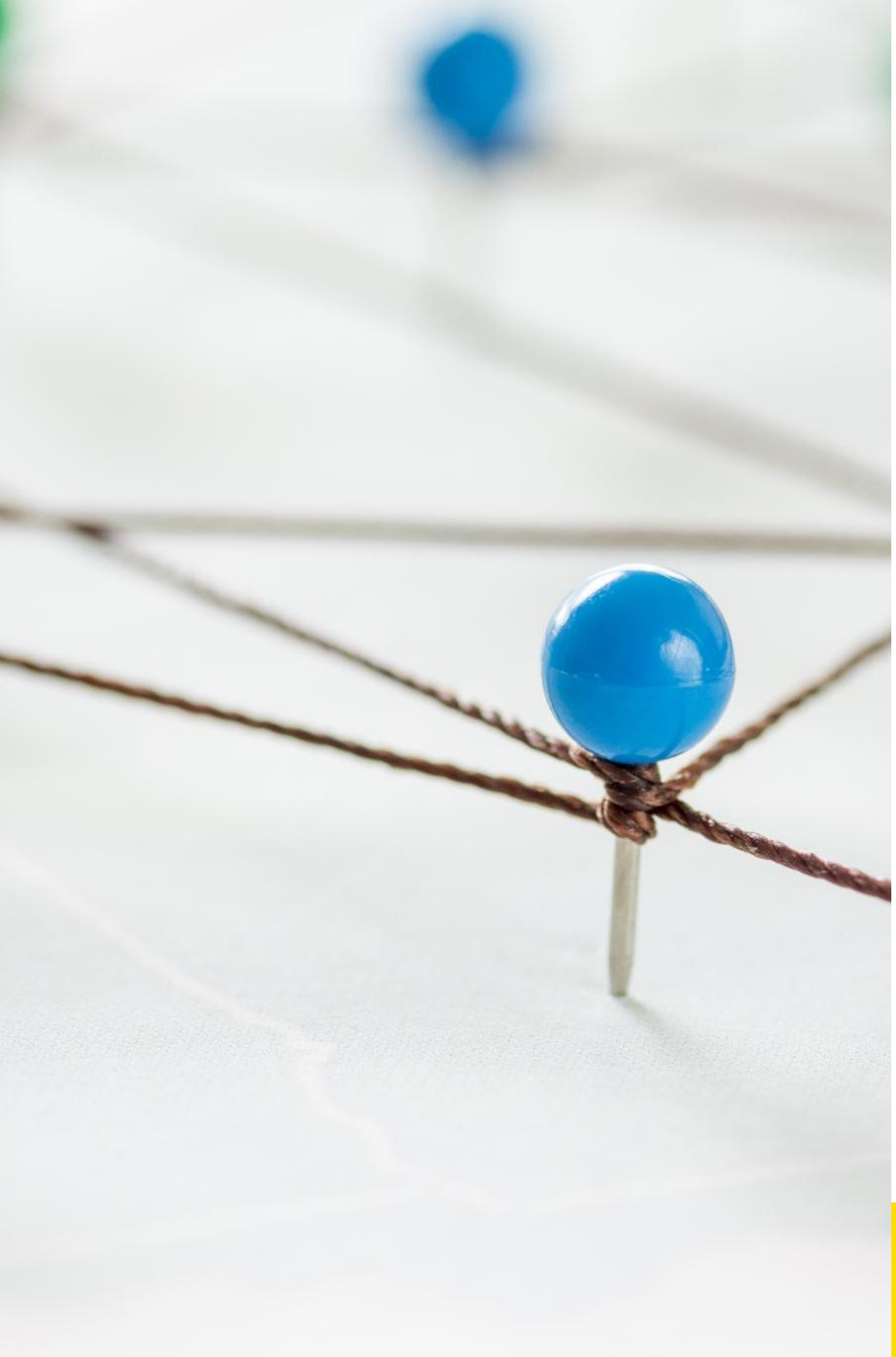
Good Software Design Practices

Things to do

- Keep design documents “live” and shared between team members
- Use design as a way to decompose work
- Discuss design changes as a team

Things to avoid

- Too much focus on notation
- Quantity over quality
- Creating something for other manager (tick boxes) and forgetting design is for team



Web resources

Sequence diagrams

- Sequence Diagram Tutorial - Complete Guide with Examples (creately.com)
- Sequence Diagram Tutorial (visual-paradigm.com)
- UML Sequence Diagram Tutorial | Lucidchart

Software design principles

- Software Design Principles | Top 5 Principles of Software Development (educba.com)