



University of
South Australia

INFS 2044

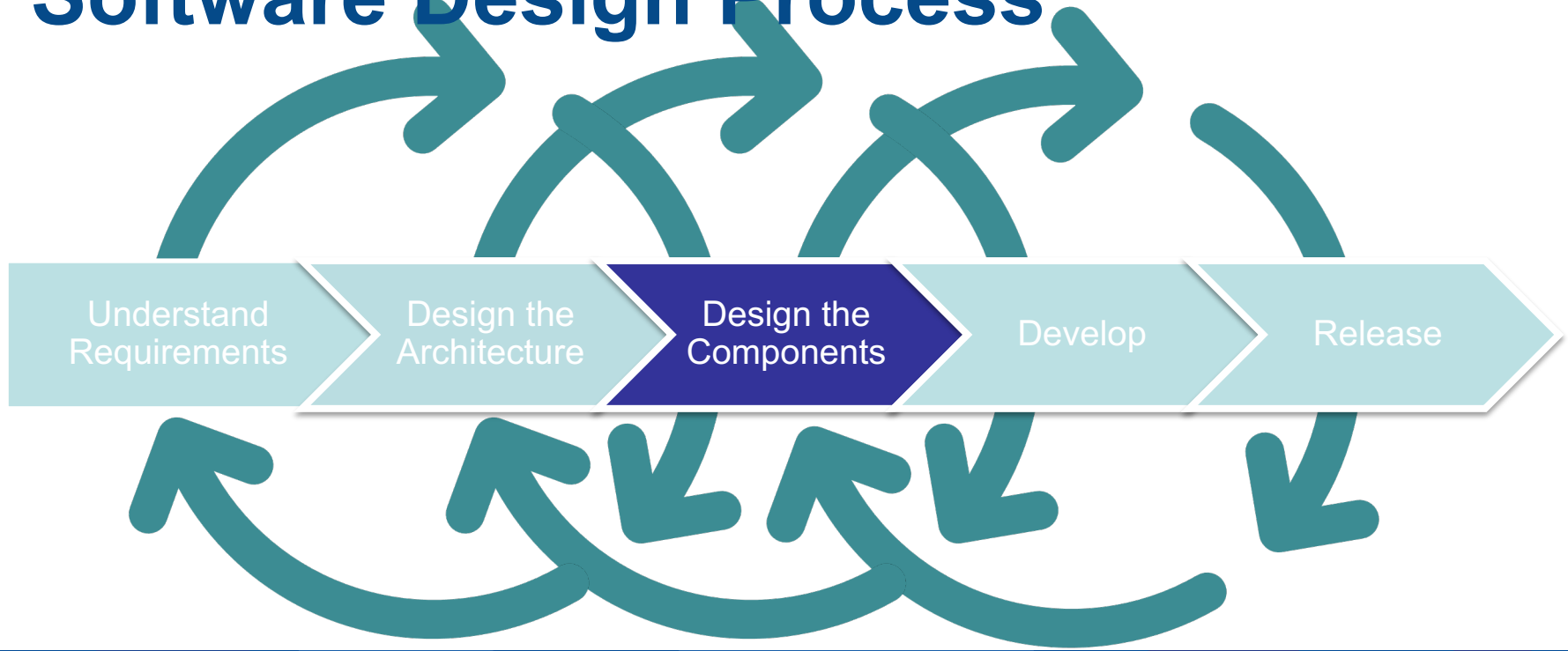
Workshop 3a

Preparation

- Read the required readings
- Watch the Week 3 Lecture
- Bring a copy of the workshop instructions (this document) to the workshop



Software Design Process



Learning Objectives

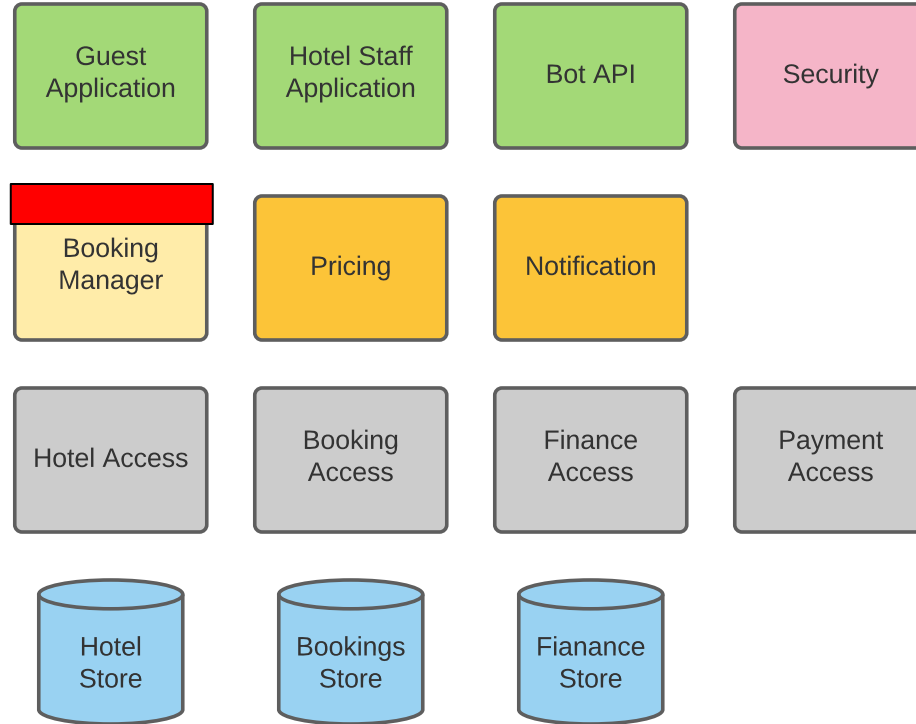
- Define system interfaces
- Assess alternate interface designs
- Detect information leakage and poor abstractions



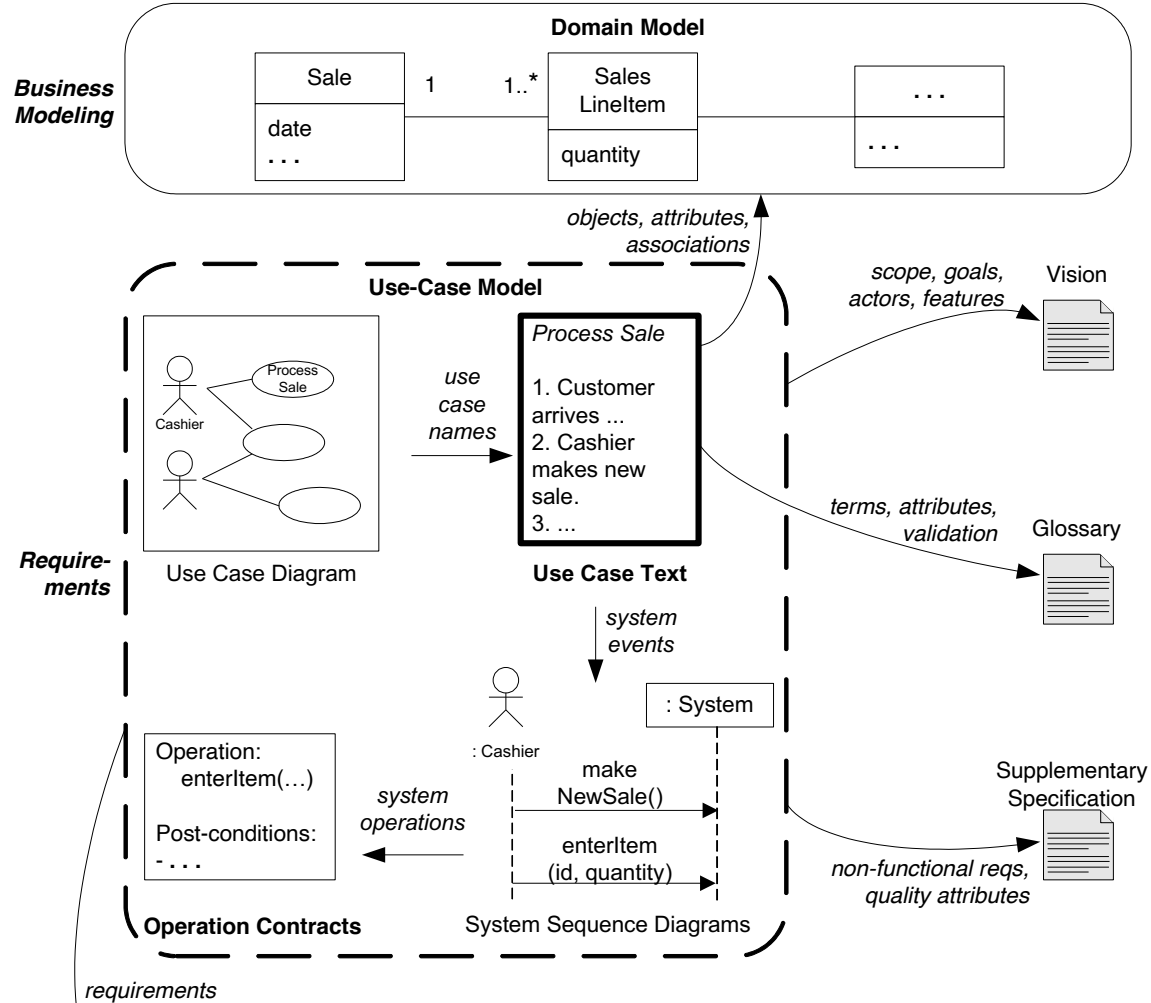
Task 1. Define System Operations

- Define the interface (API) for the system-level operations for the Make Booking use case
- For each operations, define its contract.
 - List operation signature
 - » Name
 - » Parameters
 - » Result





Sample UP Artifact Relationships



Operation Contract

Operation: **operationName(Parameters): ResultType**

CrossRef: use cases this operation occurs in

Preconditions: any assumptions about the state of the system (will not be tested by the operation)

Postconditions: the state of the system after the operation completes



findRooms(...) Example

- **findRooms(checkin:Date,checkout:Date):** list(RoomDescriptor)
 - RoomDescriptor is a data structure that includes
 - » room ID
 - » room type
 - » description
 - » daily rate in AUD
- Preconditions: none
- Postconditions: none



UC01 Make Booking Main Scenario

1. *User enters date range.*
2. *System presents available rooms, their descriptions, and daily rates.*
3. *User selects room.*
4. *System presents total price.*
5. *User enters contact details and payment details.*
6. *System verifies that room is available for the period, creates booking for the room and associates booking with guest, verifies payment, records payment confirmation, and issues booking confirmation.*

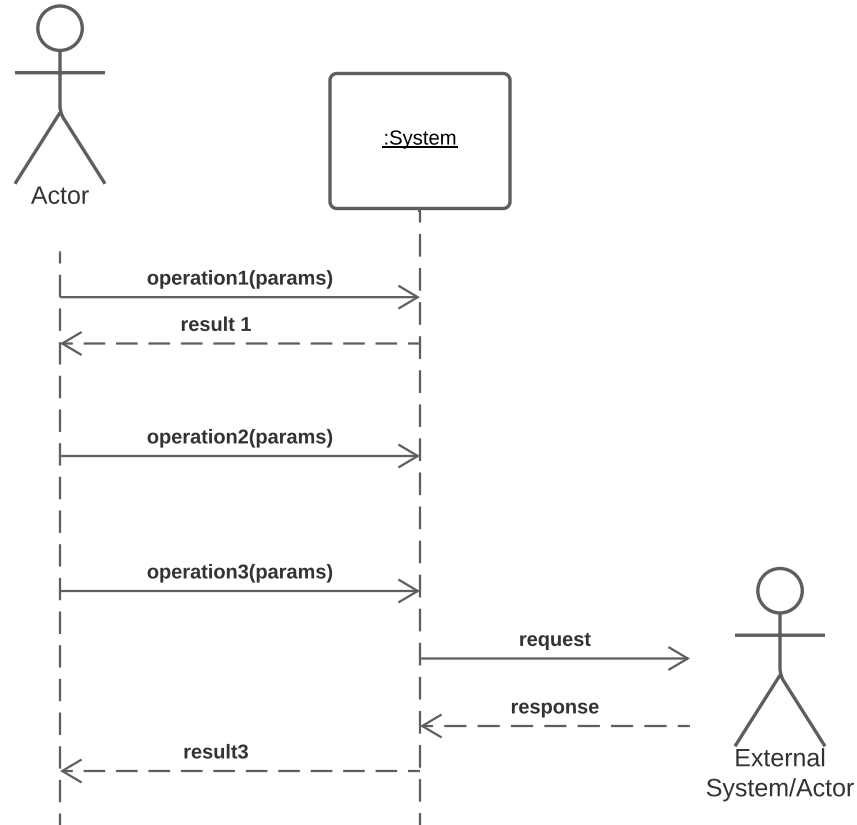


Task 2. System Sequence Diagram

- Draw a System Sequence Diagram for UC01 Make Booking showing the sequence of system operations defined in Task 1.



System Sequence Diagram Example



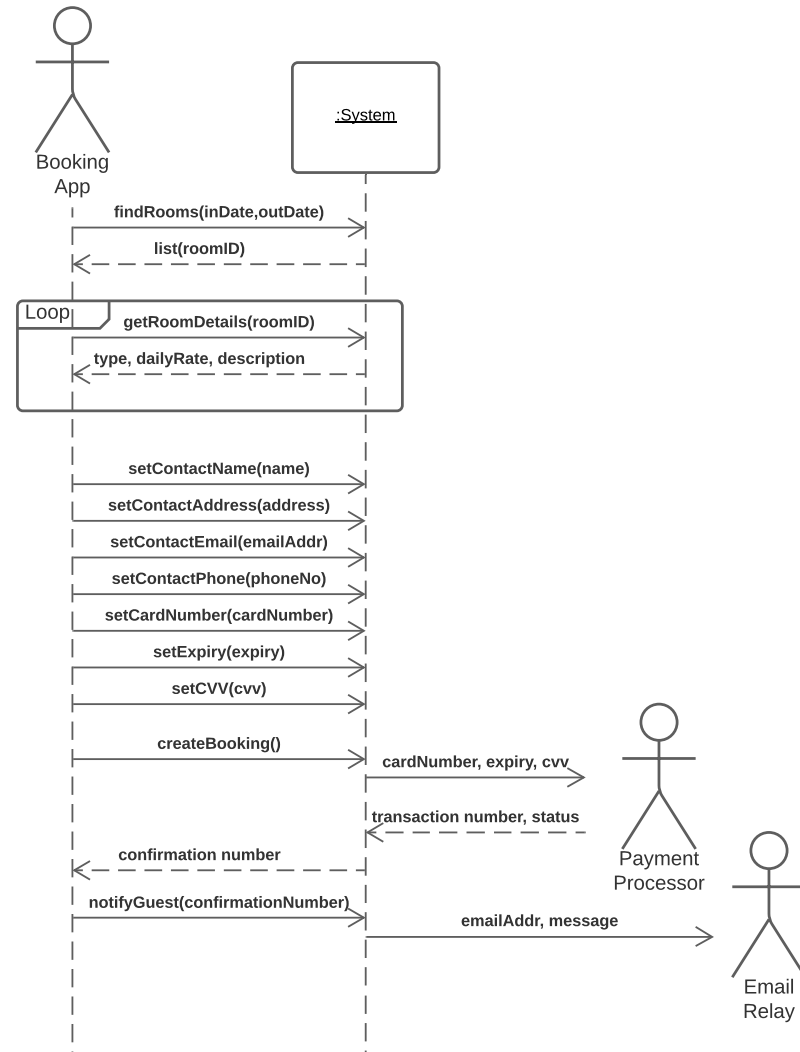
Task 3. Assess SSD Designs

- Assess the quality of the system interface shown in the Make Booking System Sequence Diagram (SSD) #2.
- Use the criteria listed on the following slides.

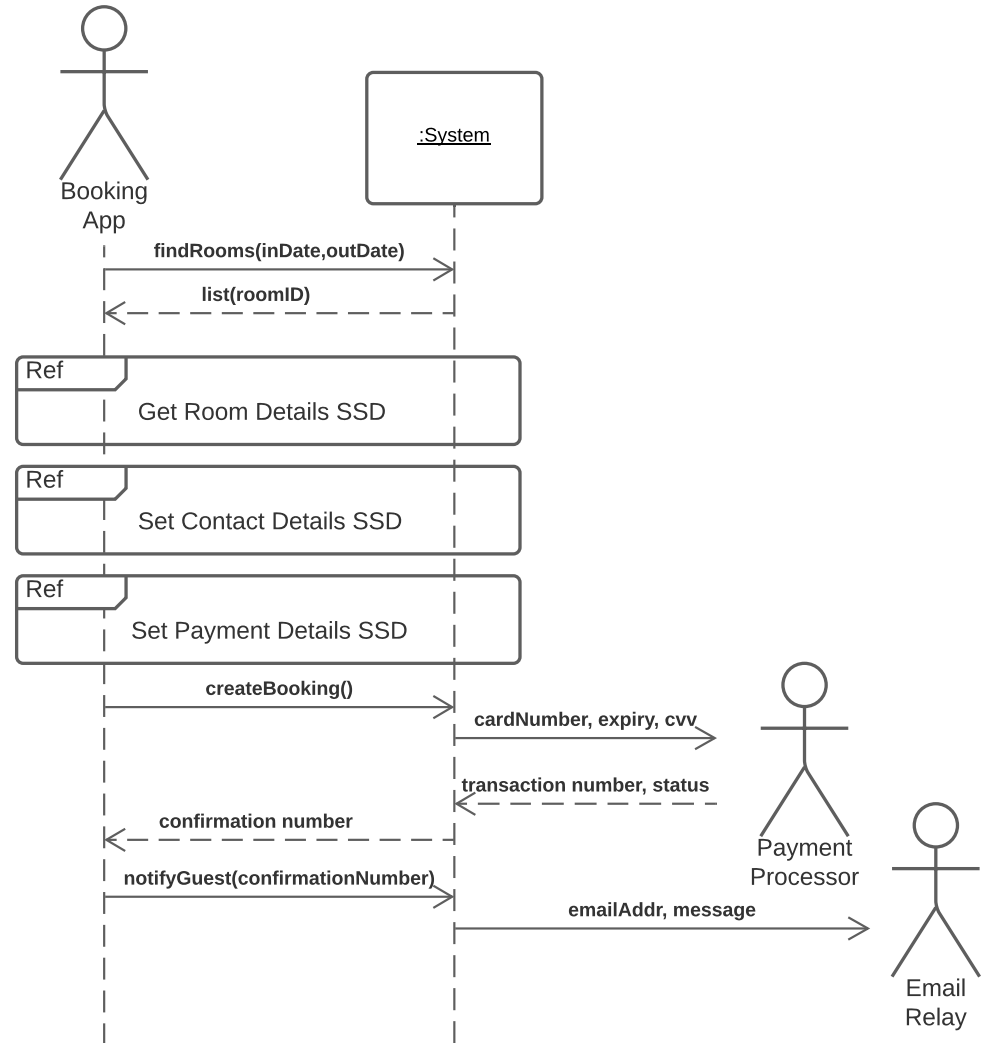


Make Booking

SSD #2

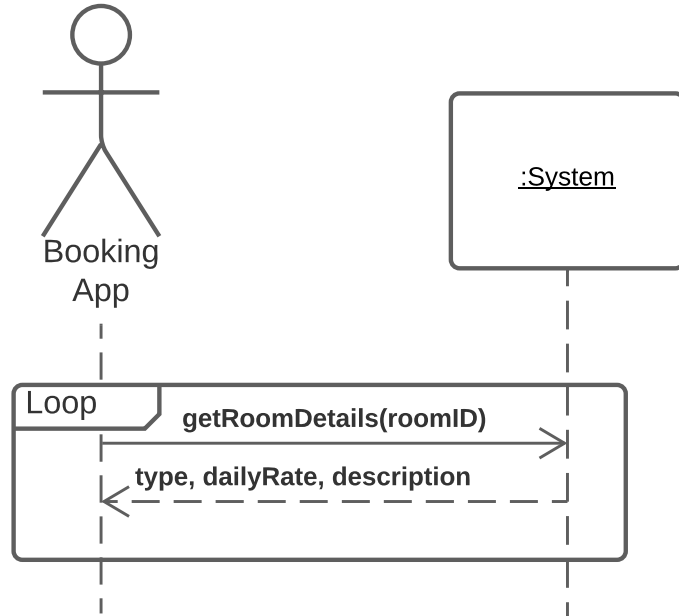


Make Booking SSD #2 Part A

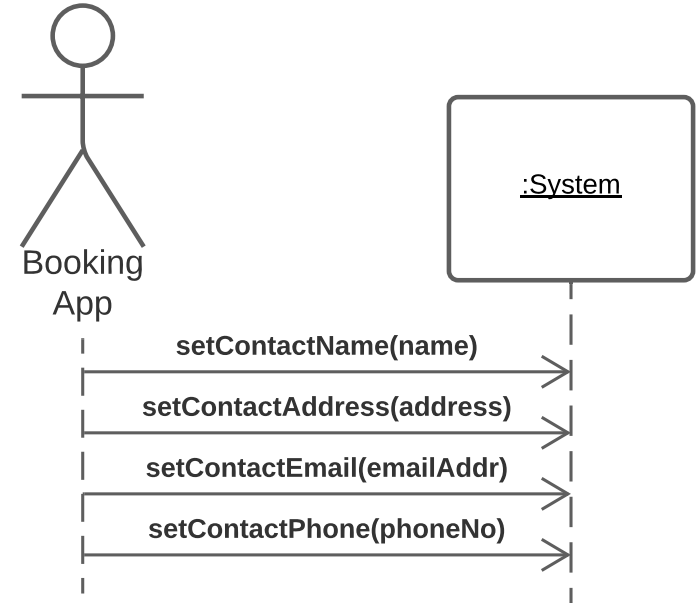


Get Room Details SSD #2 Part B

Get Room Details SSD

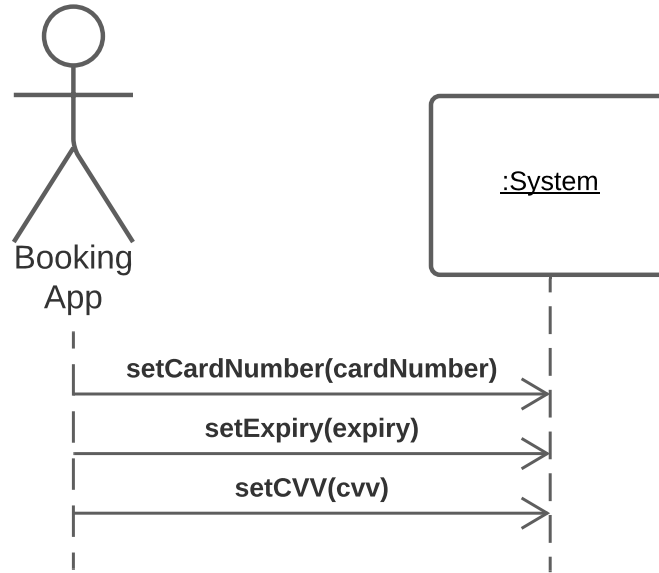


Set Contact Details SSD



Make Booking SSD #2 Part C

Set Payment Details SSD



SSD Criteria #1 – Functional

- Sufficient to implement the use case(s)
- Technically feasible
 - Start with the system receiving a trigger event
 - Follow a well-structured clear call-return patterns
 - Have the information needed to execute each step
 - Data can be properly validated
 - Exceptions can be detected
 - System can maintain consistent state



SSD Criteria #2 – Design

- Not leak implementation details
- Application logic not embedded in the client application
- Decoupled from user interface concepts and interactions
- Consistent and intention-revealing signatures
- Make common scenario easy
- Accommodate alternate use case scenarios
- Operations can be repeated without side effects



SSD Criteria #3 – Non-Functional

- Efficiency of the system
 - Frequency of invocation
 - Latency (non-local calls, marshalling overhead, ...)
 - Data payload size
- Security
 - Data & operations protected (if necessary)



You Should Know

- Design system operations and contracts
- Validate system operations on use case scenarios
- Draw System Sequence Diagrams



Activities this Week

- Attend second Workshop session
- Complete Quiz 3





**University of
South Australia**