University of
South Australia

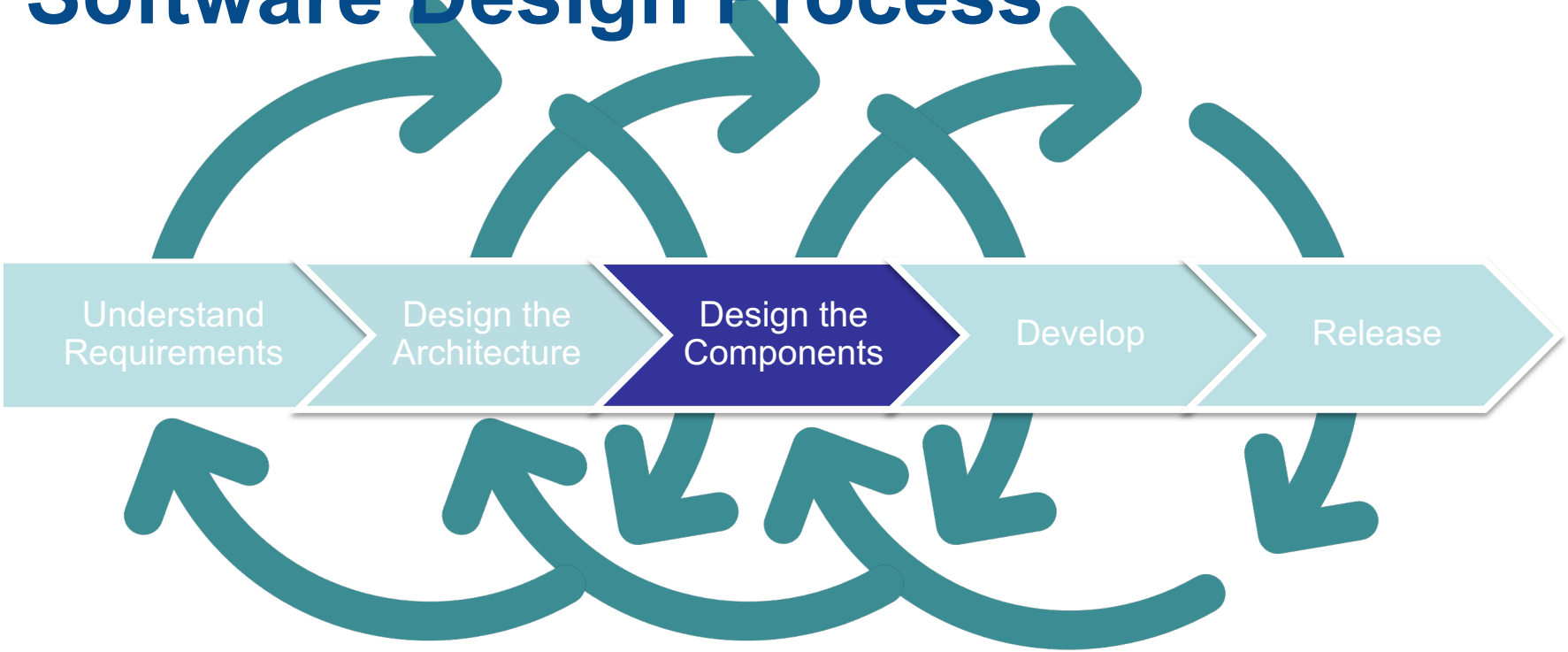
INFS 2044

Workshop 3a Answers

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



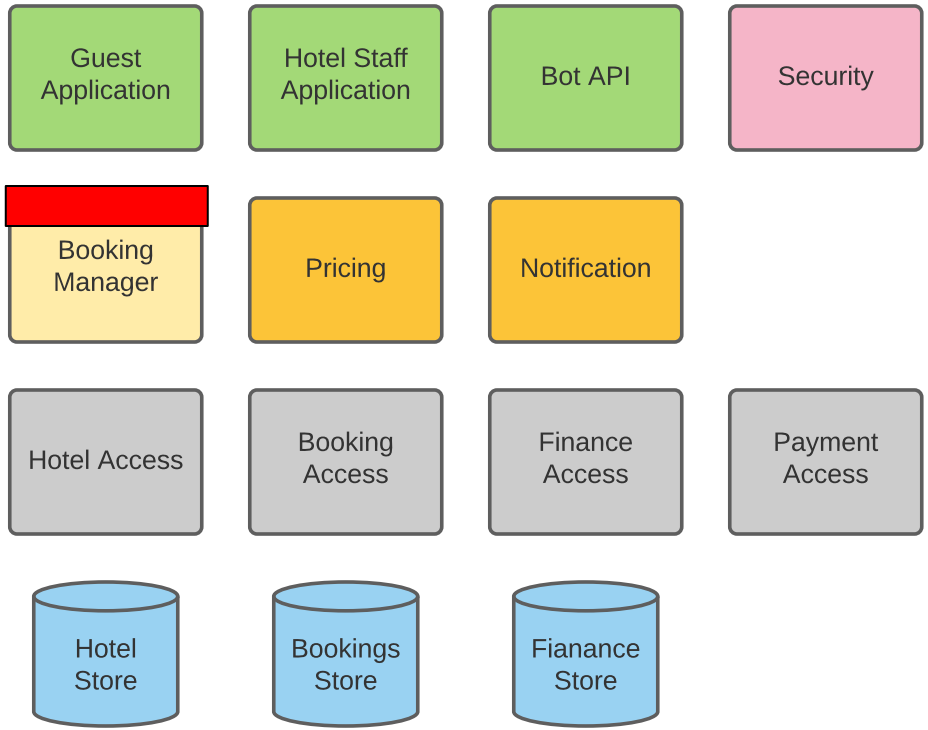
Learning Objectives

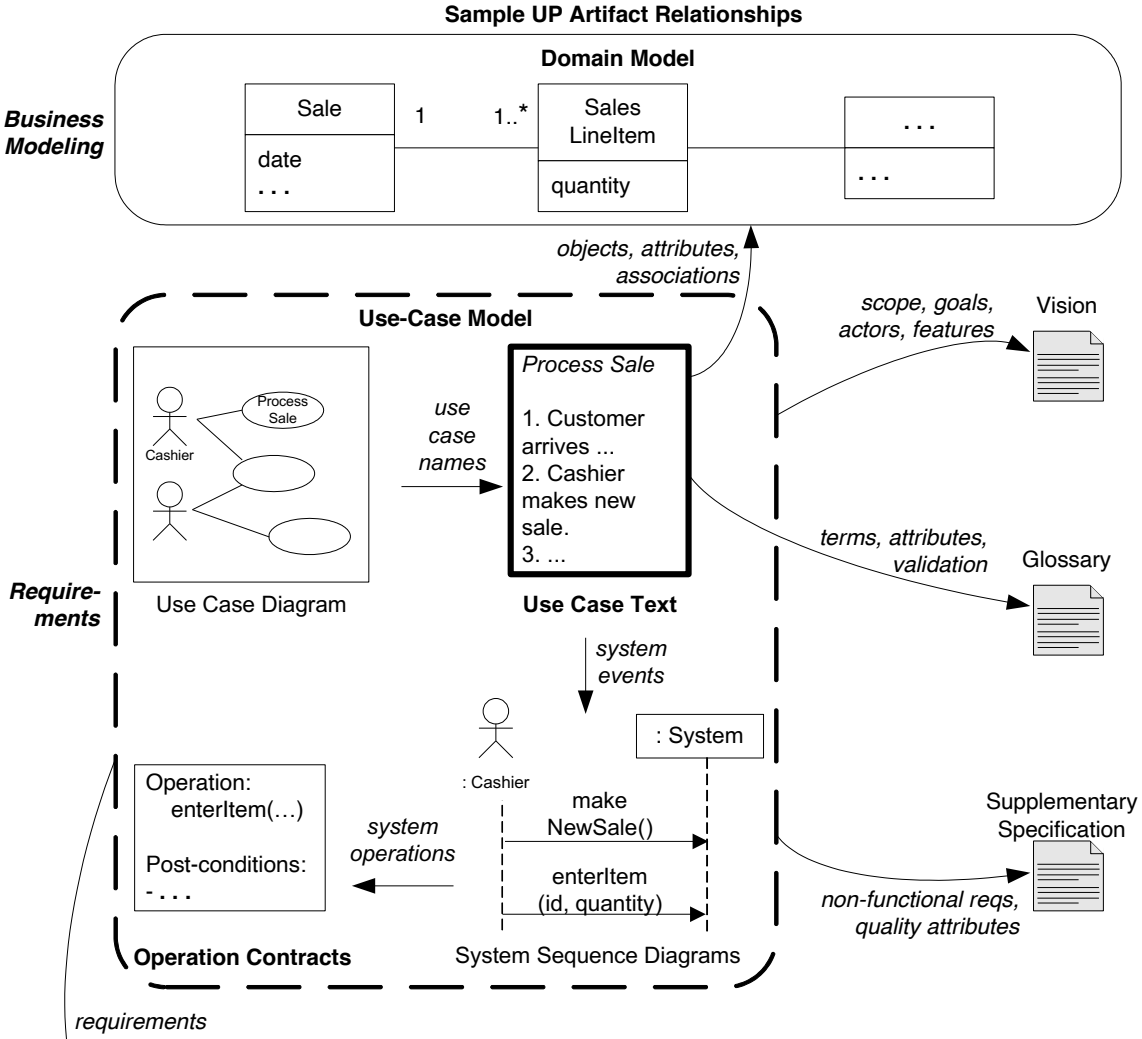
- Define system interfaces
- Assess alternate interface designs
- Detect information leakage and poor abstractions

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


Guest Application
Hotel Staff Application
Bot API
Security
Booking Manager
Pricing
Notification
Hotel Access
Booking Access
Finance Access
Payment Access
Hotel Store
Bookings Store
Fianance Store


Sample UP Artifact Relationships
Business Modeling
Domain Model
Sale
date
. . .
1
1..*
Sales LineItem
quantity
. . .
. . .
objects, attributes, associations
Use-Case Model
scope, goals, actors, features
Vision
Process Sale
Cashier
use case names
Process Sale
1. Customer arrives ...
2. Cashier makes new sale.
3. ...
terms, attributes, validation
Glossary
Require-ments
Use Case Diagram
Use Case Text
system events
: System
: Cashier
Operation:
enterItem(…)
Post-conditions:
- . . .
system operations
make NewSale()
enterItem (id, quantity)
Supplementary Specification
non-functional reqs, quality attributes
Operation Contracts
System Sequence Diagrams
requirements

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

Boundary Interface Properties

- Operations invoked across component/system boundaries often use simple types in their signatures
  - Primitive data types (int, list, str, …)
  - “Data classes”
- Careful not to expose implementation details
  - classes, data structures, storage systems

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.*

UC01 Make Booking Operations

- findRooms(in:Date,out:Date): list(RoomDescriptor)
- getTotalPrice(roomID:str, in:Date, out:Date): int
- createBooking(roomID:str, in:Date, out:Date,
  pd:PaymentDetails, cd:ContactDetails): str
- Need to define content of Payment- and ContactDetails
- Parameters should be named better

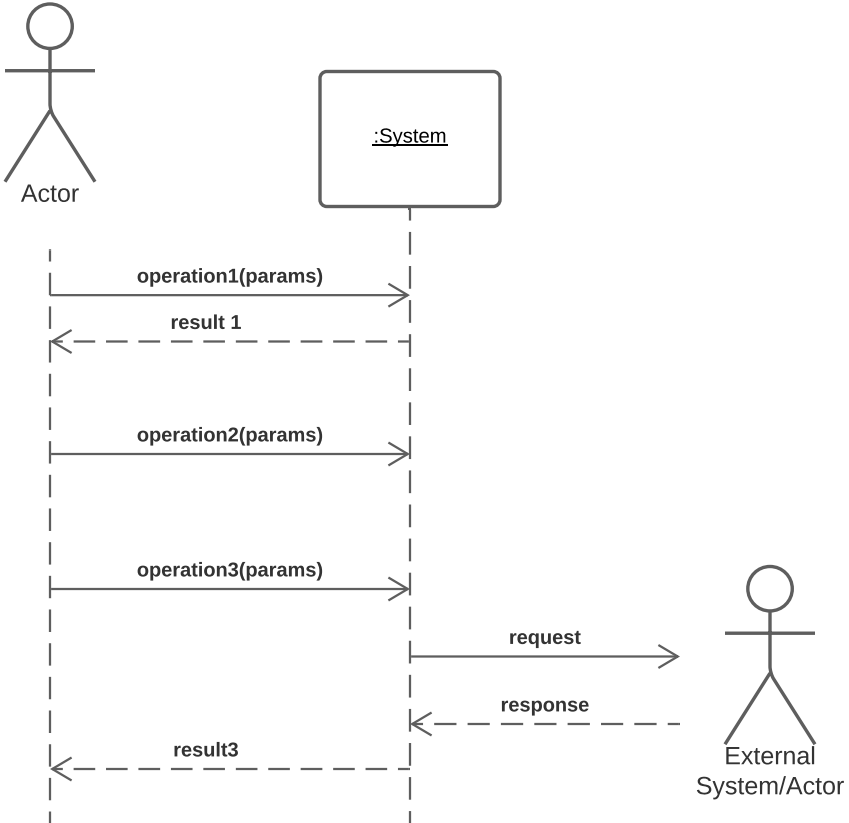
createBooking() Contract

- createBooking(roomID:str, in:Date, out:Date,
  pd:PaymentDetails, cd:ContactDetails): str
- Preconditions: none
- Postconditions:
  - Guest has been created
  - Booking has been created and associated with guest and room
  - PaymentConfirmation has been recorded
  - Guest has been notified

Task 2. System Sequence Diagram

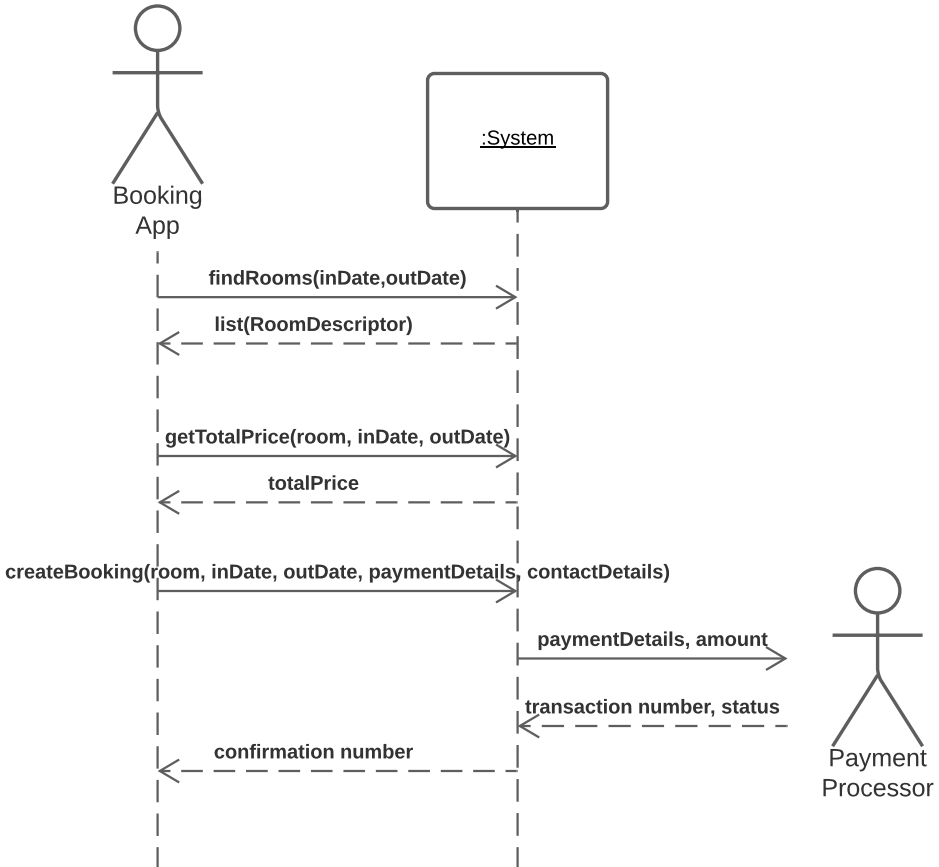
- Draw a System Sequence Diagram for UC01 Make Booking showing the sequence of system operations defined in Task 1.

System Sequence Diagram Example



SSD for Make Booking #1

Notification system not shown for brevity



RoomDescriptor

number:int
type: string
dailyRate: int
description: string

PaymentDetails

cardNumber:str
name:str
expiryDate:str
cvv:int

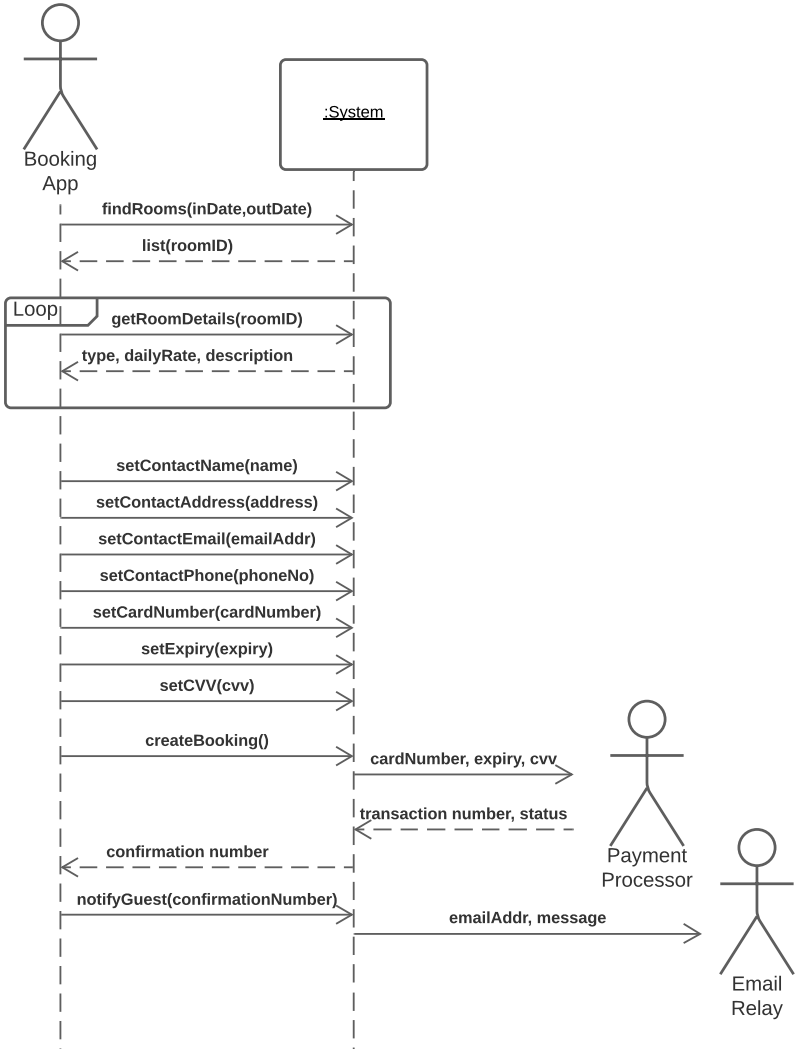
ContactDetails

name:str
address: str
phone: str
email: str

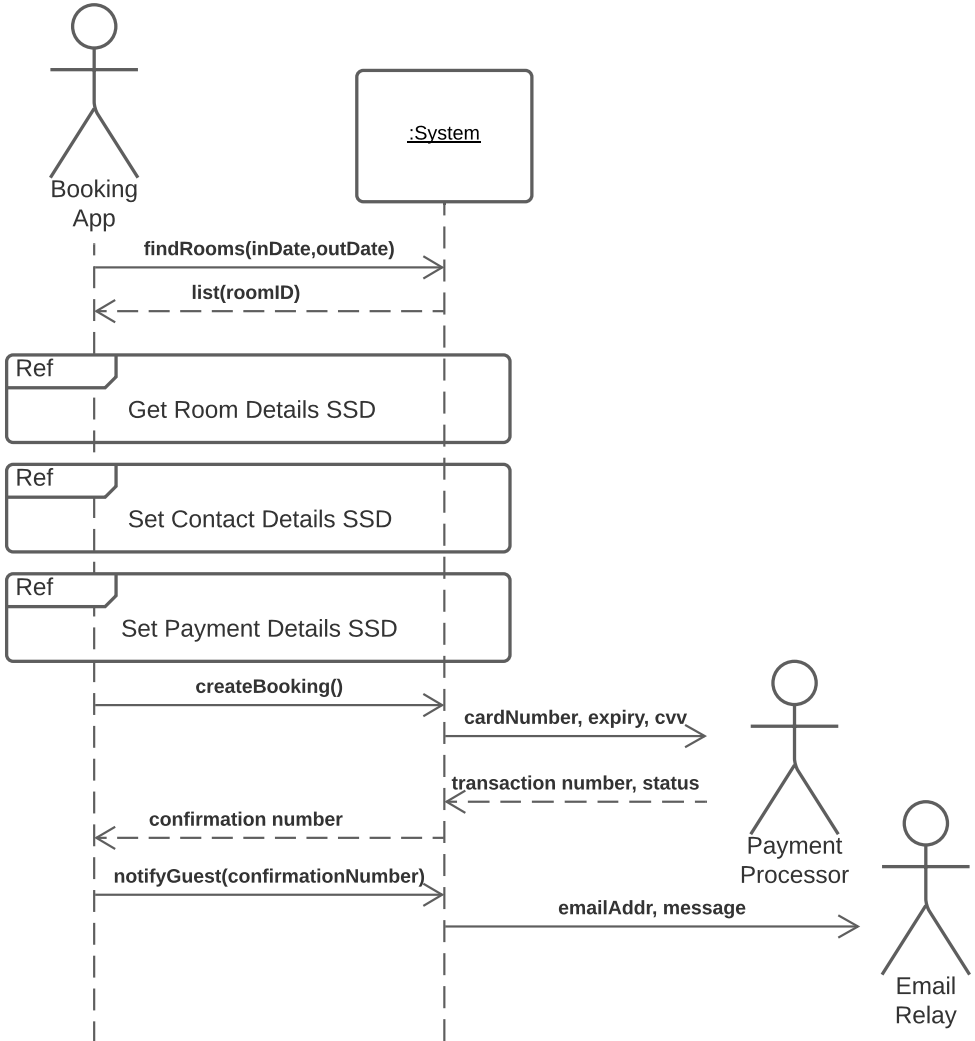
Task 3. Assess SSD Designs

- 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

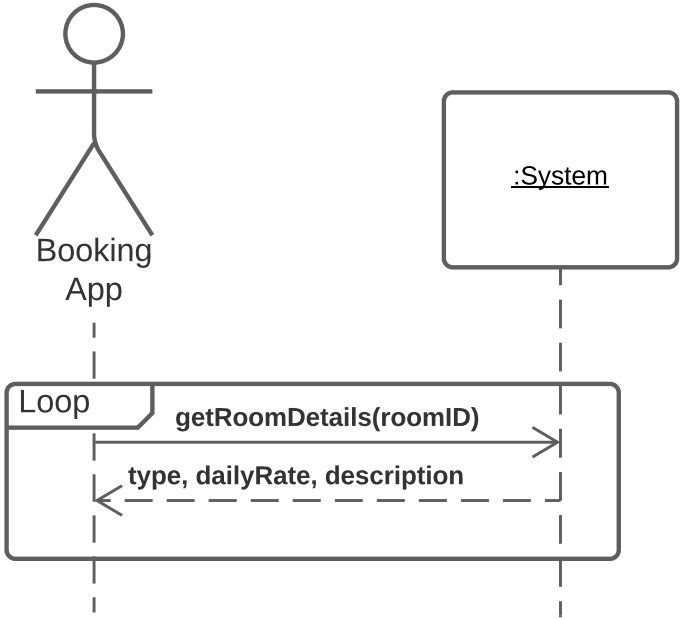


Make Booking SSD #2 Part A

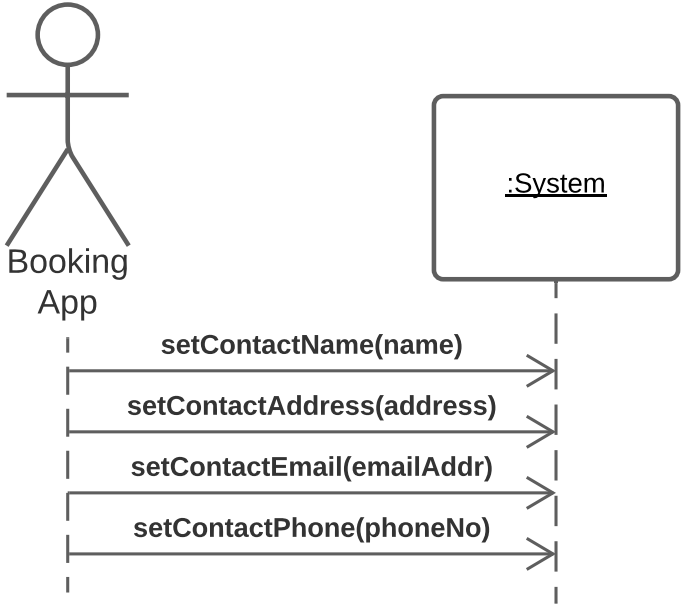


Get Room Details SSD #2 Part B

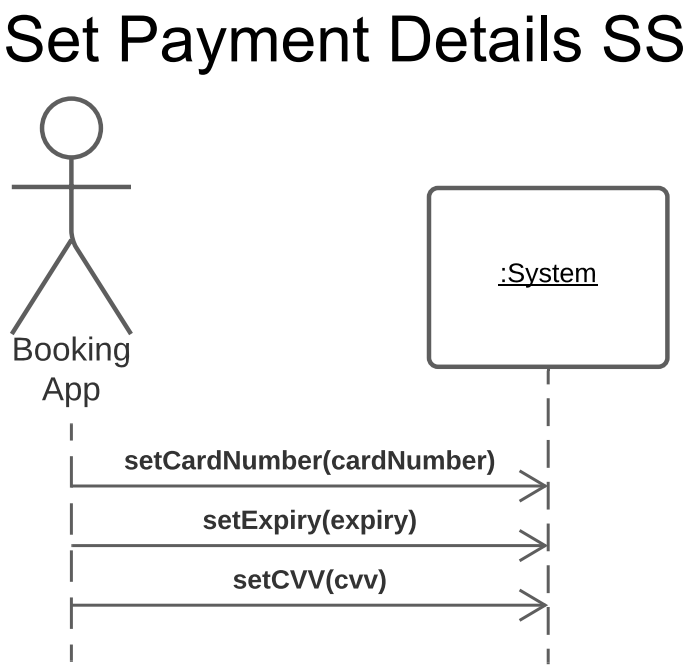
Get Room Details SSD



Set Contact Details SSD



Make Booking SSD #2 Part C



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)

Make Booking SSD #2 Issues #1

- The system does not know what room is being booked
- Business logic in the client
  - Complex protocol embedded in client logic
    - » First find, then loop, then set multiple things, then act
  - Notification should be automated.
    - » What if the client stops before notification is sent?

Make Booking SSD #2 Issues #2

- Information that belongs together split among multiple setter operations
  - Needlessly complicated
  - Difficult to generalise to other variations of the scenario
    - » PayPal payment does not require card details, CVV
    - » Registered users skip entering contact details
  - The system must store this somewhere until it is used
- Latency
  - Multiple calls needed to display list of rooms
  - Multiple calls needed to set basic information

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