University of
South Australia

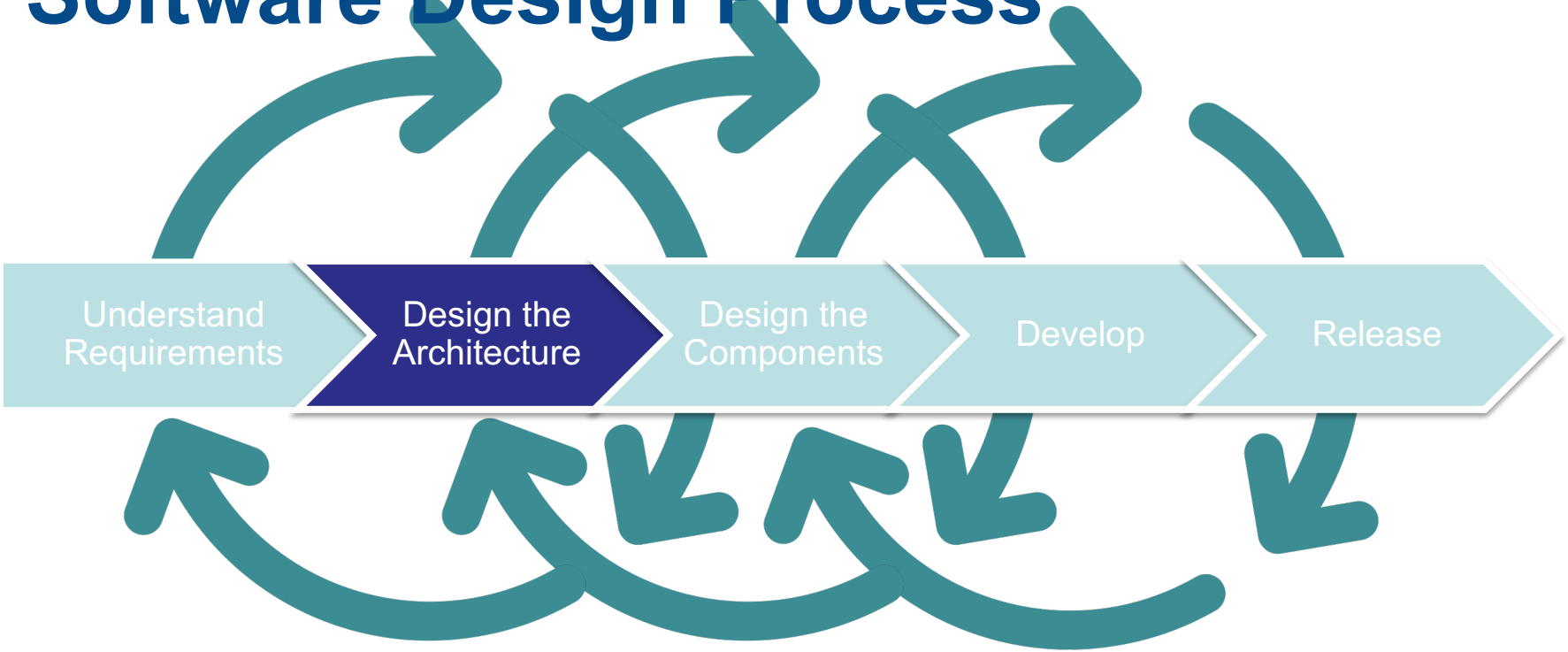
INFS 2044

Workshop 2a Answers

Preparation Already Done

- Watch the Week 2 Lecture
- Read the required readings for this week
- Revisit the volatilities for the *Booking System* discussed in Week 1
- Bring a copy of the workshop instructions (this document) to the workshop

Software Design Process



Learning Objectives

- Identify the main components of a system decomposition
- Assess the quality of system decompositions
- Validate system decompositions

Preparation. Revisit Volatilities

- Revisit the volatilities for the *Booking System* discussed in Week 1

Booking System Volatilities

- Users (guests, hotel managers, administrators, …)
- Items (rooms, function halls, catering, events, …)
- Notification methods (on screen, email, text, …)
- Payment providers (Paypal, Stripe, home-grown, …)
- Data stores (relational, distributed, noSQL, ...)
- Access channels (Web app, Dedicated client app, API, …)
- Pricing policies (taxes, discounts, fees, …)
- User enrolment & authentication (built-in, single sign on, external services)

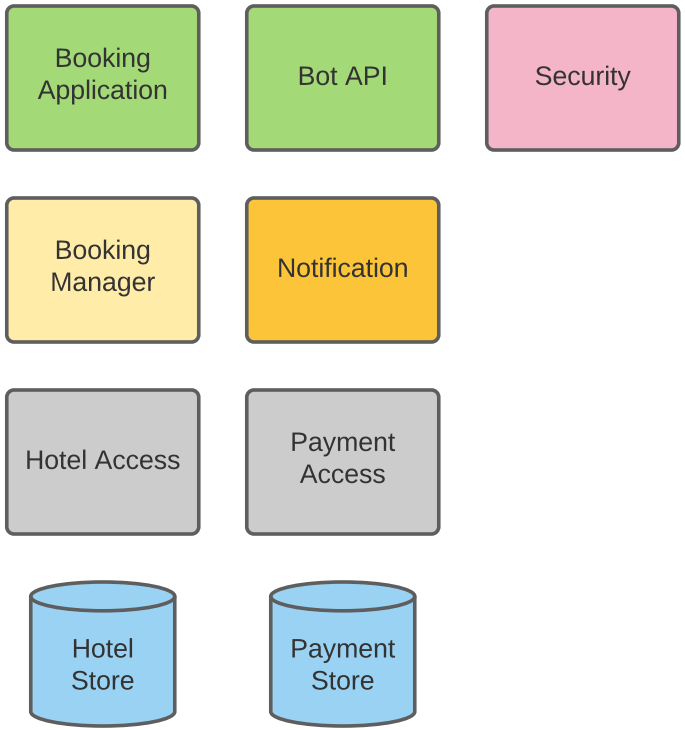
Task 1. Assess Decomposition

- Assess the quality of the candidate decompositions for the Booking System.
- Use design principles to assess which decompositions are preferred.
- Discuss advantages and disadvantages of each design.

Recall Design Principles

- Common Closure Principle
- Common Reuse Principle
- Acyclic Dependencies Principle
- Stable Dependencies Principle

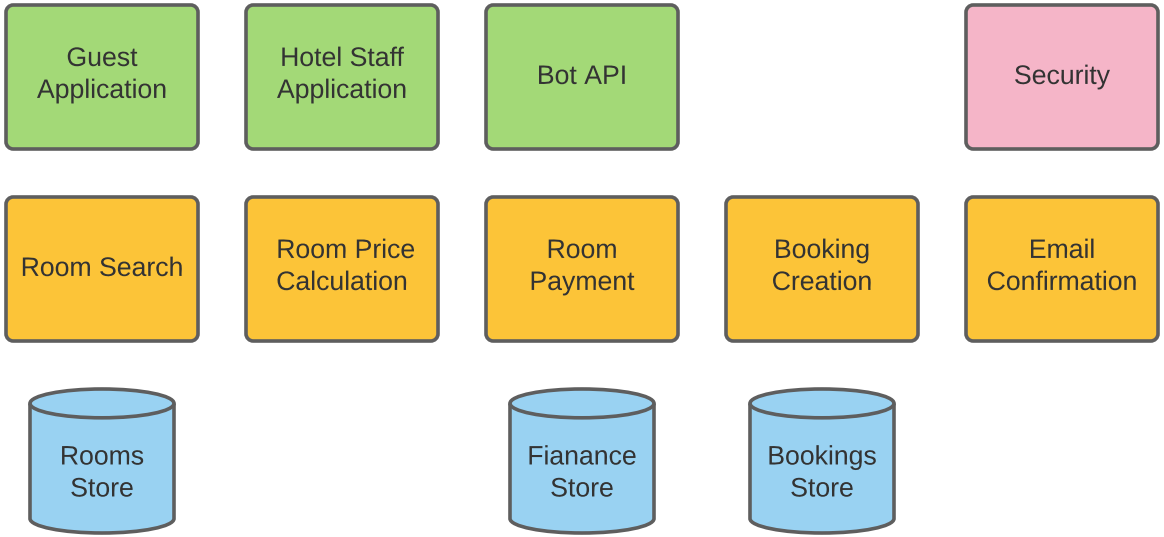
Booking System Decomposition A



Component Responsibilities A

Component	Responsibilities
Booking Application	Present the user interfaces for Guest and Staff
Bot API	Offer functions for external booking agents
Booking Manager	Orchestrate the use cases Find rooms matching given criteria Calculate total price for a booking request
Payment Access	Verify payment Record payment confirmation
Notification	Notify guests of booking confirmation
Hotel Access	Retrieve hotel and room details Record bookings
Security	User authentication & authorization

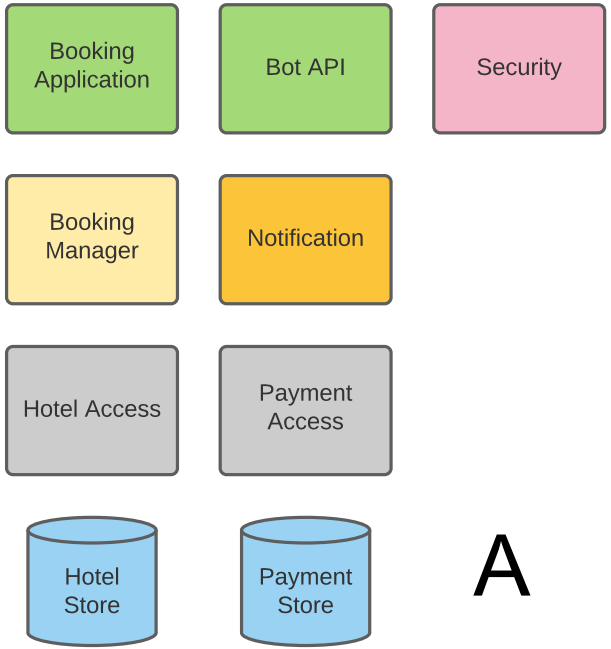
Booking System Decomposition B

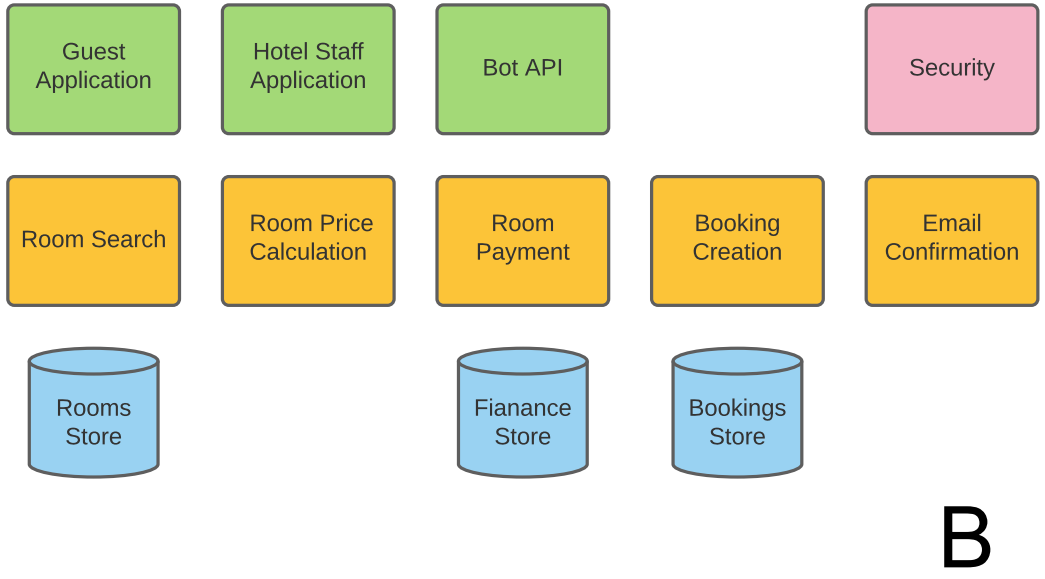


Component Responsibilities B

Component	Responsibilities
Guest- & Hotel Applications	Present the user interfaces Orchestrate the booking use cases
Bot API	Offer functions for external booking agents Orchestrate the booking use cases
Room Search	Find rooms matching given criteria
Room Pricing	Calculate total price for a booking request
Room Payment	Verify payment Record payment confirmation
Booking Creation	Create booking
Email Confirmation	Email booking confirmation to clients
Security	User authentication & authorization

Which Decomposition is Better?





Both Decomposition are Poor

- Decomposition A
  - Does not encapsulate volatility of user, pricing calculation, payment verification
  - Violates common closure principle:
  - Payment Access is responsible for storing payment details
  - Hotel Access stores both hotel/room and booking information
  - Components have more than one reason to change

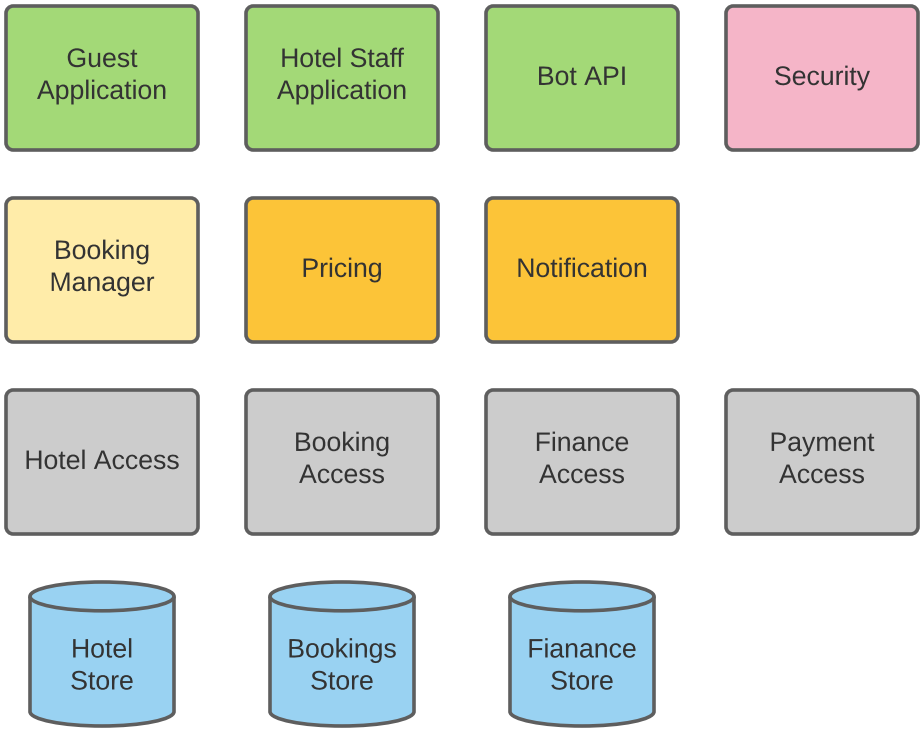
Both Decomposition are Poor

- Decomposition B
  - Does not encapsulate volatility of data access, payment verification, and notification
  - Violates common closure principle:
  - Business logic in three separate client applications
  - Booking creation is also responsible for payment details
  - Components have more than one reason to change

Task 2. Create a Better Decomposition

- Create a decomposition for the *Booking System* that accounts for the identified volatilities.
- Show how the volatilities map to components.
- Assess the quality of the decomposition.
- Does it isolate change and promote evolution and reuse?

Booking System Decomposition C



Component Responsibilities C

Component	Responsibilities
Guest- & Hotel Applications	Present the user interfaces
Bot API	Offer functions for external booking agents
BookingManager	Orchestrate the booking use cases
Pricing	Calculate the price for a booking request
Notification	Email booking confirmation to clients
Security	User authentication & authorization
Hotel Access	Retrieve room and hotel information
Booking Access	Access booking information
Finance Access	Access financial transaction details
Payment Access	Access payment verification services

Volatility Map

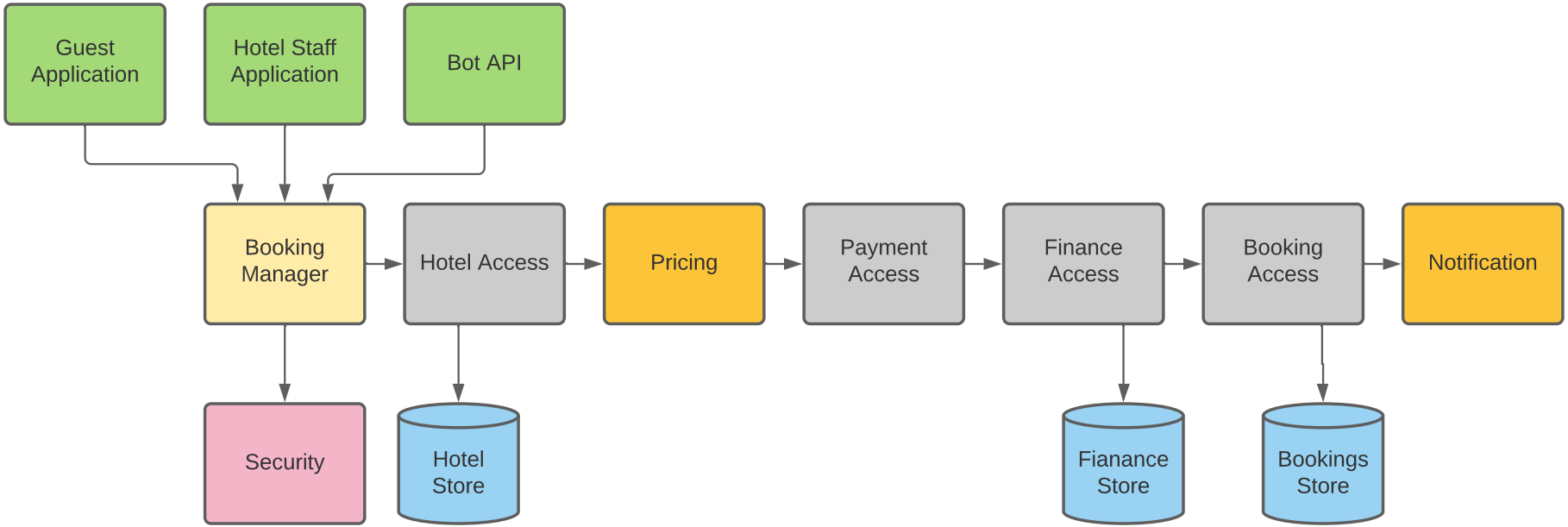
Volatility	Encapsulated in
User & Access Channels	Guest- & Hotel Applications Bot API
Items	BookingManager & Hotel Access
Pricing policies	Pricing
Notification	Notification
User authentication	Security
Data store	Hotel Access
Data store	Booking Access
Data store	Finance Access
Payment provider	Payment Access

Task 3: Composition Assessment

- Assess the quality of the composition for the use case *Make Booking* given on the following slide.

- Identify strengths and weaknesses of this decomposition.

Booking System Composition A



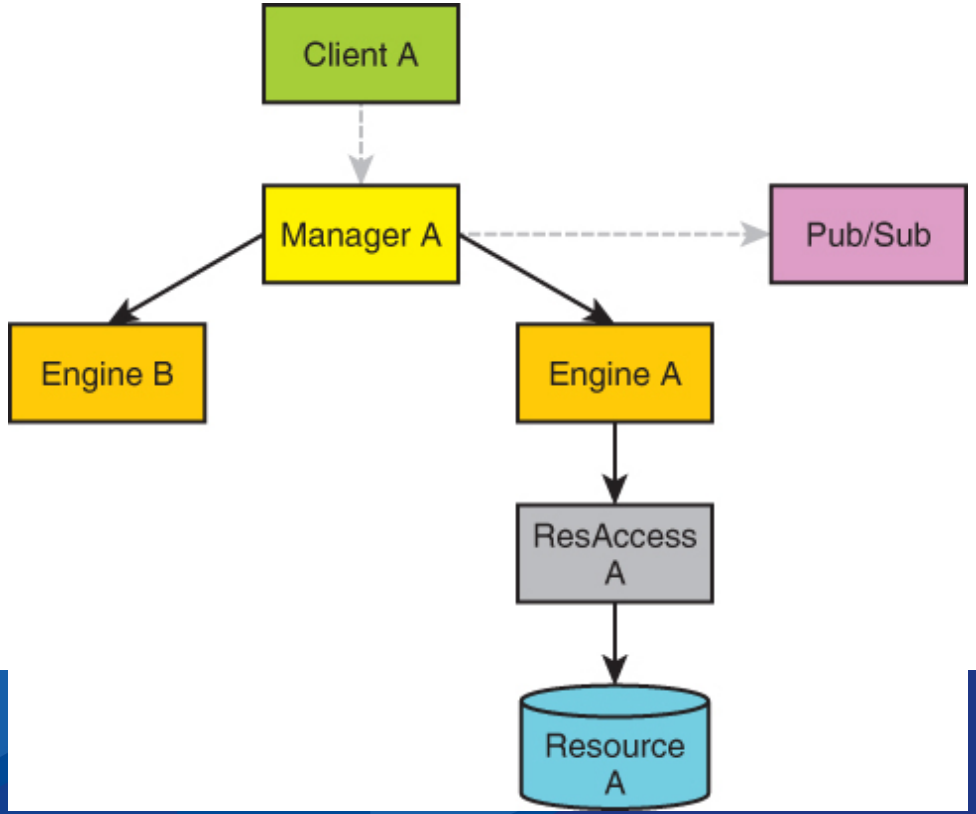
Discussion

- This is an example of Temporal Decomposition
- Introduces many dependencies
- Changes in components propagate to dependent components
  - Error handling and compensation
- Complexity hampers evolution

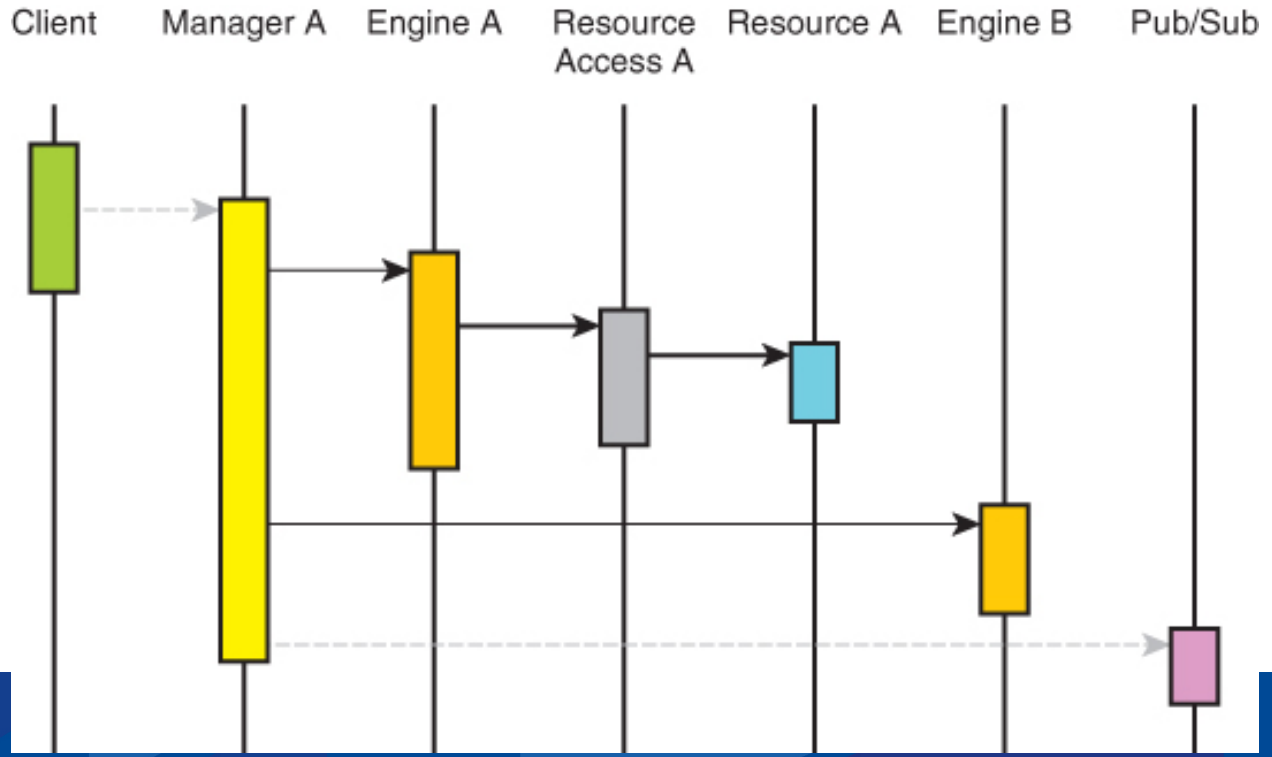
Task 4: Architecture Validation

- Validate the architecture by creating a Communication Diagram or a Sequence Diagram for use case *Make Booking*

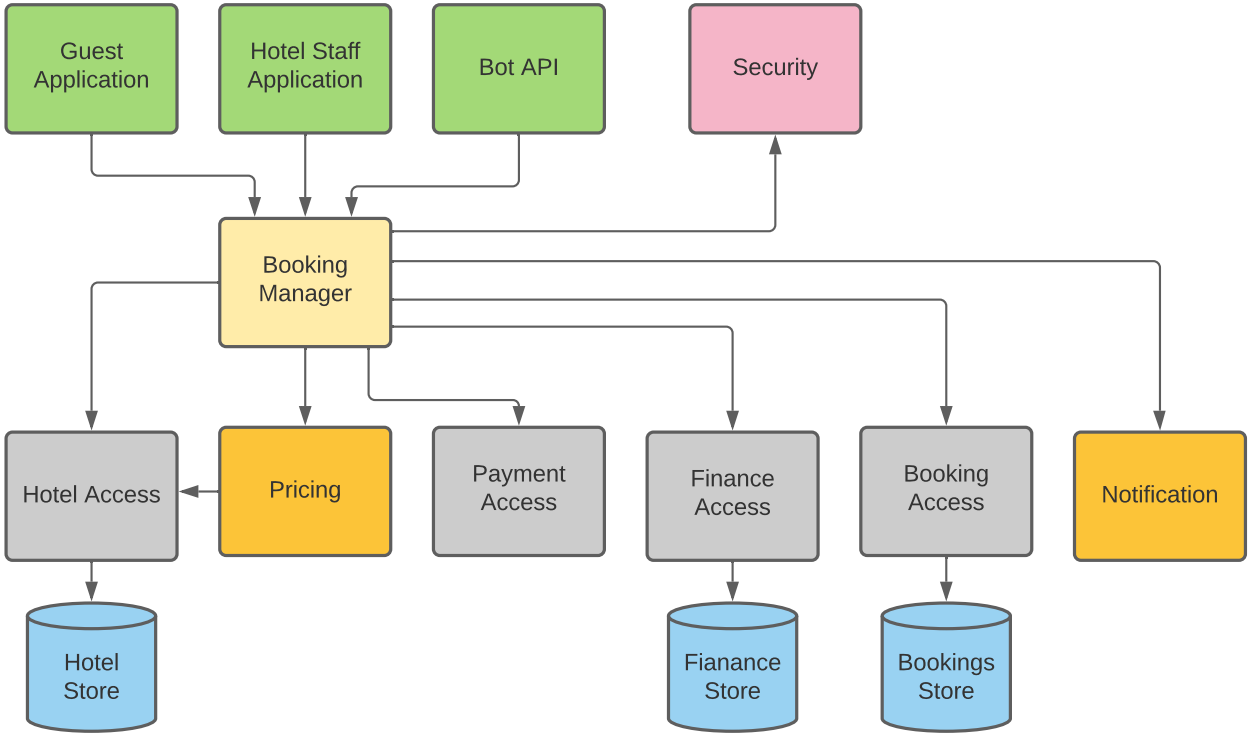
Communication Diagram Example



Sequence Diagram Example



Comm. Diagram for Make Booking



You Should Know

- Define the system components based on volatility
- Apply design principles to assess the quality of system decompositions
- Validate system decompositions using interaction diagrams

Activities this Week

- Attend second Workshop session
- Complete Quiz 2

University of
South Australia