Building Software Systems

Lecture 1.3

Software Architecture Documentation

SAURABH SRIVASTAVA
ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
IIT (ISM) DHANBAD

Revision - What is Software Architecture?

There are actually many definitions for Software Architecture

- There is, in fact a huge collection of them that you can refer to [1]
- For our purpose, let us pick a definition from a book
- *“The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.”*
  – [Software Architecture in Practice, Len Bass, Paul Clements and Rick Kazman**,** 3rd Edition, Chapter 1]
- To put it in simple words, you need to look at “smaller structures” that can come together to form …
- … a system that can provide the required functionality with expected quality under the given constraints
- You also need to consider how these smaller structure behave with each other in a cooperative fashion …
- … meaning that you will be required to depict their inter-relationships

Revision - What is Software Architecture?

There are actually many definitions for Software Architecture

- There is, in fact a huge collection of them that you can refer to [1]
- For our purpose, let us pick a definition from a book
- *"The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both."*
  – [Software Architecture in Practice, Len Bass, Paul Clements and Rick Kazman**,** 3rd Edition, Chapter 1]
- To put it in simple words, you need to look at "smaller structures" that can come together to form …
- … a system that can provide the required functionality with expected quality under the given constraints
- You also need to consider how these smaller structure behave with each other in a cooperative fashion …
- … meaning that you will be required to depict their inter-relationships

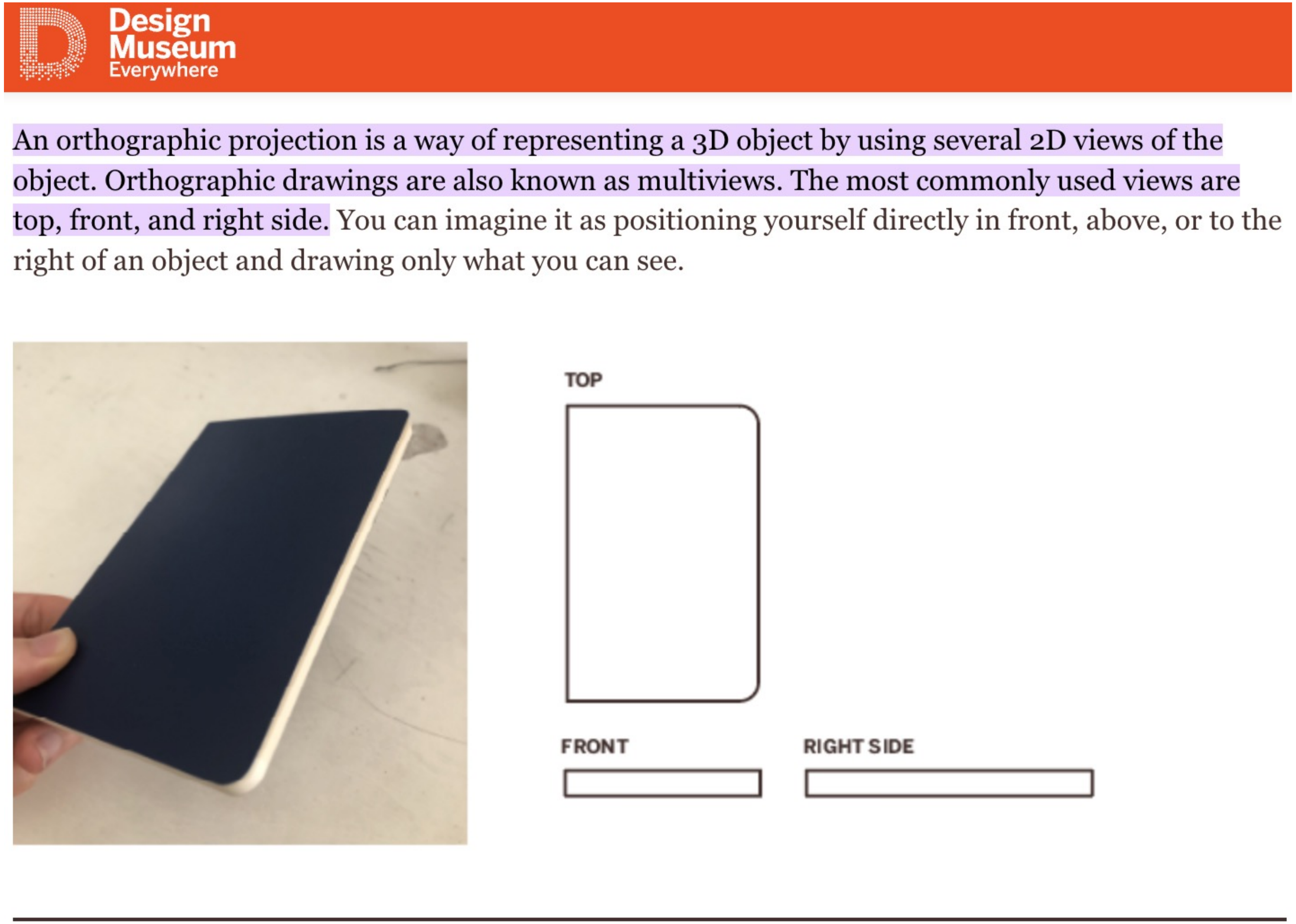
We left things here in the last lecture, and moved swiftly to Quality Attributes

- Let us now look at the idea of the architecture of a software

Architectural Views

Did you take a course in Engineering Drawing at the start of your Bachelor Program?

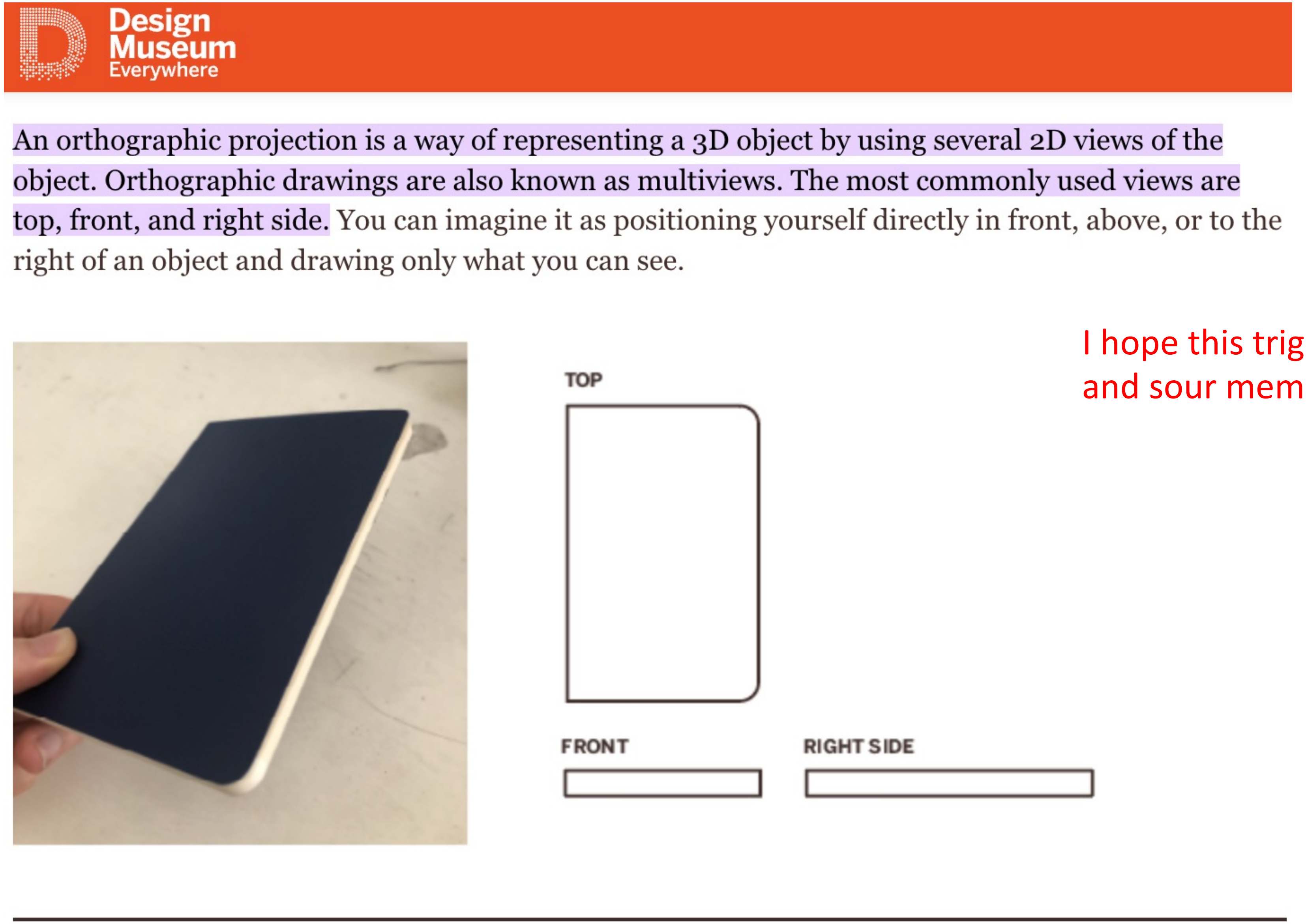
- If you did, you would know a particular term very well – *projections*



Design Museum Everywhere

An orthographic projection is a way of representing a 3D object by using several 2D views of the object. Orthographic drawings are also known as multiviews. The most commonly used views are top, front, and right side. You can imagine it as positioning yourself directly in front, above, or to the right of an object and drawing only what you can see.

Source: [Orthographic Projection (Design Museum)]()



An orthographic projection is a way of representing a 3D object by using several 2D views of the object. Orthographic drawings are also known as multiviews. The most commonly used views are top, front, and right side. You can imagine it as positioning yourself directly in front, above, or to the right of an object and drawing only what you can see.

I hope this triggered some sweet and sour memories in your mind :D

Source: [Orthographic Projection (Design Museum)]

Architectural Views

Designing the Architecture of a Software Solution requires years of experience and practice

- Usually, there is a role called *Technical Architect* in an IT firm that does this job

Did you take a course in Engineering Drawing at the start of your Bachelor Program?

- If you did, you would know a particular term very well – *projections*
- A projection helps you get an idea of how a particular object looks like, from a particular viewpoint

Architectures of Software Solutions can also be viewed from multiple “viewpoints”

- There are several “Viewsets” discussed in literature related to Software Architecture
- A Viewset is a collection of some Views, where each View represents one particular aspect of the solution

The *4+1 Viewset* is a model proposed by Philippe Kruchten in 1995

- While probably not the best, it is a compact enough viewset to reason about a system

4+1 Viewset (1/3)

The Viewset says that in order to reason about a software, you should create four views

Logical View

- Represents the functional requirements of the system
- Concerned with addressing the functional requirements and typically includes class diagrams …
- … and other UML diagrams to represent key abstractions in the system

Process View

- Deals with the dynamic aspects of the system
- Explains the system processes and how they communicate
- Focuses on the run-time behaviour of the system
- Arguably the most important view of all, it depicts concurrency and distribution aspects of the system

4+1 Viewset (2/3)

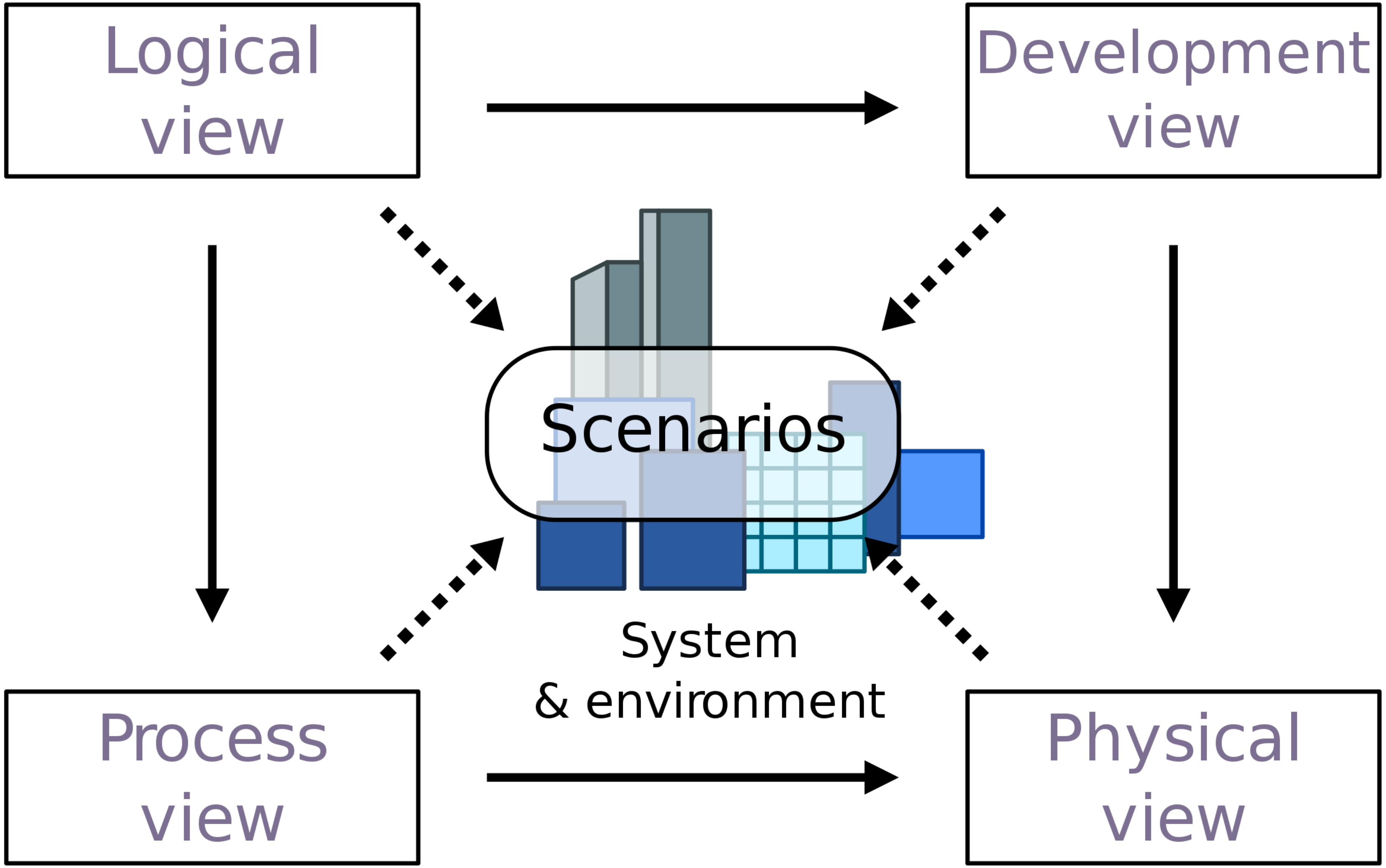
The Viewset says that in order to reason about a software, you should create four views

Development View

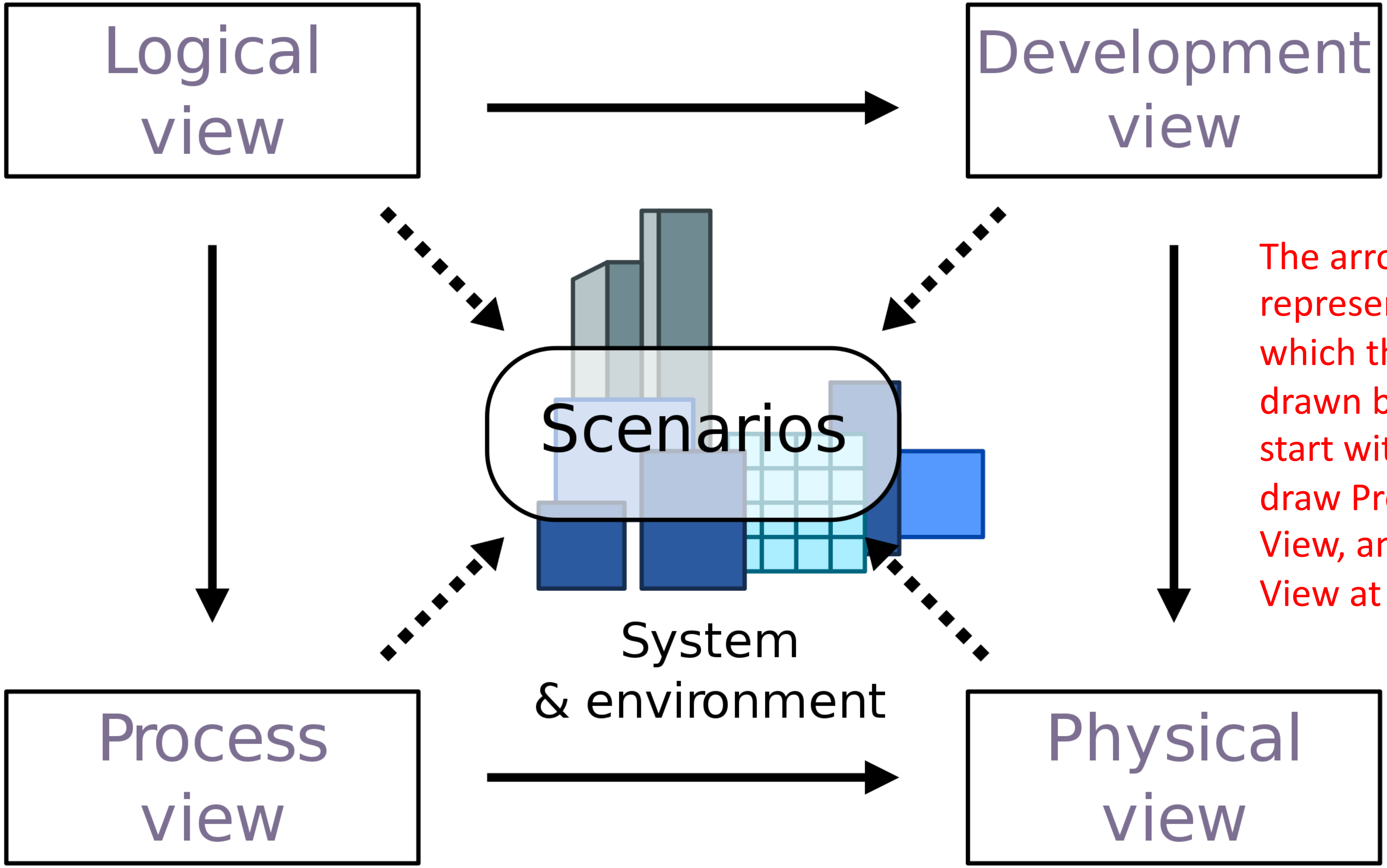
- Represents the system from a programmer's perspective
- Essentially shows the programming units (e.g., packages or libraries) involved in building the system
- Can also be used show allocations of these units to various developers/teams

Physical View

- Represents the of the system from the perspective of the system administrator
- Shows the involved hardware components, network architectures and topology of the system



Source: [4+1 architectural view model - Wikipedia]



The arrows intend to represent the sequence, in which these views should be drawn by the Architect (e.g., start with Logical View, then draw Process or Development View, and draw the Physical View at the end)

Source: 4+1 architectural view model - Wikipedia

4+1 Viewset (3/3)

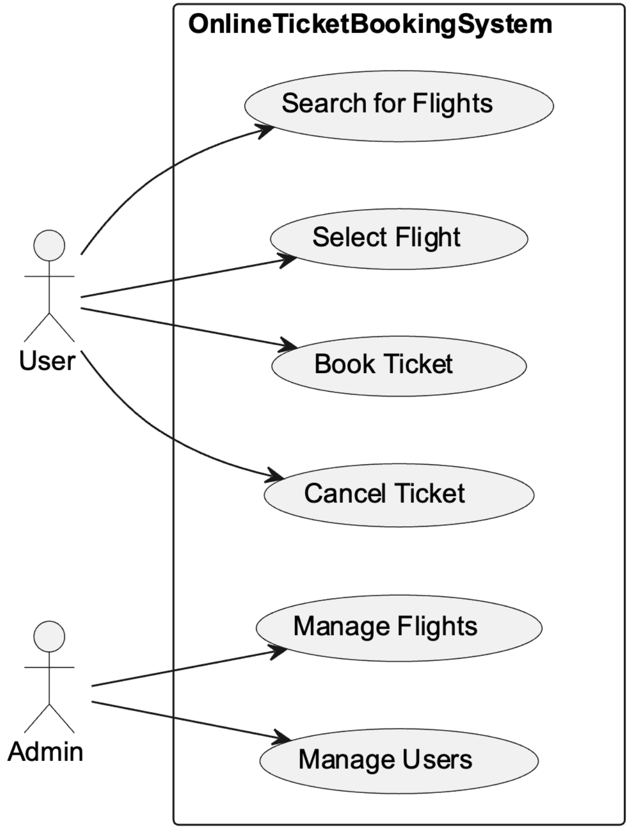
Apart from the four views, there is another aspect of the system that the Architect should document

Scenarios or Use Cases

- Scenarios or Use Cases are similar to User Stories – they depict an end-to-end usage scenario of the system
- Scenarios are helpful towards understanding the original requirements for which the system was architected

Example – Online Ticket Booking System

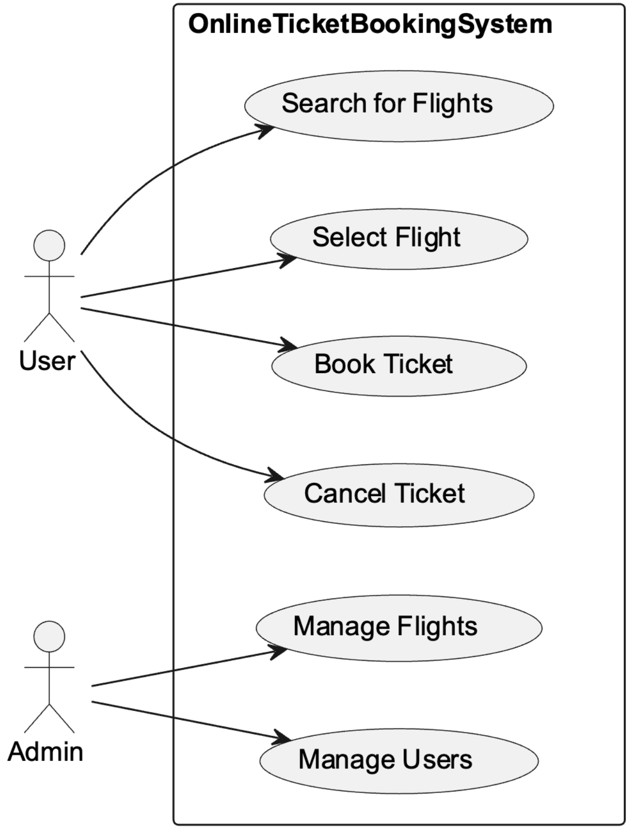
Scenarios



Generated by ChatUML | AI Assisted Diagram Generator

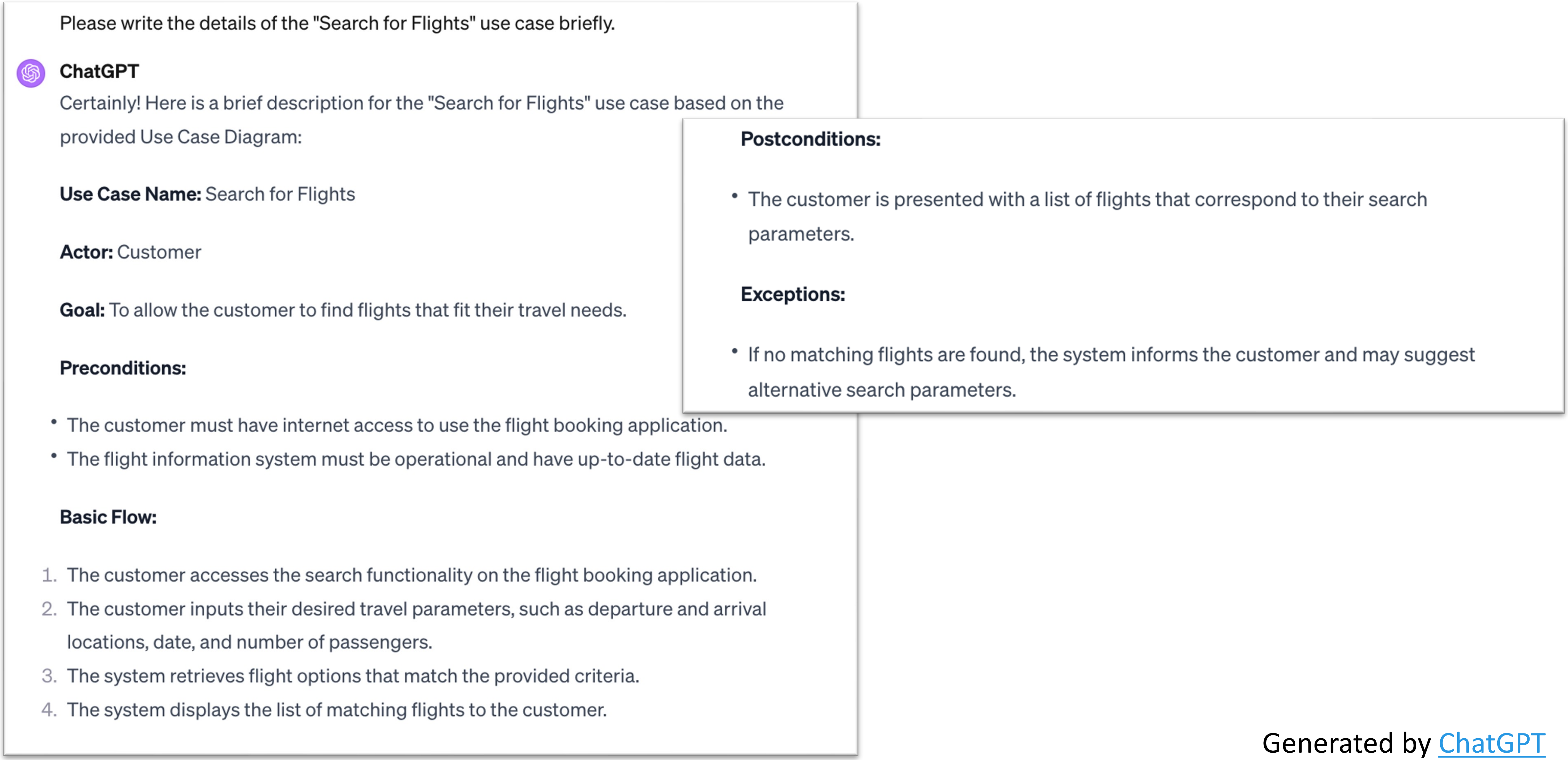
Example – Online Ticket Booking System

Scenarios



Each of the given Scenario may be explained separately in textual form or through a UML diagram called the Use

Generated by [ChatUML | AI Assisted Diagram Generator]()



Please write the details of the "Search for Flights" use case briefly.

ChatGPT

Certainly! Here is a brief description for the "Search for Flights" use case based on the provided Use Case Diagram:

Use Case Name: Search for Flights

Actor: Customer

Goal: To allow the customer to find flights that fit their travel needs.

Preconditions:

- The customer must have internet access to use the flight booking application.
- The flight information system must be operational and have up-to-date flight data.

Basic Flow:

1. The customer accesses the search functionality on the flight booking application.
2. The customer inputs their desired travel parameters, such as departure and arrival locations, date, and number of passengers.
3. The system retrieves flight options that match the provided criteria.
4. The system displays the list of matching flights to the customer.

Postconditions:

- The customer is presented with a list of flights that correspond to their search parameters.

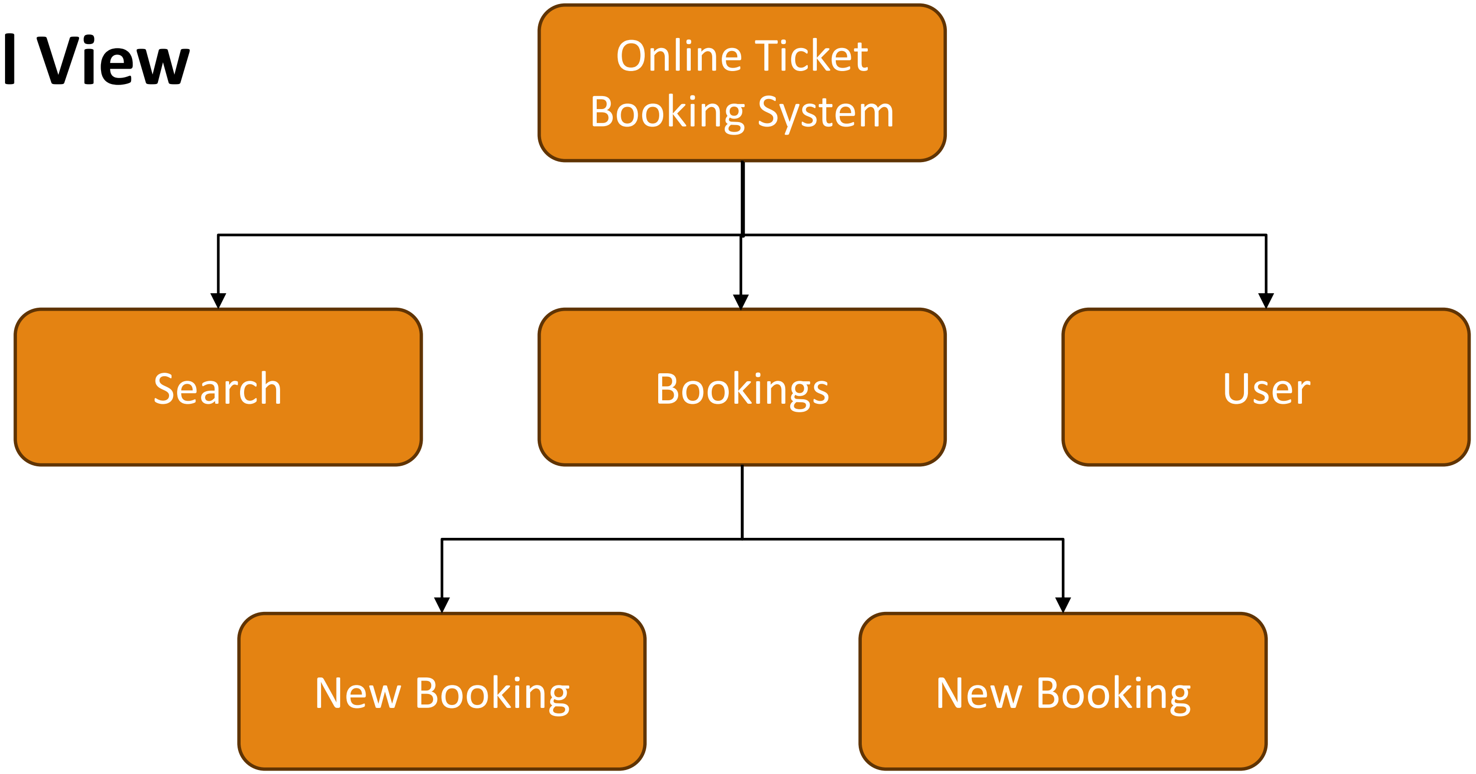
Exceptions:

- If no matching flights are found, the system informs the customer and may suggest alternative search parameters.

Generated by ChatGPT

Example – Online Ticket Booking System

Logical View



Online Ticket Booking System

Search

Bookings

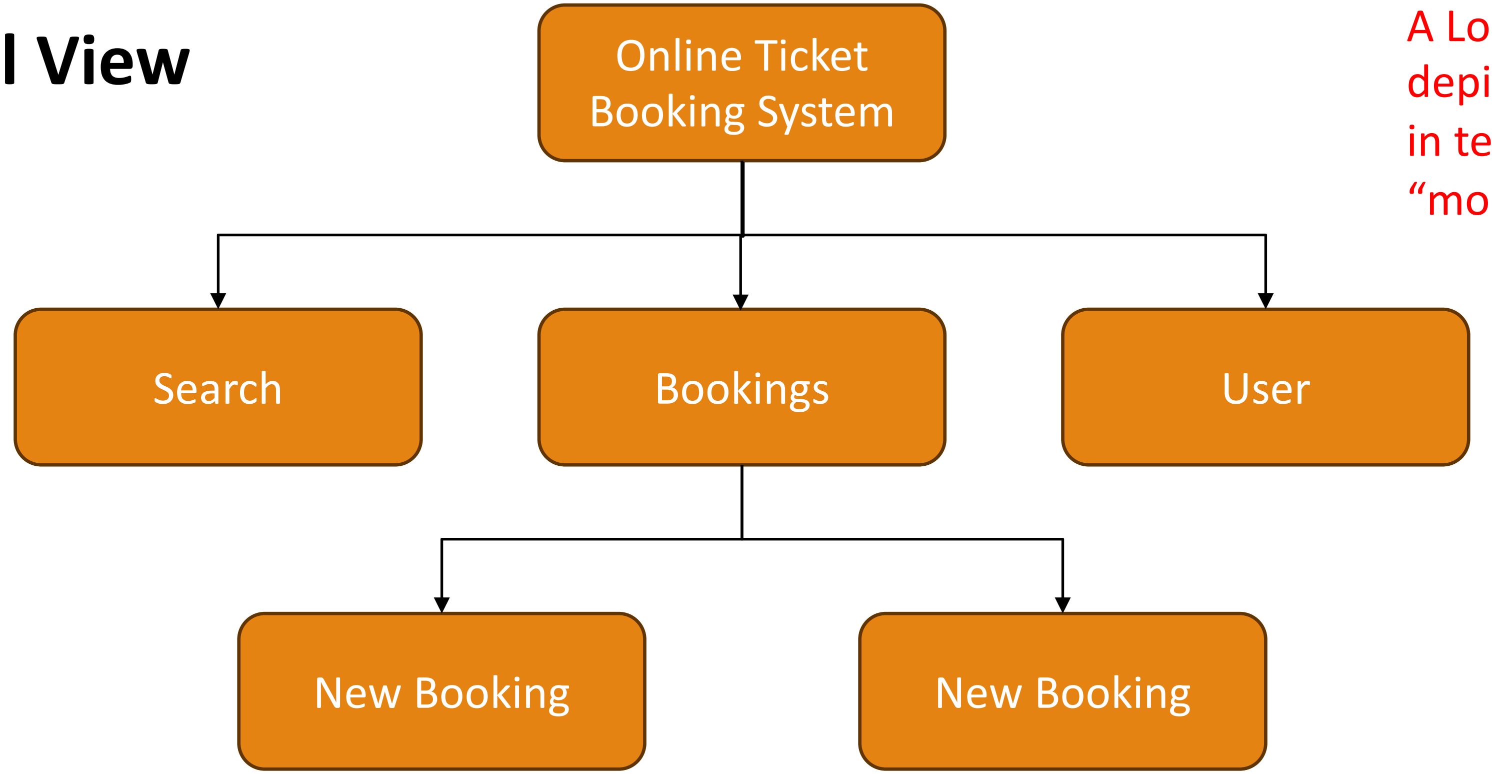
User

New Booking

New Booking

Example – Online Ticket Booking System

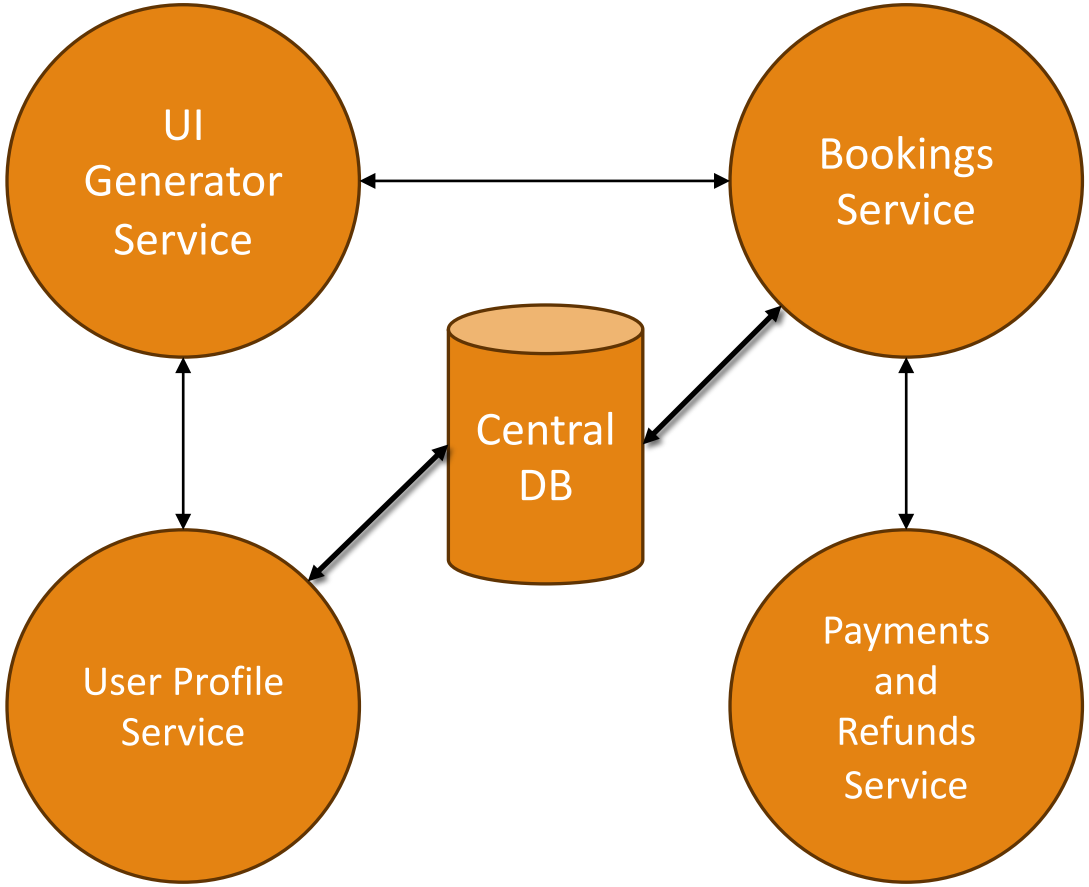
Logical View



A Logical View essentially depicts a view of the system in terms of its constituent “modules”

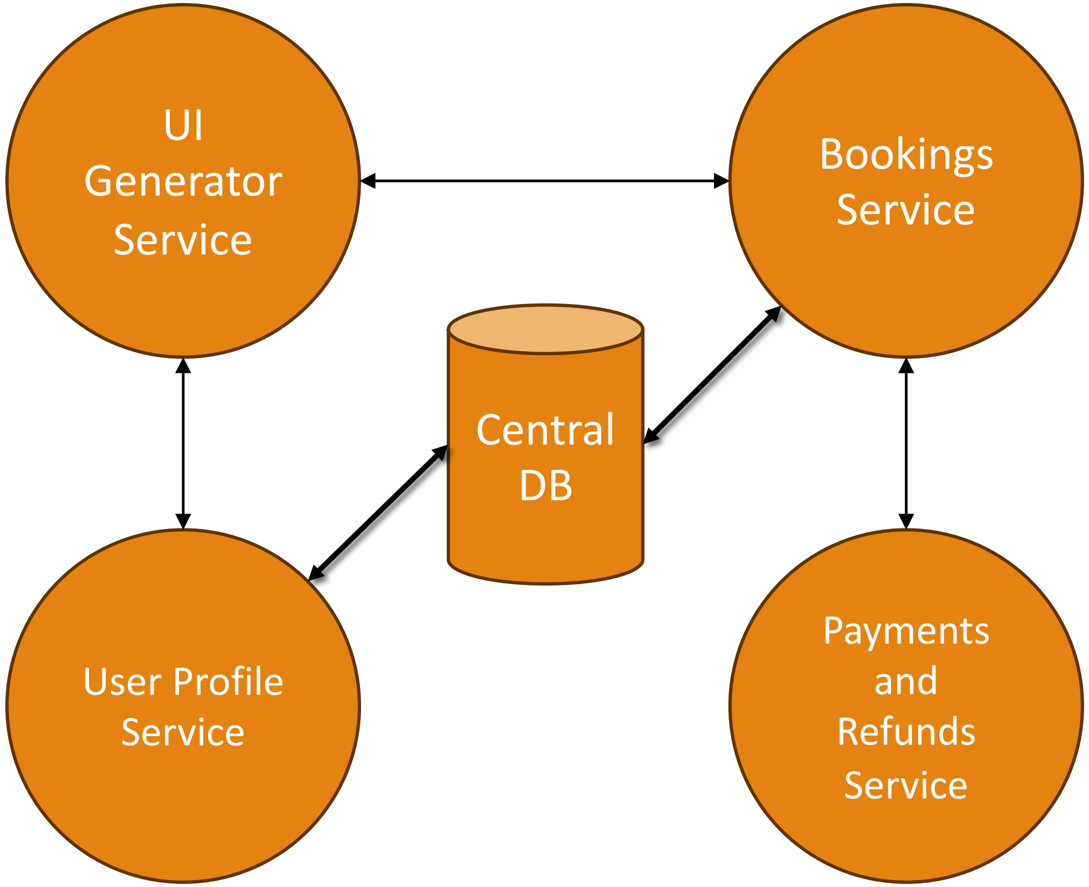
Example – Online Ticket Booking System

Process View



Example – Online Ticket Booking System

Process View



A Process View essentially depicts a view of the system in terms of its run-time entities or “components”

How many views shall we draw?

You may draw "as many views" of the architecture as you want

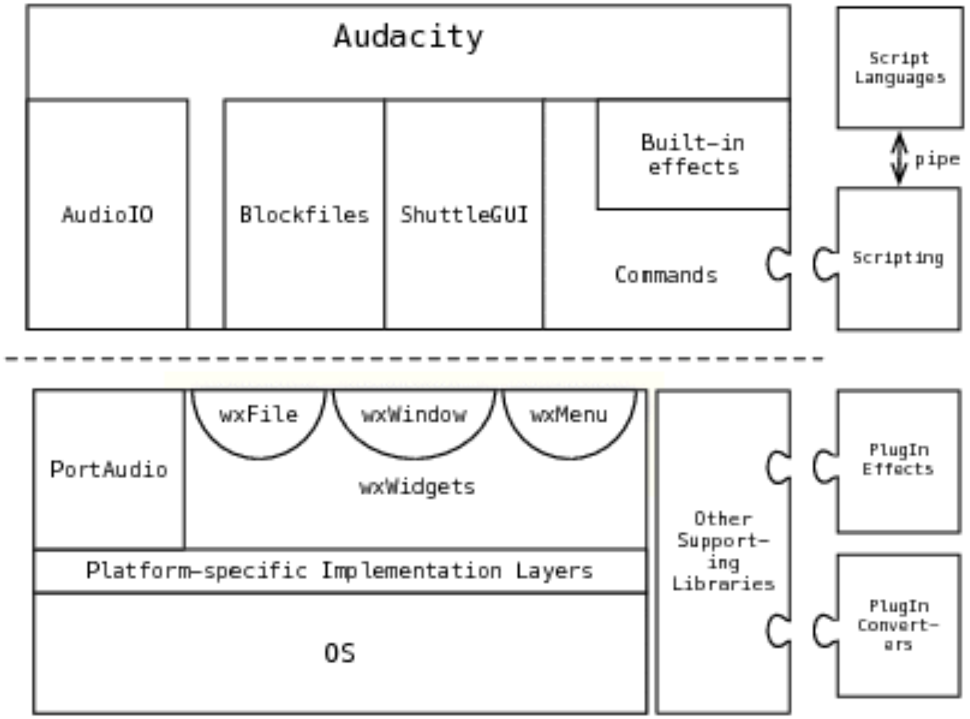
- However, as you might have already realised, it requires substantial effort to do so for a large system
- In case an architect is overburdened, it may not be possible to draw all the views in detail …
- … as it may require significant amount of time

We can optimise our documentation, if we create some "hybrid diagrams"

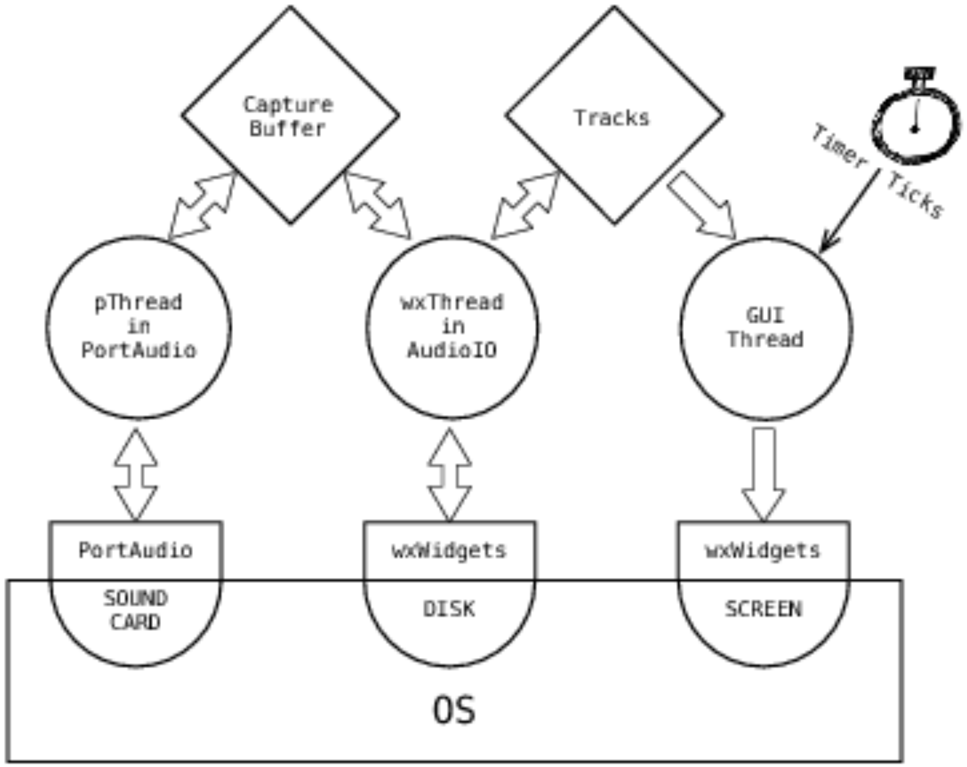
- For example, after drawing the Logical View, if we can put some annotations to depict packages/scripts …
- … through which these modules will be implemented, it also covers most of the aspects of Development View
- Similarly, if we also show the deployment hardware (real or virtual) for each component in the Process View …
- … it provides most of the information that the Physical View will provide
- Thus, it may be sufficient in most cases to draw the Logical and Physical Views with some added information
- Scenarios may be replaced by User Stories

In reality, architects may draw views which are a "combination of everything"

Example Architecture Diagrams – *Audacity*



Layers in Audacity



Threads and Buffers in Playback and Recording

Source: [The Architecture of Open Source Applications (Volume 1) - Audacity]

Homework

Check out the original 4+1 View paper by Philippe Kruchten

- You may refer this link: https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf
- It is fine if you feel confused – I do not personally like the paper too; but the idea of a Viewset is what is the major takeaway of the paper