



Full Stack Application Development

BITS Pilani



Module 4: Server Side : Implementing Web Services



BITS Pilani
Pilani Campus



Introduction to API



What exactly an API is?

API stands for application programming interface

- ✓ are the little pieces of code that make it possible for digital devices, software applications, and data servers to talk with each other
- ✓ are the essential backbone of so many services we now rely on!

An easy way to understand the definition of an API is to think about the applications that you use every day

In an internet-connected world, web and mobile applications are designed for humans to use

- ✓ while APIs are designed for other digital systems and applications to use

Question



Websites and APIs both do the same things, like return data, content, images, video, and other information.

Then what are the differences?

API vs Website



- ❑ **Purpose:** Websites are typically designed for human interaction, providing a user interface for users to interact with content or services. APIs, on the other hand, are designed for programmatic interaction between software applications.
- ❑ **Access:** Websites are accessed through web browsers by users, while APIs are accessed programmatically by software applications.
- ❑ **Content:** Websites primarily serve content directly to users, while APIs primarily serve data or functionality to other software applications.
- ❑ **Interaction:** Websites require user interaction through a graphical user interface (GUI), while APIs allow software applications to interact with each other programmatically, often through HTTP requests.

What's an API



An API is a software intermediary that allows two applications to talk to each other

- ✓ API is the messenger that delivers your request to the provider that you're requesting it from and then delivers the response back to you

An API defines functionalities that are independent of their respective implementations

- ✓ which allows those implementations and definitions to vary without compromising each other
- ✓ Therefore, a good API makes it easier to develop a program by providing the building blocks

APIs enable developers to make repetitive yet complex processes highly reusable with a little bit of code

- ✓ The speed that APIs enable developers to build out apps is crucial to the current pace of application development

Developers are now much more productive than they were before when they had to write a lot of code from scratch

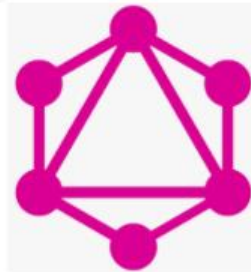
- ✓ With an API they don't have to reinvent the wheel every time they write a new program
- ✓ Instead, they can focus on the unique proposition of their applications while outsourcing all of the commodity functionality to APIs

API Paradigm (2)

- Over the years, multiple API paradigms have emerged such as

- ✓ REST
- ✓ RPC
- ✓ GraphQL
- ✓ WebHooks
- ✓ and WebSockets

{ REST }



- Broadly can be classified as

- ✓ Request-Response APIs
- ✓ Event Driven APIs

gRPC



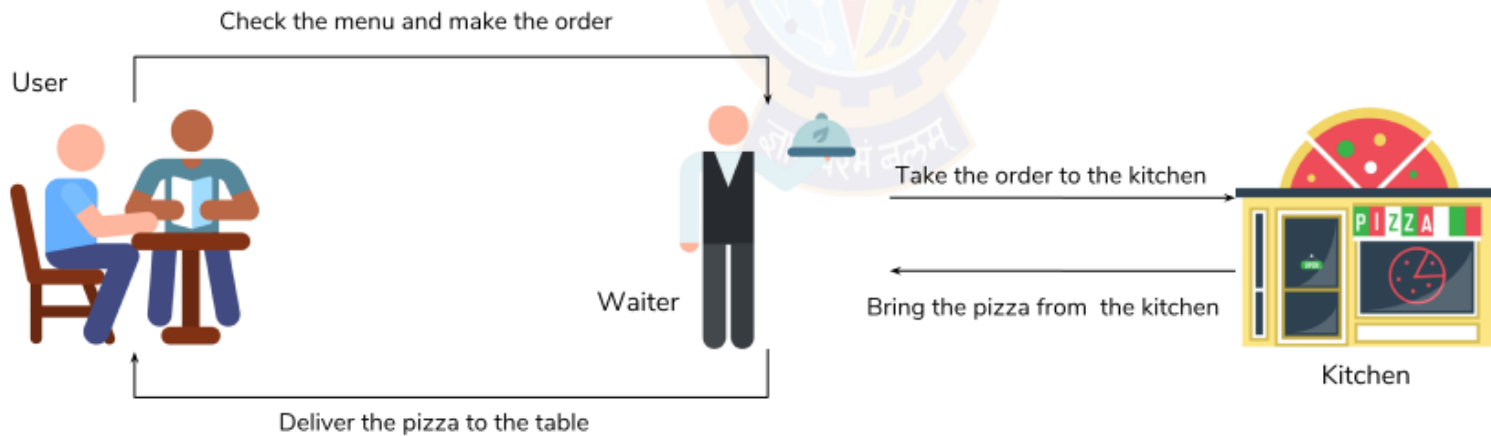
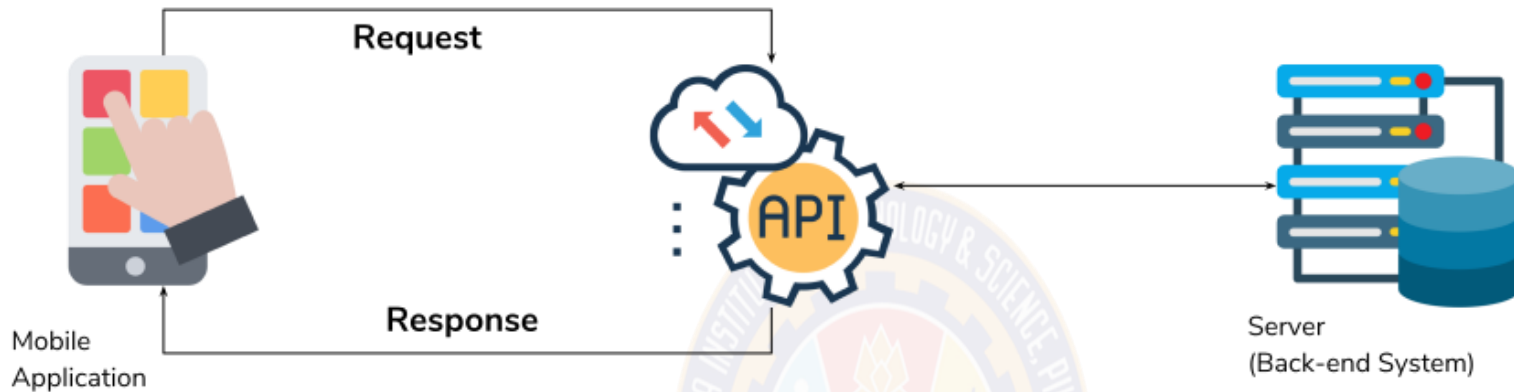
Request-Response APIs

- Typically expose an interface through a HTTP-based web server
- APIs define a set of endpoints
 - ✓ Clients make HTTP requests for data to those endpoints
 - ✓ Server returns responses
- The response is typically sent back as JSON or XML
- Three common paradigms used to expose request–response APIs:
 - ✓ REST
 - ✓ RPC
 - ✓ and GraphQL

Event Driven APIs

- Apps which want to stay up to date with the changes in data on server side often end up polling the API
- With polling, apps constantly query API endpoints at a predetermined frequency and look for new data
 - ✓ If poll is done at a low frequency, apps will not have data about all the events happened since last poll
 - ✓ If poll is done at a high frequency, would lead to a huge waste of resources, as most API calls will not return any new data
- To share data about events in real time,
 - ✓ three common mechanisms are used :
 - ❖ WebHooks
 - ❖ WebSockets
 - ❖ and HTTP Streaming

Working of API



Working of API

The customer, a waiter, and a restaurant kitchen!

- APIs work by sharing data and information between applications, systems, and devices—making it possible for these things to talk with each other
- Sometimes the easiest way to think about APIs is to think about a metaphor, and a common scenario
- The customer, a waiter, and a restaurant kitchen!
 - ✓ A customer talks to the waiter and tells the waiter what she wants
 - ✓ The waiter takes down the order and communicates it to the kitchen
 - ✓ The kitchen does their work, creating the food
 - ✓ Then the waiter delivers the order back to the customer



BITS Pilani
Pilani Campus



Design of API



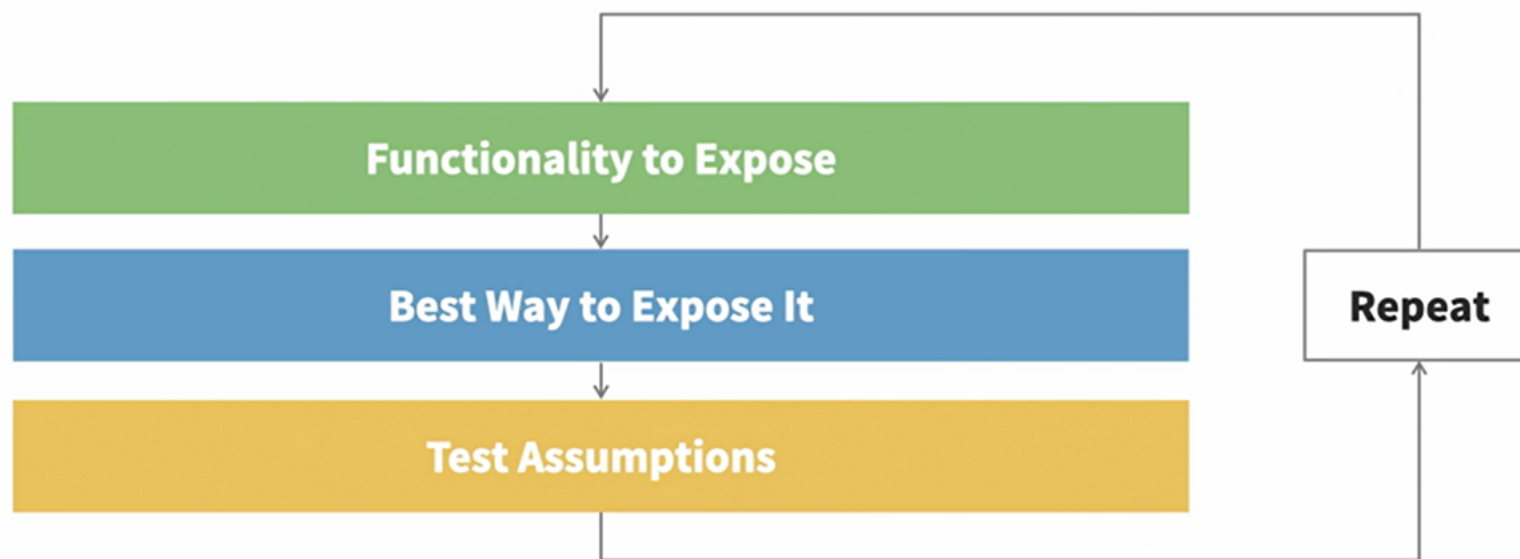
API design is complex.

innovate

achieve

lead

API Design Must Be Deliberate



Challenges of Good API Design



- Clear naming
- Clear directions
- Knowledge of use case
- Adaptability
- Versioning
- Backward compatibility

API Design by Choice



Affordance

Something that allows you to perform an action

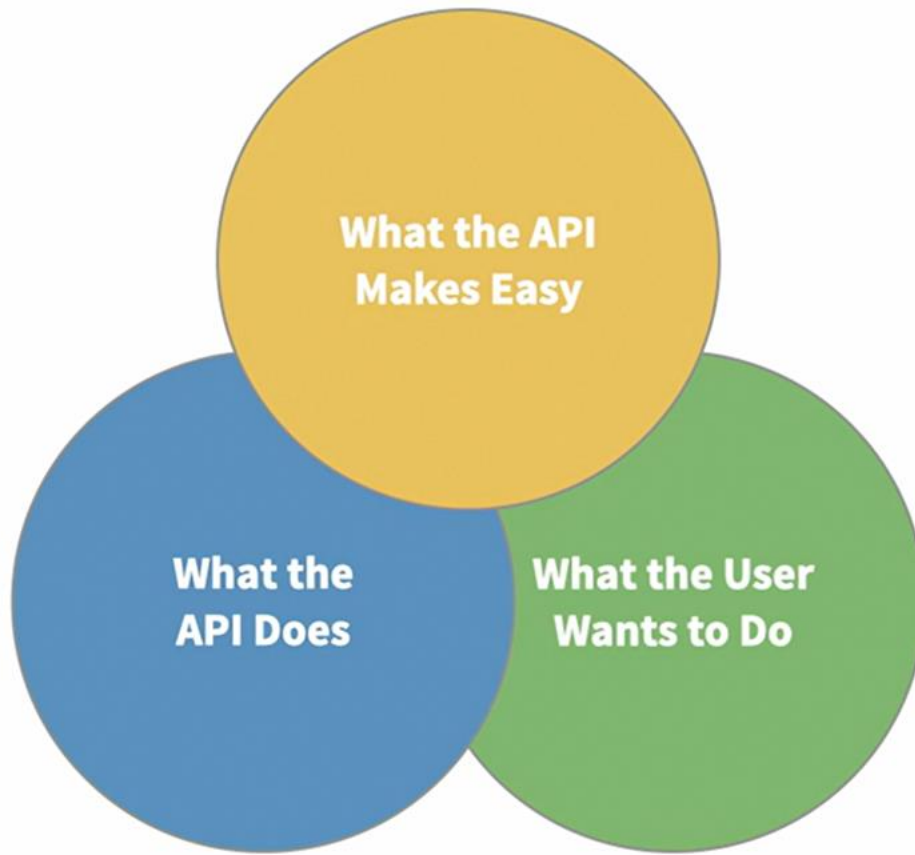
Door Knob



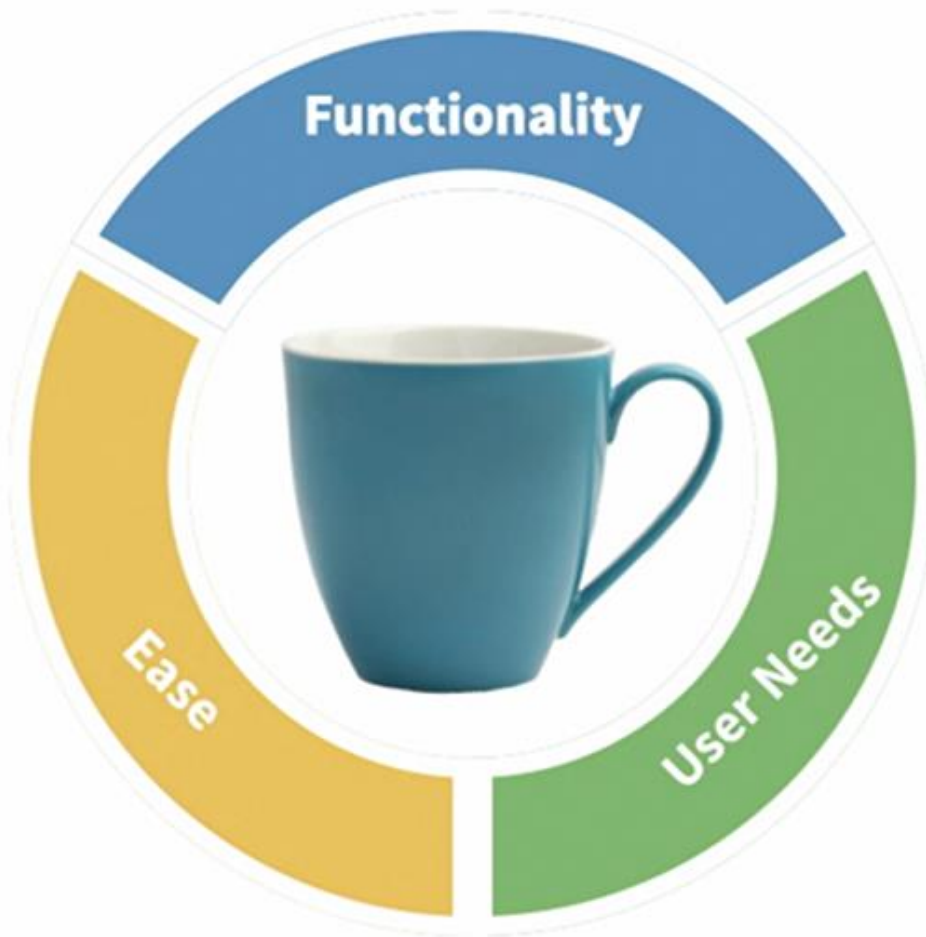
Light Switch



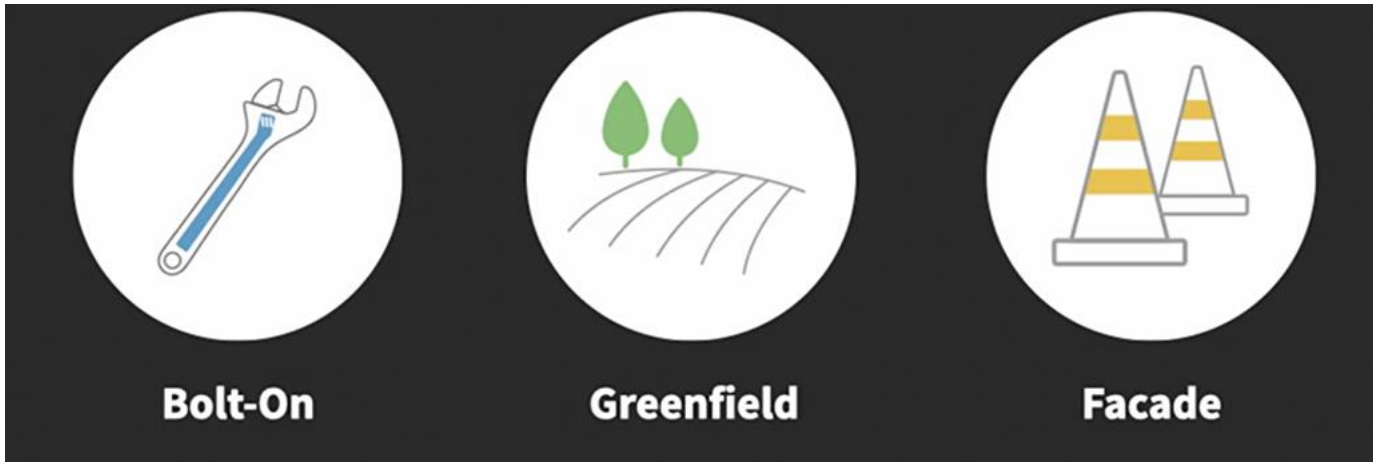
API Design Risk



API Design must be deliberate



Approach to adding an API



Regardless of which strategy you choose, modeling will be key to your success.

Approach to adding an API



Bolt-On Strategy

For existing systems

- Brute-force approach
- The fastest way to build something useful
- **Benefit:** takes advantage of existing code and systems
- **Drawback:** problems in the application or architecture *leak through* into the API



Approach to adding an API



Greenfield Strategy

For new systems

- **API** or **mobile-first** mindset
- **Benefits:** takes advantage of new technologies and architectures and may reinvigorate the team
- **Drawback:** often requires massive upfront investment before any benefits appear

Approach to adding an API



Facade Strategy

Replacing piece by piece

- **Benefit:** ideal for legacy systems as the application is always functional
- **Drawback:** multiple mindsets in the system
- **Drawback:** hard to replicate behavior for a full one-on-one conversion



How to model API?



Build Consistency

- **Involve teams early**
- **Various perspectives will help develop the project**
 - Reinforce strengths
 - Uncover weaknesses



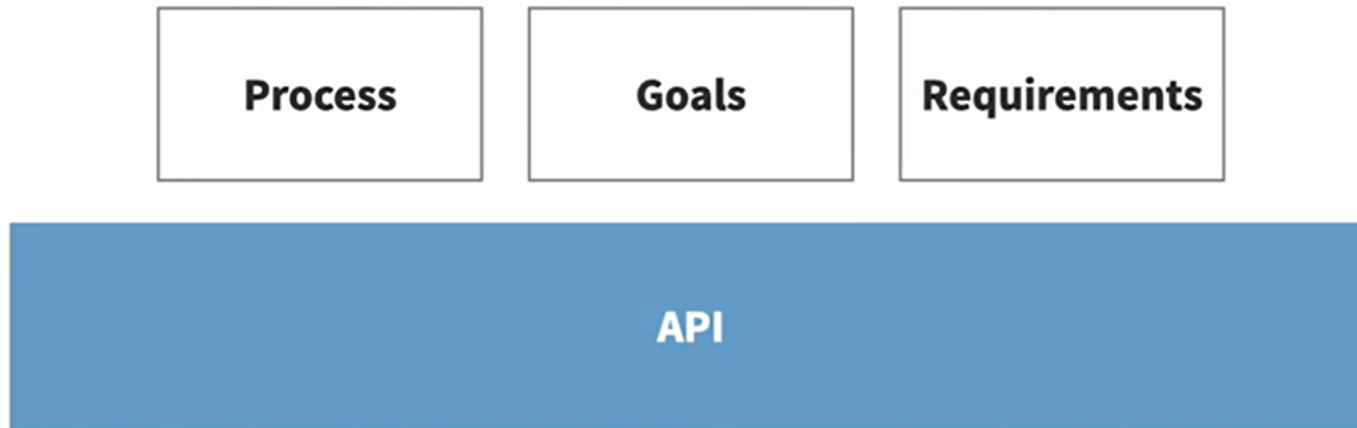
Have a consistent process. Don't worry about the tools.

Document Everything

- Assumptions
- Decisions
- Deferred tasks

Documentation is non-negotiable.

Support Business



we're not going to model the API quite yet. The first thing we need to do is have a clear understanding of our business process. If we don't know what that is and what it accomplishes, we'll never be able to build an API to support it.

API Modelling Process



Participants

innovate

achieve

lead



Participant Info

- **Who are they?**
 - Name and role
- **Internal or external?**
- **Active or passive?**
 - Active – taking an action
 - Passive – waiting for action

Case Study -1

Ordering a Cup Of Coffee @ Barista



Question



Who are the participants here?



Answer



Coffee Participants

- Customer
- Barista
- Cashier



Other Participants?



You pay with Credit/Debit Card or Cash



Are Other Customers Participants Too?

lead



Are there other customers? Are there orders coming up before or after ours? Are they participants too? This is starting to lead to one of the most dangerous problems involved in modeling any process, the boundaries. If we're not careful, we could end up modeling inventory management systems for the cups, the coffee machine for the workflow, the payment processor, and lots of other things. We must be careful here. In this case let's keep it simple. Let's consider you, the cashier, and the barista, with the acknowledgement that the cashier and the barista could be the same person.

Modelling Boundaries



Good: Strong Scope



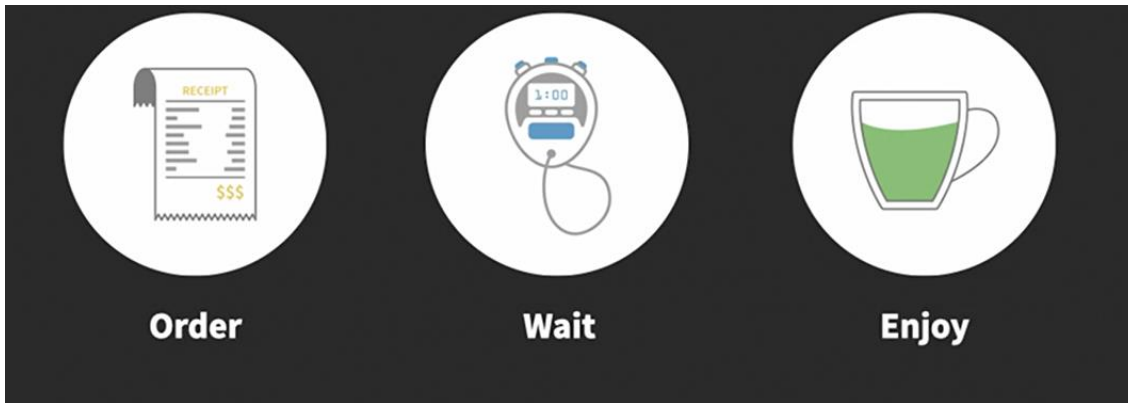
Bad: Wrong Assumptions



Question



Identify activities while ordering coffee?



Activities



- 1 The patron places an order with the cashier.
- 2 The cashier passes the order to the barista.
- 3 The barista acknowledges the order and adds it to the queue.
- 4 The cashier tells the patron their total bill.
- 5 The patron provides payment, which is accepted or rejected.
- 6 The barista makes the orders and delivers it to the patron.

Who are the participants here?

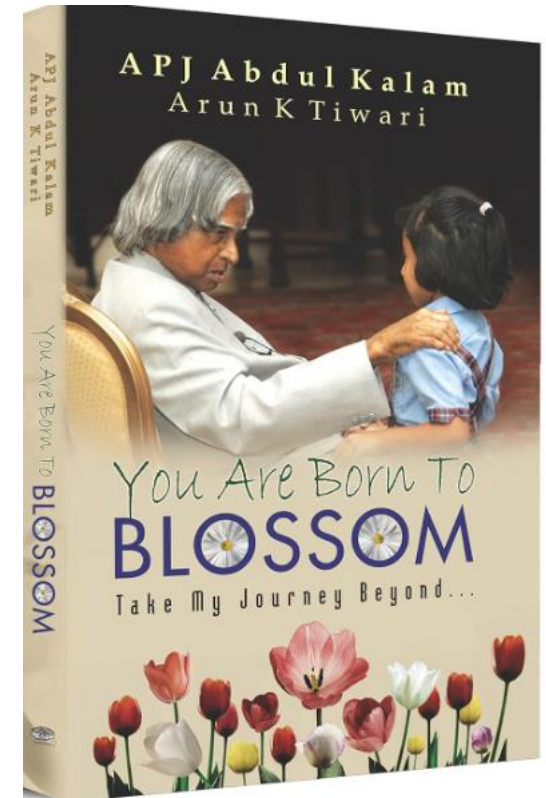
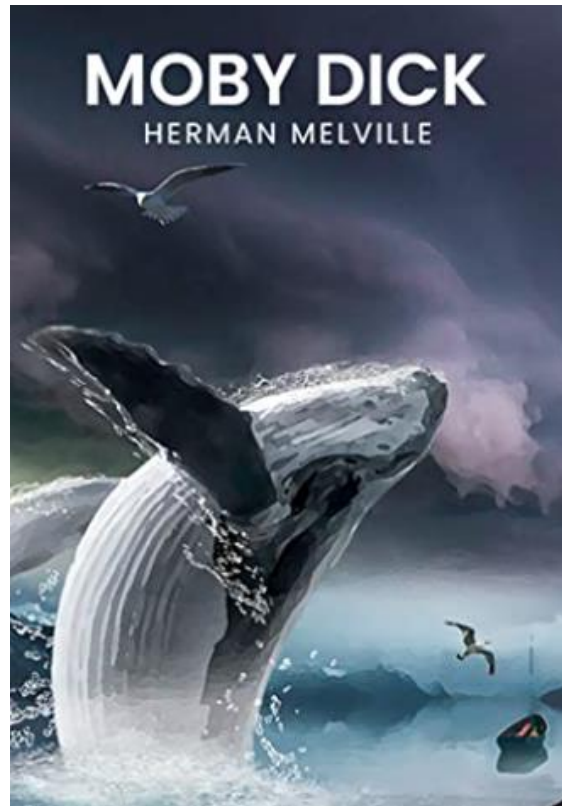
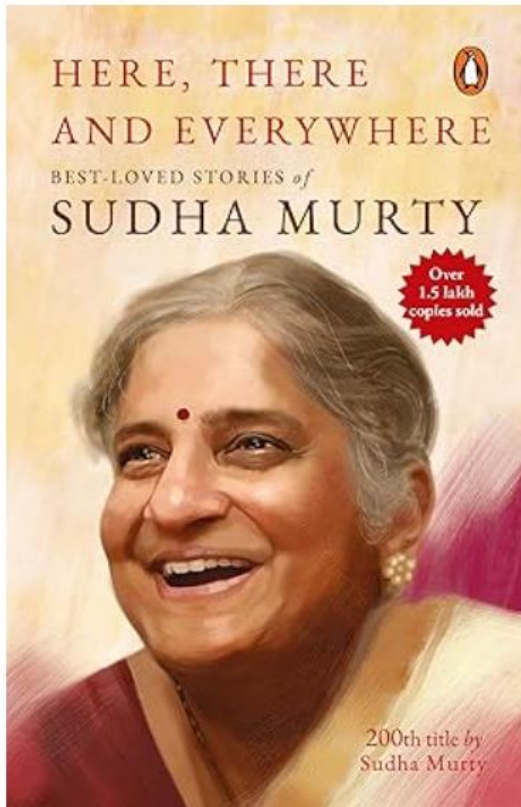


- 1 The **patron** places an order with the **cashier**.
- 2 The **cashier** passes the order to the **barista**.
- 3 The **barista** acknowledges the order and adds it to the queue.
- 4 The **cashier** tells the **patron** their total bill.
- 5 The **patron** provides payment, which is accepted or rejected.
- 6 The **barista** makes the orders and delivers it to the **patron**.

Case Study -2



Identify **participants** and **activities** when ordering a book online.



Who are our participants?

- 1 Customer
- X System admin
- X Developer
- 2 Stock clerk
- 3 Customer support

Process Steps

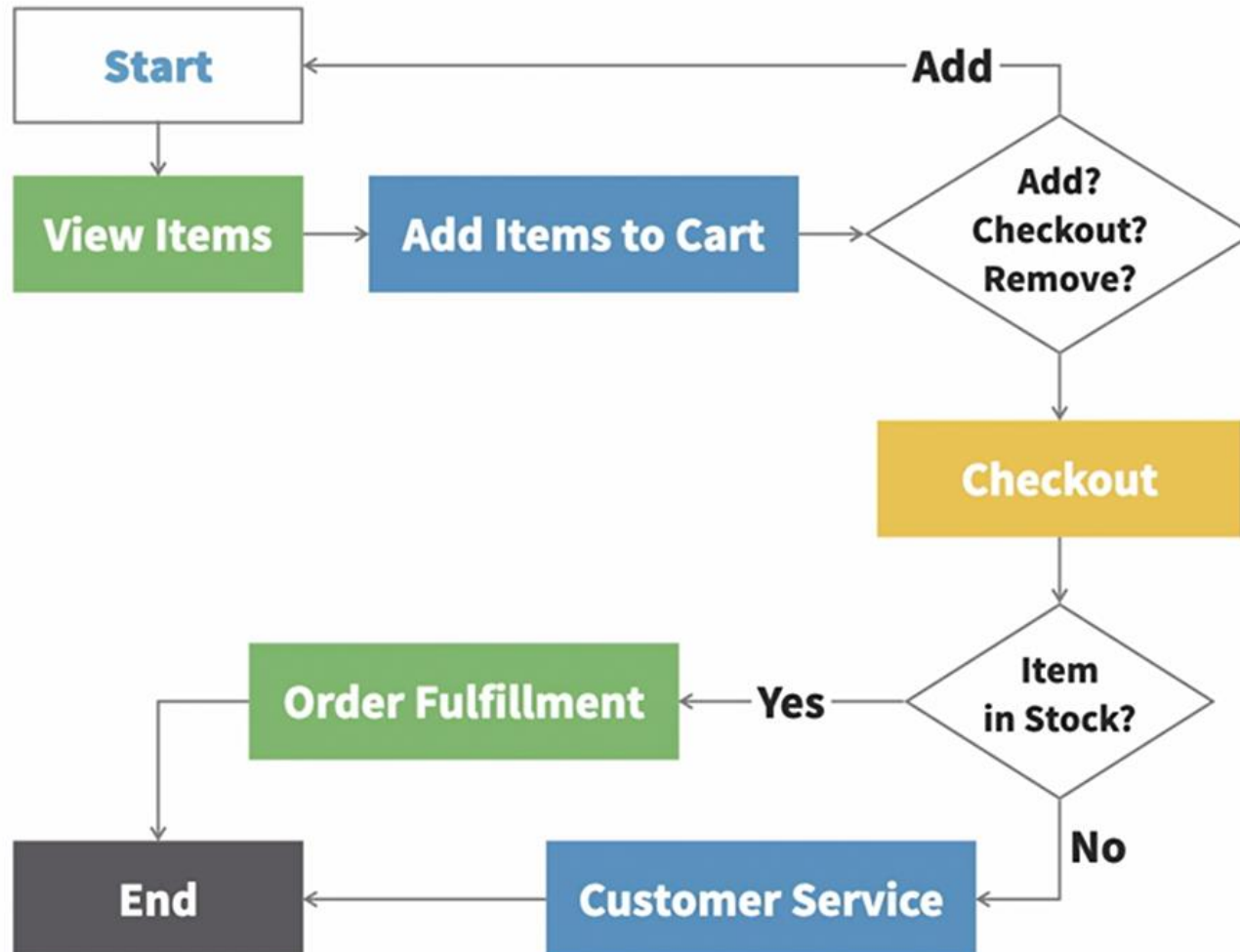


Ordering a Book Online

What are our activities?

- 1** The customer searches for the book.
- 2** The customer adds the book to their cart.
- 3** The customer adds or removes more things.
- 4** The customer checks out.
- 5** The stock clerk retrieves and ships the book.
- 6** Customer support contacts the customer about the book.

Book Order Process Flow



API Modeling Steps



Refined Process Steps

- 1 View items.
- 2 Add item to cart.
- 3 Add or remove more items.
- 4 Place order.
- 5 Ship order.
- 6 Cancel order.



- 1 View items.
- 2 Add items to cart.
- 3 Place order.



API Resources

Anything users interact with



- 1 View **items**.
- 2 Add **items** to **cart**.
- 3 Place **order**.

In our case, first we have items. You'll generally want to view, edit, and add items or in other words create, retrieve, and update, the beginning of CRUD. Listing and searching for items are typically special cases of retrieving.

Our next noun is cart. And this is the first place where we have to make a little bit of a design decision. Is our cart just a collection of items or is there a line item resource or object which holds each individual item? In this case, we'll create a cart as a collection of items. It's just referencing other items. Therefore, adding or removing is just changing the state of the cart itself.

Our next and final noun is orders. An order is simply a cart which has undergone the checkout process.

Create and Grouping API



Items

- **View** items
- **Edit** items
- **Add** items
- **List** items
- **Search** for items

CRUD

- **Create**
- **Read**
- **Update**
- **Delete**

HTTP

- **Post**
- **Get**
- **Put**
- **Delete**

Resources for the Bookstore API



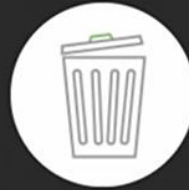
- List items
- View items
- Add item to cart
- View cart
- Check out
- List orders
- View orders
- Cancel orders

HTTP Verbs



GET

Retrieves a resource or collection of resources



DELETE

Deletes a resource



PUT

Updates an existing resource



POST

Catch-all for HTTP

- Used to create a new resource
- Used to change the status or state of a resource
- Used for anything else that doesn't cleanly fit the first three verbs



Create API Definition



Item Resource

- List items
- View item




HTTP Verbs

- GET items 
- GET item 

Cart Resource

- Create cart
- Add item to cart
- Check out (cart to order)

HTTP Verbs

- POST cart 
- PUT cart 
- POST cart 



Create API Definition



Order Resource

- Create order
- View order
- Cancel order

HTTP Verbs

- *Complete before checkout*
- GET order 
- POST order 

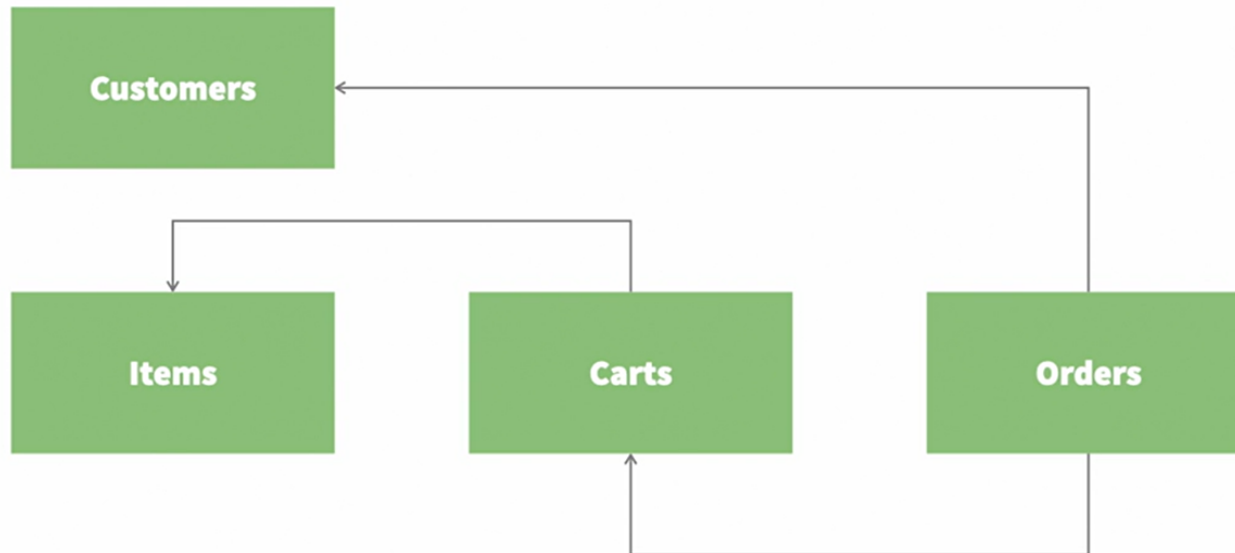
API Relationship



- **Independent:** movies and actors
- **Dependent:** characters in movies
- **Associative:** one actor plays multiple characters; one character played by multiple actors

Our Resources

- Items are independent
- Carts must have items (dependent)
- Orders come from carts (dependent)
- Orders must have customers (dependent)



Validate the API



- **List the end points** – describe what they do
- **List the parameters** – describe what they mean
- **List the response codes** – describe when you get each
- **Show the response payloads** – define the fields



API Design – Group Exercise



Step 1 : Define Business Objective

The Problem

Briefly define the problem and how it affects customers and the business.

The Impact

Define what success looks like for your API. What will the world be like after you've released your new API?

Key User Stories

List several key user stories for your API with the following template:

As a [user type], I want [action] so that [outcome].

Step2 : API Specification Template



Title

Problem

Solution

Implementation

Give a high-level description of the implementation plan.
You may wish to use additional tables or diagrams to describe your plan.

Authentication

Describe how developers will gain access to the API.

Other Things to be Considered

If you considered any other API paradigms, architectures, authentication strategies, protocols, etc., briefly mention what you considered.

Step3 : API Interactions

Inputs, Outputs (REST, RPC)

If you're designing a REST or RPC API, describe the endpoints, inputs, and outputs. You may wish to add columns or use a different table format to describe the requests and responses.

URI	Input	Output

Events, Payloads (Event-Driven APIs)

If you're designing an event-driven API, describe the events and their payloads. You may wish to add columns for additional information, such as OAuth scope.

Errors

HTTP status code	Error code	Verbose error	Description

API : Class Discussions



Group	API	Purpose
1	EchoNestAPI	Provides access to music-related data and analytics for developers building music-centric applications.
2	PulseSyncAPI	Synchronizes real-time data updates across multiple devices or applications for consistent user experiences.
3	CodeCloudAPI	Integration with cloud services for seamless code deployment and management.
4	DataForgeAPI	Facilitates data manipulation and transformation for data-driven applications.
5	SwiftConnectAPI	Provides a streamlined interface for connecting and communicating with Swift-based applications.
6	MetaMeshAPI	Manages mesh networks and facilitates communication between interconnected devices or nodes.
7	SynthifyAPI	Generates synthetic data for testing and development purposes, ensuring data privacy and security.
8	CipherFuseAPI	Offers encryption and decryption services for securing sensitive data transmission and storage.