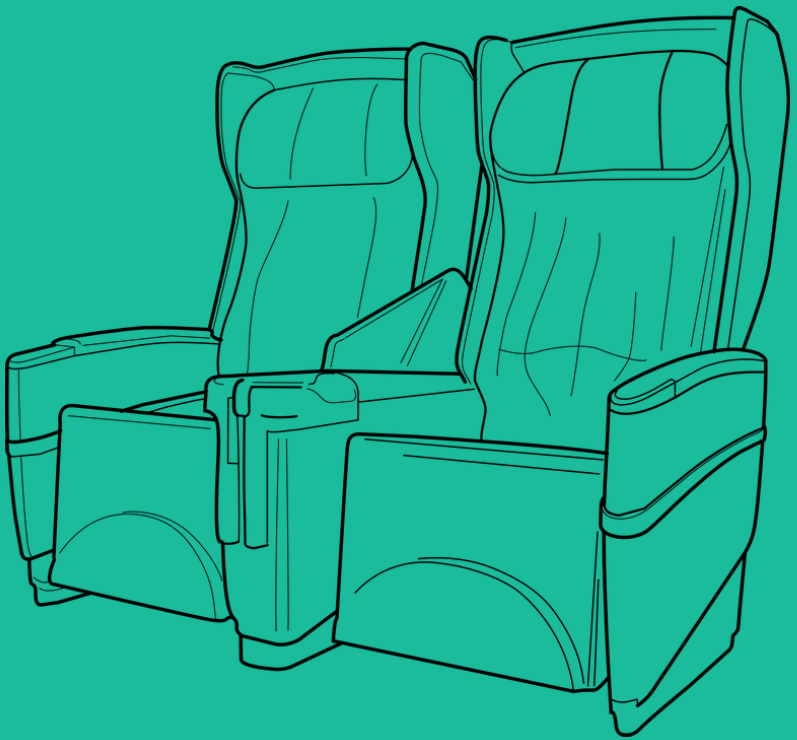
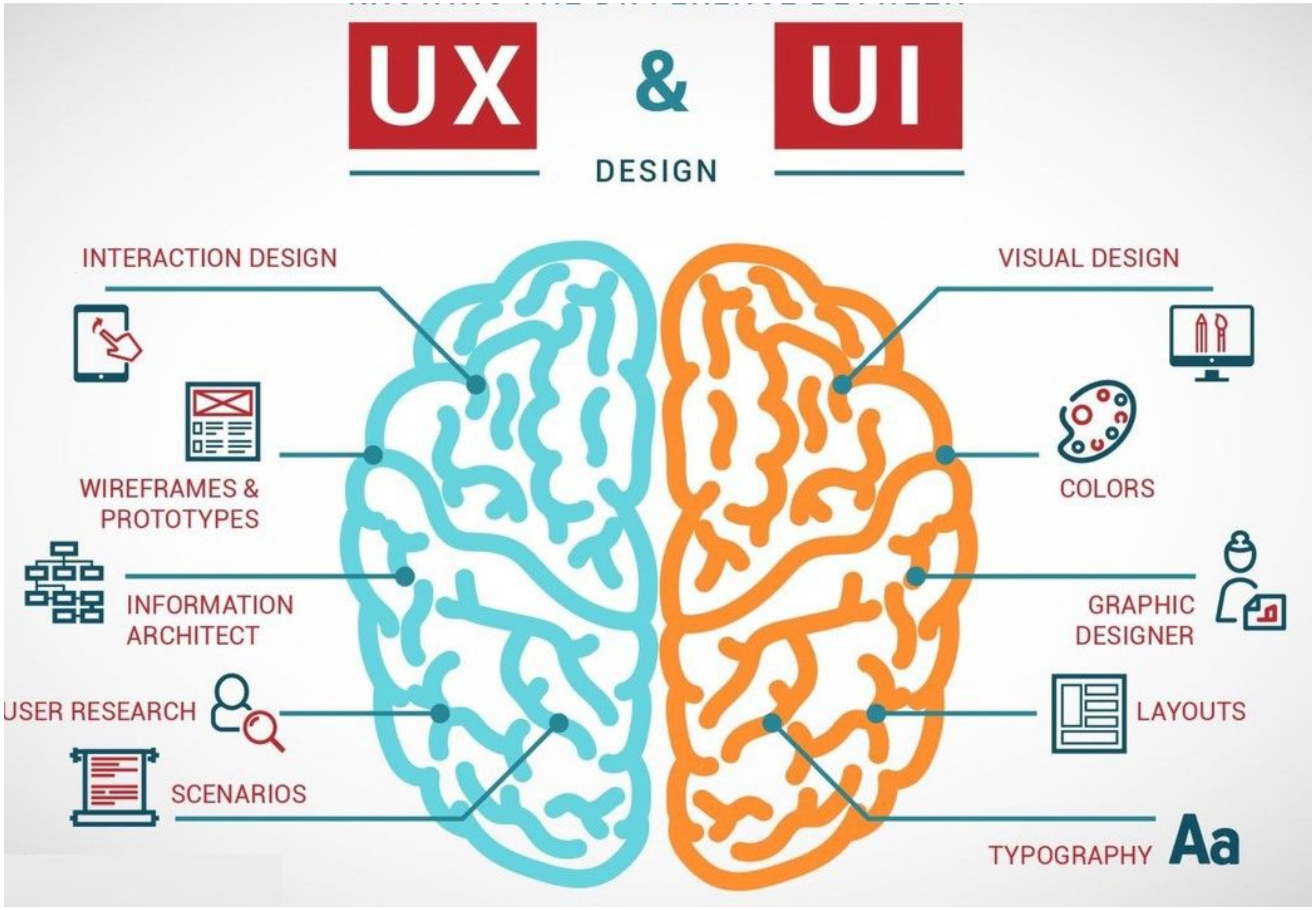


Chapter X-3. UI/UX/DX Design

Bilkent University | CS443 | 2021, Spring | Dr. Orçun Dayıbaş

User Experience (UX)

- **UI vs. UX**



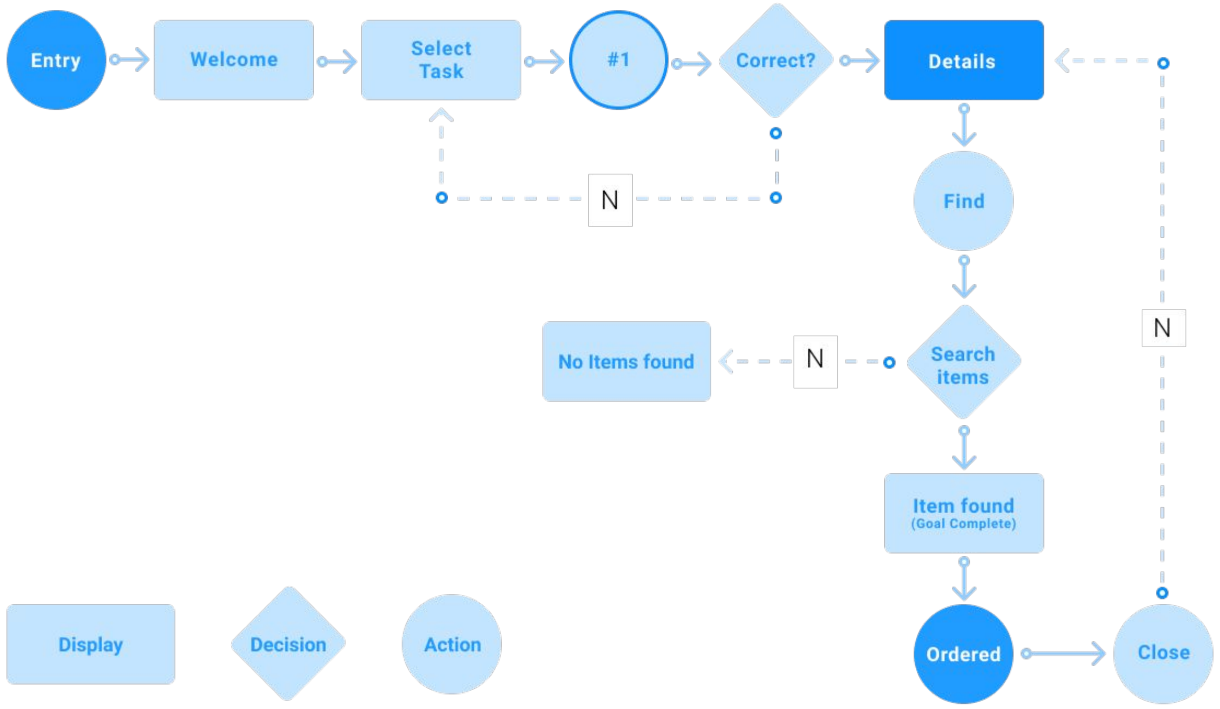
User Experience (UX)

- **7 factors**
  - Useful & Valuable
    - It has to have a purpose (clear, concise one)
    - It has to deliver a value
  - Usable
    - “The iPod wasn’t the first MP3 player but it was the first truly usable one”
  - Findable
    - The content within them must be easy to find
  - Credible
    - “The ability of the user to trust in the product that you’ve provided”
  - Desirable
    - Desirability is conveyed in design through branding, image, identity, aesthetics and emotional design
  - Accessable
    - Accessibility is about providing an experience which can be accessed by users of a full range of abilities

https://www.interaction-design.org/literature/article/the-7-factors-that-influence-user-experience

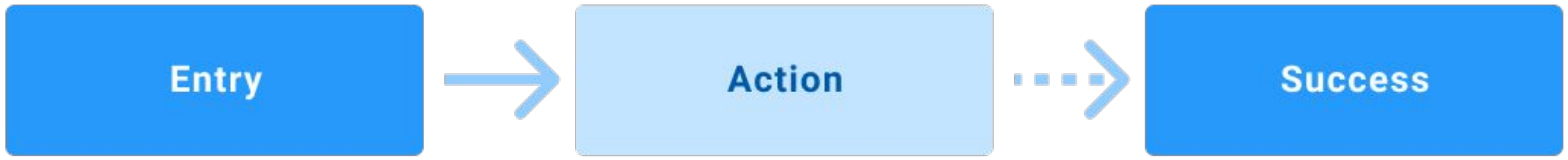
User Flow Design

- **User flows / UX flows**
  - Diagrams that display the complete path a user takes when using a product
  - The user flow lays out the **user's movement through the product**, mapping out each and every step the user takes



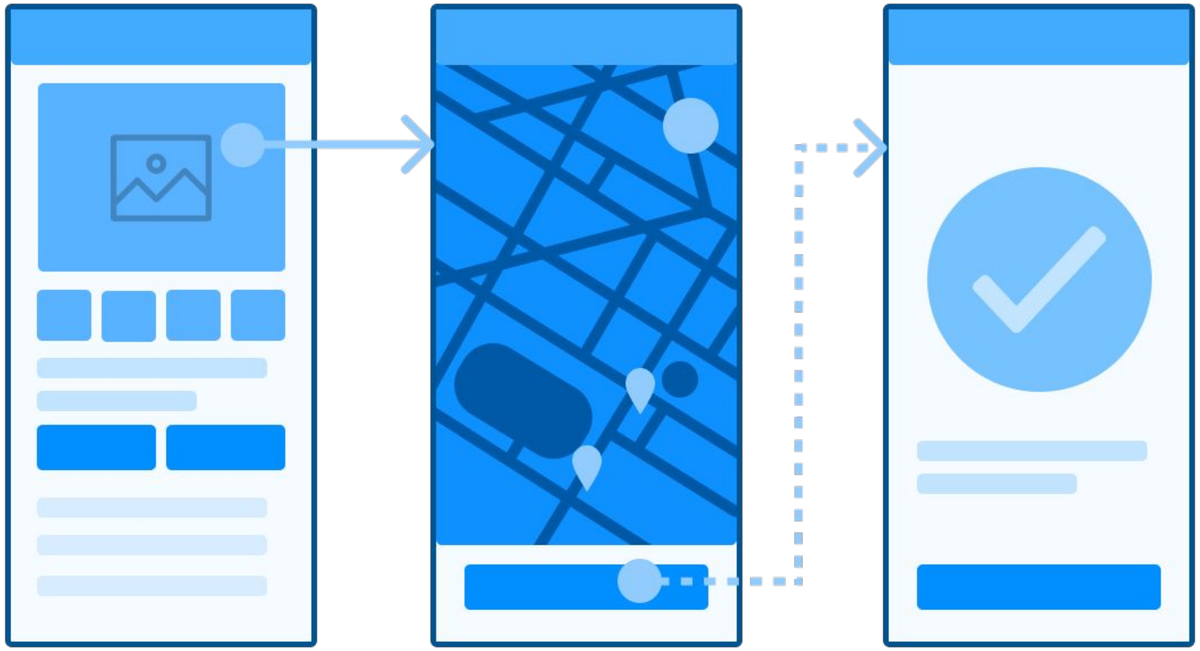
User Flow Design

- **Types of user flow charts**
  - Task flows
    - Task flows focus on how users travel through the platform while performing a specific task.
    - They generally show only **one path and don’t include multiple branches** or pathways like a traditional user flow might.
    - These are best used when the task being analyzed is accomplished similarly by all users. When using task flows, it is assumed that all users will share a common starting point and have no variability in the way the task is carried out.



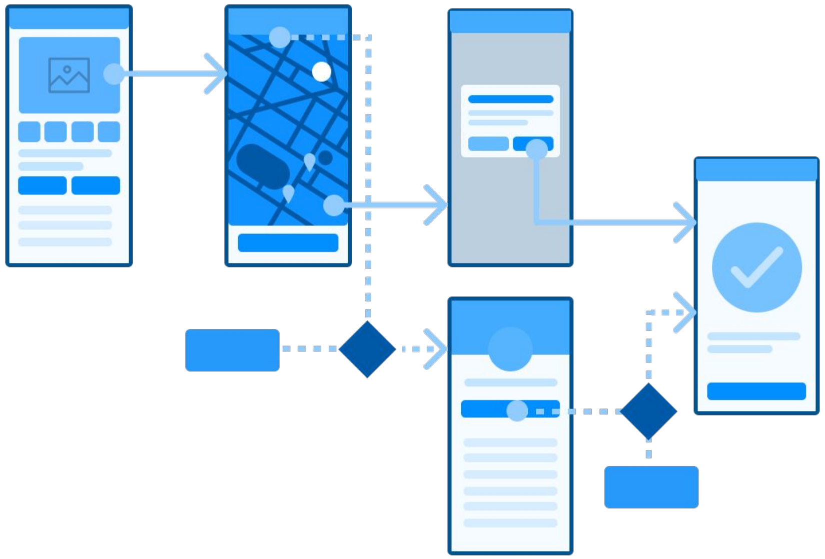
User Flow Design

- **Types of user flow charts**
  - Wire flows
    - Wireflows are a combination of wireframes and flowcharts.
    - They utilize the layout of individual screens as elements within the diagram.
    - Wireflows add page context to UX flows.



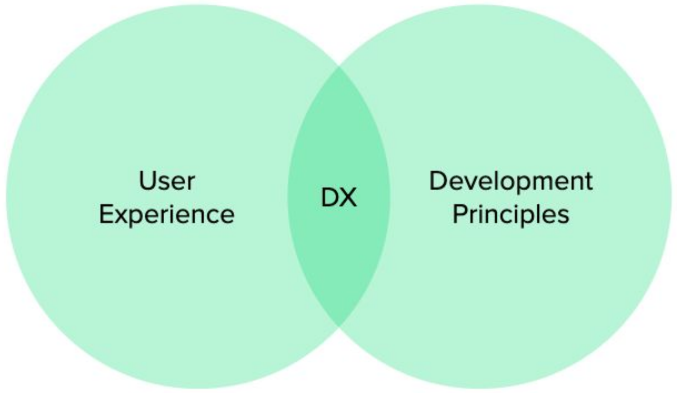
User Flow Design

- **Types of user flow charts**
  - User flows
    - User flows focus on the way your target audience will interact with the product.
    - They emphasize that all users might not perform tasks the same and may travel in different paths.
    - They are typically **attached to a specific persona** and entry point.
      - When using this type of flowchart, you may have many different scenarios that start at different places.
      - However, the main task or accomplishment is usually always the same.



Developer Experience (DX)

- **UX vs. DX**
  - Developer Experience is the equivalent of User Experience when the primary user of the product is a developer
    - DX cares about the developer experience of using a product, its libs, SDKs, documentation, frameworks, open-source solutions, general tools, APIs, etc.
    - Developers deserve solutions as well designed as non-technical people
  - DX and UX share some principles, but with differences in good practice
    - This is because developers have different needs in their daily context compared to an average user.
    - DX is important for the same reasons that UX is important

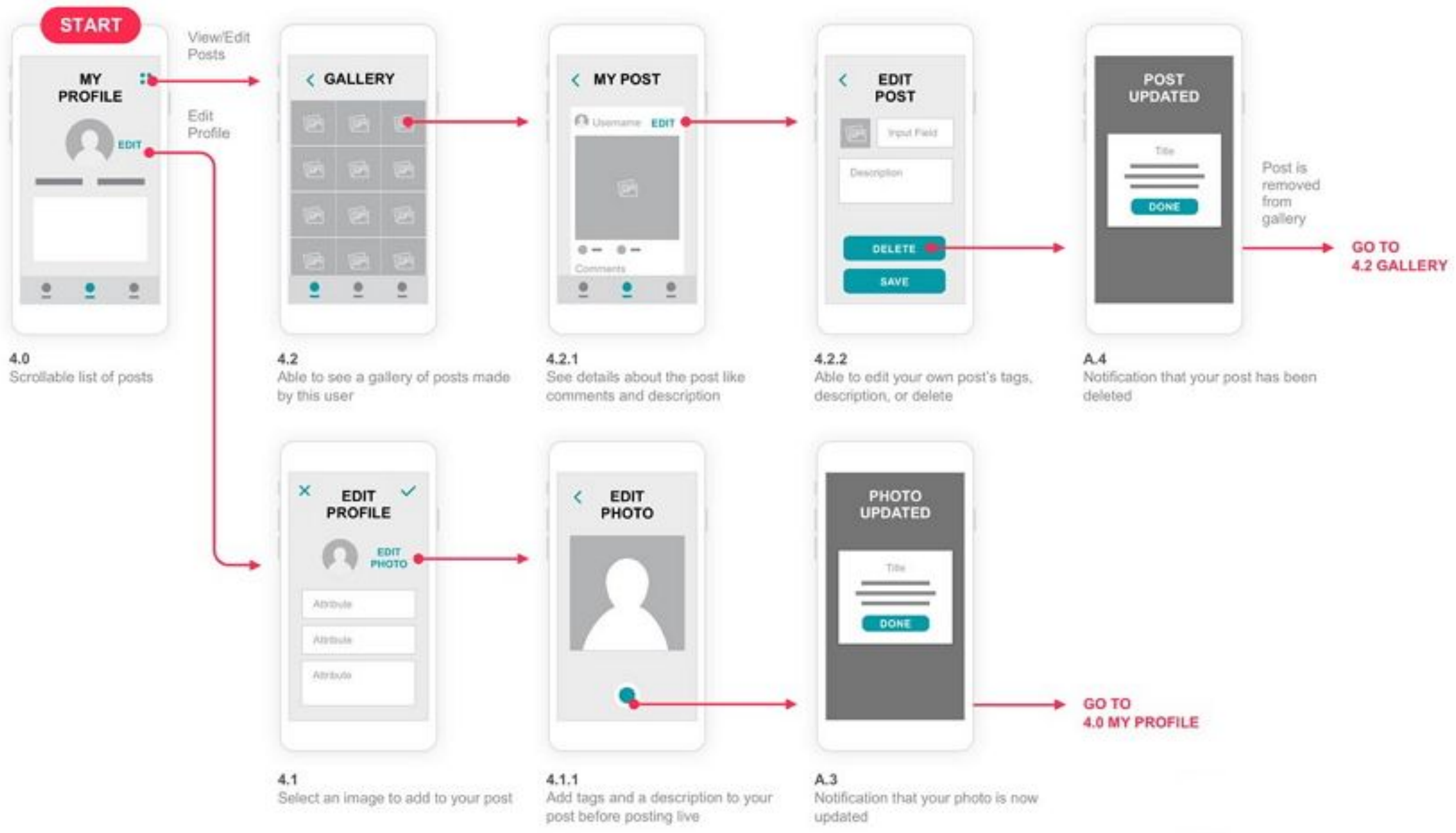


Developer Experience (DX)

- **Pillars of DX**
  - Function
    - The foundation of the developer experience, a dev tool is as good as the role it offers to perform an activity. If it doesn't work, it's no use, there's no DX.
  - Stability
    - In addition to working, your product has to have high performance and reliability, of course, the software is subject to bugs, so it is important to quickly fix product errors so as not to cause major harm to users.
    - Instability in the relationship of trust, the perception of value drops dramatically.
  - Ease of Use
    - Ease of use in DX is beyond what it seems, it's not just about navigating the tool, but also accessing what you need at all stages of the journey quickly and efficiently (Rich documentation, use cases, communities, knowledge bases, keyboard shortcuts, snippets, intuitive filters, etc.)
  - Clarity
    - Clarity is about giving the developer full visibility of the possible consequences involved in an action and the history of these actions.

Case Study

- **User flow design (Level 0)**



4.0
Scrollable list of posts

4.2
Able to see a gallery of posts made by this user

4.2.1
See details about the post like comments and description

4.2.2
Able to edit your own post's tags, description, or delete

A.4
Notification that your post has been deleted

4.1
Select an image to add to your post

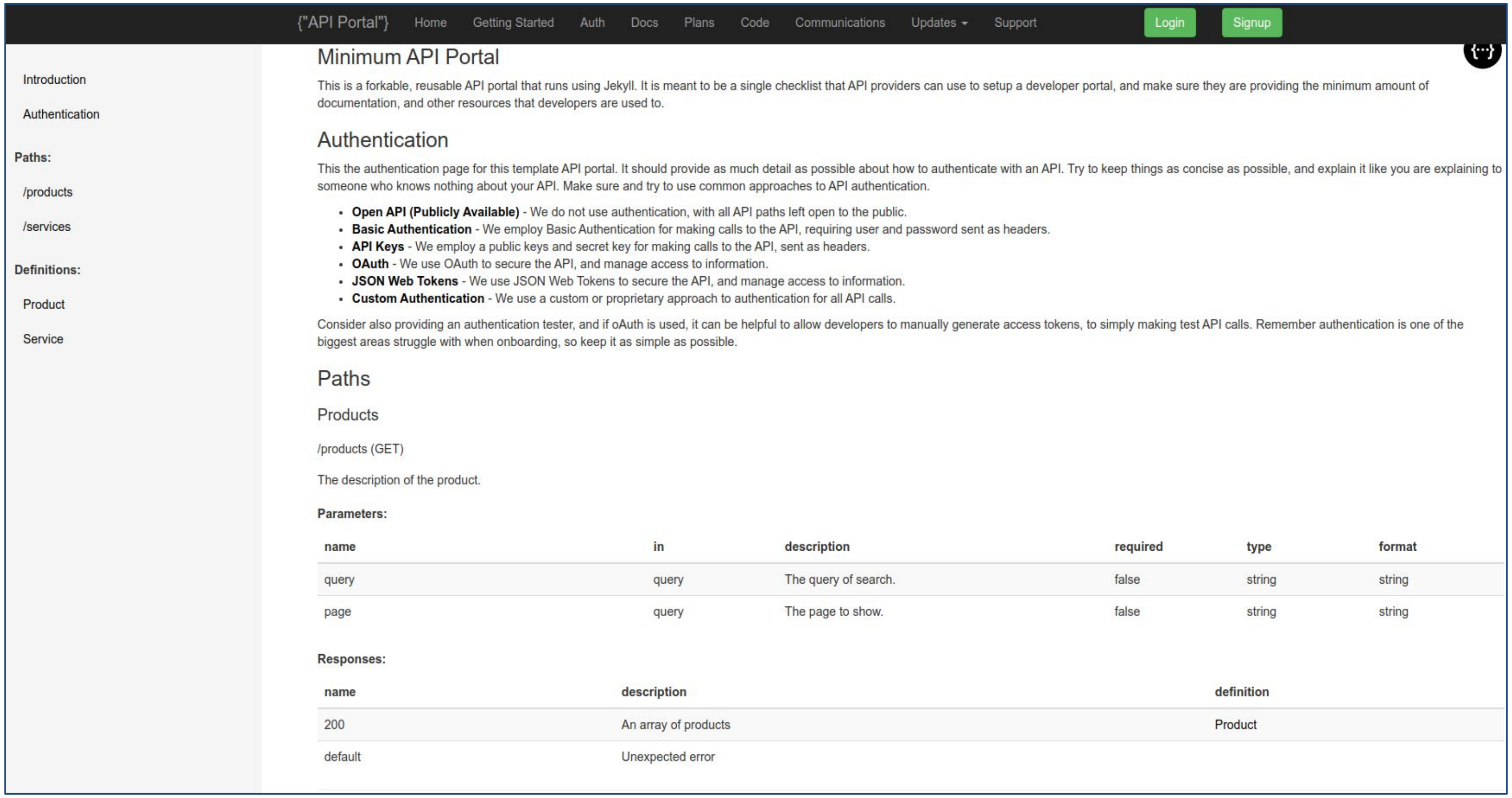
4.1.1
Add tags and a description to your post before posting live

A.3
Notification that your photo is now updated

https://www.mockplus.com/blog/post/user-flow-tools

Case Study

- **DX design (Level 0)**
  - Ex: http://portal.minimum.apievangelist.com/
  - https://api.nasa.gov/



Minimum API Portal

This is a forkable, reusable API portal that runs using Jekyll. It is meant to be a single checklist that API providers can use to setup a developer portal, and make sure they are providing the minimum amount of documentation, and other resources that developers are used to.

Authentication

This the authentication page for this template API portal. It should provide as much detail as possible about how to authenticate with an API. Try to keep things as concise as possible, and explain it like you are explaining to someone who knows nothing about your API. Make sure and try to use common approaches to API authentication.

- **Open API (Publicly Available)** - We do not use authentication, with all API paths left open to the public.
- **Basic Authentication** - We employ Basic Authentication for making calls to the API, requiring user and password sent as headers.
- **API Keys** - We employ a public keys and secret key for making calls to the API, sent as headers.
- **OAuth** - We use OAuth to secure the API, and manage access to information.
- **JSON Web Tokens** - We use JSON Web Tokens to secure the API, and manage access to information.
- **Custom Authentication** - We use a custom or proprietary approach to authentication for all API calls.

Consider also providing an authentication tester, and if oAuth is used, it can be helpful to allow developers to manually generate access tokens, to simply making test API calls. Remember authentication is one of the biggest areas struggle with when onboarding, so keep it as simple as possible.

Paths

Products

/products (GET)

The description of the product.

Parameters:

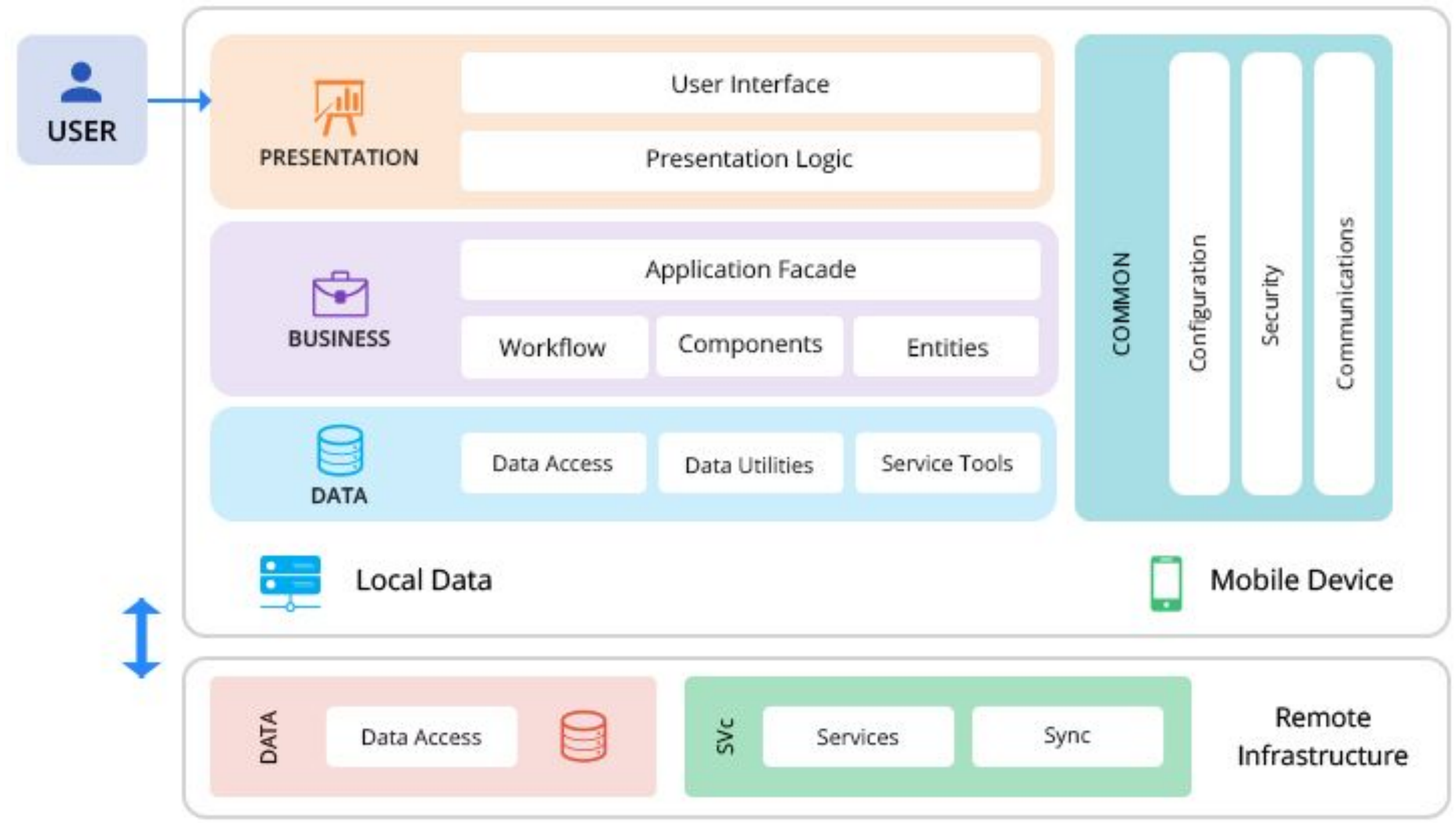
name	in	description	required	type	format
query	query	The query of search.	false	string	string
page	query	The page to show.	false	string	string

Responses:

name	description	definition
200	An array of products	Product
default	Unexpected error	

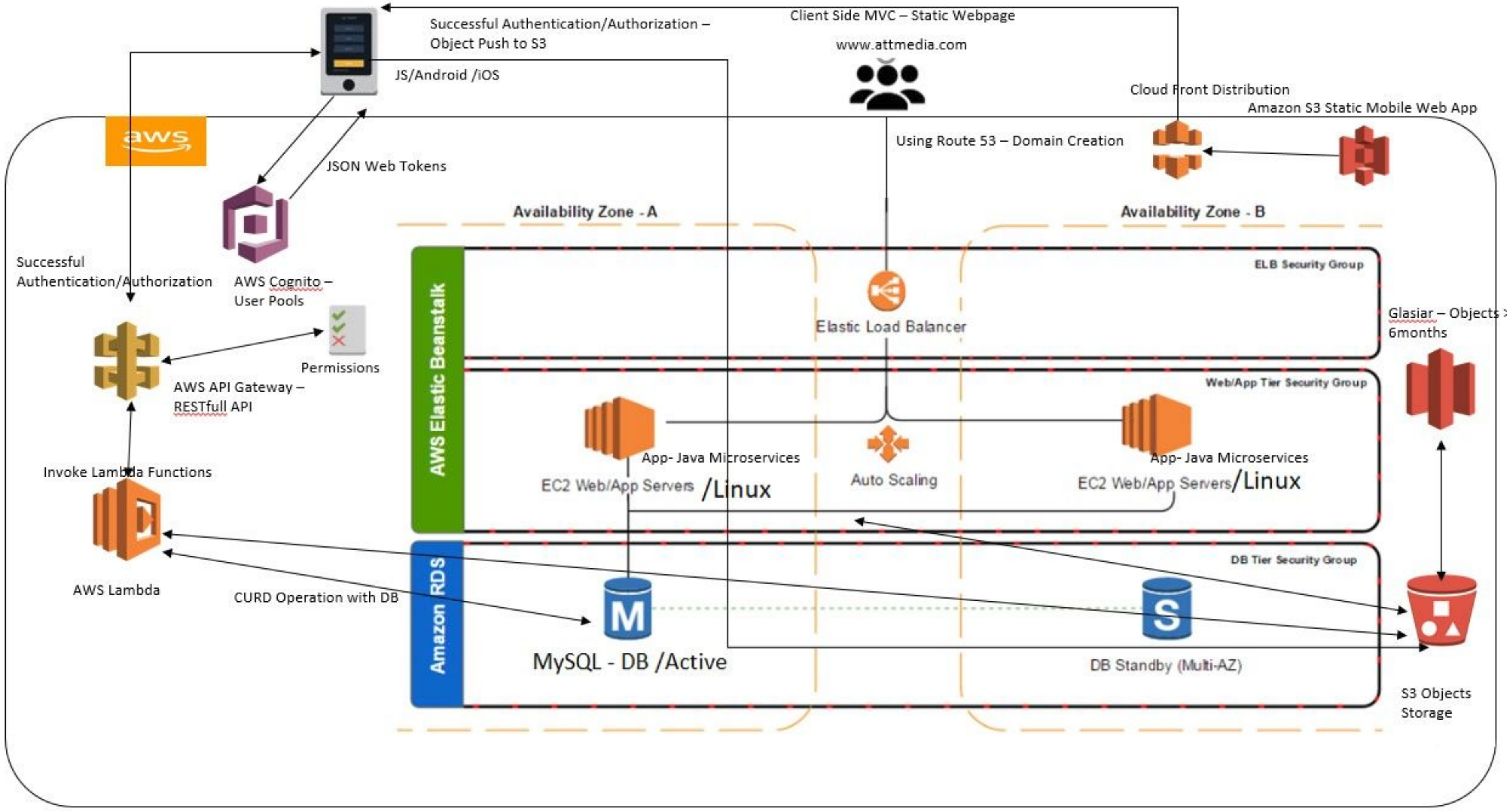
Case Study

- **Architecture (Level 1)**



Case Study

- **Architecture (Level 2)**



https://dzone.com/articles/mobile-based-architecture-on-existing-lamp-stack-e

Conclusion

- **Platform & frameworks**
    - Decide according to your business needs, ecosystem, maturity, existing know-how etc. Do not follow the trend blindly.
    - It is also valid for native / web / hybrid application dev.
- **UX & DX**
    - They are equally important if you have both B2C & B2B channels. Do not reinvent the wheel.
    - Follow practices like “Design Thinking”
    - Follow API design best practices
        - Ex: Zalando REST API Guidelines, Microsoft REST API Guidelines
- **Architecture**
    - Remember you need a “Level 3” if your solution is on-prem

Q/A