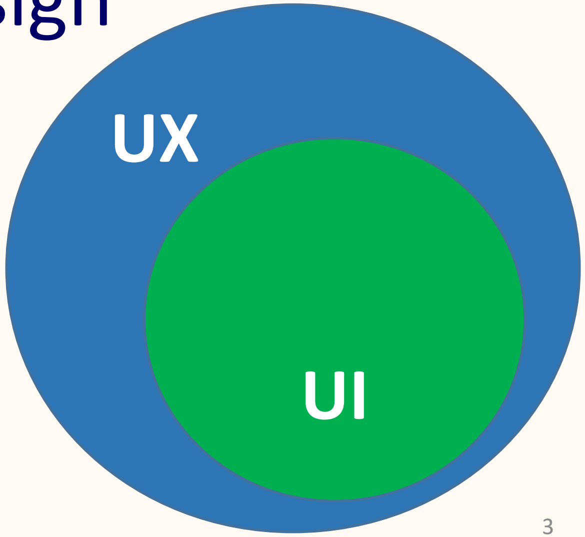
UI Design
& System Design

Outline

- UX and UI design
  - Prototyping
  - Usability testing
  - UI for your project
- System design
  - Architecture diagrams
  - Interaction Diagrams

UX and UI

- **UX** design = **U**ser e**X**perience design
  - Manage user's journey/reactions as they interact with product/service
- **UI** design = **U**ser **I**nterface design
  - Focus on actual construction of interface for product/service
  - What user sees/hears/etc.
  - How user interacts w/ product



More about UI design

- UI design is an integral aspect of UX design
- Consists primarily of two major parts
  - Visual design—how will the product look?
  - Interaction design
    - How does the UI function?
    - How are these functions logically organized?
- Goal: create a UI that makes interaction with product easy, efficient *(enjoyable would be UX)*

Design Prototyping

- Process where UI design teams ideate and experiment with design concepts
  - One extreme: Paper and pencil
  - Other extreme: Digital representation
- Results in ***prototype***: early sample of design
  - Allows users to interact prior to development
  - May even have limited functionality

Not a development prototype

- In a **development** context a prototype…
  - Means creation of preliminary implementation
  - Intent is to prove approach/technology
  - Can take fair bit of time to create
- In the **UI design** context a prototype…
  - Intent is to find good ways to present information to the user and allow them to respond
  - Should take less time to create

No-code prototyping

- Prototype created without single line of code
- Prototyping tools reduce costs
  - Allow designers to link artboards into an interactive, clickable experience
  - No developer or development time needed
- Can test prototype with users & stakeholders
  - Iron out errors/oddities before invest time & $$ in developing the product

Reasons for prototyping

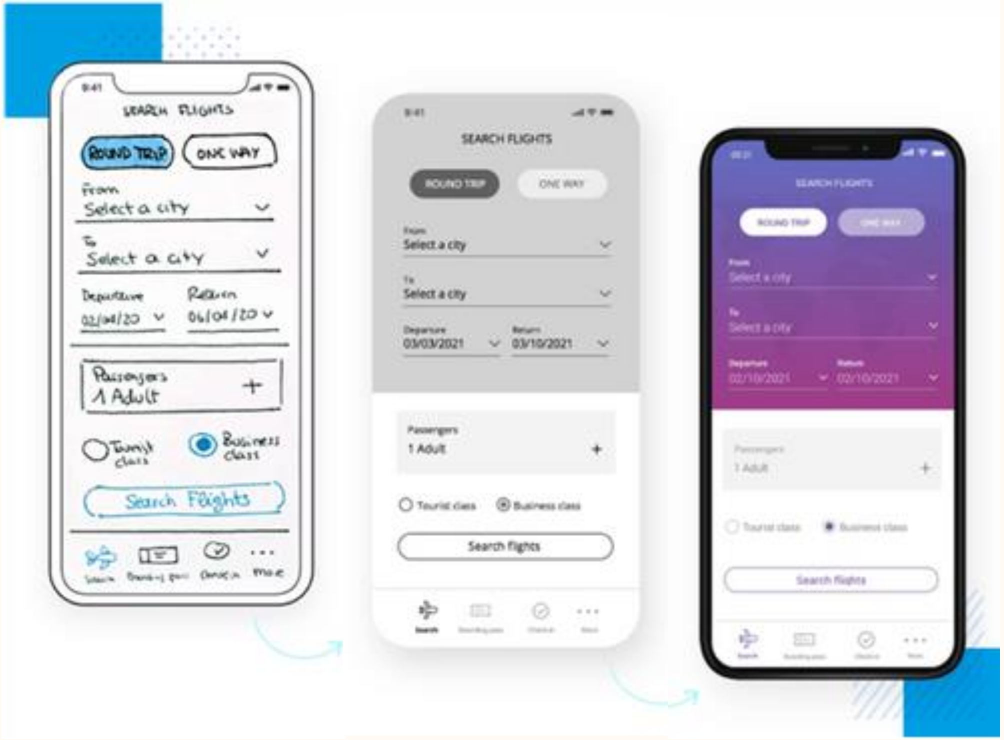
- Explore new ideas
- Discover possible problems
  - Examples: forgotten inputs, crowded screen
- Identify usability issues
  - Examples: too many clicks, needs user’s memory
- Engage stakeholders & end users
- Sell new concepts

Fidelity of prototypes

- Fidelity: level of detail & functionality
  - Different scenarios call for different levels
  - Each has its own pros and cons
- Higher fidelity → more effort/cost to create

Three levels of fidelity

- Low: **paper prototype**
- Medium: **wireframe**
  - Digital, but greyscale
  - Allows user flows
  - May have basic data
- High: **prototype**
  - Branding, colors
  - Photos, animations



Prototyping process (web/mobile)

1. Select **features** to test with users
2. Create **design prototype** highlighting **them**
3. Test **it** with users, partners, stakeholders
   - Observe/record how they interact with **it**
   - Take note of problems, usability issues
4. Make updates to your design & **prototype**
5. Repeat as necessary

Usability testing

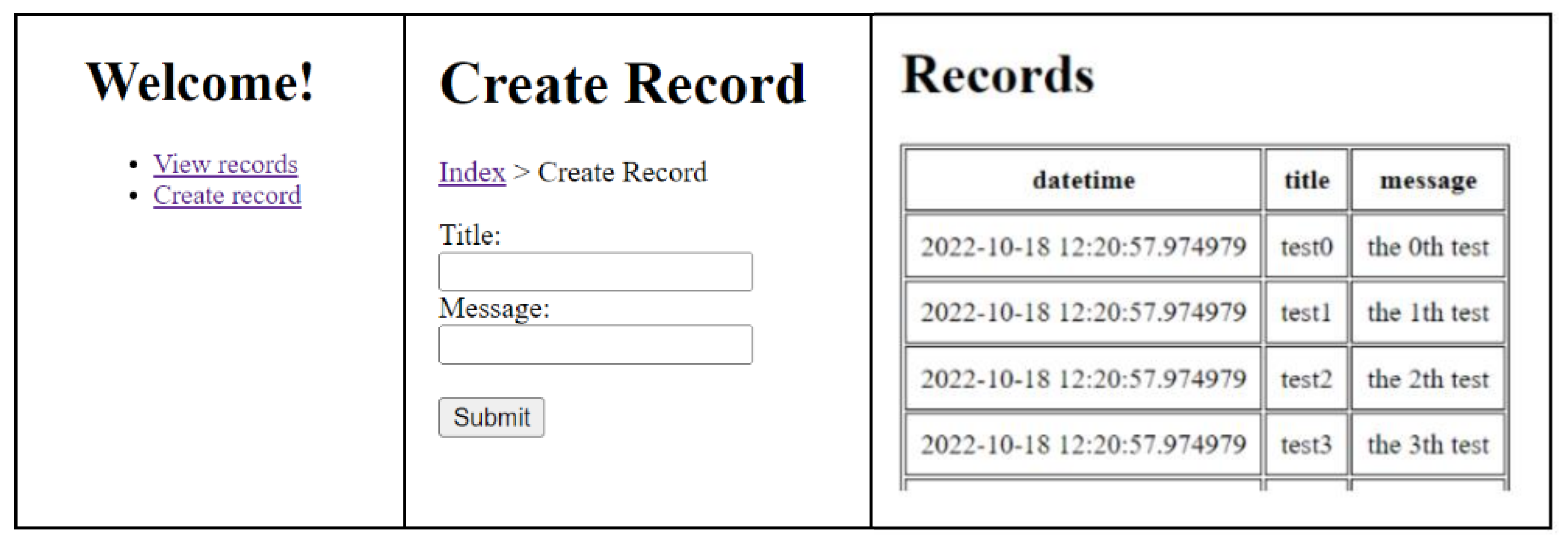
- Involves monitoring behavior of real users as they complete specific tasks with product
- Works well at various stages of development
    - From beginning of design process (can accompany user research)—using low-fidelity
    - To after product release—using product itself
- Often conducted repeatedly, different stages

UI for your project

- **UI** will depend on your project and app **data**
  - Data modeling discussed in upcoming lecture
- Start with basic **UI** that allows simple operations on basic app **data**
  - Basic UI: **Plain HTML**
  - Basic data model: **text-based data**
- We build basic **UI** together later in semester

Initial UI for project: Basic design

- Index page/table of contents
- Way to add & review data



Welcome!

- View records
- Create record

Create Record

Index > Create Record

Title:

Message:

Records

datetime	title	message
2022-10-18 12:20:57.974979	test0	the 0th test
2022-10-18 12:20:57.974979	test1	the 1th test
2022-10-18 12:20:57.974979	test2	the 2th test
2022-10-18 12:20:57.974979	test3	the 3th test

Your UI can be more creative

- Can **prototype** other UI and interaction ideas
  - mobile / responsive UI
  - mobile interactions: swipes, pinches, etc.
  - natural-language based interactions (voice, text)
- You can consider other **data** types, such as…
  - multimedia (pictures, video, etc.)
  - maps
- May not have time to **implement** this term

Outline

- UX and UI design
  - Prototyping
  - Usability testing
  - UI for your project
- System design
  - Architecture diagrams
  - Interaction Diagrams

Outline

- UX and UI design
  - Prototyping
  - Usability testing
  - UI for your project
- System design
  - Architecture diagrams
    - Client-server model
    - Microservices architecture
  - Interaction Diagrams

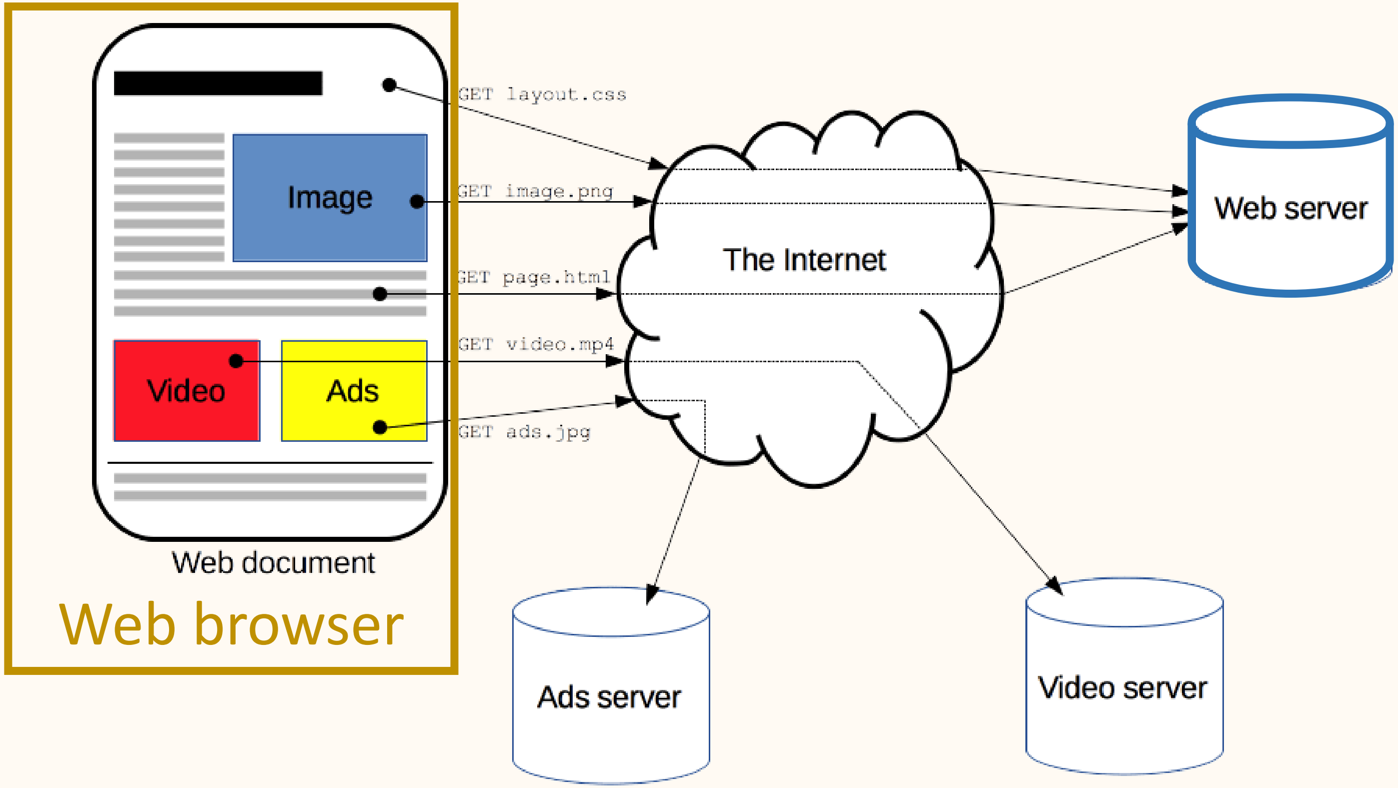
Client-server model

- Application structure where **server** provides resources & services requested by a **client**
- Examples
  - **Web browser** and **remote server** communicate using HTTP, HTML & CSS
  - **Email client** and **mail server** communicate via SMTP and POP3/IMAP
  - **FTP client** and **file server** communicate using FTP

Modern web browsers can also act as clients for these

Can be multiple clients/servers

- Web example

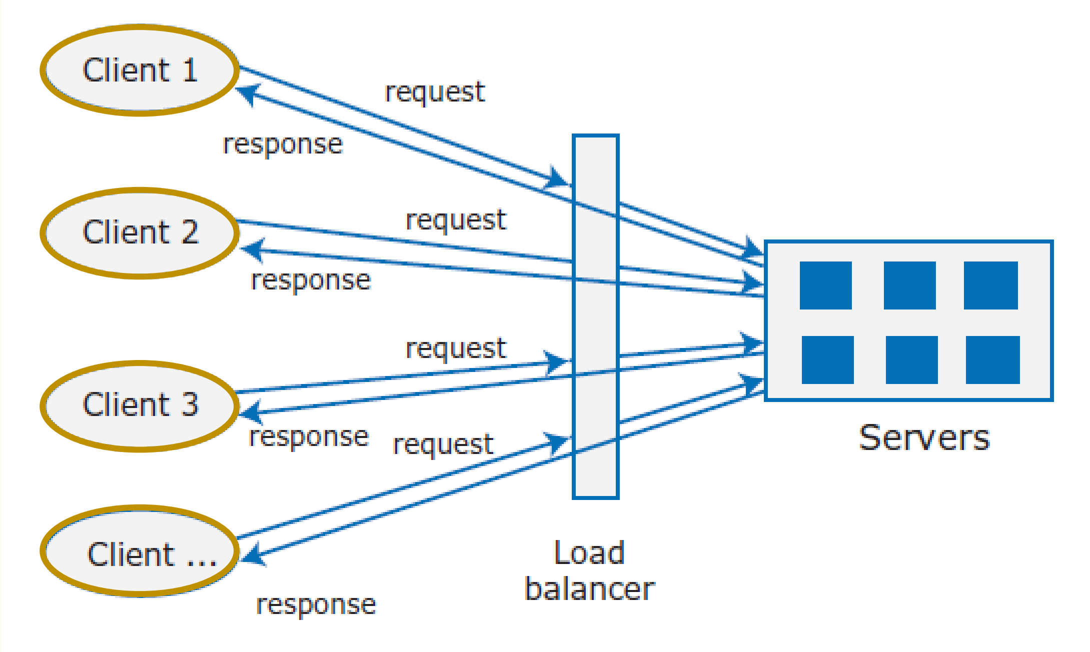


Web technologies repurposed

- Client could be **app** on laptop or phone (instead of web browser)
- Server could be any **program** using HTTP to communicate
- Could still involve HTML/CSS to specify user interfaces (even if not a web page)

General client-server architecture

- Often more complicated now than used to be
- We'll explore in more depth in a couple of weeks (Module 2.1)

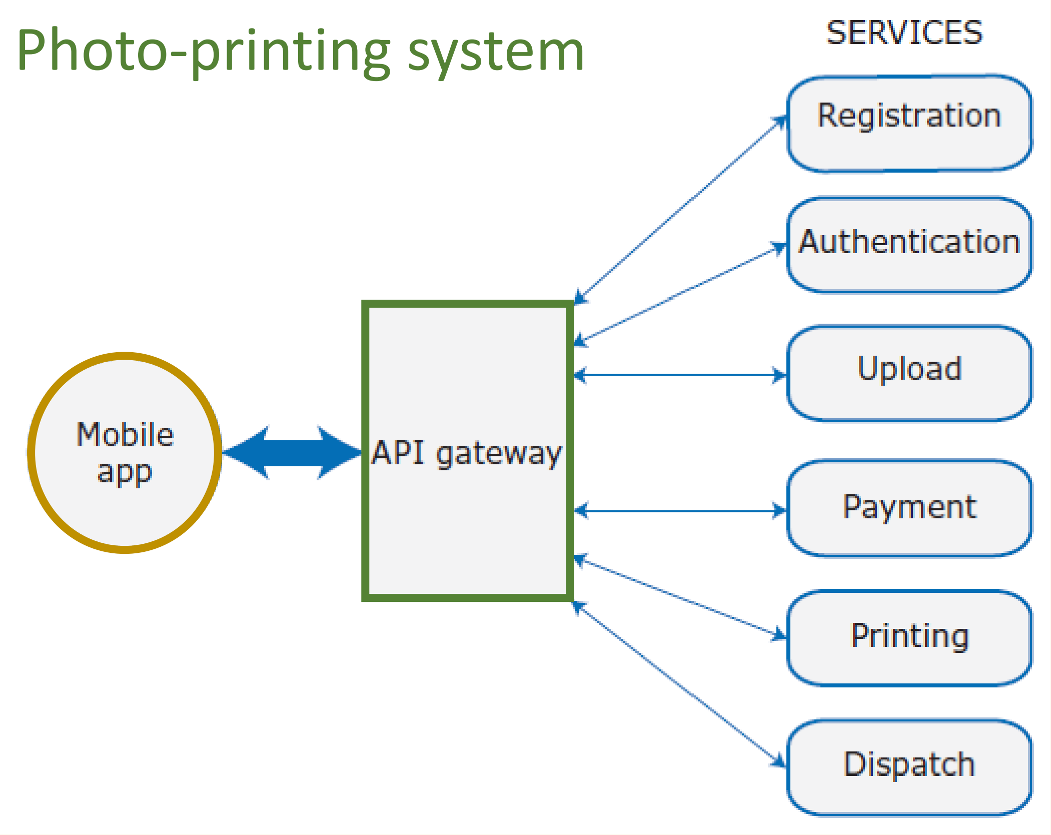


Microservices architecture

- Application structure where **system** is developed as collection of separate **service**s
  - Each **service** developed/deployed independently
  - Each has own API to communicate with **clients**
- API = Application Programming Interface
  - "Contract" between **service** and its **clients**
  - Often uses HTTP requests & responses

Microservices Architecture

- **API gateway**
  - accepts all API calls
  - aggregates the various **services** required to fulfill them
  - returns the appropriate result
- More in Module 2.3

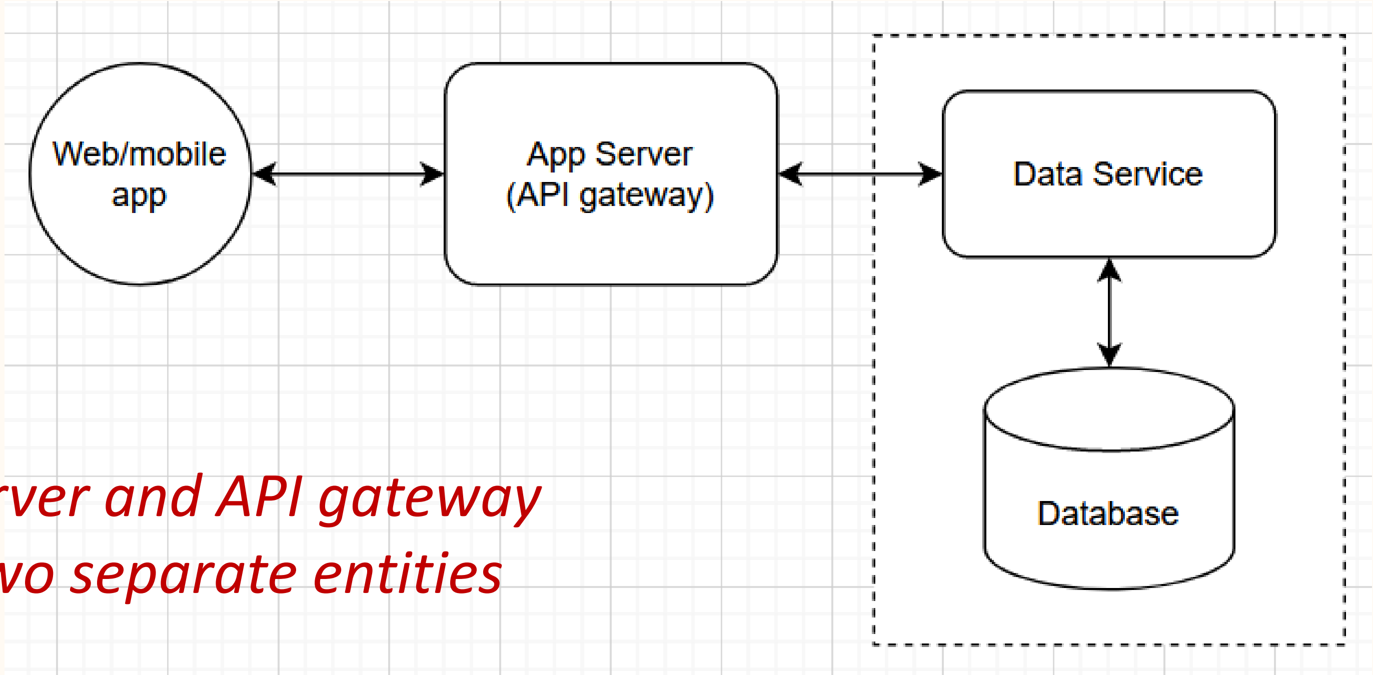


Architecture diagram

- Maps out components of a software system
- Shows the general structure of the system
  - Software elements/components
  - Associations, limitations, boundaries between elements (especially communication)
- Like the diagrams on prior few slides

Initial architecture for project

Architecture diagram for MVP for class project



Note: app server and API gateway are usually two separate entities

Outline

- UX and UI design
  - Prototyping
  - Usability testing
  - UI for your project
- System design
  - Architecture diagrams
  - Interaction Diagrams

What do we do with stories?

- Prerequisites:
  - Well-defined user-story (hopefully you created)
  - Prototype UI designs (provided)
  - Initial system architecture (provided)
- Steps to start forming a system design
  - Expand user story to include ordered steps
  - Consider which system elements do which steps
  - Create interaction diagram to capture that

Writing expanded stories

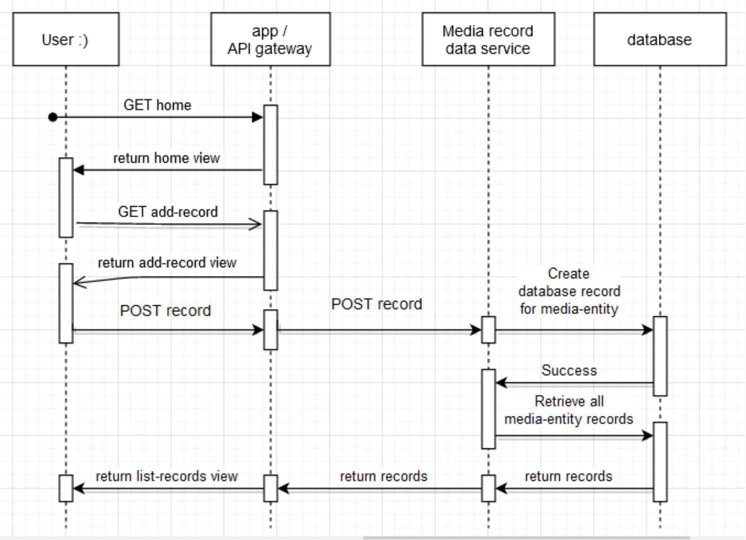
- Answer these questions:
    - What do we expect the user to do?
    - How does the system respond to each user action?
- Record the answers in the order we might expect them to happen

Example

- Persona(s): David, media consumer
- User story: *As a media consumer, I want to be able to add basic information about the media I consume (e.g. title, media type, author, etc.), so that I can keep a record of what I've consumed*
- Steps / interactions
  - David starts application
    - show home screen
  - David clicks "add record"
    - show form to add information
  - David types in information and clicks "submit"
    - propagate data to data service / DB
    - retrieve all records
    - redirect to records listing
- Tasks
  - UX designs / user testing
  - backend / data management
  - frontend / UI development
  - feature / integration testing
- Definition of done
  - a user can add records to the database and view a list of all records.
  - all tasks have been completed (developed, tested, reviewed, and validated)

Sequence diagrams (interaction diagrams)

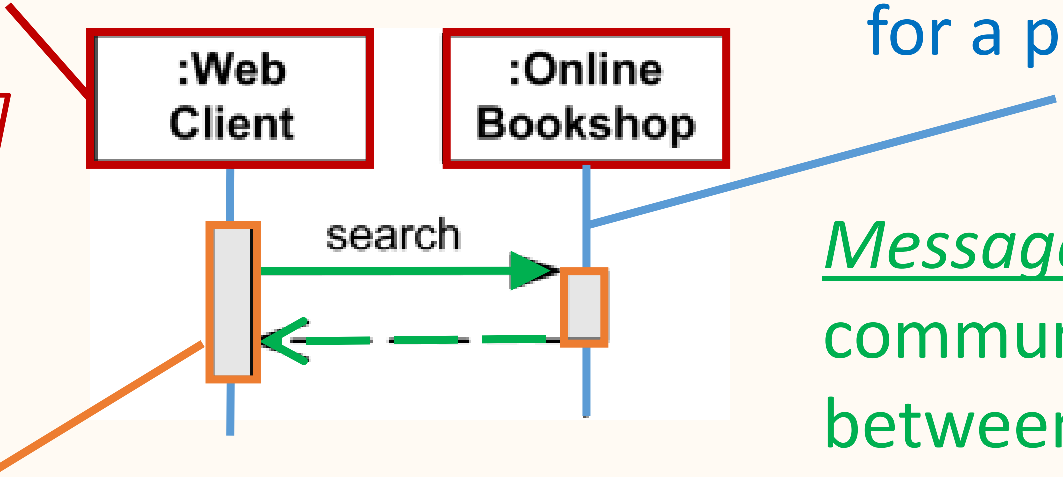
- Shows the order of messages passed between user(s) and elements of system to complete a particular task/story
- Look something like this →



Main diagram components

Participant: object or entity that acts in the diagram
[name]:[type]

Lifeline: axis representing the progression of time for a participant

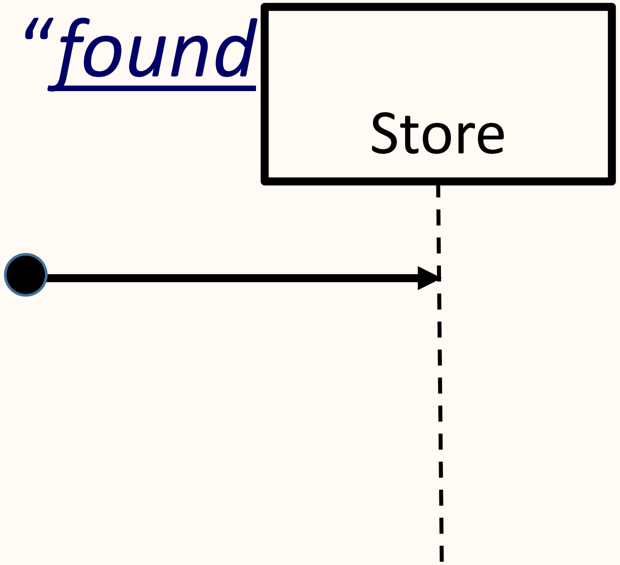


Message: communication between participants

Execution specification bar: owner of focus of control during execution

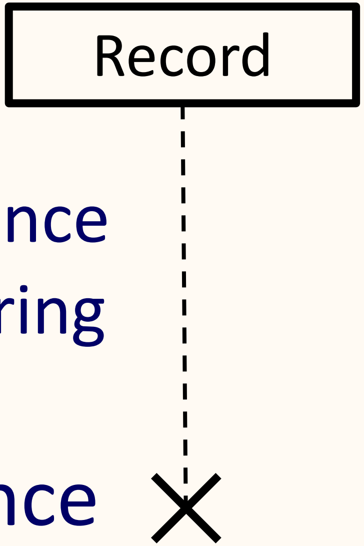
Participant

- Object or entity that acts in the diagram
- Unlabeled message at left usually starts scenario (source of unattached "*found message*" arrow)



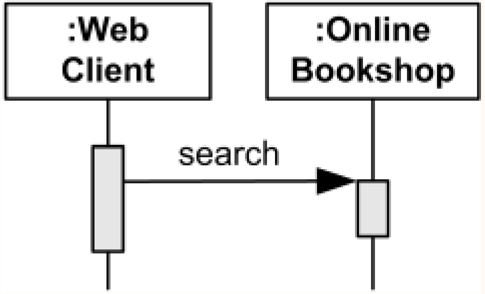
Lifeline

- Can be solid or dashed (no difference)
- Start at participant box
  - At top of diagram if exists at start of sequence
  - Lower in diagram if participant created during sequence
- Ends at large X if deleted during sequence
- Ends at bottom (without X) if exists at end of task/story

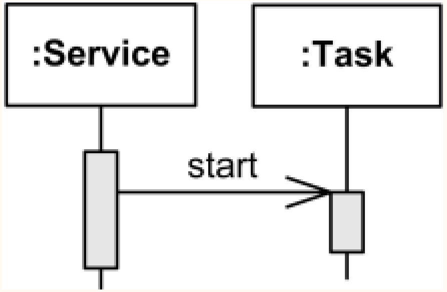


Message basics

- Represents method call or communication
- Name, arguments above arrow
- Synchronous
  - Call waits for response
  - Closed arrow

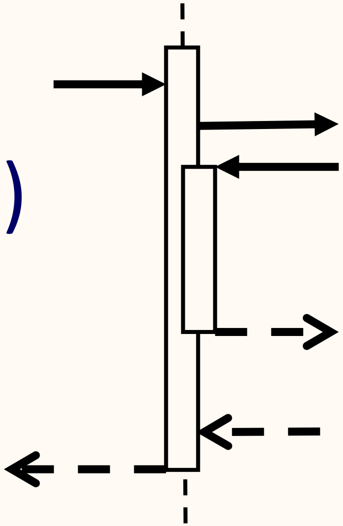


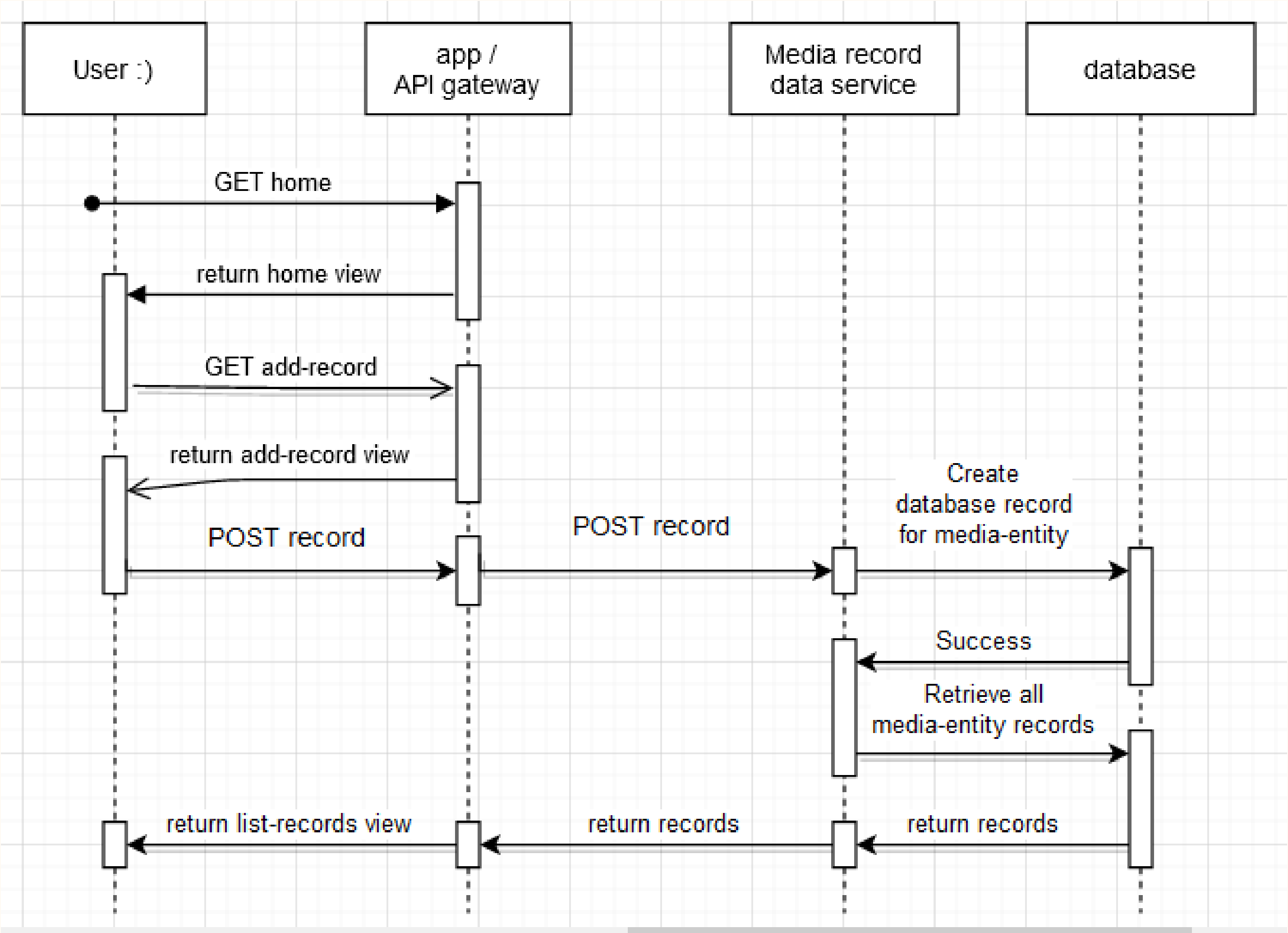
- Asynchronous:
  - Continues (in separate thread)
  - Open/stick arrow



Execution Specification Bar

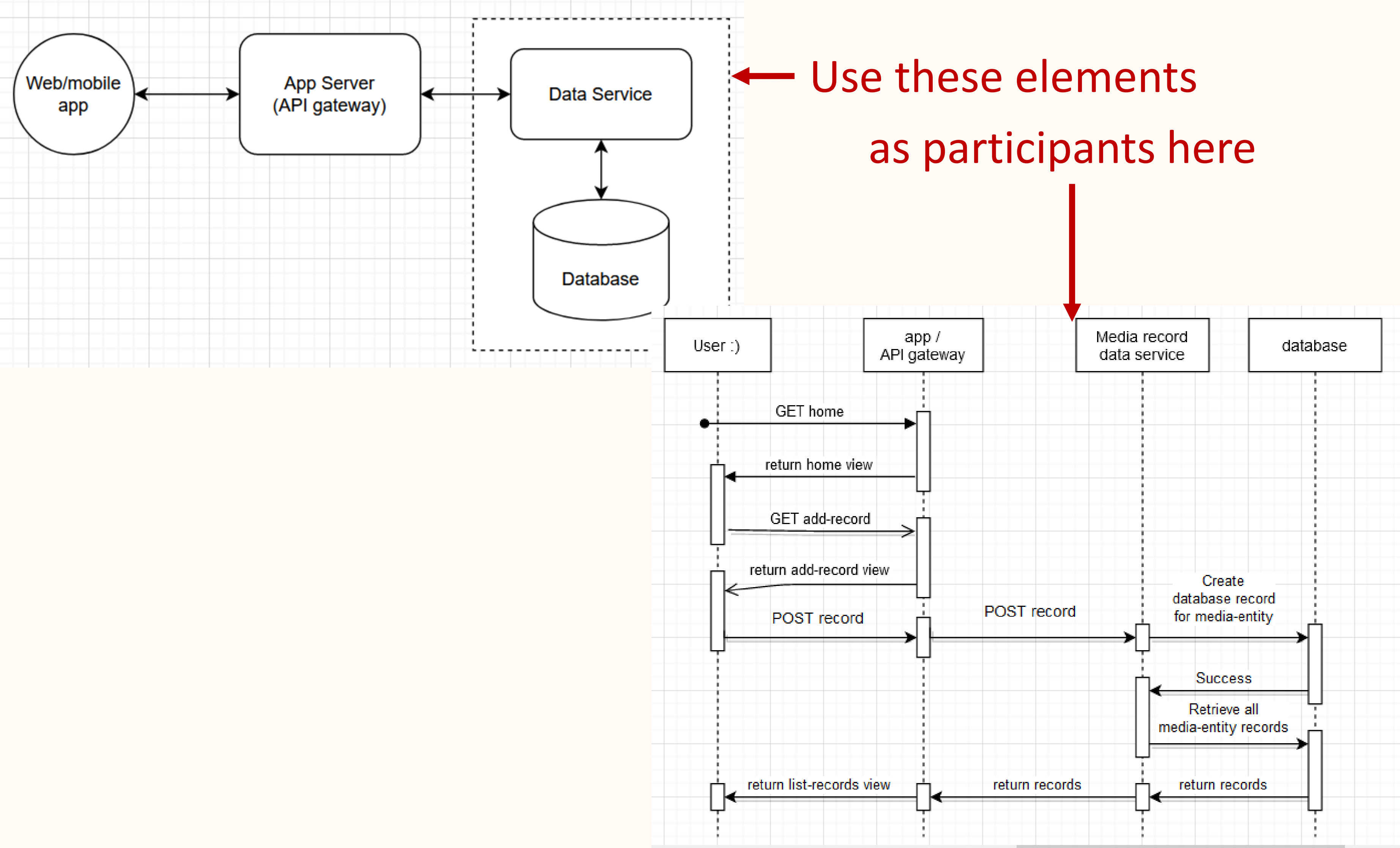
- Also called *Activation*
- Not always drawn (esp. in quick sketch)
- Thick box over lifeline indicating when a method of object is on the call stack
  - Method is running at the time
  - Method is waiting for another call to finish
  - Nested activations indicate call back to same object (like when called object asks for more info)




User :)
app /
API gateway
Media record
data service
database
GET home
return home view
GET add-record
return add-record view
POST record
POST record
Create
database record
for media-entity
Success
Retrieve all
media-entity records
return list-records view
return records
return records

Creating sequence diagrams

- Determine which elements are involved
  - Users, external systems
  - System software components
    - Architecture diagram shows major components
    - Could be broken down further, if helpful
- Determine what requests and information are being passed between elements
- Order them in diagram top to bottom



Use these elements as participants here

Lab 2b—user interactions

- See lab document
- As a group, choose a few high-priority stories
  - Make sure they're proper size/decomposed first
  - Develop expanded user stories (size of group – 1)
    - What steps must user take to achieve their goals?
    - What interactions occur within the system?
- Each individual
  - Submit one (different) diagram
  - Submit diagram to repo for team use later