Object Oriented System Analysis and Design

Chapter Six: Determining how to build your system: System Design

Ewnetu E. (MSC.)

June 9, 2023

Contents

- System design
- System design process
- Object oriented design
- Architectural design
- Class, Database modeling and
- Deployment diagrams/modeling

System Design

- System design: mainly focuses on the data structures, and components necessary to implement the system.
- System design process: transforms the analysis model into design model/system design.
  - Is a process that focuses on decomposing the system into manageable parts, and then collecting the decomposed components to work in collaboration for a desired goal.
- Like analysis, system design is an evolutionary and iterative activity.
- We need to ensure that the system design model is correct, complete, consistent, realistic, and readable.
- The system design model will be correct if the correct analysis model can be fully mapped to the system design model.

...Cont'd

- During design we will:
  - Refine the analysis and system design models.
  - Make implementation decisions, especially, object design serves as the basis of implementation.
  - Fully define the main classes in the system.
  - Evaluate implementation alternatives and chosen algorithms.
  - Optimize access paths and controls to data during external interactions.
  - Adjust class structure and associations to increase inheritance.
- As a result, the design model can be partitioned into sets of classes such that they can be implemented by individual developers.

...Cont'd

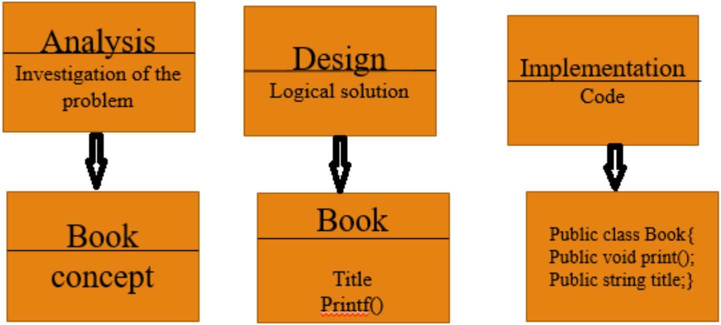
- A software design specifies how a program will accomplish the specified requirements.
- That is, a software design determines:
    - How the solution can be broken down into manageable pieces?
    - What each piece will do?
- Design transforms requirements into:
    - An architecture diagram
    - Subsystems, modules and their relationships
    - Detailed system specification

From Analysis to Design

- Each element of the analysis model provides information that is necessary to create design models.
- The data/class modeling transforms analysis classes into design classes along with the data structures & algorithms required to implement the software.
- Architectural styles and design patterns help to analyze and refine the requirements defined for the system.
- The interface design describes how the software communicates with systems that interoperate with it and with human beings that use it.

From Analysis to Design and implementation

- Fig: from analysis to design and implementation (what we do at each stage)



Analysis

Investigation of the problem

Book

concept

Design

Logical solution

Book

Title

Printf()

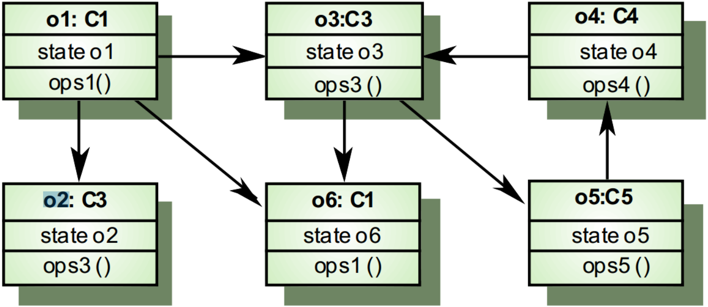
Implementation

Code

```
Public class Book{
Public void print();
Public string title;}
```

Interacting Objects

- Conceptually, objects communicate by message passing.
- Messages are often implemented by method calls.
- Name = method name.
- Information = parameter list.



OOD Activities

- Identification of existing components
- Full definition of associations
- Full definition of classes
- Specifying the contract for each component
- Choosing algorithms and data structures
- Identifying possibilities of reuse

Purpose of Design

- To transform customer requirements, business needs, and technical considerations to a product or system.
- The design model provides detail information about the software's architecture, interface, classes, objects, nodes and components.
- The design model can be assessed for quality and be improved before code is generated and tests are conducted.

Architectural Design

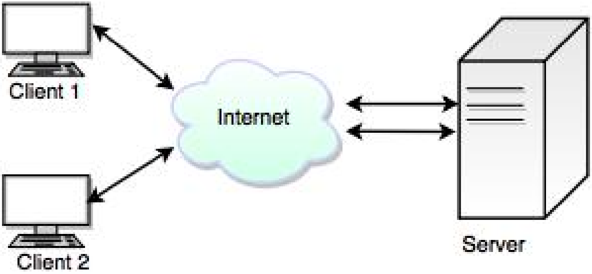
- An early stage of the system design process.
- Represents the link between specification and detailed design.
- Often carried out in parallel with some specification activities.
- It involves identifying major system components and their communications.

Architectural Types (layered approach)

- "Tier" can be defined as "one of two or more rows, levels, or ranks arranged one above another".
- 1-Tier Architecture:
  - The simplest, single tier with single user.
  - Is an equivalent of running an application on a personal computer.
  - All the required components to run the application are located within a single machine.
  - User interface and data storage are all located on the same machine.
  - They are the easiest to design, but the least scalable.
  - Because they are not part of a network, they are useless for designing web applications.

...Cont'd

- 2-Tier Architectures:
  - Supply a basic network between a client and a server.
  - For example, the basic web model is a 2-tier architecture.
  - A web browser makes request from a web server, which then processes the request and returns the desired response, in this case, web pages.

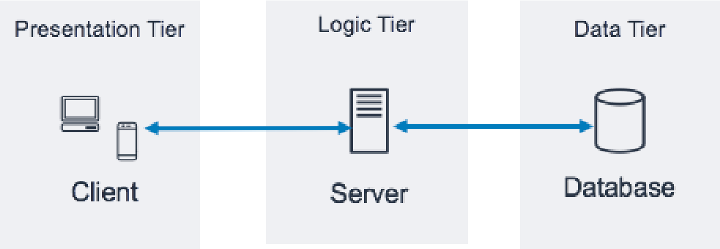


...Cont'd

- 3-Tier Architecture:
    - An architectural type, where an application consists of 3 hierarchically ordered subsystems.
    - So the 3 layers are commonly known as:
    - User interface: Presentation Layer(PL/UI)
    - Middleware: Business Logic Layer(BLL) &
    - Database system: Data Access Layer(DAL).
- It is most commonly used to build web applications.
- This approach separates business logic from display and data.
- The middleware subsystem services data requests between the user interface and the database subsystem.

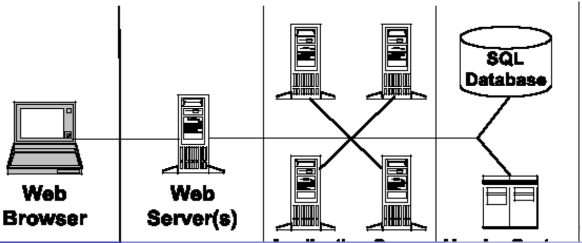
...cont'd

- Three-Layer architectural types are often used for the development of Websites:
  - The web browser represents the user interface.
  - The web server serves requests from the web browser.
  - The database manages and provides access to the persistent data.



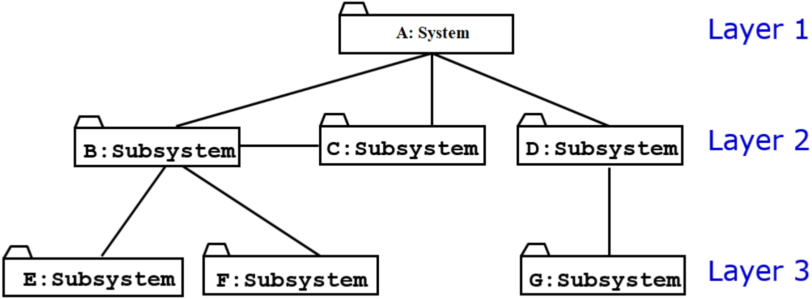
...Cont'd

- 4-Layer-architectural type (4-Tier Architectures): are usually used for the development of electronic commerce sites.
- The layers are
  - The web browser, providing the user interface.
  - A web server, serving static HTML requests.
  - An Application server, providing session management (for example the contents of an electronic shopping cart) and processing of dynamic HTML requests.
  - A back end database, that manages and provides access to the persistent data.



System Decomposition

- System decomposition begins by decomposing the system into cohesive, well-defined subsystems.
- Sub-systems are then decomposed into small sub sub components.
- Sub-system: collection of classes, associations, operations, events and constraints that are closely interrelated with each other.
- In UML subsystems are modeled as small packages.



...Cont'd

- Subsystem decomposition: is the identification of subsystems, services, and their relationship to each other.
- Sub system decomposition is a very essential task in system development.
- It helps project managers to assign system components/modules to the available team members.
- It is also advantageous for detailed system understanding by the individuals that take part in the system development.

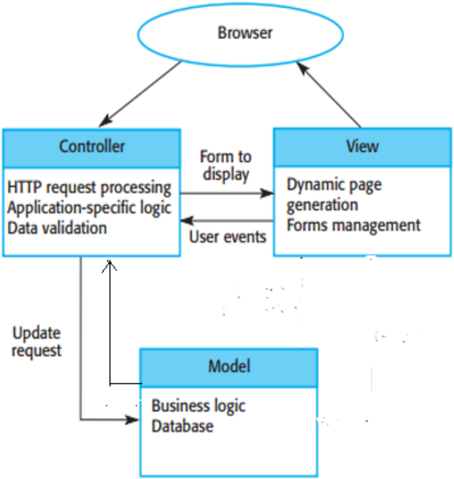
Architectural Styles

- Model/View/Controller
- Client/Server
- Peer-To-Peer
- Service-Oriented Architecture (SOA)

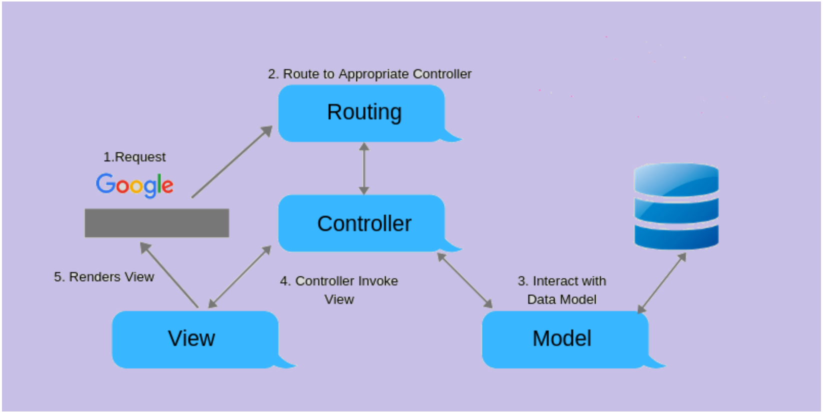
Model-View-Controller Architectural Style

- Subsystems are classified into 3 different types
- Model subsystem:
  - Responsible for application domain knowledge
  - Used for keeping persistent data
- View subsystem:
  - Responsible for displaying application domain objects to the user
  - Used for visualizing/viewing the persistent data
- Controller subsystem:
  - Responsible for sequence of interactions with the user and notifying views of changes in the model
  - Used for managing user interactions with the system
  - Helps to make communication between front end and back end.

Web application architecture using the MVC

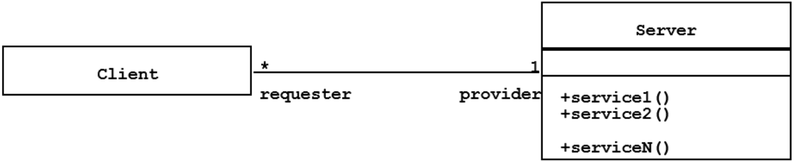


...Cont'd


2. Route to Appropriate Controller
Routing
1.Request
Google
Controller
5. Renders View
4. Controller Invoke View
3. Interact with Data Model
View
Model

Client/Server Architectural Style

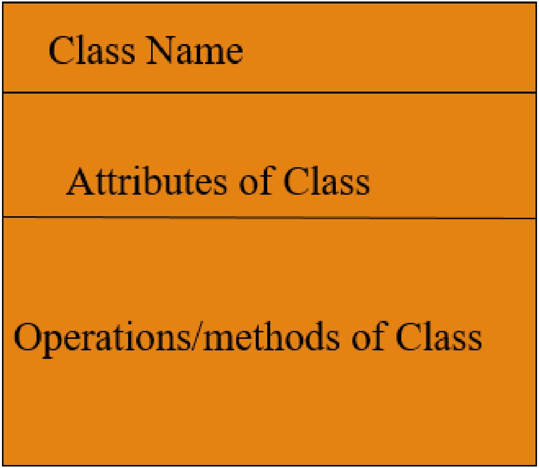
- One or many servers provide services to instances of subsystems, called clients.
- Each client calls on the server, which performs some service and returns the result.
  - The response in general is immediate.
  - End users interact only with the client.



Client/Server Architectures

- Often used in the design of small scale database systems
  - Front-end: user application (client)
  - Back end: database access and manipulation (server)
- Functions performed by client:
  - Input from the user (customized user interface)
  - Front-end processing of input data
- Functions performed by the database server:
  - Centralized data management
  - Data integrity and database consistency
  - Database security

Class Modeling

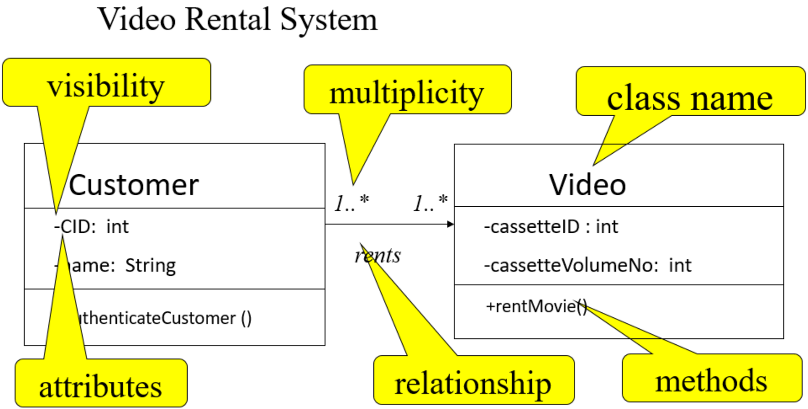


Class Name

Attributes of Class

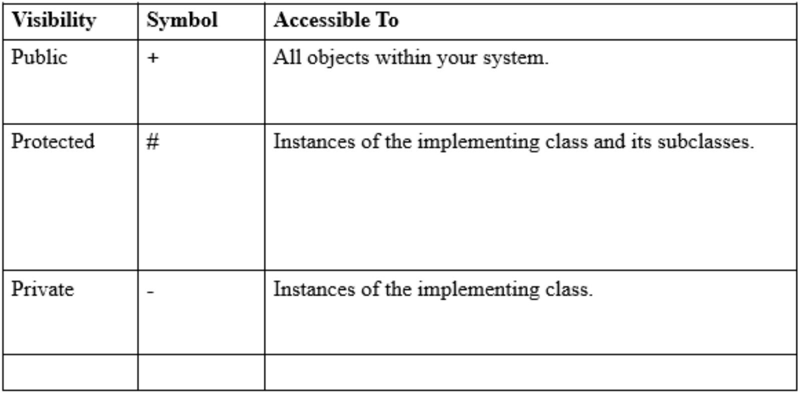
Operations/methods of Class

Example of a Class Diagram



Visibility of Attributes and Operations

- Relates to the level of information hiding to be enforced



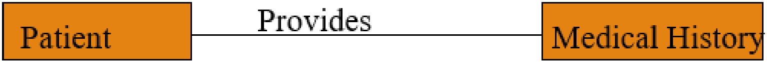
Visibility	Symbol	Accessible To
Public	+	All objects within your system.
Protected	#	Instances of the implementing class and its subclasses.
Private	-	Instances of the implementing class.

Applying Design patterns effectively

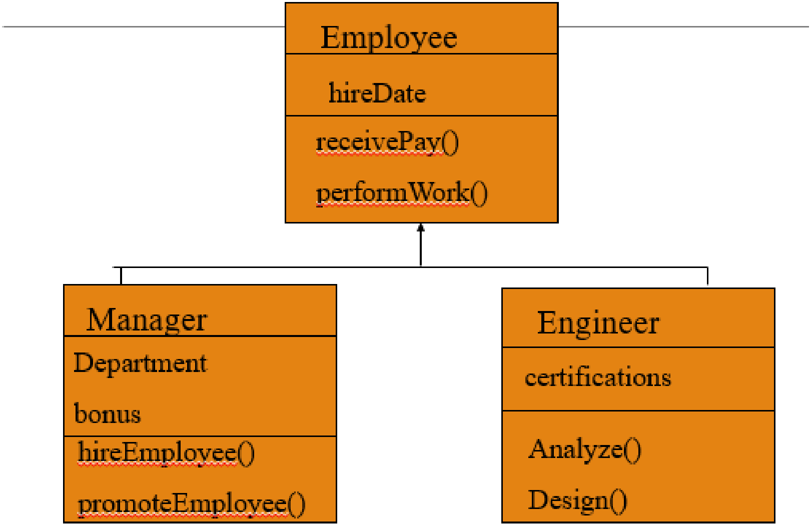
- In class diagram modeling you have to select the appropriate design pattern that fits with your system.
- Read in detail about design patterns that we have seen in chapter two briefly!

Relationships among Classes

- Represents a connection between multiple classes.
- A bidirectional semantic connection between classes
- Guideline:
  - Drawing line between classes
  - Name of relationship
  - Role that classes play in the relationship



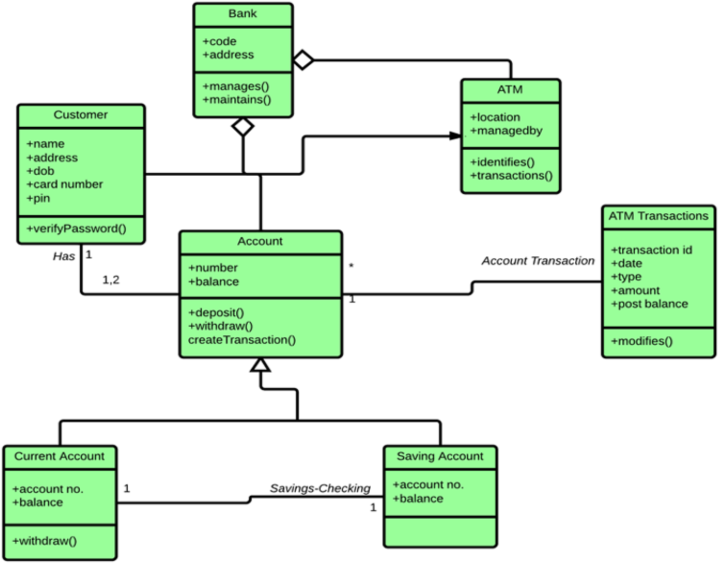
...Cont'd


Employee
hireDate
receivePay()
performWork()
Manager
Department
bonus
hireEmployee()
promoteEmployee()
Engineer
certifications
Analyze()
Design()

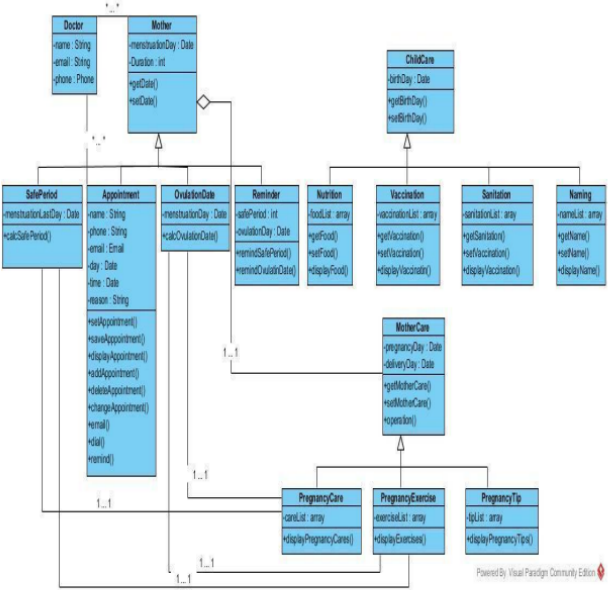
Multiplicity

- Denotes the minimum number.. maximum number of instances
  - Exactly one 1
  - Zero or more 0..* or 0..m
  - One or more 1..* or 1..m
  - Zero or one 0..1
  - Specified range 2..4
  - Multiple, disjoint ranges 1..3, 5

Class Diagrams to Design a System



...Cont'd



..
Doctor
-name : String
-email : String
-phone : Phone
Mother
-menstruationDay : Date
-Duration : int
+getDate()
+setDate()
..
ChildCare
-birthDay : Date
+getBirthDay()
+setBirthDay()
SafePeriod
-menstuationLastDay : Date
+calcSafePeriod()
Appointment
-name : String
-phone : String
-email : Email
-day : Date
-time : Date
-reason : String
+setAppointment()
+saveAppointment()
+displayAppointment()
+addAppointment()
+deleteAppointment()
+changeAppointment()
+email()
+dial()
+remind()
OvulationDate
-menstruationDay : Date
+calcOvulationDate()
Reminder
-safePeriod : int
-ovulationDay : Date
+remindSafePeriod()
+remindOvulatinDate()
Nutrition
-foodList : array
+getFood()
+setFood()
+displayFood()
Vaccination
-vaccinationList : array
+getVaccination()
+setVaccination()
+displayVaccinatin()
Sanitation
-sanitationList : aray
+getSanitation()
+setVaccination()
+displayVaccination()
Naming
-nameList : array
+getName()
+setName()
+displayName()
1...1
MotherCare
-pregnancyDay : Date
-deliveryDay : Date
+getMotherCare()
+setMotherCare()
+operation()
1...1
1...1
PregnancyCare
-careList : array
+displayPregnancyCares()
PregnancyExercise
-exerciseList : array
+displayExercises()
PregnancyTip
-tipList : array
+displayPregnancyTips()
1...1
1...1
Powered By Visual Paradigm Community Edition

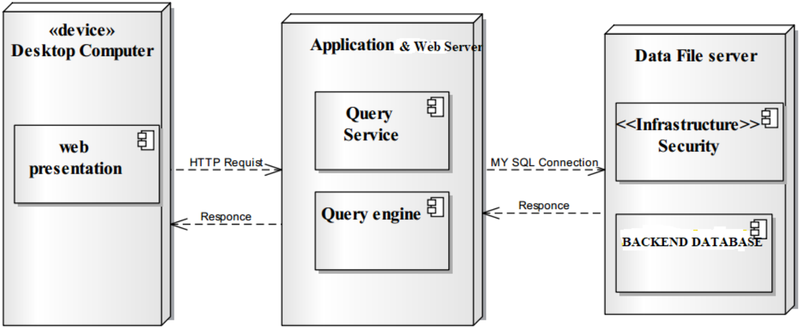
Deployment Diagrams

- Deployment diagrams are kinds of UML models that show the static structure of a system.
- UML deployment diagrams show how software components are physically deployed on processors/ nodes.
- It shows the hardware and software node components in the system.
- Describes the physical resources on which the system component run.
- Software runs on nodes.
- Shows physical arrangement of run-time computational resources such as computer and other interconnected devices.

...Cont'd

- Nodes in a deployment diagram can show hardware devices such as sensors, printers, storage devices, servers etc... that support the runtime environment of a system.
- Deployment diagrams are probably most useful when you are designing an embedded system which works on hardwares.

Example: Deployment Diagram

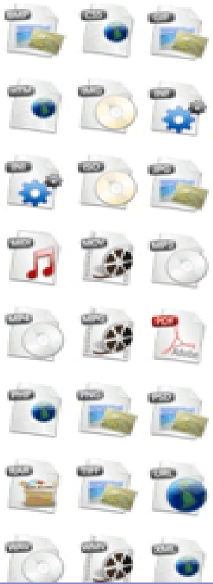


- Read about persistence modeling/diagrams

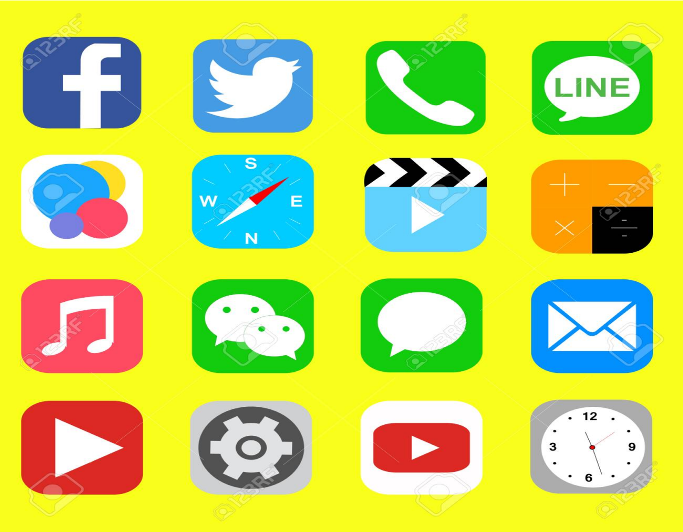
UI Design-Specific Widget Guidelines

- Menu Design
    - Common functions should be easy to reach.
    - $>$ 8 options is too much, to grouping and organize
    - Opposite and dangerous operators should be physically separated to avoid accidents.
- Icon Design
    - Immediately recognizable (small and simple)
    - Easily distinguishable from others
    - Should be very descriptive

Example

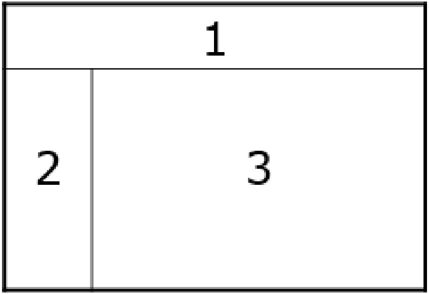


...Cont'd



...Cont'd

- Web UI design specifics
  - Hyperlink nature
  - Menu positions
  - Sidebar placements
  - Key questions web pages should answer at design phase.
    - ★ Where am I?
    - ★ Where can I go?
    - ★ Whats here?



Thank you!!!