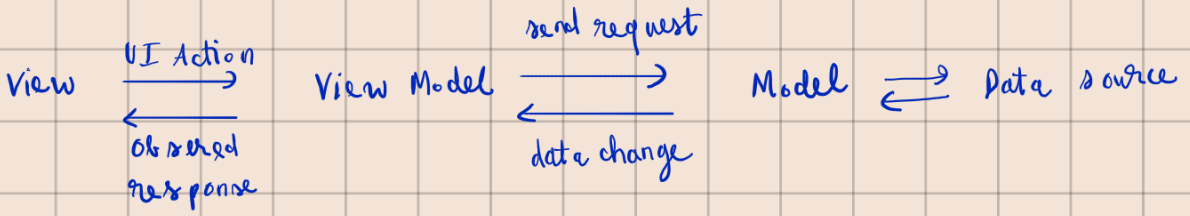
USER STORIES & USER CASES

- Use cases : How sys will interact with end-user, used to doc functional reqs
  - 3 levels : brief, casual, fully-dressed
  - Structure : Actor/user, SuD, Goal, Preconditions, main success scenario, alternate scenarios, post conditions (fully-dressed includes all)
- User stories : simple, informal way to describe user needs
  - Structure : As a [role], I want [feature], so that [benefit]
  - Accept Criteria : describe intial state; what happen in the sys when user interact; result (Given ... → And ... → When ... → Then ... → And...)

SOFTWARE ARCHITECTURE

- Def : Fundamental orga of a sys (the components, their relationships, env, principles governing its design)
  - Decided by early and hard-to-change decisions
- Design : orga of modules, easy-to-change
- Software Architect : great comm skills, use effective mechanisms to repre softwar (can communicate with all skeate holders)
- 4+1 view model : describing the arch of a sys using | logical | dev | Scenarios | process | physical | views |
- Client-Server : services crpd to servers, clients access services via a network
  - 2-tier arch style (Client & Server on 2 diff hardware)
- N-tier : > 2-tier
- Peer-to-Peer : No distinction between clients & servers, computations can be carried by any node in the network
  - Pros : fault + disconnection of node tolerance
  - Cons : malicious node (not safe)
- Blackboard : each specialist knowledge source contribute a partial solution
  - Components : blackboard (contain solution objects), knowledge sources, control component (select & execute knowledge source)
- Layered : layers w/ related funcs, where a layer depends on one beneath it and is inde w/ on top of it
  - Pros : isolation, reusability, separation of concerns
  - Cons : redundant, rigid, complex

- Pipe and Filter : Data Source $\xrightarrow{\text{Pipe}}$ Filter $\rightarrow \dots \rightarrow$ Data Sink
  - Piping : give the output of one command as input to subsequent command
- Event Driven : a component can announce events, others listen and do work
- Model-View-Controller : arch pattern, follow layered appoarch
  - Controller : model update based on user actions
  - View : render model, send user events → controller
  - Model : capture model updates, notifier changes → view
- Model-View-View Model : model and view not connected



- Object Oriented Analysis : C RE phase (not software objects)
  - Gather req : Domain Model (visualization of real-situation objects) (obtained from RE analysis)
    - Domain Model is a UML
    - Derived attb : can be derived from others, preceded by "/"
    - Relationship : Ex : [X] store [Y]
- System Sequence Diagram : show how actors and system collab, capture only one use case. Interactions go from top to down as time increases
  - Glossary : give more details about new terms in SSD
  - Follow DRY agile principle • Fragments : Loop, Alt, Opt, Ref
- Sequence Diagram : models a single scenario, no longer a black box
  - Object lifeline : "- - - -" • Activation bar : a rectangle box repr "active"
  - — message → : wait for receiver • —— message →: don't wait for receiver
- Design Class Diagram : models software classes and objects + relationships
  - visibility : public (+), protected (#), private (-), derived (/)
  - methods : vis + name (para) : return-type
  - ——▷ : inheritance | - - - - -▷ : interface implementation
  - or ——→ : X related to Y | - - - -→ : X uses Y as para
  - ——◇ : whole & part separable | ——◇ : whole & part inseparable
- Attribute Text vs Association Lines