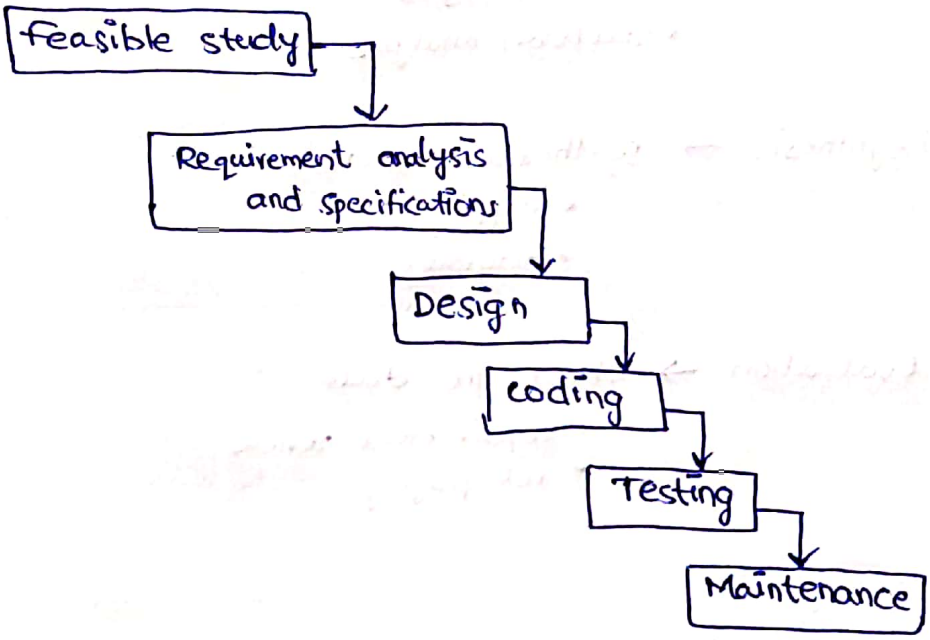
Section-4

Question-1

software process life cycle

* Waterfall Model:



→ it is like a flow of waterfall from top-to-bottom

→ assumes no defects is introduced during development

→ it will take process step-by-step from top-to-bottom

Section - 4

Question - 2

* Model-view-controller pattern

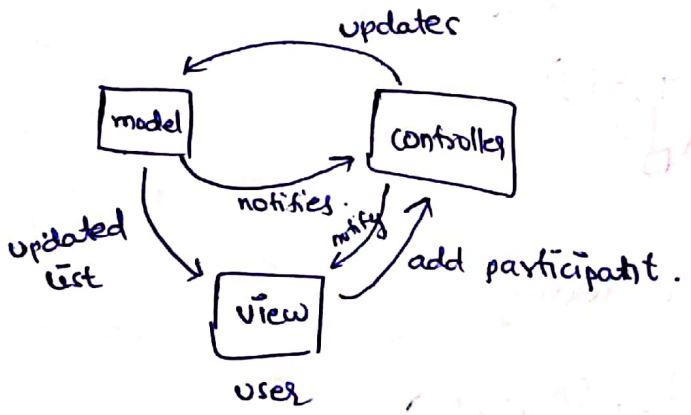
The three components are:

1) Model - This contain classes whose intances are to to viewed and (modified) manipulated.

2) view - This contain object used to render the data from the model in user interface

3) controller - This contains the objects used to control and handle the interaction with the view and the model.

Applications to zoom platform:



Section-4:

Question-3:

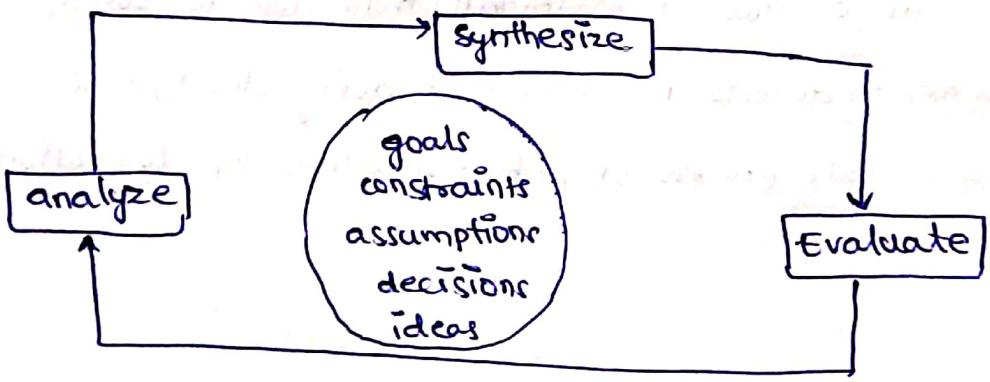
The 3 - categories of design pattern are:

1) Analysis → Analyses the design
   * task analyses
   * Stakholder analyses

2) Synthesis → Synthesises the data
   * Mapping
   * diagrams.

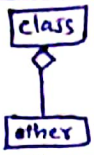
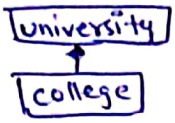
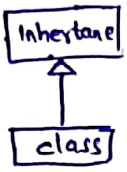
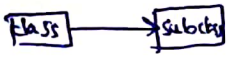
3) Evaluation → check the data
   * requirement review
   * role playing.

Diagram



Section-4

Question-4

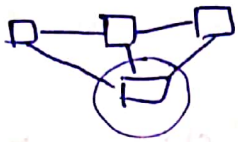
	Symbol	What they mean,
Aggregation		Representing has-a relation
composition		Representing is-a relation
Generalization/ Inheritance		The extension of classes.
Multiplicity		For intervals of the class (range)
Stereotype		For indicating that it is next step.

Section-4

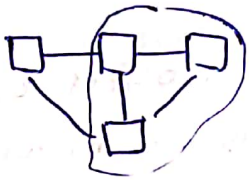
Question-5

* Testing granularity levels.

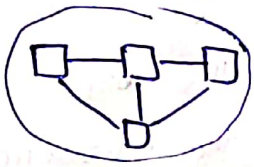
1) unit testing



2) Integration testing

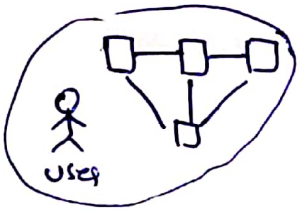


3)

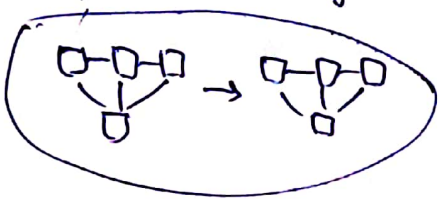


System testing

4) Acceptance testing

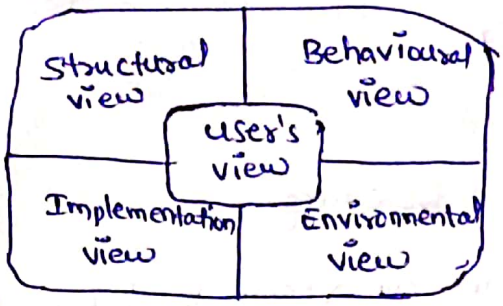


5) Regression testing



Section-4

Question-6



Structural view are represented by
↳ class diagram and object diagram

Behavioural view are represented by
↳ sequence diagram and commucation diagrams

Implementational view are represented by
↳ component Diagram

Environmental view are represented by
↳ Deployment Diagram

Section-5

Question-1

S.O.L.I.D principles are proposed by Robert.C.Martin also known as "Uncle bob".

S → Single Resposible Principle

* Resposibility == Reason for change
* If there are more than one-reason to change this is not followed
* Depend on GRasp (Hard for following)

Ex: profile.

O → open-closed principle

* open for Extension
* close for Modification

Ex: Inheritance
composition

L → Liskov substitution principle

* Let 's' be substring of 'T', then objects of type 'T' can be replaced by object of type 's' without changing of any desirable properties of that program

Ex: Rectangle and square.

I → Interference Seqregation principle

* This ensures that the clients are not be forced to depend upon the interface that they do not use.

Ex: Iphone & Android

D → Dependency inversion principle

* This ensure that the high-level modules should not depend upon low-level modules.
* Both high and low-level modules should depend upon abstraction.

Ex: modules for refresh

Section - 5

Question - 2

Architecture for multiplayer online chess game:

Requirements:

→ wifi

→ chess board design

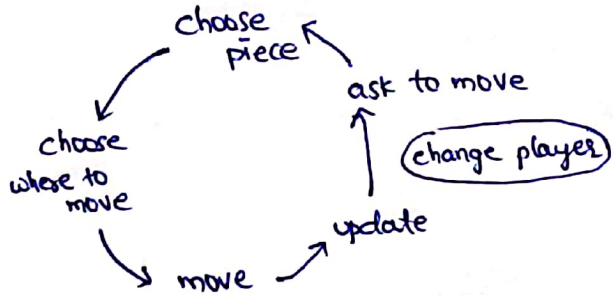
→ pieces (different types) design

→ placement of pieces

→ update the movement

→ 3-D graphics.

Architecture:



→ Asks us to move piece

→ we chose piece & destination where to move piece and move.

→ This movement will be updated to the other player.

→ asks other player to move.

Designing DECISIONS & Rationale:

→ Taking different (types) colored squares (white & black) for chess board.

→ take some shape like (triangle) to form pieces.

→ integrating those square to form chess board and pieces.

→ assigning different roles to the different pieces for movement.

→ giving button press for selecting pieces and their distinations.

Section-6

Question-1 :

Software for judicial system

Requirements:

→ list of pending cases

→ sections

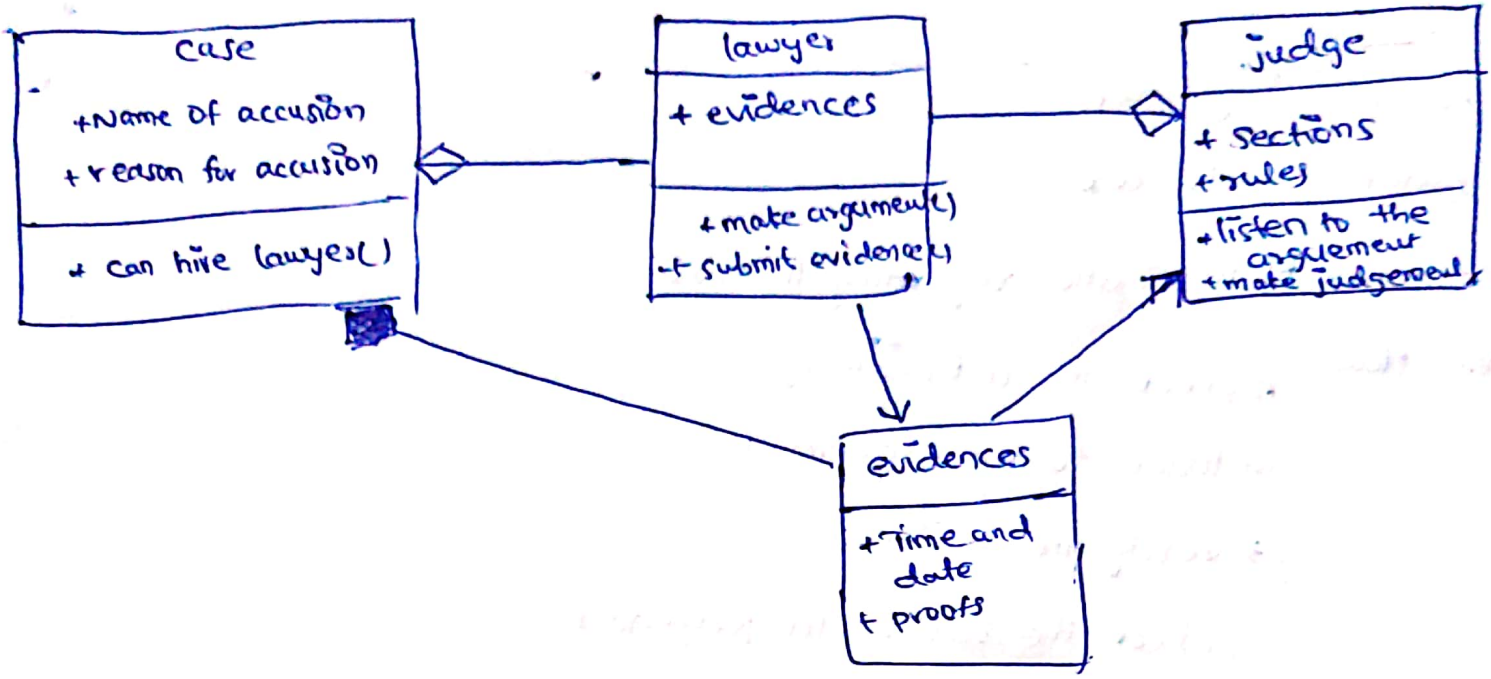
→ list to argume

→ solve case

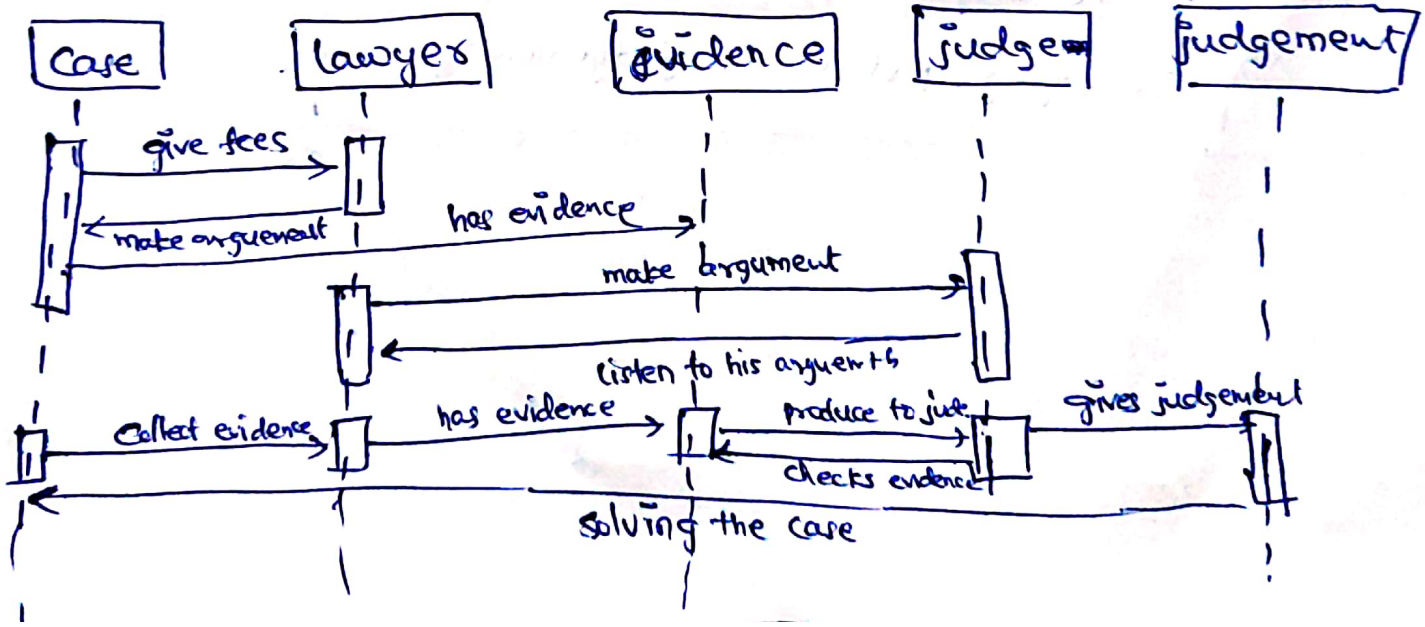
Use case

Number :	EU004
Name :	solve case
pre-requirement:	information regarding the case
Main flow :	1) check the informations 2) listen to the arguement 3) verify the evidences 4) check the sections for judgement 5) give the judgement
Alternate flow :	if the arguements and evidences are not satisfying, postpone the judgement

class diagram



sequences diagram



Design choice & Rationale:

→ make judgement only after listening to the lawyer and verifying evidences if not wrong decisions may comes out.