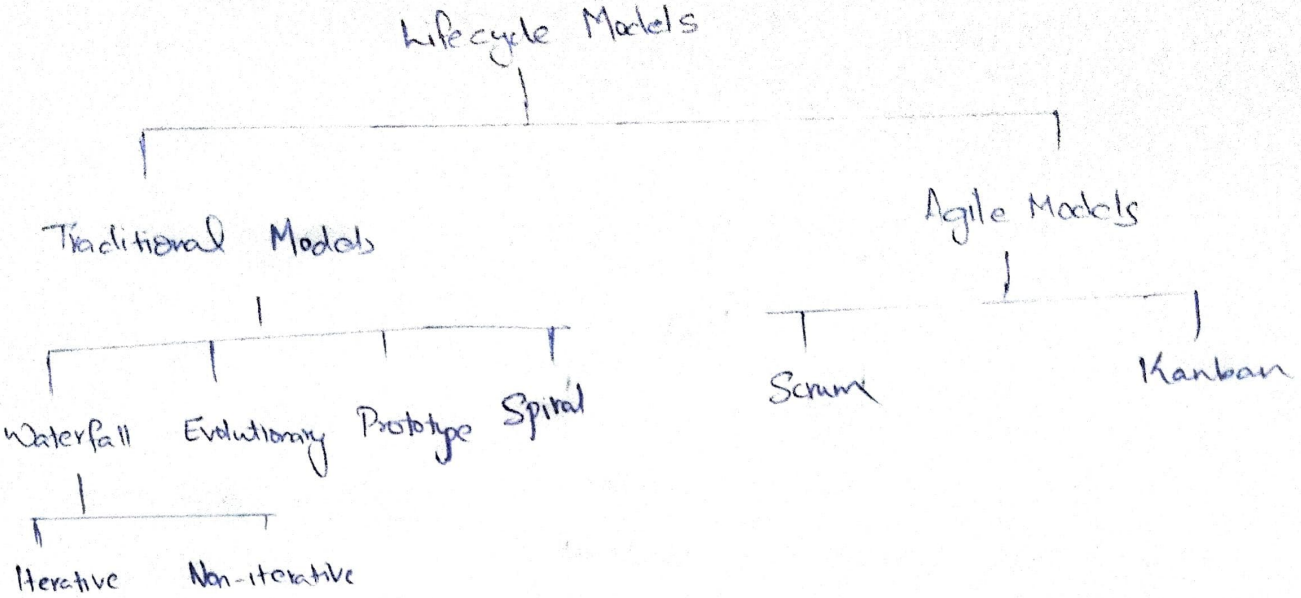
SECTION - 4

Q.1



The agile lifecycle models are mostly used in startups etc. Usually there will be a sprint of weeks or months to release a potentially shippable product. This cycle continues until the product is ready. and daily meetings will be there

Waterfall model is a traditional model of different steps like feasibility study, requirements specification design, coding, testing, etc. There is a high chance the errors to occur in each step and get accumulated.

In evolutionary, problem will be divides into different modules and each modules are consequetively delivered. Good for large systems

In prototype model, a small prototype of the product will be built, based on which the subsequent software is made. It is useful when we do not have much idea about the problem.

SECTION 4

Q.2

Model : It relates to the database schemas, relations etc

Controller : It will execute the business logic based on the request from user and data from the model.

View : It will show the views or pages on the screen depending on the response from the controller.

For Zoom; the Model will be the database schemas consisting of tables for Participant, Meeting, Recordings, Chat, etc.

The controller will be handling the functionalities like chat system, socket connection between participants (broadcasting), recording the session. It can retrieve and update the Model. For example; it can retrieve the participation list or saved chat messages. It can also update the participation list when someone new joins or exits, also when someone messages in the chat box.

The View or frontend will be displaying all the different components on the screen and update them as and when it receives something from the controller. For example; show the updated participation list on frontend when someone new joins, show new message when someone messages, show the participant's video tab when they start

SECTION 4

Q.4

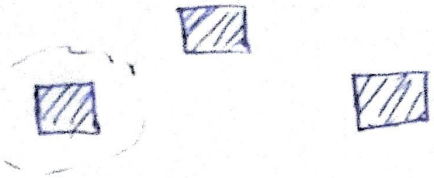
	Symbol	What do they mean
Aggregation		A component is a part of another component but cannot exist independently without it
Composition		A component is a part of another component but can exist independent of the component
Generalisation/ Inheritance.		A component can inherit certain attributes and methods from other components
Multiplicity		It shows the type of relation between two components (One to One, One to Many etc.). It will have a lower limit and upper limit.; denoted by 1...u, If u is infinite, can be denoted by *

END SEM

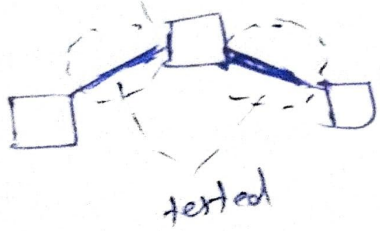
SECTION 4

Q.5

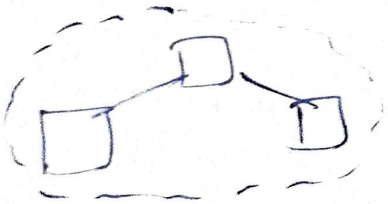
Unit testing - each individual modules are tested



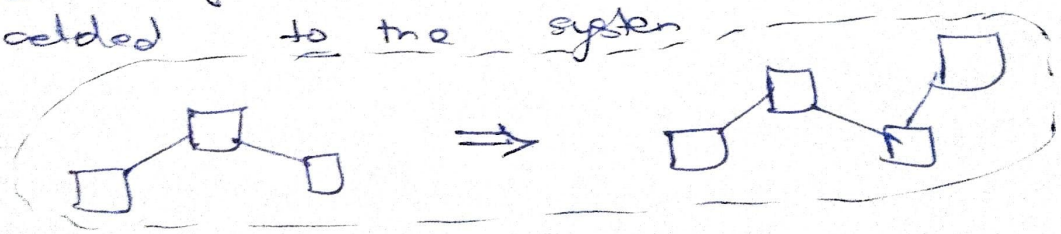
Integration testing - the connections between componet are tested individually



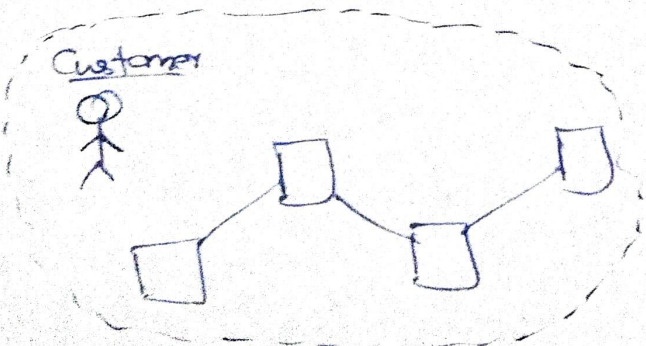
System testing - Entire system is tested based on the requirements specification.



Regression testing - Tested when new componets are added to the system

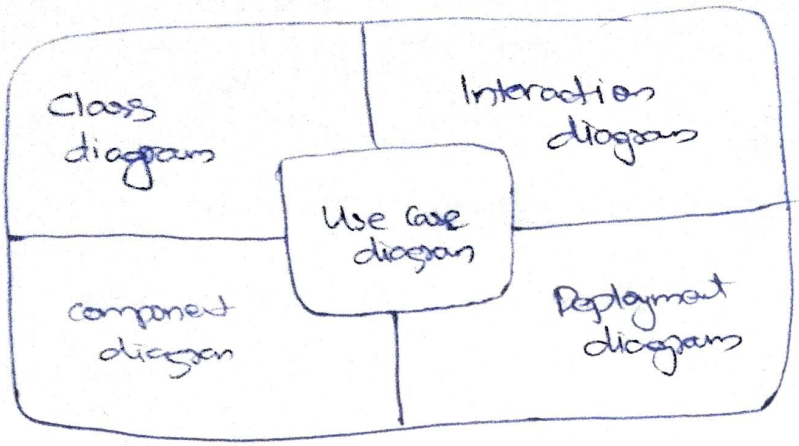


Acceptance testing - It includes the customer as well



SECTION A

Q 6



Use case :- It represent the requirements in the requirement specification

Class diagram :- It shows the different classes used in the system along with their attributes, methods and relationship with other classes.

Interaction diagrams :- They represent the dynamic flow of the system depending on the use case

eg. Sequence diagram.

Deployment diagram :- It shows the deployment architecture of the system like how the nodes (servers) are connected and communicate with each other

Component diagram :- It represent all the different independent components in the system and their connection

SECTION - 5

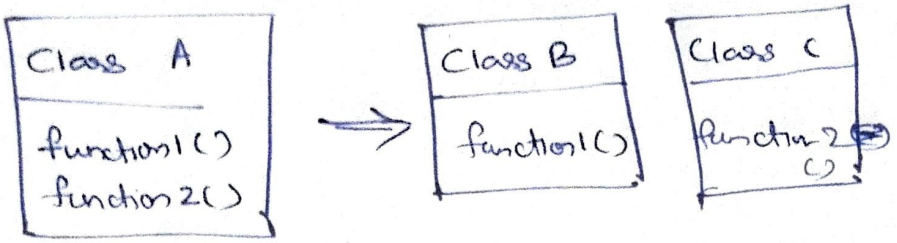
Q:1

SOLID Principle

1. Single Responsibility Principle :

A one class should have only one single responsibility so we need only one reason to change it.

eg:



2. Open close principle :

A class should be open for extension but closed for modification.

```
Class HDD                          Interface Input
{ private read()          ⇒        { private read()
  private write()                    private write()
}                                  }

                                   class USB implements Input
                                   {

                                   }
```

3. Liskov substitution principle

A subtype T of S should be able to be substituted instead of an instance of S.

eg: Suppose a new device USB comes, which has a method refresh, the interface should be modified as.

```
Interface Input
{
    private read()
    private write()
    private refresh()
}
```

4) Interface segregation Principle.

A client should not implement a behaviour that it does not need.

eg: In above example suppose HDD does not need refresh, but it still has to implement refresh method.

```
class HDD implements Input
{
    ========

    private refresh(){
}
}
```

This leads to ~~fat~~ interface so we should remove it ~~and call~~ from the interface and call it from inside the implemented class.

So. USB will become,

```
class USB implements Input
{
    private open()
    { this.refresh()
    ========

    }
    private refresh()
}
```

5. Dependency Independent Principle.

A high level module should not interact with low level modules.

eg. class Transfer (USB a, SSD b)
{
}

This should be changed to

class Transfer (Input a, Output b)
{
}

Interface Input
{
}

Interface Output
{
}

class USB implements Input
{
}

class SSD implements Output
{
}

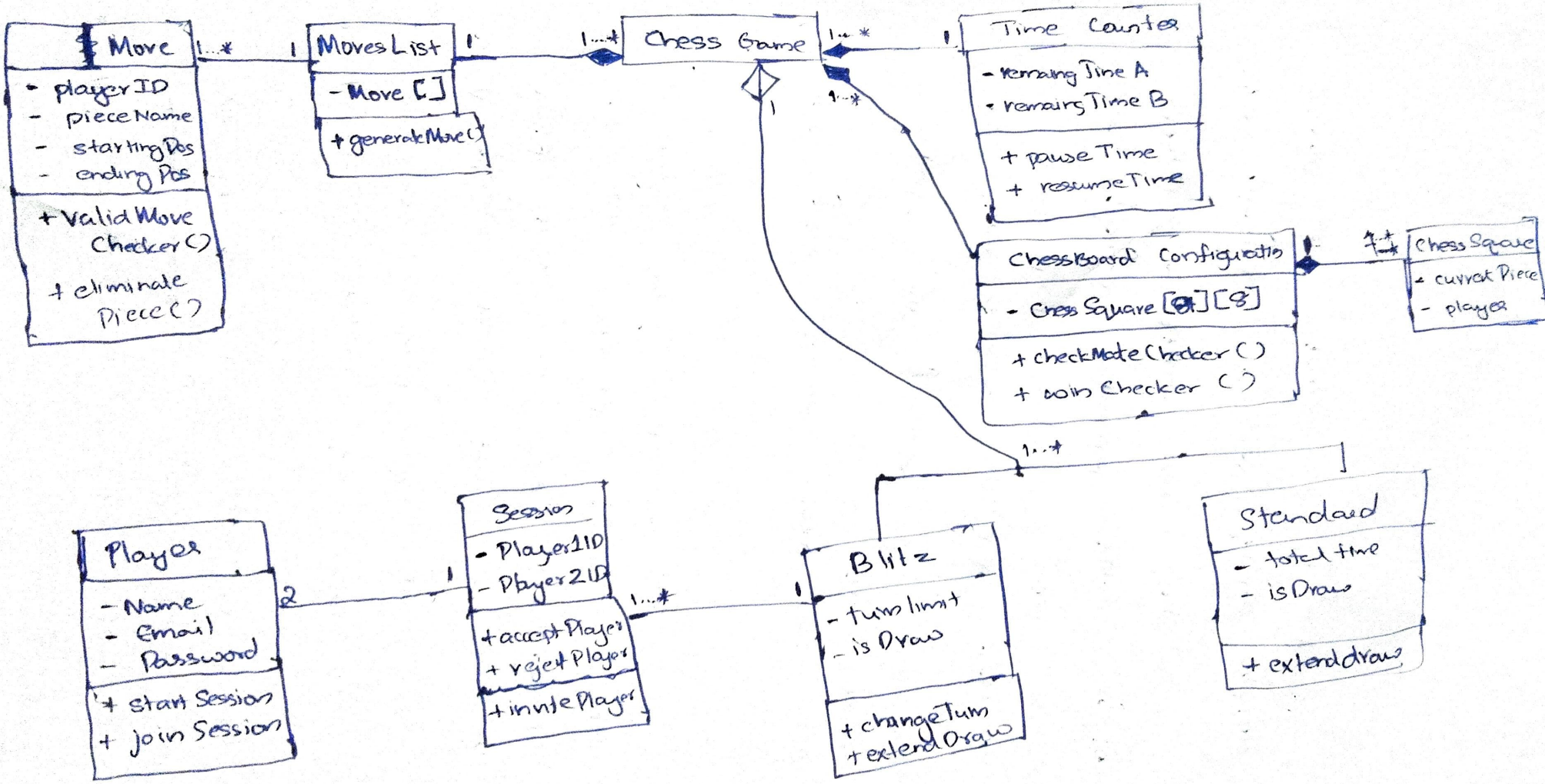
SECTION - 5

Q.2

Requirements

- As a player should be able to start a session of specific type (Blitz, standard etc.) the time per move etc., so that others can join the session.
- A player should be able to join an open session, so that the game can be started.
- A player should be able to make a valid move on the chessboard.
- The game should be able to detect the current chessboard configuration and determine whether a checkmate is present, a user can win etc.
- The game should store the move made by a player, so it can be used by computer to generate intelligent counter moves.
- The time counter should decrement when a particular user's turn is on.
- The player should have option to extend a draw or lose.

The player should have option to accept a draw or reject it.


Move
- playerID
- pieceName
- startingPos
- endingPos
+ validMove Checker()
+ eliminate Piece()
1...*
1
MovesList
- Move []
+ generateMove()
1
1...*
Chess Game
1
1...*
1...*
Time Counter
- remainingTime A
- remainingTime B
+ pauseTime
+ resumeTime
1
1...*
ChessBoard Configuration
- ChessSquare [8][8]
+ checkMateChecker()
+ coinChecker()
1...*
1...*
ChessSquare
- currentPiece
- player
1...*
Player
- Name
- Email
- Password
+ startSession
+ joinSession
2
1
Session
- Player1ID
- Player2ID
+ acceptPlayer
+ rejetPlayer
+ inviteplayer
1...*
Blitz
- turnLimit
- isDraw
+ changeTurn
+ extendDraw
Standard
- totalTime
- isDraw
+ extendDraw

Design Decisions

A class Session has been created in between player and chess game, so that the functionalities of accepting a player, rejecting a player, inviting other players can be seperated from chess game class thus implementing single responsibility principle.

Different functionalities of a chess game have been abstracted to separate classes like chessboard, time counter, move list etc.

Chess Game is an interface, so different types of chess games like blitz, standard etc. can be implented from it so it is open for extension.

SECTION B 10 min

Q.1

Requirements

- A user should be able to file a case on the portal, and receive the response as the date of hearing
- A user should be able to appeal a judgement by a lower court.
- The court should be able to conduct online trial sessions

Use Case ID	EU-001
Description	File a Case.
Actors	User, Pending Case Manager, Schedule Manager
Pre-condition	User must be logged in.
Main-flow	1. User inputs the description of the case, type of case etc. 2. The user submits the form. 3. The user will be given an option to take a suitable schedule. 4. The systems shows the date of hearing. 5. The user can print the details.
Alternate-flow	3. If it is a spam, the form will be rejected. 4. The given schedule may not be available, so system asks to take a different date.
Post condition	1. The case has been filed and hearing date has been received.