73041.

Assignment #1:

Functional Requirements:

The requirements that specifies what the system should do are known as functional requirements It should include functions performed by specific screens, outlines of work-flows performed by the system, and other business or compliance requirements the system should provide.

It should include:

- Description of data to be entered into the system.
- Description of operations performed by each screen.
- Description of work-flows performed by the system.
- Description of system reports or other output.
- Who can enter the data into the system

Non-functional requirements:

It is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. It defines the constraints on how the system will do so. They are often called "quality attributes" of a system. Non-functional requirements are applied to the whole system rather than each function. Examples of non-functional requirements are usability, reliability, performance, supportability, security, etc.

In a real-estate system for a realtor, the functional and non-functional requirements are:

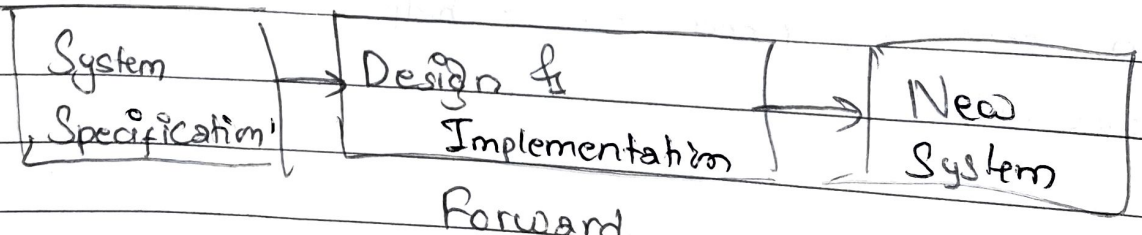
- Functional requirements:
  - Login and registration facility
  - Addition of new brokers.
  - Addition of new property by admin
  - Property search.
  - Enquiry form.

- Non-functional requirements:
  - Reliability
  - Security of the system
  - Service availability.
  - User supportability.

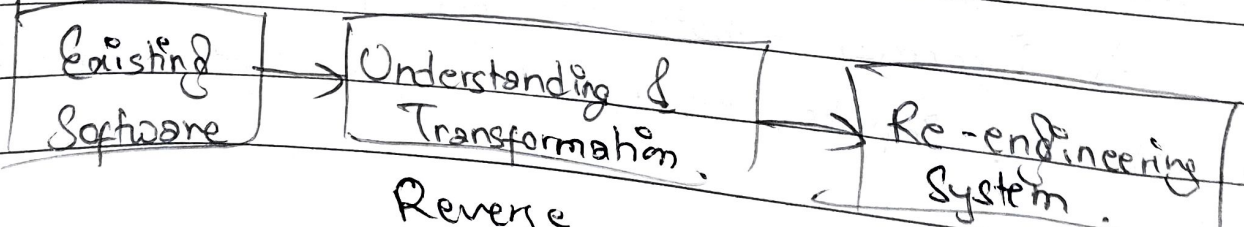
Q. Difference between Forward & Reverse Eng.

Forward Engineering	Reverse Engineering
→ In forward engineering, the application are developed with given requirements	In reverse engineering, the information are collected from given application
→ It is high proficiency skill	It is low proficiency skill.
→ It takes more time to develop an application	It takes less time to develop an application.
→ Nature of F.E. is prescriptive.	Nature of R.E is adaptive.
→ In F.E., the production is started with given requirements	In R.E., the production is started by taking existing product

Blocks:



Forward.



Reverse

Q. Agile methodology in OOAD.

Agile development methods usually apply timeboxed iterative and evolutionary development, employ adaptive planning, promote incremental delivery, and include other values and practices that encourage agility rapid and flexible response to change.

It is not possible to exactly define agile methods, as specific practices vary widely. However, short timeboxed iterations with evolutionary refinements of plans, requirements, and design is a basic practice the method are. In addition, they promote practices and principles that reflect an agile sensibility of simplicity, lightness, communication, self organizing teams & more.

Existing Agile methodologies:

- Rational Unified Process
- Extreme Programming (XP)
- Scrum
- Crystal Family of Methodologies
- Feature Driven Development

Unified Process in OOAD:

A software development process describes an approach to building, deploying and possibly maintaining software. The Unified Process has emerged as a popular iterative, incremental, architecture-centric, use case driven software development process for

building object-oriented system.
It is very flexible & open, and encourages including skillful practices from other iterative methods, such as XP, Scrum, etc.
The UP combines commonly accepted best practices, such as iterative lifecycle & risk driven development, into a cohesive and well-documented process description.

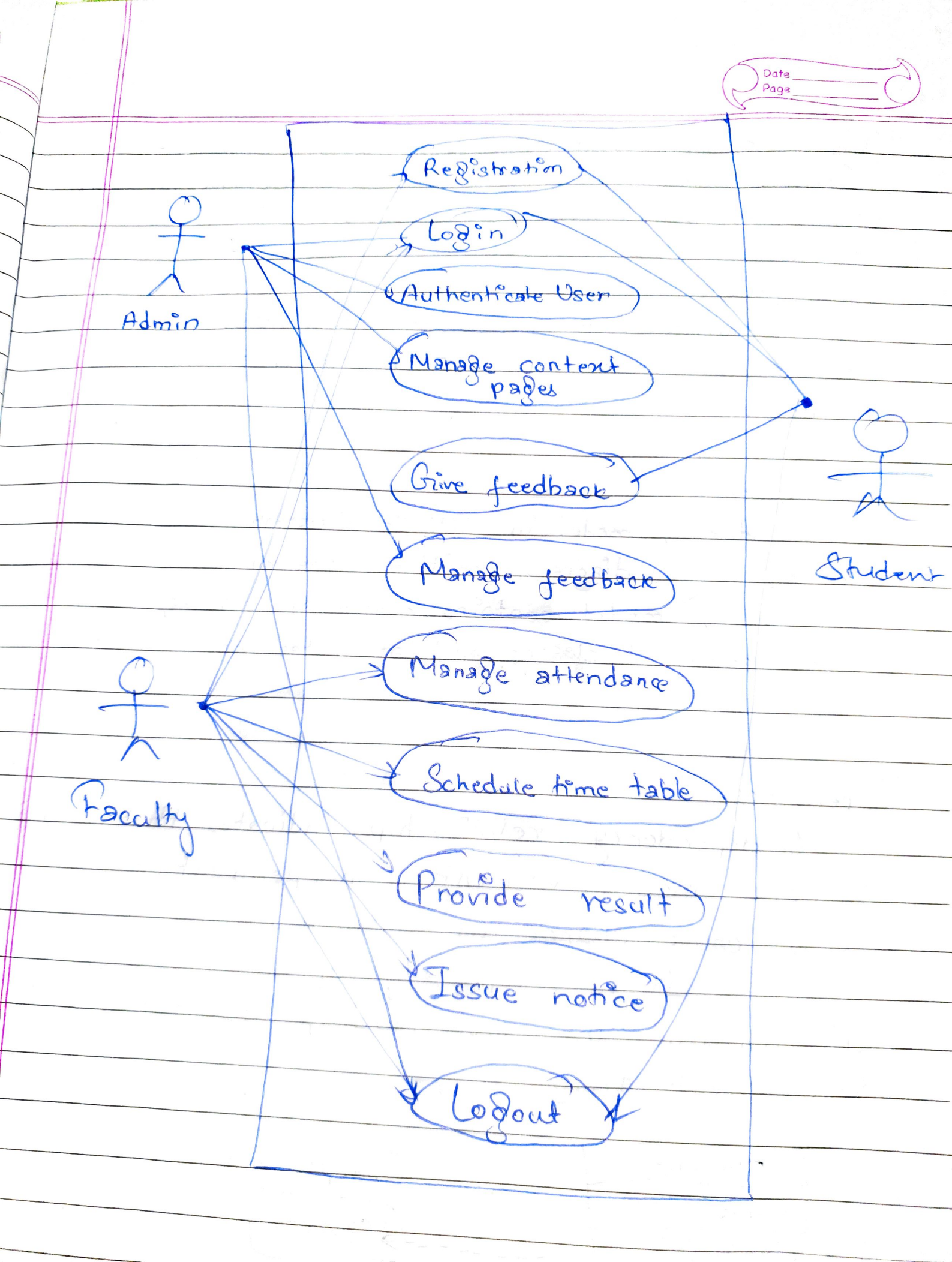
Reasons behind using UP are:

- It is an iterative process
- UP practices provide an example structure for how to do and thus how to explain OOAD.
- The UP is flexible, and can be applied in a lightweight and agile approach that includes practices from other agile methods (such as XP or Scrum).

Use case for University management System:

Actors:
- Student
- Admin
- Faculty.

Use case:
Registration
Login
Authenticate User
Manage context pages
Give feedback
Manage feedback
Manage attendance
- - - - -
- - - - -
Logout.


Registration
Login
Authenticate User
Manage content pages
Give feedback
Manage feedback
Manage attendance
Schedule time table
Provide result
Issue notice
Logout
Admin
Faculty
Student

Define conceptual classes and domain model. Explain primary relationship betn classes: dependency, association, aggregation & composition with their corresponding notation.

Ans: Conceptual class

Conceptual class is defined as a real world concept or thing. It is a conceptual or essential perspective.

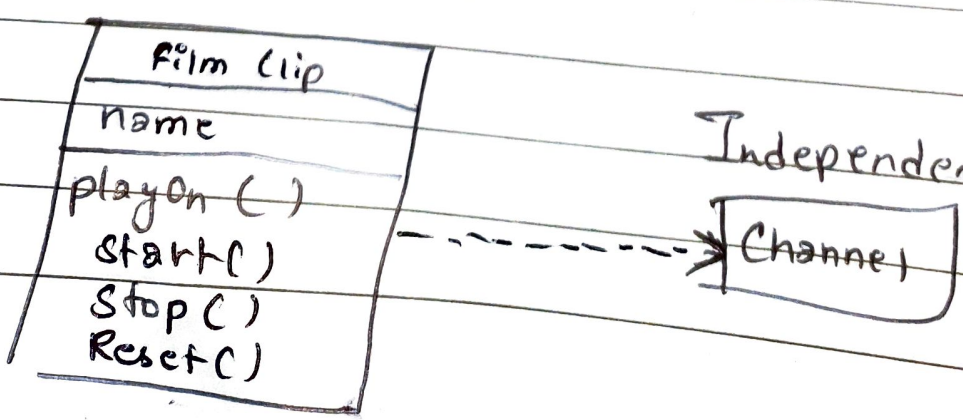
Domain model

A domain model in UML is illustrated with a set of class diagrams omitting the operations. A domain model contains conceptual classes, associations betn conceptual classes, and attributes of a conceptual class.

Dependency

A dependency relationship indicates that change to one model element (independent model) can cause change in another model.

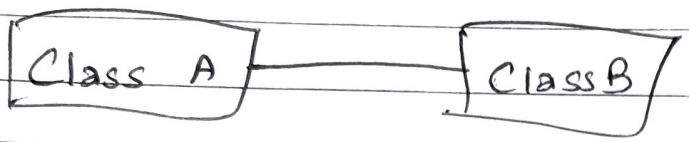
It is graphically represented as a dashed directed line. The arrow head points towards the independent thing.



Association:

It describes links between objects.
It may be of various degrees such as unary, binary, ternary.

It is presented as solid line connecting the participating class, labeled and with adornments (multiplicity & role name)

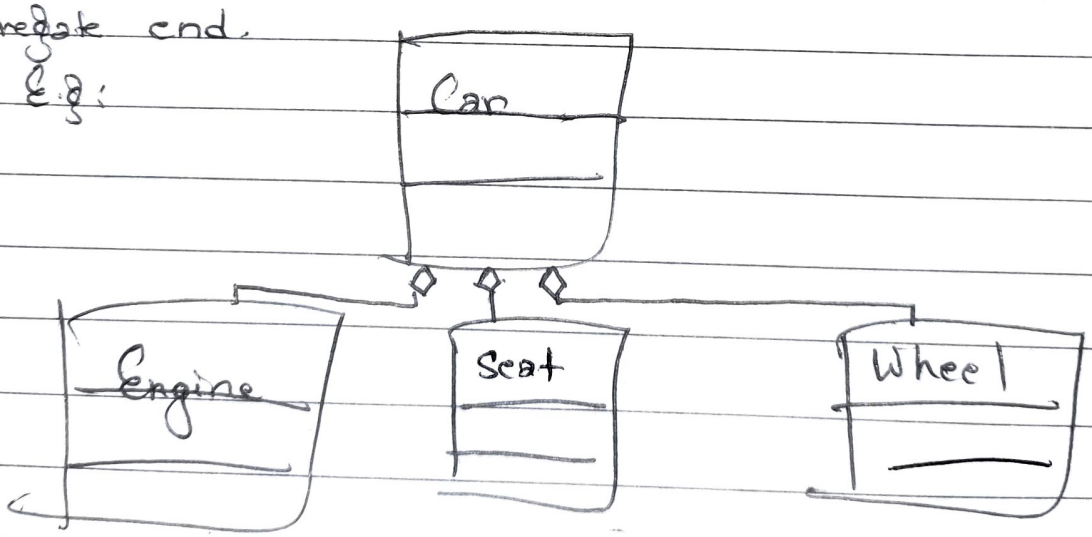


Aggregation:

It represents part of relationship between a component object and an aggregate objects.

It is represented with a hollow diamond at the aggregate end.

E.g:



Composition:

Composition implies a relationship where the child cannot exist independent of the parent.

It is represented with a solid diamond at the end.

E.g :