Bashundhara
Exercise Book
Write Your Future

ANIKA ISLAM (211O1298)
CSE471

Quiz & Assignment — 4

Short Q → Scenario based (can be tricky)

Attendance — 5%

Quiz (n-1) — 10%

Assignment (All) — 10%

Mid — 20%. [Lecture 1 to 5]

Lab — 25%

Final — 30%

Mid → Math + Short Q + Diagram → 2 hrs
(3) (3)

Consultation → Take appointment before

Grade for only → 89 [Not 84, 79]

LECTURE 1

SUNDAY DATE: 26/09/23

LECTURE 1 : INTRODUCTION & SDLC

SEGMENT 1 :

* Information System :- software systems which manage large amount of data, share info.

* Components of information system are :-
  Hardware, software, databases, networks, procedures

Systems analyst is a person – analyzing the business
– identifying opportunities for improvements
– designing information systems to implement these ideas.

System analysis is needed because :-
- Many systems were abandoned because analyst tried to build wonderful systems without understanding the organization.
- The primary goal is to create value for the organization.

System Analysis – collection of notations, methodologies and tools used to gather details and analyze a problem situation prior to information design and implementation.
– ensure :- proposed is meets user needs, can be delivered on time, can be updated inexpensively.

SEGMENT 2:

The project
- moves systemically through phases where each phases has a standard set of output.
- Produces project deliverables.
- Uses deliverables in implementation.
- Results in actual information system.
- Uses gradual refinement.

Project Phases
① Planning [Why build the system?]
- identifying business value
- Analyze feasibility
- Develop work plan
- staff the project
- control & direct project

Product = Project plan

② Analysis [who, what, when, where will the system be?]
- Analysis strategy
  - Analysis of current system
  - ways to design new system
- Requirements gathering
  - Interviews, questionaries, etc
- Process modeling
  - Data modeling

Product = System Proposal

⑤ Design

- Architectural Design
  - Hardware
  - Software
  - Network infrastructure
- Interface design
- Database and file design
- Program design

Product = System specification

④ Implementation

- Construction
  - Writing programs
  - Testing
- Installation
  - Replace old with new system
  - Training users
- Support plan

Product – New system and maintance plan

Identify the project phase from the given scenario

LECTURE 2

SUNDAY DATE: 01/10/2023

* SDLC — Software Development Life Cycle
 — Plan, Propose, Design & Implementation are together known as SDLC.

* Implementation part এর concern বেশি থাকলে — process centered

* Data এর concern বেশি থাকলে — data processed

* Parent - child এর concern বেশি থাকলে — Object processed

* Waterfall — structure development এর উপর depend করে
 — একবার output আসলে, তাকে maintain করবে → basic update করে
 — Requirements from clients are fixed.
 — Since requirements are fixed, design is also fixed.
 — Used for short budget project.

* Prototype — complete project create করে না. It's a dummy project. This is called prototype.
 — পুরো project run না করে, কিন্তু key part run করে।
 — Prototype test করে, কোনো error দেখলে আবার requirement update, design + implementation change করে আবার prototype update করে।
 — Cycle continues upto the client is fully satisfied.
 — After all the tests are done, the final project is sent for production.
 — Production এ দিয়ে দিলে there's no chance to change the requirements.
 — Cycle চলার জন্য costing increase করে।

* Agile – Requirements are being changed continuously after production, then we use agile.
  - Project টা release করে, monitor করে আবার plan করে আবার project টা design & build করি।
  - 1st project না টা exist করবে না আর, only updated project টা exist করবে।
  - Every part of the project requires its own team so costing increases.

DevOps

* DevOps – Similar to agile development
  - Operation & development team work togethers in devop which is absent in agile.
  - Work of operation team : Eg. register এর সময় phone no., digit এ input নেয় কি না, কোন string দিলে কোন warning pop out হয় কি না – এই সব operation team দেখে।

☐ Difference between DevOp & Agile.

☐ Identify the methodology from given scenario.

LECTURE 3

DATE: 03/10/23

TUESDAY

SEGMENT 5

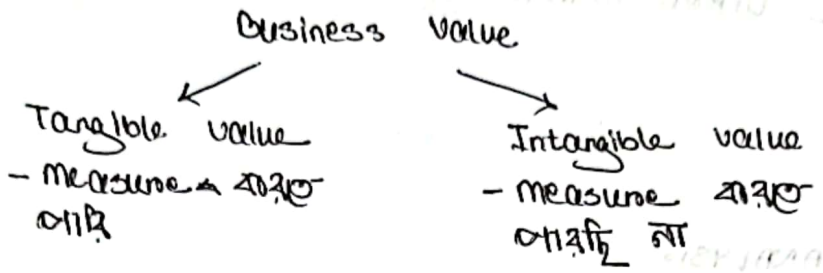
- Scenario based Question on Project Team Roles.
  [Identify the role of the analyst from the given scenario]

* System analyst — Requirements analyze করে

* Change management analyst — Existing project এ changes আনে

LECTURE 2 : FEASIBILITY ANALYSIS

* Feasibility analysis হচ্ছে under a business value এর



* System request – describes business value
  – project sponsor prepares the document
  – approval commitee reviews and judges the system request

* Business requirements – Characteristics of the project handled by admin and user.

* Business value – Cost benefit নিয়ে discuss করা হয়
  - How the project will benefit the company is analyzed here

* Special issues on constraints – যে market a project build করতে চাচ্ছ, সেখানে existing কোন project আছে কিনা – এই টা একটা constraint হতে পারে (Eg. Pathao & Food Panda)

System request is build করতে হবে

*Elements of a system request:-
- project sponser
- business need
- business requirements
- business value
- special issues on constraints

*Values are taken arbitarily.

FEASIBILITY ANALYSIS

*Whether I need to move forward with the project on not.

*Check whether the project will be implemented correctly on not.

*Check how many resources are needed to train the project and whether the resources are available on not are checked.

*Feasibility analysis is mainly done by business analyst.

PRESENT VALUE CALCULATION

$$PV = \frac{\text{total cash flow amount}}{(1 + \text{interest rate})^n}$$

→ For a specific amount of time, the ambunt of money included on excluded in the project.

→ time of cash flow

*PV is the current value of the cash flow caused by the interest rate.

* PV = 75.31 মানে current value. Interest rate = 10%. এখন receive করলে 100 টাকা 3 years পরে PV = 100 হয়ে যাবে,

LECTURE 4

SUNDAY DATE: 08/10/23

* feasibility analysis is done to check whether the project is beneficial enough or not.

Types of calculation of feasibility analysis :—

① PV (Present value)

If a cashflow amount after n years is x, then the current value is y. The current value of the cash flow is the present value.

$$PV = \frac{\text{Cash flow amount}}{(1 + \text{interest rate})^n}$$

n = time in years

② NPV (Net Present value)

Cashflow → benefit — profit হয়
→ cost — loss হয়

NPV = ∑PV of Total Benefits − ∑PV of Total Costs

* কিছু Benefit and cost ও এমন আছে, PV of Benefits and PV of costs ও বের করতে হবে, finally, total মুনাফা থেকে বাদ দিয়ে NPV এর formula ও বসিয়ে দিতে হবে,

③ Return of Investment

* কোন project এ invest করি, তার থেকে কত return পাই

$$ROI = \frac{\text{Total Benefit} - \text{Total cost}}{\text{Total cost}}$$

, if Total Benefit − Total Cost > 0
→ profit
if Total Benefit − Total cost < 0
→ loss

④ BEP (Break Even Point)

*At a time period এ loss হয় (-ve values), even though profit add হচ্ছে, During that period, at a specific time loss টা overcome হয়ে যায় (+ve values). এ specific time কে break even point বলে

*Commulative cash flow যখন +ve হয়ে যাচ্ছে, ঠিক সেই year টাই BEP হয়

$$BEP = \text{Number of years of -ve cashflow} + \frac{\text{That year's net cashflow} - \text{That year's commulative cash flow}}{\text{That year's net cashflow}}$$

*Year 0 তে কোন benefit না আসে, then সেই year কে exclude করে -ve cashflow check করব।

* ~~cost >~~ benefit > cost → ~~profit~~ & +ve cash flow

* benefit < cost → -ve cash flow

Scenario based question on Stakeholder Analysis.

Organizational Feasibility

Stakeholders – project IT manage 2013
– can affect or will be affected by project

Stakeholders Analysis

Champion
- initiates, promotes project
- allocates his/her time for the project
- provide resources

Organizational management
- knows about the project
- budget enough money for the project
- encourage user to accept & use project

System Users
- make decision that influence the project
- perform hands-on activities on the project
- ultimately decides whether or not the project is successful by using or not using project

[ ] Scenario based question on functional & non-functional :-
functional / non-functional
কোন functional/non-functional

LECTURE 3 : REQUIREMENTS DETERMINATION

Functional requirement : যে process গুলো system এর সাথে directly related.

eg. login করা, storing করা DB তে user sort করতে পারবে admin info delete, add করতে পারবে

system টা process perform করে

Non-functional requirement : system এর behaviour কেমন, system কিসে ধীরু করতে পারছি না, performance check করা হয়;

eg. uber এর performance এর উপর feedback নেয়, security

~~phone~~
একটা app android এ চলে ios এ চলে না — operational
inbox নিজের টা নিজে দেখি; অন্য কেউ দেখতে পারে না
— ~~performance~~ security
google user info নেয়, but share করতে পারছে না under laws — cultural

LECTURE 5

TUESDAY

DATE: 10/10/23

* Requirement analysis এর আরেক problem analysis আছে,
* Root analysis problem analysis আছে।
* System and sub-task তৈরি করার জন্য কত time লাগে - duration analysis
* আমরা একটি system বানাই and compare করি existing system এর সাথে — benchmarking

LECTURE 6

SUNDAY   DATE: 15/10/23

UML DIAGRAM

UML — Unified modelling language

- Visual representation of a design

Types of UML Diagram

- Behaviour Diagram
  - → Activity Diagram
  - → Use case Diagram
  - → Sequence Diagram
- Structure Diagram

- Data কিভাবে interact করে

USE CASE DIAGRAM

Actors: students, faculties of usis

Action: login, swap / add / drop courses, showing payslip
swap — faculty করতে পারে

কে কোন action perform করতে পারবে, এইসব যে diagram দিয়ে দেখায় সেটা used case diagram.

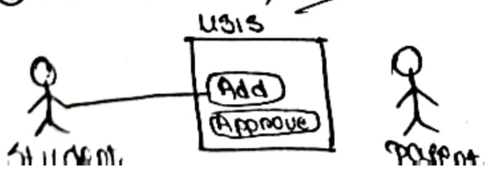
4 Components:-

① Actor → (stick figure) → Actor system boundary এর বাইরে থাকবে
→ Actor system এর সাথে তারা interact করে

② Use case (Action) → (oval) → system এর কাজ হয়ে থাকে

③ System Boundary → (rectangle)

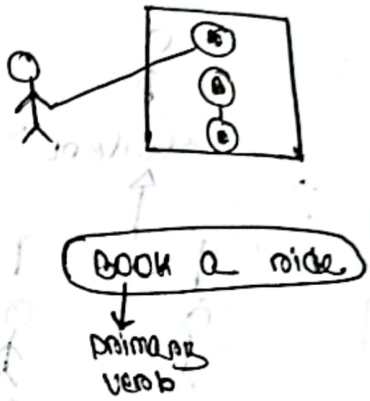
④ Relations → (line) → between actor & action



* Actor must be connected with at least one use case.

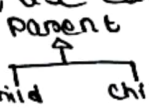
Types of Actor
- Primary actor: main system functions control করে
- Secondary actor: administrative functions control করে
- External hardware: system এর part না, but system এর সাথে interact করার জন্য external device লাগে
- Other: external system

Use case
- Unique name of use case is needed.
- Name of use case need to start with a principal verb and need to be a phrase.
- Use case need to be connected to at least one actor or another use case.



System Boundary
- interacts with the user
- Name of system boundary need to be written on top of it.

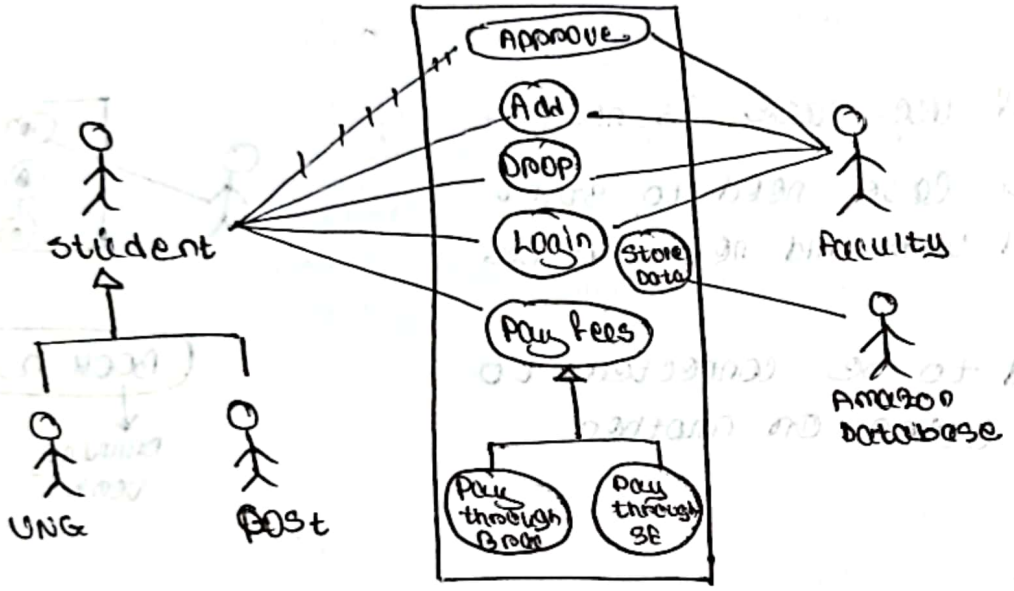
Relation
- communication line
- Actor + use case, use case + use case, actor + actor

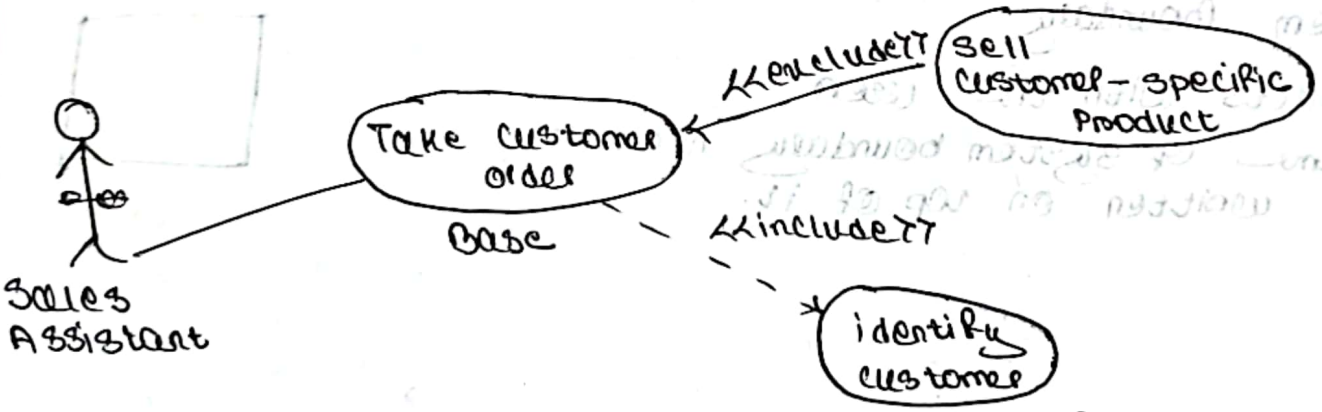
Relation
- Association: between actor & use case, denoted by —
  * Actor must have association relation
  * use case may/may not have
- Generalization: actor + actor, use case + use case (parent child relation) denoted by
  

- Dependency: use case + use case [- - - >]
  - include: এক action use করার জন্য আরেকটা action এর help লাগে
  - extend: এক action use কে extend করে আরেকটা

* include is denoted by যার মধ্যে <<include>> ----> যাকে include করছি (sub)
↳ mandatory
include করছি (main base)

* extend যার থেকে





~~ident~~{customer কে identify করে, then order নিতে পারে [include
যার মধ্যে sell to customer-specific product extend করতে পারি}

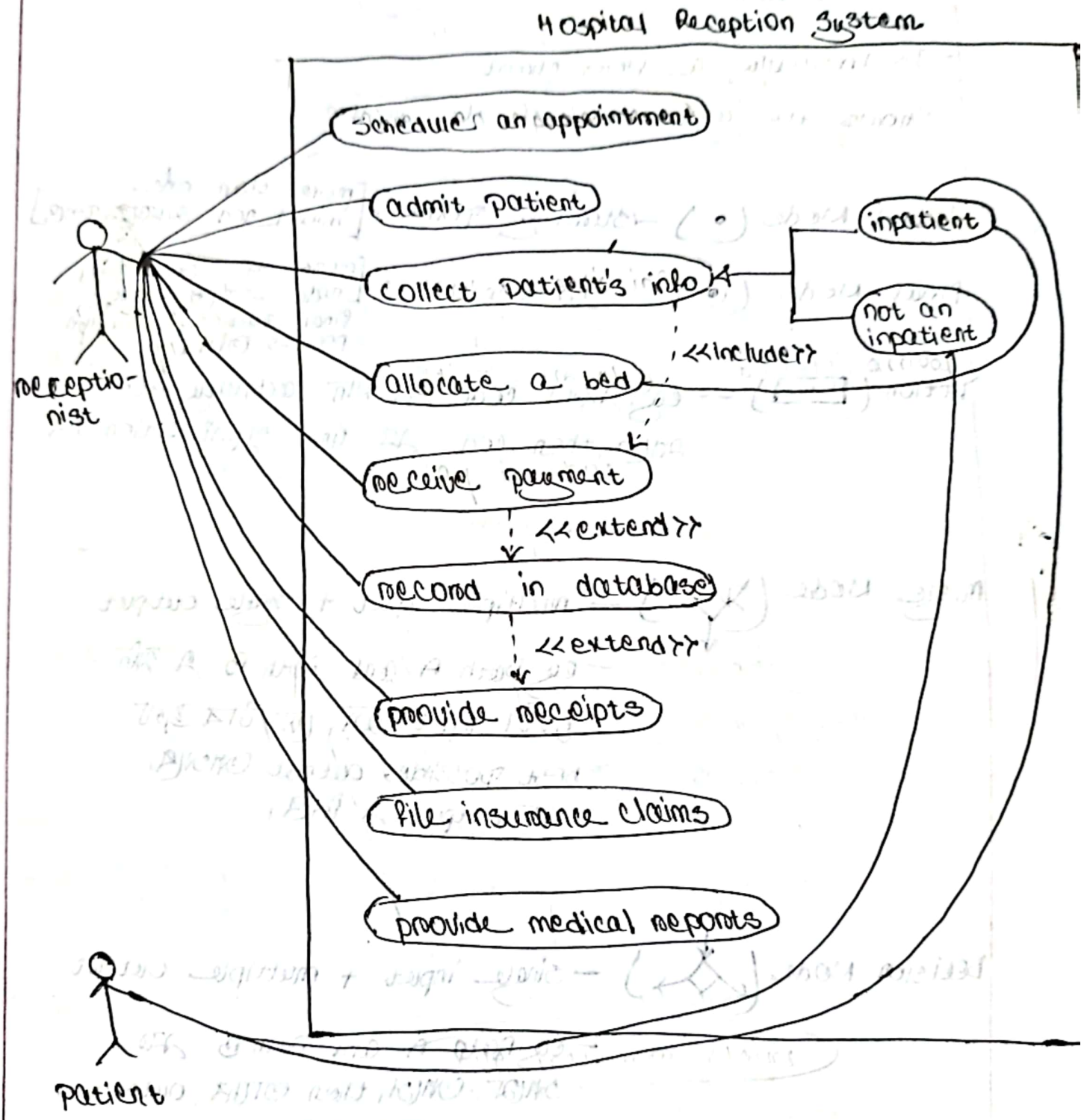
* Base থেকে arrow গেলে → include

* Base থেকে যার দিকে arrow আসলে → exclude

LECTURE 7

TUESDAY

DATE: 17/10/23



ACTIVITY DIAGRAM

- Is basically a flow chart
- Shows the work flow of the system

Initial Node (•) – starting point [more than one initial node থাকতে পারে]

Final Node (◉) — end point [more than one final node থাকতে পারে, final ~~start~~ node এ কোনো process হবে না]

Action ([Action]) — e.g. একটা code এর যদি activity হয় থাকে, then code এর line গুলো action হবে

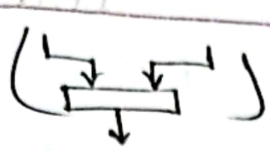
Merge Node () — multiple input + single output

- e.g. Path A and Path B এ যদি দুইটা input হয়, সে টার জন্য best possible output খোঁজে, তার output এ দিবে।

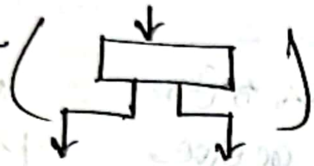
Decision Node () — single input + multiple output

- e.g. Path A and Path B একই সাথে চলে, then তাদের output seperate হয়ে যায়
- condition থাকলে decision tree use করব
- 2 টা task parallely চলতে পারে

* Decision tree এর পরে merge tree থাকতে পারে

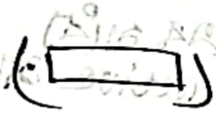
Join Node () – multiple input + single output

– eg. Path A and Path B আলাদা দুইটা আসছে as input, তারপর এর output এক সাথে combine করে সে show করে।

Fork Node () – single input + multiple output

– eg. Path A and B এক সাথে আসছে and output টা seperate হয়।

– condition থাকে না

Object Node () – eg. Patient, receptionist for a hospital management system

– থাকতেও পারে না ও পারে

~~No~~

Note () – Comment ~~box~~ or notification add করে

Swimlanes – Activity divide করে under certain ~~cat~~ department.

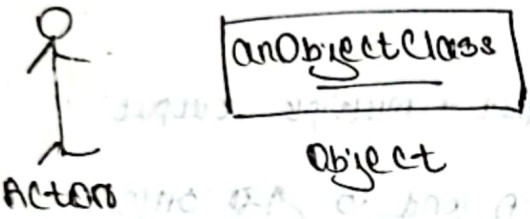
*Draw a diagram using swimlanes বললে department এ ভাগ করে, or else না করলেও হবে।

LECTURE 8

SUNDAY DATE:

SEQUENCE DIAGRAM

* sequencially each step কোন সময় এ হচ্ছে



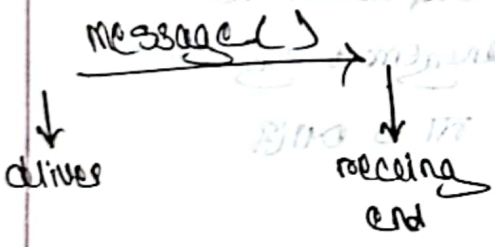
Lifeline

Object এর যতক্ষণ active থাকবে

message deliver and receive করার জন্য eligible

focus

visually object কখন কতক্ষণ active ছিল point.



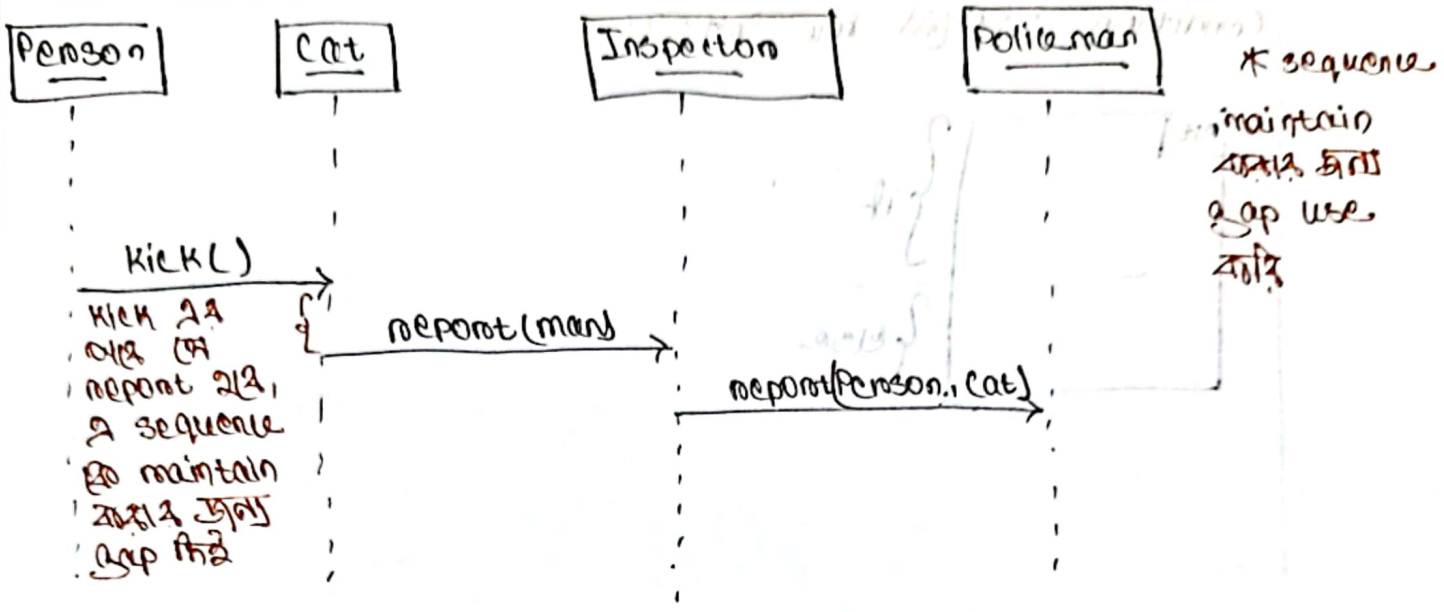
× Object destruction end of an object's lifeline

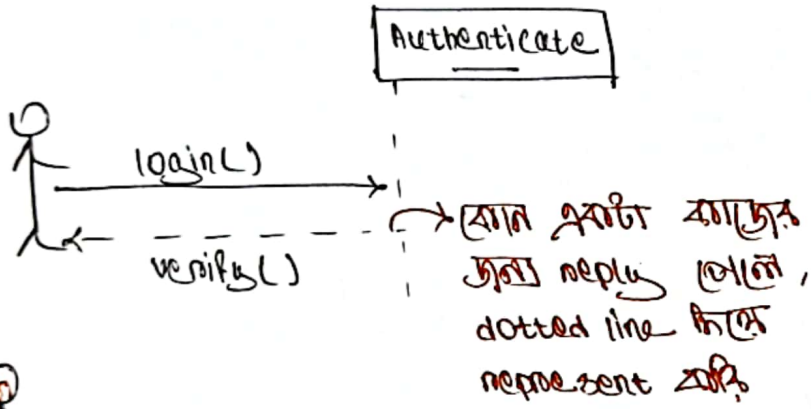
*Reply/response হলে dotted line দিয়ে represent করি

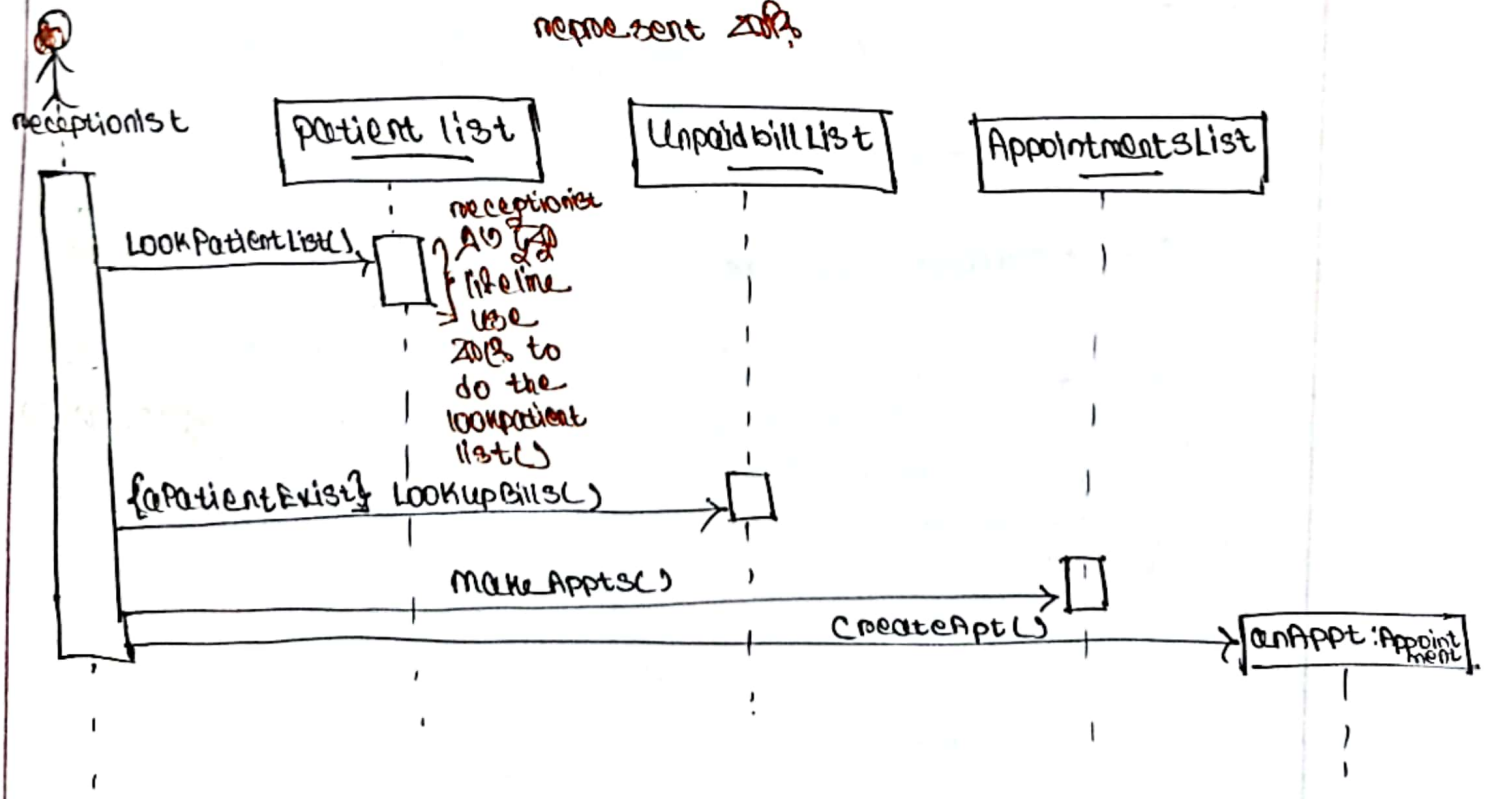
LT box → if and else

function হলে ~~to~~ hobe function এ দিয়ে with line.

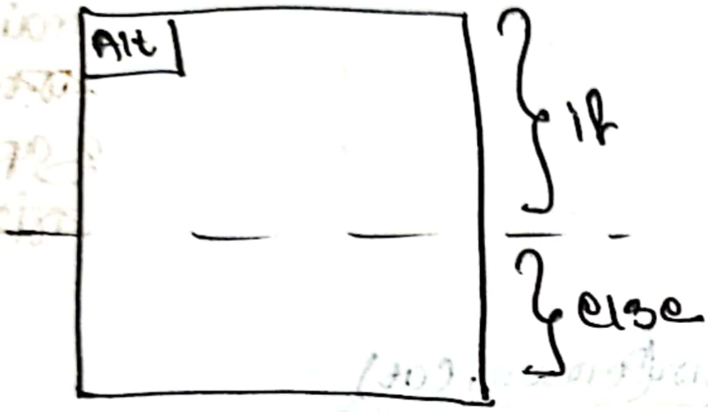
normal value ~~to~~ হলে function এ দিয়ে না and dotted line দিব


Person
Cat
Inspector
Policeman
* sequence maintain করার জন্য 2 ta use করি
kick()
kick এর পর cat report করে, sequence maintain করার জন্য 2 টি
report(man)
report(person, cat)


Authenticate
login()
verify()
যেকোন একটা কাজের জন্য reply দেখালে, dotted line দিয়ে represent করি


receptionist
Patient list
Unpaid bill List
Appointments List
LookPatientList()
receptionist এর 2টি lifeline use করে to do the lookpatient list()
[PatientExist] LookUpBills()
MakeAppts()
CreateApt()
anAppt: Appointment

Condition আসলে box আঁকবো:-



LAB 1

SATURDAY DATE: 30/09/23

frontend → Angular/React

Backend → Django (Python)
Spring boot (Java)
.Net core

Database → MySql

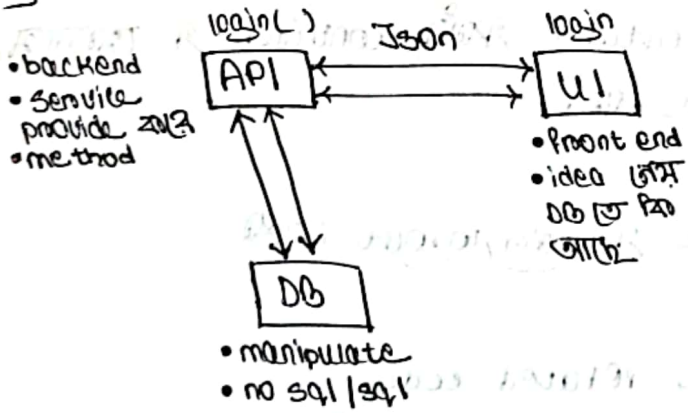
Relational → Tables এর মধ্যে relation থাকে

Non-relational → Table use করি না
→ Document use করি

Json { "username" : "ABC", "password" : "123" } (Java script Object Notation) → • Json object • API, UI এর মধ্যে মুভ করে

Front end (client side)

Back end (server side)



* API & UI তে Json response যায়,
* API info DB থেকে আনে
* UI info API থেকে নেয়ে user কে info show করে

purpose of API :- (i) Data access
(ii) Feature + functional add করা যায়
(iii) Security
(iv) Business logic

* If there's a login button in UI (front end) and it's pressed, a Json object with the login credentials is created. There's a login function in API (backend) which can hold the login credentials from Json object and then check it with DB. If the info matches with that of DB then a return value is send to API and then to UI.

LAB 2

SATURDAY DATE: 07/10/23

Report :-

(i) Group info

(ii) Table of content

(iii) Introduction

(iv) User manual

(vi) Frontend (ss)

(vii) Backend (ss)

(viii) Frame-work + Contribution + Git link

ANGULAR
[Typescript]

* module - group of feature একটা container এ রেখেছি, তাকে module বলে।
  - Facebook :-
  User module এ login, logout থাকে

* Component — feature related code
  - application এর সবচেয়ে ছোট block
  - Component এ ৪ টা file থাকে
    • html file - view related জিনিস থাকে
    • typescript file — html file এর জন্য code থাকে
    • css — html এ একটা button add করলে, তার design টা inline এ থাকে. in html.
    • scss — class create করে for a design in css.

- Component এর life cycle hook আছে।
Data change হয় at times, এর জন্য view টাও change হয়।

- ngOnInit() — view load হওয়ার আগে load হয়
  — একবার আগে load হয়
  — component load, হওয়ার আগে load হয়
- ngAfterViewInit() — view load হওয়ার পর কিছু load করতে চাইলে, afterviewInit() use করা যায়।
- ngOnChanges() — কোন data change হলে, onchanges() টা তে দিলে view টাও change করে দেবে।
- ngOnDestroy() — টা load হওয়ার পর, Angular component টা destroy করে দিয়ে
  — subscription কোন component hold করে রাখলে, subscription গুলো না নাশালে, destroy করে দেলে আরো subscription to release more space.
  ~~— কোন compone~~

* Typescript datatype ~~কে~~ কে control করতে পারে।
* Purpose of component is to ~~just view to show the data~~. take the data and make the data visible.

Routes

* URL change হয় - তোমার route

Services → Typescript class

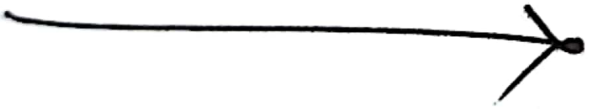
- Logic handle করে
- ~~Html~~ service এ code লেখালে, সেই code multiple purpose use ~~করে~~ করা যায়
- backend এর সাথে communicate করে component কে ডাটা দেয়, ~~to vie~~ Then component view করায়।

* Logic handling এর code component এ লেখা যায়। But, in this case, component light weight হয় না

module — কি কি component আছে module এ তাকে declare করব,

- কোন module তার component ~~অন্য~~ অন্য module কে দিতে চায়, তাকে export করি,
- অন্য কোন module তার বাইরের module এর component use করতে চায়, then import করি,
- provider এর মধ্যে services দিবো,
- bootstrap only one module এ থাকে, ~~starting~~ App component load করে,

FINAL



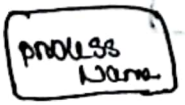
LECTURE 9

SUNDAY

DATE: 19/11/23

DATA FLOW DIAGRAM (DFD)

Process – External agent process এর মাধ্যমে data pass করে থাকে
– System এর মধ্যে যা যা task perform করা যায়, তাকে process বলে
– Active verb দিয়ে process নাম start করা লাগবে
– process এ data enter and exist করাতে হবে
– Represented by



Dataflow – flow of data from source to

source —dataflow name→ destination

– যে data pass হয় তাকে dataflow বলি

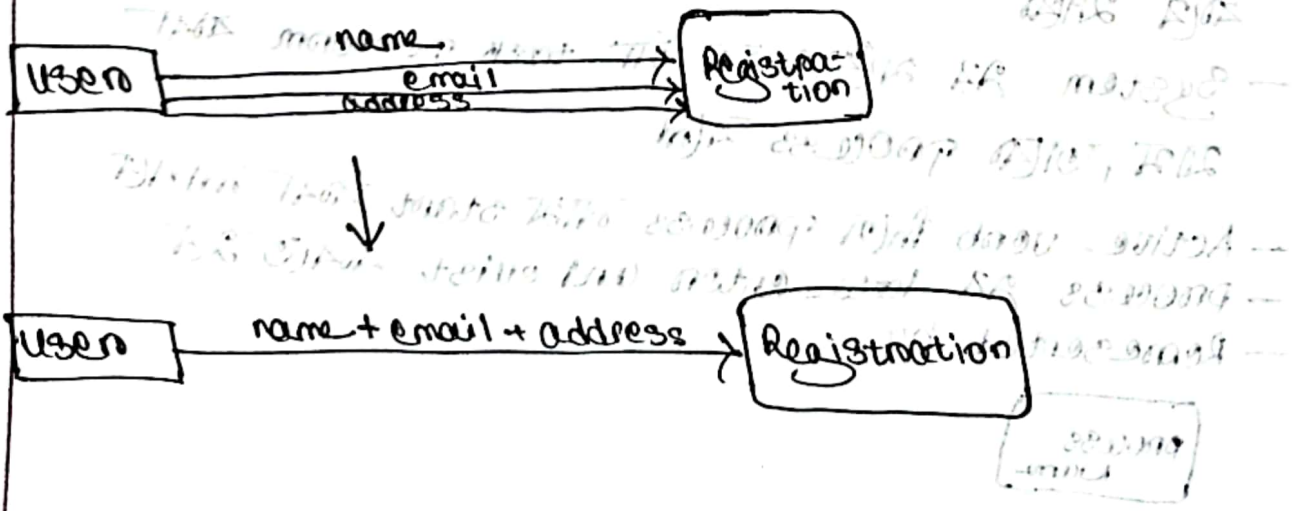
Composite dataflow — একই source & destination এর মধ্যে more than one data flow থাকে

- eg source and destination এর মধ্যে name, address, phone no. different different dataflow দিয়ে pass হচ্ছে
- এই multiple dataflow কে আমরা combine করে দেখাব

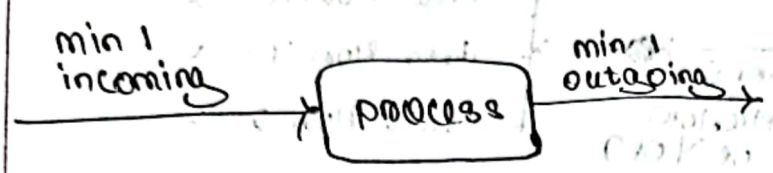
Controlled dataflow — data pass হয় না, signal pass হয়

- eg stop করার জন্য flag pass করছি, flag টা একটা signal.

Normal Dataflow:
Composite Dataflow:-



①

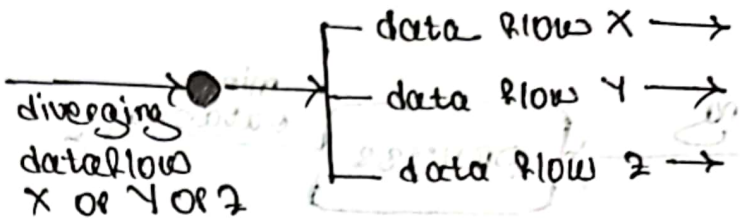


② একটা output এর যে যে inputs দরকার process এর তা input নিতে হবে as dataflows.

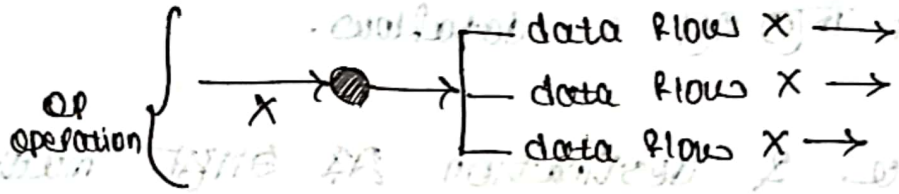
③ Source & destination এর মধ্যে multiple data flows থাকতে পারে, multiple dataflows কে combine করে একটা composite dataflow বানাবে.

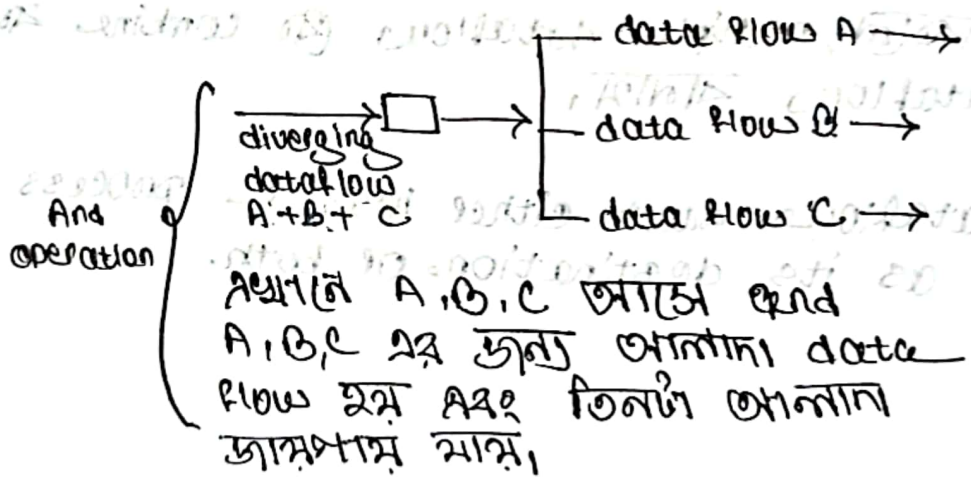
④ Dataflows must either have a process as its source or as its destination or both.

Diverging dataflow — এক single dataflow কে যখন split করে multiple destination এ data যায়।



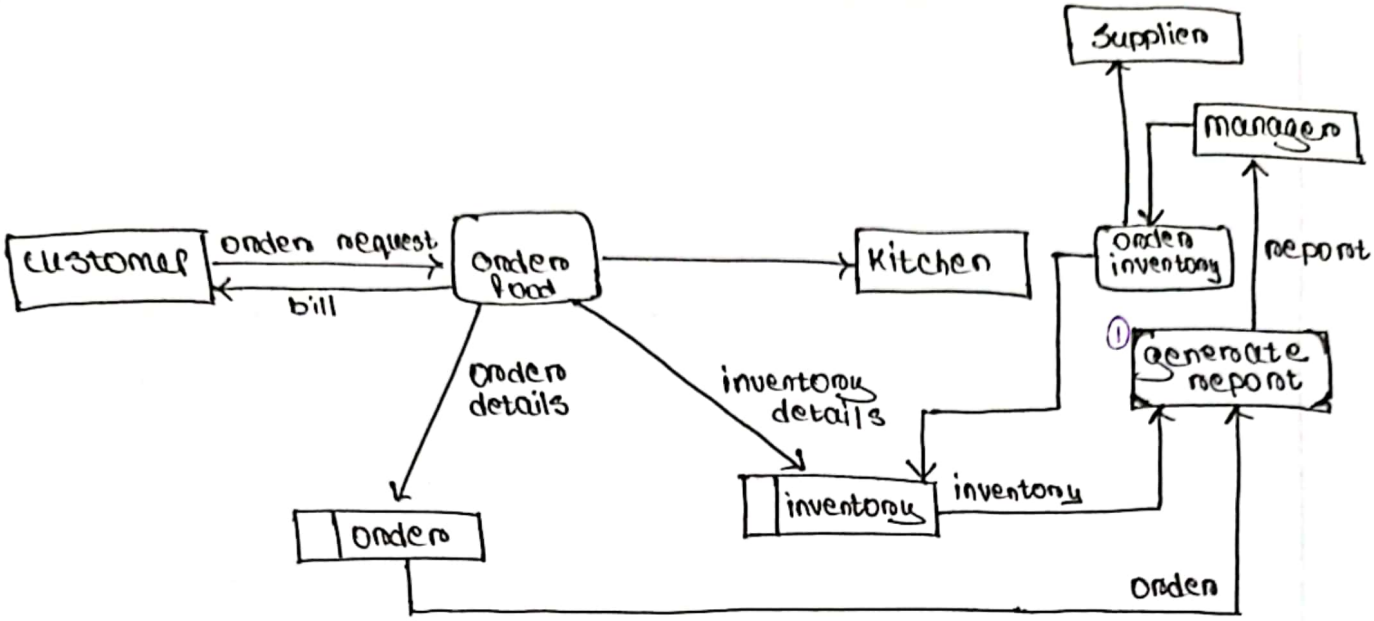
যদি X আসে X কে তিনটা copy তিনটা আলাদা জায়গায় যায়





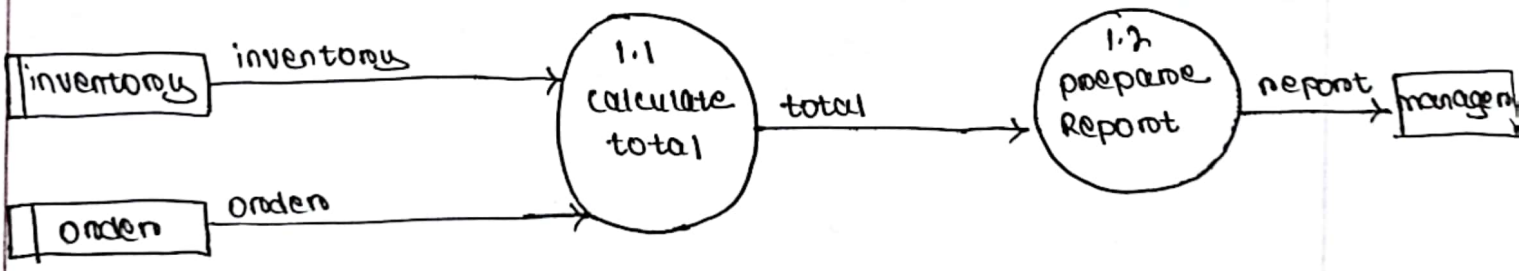
এখানে A, B, C আসে and A, B, C এর জন্য আলাদা data flow হয় এবং তিনটা আলাদা জায়গায় যায়।

Converging dataflow —



context diagram
— level 0 data flow diagram

Process Decomposition — level 0 থেকে level 1 এ যাওয়া
— level কে decompose করে কতো depth এ যাওয়া



"Level 0" to "Level 1"

CTURE
10

TUESDAY

DATE: 21/11/23

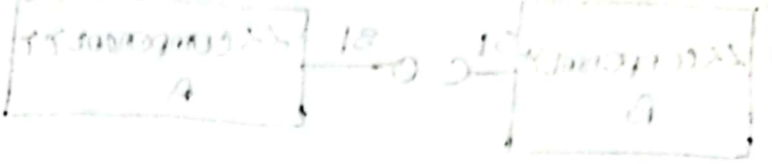
COMPONENT DIAGRAM

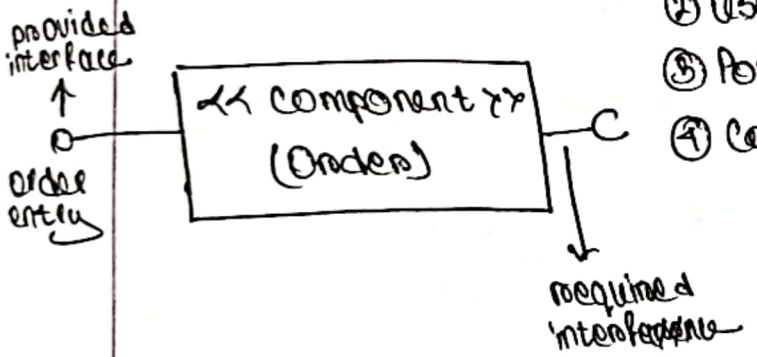
Component Notation

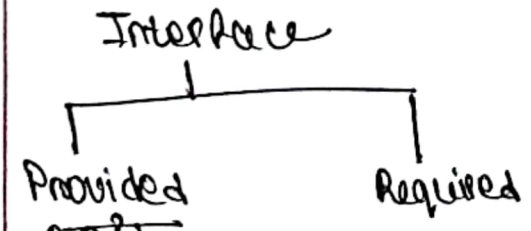
Representation

Interface — Set of operations component perform করে সাহায্য

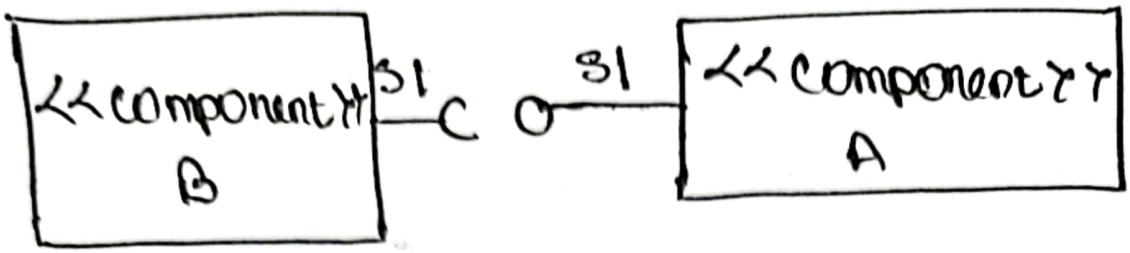
provided interface —





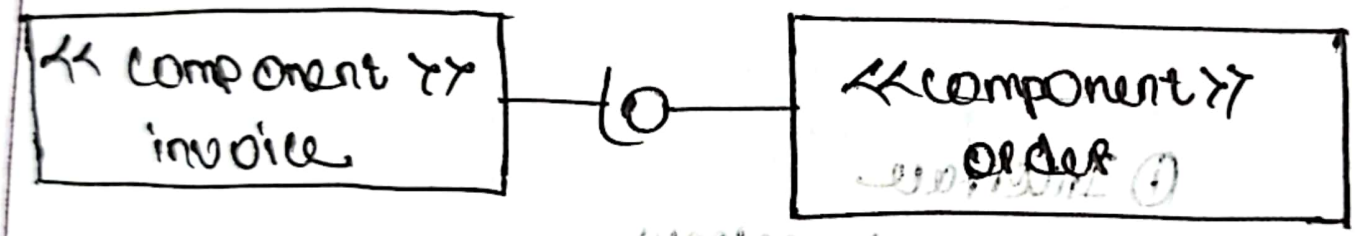


- এবার component যে যে কাজ perform করে সাহায্য



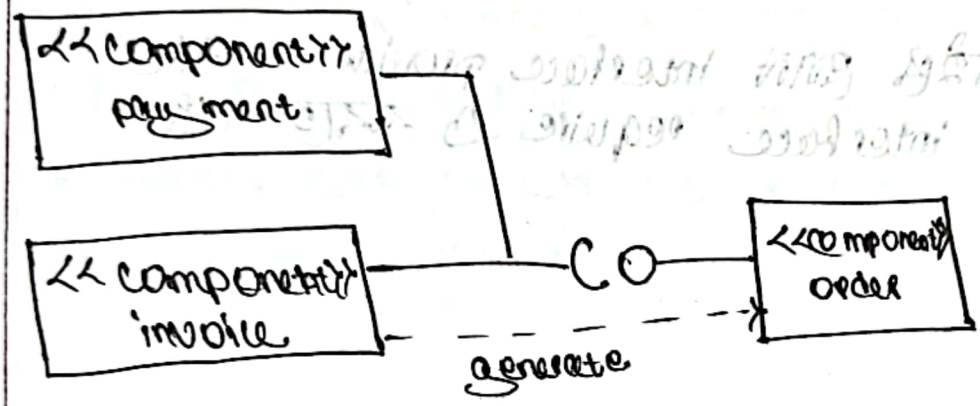
* component A SI service provide करता है
* component B SI service receive करता है from component A.

Eg :-



Usage Dependency

→ relationship which one element requires another element for its full implementations.



*Order depends on invoice

Usage dependency को dotted line र arrow ले निर्देश गरेको dependent.

Port

→ Connection point

→ যেখানে component কে কোন environment এর সাথে connect করছি

→ port কে component এর সাথে বাইরে থেকে connect করানো — port is public

→ component এর মধ্যে port আবদ্ধ — port is private/protected

→ public port দিয়ে বাইরে কোন interface provide করতে পারি and বাইরে থেকে interface require ও করতে পারি

Connector

→ Interface কে connect করে

Assembly connector

→ Ball and socket connection: —co—

→ multiple ক্লাসের থেকে same info পাস করে
এক জায়গা দিয়ে এটা connect করি

Delegation connection:-

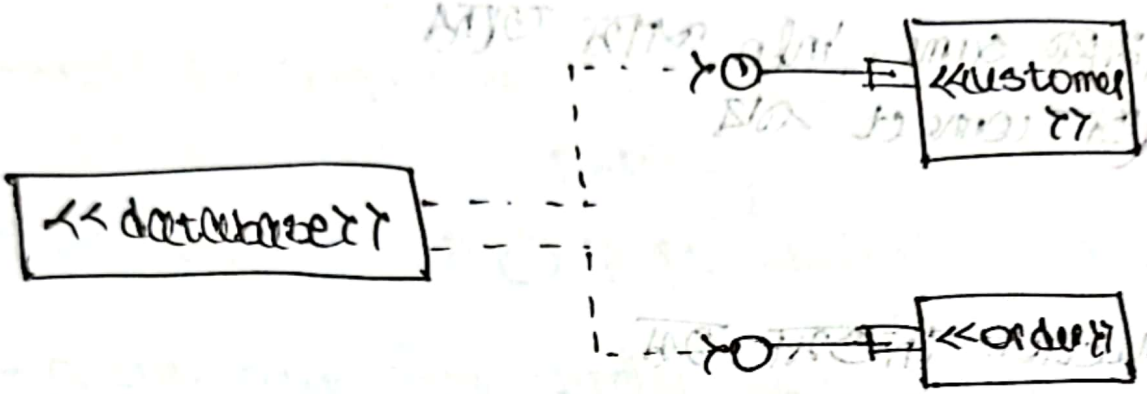
→ Port দিয়ে যে connection দেওয়া হয়

→ Public Port এর সাথে connection

→ Private Port connection is not
delegation connector.

Dependency Relation

→ One component is dependent on data from another component.



LECTURE 11

TUESDAY

DATE: 28/11/23

PROGRAM DESIGN

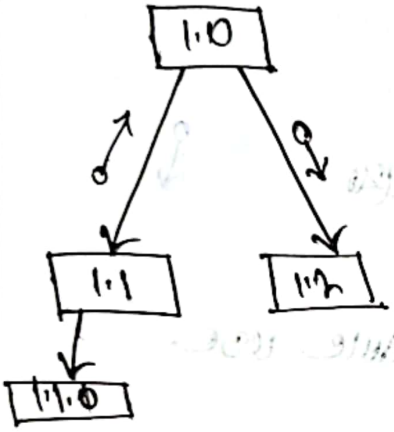
Structure Chart

→ help analyst to design the program properly.

→ কোন condition এর জন্য module টাকে চেক করতে করবে

→ কত বার condition টা আসবে

→ code এ কি কাজ হয় তা দেওয়া থাকবে, সেটা দেখে আমাদের structure chart বানাতে হবে।



→ module
- function কে বলে
e.g. sum, max, fibonacci

control module (parent module)
1.0 is the control module for 1.1 & 1.2

subordinate module (child module)
1.1 and 1.2 are the subordinate module of 1.0

→ Library module — child module থাকবে না

— piece of code which is repetitive.

— কোনো function multiple time call করা লাগে

— multiple time use করা যায় কি না check করি

→ Loop

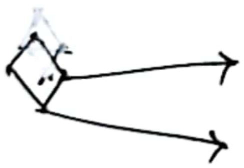
— কোনো module multiple time repeat করে

— for/while loop এর মতো

— loop কয় বার ঘুরবে fixed থাকে

pre-written function গুলো library module use করে।

scratch থেকে code লেখা লাগলে loop use করবো।



→ Condition

→ একটা arrow "if" condition এর জন্য

→ আরেকটা arrow "else" condition এর জন্য

→ Data sample

→ arrow এ data থাকে and circle থেকে data যাবে

→ Control couple

→ end of file [code certain part execute হয়ে stop হয়ে যাবে]

→ always upward pointing হবে

 → Offpage connector

→ structure অনেক জায়গা না থাকলে in a single page, next page এ গেলে এই connector use করি।

 → Onpage connector

→ structure অনেক জায়গা আছে in a single page, but parent module / child module এর সাথে connection করা থাকে, same page এর ভিতর জায়গায় এই connector দিয়ে আঁকি।

* প্রত্যেক line এর সাথে connection থাকে in a code → cohesion

*** functional cohesion — module একটা function এ কাজ করে। (best)

sequential cohesion — module একটা line এর output অন্যের line এর input হিসেবে use হয়

communicational cohesion — same input and output দিয়ে থাকে

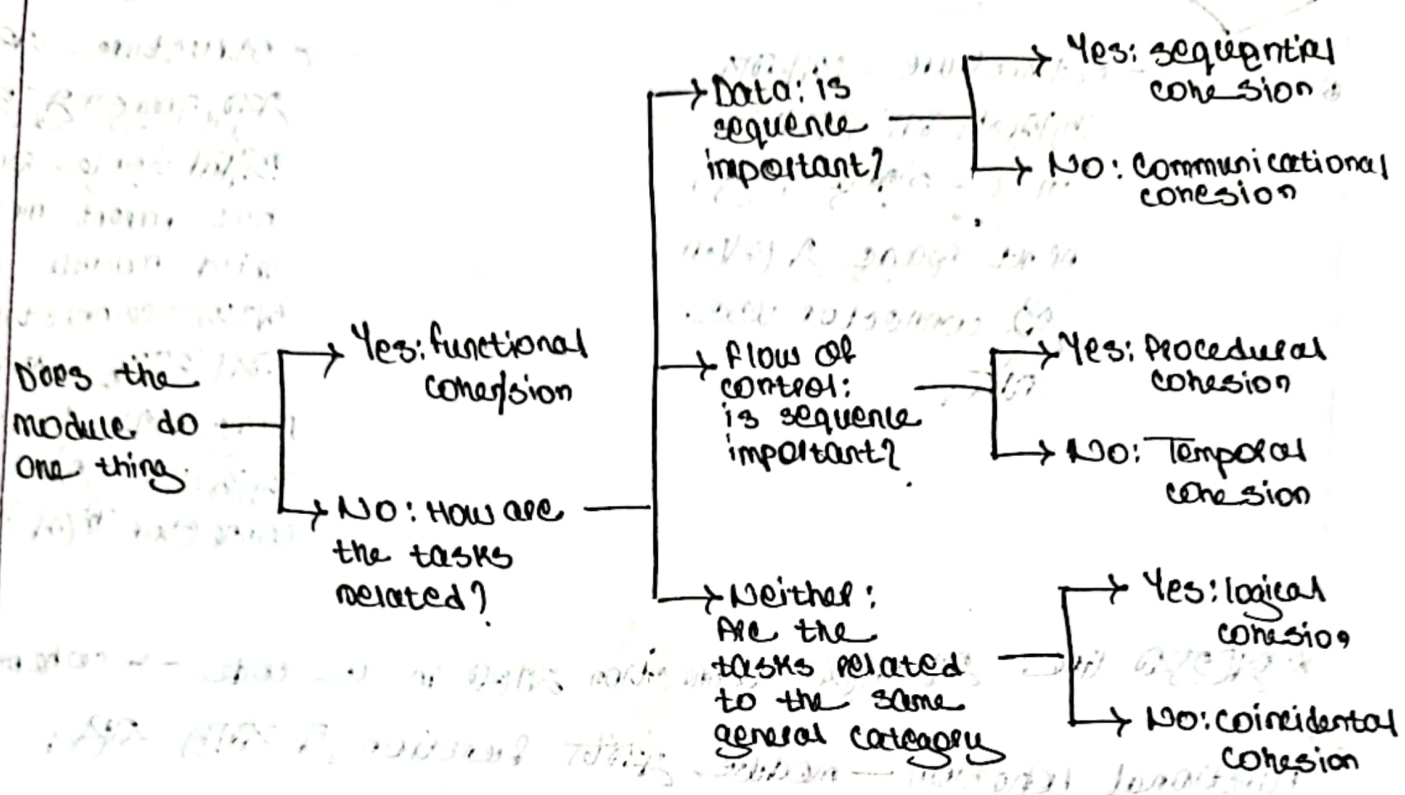
procedural cohesion — task will be performed sequentially but no data is shared.

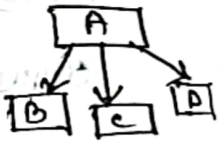
temporal cohesion — প্রত্যেক line of code এর মধ্যে connection থাকে

logical cohesion — 4+ কোনো parameter use করে একই command বারবার থাকে অনেক

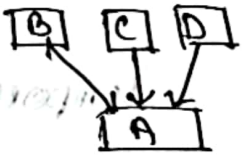
coincidental cohesion — কোনো code এর মধ্যে কোনো connection নাই।

Cohesion Decision Tree



Fan Out — একটা control module এ more than 7 subordinate module এ help নিলে তাকে


— High fan. out is not good as it's not a well-written code.

Fan In — একটা subordinate module এ multiple control module থাকে,
— একটা code কে বার বার use করতে পারি
— High fan in is good.
— একটা module কে একটা module call করে


LECTURE 12

SUNDAY

DATE: 03/12/23

* Data store design – data store 2nd
– files, databases → data store 2nd 2nd

*

ECTURE
13

TUESDAY

DATE: 05/12/23

Final Suggestion

Lecture 6: Component Diagram
→ Diagram rules জানা লাগবে
→ Diagram চিনতে লাগবে

Lecture 7: Data flow Diagram
→ Level-0/process diagram, Level-1 Diagram আকা চিনতে লাগবে

Lecture 8: Data Store
→ File and database এর difference
→ Relational, object and multidimensional database — scenario based
→ Data store optimization
- storage efficiency — normalization
- speed access — Denormalization
  - Clustering
  - Indexing

→ Data store এর volume measure করা লাগবে

Lecture 9 : Program Design

→ Structure chart এঁকে

maintaining the rule :-

→ Cohesion এর ৬ টি types clear ভাবে বলতে পারবে (scenario based)

- chart দেখে বলতে পারবে

→ Fan-in & Fan-out

↓ child
এত গুলো parent এর সাথে connected আছে

↓ parent
এর অধীনে এতগুলো child আছে

- Calculation count করে বলতে পারবে

Lector 10

→ window navigation diagram (not important)

→ Theory (not important)

UI Design

Layout → structure এর মতো
→ কোন জায়গায় কোন functionality
বসাবো।

content awareness →

aesthetic → modern way তে UI design করা

user experience → প্রতিটা page যত user
friendly

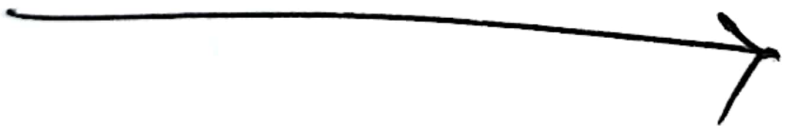
consistency → element গুলো যতটা
consistent

minimize user effort → প্রতিটা function use
করার জন্য user কে
যত effort দিতে হয়

এ scenario দিয়ে দিয়েছে এবং বলবে কোন part
improve করা যাবে।

FINAL

(MIH)



Component Diagram [সংজ্ঞা]

→ একটা software কে ছোট ছোট component এ divide করি।

→ Kind of physical graph. (Structural diagram)

→ Component and 'dependencies' relation থাকে

Component

→ module কে component হিসেবে represent করা হয়।

→ connection build করা হয় 'interface' এর।

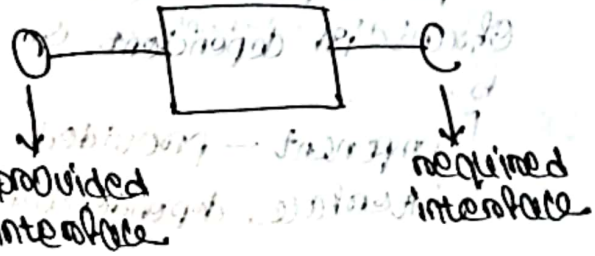
→ যে সব module এর functionality same তাদেরকে একটা component হিসেবে represent করি।

→ Representation:-

Interface

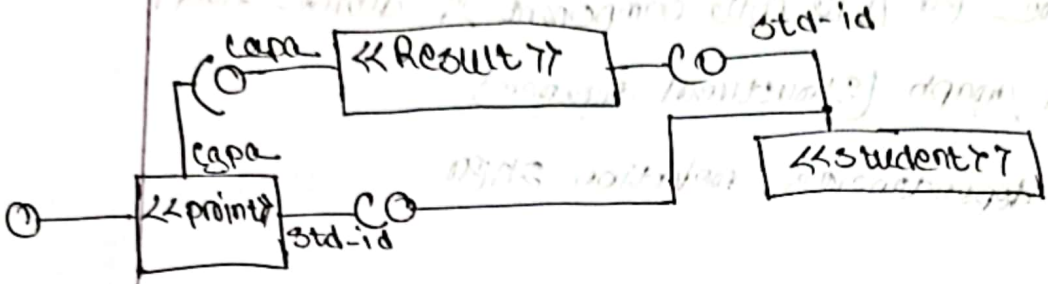
→ Kind of 'input/output'.

→ কোথায় আমরা কি provided/required.

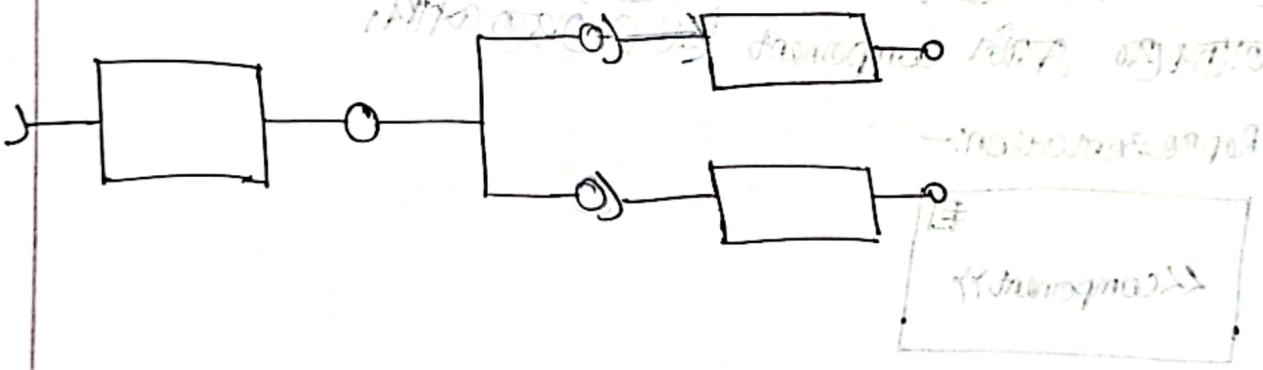


"Ball and socket connection"

* Required interface এ 'input' নিচ্ছে, component এটা এবং provided interface এ output দিচ্ছে।

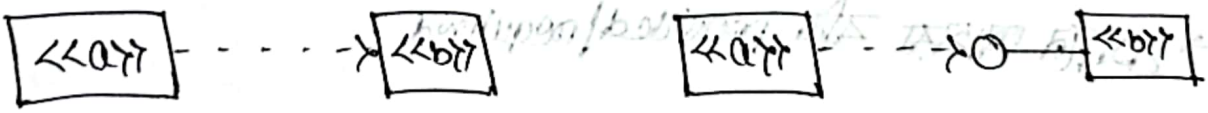


* Here, std-id is provided by student and required by result.
for print, cgpa is required and is provided by result.



Dependencies

→ একটা component এর আরেকটা component এর উপর dependent.



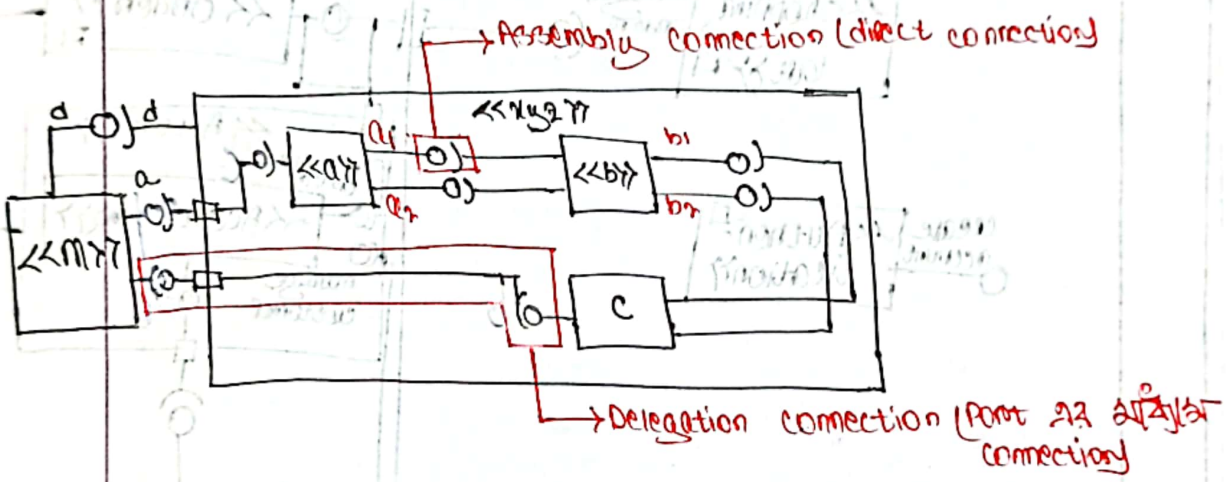
* a is dependent on b
[component – component dependencies]

* a is dependent on the provided interface of b.
[component – provided interface dependencies]

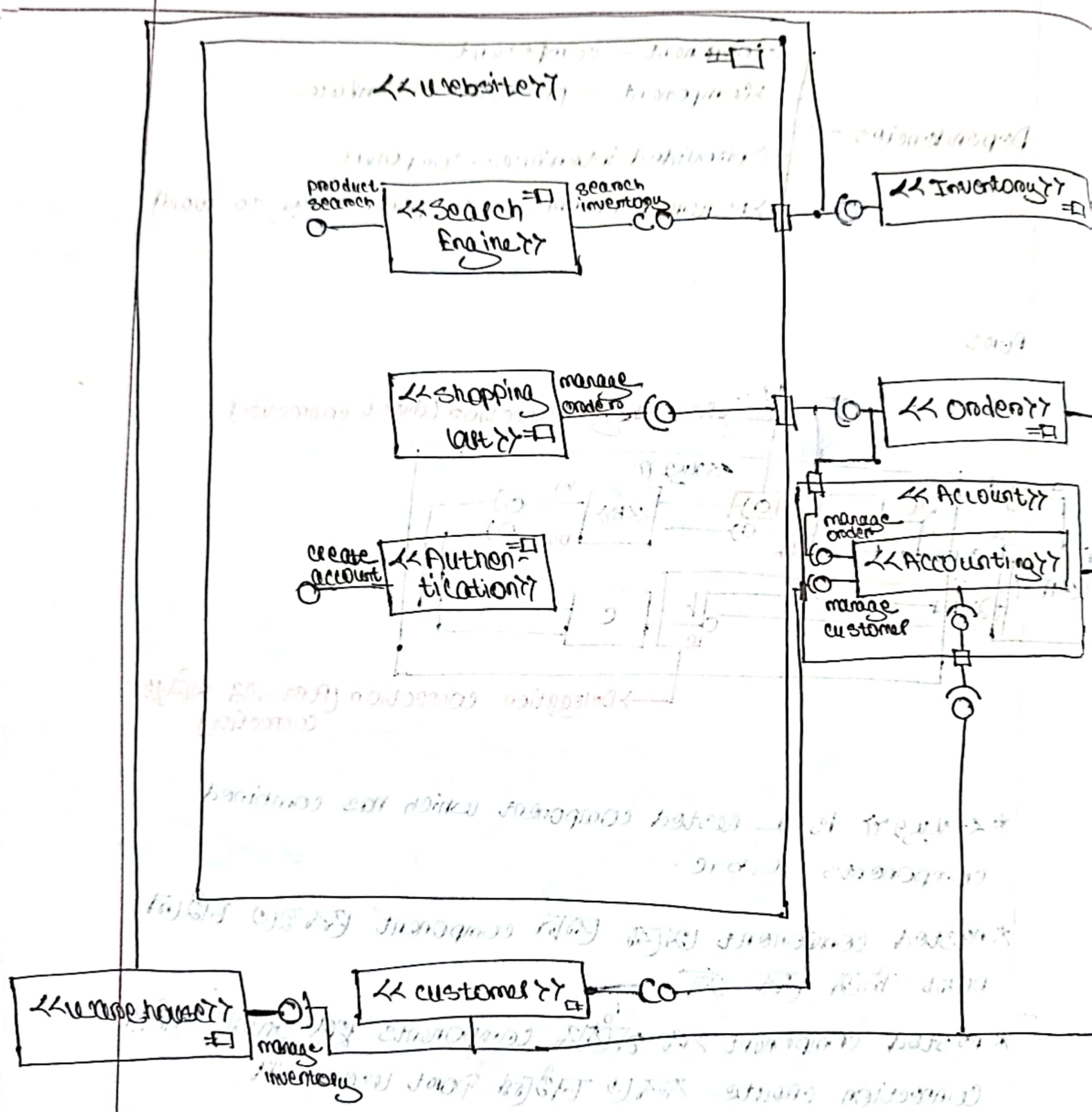
* যার উপর A dependent হয় arrow তার উপরে থাকে

Dependencies
- component – component
- component – provided interface
- provided interface – component
- required interface – component (try to avoid)

Port



* <<xyz>> is a nested component which has combined components a, b, c.
* nested component থেকে কোন component কে যুক্ত করতে চাইলে port দিয়ে করতে হয়।
* nested component এর ভিতরে components দের সাথে বাইরে connection create করতে চাইলে port use করি
* দুটো nested component এর মধ্যে connection create করার জন্য port লাগে না,
* Delegation মাধ্যমেই বুঝা যাবে যে → port আছে
  → nested component আছে


«website»
product search
«search Engine»
search inventory
«Inventory»
«shopping cart»
manage orders
«order»
«Account»
manage orders
«Accounting»
manage customer
create account
«Authentication»
«warehouse»
manage inventory
«customer»

Data flow Diagram [DFD]

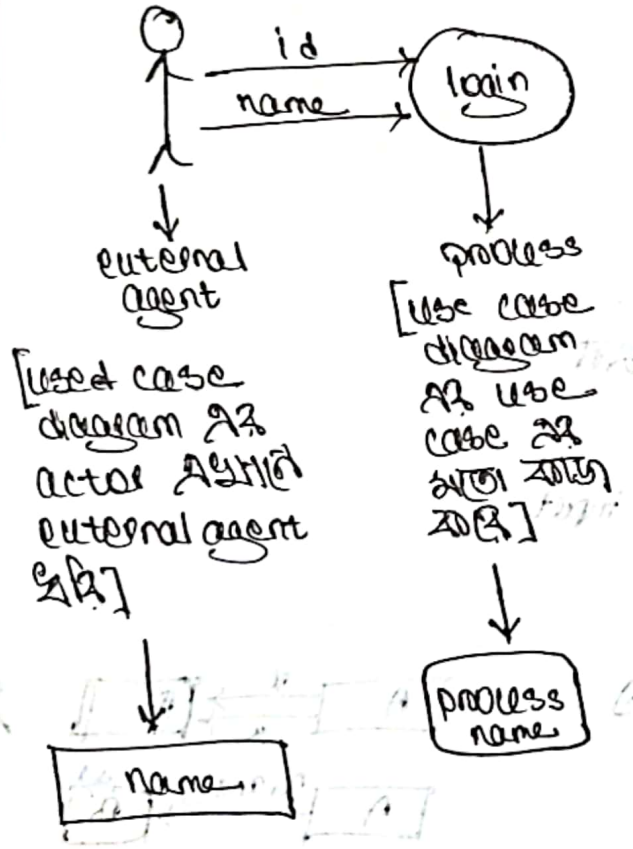
→ Data — Uses a name, id variable হিসেবে data.

→ Database — Data রাখার জন্য।

→ logical model এবং idea নিয়ে physical model build করি

→ Data flow — movement of data between external agent, data store, process

- external agent → outside person unit, system
- data store → database, file
- process → incoming dataflow নিয়ে input নিয়ে, output দেয়



Dataflow

(item.ID, amount) → dataflow
দুইটা একসাথে যেতে পারবে

dataflow name

Composite data flow

dataflow name →

* অনেকগুলো দুইটা dataflow নিয়ে একটা dataflow বানাতে use করি।

Control flow

dataflow name - - - →
True/False 1/0

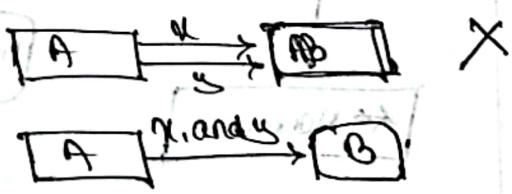
* condition থাকলে control flow use করি।

Rules:-

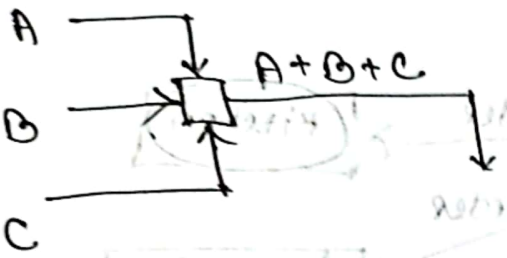
① Dataflow must be relevant.
যে process এর যে function সেরকম dataflow use করবো।

② Each process MUST HAVE one input & one output.

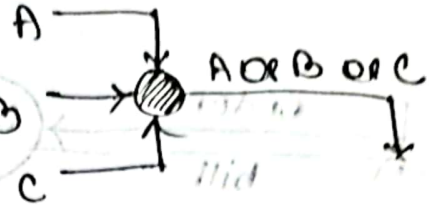
③ Sender and receiver এর মধ্যে single dataflow হতে হবে।



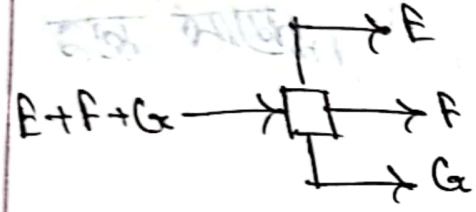
④ There must be a process present infront/behind a dataflow connected to an external pro-agent.
যে কোনো dataflow এর start বা end এ must process থাকবে।



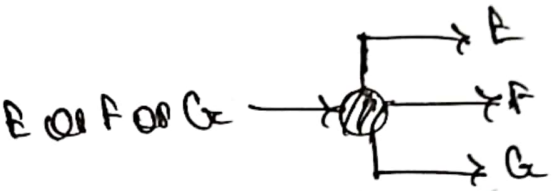
*converging (for AND) data flow [acts as JOIN]



converging (for OR) dataflow [acts as MERGE]

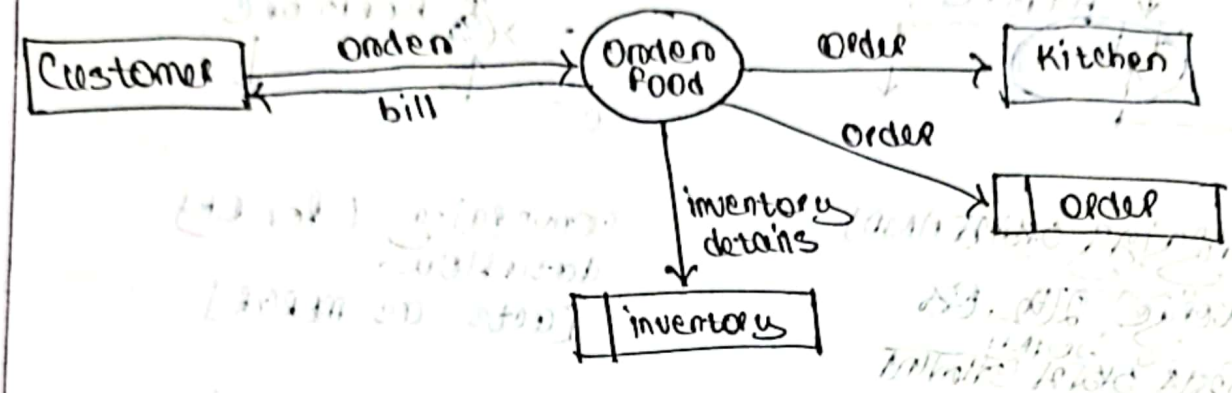


diverging (for AND) data flow [acts as FORK]



diverging (for OR) data flow [acts as DECISION]

* AND operation चलन □
* OR operation चलन ○



Process Decomposition

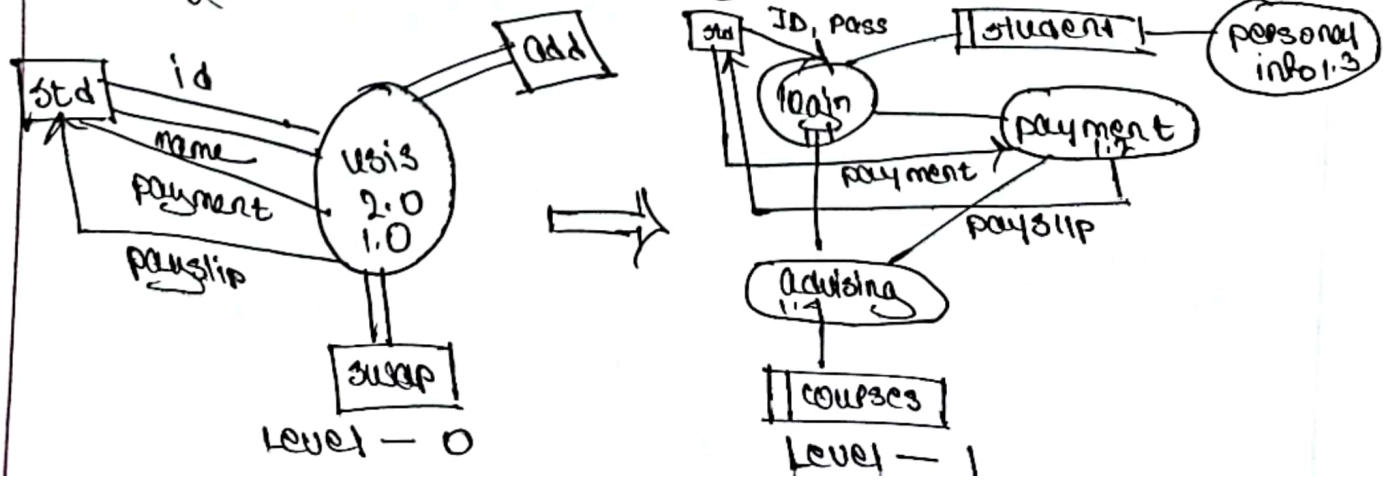
→ এটি subprocess কে ভেঙে ছোট process এ নিয়ে আসব

L-0 context → main()

- A() → a1() → L(2)
  - b1() → L(3)
- B()
- C()

L(1)

*External agent থেকে যত গুলো input output নিল তত গুলো same arrow আসবে in each level. → balancing



Context diagram/0-0 diagram → External agent के साथ
संपूर्ण system के relation

Program Design → for developers
→ instructions given to developers

↓

Physical Model

* input/output কিরকম থাকবে, সেটা use করে physical design বানায়

* conversion of logical model to physical model

* datastore কি হবে, কোন type এর হবে, dataflow কি type এর হবে, process গুলো কি — এগুলো physical model এ থাকবে।

* Human machine boundary — যে গুলো parts human করবে তাদের আলাদা করে দিবে।

Structure chart

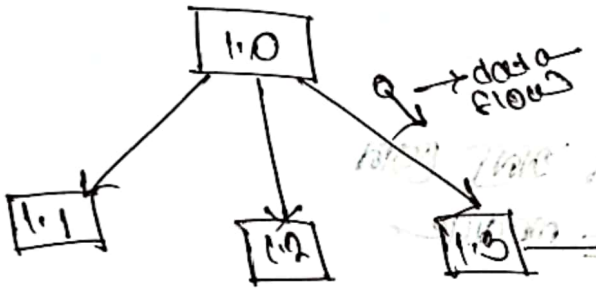
→ gives clear view to programmer how they should code.

→ sequence - যেকোন function টা আগে call করা হয়
- call bluff generate করে

→ selection - condition থাকলে

→ iteration - loop থাকলে

এই সব module/function কে একত্রে থাকলে



* 1.0 এর অধিনে 1.1, 1.2, 1.3 আছে

* □ → module → input নিবে output generate করবে
→ module number + name থাকবে

* [||] → library → already build করা থাকে
module → implement করা থাকে না

* ↻ → loop

*  → data couple condition

→ Data couple.

→ Data দুইটা module এর মধ্যে আদান প্রদান module A pass হয়

→ control couple.

→ non-data flow হয়

→ Off-page.

→ module details অন্য কোন page আছে

→ On-page.

→ same page এ অন্য কোন জায়গায় আছে module details আছে

* Program analyst এরা code create করে।

End of file → একটা file কে close করার মানে প্রোগ্রাম file এর শেষ line এ চলে যাবে end of file আসে

fan-in : একটা module এ যাওয়ার মোট কয়টা module call করেছে

fan-out: একটা module এর মধ্যে কত গুলো call করা হয়েছে

* fan-in ছবিটি ২৩৯ ভালো

program specification

→ document করা হয়

→ যার মাধ্যমে আমরা বুঝি আমাদের কি কি লাগবে program কে design করি

→ এটির উপর base করে code করব

DATA STORAGE DESIGN

→ data কোথায় store করা হবে তার design.

File v/s database

- Data handling needs to be fast and efficient.

- Database এ data store, retrieve, delete করা যায়

- file is used for storing data.
  Same type of data column-row তে থাকে,
  sequentially data থাকে

- Database এ অনেক গুলো table থাকে।
  এক table এর সাথে অন্য table এ connected থাকতে due to relations

relational database — অনেক table থাকে যার সাথে অন্য table এর connection থাকে

Object database — object হিসেবে data storage add করা হয়

multidimensional database — is a type of relational database
— used in data warehousing

Comparison → Files, relational, multidimensional

Relational database

→ অনেক table থাকে

→ primary key থাকে

→ foreign key এটা primary key
এর জন্য table এর মধ্যে connection
create করা

→ data insert/extract করি by
writing command in sql.

Object database

→ person এর info store করে রাখা

Multidimensional database

→ huge database নিয়ে কাজ করে

→ এটা database warehouse এ থাকে

→ এটা database কে ছোট database
A split করে data mart এ store
করে

- moving from logical to physical model → ER, EER

- Normalization এ আমরা একটি de-table ভেঙে আরো ছোট ছোট table বানাই।

- search করা fast, faster in DB.

Conceptual, Logical and physical

Conceptual Model :-

* ER diagram এ table আকারে

- Table এর মধ্যে relationship আকারে

- Initially, আমরা discuss করি, কি কি features থাকবে

- Table + relationship আকারে

Logical model :-

- primary key & data type আকারে
  ↓
  identifies

- relationship এর কোন কিছু info নাই তাই আমরা model create করতে পারব না।

Physical Model

- foreign key দিয়ে আটকে to create relation bet^n two tables.

- এখন DB create করতে পারব।

Cost of DB depends on :-

- Security
- backup server support দিবে
- backup করে রাখবে
- storage

mysql
→ easily hacked
→ backup problem

Optimizing Data Storage →
- → storage efficiency
  - data যত কম জায়গাতে রাখি
  - redundant data remove করি by normalization
- → speed of access
  - ↓ Denormalization
    - redundant data add করি based on my experience to increase speed
  - ↓ Indexing
    - Index page create করা যাবে for faster retrieval
  - ↓ Estimation of size of data for hardware planning
    - DB এর size প্রয়োজন হতে পারে

* joining is an expensive query. অনেক data থাকলে joining use করা খুবই difficult হয়ে যাবে। তাই আমরা order entity তে customer name, id add করে দিব, relation না follow করে, order table থেকেই name, id পাবো। That's why extra data add করি। speed বাড়ে।

* Index নামের আলাদা table বানায় রাখি।

* Index table থেকে আগেই order follows করে specific data retrieve করি without exploring the entire data.

Volumetrics

— estimate size of data

Total record size = Record size + (overhead * Record size)

↳ extra characters

Initial table volume = initial table size * total record size

Total volume @ 3 yrs = Initial table volume

↓

n

+

(growth rate / year * total record size * n)

UI Design

UI এর মাধ্যমে user interaction হয়।

3 click stroke → user যেন 3 click
এর মধ্যে কাজ করতে পারে

External interface এর সাথে UI
interact করে।

UI তে important বিষয় হচ্ছে navigation.

Navigation — এক page থেকে আরেক
page এ যাওয়া।

এখন আমরা GUI use করি instead
of UI.

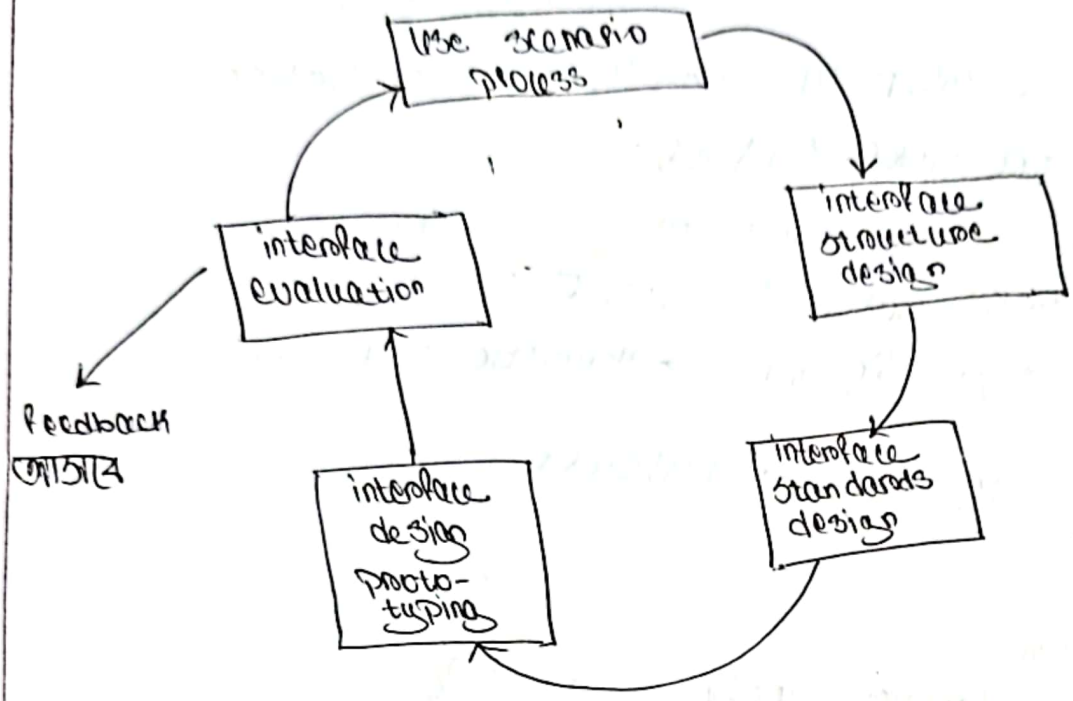
→ Linux এর
interface

User effort minimize করতে হবে।

↙ Simple to use

↘ pleasant to eyes

UI Design Process



Use process scenario → user যা যা steps follow করবে
→ আমরা scenario চিন্তা করি
→ scenario এর উপর base করে DFD include করতে পারি

Interface template → template create করা হয়েছে

prototype → website তৈরীতে যেমন হয় - - - > HTML prototype
→ to খাতায় লিখেই - - -> Language prototype → worst
→ sto Drawing করে explain করি - - - > Storyboard

Interface Evaluation

Usability tools use করে → formal usability tools

client use করে দেখা UI → Interactive evaluation

meeting call করে অনেক মানুষের সামনে present করানো → walk through evaluation

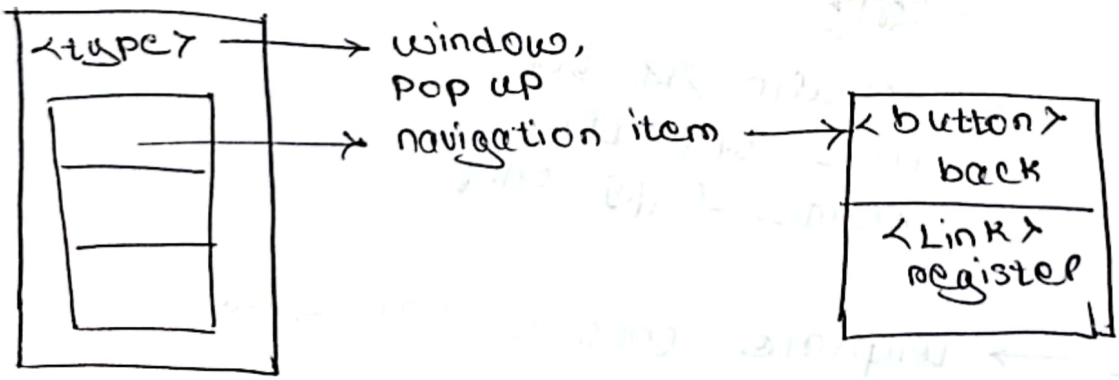
checklist এর সাথে match করে দেখা এর features গুলো ঠিক না → Heuristic evaluation

* এখানে ৫ জন থেকে feedback আসবে।

Navigation Design

→ user manual অনেক complex এর কিছু নিচ্ছি

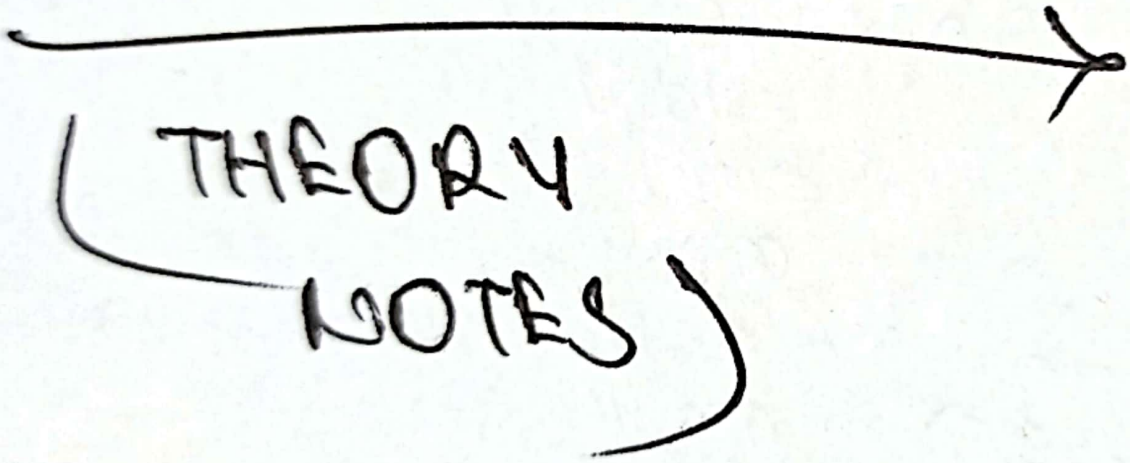
UND



Minimize Keystrokes

→ 250 বিভিন্ন click করতে, 250 টি বিভিন্ন
Keystroke টি, cost বাড়বে।

FINAL



THEORY NOTES

THEORY

LECTURE 8: DATA STORE DESIGN

Database

Relational

- collection of tables, each consisting of a primary key
- one table can be connected to the other by connecting primary key of one table as foreign key of the other.
- major strength: leader in the database market, can handle diverse data needs
- major weakness: can't handle complex data
- data type supported: simple
- Transaction processing & decision making
- existing data format: organizational dependent
- future needs: good future prospects

Object-oriented

- All things created as objects that have both data and process
- changes to one object has no change on the other due to encapsulation
- objects can be reused.
- major strength: handle complex data.
- major weakness: skills are hard to find as technology is still maturing
- Data type supported: complex
- Transaction processing
- existing data format: organizational dependent
- future needs: uncertain future prospects.

Multidimensional

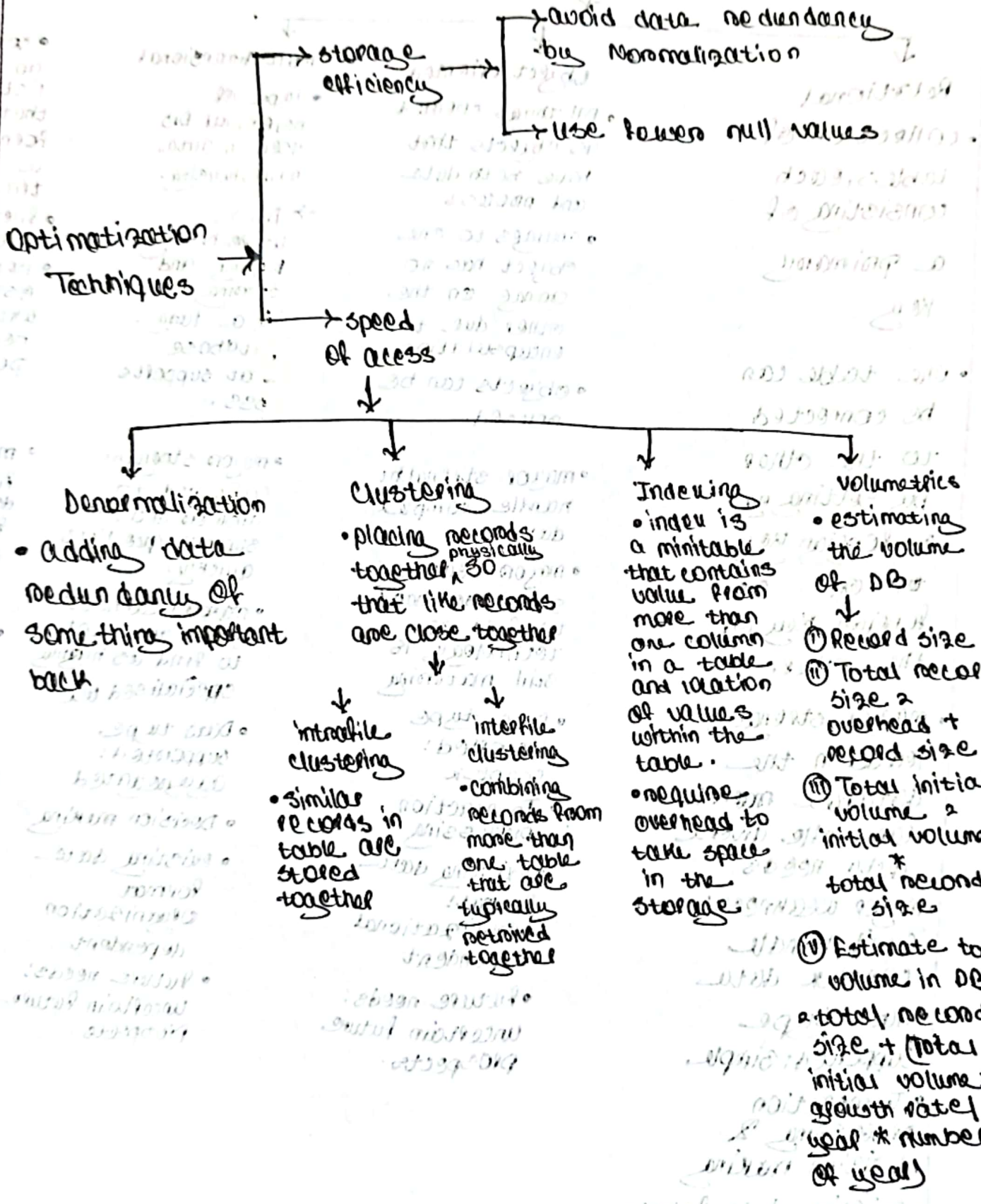
- Type of relational DB used in data ware housing.

→ Data ware housing: taking and storing data in a large database that supports DSS.

- major strength: configured to answer decision support questions quickly.
- major weakness: skills are hard to find as highly specialised use.
- Data type supported: aggregated
- Decision making
- existing data format: Organization dependent
- future needs: uncertain future prospects

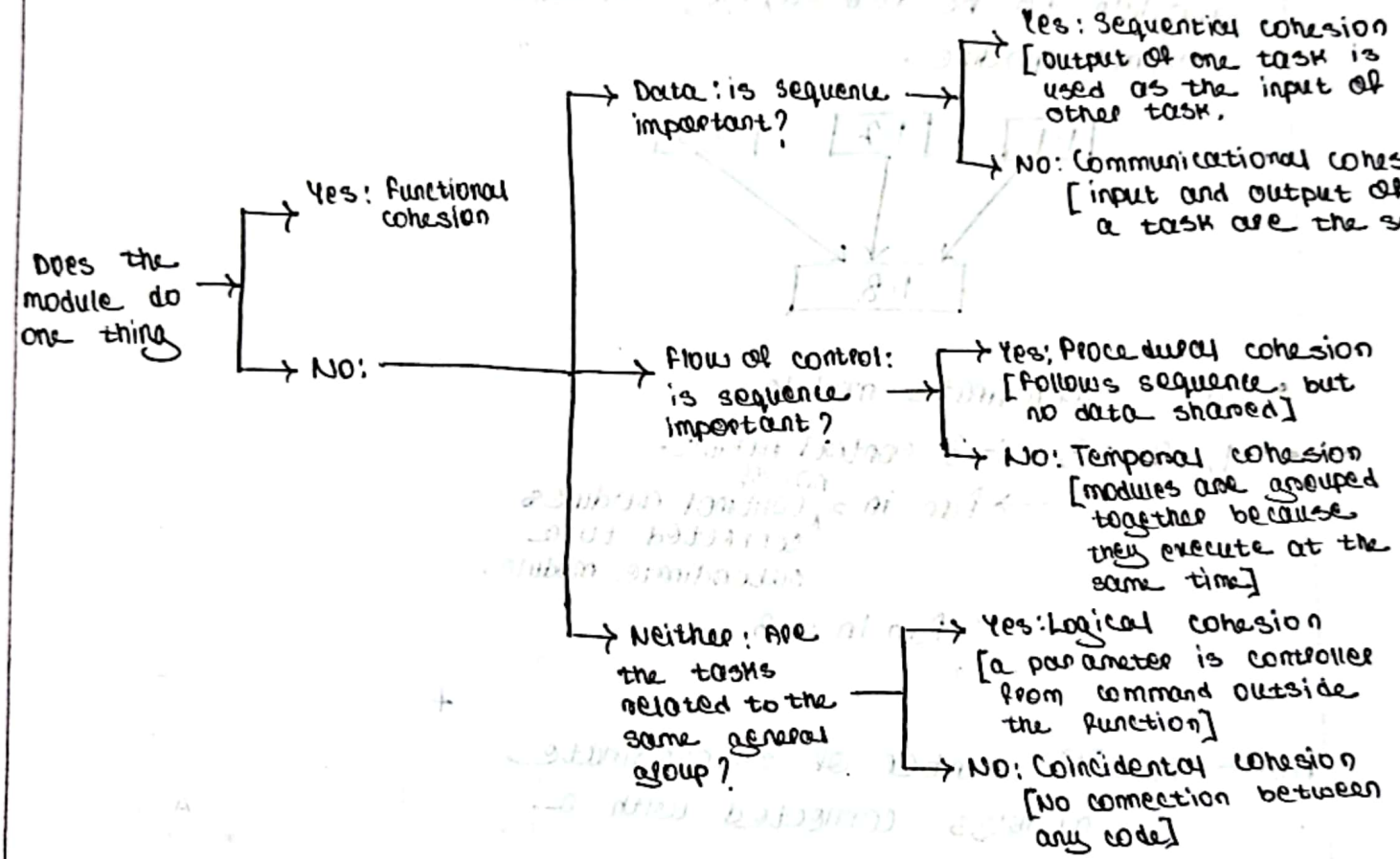
File

- contains of an electric list of info that is formatted for a particular transaction
- files organised sequentially.
- records associated with other records by pointers.
- major strength: file can be designated for fast performance good for short-term data storage
- major weakness: data redundancy data must be updated using program.
- Data type: supported: simple
- Transaction processing
- existing data format: Organization dependent
- future needs: limited future prospects.


Optimization Techniques
storage efficiency
avoid data redundancy by Normalization
use fewer null values
speed of access
Denormalization
• adding data redundancy of some thing important back
Clustering
• placing records physically together so that like records are close together
intrafile clustering
• similar records in table are stored together
interfile clustering
• combining records from more than one table that are typically retrieved together
Indexing
• index is a minitable that contains value from more than one column in a table and location of values within the table.
• require overhead to take space in the storage
Volumetrics
• estimating the volume of DB
(i) Record size
(ii) Total record size = overhead + record size
(iii) Total initial volume = initial volume * total record size
(iv) Estimate total volume in DB = total record size + (total initial volume * growth rate/year * number of years)

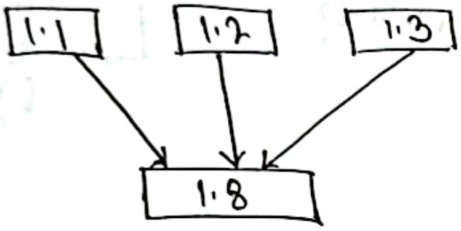
LECTURE 9: PROGRAM DESIGN

Cohesion: how well the lines of code within each module relate to one another.



Fan-IN : The number of control module communicating with a subordinate module.

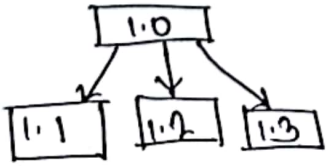
The more fan in a subordinate module has, the more the chances of that module to be reused, called by different control module.



- 1.8 → subordinate module
- 1.1, 1.2, 1.3 → 3 control module
  → fan in = no. of control modules connected to a subordinate module.
  ∴ fan in = 3

Fan-out : The number of subordinate modules connected with a control module.

More than 7 subordinate modules connected to a control module should be avoided to keep fan-out lower.



- 1.0 → control module
- 1.1, 1.2, 1.3 → subordinate modules
  → fan out = no. of subordinate modules connected to
  ∴ fan out = 3