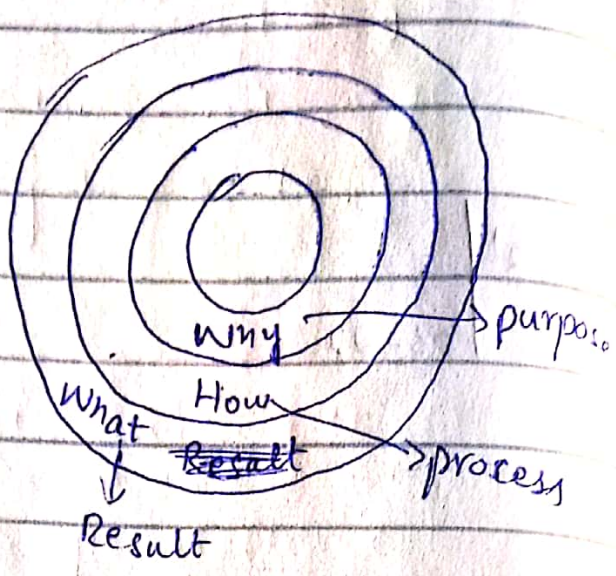
How should you start if you have to build a software

- Golden Circle

- Why → Purpose
- How → process
- What → Result



SDLC

① Requirements

⇒ User stories:

(Why, How, What)

⇒ Wire frames:

user requirements of each screen

T_1 =

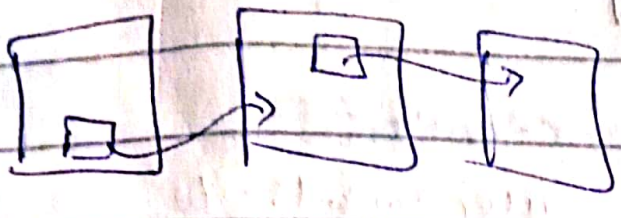
T_2 =

T_3 =

To develop screen, we use pencil tool. Here we identify each element with unique identifiers.

=> Story boards

- graphical representation of all the requirements of user story
  - user requirements
  - you make wireframes
  - ~~combining~~ by combining wireframes, make story.



=> Func/Non-func/User/Business requirements/physical product

- **Func req** → main function/main ~~work~~ work of the system.
  → If main function fails, system not acceptable

- **Non-Functional Requirements**
  → responsiveness, efficiency, performance, security of system.
  → If no-response time (i.e. late reply), no security (any other person can login), system fails

- **User requirements**
  - features of each and every screen i.e. mainframes
  - If features not completed, system fails.

- **Business requirements**
  → what user wants to achieve in his business through software i.e. increase profit
  ie. friendly GUI

- **Physical product**
  → If system is a physical product, it must meet all the requirements, it should be efficient.
  i.e., if a device is made for detecting something in water, but device itself is not waterproof, system fails

=> **Analysis** → breakdown of requirements
- What you need to build the software

- **GPU** → requirements to build Software

Scanned with CamScanner

② Design → UML diagrams

High level

- close to user language

i.e. which dept should linked with which other dept?

Low level

- close to machine language

i.e. which database used, which tables linked, what queries used?

③ Building / Development

→ judge through analysis

- select the language
- build your program that met all requirements

④ Testing

Alpha Testing

→ product tested by developer

- me developer
- other students working with me
- I check their work
- They check my work

Beta Testing

→ other persons (users) tested the product which were not involved in developing.

- beta testing identifies most of the error
- check how product works in real-life

⑤ Maintenance:

Maintenance phase requires 79% of whole effort in total development of SDLC

other SDLC 21%

→ some error fixed, modify the software & resolved the errors

developers

- 2 types of developers in software houses

developer team (skill)

- design & build product in specific time

support team (3 shifts)

- maintain the software
- repeat whole SDLC cycle
  - Design
  - Building
  - Testing

I2C → High level software house

Requirements:

→ Informal definitions

A specific description of client needs which can be used to create a real world product.

→ use case:

In use case, you identify needs & wants

- how to perform some function
- how the user will interact with your product
  - i.e. images, pages, websites.

Needs	Wants ~~(not core)~~
→ main requirement	→ in depth requirements to achieve main requirement
i.e. game	i.e. player color, etc.

→ Formal Definition of Requirements

A condition or capability that must be met by a software or its component to satisfy a contract, standard, specifications or other formal document.

SRS → Software Requirements specifications Document

- requirements
- UML
- time
- budget, wireframe, time

→ client sign + manager sign

25 | 3-22

Requirements Specification

→ write requirements in different forms
→ then develop project

① Requirement Elicitation

An interactive process which occurs when client meets with system Analyst.

→ Interview with client by client analyst

Client	User
• owner of the software • pays for the software	person • using the software

Gathering & Collecting Requirements

- When client not knows everything about software
- have some knowledge or no knowledge about software, which he wants to develop
- System analyst judge the functionality (needs & wants) of software & satisfy the client.
- The first discusion is v. imp.

Bridge the Gap

- understand Why (purpose) of client
- tell him what to do.

→ Don't mix up requirements gathering & requirements elicitation

↓

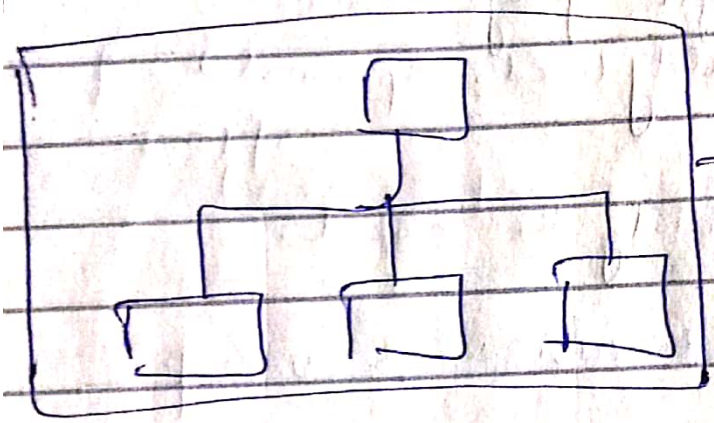
(in-depth discussion with client)

② Expressing Requirements

→ express requirements in different forms

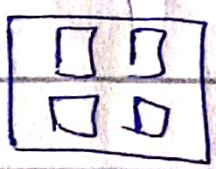
- use diagrams
- points
- schemes

=> gather all the points & express in precise way.



→ story board

=> Meaningful ways to write requirements

- make points
- think as a user, I will use this button to sign up into the system
- writing use cases
- wireframes  make like a screen draw all requirements.
- Storyboards (collection of multiple wireframes)

③ Prioritizing in Requirement

After collecting & expressing requirements,

plan → what develop → { first, second, third

⇒ D/f b/w project & product

- product is the end result of project
- start a project, made a product

How project/product improves?

2 main points/restrictions

(I) Time

(II) Money

Must be done:

Important & urgent (time)
for success

Should be done:

Important but no urgent

Could be done

not important / not urgent
→ makes product attractive

	Urgent	Not urgent
Imp	Q1 Just do it	Q2 live your life now
Not imp	Q3 Filter	Q4 Stop doing it

④ Analyzing Requirements

- clear
- complete
- consistent (if user changes his requirements after development)

Conflicting Requirements

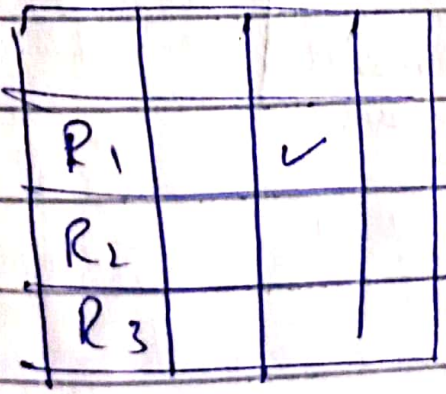
① secure but slow
② fast but less-secure

→ Asked from client
(what should be done or what skipped)

⑤ Managing Requirements

Manage project after development

≈7 Requirement Traceability Matrix



R_1 — first color red
R_1-1 — second time blue
R_1-2 — ~~or~~ orange, etc
R_1-3

↓

Keeps record of everything
→ which person person what requirements
→ req either completed in time or not?

Use Cases: → steps to be followed to execute any function

• Functional Requirements

- core functionality of the product
- a behavior that the developed system should do or support

→ main requirement of the system, system can't exists without it.

i.e. login

it includes:

① input into the product
i.e. username, password

like: we enter any data or email address for any online procedure

② output of the product

As a result of input, we recieve a notification

③ description of the behavior (process)

shall → used • to emphasize something
• to apply force
• to show importance of something

should X

shall → use to write functional requirements

- As a user, I shall login to the system.
- As a user, I shall click the register button to register myself.
- As a user, I shall click the DMS button to see result.

Business Requirements

→ purpose of project
→ benefit provided to business by project

Business rules:
↳ constraints that tell how product will perform a function

Business requirements are implemented in terms of business rules.

Capacity:

→ reverse of scalability

- current requirements
- Functions perform by how many users at same time.

→ 500 users must be able to login at one time.

Availability:

24/7

How?
Tell reason....

- by curing OPS
- by providing unredundant power supply

~~Rata~~

Reliability:

- system must be reliable
- have no error

$2+2=4$

Recoverability:

- If system has any error
  - Run time
  - Syntax
  - Logical
- system rapidly identify it
- So developer recover it

Maintainability

79% efforts require in maintainability

Servicibility

customers, subjects, orders at one time.

Security

secure from ~~usa~~ hacking, etc.

Regularity

must follow regulations of govt.

Manageability

easy to be manageble as you can easily understand text, graphics, buttons.

~~i.e. (color blindl~~

i.e. (for color blindness)
↳ be careful to use red or green colors

Environmental:

System must be environmental friendly.

i.e. If system needs power, it must used solar power, not generator.

Data Integrity:

Software must secure data of his clients.

Usability

easy to use ⟶ How?
- Mention reason

easy to understand as you must be able to read text.

Interoperability:

self explained,

easy for users.

External Interfaces Requirements

Requirements related to how the product is situated within a large system.

language ← GUI, DB

license
↳ another system added to work in a system.

Physical Requirements

- able to meet environmental situation for which the product is made.
- If it is for water, it must be water proof.

Development constraints:

standards you follow while developing any software.

Scenerio:

A student first access homepage of the website.
He sign in, register subjects, view DMC, view attendence. While a teacher add attendence and add marks. A coordinator offer subjects in a semester

Functional Requirements:

- This system shall offer student to sign-in to perform different functions.
- As a user, I shall register my subjects.
- As a ~~User~~ student, I shall see my result.
- As a student, I shall view my attendence.
- The teacher shall update result of students.
- The coordinator shall allow user to register various subjects.

- As a student, I shall see my cgpa.
- The teacher shall update attendence of students.

Non-Functional Requirements

- Performance → A student must take no more time than 5 sec to sign in.
- capacity → handling of things
  - capable of handle 1000 students at a time.
- scalability → 60000 students by next 5 years!
- Utilization → 90% of hardware & software

- Volumetric
  - ↳ data handled by a system at a time.
  - data of 16 ~~students~~ department handled at a time.

- Availability
  - ↳ by ensuring unredundant power supply.

- Reliability
  - ↳ student registered in correct subject.

    Student registered by teacher must be correct.

    Student can see only his DMC.

- Recoverability:
  - ↳ must be efficient to recover errors or faults if occur in future

- Maintainability
  - ↳ easy to maintain
  - try best that no error occur

- Servicibality → students serviced at a time
  i.e. login, register, etc.

- Security
  - → unethical access
  - Physical security → server placed in locked room

- Regulatory
  - → follow regulation
    i.e. system must not be used for hacking, see Privacy/~~Security~~ of other system

- Manageability:
  - → coordinate can see how many students have added.
    - drop students.

- Environmentability
  - → not have hazardous effect on environment
    - not allow other systems to access our system.

- Usability
  - → must be usable
  - → easy to understand text, graphics
  - → handle color blindness
  - → have shortcut keys.
- Interoperability
  - self explained.

→ Long-term strategy

Vision → Vague statement

- ~~Scope~~
  - short statement
  - as generic as possible
- shows purpose of product

→ changes occur in product must be seen in vision or changes may not change the purpose.

Scope

↳ what a project can realistically achieve.

vision → long-term strategy

scope → what included, what not included

elicitation:

↓

interaction b/w users & clients to manage project.

Apple ← vision (security)
- purpose

Scope (How to obtain that security)
- how is purpose achieved?

Vision: (purpose)

Our system will provide functionality to clients to buy products remotely.

~~Purpose:~~

Scope:

How you achieve? what included; what not included?

→ Our system will have functionality where users can buy electronics from their home using credit card & debit card payment.

Scope creep:

- convert vision to scope
- clients requirements increase
- continuous grows or change requirements.

⇒ Expectation clear.

Stakeholders of a software

- any role which has an effect on software and also affected by software
- any one affected by or who has an effect on the success of the proposal

  i.e. clients, managers, end users, administrators.

Types of Stakeholders [3]

① **Primary** → The endusers who actually use the product, perform different functions on software.

② **Secondary** → who occassionally use the product, not regularly use

ie. You login as parent & child see videoes
- you can see history.

③ **Tertiary** → who are affected by the product and make decisions about it

- owner of the software house
- Development Team
- Testers
- Managers

UI, UX

User Interface

↳ whatever you can see/watch on the software are UI.
- buttons
- text
- list
- screen

- must be responsive
- open in any device (mobile, laptop, etc)

→ It is very important for users so must be easy to understand and ~~manage~~ user friendly.

Challenges

Challenges face while developing a software.

- users inability to express what they need in their ~~software~~ (used to) UI.
- Users are biased by previous bad design ⟶ e.g. Uber, Careem
- Developer have trouble seeing through user's point of view.
  → Developer can't understand system analyst.

Solution to Challenges

Always design for the intermediate level users.

Lay → Intermediate → Expert

Users limitations

Limitations which you have to incorporate with development.

① Perceptual / Sensory Limitation

~~Blue~~ we have 6 senses.

If any sense out of order

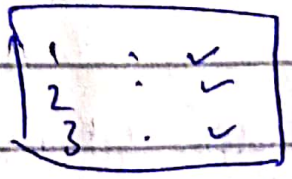
(listening, seeing, color blindness)

② Physical Limitation

: left hand user

Cognitive / Memory Limitation

③ Memorize everything



MS word may corporate it

④ Cultural Limitation

~~cultural~~ cultural (biased due to cultural limitation)

ie, icons, multimedias, etc.

→ proper icon for meat shop.

Diff b/w windows & LINUX

→ windows are user friendly.
→ In LINUX, we put commands for everything but not a single virus has been detected in the history of LINUX

How to gather requirements in industry?

① Interview: (sit down with clients, make UML diagrams, wireframes)

② Focus Group:
A group of people involved in writing the requirements or testing the requirements

- People sit in a circle
- Each has a page.
- Each write requirements on one side
- exchange page with right-sided person
- On other side of page.
- write adv & dis-adv.

⑤ Observing → just observe & write

You visit any shop / industry, there observe and write what everyone is doing there.

i.e. 1 washing clothes
1 person dying clothes, etc.

④ Consulting Previous Products:

Someone need a product like a pre-existing product, you install previous product requirements

→ i.e. install Careem software requirements.

Questionarie → 10000 people
- can't observe all
- can't interview all

→ So you write questions in form
→ ~~All fill form~~ You send form
→ Everyone fill form
→ You get the requirements.

Prototyping:

make a software with small functionality to owner for testing. If provided it then will tell you the actual requirements after using that.

Use Cases:

4-22

Use Cases

- another things to gather requirements (needs & wants) from clients.
- used to identify and write user requirements
- each and every past have specific use cases
  - → steps you follow, elements you use to execute a function in your software.
- An actor performs use cases.

Use Case Format

- **Name:**

  Each and every use case has name

  ie. Sign in
  Sign up
  Buy Products
  View DMC

Participating actors:

- whoever performs the functions
- The main actor who will use and implement the software.

i.e → students who see result on LMS
→ teachers when added result on LMS

Goals:

what you want to achieve by performing use cases

i.e. • I want to buy a product
• I want to see result.

Triggers:

↓

- when a particular function start in the system

when as a system analyst, you create the whole documentation to whom you pass it.

↳ pass to developer to implement.

But the developer must know, when a particular function start in a system.

ie. when a user want to buy product, they will select the payment methods, when a user want to sign in, he will select the sign in.

Pre Conditions:

All those conditions that must be true before executing the success scenario.

→ To open LMS
- a device
- an internet
- a browser

Post- Condition

A condition that must be obtained after the execution of basic flow.

ie. A user has successfully sign in to the account
- A user has successfully buy products

Goal → what user want

Post condition → user have achieved his want

Basic Flows

(success scenerio/main scenerio/ main success)

↓

listing down of steps in points

① Student will open a browser
② Student will open LMS.uet.edu.pk/link
③ Student will click on sign-in button
④ Student will enter username.
⑤ Student will enter password.
⑥ Student will click on sign-in button

• Each Ef every function has different use-cases.

Alternate Flow

If there is more than one methods to perform a function

i.e.

• Sign-in in LMS
  ↳ Not Available

Exceptions

always write in points

actual problems while executing basic flow scenerio.

⇒ While writing requirements, identify problems and write their solutions.

First write the problem, then write all possible solutions for that problem.

① A student does not have account on LMS

Solution: Department cordinator will create his account.

② A student does not have active internet connection.

③ A student does not have device.

④ A student does not browser in this device

⑤ A student forget his password.

Qualities

↳ each and every non-functional requirements.

i.e.

- sign_in function must not be more than 3-sec.

If all points of use-cases working, software working successfully.

Developer → use cases

Tester → Test cases

pass / fail

Viewing Bill

Name: View Bill

Participating actor: Customer

Goals: View the bill for the order

Triggers: Request to view bill

Pre-Conditions:

- Menu items on menus
- Selecting dish
- placing order

Post-Conditions:

View the bill & pay for bill.

Basic Flows:

- User request to view bill
- User views bill

Alternate Flow:

User get wait staff to print and bring them bill.

Exceptions:

No dishes ordered

Qualities:

- Bill available after order placed
- Bill takes less than 10 seconds.
- All selected dishes view on bill.
- Prices on the bill matches prices on menu.

Name: View DMC

Participating Actors: student

Goals: To see result

Triggers: when user click on view DMC button

Pre-Conditions:

- Needed a device
- an active internet connection
- needed browser in device
- an account on LMS
- student must be registered on LMS

Post Conditions:

Result has seen

Basic Flow:

- Student opens a browser
- Student will open LMS.uet.edu.pk
- Student will select sign-in button
- Student will enter name
- Student will enter password
- Student will click on sign-in button
- Student will click on DMC button.

Alternate Methods:

Not Available

Exceptions: (we can take exceptions from pre-conditions)

- Student don't have a device
  Solution: Student will arrange device
- Student don't have internet.
  Solution: Student will arrange internet
- Student is not registered on LMS
  Solution: - He will consult coordinator of his department
  - The coordinator will register him/her on LMS
- Student is registered on LMS but can't see DMC in first semester
  Solution: He will see after first Semester.

- Student forget his password
  Solution: He can reset the password

- Student do not have browser in his device.
  Solution, He will install browser in his device.

Qualities:

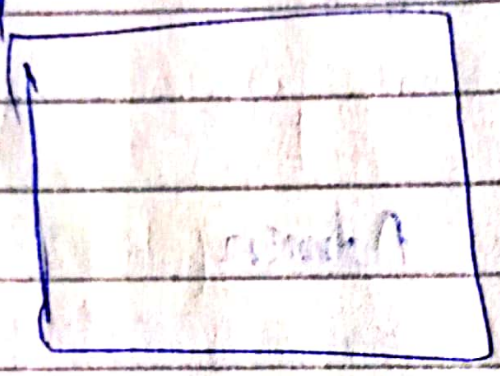
- A student must login to LMS before 3 seconds.
- A student data must be secured from unethical access.
- System must be user friendly so that it is easy to understand text, graphics..

Use-cases

↳ write Ep identify requirements from user point of view.

- Pencil tool is used to make wireframes
- each screen has a boundary
- T_1, T_2, T_3 (identifiers must be outside the boundary)
- unique identification (T_1, T_2) for each function.

In word



- Paste S.S of wireframe in MS word
- Make a Table

	USER	ADMIN
T_1		
T_2		
T_3		
T_4		

⇒ No points : for T_1, T_2, T_3, T_4

Dif b/w wireframes & use-cases

Wireframes	Use Cases
• no arrangement of functions	• explain sequence of steps you have to follow
• tell GUI	• not GUI

15/4-22

Story Boards (pictorial representation of whole project)

- create whole story
- show more than wireframes at one time and join them

Types

Story boards are of two types

1) High level
2) Low level

1) High Level Story boards

- like a comic strip
- overall discussion or overall view of project.
- not provide in-depth details

2) ~~High~~ Low Level Story Boards

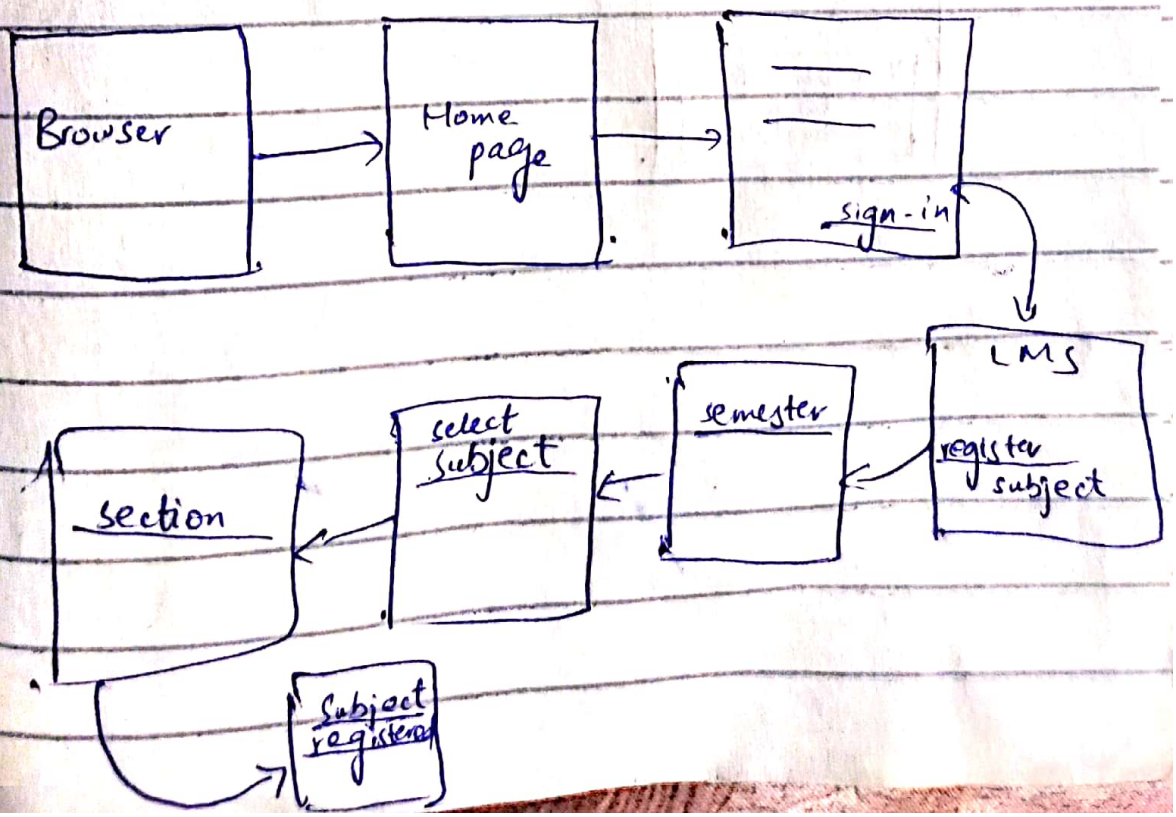
- in-depth detail of system
- provide much detail as compared to high level story boards

i.e. show exact picture, exact color

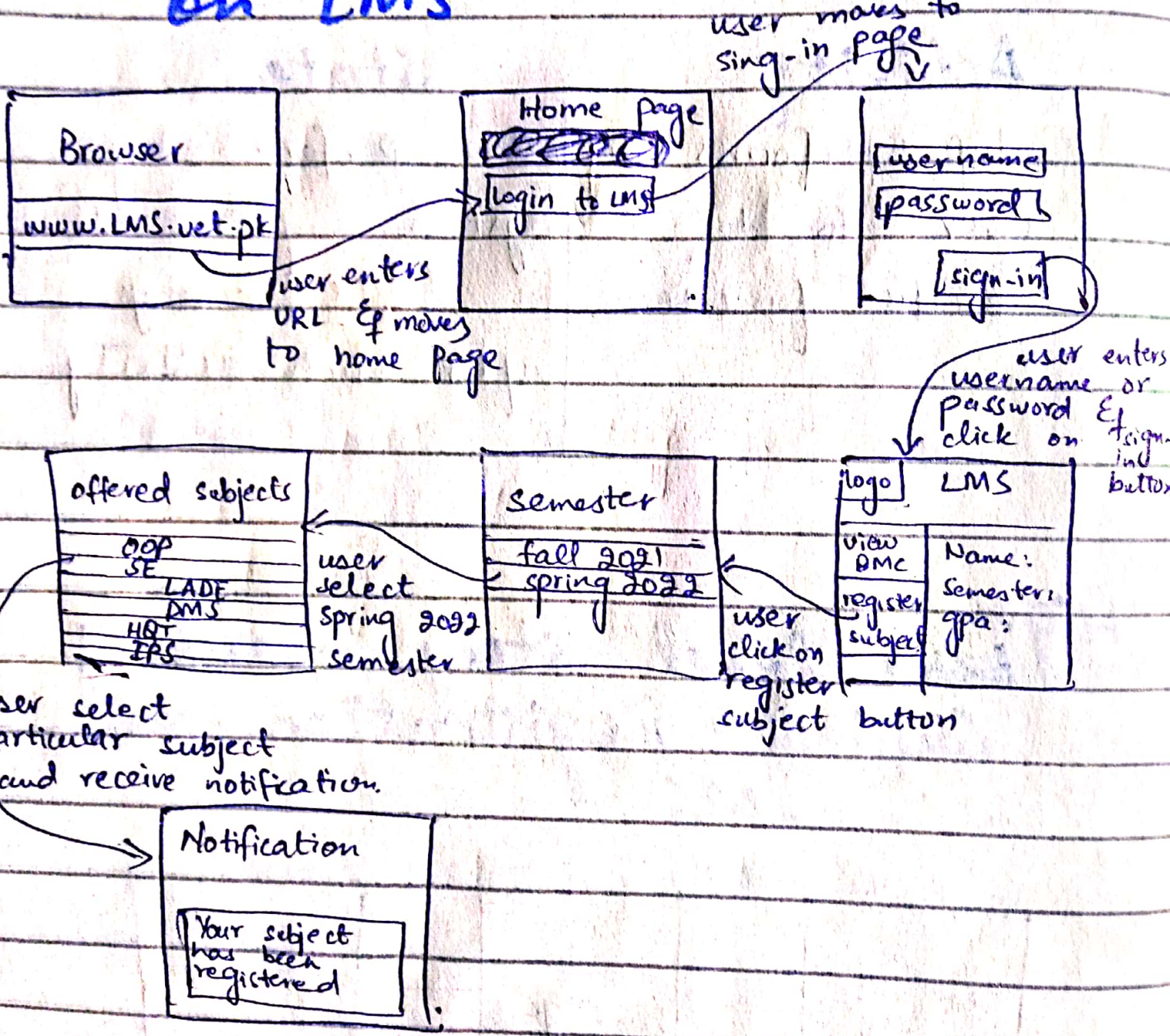
Personas: persons who will use the software

Activity	State
step performs by user to do a specific function	step performs by system after activity. State changes after each activity

High ~~Low~~ level story board of Subject Registration on LMS



Low-level Story board for subject registration on LMS



User Stories

write user requirements in form of text (lines, paragraphs)

i.e.

As a user, I want to be able to identify dietary restriction so that I can eat the food I order.

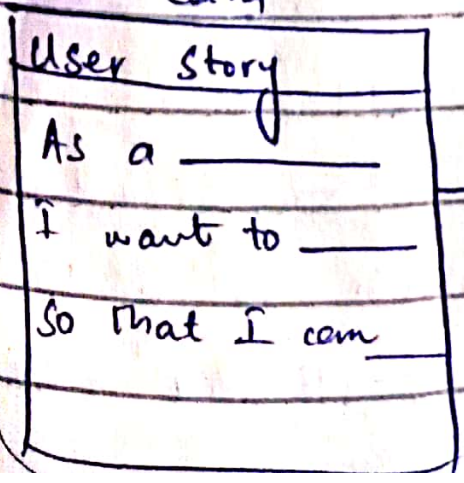
As [who], I want [what] I, so that [why] I want

who → stakeholder role (for the the requirement is formed i.e, user, student, teacher)

what → task or functionality

why → goal, what we want to achieve? benefit of product?

Index Card



→ front end

back-end → acceptance criteria

SDLC → System Development Life Cycle
(How to develop software?)

Agile Method:
↳ most usable now a days

- scrum → implementation of Agile Method
- include customer in development process
  → take customer's feedback during development period.
- If customer's change his mind during development process, Agile method is easy to modify.

SRS
↳ Software Requirement Specification Documents

It includes
- UML
- user story
- story-boards
- use cases
- func requirements
- non-func requirements

Principles you follow while Agile Method

1. Early & Continuous Delivery (call clients after 2 weeks & discuss)
2. Working prototypes as progress (chunk of actual software)
3. Technical Excellence and good design
4. Focus on Simplicity (Simple solution is the best solution)
5. Self organizing Teams
6. Encourage Face to Face Interaction
7. Deliver Frequently (Gantt chart)
8. Welcome Changing Requirements
9. Sustainable Development
10. Build Projects around Motivated People
11. Daily collaboration
12. Reflect on Team Behavior.