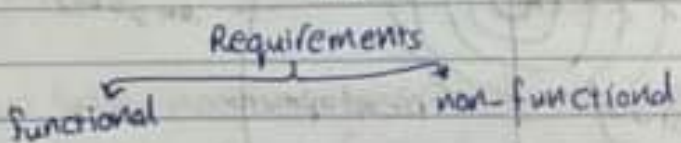
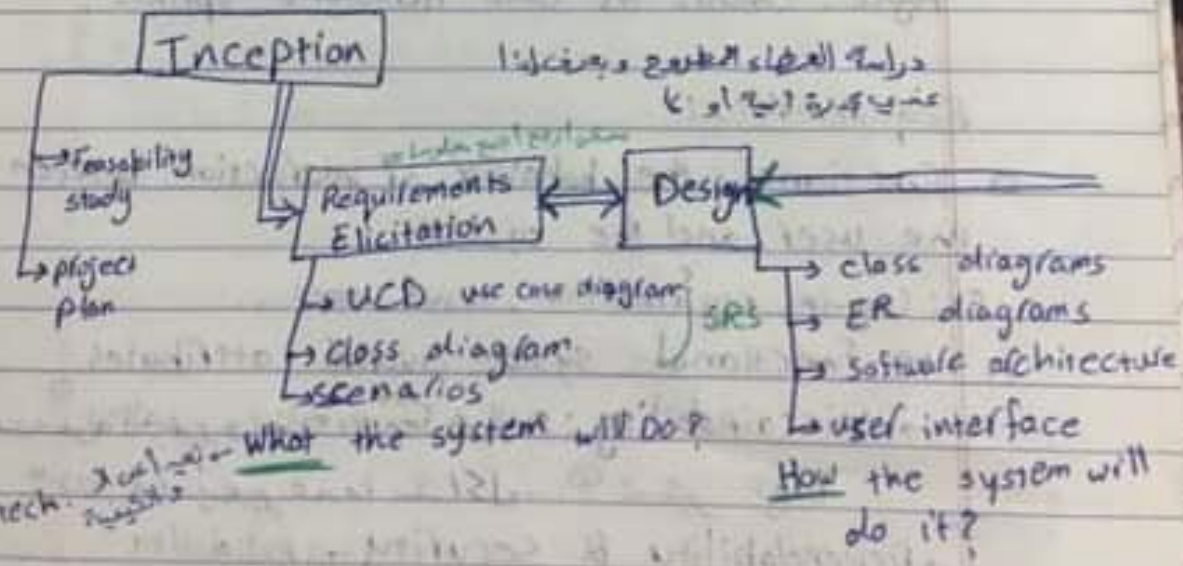
Email : szain@birzeit.edu

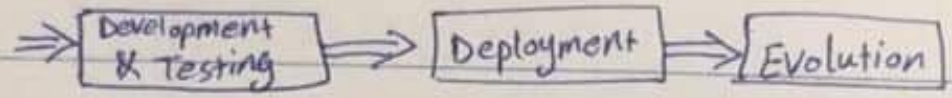
change ⇒ maintainability of software

Non functional requirements:
1. Maintainability
2. Dependability and security
3. Efficiency
4. Acceptability → usability



Software Development process: SDLC

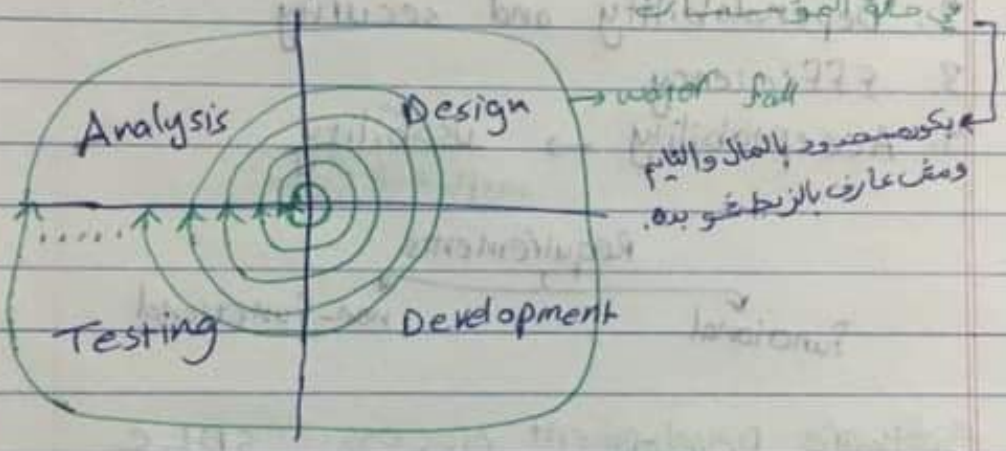




* لما أمشي بالخطوات بدون ما أرجع "linear or water fall" بتزبط
لبعض الـ applications بس مش لكل اشي. مثل الطيران والطب وكل شي critical

interviews: with one group
focus groups: more than one

* لما أبغى برنامج بشكل Agile software development يسمح iterative



Agile: consists of small iterations "sprints".

Requirements:

- Functional: the behavior of interaction between the user and the system.
  EX: log-in, registeration, send memo ...
- non-Functional: system overall attributes.
  → Maintainability: ① مدى سهولة التطوير وإضافة new features على السيستم
  ② تصغير bugs بدون ما يصير عندها مشاكل.
  → Dependability & security → Reliability
  → Safety

Dependability

- Reliability: قديش بقدر النظام يشتغل بدون مشاكل
  وكم بدو ليرجع يشتغل لما إذا صار مشكلة "recovery"
  ~~سكيو ... تم أقل~~ 96%

→ Efficiency
- → CPU
- → RAM
- → storage
- → NT bandwidth

→ Acceptability
- → usability: القياس ممكن: الوقت اللازم لتدريب يوزر جديد
  وكم ممكن يحمل process في زمن معين
  حسب ال application مش شرط D
  كل ما كان أعلى أسوأ في أشياء
  بيطلع عالي بالوضع الطبيعي

⊛ ال functional وال non-functional بيكملوا بعض زي
ال login هي functional بس بلتقي معها ال security -

Ch 2: Software processes → SPs

بيشتركوا بأشياء معينة بس كل شركة بصير العملية الي خاص فيها بناسبها.

- software specification: requirements analysis and validation

- Design & Implementation: build a design "usually UML" and validate design.
- Validation: of design and whole software testing.
- Evolution: changes and improvements.

SPs: complex but rely on judgment

→ critical system: very structured (systematic) process
- منظمة بشكل كثير كبير ومن البداية لازم أعرف كل الـ requirements

SPs
- ⇒ plan-driven, all activities are planned, measured against plan
- → Business system: much cheaper
  - بكتشف الـ requirements أثناء العمل.
- ⇒ Agile, planning incremental, flexible, cope with change.

SW processes
- → waterfall Model
- → Incremental development "Agile" like scrum
- → Reuse-oriented
  - لتقليل استخدامها مثل شركة متخصصة بإشي معين

ممكن أدمج أكثر من model مع بعض بالشغل.

waterfall model : "linear"

- plan-driven
- best for critical systems
- can be used for small systems
- لما بنتقل من خطوة للي بعدها ما بقدر ارجع

advantages:

1. straightforward planning
2. progress easily measured
3. flexibility of working in many projects in parallel
4. customer not strictly required انعزال الكستمر

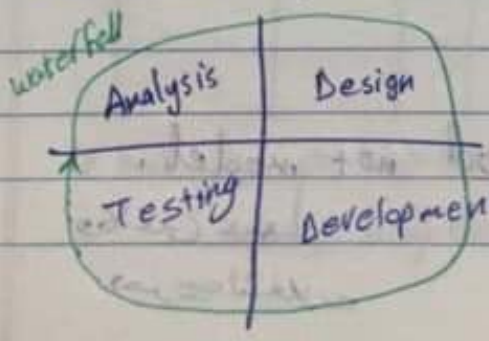
disadvantages:

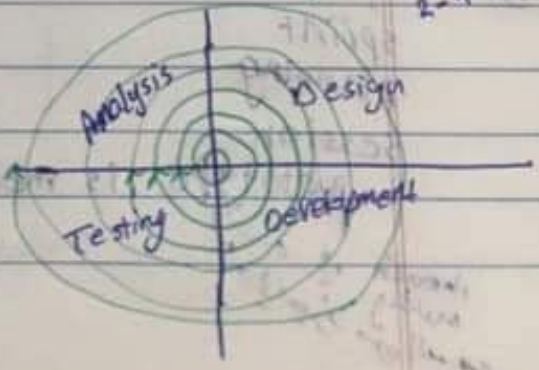
1. effectiveness of requirements
2. requirements analysis is difficult
3. if customer is dissatisfied at end "فش تابعة بعرفينا"
4. testing is at end of project

Formal system development → very expensive

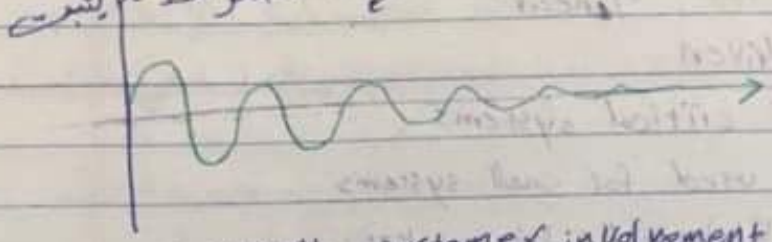
Incremental model : "Agile"

- Business systems





التغيرات بتصغر لحد ما يثبت



- constant customer involvement

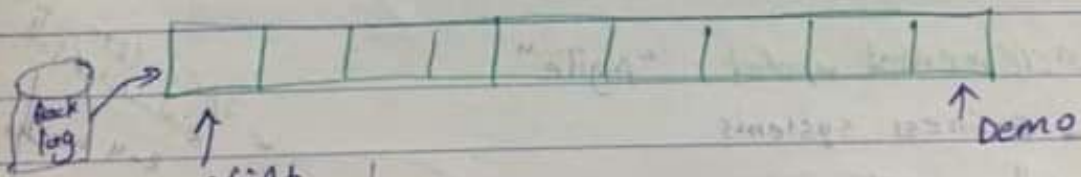
adv:

- cost of changing is reduced.
- customer feedback
- rapid delivery
- better for market competition

disadv:

- process not visible
- increments adding
- syctem arch.

planning on first day of iteration
demo on last day of iteration
and planning for next iteration



prototype is method not model.

though study

scrum → Agile "incremental" slide 37

software management system. ex: Rally

requirement:
→ user story : short paragraph
As a <<role>>, I should be able to _____

back log ← requirements جميع

Rally
Back log

I-II team في Agile عادة

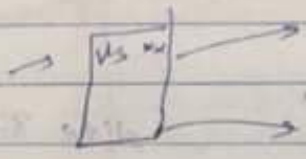
sprint planning

sprint Demo
Retrospective → right & wrong

Dev. name	availability	# of days	Total
Ahmed	1.0	10	60
salwa	0.5	10	30

6 hours/day

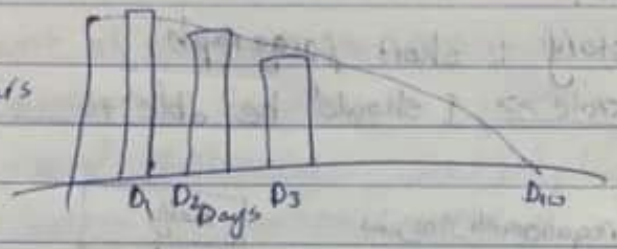
→ Back log → highest risk
or → highest value



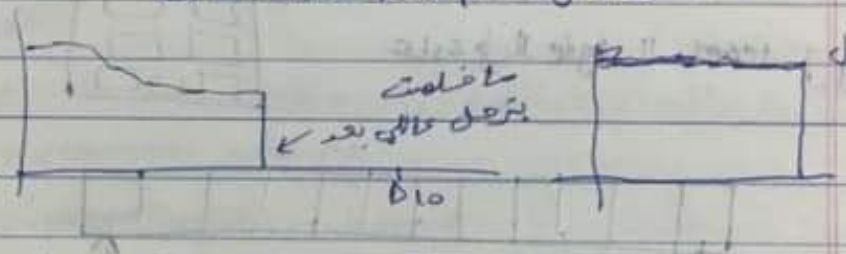
UI Dev. 2 hours Ahmad 58 remaining

0, 1, 2, 3, 5, 8, 13, 20 ← poker estimation

بنخلي نقسم الشغل لحد ما يخلص على ساعات التيم كلهم ،



Demo و planning من ضمن التاسكات



moderator → scrum master

Word fall

ORM: tool to generate code.

spiral model: مكلف جداً وفيه risk كثير

spite: proof of concept.

analysis: ال

١- جمع معلومات

٢- كتابتها

٣- Validation

formal:

Avoide subjective requirements

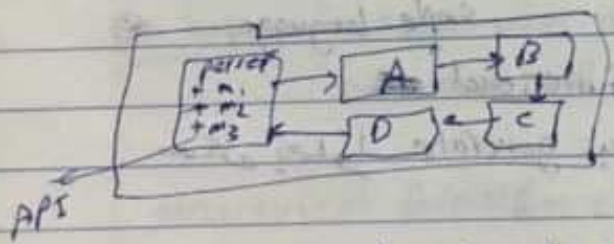
Requirements Document: legal contract

بعدها لذا بدي يغير يغيرك صاحبو.

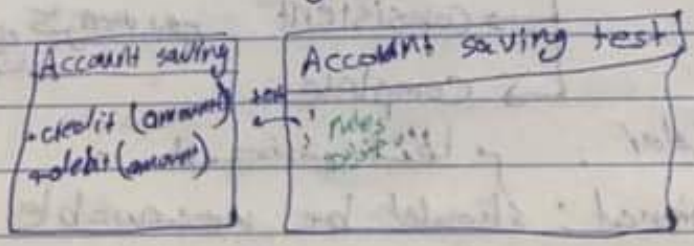
GUI: graphical user interface

Interface: java interface

API: public methods !



Unit test: done by developer, to the developer



Alpha testing: system testing by producer.

Beta testing: system testing by customer

"Acceptance":
- real data
- real use

⇒ shows bugs

Ch 4:

2 levels

1- write high-level requirements "user requirements"

2- detailed description of requirements. "system requirements"

↳ description of main features.

should: سيكون نقاط ضعف عليها

shall: أساسي

* لازم يكون لكل نظام ترقيم معين وتفصيلها الو رقم مربوط فيه

* simple language

functional & non-functional

ممكن يعملوا لبعض وما يكون في تمييز كبير بينهم.

system requirements:
- correct
- consistent — لما يكون كثير users بعبر بتقضاه
- complete

stakeholder: المستخدمين النظام

* non-functional: should be measurable
quantifiable

Metrices for measuring nonfunctional:

1. speed ~~clock~~ ecc or ms of operations ↔ processing response refreshing
2. size Mbytes & ROM chips
3. ease of use: training time & number of help frames
4. Reliability mean time of failure

slide 13

SRS: ~~sys~~ software requirements Document

tracing: for changes

→ free language ↔ user requirements / system requirements
→ use cases

at beg: focus on what should do
not how will do it

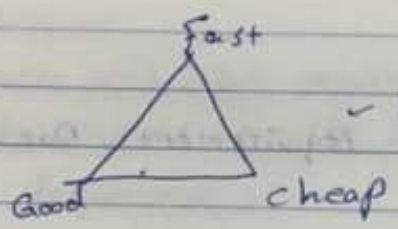
structured format ← critical system medical

discovery
specification ↑ classification ← define subsystem
↑ prioritization
negotiation ↓ Negotiate with SH

Using interviews:

- Focus groups
- observation ← expect resistance of change
- Documents Analysis

why Negotiate with sth?



library search:

UR8. The system shall provide a functionality to allow all system users to search for books.

structured format ← نظام موحد

critical ← أقرب لل

UML:

actors → users

أول خطوة → Define actors "users"?

→ interviews with users

→ scenario is presered on requirement

→ interviews:

- → Closed: predetermined questions
- → open
- → Focus: clusters of sth.

recorder & suitable place

interviewee → open minded
↳ communication skills

→ after interview : send email with summary for agreement

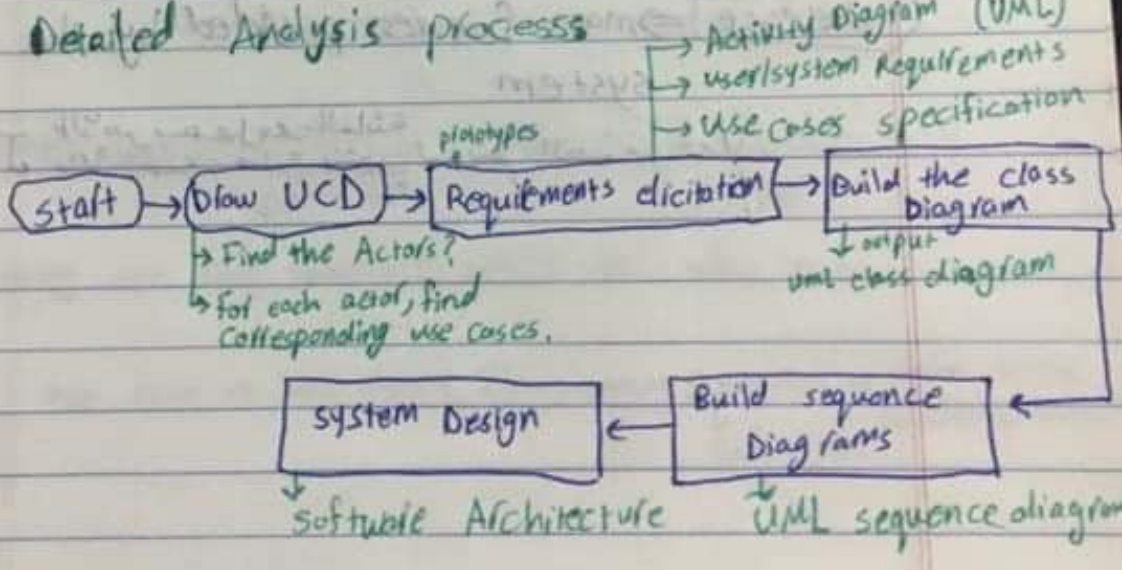
interviews + focus groups → needed

start with normal scenario "success"

search a book → found ← most common
↳ not found

use: user input → system response
for each step.

Detailed Analysis process



Requirement specification

- user stories
- forward
  - user/system
  - use cases

UML class diagram ⇒ to find entities (main entities)

use Diagram ⇒ over all look system صورة النظام

(use cases) الخصائص الأساسية

system ← الناس بيستخدم

Role ← النظام Actors

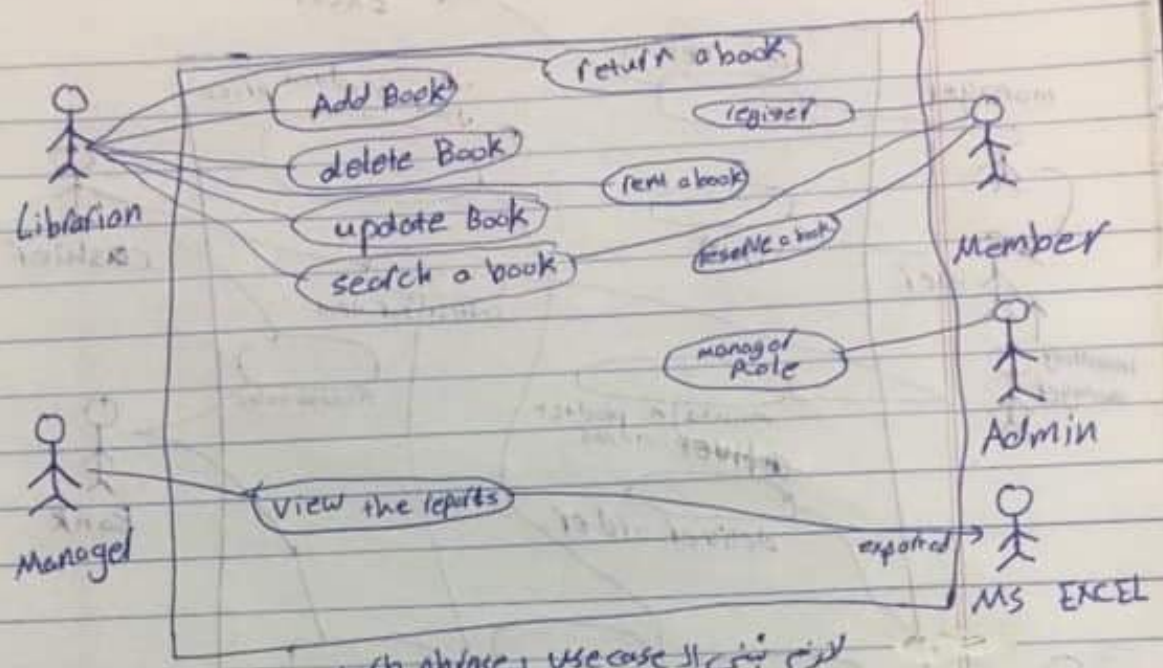
Actor ⇒ Anyone or anything that interacts with our system.

Usecase ⇒ main features provided by the system

بالأمر بمطابقة الفائدة

مثلاً صنف مادة بتحي لما يتم الصنف فعلاً؟

Library system



verb phrase بـ usecase الازم تبني

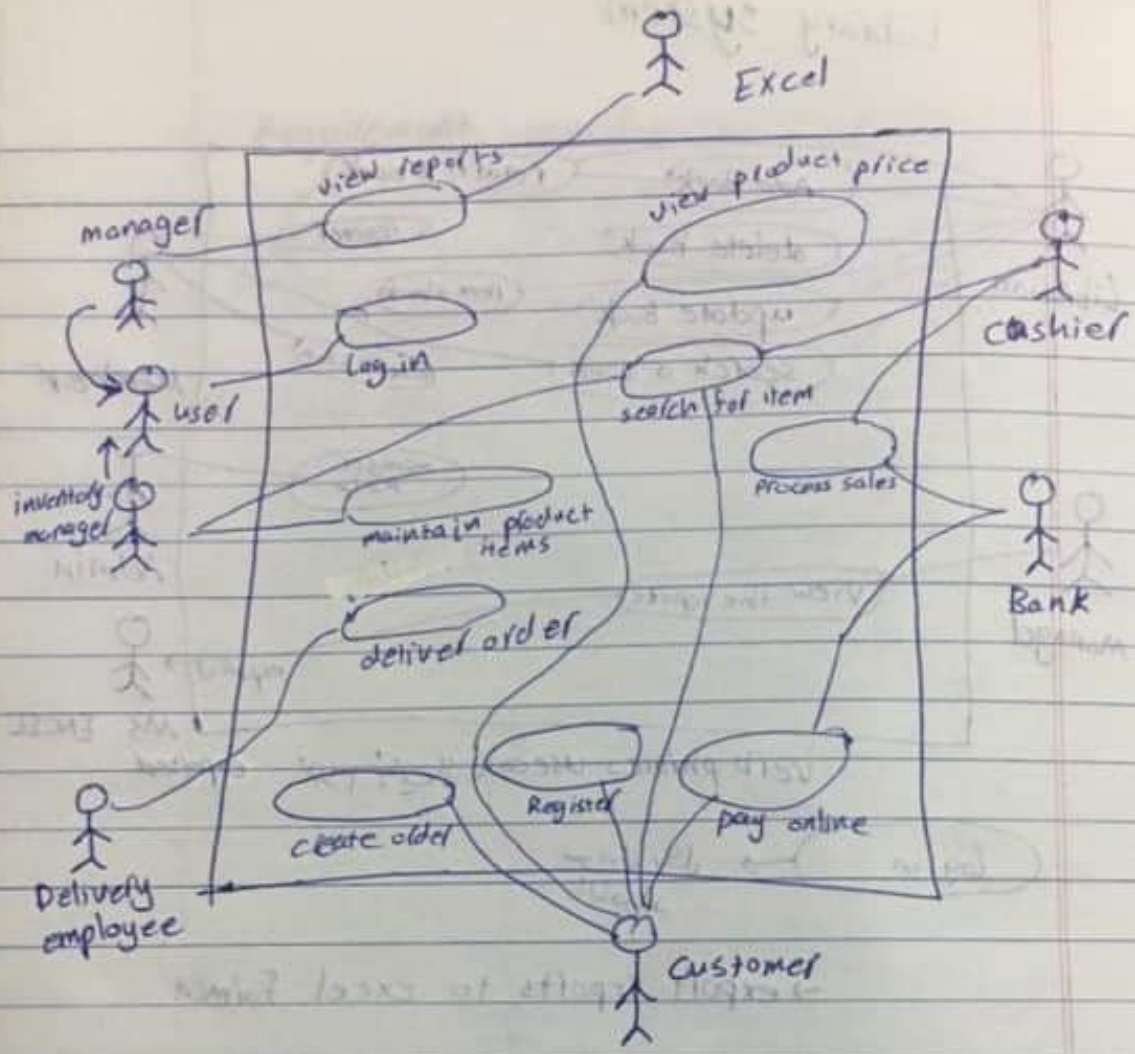
log in → يوصل بكل الافراد

→ export reports to excel format

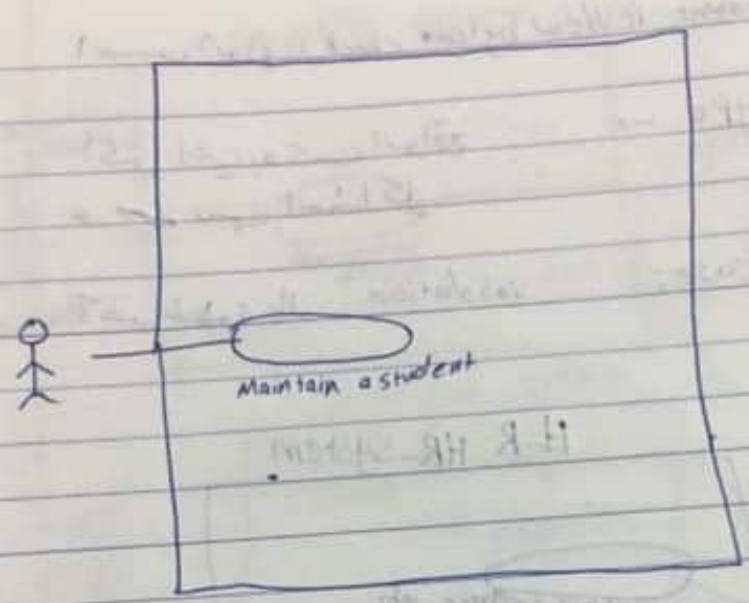
maintain يجمع بعض Add, Delete, update عشان ما يكون الـ digram كتير كبير

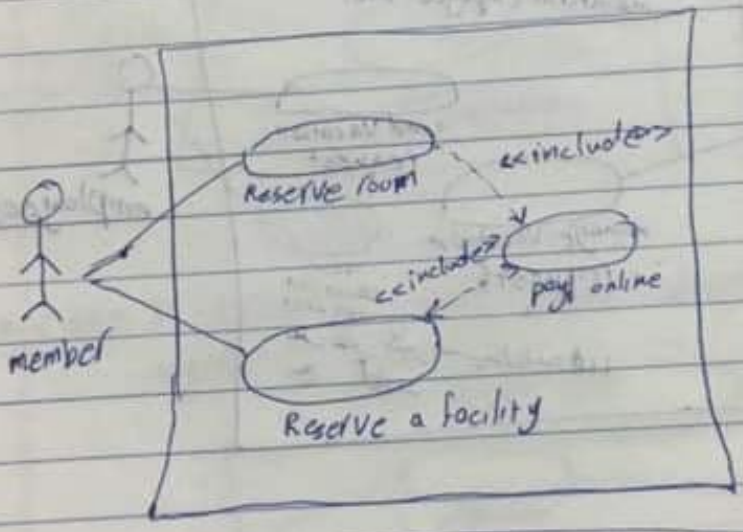
using use case specification بعدها بنكتب كل entity

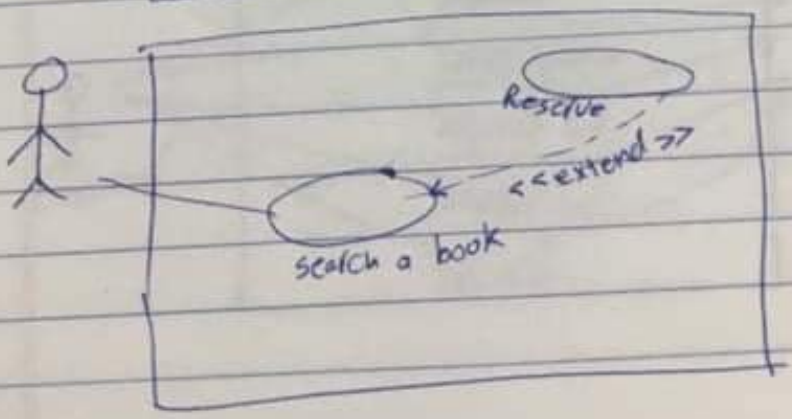
use case ⇒ has many flows → many scenarios → main view
successful view
fail views



user → make inheritance "cashier, manager, inventory"...


Maintain a student

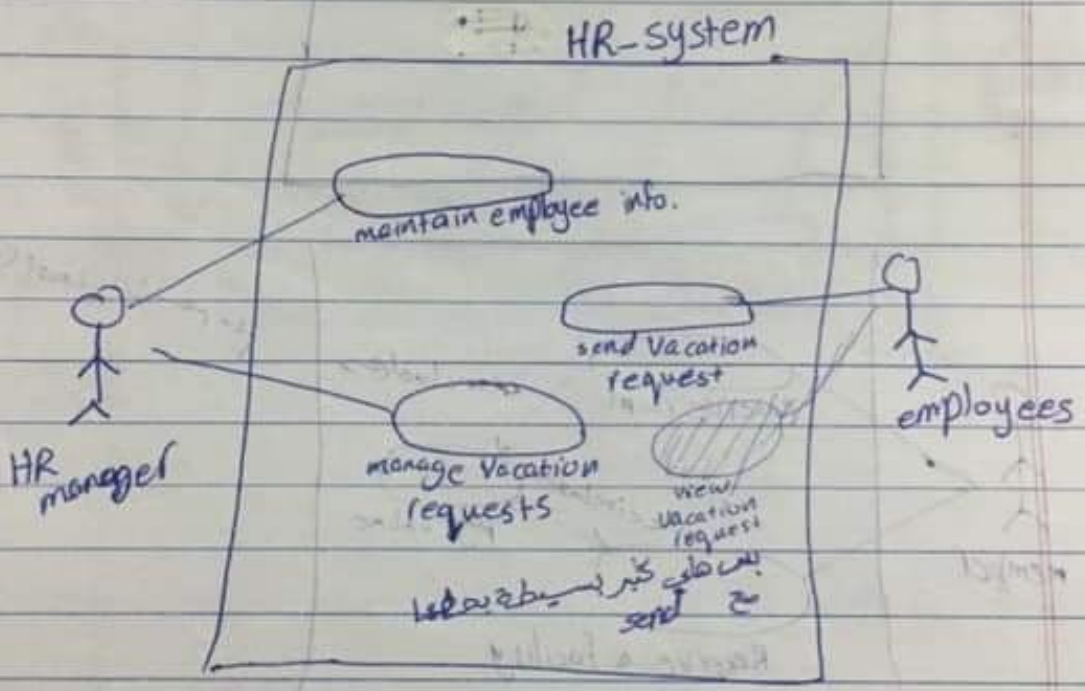

Reserve room
<<include>>
<<include>>
pay online
member
Reserve a facility

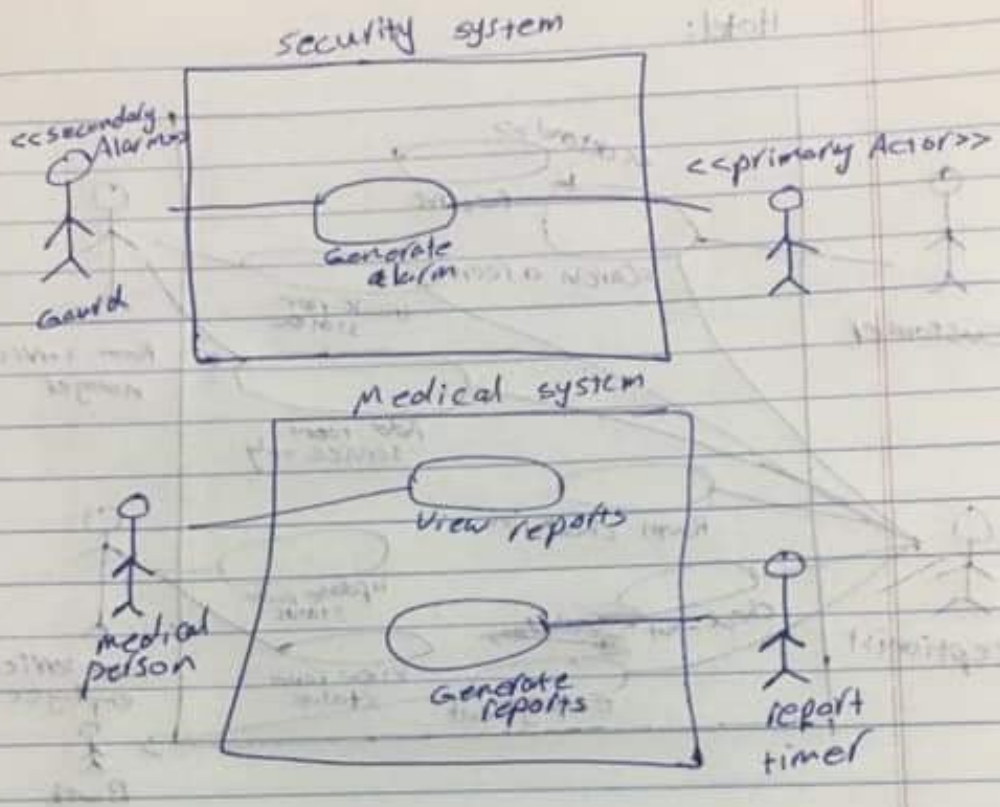

Reserve
<<extend>>
search a book

requirement review by team check أحسن أنواع الـ

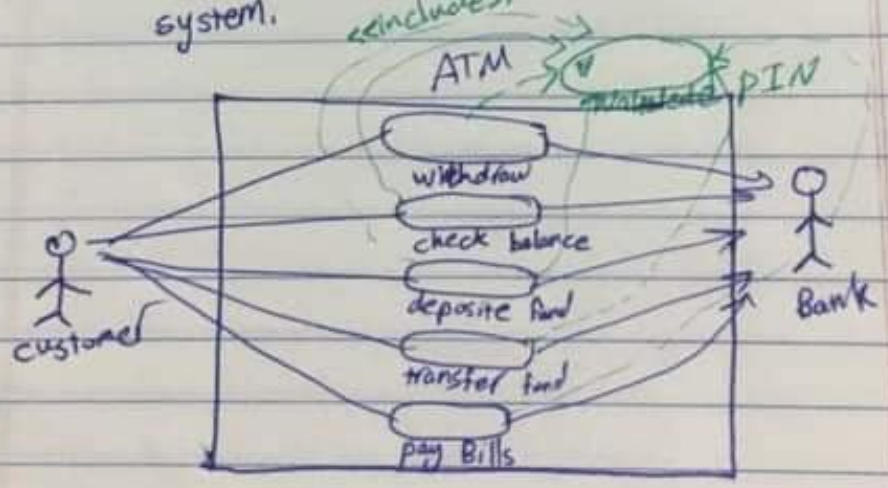
prototype → أكثر اشي يعكس الواقع
و بيبين المشاكل

Test-case: Validation آخر خطوة بال

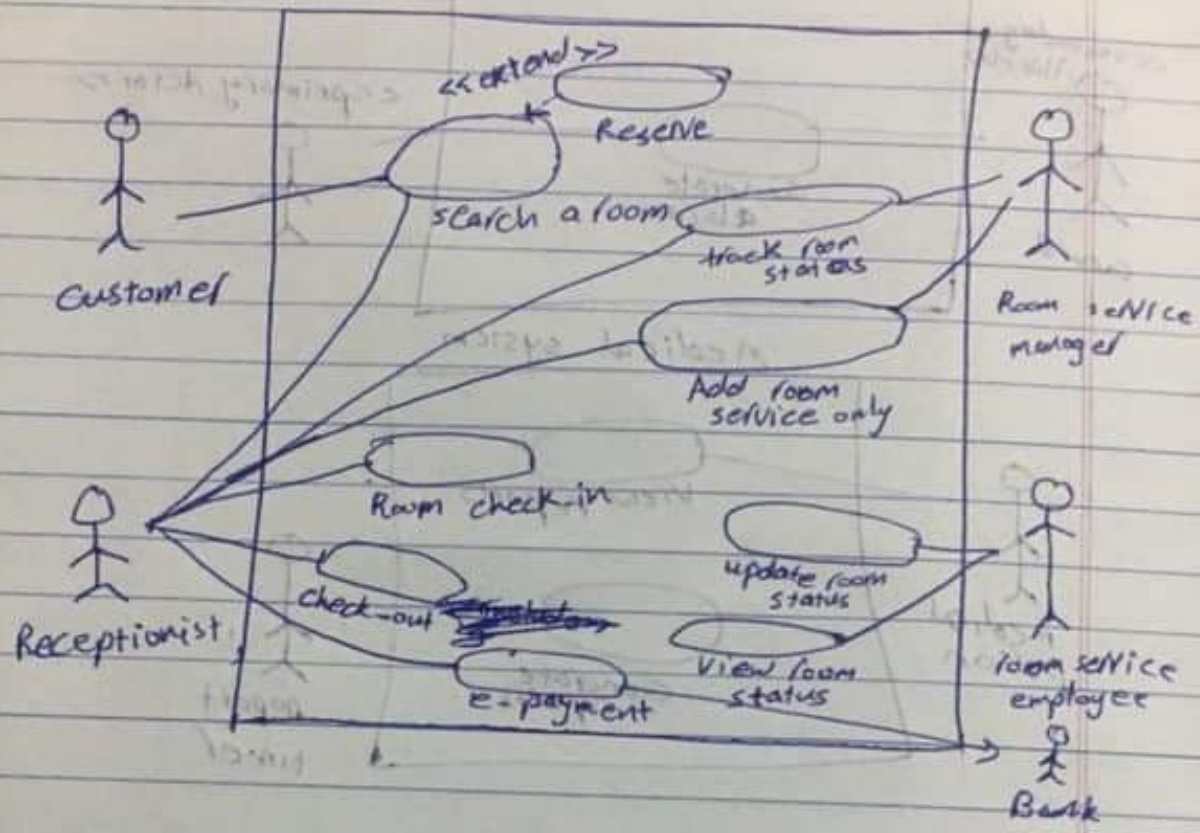




* case 1: Draw a use case diagram for ATM system.



Hotel:


<<extend>>
Reserve
search a room
track room status
Add room service only
Room check-in
check-out
e-payment
update room status
view room status
Customer
Receptionist
Room service manager
room service employee
Bank