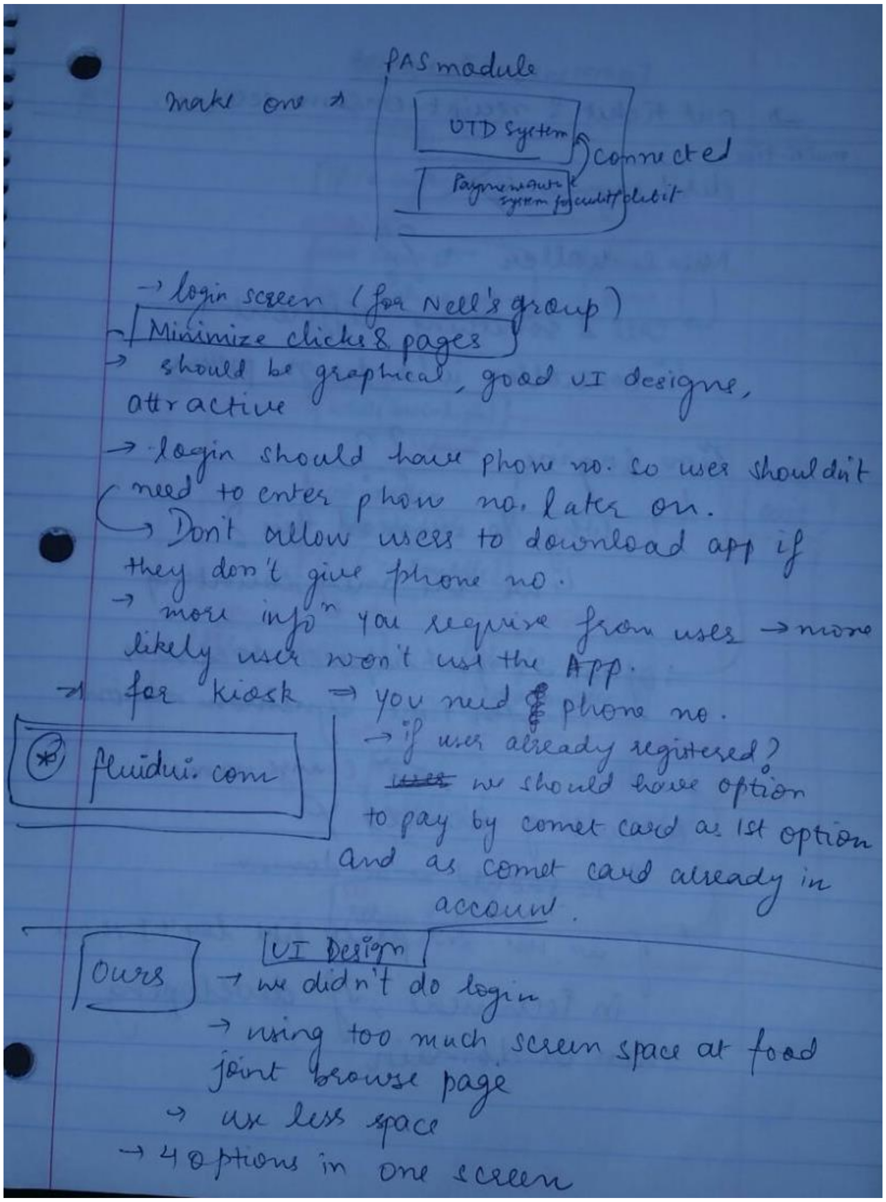


PAS module

make one →

UTD system

Payment/auth system for credit/debit

connected

→ login screen (for Nell's group)

→ Minimize clicks & pages

→ should be graphical, good UI designs, attractive

→ login should have phone no. so user shouldn't need to enter phone no. later on.

→ Don't allow users to download app if they don't give phone no.

→ more infoⁿ you require from user → more likely user won't use the App.

→ for kiosk ⇒ you need phone no.

→ if user already registered? ~~users~~ we should have option to pay by comet card as 1st option and as comet card already in account.

(*) fluidui.com

UI Design

Ours

→ we didn't do login

→ using too much screen space at food joint browse page

→ use less space

→ 4 options in one screen

⇒ put ticket & recipt on same screen.

multi-ties

debit payment ✗

Main Controller → ??

↳ call it something different

∵ controller is a <u>design pattern</u>

Time Engine → ??

↳ why do we need this?

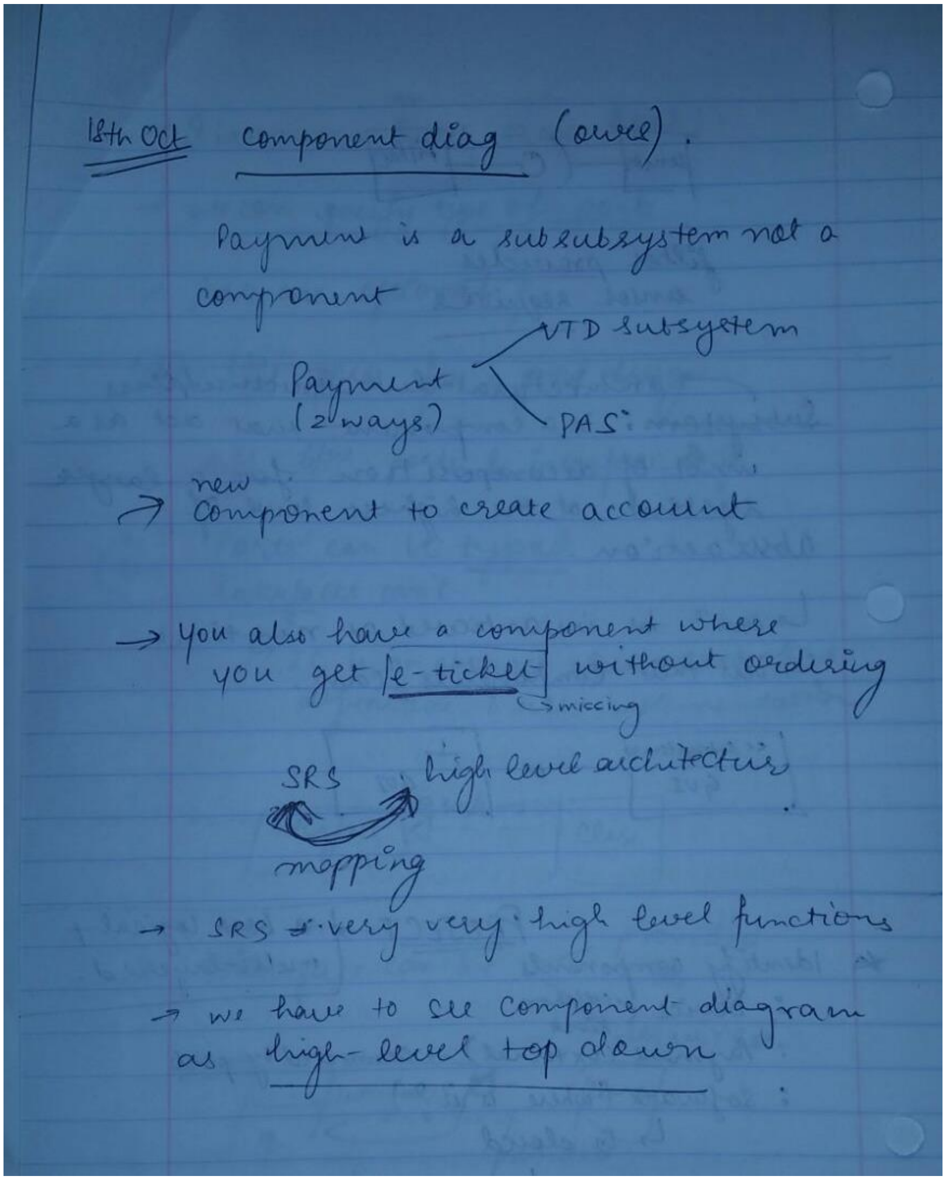
↳ we are only counting

→ Option 2 → push request to server
for time synchronization.

Accounting services φ → change name

~~↳ it should be in domain~~

→ if we use 3rd party like log4J then
in Technical, if developing
then in domain



18th Oct component diag (owee).

Paymen is a subsubsystem not a component

$$\begin{tikzcd} & \text{VTD subsystem} \\ \text{Payment (2 ways)} \arrow[ur, dash] \arrow[dr, dash] & \\ & \text{PAS} \end{tikzcd}$$

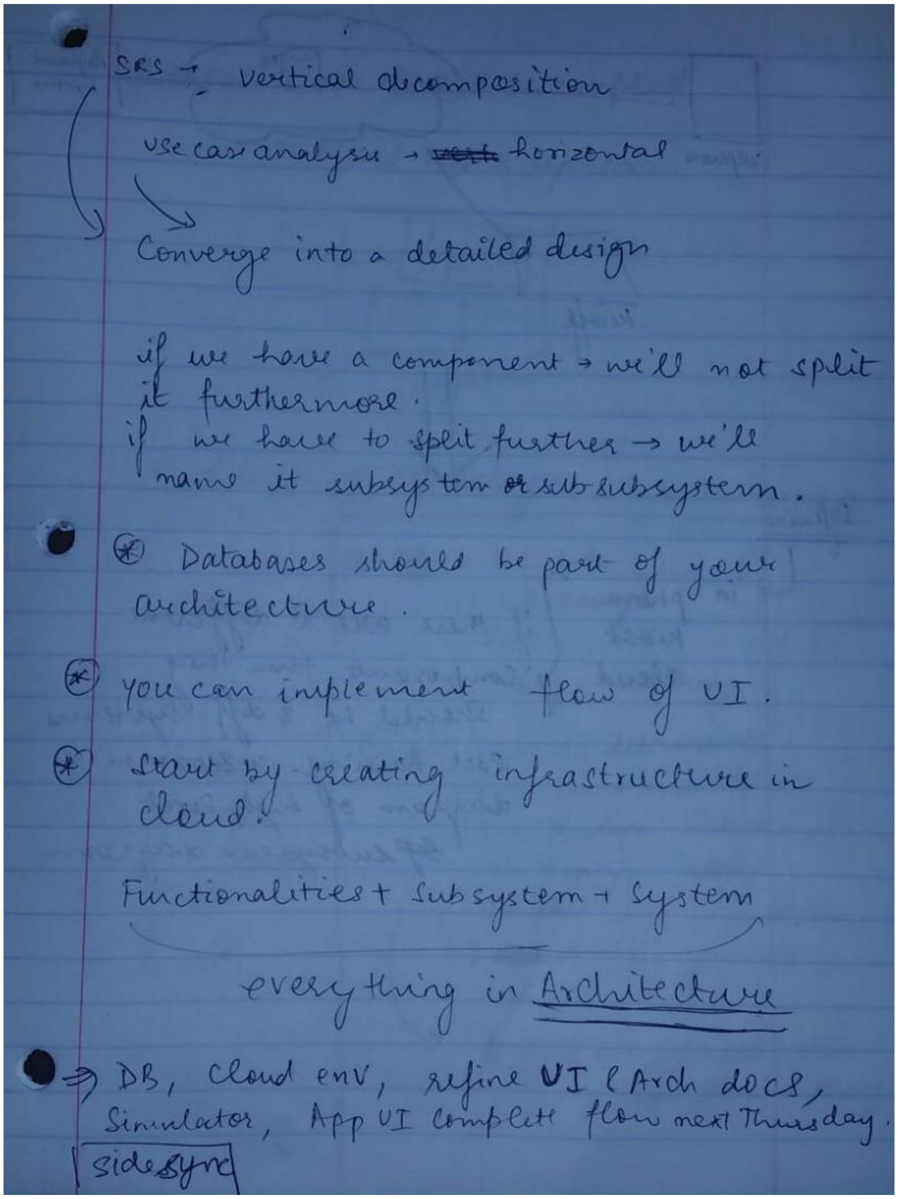
→ new Component to create account

→ you also have a component where you get e-ticket without ordering
missing

$$\begin{tikzcd} \text{SRS} \arrow[r, bend right, "\text{mopping}"'] & \text{high level architectur} \end{tikzcd}$$

→ SRS ~~=~~ very very high level functions

→ we have to see component diagram as high-level top down.



SRS → vertical decomposition

use case analysis → ~~vert~~ horizontal

Converge into a detailed design

if we have a component → we'll not split it furthermore.
if we have to split further → we'll name it subsystem or sub subsystem.

- ⊛ Databases should be part of your architecture.

- ⊛ you can implement flow of UI.

- ⊛ start by creating infrastructure in cloud.

$\underbrace{\text{Functionalities + Subsystem + System}}$

everything in **Architecture**

⇒ DB, Cloud env, refine UI & Arch docs, Simulator, App UI complete flow next Thursday.

sidesync