**SoftDev.**

* Waterfall เทียบกับ Agile
* ระบบควบคุม Centialized vs Distributed
* การ Branch / Merge vs การแก้ไข conflict
* ภาษา Ruby
* SaaS , MVC, REST, FreamWork(Rails)

Q1 : ยกตัวอย่างงานซอฟแวร์ประเภทควรใช้ Waterfall (ลูกค้าอยู่ไกล)

ยกตัวอย่างงานซอฟแวร์ประเภทควรใช้ Agile (ลูกค้าใกล้ชิด)

* Silde week10

Yes มาก Waterfall , No มาก Agile

Q2 : Contralized vs Distributed แตกต่าง ? ข้อดีข้อเสีย ? คุณสมบัติ ?

A : Contralized เป็นแบบรวมศูนย์กลางserverไว้ที่เดียว

ดี : แต่ละคน merge ในแต่ละส่วนก่อนได้ และลดปัญหาคอขวด

เสีย : versionใช้ได้ใน V. เดียว

Distributed เป็นแบบกระจาย

จุดเด่น : ไม่มีคอหนา , versionไม่นิ่ง , แต่ละคนมี repo เป็นของตัวเอง

Q3 : ยกตัวอย่างในการ Branch

A : เวลามีงาน แบ่งงานกันไปทำโดยการ branch ออกจาก master

Q4 : การ Merge vs การแก้ไข conflict

* เมื่อไหร่จะมีการ Merge

A : เมื่องานที่ branch เสร็จแล้ว ก็จะ Merge กลับไปที่ main

* เมื่อไหร่ที่จะมีการ Conflict

A : เมื่อโปรแกรมเมอร์ทำการแปะโค้ด 2 ส่วนไว้ที่เดียวกัน เมื่อ Merge ก็จะเกิด Conflict

* แก้ Conflict อย่างไร

A : มี 2 แบบ 1. คุยกันว่าจะเอาแบบไหน

2. ให้ project manager ตัดสิน

Q5 : ป้องกัน Conflict อย่างไร

A : ทำ Software Engineering แบ่งงานกันให้เรียบร้อย

Q6 : ภาษา Ruby แตกต่างกับ Java ยังไง

A : 1. Ruby ไม่ต้องประกาศ Type ตัวแปร

2. Java ต้องการ Complier ก่อน

3. Ruby สั้น

Q7 : Sass application มันดียังไงเมื่อเทียบกับ Packet application

A : เวลา Update Sass ใส่ที่เดียว เวลาใช้งานใช้ได้เลย ส่วน Packet ต้องใส่หลายที่ เวลาใช้งานต้อง install

Q8 : อธิบาย MVC

A : Model คือ ส่วนที่เราเอาไว้ใช้ทำงานกับ ข้อมูลดิบ ๆ มันจะทำงานเมื่อ Controller ต้องการข้อมูล มันก็จะดึงข้อมูลส่งไปให้ Controller ไปจัดการ ถ้านึกภาพเล่น ๆ ก็เหมือนกับ บรรณารักษ์ ที่คอยเฝ้า, จัดเรียง และสามารถค้นหาหนังสือ ที่เราต้องการได้

Views คือ พริตตี้สาว สวย หมวย ที่คอยเป็นหน้า เป็นตาให้กับโปรแกรมเรา บางครั้ง เราก็จะเรียกมันว่า User Interface เพราะมันเป็นส่วนที่ใช้ติดต่อกับผู้ใช้นั่นเอง

Controller คือ กรรมกรหนุ่มล่ำบึก ที่คอยเป็น เบ้ และ มันสมอง ให้กับโปรแกรม โดยส่วนนี้จะรับหน้าที่ในการจัดการโปรแกรมเรา เช่นการประมวลผล หรือรับข้อมูลเป็นต้น และเป็นส่วนที่เราจะเก็บพวก Logic ของโปรแกรมเราไว้ในนี้เลย

Q9 : REST อธิบายแนวคิด

ประโยชน์ของ FraemWork ? = รวบรวมโค้ดที่ใช้บ่อยๆทำให้ไม่ต้องเขียนเอง

Code ส่วนไหน ไม่ต้องเขียนเอง ? = รวบรวมโค้ดที่ใช้บ่อยๆ

A : REST แนวคิดออกแบบเว็บใหม่

* URI รู้เลยว่าเกี่ยวกับอะไร , ทำอะไร
* Resoure
* Action

Q10 : User Story เอาไว้ทำอะไร

A : เปลี่ยน Requesment สั้นๆว่าระบบต้องการทำอะไร

เทียบเหมือน usecase diagram , ใช้ point ได้ คำณวน point ได้ , ทำทางกายภาพก่อนค่อยทำ sw

เครื่องมือ software

* StoriesOnBoard
* Git , GitHub
* Ruby
* Rails FremeWork
* - Heroku