

## รายงานโครงงาน

เรื่อง Application KMITL News

### นำเสนอ

อ. ปริญญา เอกปริญญา

### โดย

63010366 นาย ธีรวัฒน์	จิตรานุเคราะห์
63010422 นาย ธนภูมิ	ชัยพรรณา
63010487 นาย นพสิทธิ์	จันทวีระกุล
63010519 นาย นันทวัฒน์	รัตนเรืองวิมาน
63010584 นาย ปัญญวัฒน์	ชนัตธาดา
63010633 นาย พนธกร	สุจิตขวานนท์
63010656 นาย พศิน	แก้วนิล

โครงงานนี้เป็นส่วนหนึ่งของการศึกษา

วิชา 01076024 SOFTWARE ARCHITECTURE AND DESIGN ปีการศึกษา 2565

## บทคัดย่อ

ชื่อโครงการ	KMITL News	
ผู้จัดทำโครงการ	63010366 นาย ธีรวัฒน์	จิตรานุเคราะห์
	63010422 นาย ธนภูมิ	ชัยพรรณา
	63010487 นาย นพสิทธิ์	จันทร์วีระกุล
	63010519 นาย นันทวัฒน์	รัตนเรืองวิมาณ
	63010584 นาย ปิณณวัฒน์	ธนทัตธาดา
	63010633 นาย พนธกร	สุจิตขวานนท์
	63010656 นาย พศิน	แก้วนิล
อาจารย์ที่ปรึกษา	อ.ปริญญา เอกปริญญา	
ปีการศึกษา	2565	

โครงการ Application KMITL News มีแนวคิดมาจากปัญหาในชีวิตประจำวันที่ทางผู้พัฒนาได้พบเจอในปัจจุบันหากมีความต้องการติดตามข่าวสารจากทางสถาบันได้อย่างครบถ้วน นักศึกษามีความจำเป็นที่ต้องติดตามเพจเฟซบุ๊กมากกว่า 10 เพจเพื่อตามข่าวได้ครบซึ่งในแต่ละเพจประกอบด้วยโพสต์หลากหลายเรื่องต่างกันไปและ content ในเพจนั้นประกอบทั้งที่นักศึกษาอยากทราบและไม่อยากทราบ อีกทั้งในบางครั้งข่าวสารที่มาจากอาจารย์ของแต่ละคณะนั้นไม่ได้มีการโพสต์ลงในเพจแต่ถูกโพสต์ลงในเว็บไซต์ของทางสถาบัน ถ้าหากทางนักศึกษาจำเป็นต้องเปิดเข้าไปดูด้วยตัวเองซึ่งสร้างความยุ่งยากและอาจทล่นข่าวสารที่จำเป็นได้

ทางผู้พัฒนาได้เล็งเห็นถึงปัญหาเหล่านี้ ด้วยความต้องการที่จะเพิ่มความสะดวกสบายในการใช้งานและแก้ไขปัญหาให้นักศึกษาได้รับข่าวสารอย่างครบถ้วนโดยพัฒนา application KMITL News ที่จะรวบรวมข่าวสารจากทั้งอาจารย์และสถาบันมารวมกันในที่เดียวและยังทำให้ผู้ใช้งานสามารถรองข่าวที่ตัวเองสนใจได้โดยข่าวสารนั้นถูกโพสต์ได้ทั้งแอดมินและผู้ใช้งานแต่ผู้ใช้งานจะต้องผ่านการคัดกรองเพื่อให้ข่าวภายในแอปมีความถูกต้องมากที่สุด

## สารบัญ

เรื่อง	หน้า
บทที่ 1 บทนำ	1
1.1 ที่มาของโครงการ	
1.2 จุดประสงค์ของโครงการ	
บทที่ 2 ภาพรวมและการออกแบบระบบ	2
2.1 ภาพรวมขั้นตอนการทำงานของระบบ	
2.1.1 ฟังก์ชันใช้งาน	
2.1.1.1 Login/Register	
2.1.1.2 OTP	
2.1.1.3 Filter	
2.1.1.4 Tags display page	3
2.1.1.5 Profile page	
2.1.1.5.1 Profile ตัวเอง	
2.1.1.5.1.1 แก้ไขโปรไฟล์	
2.1.1.5.2 Profile คนอื่น	
2.1.2 ฟังก์ชันแอดมิน	4
2.1.2.1 แอดมิน Display page	
2.1.2.1.1 Admin Filter	
2.1.2.2 แอดมิน Post approve	
2.1.2.3 แอดมิน Report	
2.1.2.4 แอดมิน Verified	
2.2 การออกแบบหน้า	5
2.2.1 ออกแบบหน้าผู้ใช้งาน	
2.2.1.1 ออกแบบหน้า Login/Register	
2.2.1.2 ออกแบบหน้า Login	6
2.2.1.3 ออกแบบหน้า Register	7
2.2.1.4 ออกแบบหน้า Display feed	8
2.2.1.5 ออกแบบหน้า Filter	9
2.2.1.6 หน้า Profile	10
2.2.1.7 ออกแบบส่วนเสริม	12

เรื่อง	หน้า
2.2.2 ออกแบบหน้าแอดมิน	13
2.2.2.1 ออกแบบ Display feed admin	
2.2.2.2 ออกแบบหน้า Filter ของแอดมิน	14
2.2.2.3 ออกแบบหน้า Post approve ของแอดมิน	15
2.2.2.4 ออกแบบหน้า report ของแอดมิน	16
2.2.2.5 ออกแบบหน้า verified account ของแอดมิน	17
บทที่ 3 การประยุกต์บทเรียน	18
3.1 Event Storming	
3.1.1 Collect Domain Events	
3.1.2 Refine Domain Events	19
3.1.3 Track cause รวมกับการใส่ aggregates และ จัดการโยงลำดับ	20
3.2 UML	21
3.2.1 UML Component	
3.2.2 domain model	22
3.2.2.2 ContentContext	
3.2.2.3 AdministrationContext	23
3.3 Quality Attribute	24
3.3.1 Availability	
3.3.1.1 สามารถ Display feed	
3.3.1.2 ความสามารถในการเชื่อมต่อ Database	
3.3.1.3 แสดง Tags	25
3.3.1.4 จัดการ content	
3.3.2 Integrability	
3.3.2.1 Pub.dev	
3.3.3 Modifiability	26
3.3.3.1 การแก้ไข code ในส่วน Backend	
3.3.3.2 การแก้ไข code ในส่วน frontend	
3.3.4 Performance	27
3.3.4.1 การส่ง request ข้อมูล	

เรื่อง	หน้า
3.4 Software Architectural Style	28
3.4.1 MVC	
3.4.2 Plug-in	
3.4.3 Representational State Transfer (REST)	
3.5 Design pattern	29
3.5.1 Façade	
3.5.2 Iterator	30
บทที่ 4 การพัฒนา	31
4.1 Frontend	
4.1.1 ภาษาที่ใช้พัฒนา Frontend	
4.1.2 Main Page	
4.1.2.1 userdisplay.dart	
4.1.2.2 post_page.dart	32
4.1.2.3 editprofilepage.dart	
4.1.2.4 othersprofilepage.dart	33
4.1.2.5 userprofilepage.dart	
4.1.2.6 home.dart	34
4.1.2.7 login.dart	
4.1.2.8 register.dart	35
4.1.2.9 postapprove.dart	36
4.1.2.10 postreport.dart	37
4.1.2.11 userreport.dart	38
4.1.2.12 userverified.dart	
4.1.3 Widget	39
4.1.3.1 circle_button.dart	
4.1.3.2 create_post_contrainer.dart	
4.1.3.3 icbutton.dart	40
4.1.3.4 profile_container.dart	

เรื่อง	หน้า
4.1.3.5 navigation_drawer.dart	41
4.1.3.6 post_contrainer.dart	
4.1.3.7 tag_button.dart	42
4.1.3.8 emailfield.dart	
4.1.3.9 passwordfield.dart	43
4.1.3.10 bottom_banner_ad.dart	
4.1.3.11 tags_field.dart	44
4.2 Back-end	45
4.2.1 ภาษาที่ใช้ในการพัฒนา backend	
4.2.2 Data	
4.2.2.1 DataContext.cs	
4.2.3 Controller	46
4.2.3.1 AdminController.cs	
4.2.3.2 AdvertiserController.cs	
4.2.3.3 LoginController.cs	
4.2.3.4 PostController.cs	47
4.2.4 Model	48
4.2.4.1 Advertiser.cs	
4.2.4.2 Post.cs	
4.2.4.3 Posts_Users.cs	
4.2.4.4 Tags_Follows.cs	49
4.2.4.5 Tags_Posts.cs	
4.2.4.6 User.cs	
4.2.4.7 User_Follows.cs	50
4.2.4.8 Users_SharedPosts.cs	
4.2.5 Program.cs	
4.3 Github	51
4.3.1 Frontend	
4.3.2 Backend	

เรื่อง	หน้า
บทที่ 5 การแบ่งงาน	52
5.1 ฝ่ายในการทำงาน	
5.1.1 ผู้ดูแลฝ่าย Backend	
5.1.2 ผู้ดูแลฝ่าย Frontend	
บทที่ 6 สรุปการทำงาน	53

## สารบัญรูปภาพ

รูป	หน้า
รูปที่ 1 ออกแบบหน้า home	5
รูปที่ 2 ออกแบบหน้า login	6
รูปที่ 3 ออกแบบหน้า register	7
รูปที่ 4 ออกแบบหน้า Display feed	8
รูปที่ 5 ออกแบบหน้า Filter	9
รูปที่ 6 หน้าโปรไฟล์ผู้ใช้งานท่านอื่น	10
รูปที่ 7 ออกแบบหน้าโปรไฟล์ตัวเอง	11
รูปที่ 8 ออกแบบส่วนเสริม	12
รูปที่ 9 ออกแบบหน้า Display ของแอดมิน	13
รูปที่ 10 ออกแบบหน้า Filter ของแอดมิน	14
รูปที่ 11 ออกแบบหน้า Post approve ของแอดมิน	15
รูปที่ 12 ออกแบบหน้า report ของแอดมิน	16
รูปที่ 13 ออกแบบหน้า verified account ของแอดมิน	17
รูปที่ 14 Collect Domain Events	18
รูปที่ 15 Refine Domain Events	19
รูปที่ 16 Track cause	20
รูปที่ 17 component diagram	21
รูปที่ 18 UserContext	22
รูปที่ 19 ContentContext	22
รูปที่ 20 AdministrationContext	23
รูปที่ 21 import font awesome	28
รูปที่ 22 Façade ใน DataContext	29
รูปที่ 23 Iterator ใน database	30
รูปที่ 24 ส่วนหนึ่งของ code ใน userdisplay.cs	31
รูปที่ 25 ส่วนหนึ่งของ code ใน post_page.dart	32
รูปที่ 26 ส่วนหนึ่งของ code ใน editprofilepage.dart	32
รูปที่ 27 ส่วนหนึ่งของ code หน้า othersprofilepage	33



รูป	หน้า
รูปที่ 28 ส่วนหนึ่งของ code หน้า userprofilepage	33
รูปที่ 29 ส่วนหนึ่งของ code หน้า home.dart	34
รูปที่ 30 ส่วนหนึ่งของ code หน้า login.dart	34
รูปที่ 31 ส่วนหนึ่งของ code หน้า register.dart	35
รูปที่ 32 ส่วนหนึ่งของ code หน้า postapprove.dart	36
รูปที่ 33 ส่วนหนึ่งของ code หน้า postreport.dart	37
รูปที่ 34 ส่วนหนึ่งของ code หน้า userreport.dart	38
รูปที่ 35 ส่วนหนึ่งของ code หน้า userverified.dart	38
รูปที่ 36 code ใน circle_button.dart	39
รูปที่ 37 ส่วนหนึ่งของ code ใน create_post_contrainer.dart	39
รูปที่ 38 code ในการสร้างปุ่ม icbutton	40
รูปที่ 39 ส่วนหนึ่งของ code ใน profile_container.dart	40
รูปที่ 40 ส่วนหนึ่งของ code ใน navigation_drawer	41
รูปที่ 41 ส่วนหนึ่งของ code ใน post_contrainer	41
รูปที่ 42 code ใน tag_button	42
รูปที่ 43 ส่วนหนึ่งของ code ใน emailfield	42
รูปที่ 44 ส่วนของ code ใน passwordfield	43
รูปที่ 45 ส่วนหนึ่งของ code ใน widget bottom_banner_ad	43
รูปที่ 46 ส่วนหนึ่งของ code ใน widget tags_field	44
รูปที่ 47 ตัวอย่าง code DataContext.cs	45
รูปที่ 48 code ดู post ที่โดน report	46
รูปที่ 49 code ดึง data ของ ads	46
รูปที่ 50 code ในการ register	47
รูปที่ 51 code ในการเพิ่ม tags ลงไปในโพส	47
รูปที่ 52 code Advertiser.cs	48
รูปที่ 53 code Post.cs	48
รูปที่ 54 code Posts_Users.cs	48
รูปที่ 55 code Tags_Follow.cs	49
รูปที่ 56 code Tags_Posts.cs	49
รูปที่ 57 code User.cs	49

รูป	หน้า
รูปที่ 58 code User_Follows.cs	50
รูปที่ 59 code Users_SharedPosts.cs	50
รูปที่ 60 ส่วนหนึ่งของ code ใน Program.cs	50

## บทที่ 1

### บทนำ

#### 1.1 ที่มาของโครงการ

โครงการ Application KMITL News มีแนวคิดมาจากปัญหาในชีวิตประจำวันที่ทางผู้พัฒนาได้พบเจอในปัจจุบันหากมีความต้องการติดตามข่าวสารจากทางสถาบันได้อย่างครบถ้วน นักศึกษามีความจำเป็นที่ต้องติดตามเพจเฟซบุ๊กมากกว่า 10 เพจเพื่อตามข่าวได้ครบซึ่งในแต่ละเพจประกอบด้วยโพสหลากหลายเรื่องต่างกันไปและ content ในเพจนั้นประกอบทั้งที่นักศึกษาอยากทราบและไม่อยากทราบ อีกทั้งในบางครั้งข่าวสารที่มาจากอาจารย์ของแต่ละคณะนั้นไม่ได้มีการโพสลงไปในเพจแต่ถูกโพสลงไปในเว็บไซต์ของทางสถาบัน ถ้าหากทางนักศึกษาจำเป็นต้องเปิดเข้าไปดูด้วยตัวเองซึ่งสร้างความยุ่งยากและอาจทกหล่นข่าวสารที่จำเป็นได้

จากเหตุผลข้างต้นด้วยความต้องการที่จะเพิ่มความสะดวกสบายในการใช้งานและแก้ไขปัญหาให้นักศึกษาได้รับข่าวสารอย่างครบถ้วนโดยพัฒนา application KMITL News ที่จะรวบรวมข่าวสารจากทั้งเพจเฟซบุ๊กของสถาบัน อาจารย์ และในเว็บไซต์ของทางสถาบันหรือภาควิชาอื่นๆ มารวมกันในที่เดียวและยังทำให้ผู้ใช้งานสามารถกรองข่าวที่ตัวเองสนใจได้และสามารถค้นหาและติดตามได้ผ่าน tags โดยข่าวสารนั้นถูกโพสได้ทั้งแอดมินและผู้ใช้งานแต่ผู้ใช้งานจะต้องผ่านการคัดกรองเพื่อให้ข่าวภายในแอปมีความถูกต้องมากที่สุด

#### 1.2 จุดประสงค์ของโครงการ

พัฒนาแอปพลิเคชันที่สามารถแก้ปัญหาการที่นักศึกษาจำเป็นต้องติดตามหลายช่องทางเพื่อที่สามารถติดตามข่าวสารได้อย่างครบถ้วน โดยแอปพลิเคชันที่พัฒนาขึ้นจะให้นักศึกษาสามารถติดตามหัวข้อที่ตัวเองสนใจหรือติดตามผู้ใช้งานอาจารย์ที่ต้องการได้ อีกทั้งยังมีการจัดหมวดหมู่ติด tags เพื่อที่สามารถให้ผู้ใช้งานตามได้โดยง่าย และมีการคัดกรองข่าวสารภาพในแอปพลิเคชันให้มีความถูกต้องอยู่ตลอดเสมอ และยังสามารถให้ third party เข้ามาติดต่อเพื่อขอให้แสดงโฆษณาภายในแอปพลิเคชัน

## บทที่ 2

### ภาพรวมและการออกแบบระบบ

#### 2.1 ภาพรวมขั้นตอนการทำงานของระบบ

##### 2.1.1 ฝั่งผู้ใช้งาน

###### 2.1.1.1 Login/Register

เป็นหน้าแรกเมื่อผู้ใช้งานเปิดเข้าแอปพลิเคชันจะแบ่งออกเป็น 2 แบบคือผู้ใช้เก่าและผู้ใช้ใหม่ หากผู้ใช้งานเป็นผู้ใช้งานเก่าจะสามารถทำการ login แต่หากเป็นผู้ใช้งานใหม่จะต้องทำการ register เพื่อสร้างบัญชีผู้ใช้งานก่อนซึ่งการ register จำเป็นต้องใส่ข้อมูลประกอบด้วย ชื่อ-สกุล เบอร์โทรศัพท์ อีเมล และ password เพื่อใช้ในการ login ต่อไป

###### 2.1.1.2 Display feed

เป็นหน้าหลักที่จะมีการแสดงโพสต์ทุกอย่างทั้งที่ติดตามและ random ขึ้นมาซึ่งในหน้าหลักนี้ผู้ใช้งานสามารถโพสต์ได้เช่นกันโดยจะมีกล่องให้สร้างโพสต์อยู่ทางด้านบนของหน้า feed และด้านบนใน AppBar จะมีปุ่มให้เปิดหน้าต่าง slide เพื่อดูเมนูในการเปลี่ยนหน้าไปยังหน้าที่มีให้เลือก เพิ่มความสะดวกในการใช้งานแก่ผู้ใช้งาน

###### 2.1.1.3 Filter

หน้า filter จะเกิดจากปุ่มเปิด menu filter ขึ้นมาเพื่อให้สามารถเปลี่ยนหน้าไปยังหน้าต่างๆ โดยสามารถเลือกได้เป็น Home page, For you page, Tags ซึ่งแบ่งเป็น Tags search และ Following tags, Profile page, Request Verified page สำหรับเพื่อยืนยันตัวตนทำให้สามารถโพสต์ข่าวได้โดยไม่ต้องรอการ verify สิ่งที่ต้องการโพสต์จากแอดมินและด้านล่างสุดจะมีปุ่มให้ logout ออกจากระบบ

#### 2.1.1.4 Tags display page

เป็นหน้าที่จะเข้ามาดูโพสต์ด้วย tag ที่ผู้ใช้งานค้นหาเมื่อ search tag ที่ต้องการหน้า Tags display จะแสดงทุกโพสต์ที่เกี่ยวกับ tag นั้นๆ ออกมาทั้งหมดไม่ว่าจะเป็นผู้ใช้งานท่านใดโพสต์ก็ตามและยังสามารถกดติดตาม tag นั้นๆ ได้อีกด้วย

#### 2.1.1.5 Profile page

##### 2.1.1.5.1 Profile ตัวเอง

เป็นหน้าโปรไฟล์ตัวเองที่แสดงรูปภาพตัวเองและชื่อของผู้ใช้งานตัวเองที่สามารถเข้าไปเพื่อแก้ไขโปรไฟล์ตัวเองอีกทั้งยังมีการ display สิ่งที่ผู้ใช้งานได้ทำการโพสต์ไปทั้งหมด ซึ่งตรงโพล์นั้นๆ สามารถที่จะกดลบโพสต์ได้

##### 2.1.1.5.1.1 แก้ไขโปรไฟล์

สามารถแก้ไขชื่อและรูปที่จะ display ออกไป

##### 2.1.1.5.2 Profile คนอื่น

เป็นหน้าที่ผู้ใช้งานเข้าไปดูหน้า profile ผู้ใช้งานคนอื่นโดยจะแสดงรูปและชื่อของผู้ใช้งานคนนั้นๆ และมีปุ่มให้ติดตามผู้ใช้งานคนนั้นเพื่อที่จะได้ไม่พลาดสิ่งที่ผู้ใช้งานคนนั้นโพสต์ ซึ่งจำเป็นมากๆ ผู้ใช้งานที่ต้องการติดตามเป็นอาจารย์ เช่น ต้องการติดตามทุกโพสต์ที่อาจารย์ปริญญาเป็นคนโพสต์ เป็นต้น อีกทั้งยังมีปุ่ม report เพื่อใช้ในกรณีที่ผู้ใช้งานนั้นๆ มีการ display ชื่อไม่เหมาะสมหรือผู้ใช้งานที่ได้รับการ verify โพสต์สิ่งที่ไม่เป็นความจริงมากเกินไปหรือเกิดการ spam อีกทั้งยังมีการแสดงโพสต์ที่ผู้ใช้งานคนนั้นเคยโพสต์ไปและยังสามารถ report โพสต์ที่ผู้ใช้งานคนนั้นได้ทำการโพสต์ไปได้

## 2.1.2 ฝั่งแอดมิน

### 2.1.2.1 แอดมิน Display page

หน้า display feed จะเหมือนกันกับฝั่ง user แต่มีการเปลี่ยนจากปุ่ม report เปลี่ยนเป็นปุ่มลบโพสต์เพื่อให้แอดมินสามารถลบโพสต์ที่เห็นบนหน้า feed ได้ทันทีไม่ต้องรอผู้ใช้งานคนอื่นๆ มา report

#### 2.1.2.1.1 Admin Filter

หน้า Admin Filter จะแตกต่างจากของผู้ใช้งานทั่วไปคือแทนที่ request verified เปลี่ยนเป็น admin panel ซึ่งประกอบด้วย post นั่นคือ การที่แอดมินจะทำการ verify โพสต์ต่างๆ ที่ผู้ใช้งานทำการโพสต์ลงหน้าฟีดคล้ายกับกลุ่มในเฟซบุ๊ก report คือการดูว่ามีการ report โพสต์อะไรบ้างแล้วทำการลบหรือปิดตกรายการนั้นไป verified คือการยืนยันตัวตนผู้ใช้งานที่ทำการขอ verified account ของตัวเองมา และด้านล่างยังมีปุ่ม logout เพื่อออกจากระบบ

### 2.1.2.2 แอดมิน Post approve

เป็นหน้ารวมโพสต์ที่ต้องการขอ approve เพื่อให้โพสต์นั้นถูกโพสต์ลงใน feed ได้อย่างเสร็จสมบูรณ์ ซึ่งแอดมินจะเห็นโพสต์ที่รอการ approve และมีปุ่มให้แอดมินกดว่าผ่านหรือไม่ผ่าน

### 2.1.2.3 แอดมิน Report

เป็นหน้าที่แอดมินเห็นโพสต์หรือโปรไฟล์ที่ถูก report ส่งมาจากผู้ใช้งานคนอื่นๆ และจะมีปุ่มให้กดว่าจะลบโพสต์นี้หรือไม่หากไม่ก็ถูกปล่อยผ่านไป

### 2.1.2.4 แอดมิน Verified

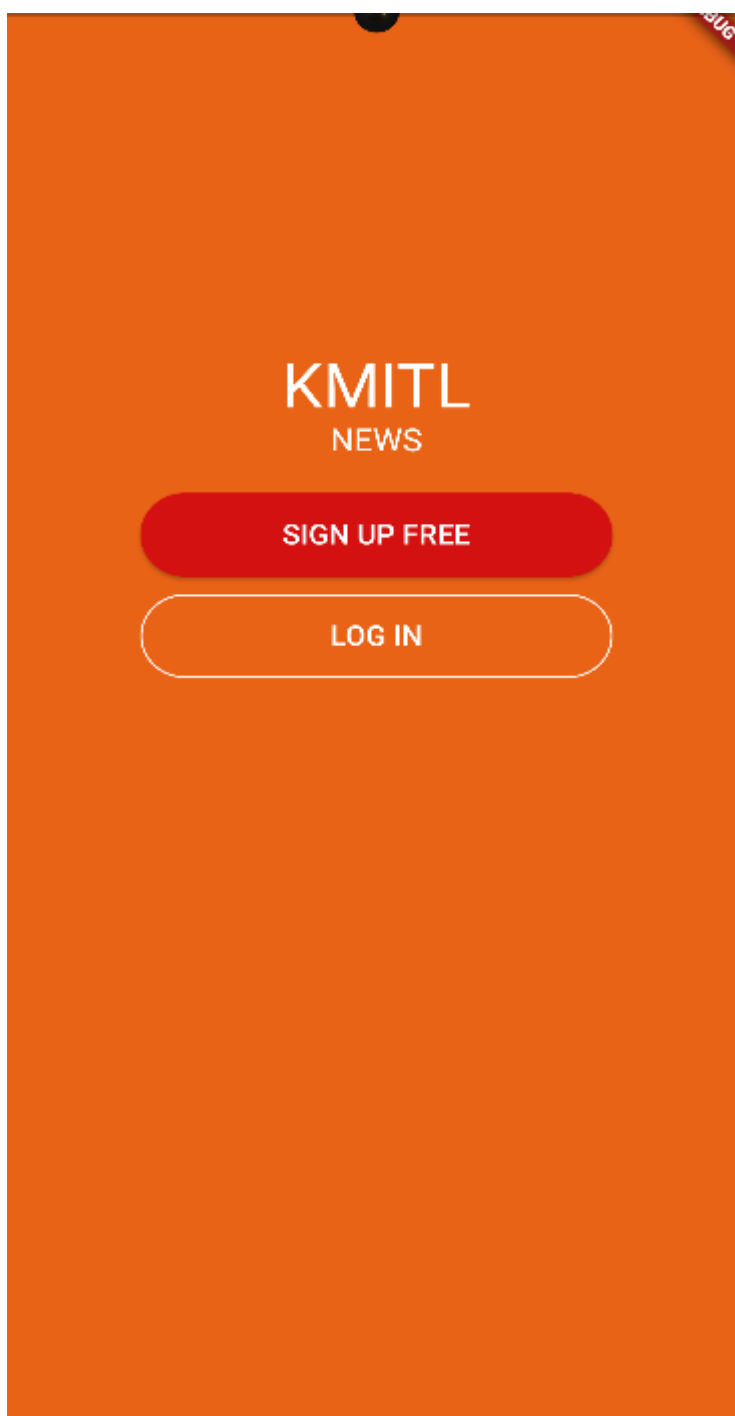
เป็นหน้าที่จะเห็นข้อมูลของผู้ใช้งานคนอื่นๆ และทำการกดปุ่มให้ว่าผ่านการ verified หรือไม่ โดยข้อมูลที่ส่งมาเพื่อ verified จะประกอบด้วยชื่อ-สกุล อีเมล และเบอร์โทร

## 2.2 การออกแบบหน้า

ออกแบบ UX/UI บน figma เพื่อนำเป็นแบบในการทำงานจริง

### 2.2.1 ออกแบบหน้าผู้ใช้งาน

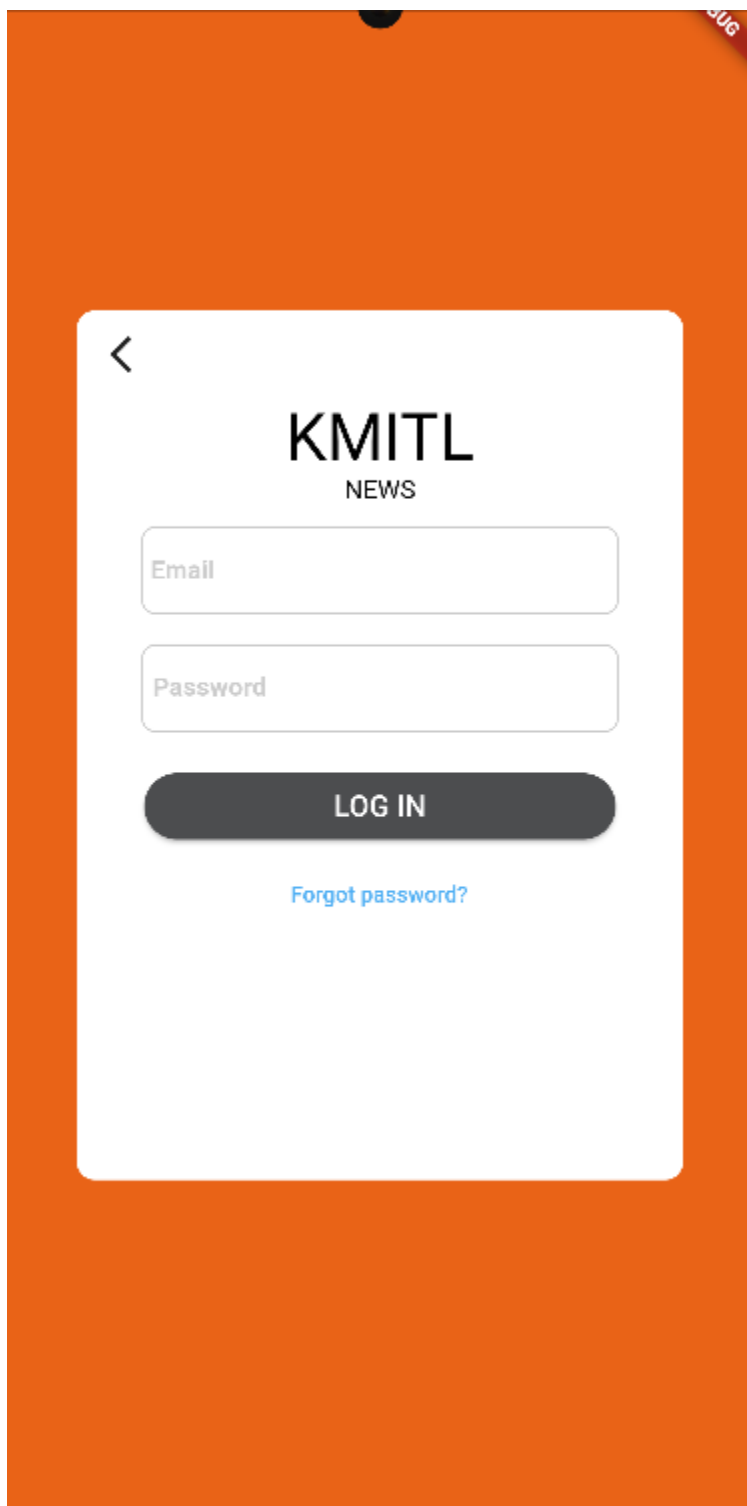
#### 2.2.1.1 ออกแบบหน้า Login/Register



รูปที่ 1 ออกแบบหน้า home

### 2.2.1.2 ออกแบบหน้า Login

ให้ผู้ใช้กรอกอีเมลและ password ก่อน login

The image shows a mobile application login screen for KMITL NEWS. The background is a solid orange color. In the top right corner, there is a small red banner with the word "BUG" in white. The login form is a white rounded rectangle centered on the screen. It features a back arrow icon in the top left corner. Below the arrow, the text "KMITL" is displayed in a large, bold, black font, with "NEWS" in a smaller, regular black font directly underneath. There are two input fields: the first is labeled "Email" and the second is labeled "Password", both in a light gray font. Below these fields is a dark gray rounded button with the text "LOG IN" in white. At the bottom of the form, there is a blue text link that says "Forgot password?".

รูปที่ 2 ออกแบบหน้า login



### 2.2.1.3 ออกแบบหน้า Register

ให้ผู้ใช้กรอก ชื่อจริง นามสกุลจริง อีเมล เบอร์โทรศัพท์ password และ ยืนยัน password อีกครั้งเพื่อให้ชัวร์ว่าผู้ใช้กรอกรหัสถูกต้อง

The image shows a mobile application registration screen. At the top, there is a black header bar with a white back arrow on the left and the word 'Register' in white text. A small red banner with the number '06' is visible in the top right corner of the header. Below the header, the registration form consists of six rounded rectangular input fields stacked vertically. The first four fields are labeled 'First Name', 'Last Name', 'Email', and 'Mobile Number'. The fifth field is labeled 'Password' and has a small icon of an eye with a slash through it on the right side, indicating a toggle for password visibility. The sixth field is labeled 'Confirm Password' and also has the same eye icon on the right. Below these input fields is a dark gray rounded rectangular button with the word 'Continue' in white text. At the bottom of the form, there is a blue text link that reads 'Term & Privacy Policy'.

รูปที่ 3 ออกแบบหน้า register

### 2.2.1.4 ออกแบบหน้า Display feed

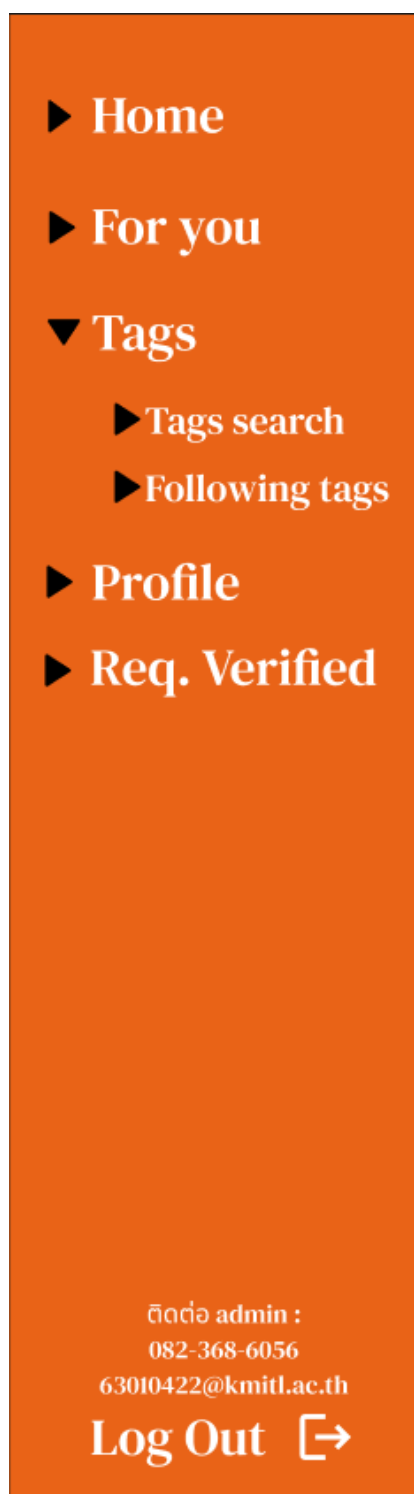
เป็นหน้าที่แสดงโพสต์ทั้งที่ผู้ใช้งานกดติดตามและ random ขึ้นมา



รูปที่ 4 ออกแบบหน้า Display feed

### 2.2.1.5 ออกแบบหน้า Filter

หน้าที่สามารถกดแล้วสามารถเปลี่ยนหน้าไปยังหน้าต่างๆ ภายในแอปพลิเคชัน



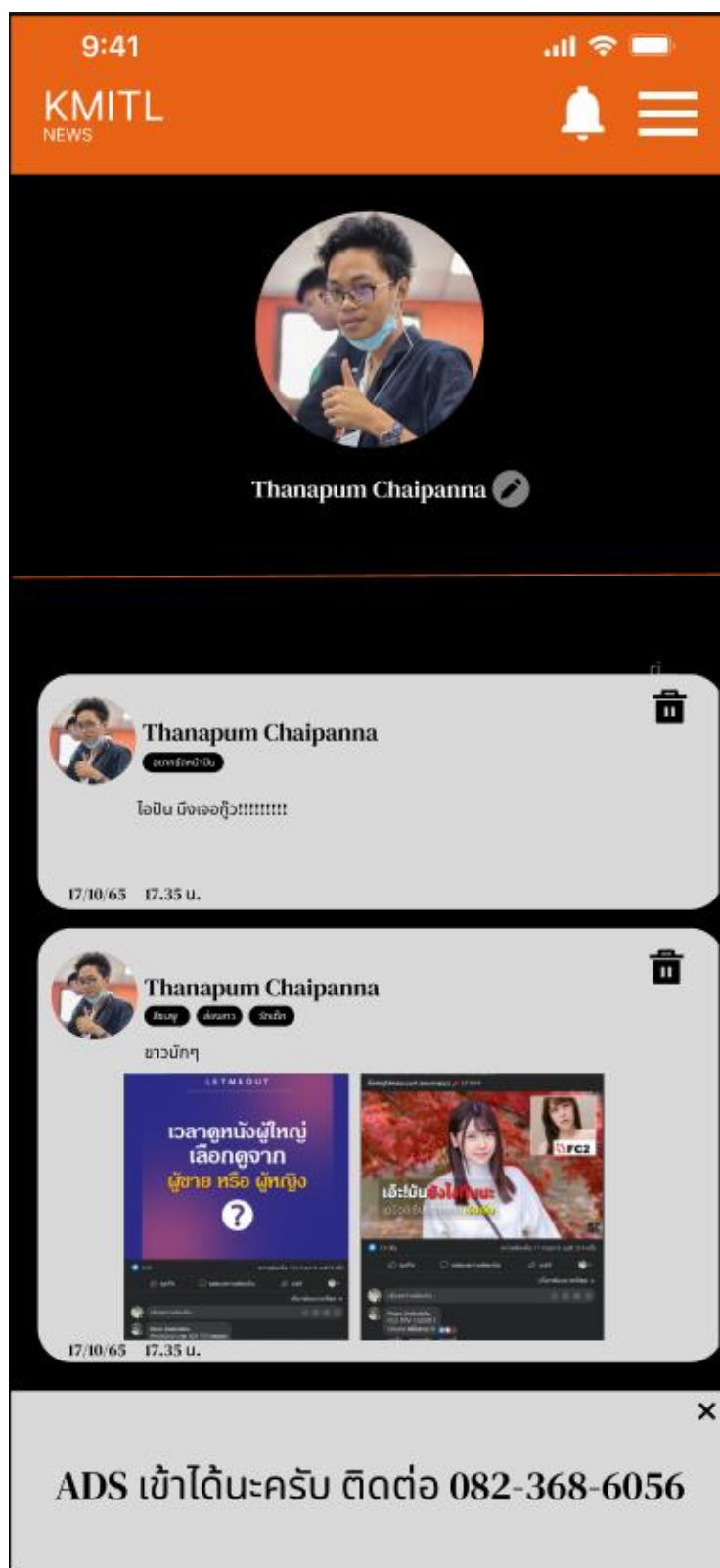
รูปที่ 5 ออกแบบหน้า Filter

### 2.2.1.6 หน้า Profile

แสดงหน้า profile ของผู้ใช้งานเองและผู้ใช้งานท่านอื่น



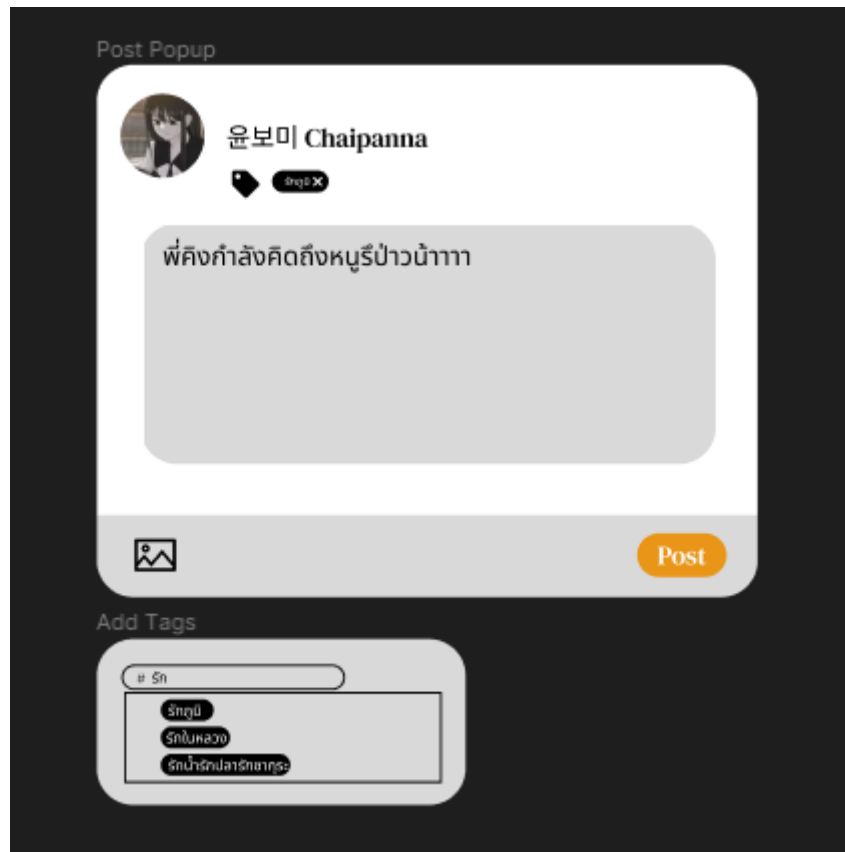
รูปที่ 6 หน้าโปรไฟล์ผู้ใช้งานท่านอื่น



รูปที่ 7 ออกแบบหน้าโปรไฟล์ตัวเอง

### 2.2.1.7 ออกแบบส่วนเสริม

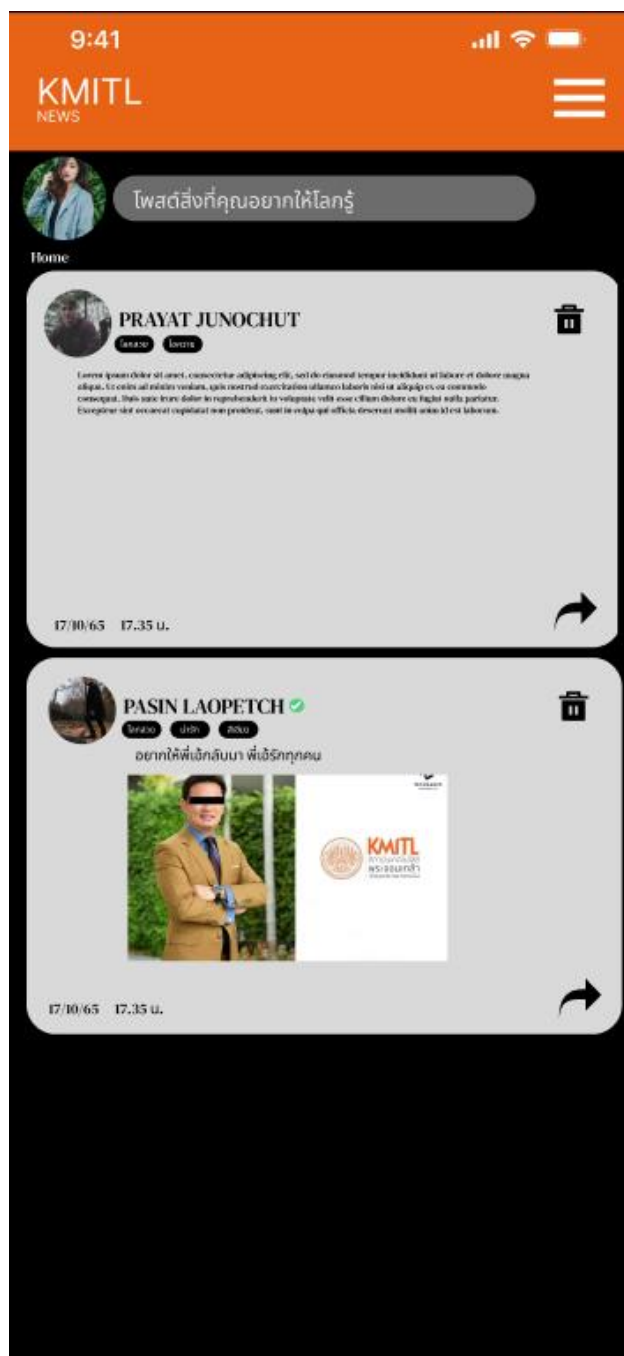
เป็นส่วนที่เกิดขึ้นเมื่อกดการโพสต์ ใส่ tags



รูปที่ 8 ออกแบบส่วนเสริม

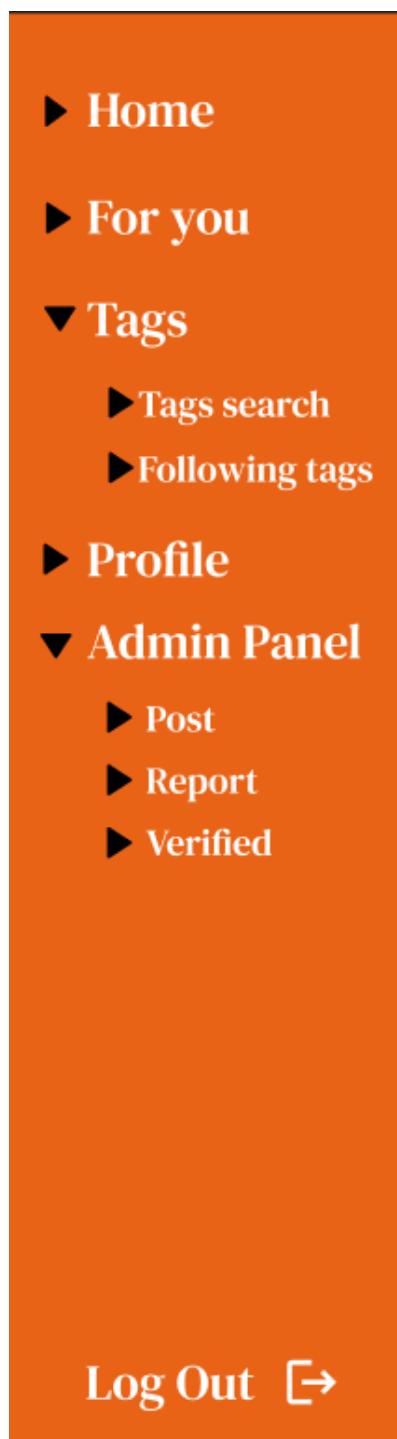
## 2.2.2 ออกแบบหน้าแอดมิน

### 2.2.2.1 ออกแบบ Display feed admin



รูปที่ 9 ออกแบบหน้า Display ของแอดมิน

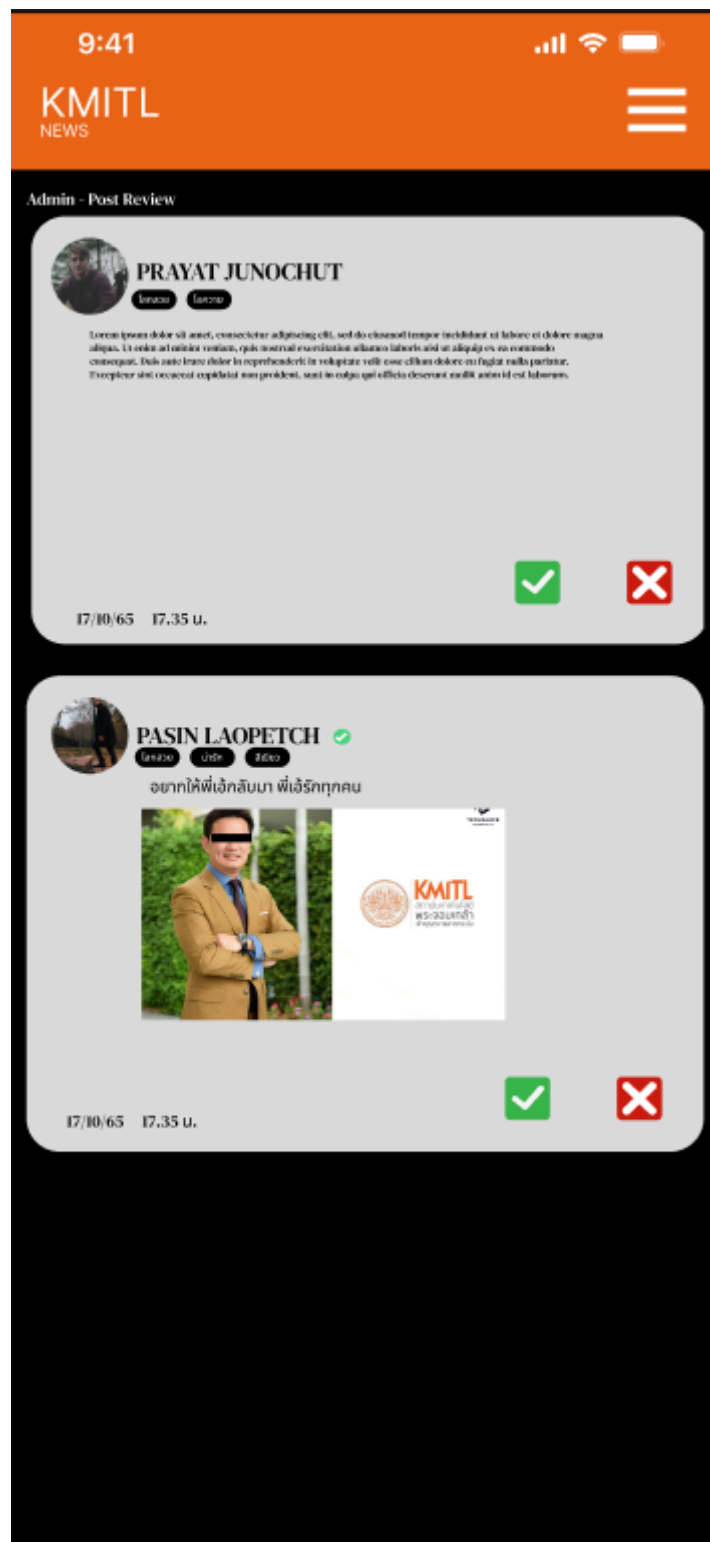
#### 2.2.2.2 ออกแบบหน้า Filter ของแอดมิน



รูปที่ 10 ออกแบบหน้า Filter ของแอดมิน

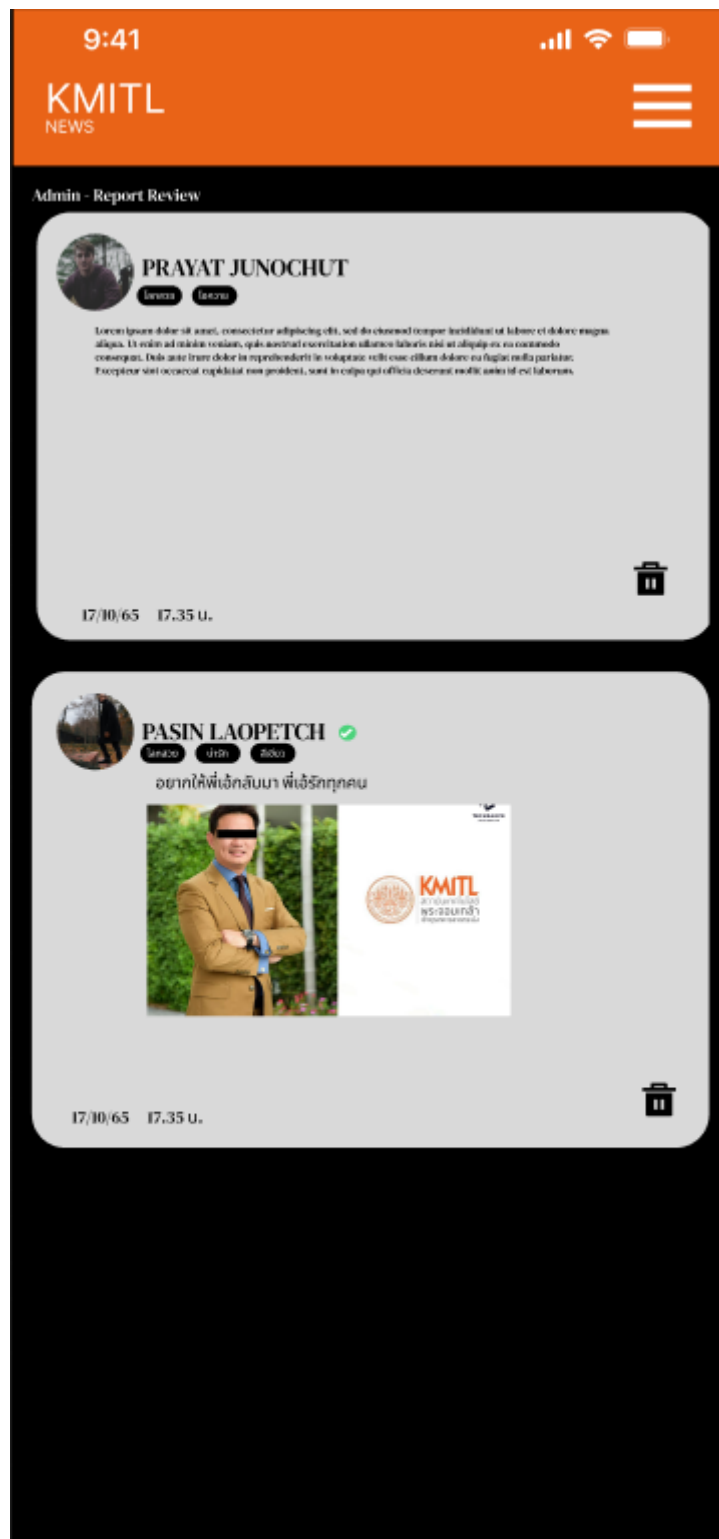


### 2.2.2.3 ออกแบบหน้า Post approve ของแอดมิน



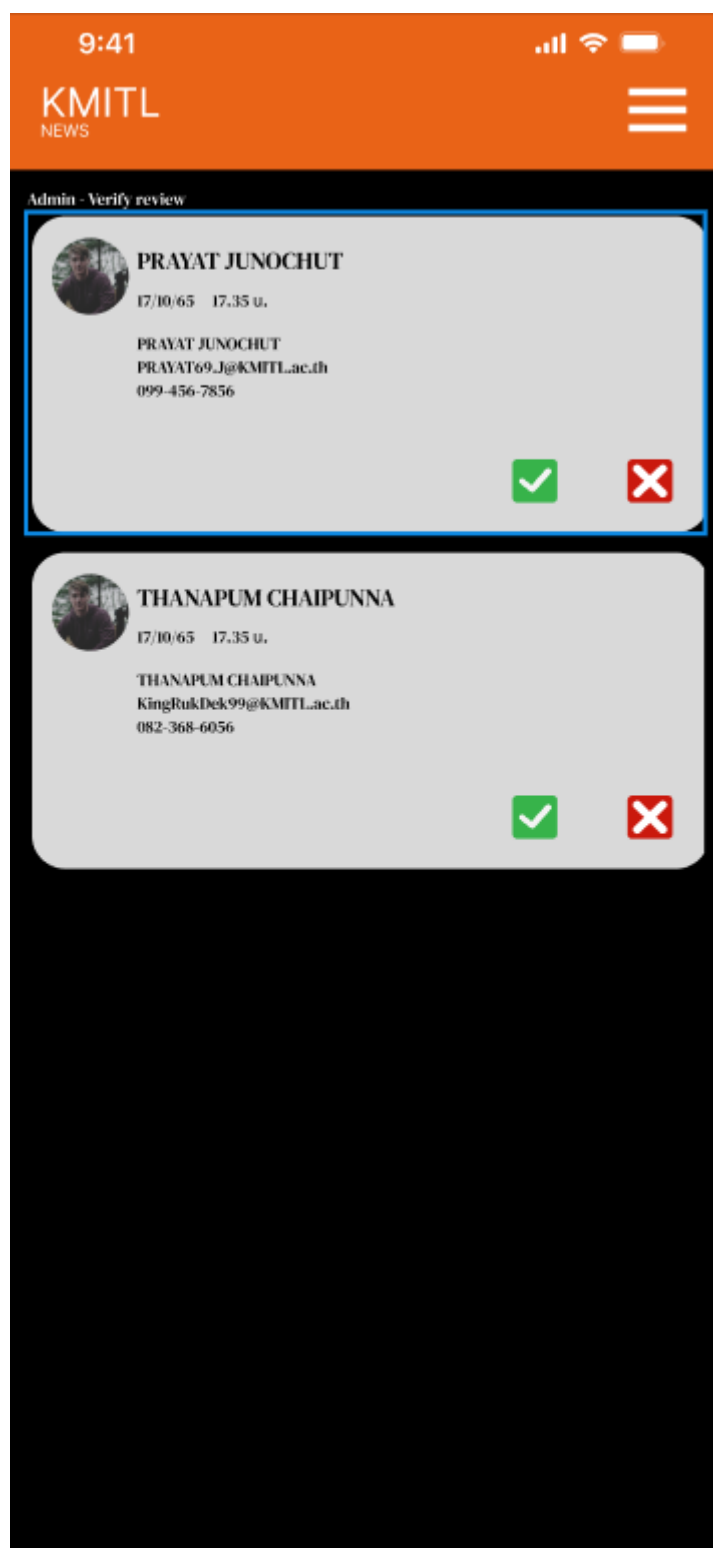
รูปที่ 11 ออกแบบหน้า Post approve ของแอดมิน

### 2.2.2.4 ออกแบบหน้า report ของแอดมิน



รูปที่ 12 ออกแบบหน้า report ของแอดมิน

## 2.2.2.5 ออกแบบหน้า verified account ของแอดมิน



รูปที่ 13 ออกแบบหน้า verified account ของแอดมิน

## บทที่ 3

### การประยุกต์บทเรียน

#### 3.1 Event Storming

[Event Storming, Online Whiteboard for Visual Collaboration \(miro.com\)](https://miro.com)

Event Storming แบ่งเป็น 3 steps คือ

##### 3.1.1 Collect Domain Events

เป็นการนำ Event หลัต่างๆ มาวางเอาไว้บน board ด้วย post-it สีส้ม

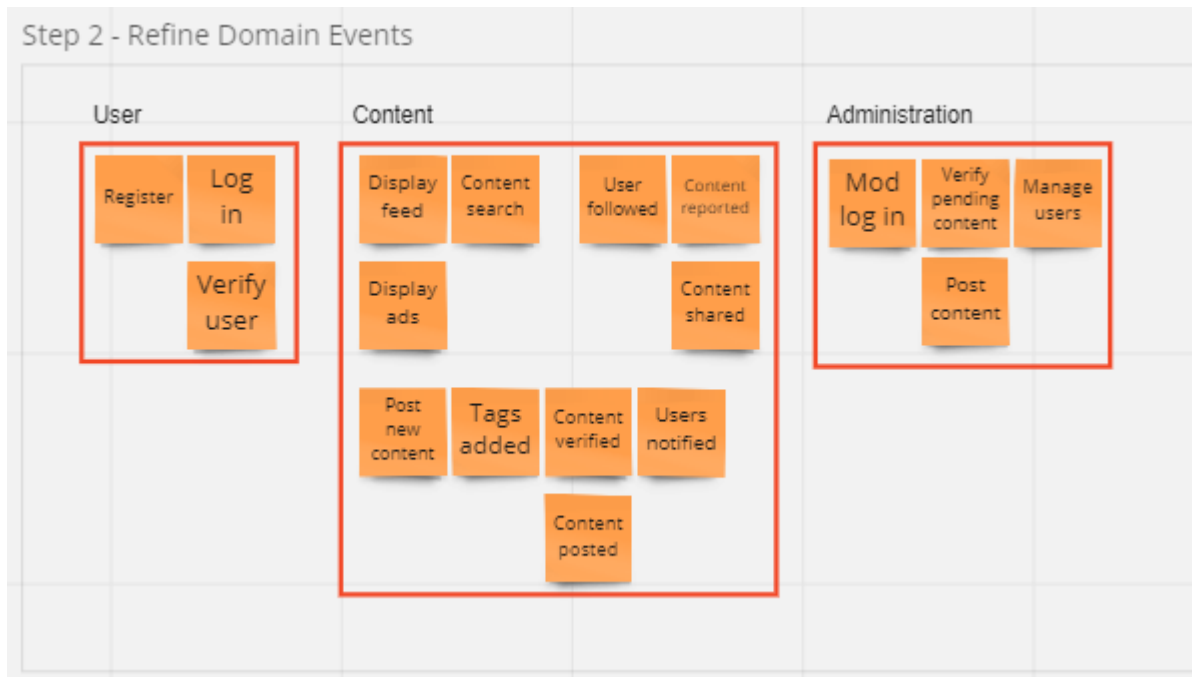


รูปที่ 14 Collect Domain Events

ในรูปจะประกอบด้วย Domain หลักที่เป็นการทำงานหลักของแอปพลิเคชันโดยแบ่งเป็น 2 สายคือ user และ admin โดยฝั่ง user ประกอบไปด้วย Register, Login, Verify user, Display feed, Display ads, Post new content, content verified, Content posted, Users notified, Content interacted with และฝั่งadmin ประกอบด้วย Admin login, Verify pending content, Post content, manage users.

### 3.1.2 Refine Domain Events

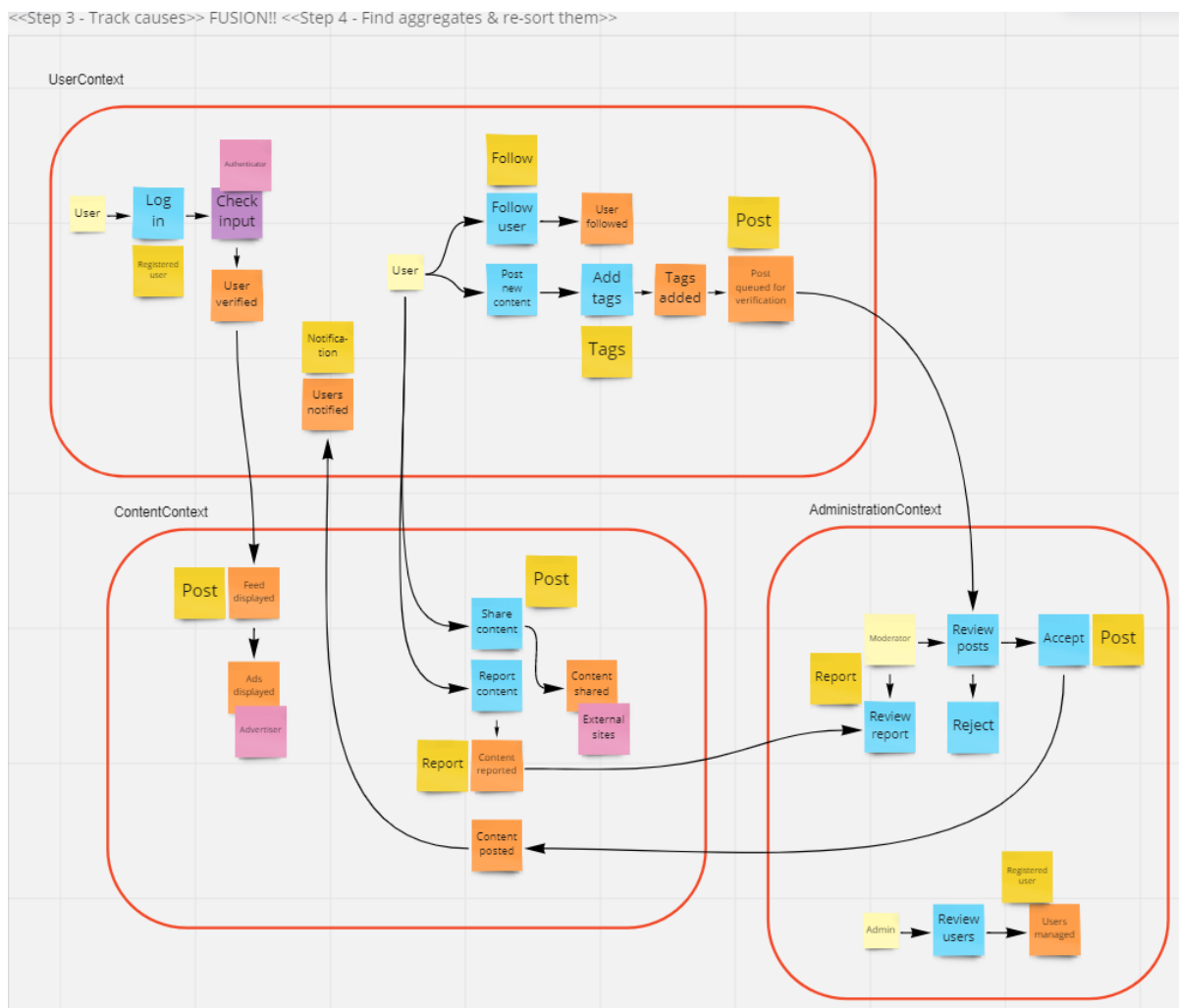
เป็นการแบ่งกรอบ domain ที่มีความเกี่ยวข้องกันเข้าไปอยู่ด้วยกัน ในแอปพลิเคชัน KMITL News แบ่งออกเป็น 3 กรอบนั้นคือ User, Content และ Administration



รูปที่ 15 Refine Domain Events

### 3.1.3 Track cause รวมกับการใส่ aggregates และ จัดการโยงลำดับ

เป็นส่วนในการใส่การทำงานการโยงข้อมูลการทำงานเข้าด้วยกันเป็นลำดับเหตุการณ์ โดยแบ่งสี่เป็น สี่เหลืองอ่อนเป็น user หรือ actor สีครามคือสิ่งที่ user ต้องการทำให้ (command) สีเหลืองเข้มคือ aggregate สีม่วงคือ logic และสีส้มกำหนดต่างๆ สีชมพูคือ service จากภายนอก และมีลูกศรชี้โยงการทำงานได้ดัง รูปที่() และทำการแบ่งออกเป็น bounded context ซึ่งได้เป็น 3 bounded คือ UserContext, ContentContext และ AdministrationContext

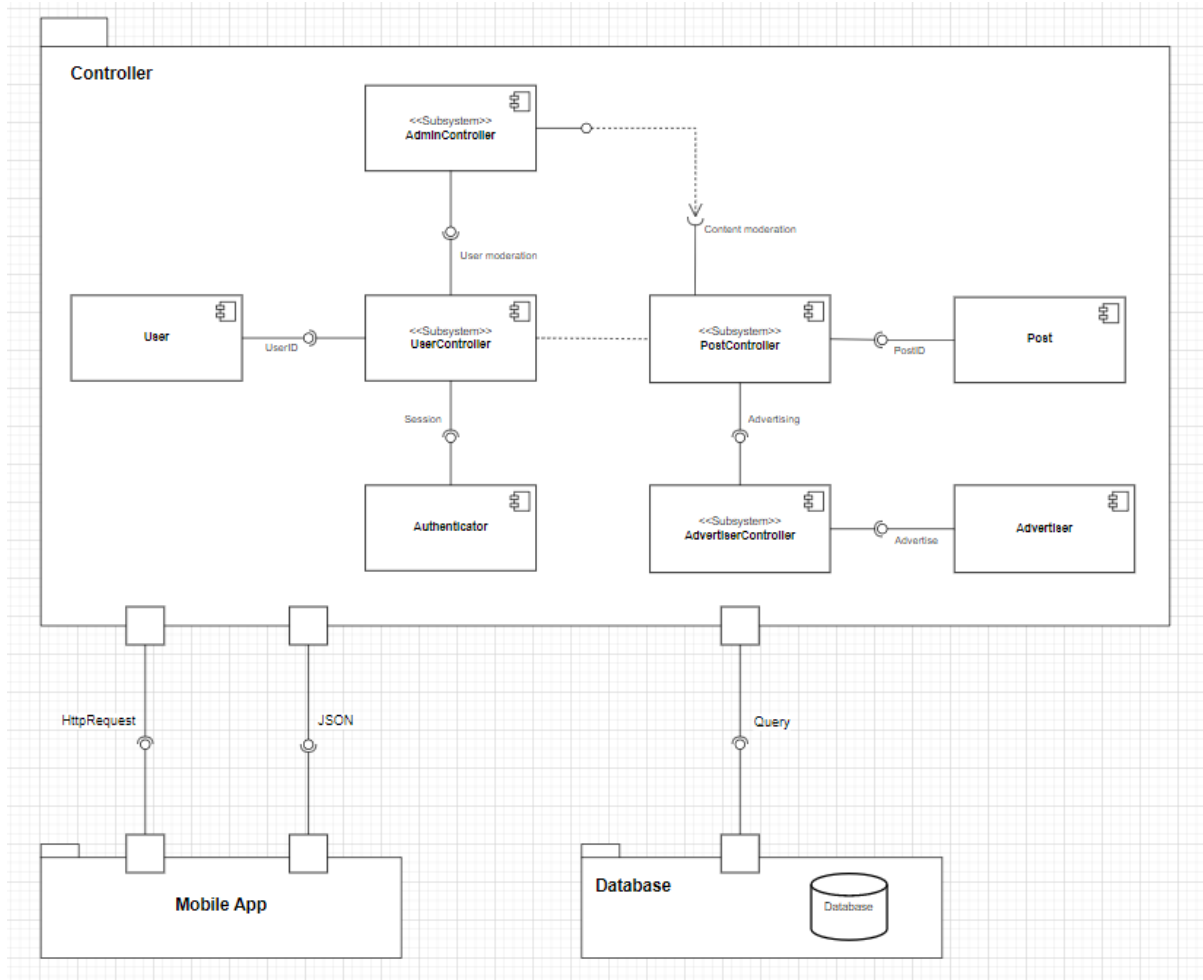


รูปที่ 16 Track cause

### 3.2 UML

[SAD Diagram.drawio - diagrams.net](http://SAD-Diagram.drawio-diagrams.net)

#### 3.2.1 UML Component

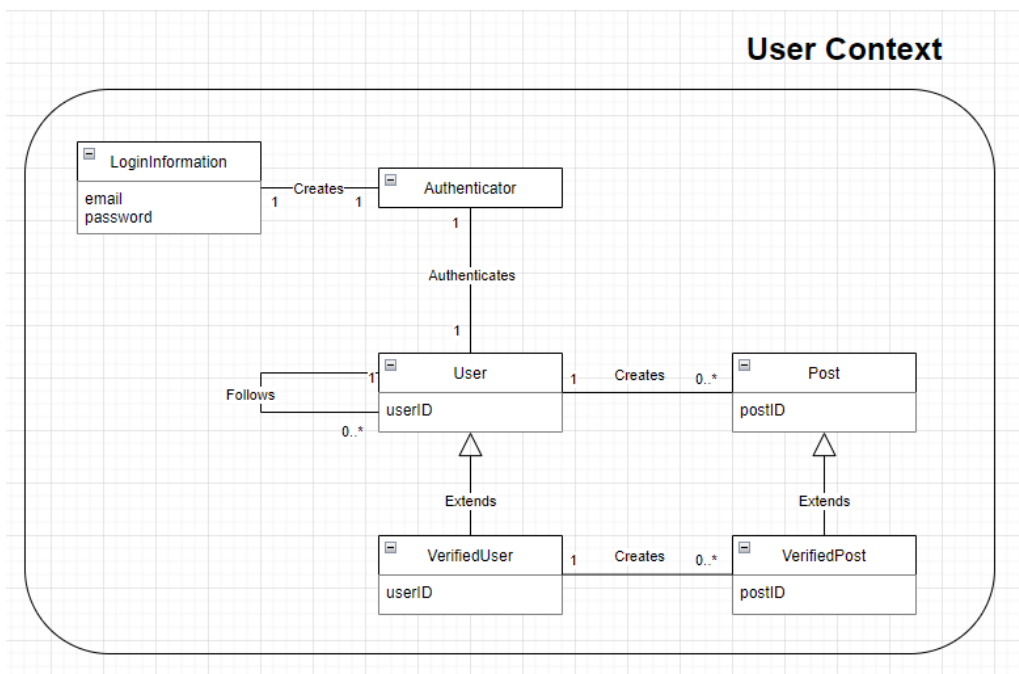


รูปที่ 17 component diagram

### 3.2.2 domain model

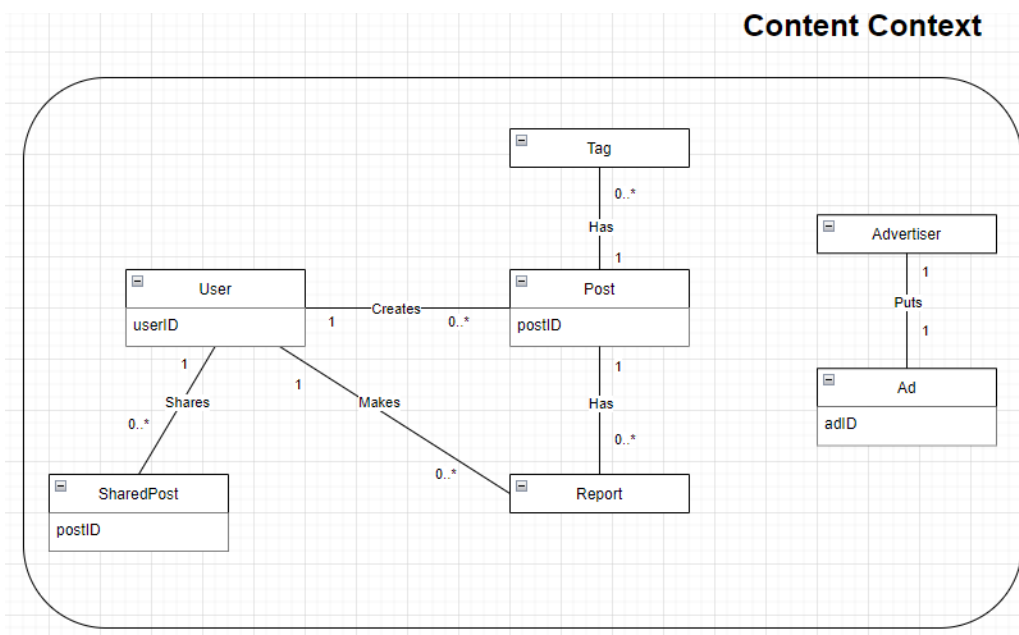
แบ่งออกเป็น 3 bounded context

#### 3.2.2.1 UserContext



รูปที่ 18 UserContext

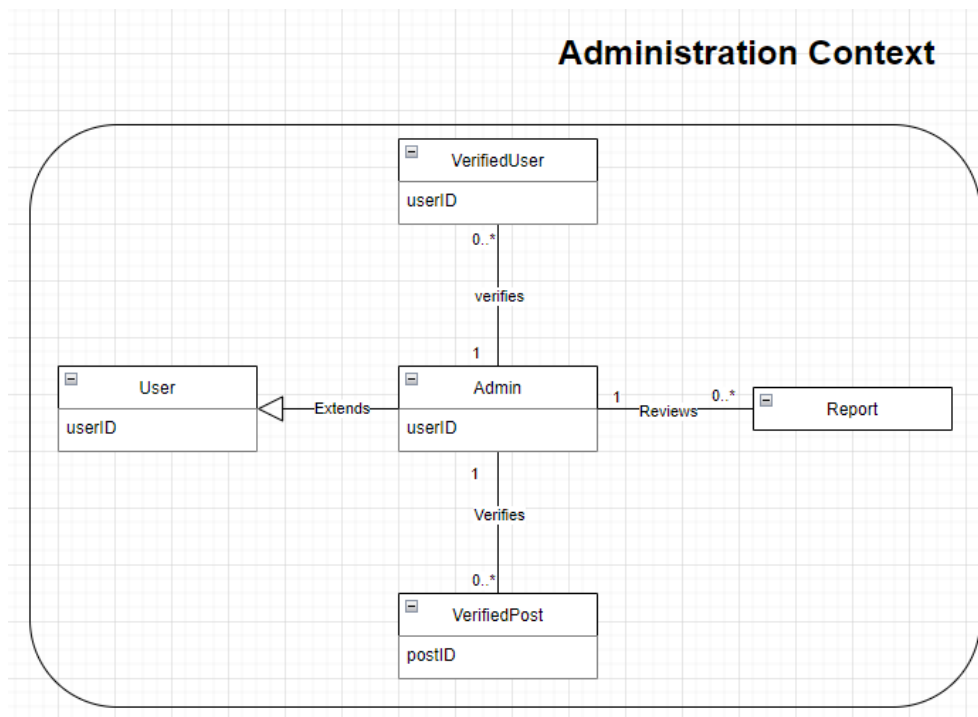
#### 3.2.2.2 ContentContext



รูปที่ 19 ContentContext



## 3.2.2.3 AdministrationContext



รูปที่ 20 AdministrationContext

### 3.3 Quality Attribute

#### 3.3.1 Availability

โปรแกรมสามารถใช้งานได้ตามจุดประสงค์

##### 3.3.1.1 สามารถ Display feed

หน้าแอปพลิเคชันสามารถแสดงผล feed ข่าวตามที่คุณพัฒนาต้องการตามจุดประสงค์

Source of stimulus	Users, Developers
Stimulus	Crash, incorrect response
Artifacts	Application
Environment	Normal Operation, Startup
Response	Display error window
Response measure	99.99% Availability

##### 3.3.1.2 ความสามารถในการเชื่อมต่อ Database

ในการเชื่อมต่อกับ database สามารถเชื่อมต่อได้ตลอดเวลาเนื่องจาก database ที่เลือกใช้เป็นแบบ Local ซึ่งทำให้สามารถเปิดและปิดตามความต้องการได้อีกด้วย

Source of stimulus	Users, Developers
Stimulus	Connecting to Backend
Artifacts	Application
Environment	Normal Operation, Startup
Response	Able to connect and communicate with Backend
Response measure	24-hour connection

### 3.3.1.3 แสดง Tags

เมื่อทำการค้นหา tags และสามารถแสดงโพสต์ทุกโพสต์ที่มี tags นั้นอยู่ออกมา

Source of stimulus	Users, Developers
Stimulus	Tags search
Artifacts	Application
Environment	Normal Operation
Response	Display only posts containing the searched tag
Response measure	Response time < 1 second

### 3.3.1.4 จัดการ content

ทำให้แอดมินสามารถคัดกรองเนื้อหาภายในแอปพลิเคชันให้มีความถูกต้อง

Source of stimulus	Users, Developers
Stimulus	Content Management
Artifacts	Application
Environment	Normal Operation
Response	Admin is able to review and remove content
Response measure	Response time < 1 second

## 3.3.2 Integrability

### 3.3.2.1 Pub.dev

สามารถเพิ่มเครื่องมือใช้งานจากภายนอกเข้ามาใช้ร่วมกับ code ที่มีอยู่ได้ เนื่องจาก  
การเลือกใช้ภาษา dart ทำให้มีความ flexible ในการเพิ่ม plugin เข้ามา

Source of stimulus	Developers
Stimulus	Add new plugin
Artifacts	Entire code
Environment	Development
Response	Plugin is successfully added and ready to use
Response measure	New plugin

### 3.3.3 Modifiability

#### 3.3.3.1 การแก้ไข code ในส่วน Backend

สามารถแก้ไขได้ง่ายเนื่องจากทางผู้พัฒนาได้เลือกใช้ Architectural pattern แบบ MVC

Source of stimulus	Developers
Stimulus	Add/ Delete/ Modify functionality
Artifacts	Code
Environment	Build complete time
Response	Build complete
Response measure	Build time

#### 3.3.3.2 การแก้ไข code ในส่วน frontend

สามารถแก้ไขได้ง่ายเนื่องจาก code ส่วนมากมีการแยกออกมาเป็น Widget ทำให้สามารถหาและแก้ไขเพื่อส่งผลกับทุกจุด ทำให้ไม่เสียเวลาในการแก้ไข

Source of stimulus	Developers
Stimulus	Add/ Delete/ Modify functionality and widget
Artifacts	Code
Environment	Build complete time
Response	Build complete, Make/Deploy modification
Response measure	Widget is able to be used in other classes

### 3.3.4 Performance

#### 3.3.4.1 การส่ง request ข้อมูล

ในการเชื่อมต่อและส่ง request ไปยัง database เพื่อขอข้อมูลจำนวนมาก ประกอบด้วยโพส โปรไฟล์ โฆษณา เป็นต้นโดยใช้เวลาในการ response ไม่มาก

Source of stimulus	Users, Developer
Stimulus	User request
Artifacts	Application
Environment	Normal Operation
Response	Request response
Response measure	Latency < 1 seconds

### 3.4 Software Architectural Style

#### 3.4.1 MVC

ใช้ในส่วนของ backend ใช้จัดการกับส่วนของ code component เพื่อให้กลายเป็นโครงสร้างให้เรียกใช้งานได้ง่าย สะอาด ไม่มีการเรียกที่ทับซ้อนกันไม่ให้มีการเรียกเป็นทอดๆ เช่น ไม่จำเป็นต้องเรียก A แล้วไปหา B แต่สามารถเรียก B ได้โดยตรง

#### 3.4.2 Plug-in

ใช้ในส่วน Frontend ร่วมกับ flutter ใช้ในการตกแต่งเพิ่มความสวยงามของแอปพลิเคชัน ซึ่งทางผู้พัฒนาเลือกใช้ Font Awesome ซึ่งเป็นเว็บที่รวม icon ต่างๆ ในเลือกและเป็นเว็บที่ creator dev designer ส่วนมากเลือกใช้

```
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
```

รูปที่ 21 import font awesome

#### 3.4.3 Representational State Transfer (REST)

ในส่วน frontend มีการใช้ Restful API เพื่อติดต่อกับ Backend โดยการยิง HTTP Methods เช่น GET.POST GET.DELETE เป็นต้น

## 3.5 Design pattern

### 3.5.1 Façade

ใช้ในส่วนของ DataContext ซึ่งเอาไว้เชื่อมกับทาง database

```
public class DataContext : DbContext {
    public DataContext(DbContextOptions<DataContext> options) : base(options) {
    }

    //Entities

    public DbSet<User> Users { get; set; }
    public DbSet<Post> Posts { get; set; }
    public DbSet<Advertiser> Advertisers { get; set; }

    //Relations

    public DbSet<Posts_Users> Posts_Users { get; set; }
    public DbSet<Users_SharedPosts> Users_SharedPosts { get; set; }
    public DbSet<Users_Follows> Users_Follows { get; set; }
    public DbSet<Tags_Posts> Tags_Posts { get; set; }
    public DbSet<Tags_Follows> Tags_Follows { get; set; }

    //Extra methods

    public static IQueryable<T> RemoveFrom<T>(DbSet<T> dest, Expression<Func<T, bool>> predicate) where T : class {
        if (dest == null)
            throw new ArgumentNullException(nameof(dest));
        if (predicate == null)
            throw new ArgumentNullException(nameof(predicate));

        var arr = dest.Where(predicate);
        foreach (var i in arr)
            dest.Remove(i);

        return arr;
    }
}
```

รูปที่ 22 Façade ใน DataContext

### 3.5.2 Iterator

ใช้เพื่อจัดการกับ listdata เพื่อให้ข้อมูลเป็นลำดับจาก database

```
10 references
public static IQueryable<T> RemoveFrom<T>(DbSet<T> dest, Expression<Func<T, bool>> predicate) where T : class {
    if (dest == null)
        throw new ArgumentNullException(nameof(dest));
    if (predicate == null)
        throw new ArgumentNullException(nameof(predicate));

    var arr = dest.Where(predicate);
    foreach (var i in arr)
        dest.Remove(i);

    return arr;
}
```

รูปที่ 23 Iterator ใน database



## บทที่ 4

### การพัฒนา

#### 4.1 Frontend

##### 4.1.1 ภาษาที่ใช้พัฒนา Frontend

ในการพัฒนาแอปพลิเคชัน KMITL News ผู้พัฒนาเลือกใช้ Flutter ในการพัฒนา mobile application นี้ ซึ่ง flutter เป็นเครื่องมือในการพัฒนา mobile application ที่สามารถทำงานข้าม platform ได้ในทั้ง IOS และ Android ใน flutter นั้นจะเป็นภาษา **Dart** ซึ่งถูกพัฒนาโดย google ออกมาเป็น open source ให้ใช้ฟรี

ภาษา dart ที่ใช้ใน Flutter ซึ่งจะมีความคล้ายกับภาษา Java เนื่องจาก dart เป็นภาษาที่รองรับ OOP และมีแนวคิดของ class และ inheritance เช่นเดียวกับภาษา Java

##### 4.1.2 Main Page

###### 4.1.2.1 userdisplay.dart

เป็นหน้าที่แสดง feed หลักของผู้ใช้งาน โดยจะแสดงโพสต์ที่มีใน database

```
key: _scaffoldKey,
endDrawer: NavigateDrawer(),
body: CustomScrollView(
  slivers: [
    SliverAppBar(
      backgroundColor: Color.fromARGB(255, 222, 105, 21),
      title: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          Text(
            'KMITL',
            style: TextStyle(
              fontSize: 24,
              color: Colors.white,
              fontWeight: FontWeight.bold),
          ),
          Row(
            children: <Widget>[
              Text(
                'NEWS',
                style: TextStyle(
                  fontSize: 12,
                  color: Colors.white,
```

รูปที่ 24 ส่วนหนึ่งของ code ใน userdisplay.cs

#### 4.1.2.2 post\_page.dart

เป็นส่วนหนึ่งของหน้าหลักผู้ใช้งานแต่ทำหน้าที่ตอนที่ผู้ใช้งานทำการสร้างโพสต์

```
SliverToBoxAdapter(
  child: Column(
    children: [
      Container(
        height: MediaQuery.of(context).size.height * 0.2,
        width: MediaQuery.of(context).size.height * 1,
        decoration: BoxDecoration(
          color: Color.fromARGB(255, 206, 204, 204),
        ),
      ),
      child: Row(
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.all(20),
            child: GestureDetector(
              onTap: () => print('หน้าผู้ใช้งาน'),
              child: Container(
                margin: EdgeInsets.fromLTRB(
                  MediaQuery.of(context).size.height * 0.01,
                  MediaQuery.of(context).size.height * 0.06,
                  0,
                  0),
                width: avatarDiameter,
                height: avatarDiameter,
                decoration: BoxDecoration(
                  color: Colors.blue,
                  shape: BoxShape.circle,
                ),
              ),
            ),
          ],
        ),
      ),
    ],
  ),
)
```

รูปที่ 25 ส่วนหนึ่งของ code ใน post\_page.dart

#### 4.1.2.3 editprofilepage.dart

เป็นหน้าที่ทำหน้าที่ให้ผู้ใช้งานแก้ไขโปรไฟล์ตัวเองไม่ว่าจะเป็นแก้ไขรูปหรือชื่อที่ display ให้ผู้ใช้งานคนอื่นเห็น

```
TextField(
  style:
    TextStyle(color: Colors.white, fontWeight: FontWeight.bold),
  decoration: InputDecoration(
    contentPadding: EdgeInsets.only(bottom: 3),
    enabledBorder: UnderlineInputBorder(
      borderSide: BorderSide(color: Colors.white),
    ),
    labelText: "Display Name",
    labelStyle: TextStyle(color: Colors.white),
    floatingLabelBehavior: FloatingLabelBehavior.always,
    hintText: "Ex: Under",
    hintStyle: TextStyle(
      fontSize: 16,
      fontWeight: FontWeight.bold,
      color: Colors.white,
    ),
  ),
)
```

รูปที่ 26 ส่วนหนึ่งของ code ใน editprofilepage.dart

#### 4.1.2.4 othersprofilepage.dart

เป็นหน้าโปรไฟล์ของผู้ใช้งานคนอื่นเมื่อเข้าไปในหน้าโปรไฟล์คนอื่นสามารถทำการกด follow ผู้ใช้งานอื่นได้ และสามารถเห็นโพสต์ที่ผู้ใช้งานคนนั้นได้โพสต์ไปอีกทั้งสามารถ report โปรไฟล์และโพสต์ได้

```
GestureDetector(
  onTap: () {
    if (follow)
      setState(() => follow = false);
    else
      setState(() => follow = true);
    print("Follow");
  },
  child: AnimatedContainer(
    alignment: Alignment.center,
    duration: Duration(milliseconds: 300),
    height: 35,
    width: 200,
    decoration: BoxDecoration(
      color: follow?Colors.orange : Colors.transparent,
```

รูปที่ 27 ส่วนหนึ่งของ code หน้า othersprofilepage

#### 4.1.2.5 userprofilepage.dart

เป็นหน้าโปรไฟล์ของผู้ใช้งานเอง ในหน้านี้ผู้ใช้งานสามารถกดแก้ไขโปรไฟล์ตัวเองได้ โดยจะเข้าไปที่หน้า editprofilepage และสามารถเห็นโพสต์ที่ตัวเองได้โพสต์ไปอีกทั้งสามารถลบโพสต์ที่โพสต์ไปได้

```
Expanded(
  child: Container(
    color: Colors.black,
    child: ListView.separated(
      itemCount: posts.length,
      itemBuilder: (BuildContext context, int index) {
        final post = posts[index];
        if (index == 0) return Container();
        return PostContainer(post: post, type: 'owner');
      },
      separatorBuilder: (context, index) => SizedBox(
        height: 10,
      ),
    ),
  ),
```

รูปที่ 28 ส่วนหนึ่งของ code หน้า userprofilepage

#### 4.1.2.6 home.dart

เป็นหน้าแรกที่ผู้ใช้งานเข้าแอปพลิเคชัน โดยให้ผู้ใช้งานเลือกระหว่าง Login หรือ Register

```
ElevatedButton(
  style: ElevatedButton.styleFrom(
    shape:
      RoundedRectangleBorder(borderRadius: BorderRadius.circular(30)),
    fixedSize: const Size(256, 48),
    primary: const Color.fromRGB0(212, 17, 17, 1), // background
    onPrimary: Colors.white, // foreground
  ),
  onPressed: () {
    Navigator.push(context, MaterialPageRoute(builder: (context) {
      return RegisterScreen();
    }));
  },
  child: const Text(
    'SIGN UP FREE',
    style: TextStyle(fontSize: 16),
```

รูปที่ 29 ส่วนหนึ่งของ code หน้า home.dart

#### 4.1.2.7 login.dart

สืบเนื่องจากหน้า home หากผู้ใช้งานเป็นผู้ใช้งานเก่าหรือผู้ใช้งานที่มีบัญชีอยู่แล้ว หน้า login เป็นหน้าที่ให้ผู้ใช้งานกรอกอีเมลและ password ให้ถูกต้องแล้วจึงสามารถเข้าใช้งานแอปพลิเคชันได้

```
(onSavedVal) => {
  this.password = onSavedVal.toString().trim(),
},
onChange: (val) {},
initialValue: "",
obscureText: false,
borderFocusColor: Colors.black,
prefixIconColor: Theme.of(context).primaryColor,
borderColor: Colors.black.withOpacity(0.2),
borderRadius: 8,
borderWidth: 1,
focusedBorderWidth: 1,
hintColor: Colors.black.withOpacity(0.2),
fontSize: 14,
hintFontSize: 14,
paddingLeft: 35,
paddingRight: 35,
```

รูปที่ 30 ส่วนหนึ่งของ code หน้า login.dart



#### 4.1.2.9 postapprove.dart

ในหน้านี้เป็นส่วนประกอบของฝั่งแอดมินหากผู้ใช้งานที่ทำการ login เข้ามาไม่ได้ เป็นบัญชีแอดมิน ผู้ใช้งานจะไม่สามารถเห็นและเข้ามาในหน้านี้ได้ ในหน้า postapprove เป็นหน้าที่จะแสดงโพสที่รอการยืนยันให้สามารถโพสได้จากแอดมิน

```
key: _scaffoldKey,
endDrawer: AdminNavigateDrawer(),
body: Container(
  color: Colors.black,
  child: ListView.separated(
    itemCount: posts.length,
    itemBuilder: (BuildContext context, int index) {
      final post = posts[index];
      if (index == 0)
        return Container(
          padding: EdgeInsets.fromLTRB(20, 12, 0, 0),
          height: 24,
          child: Text(
            'Admin - Post Review',
            textAlign: TextAlign.left,
            style: TextStyle(
              fontSize: 12,
              color: Colors.white,
              fontWeight: FontWeight.bold,
            ),
          ),
        );
      return PostContainer(post: post, type: 'approve');
    },
  ),
);
```

รูปที่ 32 ส่วนหนึ่งของ code หน้า postapprove.dart

#### 4.1.2.10 postreport.dart

ในหน้านี้เป็นส่วนประกอบของฝั่งแอดมินหากผู้ใช้งานที่ทำการ login เข้ามาไม่ได้เป็นบัญชีแอดมิน ผู้ใช้งานจะไม่สามารถเห็นและเข้ามาในหน้านี้ได้ ในหน้า postreport จะรวมโพสต์ที่ถูก report เอาไว้ให้แอดมินได้ทำการตรวจสอบอีกครั้งว่าสมควรถูกลบออกหรือไม่

```
key: _scaffoldKey,
endDrawer: AdminNavigateDrawer(),
body: Container(
  color: Colors.black,
  child: ListView.separated(
    itemCount: posts.length,
    itemBuilder: (BuildContext context, int index) {
      final post = posts[index];
      if (index == 0)
        return Container(
          padding: EdgeInsets.fromLTRB(20, 12, 0, 0),
          height: 24,
          child: Text(
            'Admin - Post Report Review',
            textAlign: TextAlign.left,
            style: TextStyle(
              fontSize: 12,
              color: Colors.white,
              fontWeight: FontWeight.bold,
            ),
          ),
        ),
      );
      return PostContainer(post: post, type: 'report');
    },
  ),
);
```

รูปที่ 33 ส่วนหนึ่งของ code หน้า postreport.dart

#### 4.1.2.11 userreport.dart

ในหน้านี้เป็นส่วนประกอบของฝั่งแอดมินหากผู้ใช้งานที่ทำการ login เข้ามาไม่ได้เป็นบัญชีแอดมิน ผู้ใช้งานจะไม่สามารถเห็นและเข้ามาในหน้านี้ได้ ในหน้า userreport จะแสดงผู้ใช้งานที่โดน report ให้แอดมินได้ทำการตรวจสอบหากเป็นผู้ใช้งานที่ประพฤติตนไม่เหมาะสมหรือเป็นผู้ใช้งานปลอมแอดมินสามารถทำการลบบัญชีผู้ใช้งานได้

```
if (index == 0)
  return Container(
    padding: EdgeInsets.fromLTRB(20, 12, 0, 0),
    height: 24,
    child: Text(
      'Admin - User Report Review',
      textAlign: TextAlign.left,
      style: TextStyle(
        fontSize: 12,
        color: Colors.white,
        fontWeight: FontWeight.bold,
      ),
    ),
  );
return ProfileContainer(profile: profile, type: 'report');
```

รูปที่ 34 ส่วนหนึ่งของ code หน้า userreport.dart

#### 4.1.2.12 serververified.dart

ในหน้านี้เป็นส่วนประกอบของฝั่งแอดมินหากผู้ใช้งานที่ทำการ login เข้ามาไม่ได้เป็นบัญชีแอดมิน ผู้ใช้งานจะไม่สามารถเห็นและเข้ามาในหน้านี้ได้ ในหน้า serververified เป็นหน้าที่แอดมินจะทำการตรวจสอบว่าผู้ใช้งานที่ขอการ verified มาเป็นผู้ใช้งานคนนั้นจริงๆ สามารถพิสูจน์ได้

```
if (index == 0)
  return Container(
    padding: EdgeInsets.fromLTRB(20, 12, 0, 0),
    height: 24,
    child: Text(
      'Admin - Verify Review',
      textAlign: TextAlign.left,
      style: TextStyle(
        fontSize: 12,
        color: Colors.white,
        fontWeight: FontWeight.bold,
      ),
    ),
  );
return ProfileContainer(profile: profile, type: 'verify');
```

รูปที่ 35 ส่วนหนึ่งของ code หน้า serververified.dart



### 4.1.3 Widget

#### 4.1.3.1 circle\_button.dart

ใช้ช่วยในการสร้างปุ่มเป็นวงกลม

```

9      const CircleButton(
10        {super.key,
11         required this.icon,
12         required this.iconSize,
13         required this.onPressed});
14
15      @override
16      Widget build(BuildContext context) {
17        return Container(
18          margin: const EdgeInsets.all(6.0),
19          decoration: BoxDecoration(
20            color: Color.fromARGB(255, 226, 141, 95),
21            shape: BoxShape.circle,
22          ),
23          child: IconButton(
24            icon: icon,
25            iconSize: iconSize,
26            color: Colors.black,
27            onPressed: onPressed,

```

รูปที่ 36 code ใน circle\_button.dart

#### 4.1.3.2 create\_post\_contrainer.dart

ใช้ในการสร้างกล่องสำหรับโพสต์

```

Expanded(
  child: ElevatedButton(
    onPressed: () => Navigator.push(context, MaterialPageRoute(builder: (context){return PostPage();})),
    child: Align(
      alignment: Alignment.bottomLeft,
      child: Text("อยากบอกอะไรไหม KMIL รู้?",
        textAlign: TextAlign.left,
      ),
    ),
    style: ElevatedButton.styleFrom(
      primary: Color.fromARGB(255, 118, 117, 117),
      padding: EdgeInsets.all(4),
      textStyle: TextStyle(
        fontSize: 18,
      ),
      shape: StadiumBorder(),
    ),
  ),
),
),
),
),

```

รูปที่ 37 ส่วนหนึ่งของ code ใน create\_post\_contrainer.dart

#### 4.1.3.3 icbutton.dart

ใช้สร้างปุ่มเพื่อเอาไปใช้ในหน้าอื่นๆ

```
return Container(
  margin: const EdgeInsets.all(6.0),
  child: IconButton(
    icon: icon,
    iconSize: iconSize,
    color: Colors.black,
    onPressed: onPressed,
```

รูปที่ 38 code ในการสร้างปุ่ม icbutton

#### 4.1.3.4 profile\_container.dart

สร้างพื้นที่ในการใส่ข้อความเอาไปใช้ในส่วนของการ verified ผู้ใช้งานและดูการถูก report ซึ่งอยู่ในส่วนของแอดมิน

```
if (widget.type == 'report')
  Row(
    mainAxisAlignment: MainAxisAlignment.end,
    children: [
      Text(
        'Report count : ' +
          widget.profile['report_count'].toString(),
        style: TextStyle(
          color: Colors.black,
          fontWeight: FontWeight.bold,
        ),
      ),
      IconButton(
        icon: FaIcon(FontAwesomeIcons.trashCan),
        iconSize: 23.0,
        onPressed: () => print("share")),
    ],
```

รูปที่ 39 ส่วนหนึ่งของ code ใน profile\_container.dart

#### 4.1.3.5 navigation\_drawer.dart

Widget เสริมในการสร้าง slide menu ในการเข้าไปยังหน้าต่างๆ ประกอบด้วย Home page, For you page, Tags, Profile page, Request Verified, Logout เพื่อเอาไปใช้ในหลายๆ หน้าในแอปพลิเคชัน ไม่ว่าจะเป็นหน้า home หน้า profile ตัวเองหรือคนอื่น

```
ListTile(
  leading: Icon(
    Icons.arrow_right,
    size: 30,
  ),
  title: const Text(
    'Profile',
    style: TextStyle(color: Colors.white, fontSize: 24),
  ),
  onTap: () {
    Navigator.push(context,
      MaterialPageRoute(builder: (context) => UserProfilePage()));
  },
),
```

รูปที่ 40 ส่วนหนึ่งของ code ใน navigation\_drawer

#### 4.1.3.6 post\_contrainer.dart

ใช้ส่วนที่เอาไว้สร้างโพสเพื่อเอาไป display ให้หน้า display feed

```
Expanded(
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Row(
        children: [
          Text(
            //ใส่ชื่อแต่ละคนโพส
            widget.post['username'],
            style: TextStyle(
              fontWeight: FontWeight.bold,
              fontSize: 16,
            ),
          ),
          SizedBox(width: 10),
          if (widget.post['user_verify'] != null)
            if (widget.post['user_verify'])
              FaIcon(
                FontAwesomeIcons.userCheck,
                size: 15,
                color: Colors.green,
              )
        ],
      ),
    ],
  ),
```

รูปที่ 41 ส่วนหนึ่งของ code ใน post\_contrainer

#### 4.1.3.7 tag\_button.dart

ใช้ในการใส่เพื่อให้ tags ในโพสสามารถกดได้

```
Widget build(BuildContext context) {
  return Container(
    margin: EdgeInsets.fromLTRB(0, 0, 8, 0),
    child: ElevatedButton(
      onPressed: onPressed,
      child: Text(
        "tag",
        style: TextStyle(
          fontSize: 12,
          color: Colors.white,
        ),
      ),
    ),
    style: ElevatedButton.styleFrom(
      primary: Colors.black,
      elevation: 3,
      shape: StadiumBorder(),
    ),
  );
}
```

รูปที่ 42 code ใน tag\_button

#### 4.1.3.8 emailfield.dart

Widget ในการใส่อีเมลและเช็คที่ว่าที่ใส่ลงไปนั้นถูกหรือไม่

```
contentPadding: EdgeInsets.only(left: 5),
hintText: 'Email',
hintStyle: TextStyle(
  fontSize: 14,
  color: Colors.black.withOpacity(0.2),
  fontWeight: FontWeight.bold,
),
),
keyboardType: TextInputType.emailAddress,
validator: (email) => email != null && !EmailValidator.validate(email)
  ? 'Email invalid.'
  : null,
```

รูปที่ 43 ส่วนหนึ่งของ code ใน emailfield

#### 4.1.3.9 passwordfield.dart

Widget การใส่ password และ ตรวจสอบว่า password ผ่านหรือไม่โดย password ต้องประกอบ 1 ตัวเลข ตัวอักษรใหญ่ 1 ตัว และความยาวอย่างน้อย 8 ตัวอักษร

```
String errortext =
    "Password must have\nMore than 8 character long\nAt least 1 number\nAt least 1 uppercase letter";

onPasswordChanged(String password) {
    final numericRegex = RegExp(r'[0-9]');
    final upperRegex = RegExp(r'[A-Z]');
    errortext = "Password must have\n";
    setState(() {
        _isPasswordEightCharacters = false;
        if (password.length >= 8)
            _isPasswordEightCharacters = true;
        else
            errortext += 'More than 8 character long\n';

        _hasPasswordOneNumber = false;
        if (numericRegex.hasMatch(password))
            _hasPasswordOneNumber = true;
        else
            errortext += 'At least 1 number\n';

        _hasPasswordUpperCase = false;
        if (upperRegex.hasMatch(password))
            _hasPasswordUpperCase = true;
        else
            errortext += 'At least 1 uppercase letter';
    });
}
```

รูปที่ 44 ส่วนของ code ใน passwordfield

#### 4.1.3.10 bottom\_banner\_ad.dart

เป็น Widget ที่ใช้ในการแสดงผลโฆษณาที่ด้านล่างของแอปพลิเคชันที่สามารถกดเข้าเพื่อไปยังหน้าเว็บของโฆษณานั้นๆ ผ่านลิงก์ที่ผูกเอาไว้

```
_launchURL() async {
    const url = 'https://www.facebook.com/profile.php?id=100004449705370';
    if (await canLaunch(url)) {
        await launch(url);
    } else {
        throw 'Could not launch $url';
    }
}
```

รูปที่ 45 ส่วนหนึ่งของ code ใน widget bottom\_banner\_ad

#### 4.1.3.11 tags\_field.dart

มีเป็น widget ที่ใช้ในส่วนของการสร้างโพส ทำหน้าที่คล้ายกับ tags\_button เพียงแต่วิธีใช้และจุดประสงค์เวลาที่ใช้ต่างกัน

```
return Container(  
  margin: EdgeInsets.fromLTRB(0, 0, 8, 0),  
  decoration: BoxDecoration(  
    borderRadius: BorderRadius.circular(10), color: Colors.black),  
  child: Row(  
    children: <Widget>[  
      Padding(  
        padding: const EdgeInsets.only(left: 3),  
        child: Container(  
          alignment: Alignment.centerRight,  
          child: Text(  
            widget.tags,  
            textAlign: TextAlign.end,  
            style: TextStyle(  
              fontSize: 16,  
              color: Colors.white,  
            ),  
          ),  
        ),  
      ],  
    ),  
  ),  
);
```

รูปที่ 46 ส่วนหนึ่งของ code ใน widget tags\_field

## 4.2 Back-end

### 4.2.1 ภาษาที่ใช้ในการพัฒนา backend

ในการพัฒนา database ของแอปพลิเคชัน KMITL News ทางผู้พัฒนาได้เลือกใช้ภาษา C# เป็นภาษาหลักในการพัฒนา ซึ่งเลือกใช้ C# เนื่องจากภาษา C# มีความเอกประสงค์ในการทำงาน ง่ายสำหรับผู้เริ่มต้น มีความคล้ายกับภาษาอื่นซึ่งทำให้ง่ายต่อการทำความเข้าใจ ง่ายต่อการ maintain และทำงานได้อย่างรวดเร็ว

### 4.2.2 Data

#### 4.2.2.1 DataContext.cs

ใช้เชื่อมกับ database เป็นการตั้งชื่อ schema และเป็นการบอก database ให้ทราบถึงโครงสร้าง

```
public class DataContext : DbContext {
    public DataContext(DbContextOptions<DataContext> options) : base(options) {
    }

    //Entities

    public DbSet<User> Users { get; set; }
    public DbSet<Post> Posts { get; set; }
    public DbSet<Advertiser> Advertisers { get; set; }

    //Relations

    public DbSet<Posts_Users> Posts_Users { get; set; }
    public DbSet<Users_SharedPosts> Users_SharedPosts { get; set; }
    public DbSet<Users_Follows> Users_Follows { get; set; }
    public DbSet<Tags_Posts> Tags_Posts { get; set; }
    public DbSet<Tags_Follows> Tags_Follows { get; set; }
```

รูปที่ 47 ตัวอย่าง code DataContext.cs

## 4.2.3 Controller

### 4.2.3.1 AdminController.cs

จัดการ action ที่เกี่ยวกับ admin ทั้งหมด โดยควบคุมการ verify post, verify user  
ดู post ที่ยังไม่ถูก verify ดู user ที่โดน report และการลบ post

```
[HttpGet("GetAllReportedPosts")]
public async Task<ActionResult<IEnumerable<Post>>> GetAllReportedPosts([FromQuery] Request_Admin_Basic request) {
    if (!CheckAuthorization(request.RequesterUserID))
        return Unauthorized("No authorization.");

    var res = await _context.Posts.Where(i => i.report_count > 0).ToListAsync();
    return Ok(res);
}
```

รูปที่ 48 code ดู post ที่โดน report

### 4.2.3.2 AdvertiserController.cs

ใช้ในการดึงโฆษณาเพื่อมาแสดงใน frontend

```
[HttpGet("GetAllAdvertiserData")]
public async Task<ActionResult<IEnumerable<Advertiser>>> GetAllAdvertiserData()
{
    return Ok(await _context.Advertisers.ToListAsync());
}
```

รูปที่ 49 code ดึง data ของ ads

### 4.2.3.3 LoginController.cs

ใช้ในการจัดการในหน้า login และการแก้ไขโปรไฟล์ สามารถทำได้ครอบคลุมทั้งการ  
หา id ว่ามีอยู่แล้วในระบบหรือไม่ในการ login การ check password ว่าตรงกับอีเมลที่ใส่  
ในการ login หรือไม่ ครอบคลุมการ register เพื่อสร้าง account ขึ้นมา การ check อีเมล  
ว่ามีการใช้ซ้ำกันหรือไม่ การ reset password การ reset จำนวนในการ report และนับ  
จำนวนที่โดน report ในฝั่งของ admin



```
[HttpPost("register")]
public async Task<ActionResult> RegisterUser(Request_User_Register request)
{
    // check exist email
    if (_context.Users.Any(u => u.email == request.email))
    {
        return BadRequest("User already exists.");
    }

    // สร้าง hast salt รหัส
    CreatePassHash(request.password, out byte[] pass_hash, out byte[] pass_salt);

    // data User
    var user = new User
    {
        email = request.email,
        pass_hash = pass_hash,
        pass_salt = pass_salt,
        mobile_no = request.mobile_no,
        first_name = request.first_name,
        last_name = request.last_name,
        verificationToken = CreateRandomToken()
    };

    _context.Users.Add(user);
    await _context.SaveChangesAsync(); // รอให้ save การเปลี่ยนแปลงข้อมูลลง DB

    return Ok("User created successfully.");
}
```

รูปที่ 50 code ในการ register

#### 4.2.3.4 PostController.cs

ใช้เกี่ยวกับโพสต์ทั้งหมดไม่ว่าจะเป็นการดึงโพสต์ทั้งหมดใน database และสามารถดูได้ว่า post นั้นๆ โดน report ไปกี่ครั้งและ reset จำนวนที่โดน report อีกทั้งเป็นตัวควบคุมในการสร้างโพสต์พร้อมกับการใส่ tags และยังมี function ในการ share โพสต์นั้นๆ อีกทั้งยังสามารถดูได้ว่ามีโพสต์ไหนบ้างที่ถูก verified ไปแล้วบ้าง สามารถดูได้ว่าผู้ใช้งานคนอื่นๆ ได้ share อะไรไปแล้วบ้างดูโพสต์ทั้งหมดของผู้ใช้งานนั้นๆ สามารถแสดงโพสต์ที่มี tags ตามที่ต้องการได้ สามารถดู followers

```
[HttpPut("AddTagsToPost/{postID}")]
public async Task<ActionResult> AddTagsToPost(int postID, Request_Post_AddTagsToPost request) {
    Post? post = await _context.Posts.FindAsync(postID);
    if (post == null)
        return BadRequest("Post not found.");

    AddTags(postID, request.Tags);

    await _context.SaveChangesAsync();
    return Ok("Success.");
}
```

รูปที่ 51 code ในการเพิ่ม tags ลงไปในโพสต์

## 4.2.4 Model

### 4.2.4.1 Advertiser.cs

เป็นส่วนประกอบของโครงสร้างข้อมูลของ advertiser

```
public class Advertiser {
    [Key]
    public int advertiser_id { get; set; }
    public string advertiser_name { get; set; } = string.Empty;

    public string ad_image_url { get; set; } = string.Empty;
}
```

รูปที่ 52 code Advertiser.cs

### 4.2.4.2 Post.cs

เป็นส่วนประกอบของโครงสร้างข้อมูลของโพสต์

```
public class Post {
    [Key]
    public int post_id { get; set; }
    public DateTime post_date { get; set; }

    public string post_text { get; set; } = string.Empty;
    public string attached_image_url { get; set; } = string.Empty;

    public bool verified { get; set; }
    public int report_count { get; set; }
}
```

รูปที่ 53 code Post.cs

### 4.2.4.3 Posts\_Users.cs

เป็นส่วนประกอบของโครงสร้างข้อมูลของโพสต์ที่ผู้ใช้งานได้ทำการโพสต์

```
[PrimaryKey(nameof(post_id), nameof(user_id))]
public class Posts_Users {
    public int post_id { get; set; }
    public int user_id { get; set; }
}
```

รูปที่ 54 code Posts\_Users.cs

#### 4.2.4.4 Tags\_Follows.cs

เป็นส่วนประกอบของโครงสร้างข้อมูลของ tags ที่สามารถบอกได้ว่ามีผู้ใช้งานใดติดตาม tags บ้าง

```
[PrimaryKey(nameof(tag_name), nameof(follower_id))]
public class Tags_Follows {
    public string tag_name { get; set; } = string.Empty;
    public int follower_id { get; set; }
}
```

รูปที่ 55 code Tags\_Follow.c

#### 4.2.4.5 Tags\_Posts.cs

เป็นส่วนประกอบของโครงสร้างข้อมูลของ tags ที่สามารถบอกได้ว่ามีโพสต์ใดมีการใส่ tags บ้าง

```
[PrimaryKey(nameof(tag_name), nameof(post_id))]
public class Tags_Posts {
    public string tag_name { get; set; } = string.Empty;
    public int post_id { get; set; }
}
```

รูปที่ 56 code Tags\_Posts.cs

#### 4.2.4.6 User.cs

เป็นส่วนประกอบของโครงสร้างข้อมูลของผู้ใช้งาน

```
[Key] // บอกถึงว่าเป็น primary key
public int user_id { get; set; }
public string email { get; set; } = string.Empty;
public byte[] pass_hash { get; set; } = new byte[32];
public byte[] pass_salt { get; set; } = new byte[32];
public string mobile_no { get; set; } = string.Empty;
public string first_name { get; set; } = string.Empty;
public string last_name { get; set; } = string.Empty;
public string verificationToken { get; set; } = string.Empty;
public string profile_pic_url { get; set; } = string.Empty;
public string display_name { get; set; } = string.Empty;
public int user_type { get; set; }
public int verified { get; set; }
public int report_count { get; set; }
```

รูปที่ 57 code User.cs

#### 4.2.4.7 User\_Follows.cs

เป็นส่วนประกอบของโครงสร้างข้อมูลของผู้ใช้งานที่สามารถบอกได้ว่ามีผู้ใช้งานใดติดตามผู้ใช้งานคนใดบ้าง

```
[PrimaryKey(nameof(user_id), nameof(follower_id))]
public class Users_Follows {
    public int user_id { get; set; }
    public int follower_id { get; set; }
}
```

รูปที่ 58 code User\_Follows.cs

#### 4.2.4.8 Users\_SharedPosts.cs

เป็นส่วนประกอบของโครงสร้างข้อมูลของผู้ใช้งานที่สามารถบอกได้ว่ามีผู้ใช้งานใดทำการ share โพสต์อะไรไปบ้าง

```
[PrimaryKey(nameof(user_id), nameof(shared_post_id))]
public class Users_SharedPosts {
    public int user_id { get; set; }
    public int shared_post_id { get; set; }
}
```

รูปที่ 59 code Users\_SharedPosts.cs

#### 4.2.5 Program.cs

เป็นส่วนหลักเพื่อใช้ในการ run โปรแกรม มี services ต่างๆ เมื่อทำการเริ่มเรียกใช้แล้วจะทำการดึง services อื่นๆ ต่อกันไปเรื่อยๆ

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddDbContext<DataContext>(options => {
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"));
});
```

รูปที่ 60 ส่วนหนึ่งของ code ใน Program.cs

## 4.3 Github

### 4.3.1 Frontend

[underpoom/softarchfinalprojectflutter \(github.com\)](https://github.com/underpoom/softarchfinalprojectflutter)

### 4.3.2 Backend

[Natashi/KMITL\\_SoftArch\\_KMITLNews\\_Backend: Backend Web API for my Software Architectures class group project \(github.com\)](https://github.com/Natashi/KMITL_SoftArch_KMITLNews_Backend)

## บทที่ 5

### การแบ่งงาน

#### 5.1 ฝ่ายในการทำงาน

ในการทำงานแบ่งออกเป็น 2 ฝ่ายคือ Backend และ Frontend

##### 5.1.1 ผู้ดูแลฝ่าย Backend

ประกอบด้วยทั้งหมด 4 ท่าน ได้แก่

- |                  |                |
|------------------|----------------|
| 1. นาย ธีรวัฒน์  | จิตรานุเคราะห์ |
| 2. นาย ธนภูมิ    | ชัยพรรณา       |
| 3. นาย นันทวัฒน์ | รัตนเรืองวิมาน |
| 4. นาย ปัญญวัฒน์ | ธนทัตธาดา      |

##### 5.1.2 ผู้ดูแลฝ่าย Frontend

ประกอบด้วยทั้งหมด 3 ท่าน ได้แก่

1. นาย นพสิทธิ์ จันทรวีระกุล
2. นาย พนธกร สุจิตขวานนท์
3. นาย พศิน แก้วนิล

## บทที่ 6

### สรุปการทำงาน

ในการทำแอปพลิเคชัน KMITL News ทางทางทีมผู้พัฒนาได้ทำการแบ่งงานให้แต่ละคนในทีมทำ แต่ถึงจะมีการแบ่งงานอย่างชัดเจน ทางทีมเน้นการทำงานแบบช่วยเหลือซึ่งกันและกัน ในการทำงานหากมีคนใดในทีมติดปัญหาในการทำงาน คนในทีมพร้อมที่จะเข้าไปช่วยร่วมแก้ปัญหาไม่ว่าจะเป็นในฝ่าย frontend หรือ backend

ในการทำงาน coding อาจพบเจอปัญหาในการทำงานอยู่บ้างซึ่งเกิดจากปัญหาเรื่องความชำนาญของภาษา เนื่องจากบางคนในทีมเริ่มใช้ภาษา dart และ C# เป็นภาษาแรก ทำให้ต้องใช้เวลาในการเรียนรู้ เมื่อเริ่มดำเนินงานความเชี่ยวชาญเพิ่มมากขึ้นทำให้งานสามารถดำเนินได้รวดเร็วมากยิ่งขึ้นส่งผลให้สามารถพัฒนาแอปพลิเคชันออกมาได้ตามที่ตั้งเป้าหมายเอาไว้และสามารถพัฒนาจนเสร็จภายในเวลาที่กำหนดพร้อมกับประยุกต์ร่วมกับบทเรียน

การประยุกต์ร่วมกับบทเรียนเข้ามาใช้ทั้ง MVC, Client server & N-tier เข้ามาใช้ในการจัดการใน backend เพื่อให้สามารถแก้ไขได้ง่ายและเชื่อมต่อกับ backend อย่างมีประสิทธิภาพ ซึ่งส่งเสริมได้ quality attribute ทั้ง Availability Performance และ Modifiability อีกทั้งยังมีการนำ Plugin จากภายนอกเข้ามาเพื่อให้หน้า frontend มีความสามารถมากยิ่งขึ้น ซึ่งส่งเสริมเรื่อง integrability เพื่อให้แอปพลิเคชันที่พัฒนาขึ้นมาสามารถใช้งานได้อย่างมีประสิทธิภาพและง่ายต่อผู้ใช้งาน