รายงานสหกิจศึกษาฉบับสมบูรณ์

การพัฒนาแอปพลิเคชันทางการเงินบนระบบปฏิบัติการแอนดรอยด์

Development of Android Banking Application

นางสาววริศรา ศิริอักษร

สาขาวิศวกรรมสารสนเทศ

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2562

ชื่อโครงงานสหกิจศึกษา การพัฒนาแอปพลิเคชันทางการเงินบนระบบปฏิบัติการแอนดรอยด์

ชื่อ-สกุล นักศึกษา นางสาววริศรา ศิริอักษร

คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมคอมพิวเตอร์ **สาขาวิชา** วิศวกรรมสารสนเทศ

ชื่อ-สกุล อาจารย์นิเทศ ผศ.ไพศาล สิทธิโยภาสกุล

ชื่อ-สกุล ผู้นิเทศงาน คุณปวีณ ปิยะศิลป์

สถานประกอบการ บริษัท กสิกร เทคโนโลยี กรุ๊ป เซเครเทเรียต จำกัด

บทคัดย่อ

แอปพลิเคชัน K-PLUS ของธนาคารกสิกรไทยเป็นแอปพลิเคชันสำหรับการทำธุรกรรมทางการเงินบนโทรศัพท์มือถือ และเพื่อตอบสนองกับไลฟ์สไตล์ปัจจุบันของผู้คนให้ได้มากที่สุด ทางธนาคารจึงมีการพัฒนาความสามารถและฟีเจอร์ (Feature) ใหม่ ๆ ของแอปพลิเคชันอยู่ตลอดเวลา โดยในโครงงานฉบับนี้ผู้จัดทำได้ร่วมพัฒนาฟีเจอร์ของแอปพลิเคชัน K-PLUS กับทางองค์กรในบทบาทของ Front-end Client Developer และมีฟีเจอร์ใหม่ที่ได้ร่วมพัฒนากับทางบริษัท ได้แก่ Transaction Suggestions ที่จะแนะนำรายการใช้บ่อยของผู้ใช้งานขึ้นมาที่หน้า Home โดยไม่ต้องเพิ่ม Favorites การพัฒนาหน้า Credit card 2019 ที่ประกอบด้วยเพิ่มฟีเจอร์ Installment Plans ที่ช่วยจัดการการแบ่งชำระค่าบัตรเครดิต และฟีเจอร์ Get Cash ที่ทำให้สามารถกดเงินสดโดยตัดผ่านยอดบัตรเครดิตได้ และฟีเจอร์ Voice Command หรือระบบสั่งการด้วยเสียง ที่ช่วยให้การทำธุรกรรมออนไลน์บนโทรศัพท์มือถือให้สะดวกสบายขึ้น และยังได้ร่วมพัฒนาความสามารถของฟีเจอร์เก่าที่มีอยู่แล้วให้ดีขึ้น ทั้งการปรับแก้หน้ารูปแบบของหน้า Home การ Refator ตัว Code ของแอปพิลเคชัน การแก้ข้อผิดพลาดที่ไม่พึงประสงค์ (Bug) เพื่อทำให้แอปพลิเคชัน K-PLUS มีความทันสมัยและตอบโจทย์การทำธุรกรรมในปัจจุบันให้มากที่สุด โดยในการทำโครงงานนี้มีการใช้ภาษา Kotlin เป็นภาษาหลักในการพัฒนา ใช้ Design Patten แบบ MVP ภายใต้การทำงานในรูปแบบแนวคิดของ Agile ที่ช่วยเพิ่มประสิทธิภาพการทำงานทั้งในทีมและองค์กร

Coopoertive Title: Development of Android Banking Application

Student Intern Name: Waritsara Siriaksorn

Faculty: Engineering **Department:** Computer Engineering

Major: Information Engineering

Advisor Name: Asst.Prof. Paisan Sithiyopasakul

Mentor Name: Mr. Paween Piyasil

Company: Kasikorn Technology Group Secretariat Co.,Ltd

ABSTRACT

K-PLUS is a Kasikorn bank's mobile banking application that has been developed continuously. In this I joined K-PLUS developer team as Front-end Client Developer and developed new features, the first one is Transaction Suggestions the feature that suggest your most frequent transactions on Home page. The second one is Credit card 2019 that consists of Installment Plans which is the feature that can help you to plan your payment of credit card debt, and Get Cash the feature that make you can withdraw cash by using credit card. And the last one is Voice command feature that can help you use mobile banking easier by your voice. Moreover, I involved with the previous features such as Home page improvement, refactoring Java code to Kotlin and resolve deflect from tester team. In this project the main languages that used is Kotlin and used MVP design pattern under Agile methodlogy to improve the performace of team and organize.

กิตติกรรมประกาศ

ข้าพเจ้าได้มีโอกาสเข้าร่วมทำงานกับทางบริษัท กสิกร บิซิเนส-เทคโนโลยี กรุ๊ป ในทีม K-PLUS ตั้งแต่วันที่ 4 มิถุนายน 2562 จนถึงวันที่ 29 พฤศจิกายน 2562 รวมระยะเวลาทั้งสิ้น 125 วัน ซึ่งการได้ร่วมงานกับบริษัทเทคโนโลยีชั้นนำของประเทศ ทำให้ข้าพเจ้าได้รับความรู้และประสบการณ์มากมายจากการทำงาน การเข้าร่วมกิจกรรมต่าง ๆ ทั้งได้เรียนรู้ถึงวัฒนธรรมองค์กร และการทำงานอย่างมีประสิทธิภาพ ซึ่งเป็นประโยชน์ต่อข้าพเจ้าในการนำไปปรับใช้ในอนาคต

ข้าพเจ้าขอขอบพระคุณบริษัท กสิกร บิซิเนส-เทคโนโลยี กรุ๊ป ที่ให้โอกาสให้ข้าพเจ้าในการมาร่วมทำงานและกิจกรรมต่าง ๆ ภายในองค์กรมากมาย

ข้าพเจ้าขอขอบพระคุณ คุณปวีณ ปิยะศิลป์ ผู้นิเทศงานของข้าพเจ้า ตลอดจนพี่ๆในทีม K-PLUS ที่คอยให้ความช่วยเหลือ ดูแล คอยแนะนำ ให้ความเมตตาและให้การสนับสนุนข้าพเจ้าเป็นอย่างดี ข้าพเจ้าจะนำความรู้และคำแนะนำต่าง ๆ ไปปรับใช้ในการทำงานในภายภาคหน้า และในการใช้ชีวิตประจำวันต่อไป

ข้าพเจ้าขอขอบพระคุณ ผศ.ไพศาล สิทธิโยภาสกุล และอาจารย์ที่ปรึกษาท่านอื่น ๆ ที่คอยให้ความสนับสนุนและช่วยเหลือข้าพเจ้ามาตลอด ทำให้ข้าพเจ้าสามารถทำงานสหกิจศึกษานี้สำเร็จลุล่วงได้ด้วยดี

และท้ายที่สุด ขอขอบพระคุณท่านอื่น ๆ ที่ไม่ได้กล่าวนาม ทั้งเพื่อนพ้องและครอบครัว ที่ได้มีส่วนเกี่ยวข้องในการช่วยเหลือข้าพเจ้าทั้งด้วยทางตรงและทางอ้อม ทำให้ข้าพเจ้าได้มีประสบการณ์และความรู้ในการทำงานที่เพิ่มมากขึ้น ข้าพเจ้าขอขอบพระคุณไว้ ณ ที่นี้ด้วย

วริศรา ศิริอักษร

สารบัญ

หน้า

สารบัญรูปภาพ

หน้า

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

แอปพลิเคชัน K-PLUS เป็นแอปพลิเคชันทางการเงินของธนาคารกสิกรไทยที่ช่วยให้ลูกค้าของธนาคารกสิกรสามารถทำธุรกรรมออนไลน์บนสมาร์ตโฟนได้ทุกที่ทุกเวลาอย่างรวดเร็ว

ในการเข้าร่วมโครงการสหกิจในครั้งนี้ผู้ทำโครงงานได้เข้าร่วมทำงานในบริษัท กสิกร เทคโนโลยี กรุ๊ป เซเครเทเรียต จำกัด ซึ่งเป็นบริษัทเทคโนโลยีของธนาคารกสิกรไทย ที่มีจุดมุ่งหมายและวิสัยทัศน์ว่า “เพื่อเป็นผู้นำด้านเทคโนโลยีในการสร้างธนาคารแห่งอนาคตแบบดิจิทัลที่สมบูรณ์แบบ เพื่อให้ลูกค้าสามารถเข้าถึงบริการและทำกิจกรรมทางการเงินทุกประเภท ในทุกช่องทางเน้นการให้บริการที่ง่าย สะดวก รวดเร็วและปลอดภัย เป็นส่วนประกอบสำคัญในชีวิตประจำวันของลูกค้าและช่วยลูกค้าตัดสินใจทางการเงิน เพื่อให้การใช้ชีวิตในรูปแบบที่เป็นดิจิทัลในทุกวันได้อย่างไร้ขีดจำกัด[1]”

โดยในปัจจุบันแอปพลิเคชัน K-PLUS มียอดดาวน์โหลดมากกว่า 10 ล้านดาวน์โหลด* และมีการพัฒนาอย่างต่อเนื่องเพื่อใช้เทคโนโลยีในปัจจุบันให้เกิดประโยชน์สูงสุดและความสะดวกสบายต่อลูกค้าอย่างต่อเนื่อง จึงเป็นที่มาของการทำโครงงานและพัฒนาฟีเจอร์เพิ่มเติมใน K-PLUS

1.2 วัตถุประสงค์ของโครงงาน

- เพื่อเพิ่มประสิทธิภาพให้แก่แอปพลิเคชัน K-PLUS
- เพื่อเพิ่มฟีเจอร์การใช้งานแก่แอปพลิเคชัน K-PLUS
- เพื่อเพิ่มรายได้และกลุ่มลูกค้าใหม่ ๆ ให้แก่บริษัท
- เพื่อทดสอบการใช้งานแอปพลิเคชันก่อนปล่อยสู่ท้องตลาด
- เพื่อผู้ทำโครงงานได้เรียนรู้การทำงานจริงและเตรียมความพร้อมเข้าสู่โลกแห่งการทำงานในอนาคตอันใกล้

* ข้อมูลจำนวนการดาวน์โหลดจาก Play Store ณ เดือน พฤศจิกายน 2562

1.3 วิธีการดำเนินโครงงาน

- รับ Requirement จาก BA และรับ UI จากทาง Designer
- ประชุมแบ่งงานภายในทีมและวางแผนการทำงาน
- เริ่มพัฒนาตัวฟีเจอร์
- ทดสอบและแก้ไขข้อผิดพลาด โดยเปรียบเทียบกับฟีเจอร์บนระบบปฏิบัติการ iOS ประกอบคู่กันไปด้วย
- นำฟีเจอร์ไปใช้งานจริง (ภายในบริษัท)
- นำฟีเจอร์ไปใช้งานจริง (ภายนอกบริษัท)

1.4 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่ได้รับจากการเข้าร่วมโครงการสหกิจกับทางบริษัท กสิกร บิซิเนส-เทคโนโลยี กรุ๊ป จำแนกได้เป็น 2 ส่วน ดังนี้

1.4.1 ประโยชน์ต่อบริษัท

- ได้พัฒนาฟีเจอร์การทำธุรกรรมผ่านสมาร์ตโฟนรูปแบบใหม่ ที่ช่วยเหลือและเอื้อประโยชน์แก่ผู้พิการทางสายตา
- ได้เพิ่มประสิทธิภาพการทำงานของแอปพลิเคชัน
- ได้แนวความคิดเพิ่มเติมในการพัฒนาแอปพลิเคชันไปในทางใหม่ ๆ

1.4.2 ประโยชน์ต่อตัวนักศึกษา

- ได้เรียนรู้การทำงานในสภาพแวดล้อมจริงของบริษัทและศึกษาวัฒนธรรมองค์กร
- ได้เรียนรู้การทำงานภายใต้แนวคิด Agile
- ได้เรียนรู้วิธีการเขียน Design Pattern MVP ที่ถูกต้องและมีประสิทธิภาพ
- ได้เรียนรู้และต่อยอดการเขียนแอนดรอยด์แอปพลิเคชันด้วยภาษา Kotlin
- ได้ฝึกการรับมือกับปัญหา และแก้ไขปัญหาเฉพาะหน้า
- ได้พัฒนาทักษะการสื่อสารภาษาอังกฤษในชีวิตประจำวัน

บทที่ 2
แนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ภาษา Kotlin

ภาพที่ 2.1 สัญลักษณ์ของภาษา Kotlin

(ที่มา: https://kotlinlang.org/user-groups/branding.html)

Kotlin เป็นภาษาที่ใช้ในการพัฒนาแอปพลิเคชันแอนดรอยด์ ที่ถูกพัฒนามาจากภาษา Java เพื่อแก้ไขจุดบกพร่องในตัวภาษาเดิม เช่น การไม่มี Null Safety หรือ Extension Function โดยมีแนวคิดที่จำเป็นต้องเข้ากับ Java ได้แบบ 100% เพื่อสามารถใช้ประโยนช์จาก Tools หรือ Library เก่าของ Java ได้ และสามารถเพิ่ม Feature อื่น ๆ ที่ไม่มีใน Java ได้ และมีเป้าหมายหลักคือการนำ Kotlin มาใช้แทน Java ได้ในทันที โดยยังรักษา Code Java เดิมได้และยังทำงานร่วมกันได้อย่างปกติ

2.1.1 เปรียบเทียบข้อดีข้อเสียของ Kotlin ต่อ Java

ข้อดี

- สามารถทำงานกับภาษา Java ได้อย่างสมบูรณ์แบบ
- เป็นภาษาที่สั้นและกระชับมากว่า Java จึงสามารถอ่านและพัฒนาได้ง่ายกว่าและไม่จำเป็นต้องมี ; ปิดท้ายคำสั่ง
- Kotlin ได้ถูกพัฒนามาเป็นภาษาที่ฉลาดขึ้นและปลอดภัยมากขึ้น เนื่องจากมีการรองรับ Error ที่มากขึ้น และป้องกันการปิดตัวอย่างกระทันหัน (Crash) ของแอปพลิเคชัน เช่น มีการใส่ Null Safety ที่มีการตรวจสอบค่าที่ได้รับว่าเป็นชนิด null หรือไม่ก่อนจะทำงานหรือเรียกใช้ฟีเจอร์

ข้อเสีย

- มีจำนวน Method ที่เพิ่มมากขึ้นและขนาดแอปพลิเคชันที่จะใหญ่ขึ้นกว่าเดิม
- การประมวลผลที่นานขึ้นในระหว่างการใช้งานเนื่องจากต้องแปลงกลับเป็น Java

2.1.2 เปรียบเทียบการใช้งานและคำสั่งของ Kotlin และ Java[2]

ผู้จัดทำโครงงานได้ทำการเปรียบเทียบการใช้งานและคำสั่งของ Kotlin และ Java ไว้ตามตารางที่ 2.1

คำสั่ง \ ภาษา	Java	Kotlin
การใช้ ;	ต้องมี ; ปิดท้าย	ไม่ต้องมี ;
การประกาศค่าคงที่	ใช้ final	ใช้ val
การ New Object	ใช้คำสั่ง new	ไม่ต้องใช้ new
การเก็บค่า Null	บาง Type ไม่สามารถเก็บค่า null ได้ และอาจส่งผลให้เกิดการแครช (Crash)	ทุก Type สามารถใส่ ? ต่อท้ายเพื่อเก็บค่า null ได้และหาเป็นค่าห้าม null จะต่อท้ายด้วย !!

การใช้ Null Safety	`String data = null;`
	`if (data != null) {`
	`data = data.toUpperCase();`
	`}`
		`var data: String? = null`
		`data = data?.toUpperCase()`

การใช้ For Loop
(กำหนดให้ I, A, B เป็นค่าคงที่)
	`for(i = a ; i < b; i++){`
	`}`
		`for (i in a..b) {`
		`}`
		หรือ
		`for (i in a until b) {`
		`}`

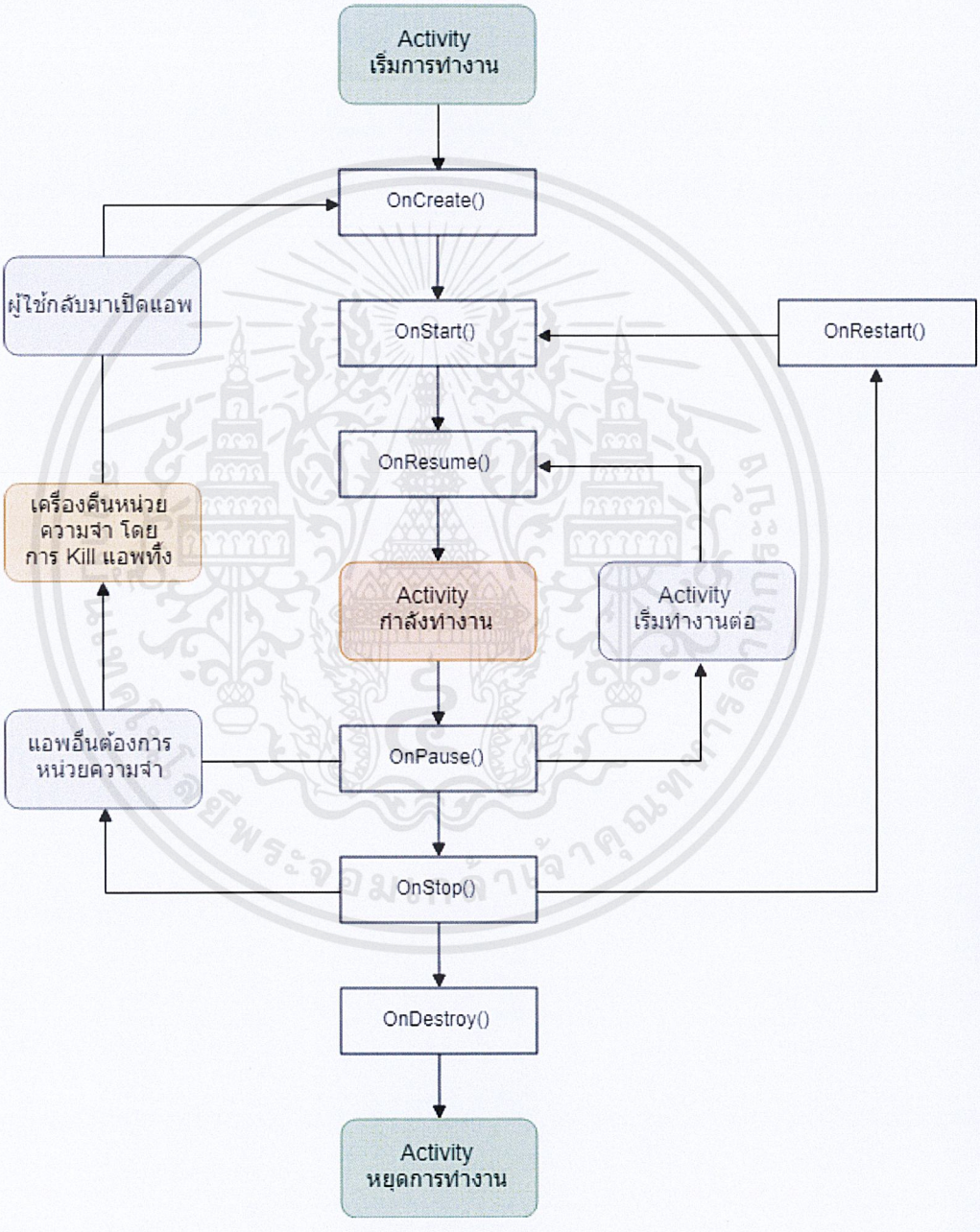
การใช้ Switch Case
(กำหนดให้ I เป็นค่าคงที่)
	`switch (i) {`
	`case i:`
	`break;`
	`default:`
	`break;`
	`}`
		`when (i) {`
		`i -> {`
		`}`
		`else -> {`
		`}`
		`}`
		หรือ
		`when (1){`
		`in 1..2 -> {`
		`}`
		`else -> {`
		`}`
		`}`

ตารางที่ 2.1 เปรียบเทียบคำสั่งของภาษา Java และ Kotlin

2.2 Android Lifecycle[3]

Lifecycle คือวงจรชีวิตการเกิดดับของเมธอด (Method) ภายในแอปพลิเคชันเมื่อเกิดกิจกรรมต่าง ๆ ขึ้น เช่น การเปิด-ปิดแอปพลิเคชัน โดยทางผู้จัดดทำได้ทำการแบ่ง Lifecycle ออกเป็น 2 ชนิดคือ Lifecycle ของ Activity และ Lifecycle ของ Fragment ดังนี้

2.2.1 Activity Lifecycle



ภาพที่ 2.2 Lifecycle ของ Android Activity

Lifecycle Activity หรือวงจรชีวิตของ Activity คือการบอกว่า Method ใดจะเริ่มใช้งานเมื่อเกิดเหตุการณ์ใดบ้าง ดังภาพที่ 2.2 ผู้พัฒนาแอปลิเคชันจำเป็นต้องทราบถึงการ

ทำงานเบื้องต้นของ Lifecycle เพื่อใช้ Method เหล่านั้นอย่างถูกต้องและเหมาะสม โดยมี Method หลัก ๆ ดังนี้

- **onCreate():** Android จะทำการเรียก onCreate() เมื่อ Activity เริ่มการทำงานในหนึ่งช่วงเวลาของแอปพลิเคชันนั้น ๆ โดยอาจมีการสร้าง (Create) และ ทำลาย (Destroy) Activity อยู่เรื่อย ๆ เช่น เมื่อ User ทำการหมุนหน้าจอ (Rotate Screen) จะส่งผลให้ Activity ถูก Destroy และ Instance ใหม่ของ Activity เดิมก็จะถูก Create อีกครั้ง

มีความเป็นไปได้ที่แอปพิลเคชันที่ใช้อาจถูก Kill ถ้าหากว่าแอปพลิเคชันทำงานอยู่แบบพื้นหลัง (Background) และระบบ (System) กำลังอยู่ในสภาวะต้องการทรัพยากรเพิ่มเติม (Low Resources) ซึ่งหาก Application ถูก Kill ไปแล้ว และตัวแอปพลิเคชันที่กำลังใช้งานอยู่ กลับขึ้นมาอยู่บนเบื้องหน้า (Foreground) การร้องขอใหม่ของ Activity ตัวเดิมจะถูกสร้างขึ้นและ onCreate() ก็จะถูกเรียกอีกครั้ง

- **onStart():** หาก Application ถูกสั่งให้ไปอยู่ใน Background โดยไม่มีการคืนหน่วยความจำ onStart() จะถูกเรียกเมื่อกลับมาที่ Application อีกครั้ง

- **onResume():** เมื่อแอปพลิเคชันที่ถูก Start ขึ้นมา onResume() จะถูกเรียกหลังจาก onCreate() และ onStart() หรือเมื่อ Application เปลี่ยนสถานะจาก Background กลับมาอยู่บน Foreground อีกครั้ง onResume() ก็จะถูกเรียกเช่นเดียวกัน onResume() จะถูกเรียก ก่อนที่ Activity จะสามารถมองเห็นได้บน Screen

- **onPause():** onPause() จะถูกเรียกเมื่อ Application กำลังเปลี่ยนจากสถานะจาก Foreground ไปยัง Background ซึ่งถ้าหากมีการตรรกะการประมวลผลที่ควรจะ Run เฉพาะตอนที่ Activity อยู่บน Screen เช่น การแสดง Animation ผู้พัฒนาควรจะสั่งหยุดการประมวลผลดังกล่าวใน onPause()

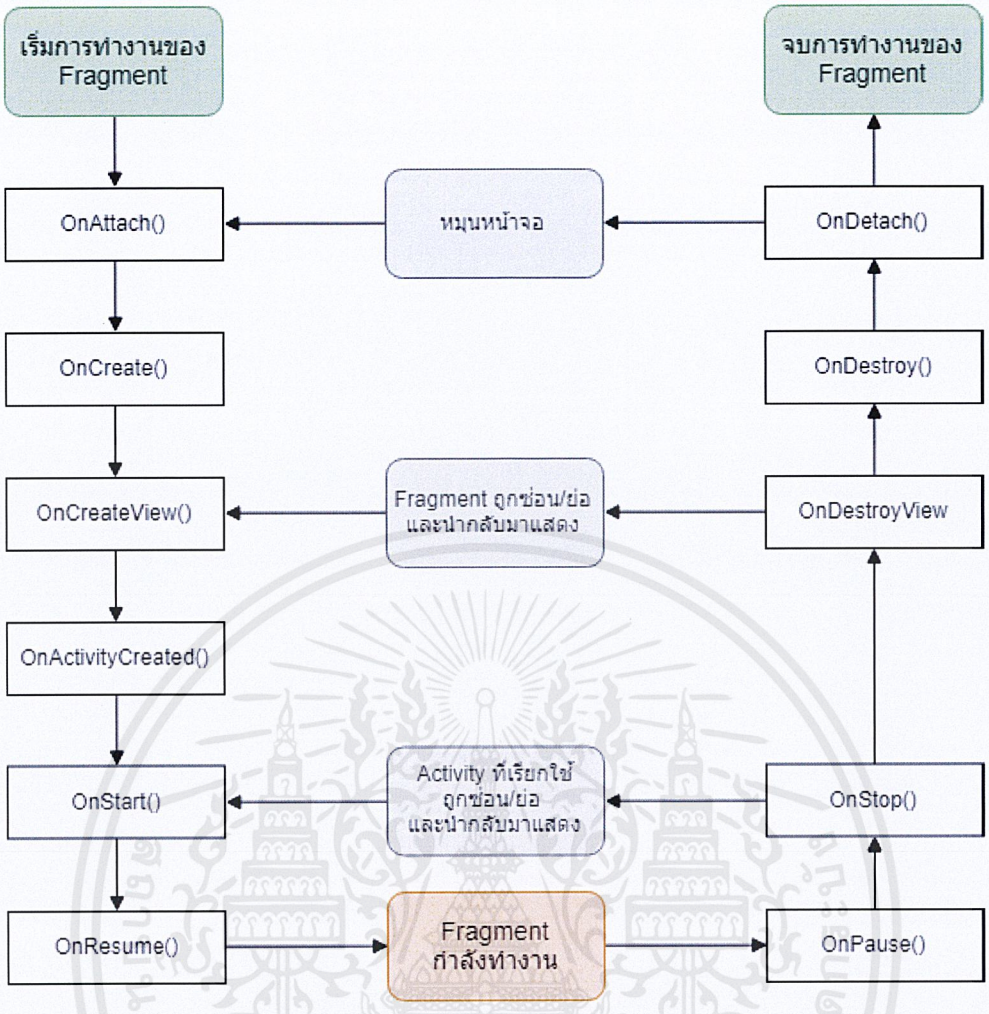
onPause() จะถูกเรียกในเวลาที่สั่งเปิด Activity อื่นขึ้นมาอยู่เหนือ Activity ที่กำลังแสดงผลอยู่ Application อาจถูก Kill เมื่อ System อยู่ในสภาวะ Low Resources ซึ่งกระบวนการ Kill จะเกิดขึ้นหลังจาก onPause() ซึ่งไม่ได้เกิดเป็นปกติ แต่อย่างน้อยก็ควรเตรียมรับมือไว้ โดย Method นี้เป็น Method ที่สำคัญ เพราะเป็นเหมือนการเตือนครั้งสุดท้ายก่อนที่ Activity จะหายออกจาก Screen ซึ่ง Method นี้ เป็นที่ที่ควรจะ Save ข้อมูลที่สำคัญลง Disk, Database หรือ Preferences

- **onStop():** onStop() จะถูกเรียกเมื่อ Activity ออกจาก Screen เรียบร้อยแล้ว หรือเมื่อเปลี่ยนไปติดต่อกับอีก Activity หนึ่งแทน (เปลี่ยนจากสถานะ Active ไปเป็น Inactive) แต่นี่ไม่ได้หมายความ Activity ถูกปิดตัวลง เพียงแค่เปลี่ยนมาอยู่ในสถานะ Inactive ถ้าหากมีการประมวลผลใดที่ต้องการ Run เฉพาะตอนที่ Active นี่คือ Method ที่เหมาะสมที่จะสั่งหยุดการประมวลผลดังกล่าว

- **onDestroy():** onDestroy() เป็น Method สุดท้ายที่จะถูกเรียก ก่อนที่ Activity จะปิดตัวลงอย่างถาวร เป็น Method ที่ใช้ในการคืนค่า Resources หรือ Clean Up ใด ๆ โดยภายใน Method นี้ ควรจะสั่งปิด Process ใด ๆ ที่สั่ง Run ไว้ใน Background ให้หมด

2.2.2 Fragment Lifecycle[4]

Fragment นั้นจะมี Lifecycle ที่คล้ายกับ Activity อยู่ และจะมีส่วนที่แตกต่างกันอยู่เล็กน้อย ดังนั้นแล้วก่อนที่ผู้พัฒนาจะเรียนรู้ถึง Lifecycle ของ Fragment นั้นต้องทำความเข้าใจใน Lifecycle ของ Activity ให้ดี เพื่อที่จะสามารถประยุกต์ใช้งานกับ Fragment ได้อย่างมีประสิทธิภาพ โดยเฉพาะการทำงานที่ต้องใช้ร่วมกับ Lifecycle โดย Lifecycle ของ Fragment มี Method คร่าว ๆ ตามดังภาพที่ 2.3



ภาพที่ 2.3 Lifecycle ของ Android Fragment

ซึ่งแต่ละ Method จะถูกเรียกใช้งานเมื่อเกิด Event ดังนี้

- **onAttach():** เมื่อ Activity เรียกใช้งาน Fragment นั้น ๆ (เป็นส่วนที่ไว้สำหรับการเชื่อม Fragment เข้ากับ Activity)

- **onCreate():** เมื่อทำการสร้าง Fragment ขึ้นมา (Initial)

- **onCreateView():** ใช้กำหนด Layout ที่จะใช้กับ Fragment (เป็นส่วนที่ไว้สำหรับการเชื่อม Layout ที่ทำไว้เข้ากับ Fragment)

- **onActivityCreated():** เมื่อ Activity ถูกสร้างขึ้นและผูก Fragment เข้ากับ Activity เรียบร้อยแล้ว

- **onStart():** เมื่อ Fragment แสดงขึ้นมาให้เห็นบนหน้าจอเรียบร้อยแล้ว (Visible)

- **onResume():** เมื่อ Fragment พร้อมสำหรับติดต่อกับผู้ใช้งาน (Interactive)

- **onPause():** เมื่อ Fragment หยุดติดต่อกับ User (No Longer Interactive)

- **onStop():** Fragment หยุดแสดงให้เห็นบนหน้าจอ (No Longer Visible)

- **onDestroyView():** ใช้เพื่อเคลียร์ทรัพยากรที่เกี่ยวข้องกับการแสดงผล

- **onDestroy():** ใช้สำหรับการเคลียร์สถานะ (State) ทั้งหมดของ Fragment นั้น ๆ

- **onDetach:** เมื่อ Activity เลิกใช้งาน Fragment นั้น ๆ (Fragment กับ Activity ไม่เชื่อมต่อกันแล้ว)

2.3 Design pattern แบบ MVP[5]

เนื่องจากการพัฒนาแอปพลิเคชันที่เป็นโปรเจกต์ ขนาดใหญ่มักจะมีความซับซ้อนสูง จึงเป็นเรื่องยากที่จะหาส่วนการทำงานเฉพาะจุดเมื่อต้องการแก้ไขหรือพัฒนาต่อในจุดนั้น ๆ ยิ่งถ้าหากมีผู้พัฒนาหลายคนในโปรเจกต์หนึ่งก็มักมีปัญหาเรื่องสไตล์การเขียนโค้ดของแต่ละคนที่แตกต่างกัน ดังนั้นแล้วการแบ่งหัวข้อเป็นชนิดต่าง ๆ รวมถึงการวางแผนและแนวทางการพัฒนาก่อนจะทำให้การพัฒนาและแก้ไขง่ายขึ้น โดยการวางแผนงานล่วงหน้าเพื่อแก้ปัญหาข้างต้นก็คือการมีสิ่งที่เรียกว่า Design Patten

สำหรับ Design Pattern ทุกรูปแบบจะจุดประสงค์หรือมี Separation of Concerns อย่างเดียวกัน คือการแยกการทำงานและแบ่งส่วนเพื่อพยายามไม่ให้ View ติดต่อกับ Model โดยตรงหรือให้ทั้งสองอย่างนั้นรู้ถึงการทำงานของกันและกันน้อยที่สุด โดยสามารถดูความหมายของคำว่า View และ Model ได้ดังนี้

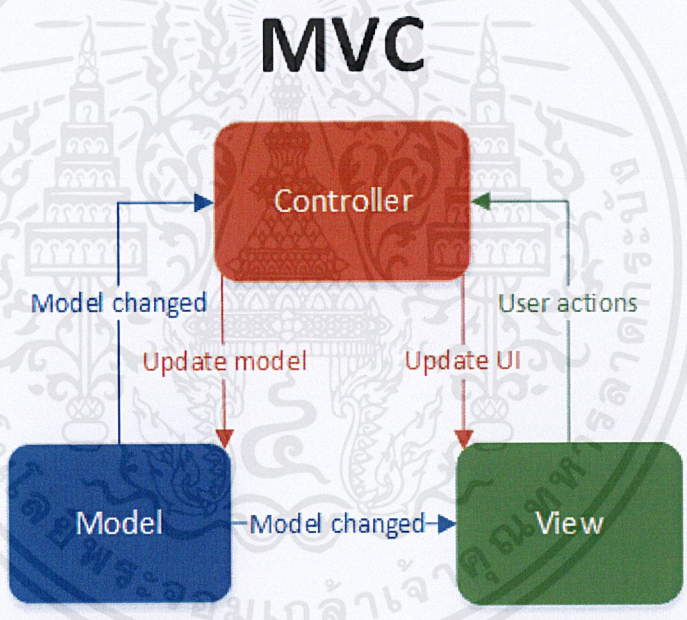
- **View:** คือส่วนของ Interface หรือหน้าตาภายนอกของแอปพลิเคชัน ใช้ในการติดต่อกับ User หรือแสดงผลข้อมูลต่าง ๆ โดยนำข้อมูลที่จำแสดงมาจากตัว Model

- **Model:** คือส่วนที่เก็บข้อมูลภายในแอปพลิเคชัน ที่ทำหน้าที่การแปลการทำงานของระบบ สามารถหมายถึงส่วนที่เก็บข้อมูลโดยถาวรก็ได้หรือส่วนของการเก็บข้อมูล

ในปัจจุบันการเขียนแอปพลิเคชันแพลตฟอร์มแอนดรอยด์จะมี Design Patten ที่นิยมอยู่ 3 รูปแบบนั้นคือ MVC MVP และ MVVM โดยมีความแตกต่างกันที่การพยายามนำตัวกลางที่แตกต่างกันไปในแต่ละ Patten มาใช้แยก View และ Model ออกจากกัน โดยในการทำโปรเจกต์ “การพัฒนาแอปพลิเคชันทางการเงินบนระบบปฏิบัติการแอนดรอยด์” มีการใช้ Design Patten แบบ MVP (Model View Presenter) ที่ได้รับการพัฒนามาจาก MVC ให้มีความชัดเจนในการแบ่ง View กับ Model มาขึ้น

- **MVC (Model View Controller)** คือ Pattern ในยุคเริ่มแรกของความพยายามที่จะแยกส่วนของ Model และ View ออกจากกันตามหลักของ Separation of Concerns โดยใน MVC จะมี Controller เพิ่มขึ้นมาเป็นตัวกลาง โดยหน้าที่ของ Controller ก็คือการจัดการกับ Model โดยขึ้นอยู่กับการกระทำที่ได้จาก View และยังทำหน้าที่สรรหาข้อมูลจาก Model เพื่อนำไปแสดงผลที่ View อีกด้วย ใน Pattern แบบ MVC นั้น View อาจจะสามารถรับรู้ถึงการเปลี่ยนแปลงของข้อมูลผ่าน Model ได้โดยตรง โดยมีข้อเสียดังนี้

- Controller ยึดติดกับส่วนของ View มากเกินไป ทำให้ยังไม่ชัดเจนนักที่จะแบ่งส่วนระหว่าง View กับ Model
- Maintenance ยาก เมื่อเวลาผ่านไป จะมี Code เพิ่มขึ้นในตัว Controller มากขึ้นเรื่อย ๆ ทำให้ดูแลลำบาก



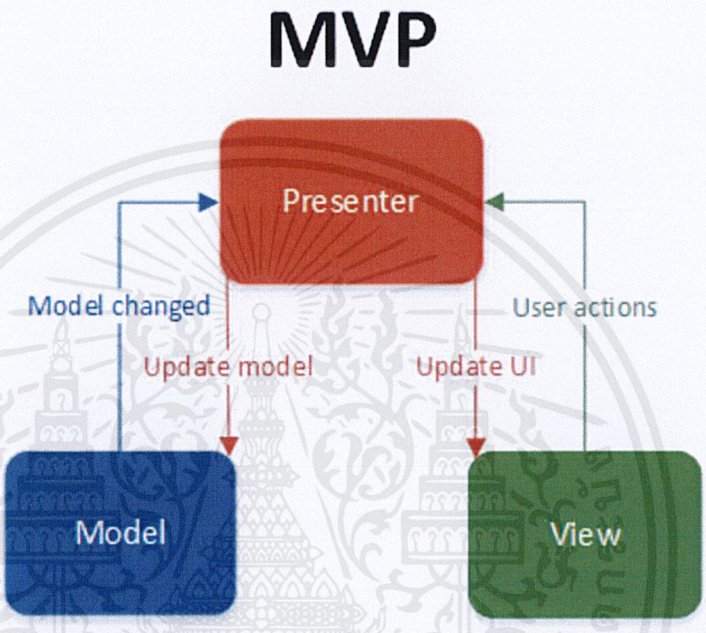
ภาพที่ 2.4 หลักการทำงานของ Design pattern แบบ MVC

(ที่มา: https://medium.com/@beyram.ghali/android-model-view-presenter-mvp-design-pattern)

ด้วยข้อเสียที่กล่าวมาของ MVC นั่นเองทำให้เหมาะกับการใช้ในโปรเจกต์ขนาดเล็ก และสำหรับการใช้งานในโปรเจกต์ที่มีขนาดใหญ่ขึ้นก็ได้เกิดการพัฒนา Design patten แบบ MVP (Model View Presenter) ขึ้นมา

- **MVP (Model View Presenter)** คือความพยายามต่อมาที่ต้องการจะแยก View กับ Model ออกจากกันให้มากขึ้น โดยมีเป้าหมายคือแก้ไขปัญหาที่มีใน MVC ดังที่กล่าวมา เช่น View กับ Controller ยึดติดกันมากเกินไป และ View กับ Model ไม่ได้แยก

ขาดกันอย่างแท้จริง โดย MVP นั้นจะมี Presenter ที่ทำหน้าที่เป็นตัวกลางเพิ่มขึ้นมา ซึ่ง Presenter จะอยู่ระหว่าง View และ Model คล้ายกับ Controller ใน MVC แต่ว่า การกระทำทุกอย่างที่ผ่าน View จะต้องถูกส่งผ่าน Presenter และ ทุกการเปลี่ยนแปลงของ Model ก็จะต้องถูกส่งผ่าน Presenter เช่นกัน โดยที่ Model กับ View จะไม่สามารถติดต่อกันได้โดยตรง

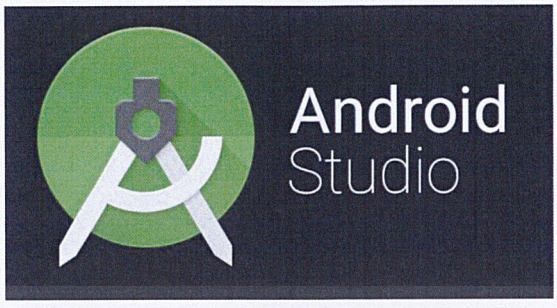


ภาพที่ 2.5 หลักการทำงานโดยของ Design pattern แบบ MVP

(ที่มา: https://medium.com/@beyram.ghali/android-model-view-presenter-mvp-design-pattern)

2.4 เครื่องมือที่ใช้ในการพัฒนา

2.4.1 Android Studio[6]

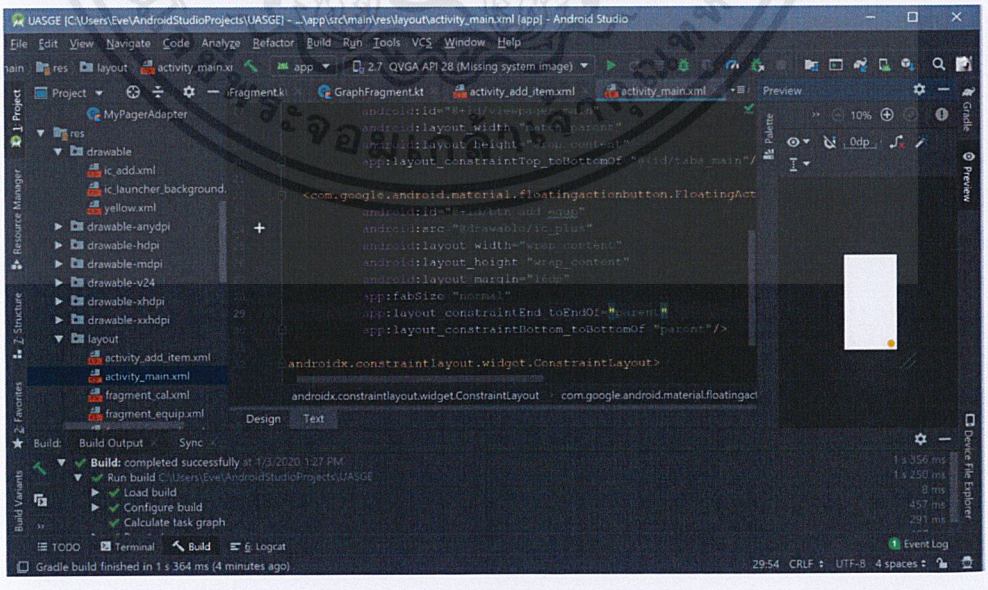


ภาพที่ 2.6 สัญลักษณ์ของ Android Studio

(ที่มา: https://developer.android.com/studio)

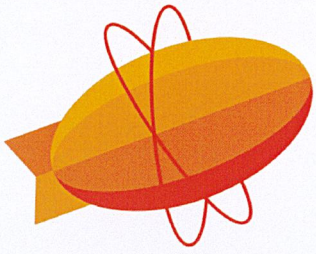
Android Studio เป็น IDE Tool (Integrated Development Environment) จาก Google ไว้สำหรับการพัฒนา Android โดยพัฒนาจากแนวคิดพื้นฐานมาจาก InteliJ IDEA ซึ่งมีการทำงานที่คล้ายกับการทำงานของ Eclipse และ Android ADT Plugin

วัตถุประสงค์ของ การสร้างAndroid Studio คือต้องการพัฒนาเครื่องมือ IDE ที่สามารถพัฒนา แอปพลิเคชัน บน Android ให้มีประสิทธิภาพมากขึ้น ทั้งด้านการออกแบบ GUI ที่ช่วยให้สามารถ Preview ตัว App มุมมองที่แตกต่างกันบน Smart Phone แต่ล่ะรุ่น สามารถแสดงผลบางอย่างได้ทันทีโดนไม่ต้องทำการรัน App บน Emulator รวมทั้งยังแก้ไขปรับปรุงในเรื่องของความเร็วของ Emulator ที่ยังเจอปัญหากันอยู่ในปัจจุบัน



ภาพที่ 2.7 ตัวอย่างภายใน Android Studio

2.4.2 Zeplin[7]

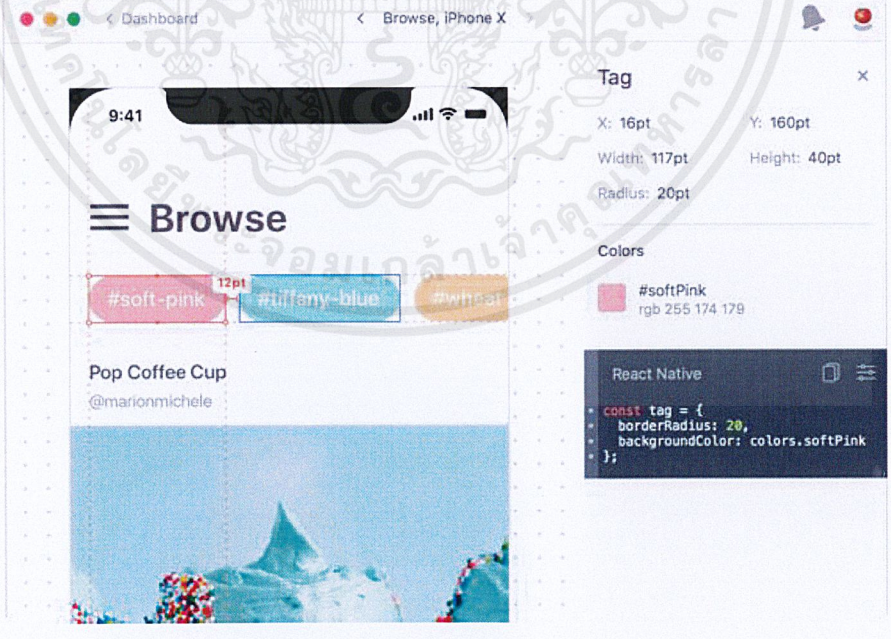


ZEPLIN

ภาพที่ 2.8 สัญลักษณ์ของ Zeplin

(ที่มา: https://zeplin.io/)

Zeplin เป็น tool สำหรับการออกแบบที่ทางทีม Designer ส่วนมากนิยมใช้กัน เนื่องจากสามารถส่งต่อให้กับ Dev ได้ทันที และเข้าใจตรงกัน เนื่องจากใน Zeplin นั้นจะมีการระบุถึงขนาดและระยะห่างต่าง ๆ ไว้ ซึ่งผู้พัฒนา Front-End สามารถทำตามได้เลย รวมถึงการบันทึกรูปภาพที่ใช้ในการพัฒนา เช่น ภาพปุ่มต่าง ๆ ภาพแนะนำการใช้งาน หรือรูปค่าพื้นฐาน (Default)



ภาพที่ 2.9 ตัวอย่างการใช้งานภายใน Zeplin

(ที่มา: https://zeplin.io/why-zeplin)

2.4.3 GitLab[8]

ภาพที่ 2.10 สัญลักษณ์ของ Gitlab

(ที่มา: https://about.gitlab.com/)

GitLab เป็นซอฟต์แวร์ที่พัฒนาขึ้นจาก Git โดยมีหน้าที่สองส่วนคือเก็บ Source Code (repository) และการจัดการโครงการ (CI/CD ย่อมาจาก continuous integration and continuous delivery) ซึ่งเหมาะกับรูปแบบงานที่ใช้ Scrum board จัดการโปรเจกต์ต่าง ๆ

ในส่วนของผู้จัดทำโครงงาน ได้ใช้ GitLab เพื่อการพัฒนาแอปพลิเคชัน โดยเพิ่ม Branch ของตัวเองและ Clone โปรเจกต์มาเพื่อพัฒนาฟีเจอร์ และหลังจากได้พัฒนาจนถึงความต้องการที่พอใจแล้วก็ทำการ Merge Code ที่ได้พัฒนาเพิ่มเข้าไปในตัวโปรเจกต์หลัก โดยในทุกครั้งที่ต้องการ Merge Code ต้องผ่านการทำ Merge Requset ก่อน

การทำ Merge Request คือการแสดง Code ส่วนที่ต้องการ Merge ให้สมาชิกคนอื่นในโปรเจกต์ได้ตรวจสอบก่อนทำการ Merge เพื่อแนะแนวทางที่ดีกว่า หรือการช่วยตรวจสอบความถูกต้องของ Code โดย Code ที่ทำการเพิ่มมา อาจมีทั้งเขียนขึ้นมาใหม่และปรับแก้ของเก่า

ภาพที่ 2.11 ตัวอย่างการทำ Merge Request

(ที่มา: https://forge.etsi.org/rep/help/user/project/merge_requests/index.md)

2.4.4 JIRA[9]

ภาพที่ 2.12 สัญลักษณ์ของ Jira

(ที่มา: https://www.atlassian.com/software)

Jira คือ Proprietary issue tracking tools ที่จะคอยเข้ามาช่วยจัดการ Bug tracking โปรเจกต์ management รวมถึง Plan, Track, Release การติดตามหาความผิดพลาดและปัญหา ที่เกิดจากการพัฒนาและทดสอบซอฟท์แวร์ Jira เป็นผลิตภัณฑ์ของ Atlassian ที่ได้รับการขนานนามว่าเป็นอันดับ 1 Software development tool used by agile teams

โดยในโครงงานนี้ ผู้จัดทำได้ใช้ Jira เพื่อใช้ติดตามและแก้ไข Defect ที่เกิดขึ้น (ในโครงงานนี้ผู้จัดทำจะกล่าวถึงข้อผิดพลาดไม่พึงประสงค์รวมถึง Bug ว่าเป็น Defect) โดย Defect ที่เกิดขึ้นจะถูกส่งมาจากทีม Tester

ภาพที่ 2.13 ตัวอย่างภายใน Jira

(ที่มา: https://www.cloudmunch.com/jira-insights-using-cloudmunch-devops-intelligence-platform-2/)

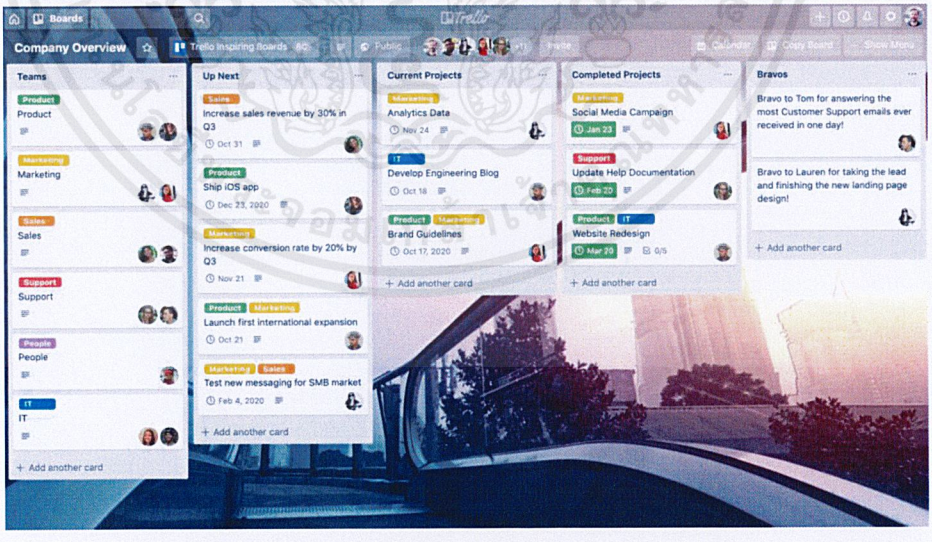
2.4.5 Trello[10]

ภาพที่ 2.14 สัญลักษณ์ของ Trello

(ที่มา: https://trello.com/th)

Trello คือ Collaboration Software ที่มีไว้สำหรับการบริหารจัดการ การทำงานร่วมกันระหว่าง ทีมพัฒนาและลูกค้า หรือทุก ๆ แผนกที่มีส่วนในการพัฒนาและตรวจสอบคุณภาพของงานและ Process การทำงาน โดยใช้งานได้ทั้งผ่านบนหน้าเว็บไซต์หลัก หรือแอปพลิเคชันบทโทรศัพท์มือถือ

ข้อดีของ Trello คือ ทำให้ทั้งฝั่ง Production คือทีมงานผู้พัฒนา เช่น โปรเจกต์ Manager, Designer, Programmer สามารถรู้ความคืบหน้าของโปรเจกต์และทางฝั่ง Client หรือผู้ว่าจ้างโปรเจกต์ สามารถเข้ามาตรวจสอบขั้นตอนการทำงานและระยะเวลาในการพัฒนาได้เช่นกัน ส่งผลให้ มีความเป็น CRM สร้างความเชื่อมั่นให้กับลูกค้าได้ และยังสามารถใช้งานได้ทั้ง Mobile Apps และบน Website



ภาพที่ 2.15 ตัวอย่างการใช้งาน Trello บน Website

(ที่มา: https://blog.trello.com/trello-manage-team)

2.4.6 Jenkins[11]

ภาพที่ 2.16 สัญลักษณ์ของ Jenkins

(ที่มา: https://jenkins.io/)

Jenkins เป็น Open Source Tools ที่ถูกเขียนขึ้นด้วยภาษา Java โดยสามารถสั่งการรันเทส และเก็บผลทดสอบได้ เช่น คอยแจ้งปัญหาเมื่อมีการ Test fail โดยจุดประสงค์ในการสร้าง Jenkins ขึ้นมาคือเพื่อตอบสนอง Concept อย่าง Continuous integration (CI) โดย Continuous Integration คือ Concept ที่เป็นหลักการ ให้ทีมพัฒนาทำการ Integrate Code หรือ การรวมโค้ดซึ่งมีจำนวนมากและกระจัดกระจายอยู่เป็นประจำมีความถี่ที่ต่อเนื่องและอยู่บนรากฐานของคุณภาพ กล่าวคือ CI เป็นกระบวนการที่นำการเปลี่ยนแปลงต่าง ๆ ของ Source Code ไปรวมกันไว้ที่ Version Control System บ่อย ๆ และ อย่างต่อเนื่อง

2.5 Library และ Plugin ที่ใช้ในการพัฒนา

2.5.1 Lottie[12]

ภาพที่ 2.17 สัญลักษณ์ของ Lottie

(ที่มา: https://lottiefiles.com/)

Lottie คือ Library ที่ใช้สำหรับสร้างแอนิเมชันถูกพัฒนาขึ้นโดย Airbnb โดยลักษณะจะเป็นกลายเคลื่อนไหวที่ลักษณะคล้ายไฟล์ GIF แต่ Lottie จะมีการแสดงผลที่ประหยัดหน่วยความจำและหน่วยประมวลผลมากกว่า และมีจุดเด่นคือการนำไปใช้ได้ครอบคลุมทุก

Designer สามารถออกแบบแอนิเมชันใน Adobe After Effects และใช้ Plug-in Bodymovin ทำการ Export ไฟล์ที่สร้างออกมาให้เป็นไฟล์ json และสามารถส่งต่อให้กับทีมผู้พัฒนาได้ทันที

2.5.2 Mockito[13]

ภาพที่ 2.18 สัญลักษณ์ของ Mockito

(ที่มา: https://site.mockito.org/)

Mockito เป็น Framework สำหรับการทำ Unit Test ซึ่งทำให้ผู้พัฒนาสามารถเขียนการทดสอบด้วย Clean & Simple API โดย Mockito จะไม่ทำให้สบสนกับการทดสอบที่เขียนขึ้น และมีการทำงานร่วมกับ JUnit ที่ใช้งานกันอย่างแพร่หลายสำหรับ Java และใช้ง่านร่วมกับ Spring Framework ได้เป็นอย่างดี

2.5.3 Dependency Injection (DI) [14]

DI เป็นเทคนิคการพัฒนา Code รูปแบบหนึ่งที่จะช่วยให้ Code มีความยืดหยุ่นมากขึ้น ไม่ผูกกับคลาสด้วยกันจนเกินไป ซึ่งเป็นหัวใจสำคัญสำหรับการเขียน Code ที่มีโครงสร้างขนาดใหญ่ที่จะต้องมีการดูแลตลอดเวลา เพื่อให้ในท้ายที่สุดสามารถเขียนการทดสอบให้กับ Code เหล่านี้ได้อย่างสะดวก ลักษณะการเขียนในยุคเริ่มแรกก่อนจะมี Dagger คือการส่งข้อมูลที่ต้องการผ่านเข้ามาทาง Constructor ของตัว Main Repository

แต่ในการพัฒนาแอปพลิเคชันหรือฟังก์ชันที่มีขนาดค่อนข้างใหญ่ จะส่งผลให้ตัวโค้ดต้นทางที่ส่งข้อมูลเข้ามาทาง Constructor ได้รับภาระที่หนักจนเกินไปและยากต่อตัวผู้พัฒนาเองเช่นกัน เนื่องจากปริมาณโค้ดที่ต้องเขียนมากขึ้นตามจำนวนข้อมูล

```kotlin
// MainActivity.kt
class MainActivity : AppCompatActivity() {
    private var repository: MainRepository

    init {
        val locationManager = getSystemService(Context.LOCATION_SERVICE) as LocationManager
        val userUtil = UserUtil()
        val serviceUtil = ServiceUtil(this)
        val networkManager = NetworkManager(this, serviceUtil)
        val addressManager = AddressManager(locationManager)
        val userPreferenceManager = UserPreferenceManager(this)
        repository = MainRepository(userPreferenceManager,
                networkManager,
                addressManager,
                userUtil)
    }
    ...
}
```

ภาพที่ 2.19 ตัวอย่างปัญหาการส่งโค้ดปริมาณมากเข้ามาทาง Constructor

(ที่มา: http://www.akexorcist.com/2018/07/beautiful-dependency-inject-in-android-with-dagger-2)

2.5.4 Dagger 2 [15]

จากปัญหาที่เกิดขึ้นใน การใช้ ID (อ้างถึงหัวข้อที่ 2.5.5) Dagger 2 คือ Library ที่ช่วยลดปริมาณโค้ดที่เป็นภาระต่อผู้พัฒนาเหล่านั้น โดยเดิมที Dagger 2 ถูกพัฒนาให้เป็น Java Library เพื่อการสนับสนุนการทำ Dependency Injection ใน Java ให้ง่ายขึ้น ซึ่งส่งผลให้ Android ได้รับผลประโยชน์ไปด้วย แต่ทว่าด้วยความเป็น Android จึงทำให้มีโค้ดบางส่วนที่จำเป็นต้องเขียนไว้อย่างน่าเกลียดเพื่อให้สามารถใช้งาน Dagger 2 ได้ แต่สำหรับเวอร์ชัน 2.10 ขึ้นไป ทีมพัฒนาของ Dagger 2 ก็ได้เพิ่มความสามารถเพื่อให้รองรับกับคลาสหลัก ๆ ของ Android แล้ว ดังนั้นโค้ดจะมีความสวยงามมากขึ้นและมีคำสั่งบางส่วนที่เปลี่ยนแปลงด้วยเช่นกัน

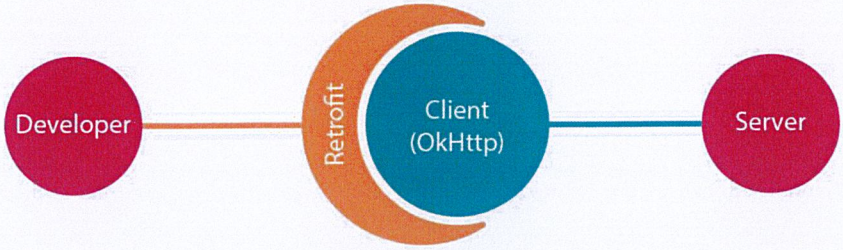
```kotlin
// MainActivity.kt
class MainActivity : AppCompatActivity() {
    @Inject
    lateinit var repository: MainRepository
    ...
}
```

ภาพที่ 2.20 ตัวอย่างโค้ดผลลัพท์ที่ได้หลังการใช้ Dagger 2

(ที่มา: http://www.akexorcist.com/2018/07/beautiful-dependency-inject-in-android-with-dagger-2)

2.5.5 Retrofit[16]

Retrofit เป็น Library สำหรับการใช้ HTTP Client ที่ใช้ติดต่อกับฝั่ง Server สำหรับการดึงและใส่ข้อมูลผ่าน API ที่เป็น Web Service หรือ แบบ Restful API ในแอปพลิเคชันแพลตฟอร์มแอนดรอยด์ โดย Library Retrofit มีการรองรับทั้งภาษา xml และ Json และเป็นที่นิยมในการใช้งานเนื่องจากง่ายต่อการทำความเข้าใจ



ภาพที่ 2.21 หลักการทำงานของ Retrofit 2

(ที่มา: https://medium.com/@anirut.311)

2.6 Software Development Life Cycle (agile) [17]

Agile คือกลุ่มแนวคิดของ Software Methodologies ที่มีลักษณะการพัฒนาแบบ Iteration หรือการวนซ้ำ คือ แบ่งส่วนย่อย ๆ แล้วพัฒนาต่อเรื่อย ๆ แต่ละวงของ Iteration จะต้องเกิด Working Software และต้องพัฒนาต่อ เรื่อย ๆ และแนวคิด Agile มีลักษณะให้ความสำคัญกับคนที่ทำมากกว่า โดย Process Agile มี ใจความสำคัญ (Core Value) ทั้งหมด 4 ข้อ ดังนี้

- **Individuals interactions over process and tool:** คือการให้ความสำคัญกับบุคคลมากกว่า Process และ Tool

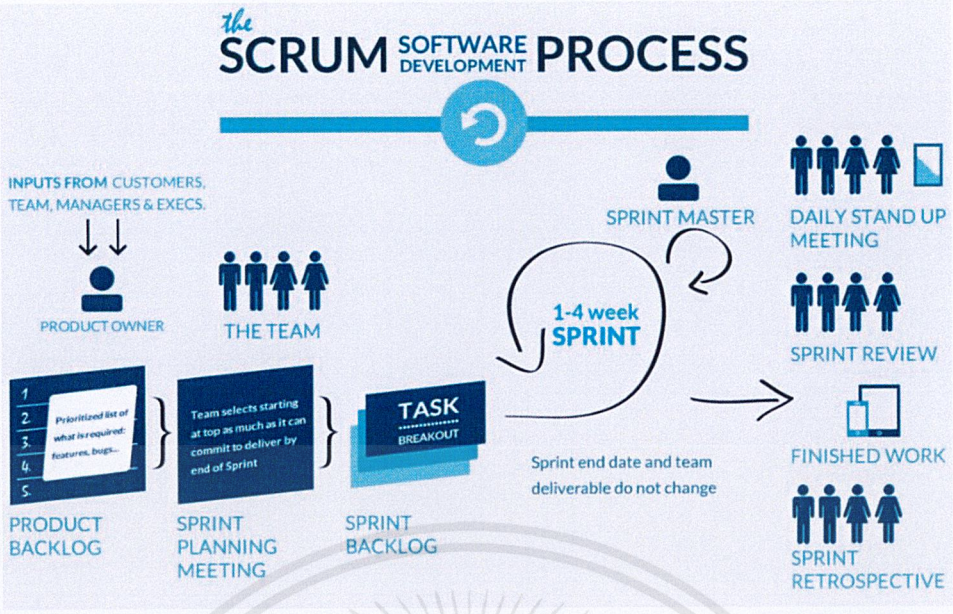
- **Working software over comprehensive documentation:** คือให้ความสำคัญกับการส่งมอบชิ้นงานหรือซอฟต์แวร์ที่นำไปใช้งานได้จริง มากกว่าการทำเอกสารที่ครบสมบูรณ์

- **Customer collaboration over contract negotiation** คือให้ความสำคัญกับการเจอลูกค้าในการพัฒนาตลอดระยะ เวลา มากกว่าการทำตามสัญญา

- **Responding to change over following a plan** คือการยอมรับการเปลี่ยนแปลงมากกว่าการทำแผนที่วางไว้

เพื่อให้แนวคิด Agile เกิดขึ้นได้จริง องค์กรจำเป็นต้องมี Tools หรือ Aglie Method ที่รองรับการทำงานเพื่อให้เกิดประสิทธิภาพสูงสุด โดยในบริษัท KBTG ที่ผู้จัดทำโครงงานได้เข้าร่วมนั้นได้ใช้รูปแบบการทำงานแบบ Scrum

Scrum คือ framework ของ Agile สำหรับการพัฒนา software ชนิดหนึ่ง ที่มีลักษณะ cross-functional teams ซึ่งเป็นกลุ่มของคน ที่ทำให้ software เกิด product ไม่จำเป็นต้องหน้าที่ใครหน้าที่ มัน แต่ใครๆก็สามารถทำได้[18]



ภาพที่ 2.22 ขั้นตอนการทำงานของ Scrum

(ที่มา: http://www.riverpark.co.th/blog/agilesdlc.html)

จากภาพที่ 2.22 Process ของ Scrum โดยย่อ คือ

1. Product backlog คือการทำบอร์ดที่รวบรวม Requirement ทั้งหมดของลูกค้าทั้งหมด

2. Sprint backlog คือบการทำบอรด์ที่รวบรวม Task ของ Iteration นั้นๆ ว่าต้องทำอะไรบ้างใน Sprint ซึ่งนำ Task นั้นๆ มาจาก Product Backlog ซึ่งภายในโครงงานนี้ ผู้จัดทำได้ใช้ Trello เป็น Sprint backlog

3. Sprint คือ การทำ Iteration Development ให้เกิด Working Software ขึ้นมา โดยมีขั้นตอนภายในเพิ่มเติม เช่น

3.1 การทำ daily scrum คือการทำ Stand-Up Meeting หน้า Sprint Backlog อธิบายว่า ใครทำอะไรบ้าง และ ทำไปถึงไหนแล้ว ซึ่งภายในโครงงานนี้ ทีมของผู้จัดทำจะมีการทำ Daily Scrum เพื่ออัปเดตความก้าวหน้าของงานในทุก ๆ เช้าเวลา 9.00 น.

3.2 การกำหนดระยะเวลาของ Sprint เช่น ทุก ๆ 30 วัน หรือ อาจน้อยกว่านี้ได้ตามความเหมาะสม จนเกิด Working Software ซึ่งภายในโครงงานนี้ ผู้จัดทำได้กำหนดเวลาอยู่ที่ 1 Sprint เท่ากับ ช่วงเวลา 2 สัปดาห์

2.7 Data Source

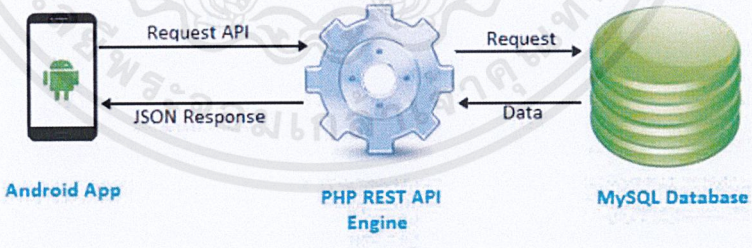
การเขียนแอปพลิเคชันขึ้นมาหนึ่งแอปพลิเคชัน สิ่งที่ขาดไม่ได้เลยก็คือส่วนของการจัดเก็บและเรียกใช้ข้อมูล ไม่ว่าแอปพลิเคชันที่พัฒนาขึ้นนั้นจะเป็นแอปพลิเคชันแบบสามารถต่ออินเทอร์เน็ตได้หรือเป็นแอปพลิเคชันที่ไม่สามารถต่ออินเทอร์เน็ตได้ก็ตาม ก็ล้วนแล้วแต่ต้องการฐานข้อมูลเพื่อการใช้งานทั้งเรียกใช้และจัดเก็บ

ในหัวข้อนี้จะกล่าวถึงสิ่งที่เกี่ยวกับฐานข้อมูลที่ใช้ในการพัฒนาโครงงานนี้หลัก ๆ โดยจะทำการแยกเป็นของฐานข้อมูลที่ใช้ในโครงงานเป็น 2 ประเภทหลัก ๆ คือการเรียกฐานข้อมูลใช้จากภายนอก (Remote) และ การเรียกใช้จากภายใน (Local) ดังนี้

2.7.1 Remote

- RESTful API[19]

Web Service ที่ใช้ REST Architectural Style เป็นที่รู้จักกันในชื่อ RESTful Web Services (RWS). RESTful Web service อนุญาตให้ระบบ Request และเข้าถึง Resource บนเว็บโดยใช้ชุดคำสั่งที่กำหนดเอาไว้ล่วงหน้า โดยที่การโต้ตอบของระบบที่ใช้ REST จะอยู่บนพื้นฐานของ Hypertext Transfer Protocol (HTTP). Request จะส่งคำขอไปยัง URI ที่กำหนด และล้วงเอา Response กลับมาเป็น Payload ในแบบ HTML, XML, JSON หรือ Format อื่น ๆ



ภาพที่ 2.23 ตัวอย่างการติดต่อเรียกใช้ API ของ แอปพลิเคชันแอนดรอยด์

(ที่มา: https://phppot.com/php/php-mysql-rest-api-for-android/)

6 ข้อกำหนดของ RESTful API ซึ่งถือเป็นสิ่งที่สำคัญในการสร้าง RESTful API ตามมาตรฐานซึ่งทำให้ง่ายต่อการพัฒนา และทำให้เป็นที่ยอมรับ (หากไม่ทำตามให้ครบทั้ง 6 ข้อ จะไม่ถือว่าเป็น RESTful API ยกเว้น optional) มีดังนี้

1) Client-server architecture: Client ไม่จำเป็นต้องรู้อะไรเกี่ยวกับ Business logic ภายใน ไม่มีหน้าที่เกี่ยวกับการจัดเก็บข้อมูล ส่วน Server มีหน้าที่เก็บ Resource และไม่จำเป็นต้องรู้อะไรเกี่ยวกับ UI Frontend หรือสถานะของผู้เรียก

2) Statelessness: ส่ง Request รับ Response จาก Server แล้วเลิกทำการติดต่อ

3) Cacheability: สามารถ Cache Response ได้ การ Response จะต้องสามารถกำหนดได้ว่าจะ Cache หรือไม่ เพื่อป้องกันไม่ให้ User หรือ Client ได้รับข้อมูลเก่า หรือท่านสามารถดูเพิ่มเติมเรื่อง Idempotent ได้ที่นี่

4) Layered system: ปกติ Client ไม่รู้ว่าที่ทำการเชื่อมต่อนั้น ได้เชื่อมต่อโดยตรงกับ Server ปลายทาง หรือไปยังตัวกลางอื่น ๆ ระหว่างทาง Server ตัวกลางควรสามารถปรับปรุงความสามารถในการขยายระบบได้ โดยการใช้งานการทำ Load balance

5) Code on demand (optional): Server สามารถขยายได้ชั่วคราว หรือปรับแต่งการทำงานของไคลเอนต์ได้ ตัวอย่างเช่น ทำ Client-Side Scripts ใน Javascript

6) Uniform Interface: ถือเป็นข้อสำคัญจะที่แยกระหว่าง REST API และ Non-REST API ซึ่งแสดงให้เห็นถึงวิธีการที่จะคุยกับ Server โดยไม่คำนึงถึงประเภทของอุปกรณ์ หรือประเภทของ Application และ Uniform Interface ได้แยกออกไปอีก 4 อย่างดังนี้

6.1) Resource-Based: เช่น API/users

6.2) Manipulation of Resources Through Representations: เช่น User get user_id หรือ Request list of users แล้วทำการ Delete หรือ Modify user

6.3) Self-descriptive Messages: แต่ละ Message มีข้อมูลเพียงพอที่จะนำมาอธิบายวิธีการ Process Message เพื่อให้ Server ทำการวิเคราะห์ได้ง่าย

6.4) Hypermedia as the Engine of Application State (HATEOAS): จำเป็น ต้องมี Links สำหรับทุก ๆ Response เพื่อให้ Client สามารถค้นหาได้ง่าย

2.7.2 Local

- Realm Database[20]

Realm Database เป็น Mobile Database มีจุดเด่นคือการใช้ Engine ในการจัดเก็บข้อมูลของตัวเอง จุดเด่นในการทำงานของ Realm เมื่อเทียบกับ SQLite และ Core

Data ที่เป็น Mobile Database เก่าคือการที่มีระบบจัดการและเก็บบันทึกข้อมูลของตัวเอง มีการออกแบบให้ใช้ง่ายสำหรับการพัฒนา เน้นการทำงานที่ว่องไว และสามารถใช้งานข้ามแพล็ตฟอร์มได้ เช่น Java, Swift, React Native และ Realm ยังสามารถรองรับการทำงานขั้นสูง เช่น การเข้ารหัส (Encryption)

การเขียนตัว Models ของ Realm จะอยู่ในรูปแบบเหมือนกับ JavaBeans โดยต้อง Extend RealmObject และใช้ Annotation ใช้การจัดการข้อมูล ดังตัวอย่างภาพที่ 2.24

```
public class User extends RealmObject {

    @PrimaryKey
    private String          name;
    private int             age;

    @Ignore
    private int             sessionId;

    // getters & setters
}
```

ภาพที่ 2.24 ตัวอย่างการเขียน Model ของ Realm

(ที่มา: https://try-droid.blogspot.com/2015/12/android-realm-mobile-database.html)

จากโค้ดในภาพ 2.24 จะเห็นว่ามีการใช้ Annotation เพื่อจัดการข้อมูลซึ่งมีความหมายดังนี้

@Required: ใช้กับ field ที่ห้ามเป็น null

@PrimaryKey: กำหนดให้ Field นั้นเป็น Primary key ซึ่งจะสามารถมีได้เพียง 1 field เท่านั้น (รองรับ Field types ดังต่อไปนี้ String, short, int และ long) และหากกำหนดให้เป็น Primary key แล้วจะถือเป็นการกำหนดให้ Field นั้น @Required ไปในตัว

@Index: เป็นการบอกว่าให้เพิ่ม Search Index ให้ Field นั้น ๆ ซึ่งส่งผลให้การ Insert ช้าลงและข้อมูลมีขนาดใหญ่ขึ้น แต่สามารถ Query ได้เร็วขึ้น (รองรับ field types ดังต่อไปนี้ String, byte, short, int, long, boolean และ Date)

@Ignore: ใช้กับ Field ที่ไม่ต้องการให้เก็บข้อมูลลง Database

- การเขียนข้อมูล (Writes)

การเขียนข้อมูลทำได้ทั้งแบบ Synchronous และ Asynchronous แต่ว่าหากเขียน Synchronous จะเกิดการ Blocking UI Thread เพื่อรอให้เขียนเสร็จ ซึ่งในแง่ของ UX แล้ว

อาจไม่เหมาะสมเสียเท่าไร เนื่องจากแอปพลิเคชันจะเกิดการหยุดค้างเพื่อรอการตอบกลับของข้อมูล ดังนั้นในส่วนการเขียนจึงขอกล่าวถึงการเขียนแบบ Asynchronous

ก่อนจะเขียนข้อมูล ต้องสร้าง Object ก่อนโดยการสร้าง object ทำได้ 2 วิธี คือ

1. ให้ Realm สร้างขึ้นมาโดยตรงจาก class ที่ Extends RealmObject

2. ใช้คำสั่ง copyToRealm() หลังจากที่ new instance object ขึ้นมา (class นั้นต้อง extends RealmObject)

และสามารถเขียนได้ดังภาพที่ 2.25

```
// สร้าง Realm object ขึ้นมาก่อน
Realm realm = Realm.getInstance(this);

// เรียกการใช้งานแบบ asynchronous
realm.executeTransaction(new Realm.Transaction() {
 @Override
 public void execute(Realm realm) {
  User user = realm.createObject(User.class);
  user.setName("John");
  user.setEmail("john@corporation.com");
 }
});
```

ภาพที่ 2.25 ตัวอย่างการเขียนข้อมูลลงใน Realms

(ที่มา: https://try-droid.blogspot.com/2015/12/android-realm-mobile-database.html)

- การอ่านข้อมูล (Queries)

Realm's query อยู่ในรูปแบบของ Fluent interface ซึ่งจะทำให้ง่ายต่อการ query หลายๆ เงื่อนไข ดังภาพที่ 2.26

```
RealmResults<User> result = realm.where(User.class)
                        .equalTo("name", "John")
                        .or()
                        .equalTo("name", "Peter")
                        .findAll();
```

ภาพที่ 2.26 ตัวอย่างการอ่านข้อมูลใน Realms

(ที่มา: https://try-droid.blogspot.com/2015/12/android-realm-mobile-database.html)

- การลบข้อมูล (Remove)

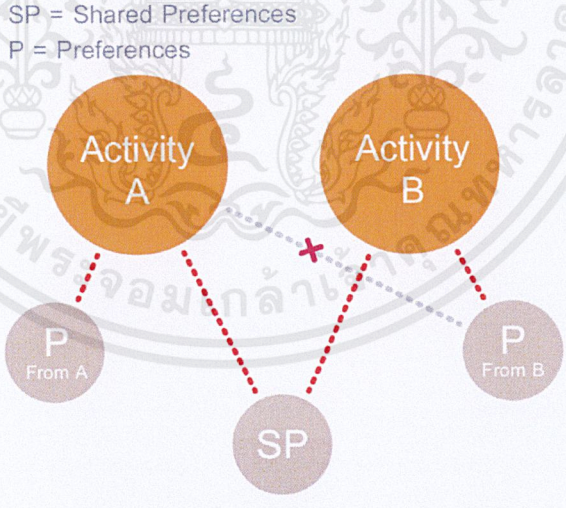
```
// remove single match
result.remove(0);
result.removeLast();

// remove a single object
Dog dog = result.get(5);
dog.removeFromRealm();

// Delete all matches
result.clear();
```

ภาพที่ 2.27 ตัวอย่างการลบข้อมูลใน Realms

(ที่มา: https://try-droid.blogspot.com/2015/12/android-realm-mobile-database.html)

- SharePerference[21]

Shared Preferences เป็นคลาสที่ใช้สำหรับเก็บข้อมูลถาวรที่เป็นค่าของตัวแปรทั่วไป อย่างเช่น Boolean Integer หรือ Float เป็นต้น โดย Shared Preferences จะมีอยู่ด้วยกันสองแบบคือ Shared Preferences กับ Preferences



ภาพที่ 2.28 การดึงข้อมูลระหว่าง Activity

(ที่มา http://www.akexorcist.com/2014/09/android-shared-preferences.html)

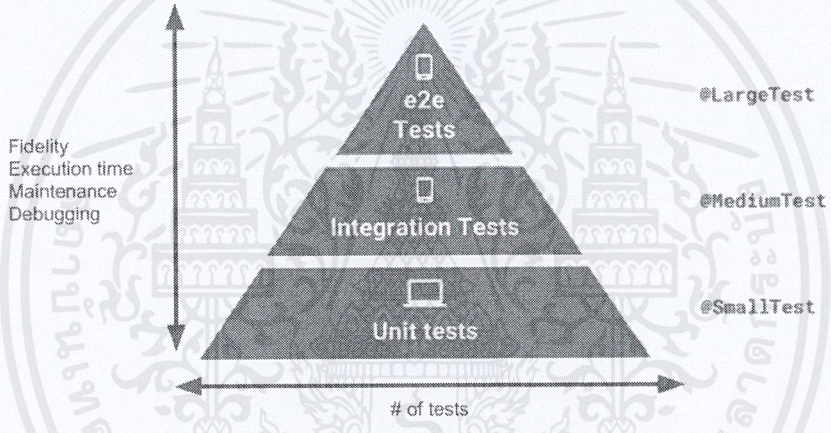
จากภาพที่ 2.28 จะสังเกตได้ว่า Shared Preference จะเป็นการเก็บข้อมูลตัวแปรที่สามารถดึงไปใช้งานที่ไหนก็ได้ภายในแอปพลิเคชันนั้นๆ ดังนั้นถ้าต้องการส่งข้อมูลระหว่าง

Activity ก็สามารถใช้ Shared Preferences ได้เช่นกัน เช่น Activity A เก็บข้อมูลบางอย่างแล้วให้ Activity B ดึงขึ้นมาใช้งาน เป็นต้น

Preferences จะเป็นการเก็บข้อมูลตัวแปรที่ใช้งานได้เฉพาะที่ที่เรียกใช้เท่านั้น เช่น Activity A สร้าง Preferences เก็บค่าบางอย่างไว้ เมื่อ Activity B จะดึงค่าที่ Activity A เก็บไว้ก็จะไม่สามารถทำได้

2.8 การทำ Unit Test[22]

การทำ Unit test คือวิธีการทดสอบ Software อย่างหนึ่งที่ใช้ทดสอบส่วนที่เล็กที่สุด (Unit) ของโค้ดทีละส่วน โดยในเว็บไซต์ของ Android Developer ได้อธิบาย Fundamentals of Testing โดยแบ่งประเภทการ Test แบบต่างๆ โดยใช้ Testing Pyramid ดังภาพที่ 2.29



ภาพที่ 2.29 Testing Pyramid

(ที่มา https://www.youtube.com/watch?v=pK7W5npkhho)

- Unit Test / small test : 70%

เป็นการ Test ในระดับที่ย่อยที่สุด ตามปกติเราจะใช้ JUnit เขียนกัน เขียนเสร็จก็สามารถรันได้บน Android Studio ได้เลย ตอนนี้ก็มี Mockito ซึ่งเป็นการ mock object ขึ้นมาตัวนึงเพื่อ Test สามารถแบ่งแยกย่อยได้อีก 2 แบบ คือ

1) Local Unit Test : <module_name>/src/test/java/

คือการ Test สำหรับ 1 file เช่น file ของ service ที่เรียก API แสดงชื่อศิลปิน โดยจะต้องไม่เชื่อมต่อกับ server จริง กล่าวคือต้อง mock server เพื่อดูว่าหาก Response Code นี้ ผลควรออกเป็นแบบใด

2) Instrumented Unit Test : <module_name>/src/androidTest/java/

ทำการทดสอบการเรียกใช้ Resource ต่าง ๆ บนเครื่องจริงหรือ Emulator โดยใช้ JUnit, Espresso หรือ UI Automator ในการทำ และสร้าง APK file แยกออกมาอีกตัวเพื่อกระทำการนี้โดยเฉพาะ

- Integration Test : medium test : 20%

ใช้ในการทำ service tests, integration tests, hermetic UI และสามารถนำไปทดสอบบน Firebase Test Lab ได้เช่นกัน

- UI Test : large test : 10%

เป็นการทดสอบการทำงานของหน้า UI ของ Activity/Fragment ต่าง ๆ ปกติจะทำแค่ในส่วนที่ใหญ่ที่สุดเพียงอันเดียวโดยใช้ AndroidJUnitRunner, JUnit4 Rule และ Espresso ในการทำ

คุณสมบัติในการทำ Unit test ที่ดีคือ ทดสอบอย่างทั่วถึง (Thorough) สามารถทำซ้ำได้ (Repeatable), มุ่งไปในสิ่งที่เราจะทำ test ว่าควรจะได้ผลอะไร (Focused) เข้าใจพฤติกรรมของตัวโค้ด (Verifies Behaviour) สามารถ test ได้อย่างรวดเร็ว (Fast) และ กระชับ (Concise)

หลักการในการทำ Unit Test คือจะมีการ Compare ค่าสองฝั่ง กล่าวคือค่าที่ยอมรับได้ (expect) หรือค่าที่ควรจะเป็น และ ค่าแท้จริง (actual) ที่ได้รับหลังการทำ test ต้องมีค่าเท่ากัน โดยในการพัฒนาโครงงานนี้ ผู้จัดทำได้ใช้หลักการของการทดสอบระบบงานแบบ BDD style หรือ Behavior-Driven Development ในการทำ Unit test ตามข้อตกลงกลางของทีม ซึ่งจะมีแบ่งส่วนการทำงานออกเป็น 3 ส่วน ได้แก่

- Given สำหรับกำหนด pre-condition ต่าง ๆ ของการทดสอบ เช่นการกำหนดค่าและสถานะต่าง ๆ ของฟังก์ชันที่ทำการทดสอบ

- When สำหรับการกระทำต่าง ๆ ที่ต้องการทดสอบ

- Then ส่วนตรวจสอบผลการทำงาน ว่าตรงตามที่คาดหวังหรือไม่

บทที่ 3
ขั้นตอนและวิธีการดำเนินงาน

โครงงานฉบับนี้เป็นโครงงานที่เน้นการพัฒนาในด้านฟีเจอร์การใช้งานภายในแอปพลิเคชัน K-PLUS ของธนาคารกสิกรไทย ที่ดำเนินงานผ่านองค์กร KBTG ซึ่งเป็นองค์กรเทคโนโลยีขนาดใหญ่ และเพื่อให้งานที่ออกมามีประสิทธิภาพที่ดีที่สุด จึงจำเป็นต้องมีการติดต่อประสานงานกับทีมอื่น ๆ โดยตัวองค์กรมีการนำแนวคิดแบบ Agile มาประยุกต์ใช้ จึงมีการใช้บางส่วนของ Scrum เข้ามาเป็นเครื่องมือ (ในโครงงานฉบับนี้จะเน้นบรรยาย Scrum ในส่วนของผู้พัฒนา) เพื่อระบุปัญหาที่มีความซับซ้อนและเพื่อให้สามารถส่งมอบงานที่ตอบสนองต่อการเปลี่ยนแปลงความต้องการได้อย่างรวดเร็ว โดยมีการกำหนดข้อตกลงว่า 1 Sprint คือระยะเวลา 2 สัปดาห์

หัวข้อของการทำโครงงาน “การพัฒนาแอปพลิเคชันทางการเงินบนระบบปฏิบัติการแอนดรอยด์” มีหัวข้อที่ได้รับมอบหมายให้พัฒนาหลักๆอยู่ 3 หัวข้อคือ

- Voice Command
- Transaction Suggestions
- Credit card 2019 (ประกอบด้วย Installment Plans และ Get Cash)

นอกจากหัวข้อหลักที่ได้รับมอบหมายแล้วผู้ทำโครงงานก็ได้รับหน้าที่ในการซัพพอร์ต (Support) แอปพลิเครชันทั้งการปรับแก้หน้ารูปแบบของหน้า Home การ Refator ตัว Code ของแอปพิลเคชัน การแก้ข้อผิดพลาดที่ไม่พึงประสงค์ (Bug) ในช่วงเวลาที่ไม่มีการรับมอบหมายงานหลัก

หัวข้อ\ ระยะเวลา	ส.ค.				ก.ย.				ต.ค.				พ.ย.			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Voice Command																
Transaction Suggestions																
Refactor + Support																
Credit card 2019																

ตารางที่ 3.1 ระยะเวลาการทำงานในแต่ละหัวข้อ

3.1 รวบรวมความต้องการ

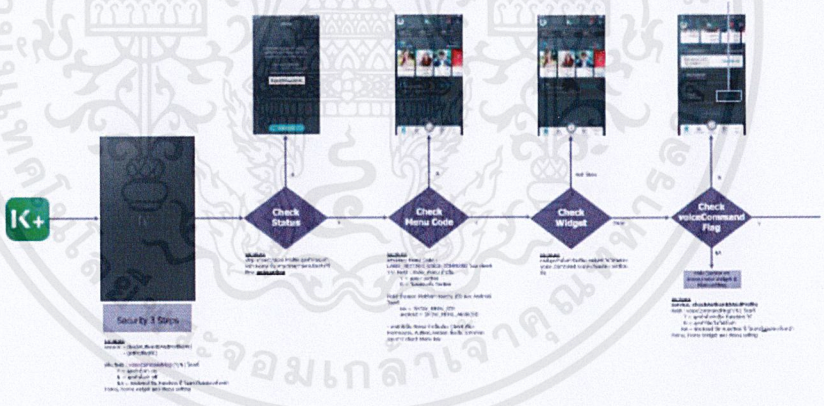
ทางผู้ทำโครงงานที่มีบทบาทเป็น Front-end Client Developer จึงจำเป็นต้องรับฟังความต้องการ (Requirement) จากหลาย ๆ ฝ่าย และประสานงานเพื่อรายงานความคืบหน้าของโครงงานที่จำเป็นต่อการพัฒนาแอปพลิเคชัน สำหรับโครงงานนี้ผู้จัดทำได้ทำการแบ่งประเภทของความต้องการในการพัฒนาออกเป็น 2 ส่วน ดังนี้

3.1.1 การรวบรวมความต้องการจากทีม/ฝ่ายต่าง ๆ

- **ฝ่าย BA (Business Analyst)**

ก่อนที่จะเริ่มทำการพัฒนาตัวฟีเจอร์ใหม่ ๆ จะมี BA หรือ Business Analyst ซึ่งมีหน้าที่คอยรับและรวบรวม Requirement จากลูกค้าเพื่อนำมากลั่นกรองความเป็นไปได้และสรุปออกมาเป็นเอกสารในรูปแบบของ Flow ให้แก่ตัวนักพัฒนา

ภายใน Flow จะมีบอกถึงสิ่งที่ต้องมีในฟีเจอร์นั้น ๆ เช่น fact ที่ต้องตรวจสอบ, ตำแหน่งของหน้าการตรวจสอบได้รับอนุญาต (Permisssion) รูปแบบการส่งหรือรับข้อมูล, Action จากการกดปุ่มหรือการตอบสนองของตัวแอปพลิเคชันต่อUser



ภาพที่ 3.1 ตัวอย่าง Screen Flow ที่ได้รับจากทีม BA

- **ทีม Designer**

ในบางกรณีที่การรับ Requirement หรือการประสานงานจากเพียงแค่ BA อาจยังเป็นการรับข้อมูลที่ไม่เพียงพอ เนื่องจากการพัฒนาต่อยอดจากทรัพยากรเดิมอาจไม่รองรับต่อการเปลี่ยนแปลง เช่น การเปลี่ยนหน้าตาฟังก์ชันการใช้งานบางอย่าง หรือการเปลี่ยนรูปแบบการจัดวาง ประกอบกับการมีกำหนดระยะเวลาที่แน่นอนในหัวข้อนั้น ๆ มาแล้ว

จากโครงสร้างภายในของแอปพลิเคชันที่มีการเชื่อมต่อถึงกันเป็นส่วนใหญ่ ทำให้เมื่อต้องการพัฒนาต่อยอดในส่วนที่ส่งผลกระทบต่อส่วนอื่นอาจเกินระยะเวลาที่จำกัดไว้ ทำให้ต้องพูดคุยเจรจาต่อรองกับทาง Designer เพื่อหาทางออกที่ลงตัวแก่ทั้งสองฝ่าย

บทบาทหน้าที่ของ Designer คือการออกแบบหน้าตาของฟีเจอร์ใหม่ หรือ User Interface ที่จำเป็นต้องคำนึงถึงความง่ายในการใช้งาน User และต้องสอดคล้องกับดีไซน์เดิมของตัวแอปพลิเคชัน โดย Designer จะทำการส่งมอบ UI ให้แก่ผู้พัฒนาผ่านทาง Zeplin ดังที่อธิบายไว้ในหัวข้อ 2.4.2

- ทีม Back-End

ทีม Back-End หรือทีมหลังบ้าน มีหน้าที่ในการจัดเตรียมเซอร์วิส (Service) เพื่อให้ฝั่ง Client สามารถเรียกใช้งานได้ง่าย พร้อมเอกสารการเชื่อมต่อเซอร์วิส (API - Document) โดยภายในเอกสารระบุถึงความต้องการของเซอร์วิสหรือสเปค (Specification) ว่าตัวต้องการรับพารามิตเตอร์ (Request) ชนิดใดและตัวเซอร์วิสจะมีการตอบสนอง (Respone) กลับมาเป็นอะไร

ดังนั้นแล้วเพื่อการทำงานอย่างมีประสิทธิภาพสูงสุดต่อทุกฝ่าย ฝั่ง Client จำเป็นต้องคุยกับทางฝั่ง Back-end เพื่อสรุปเรื่องแผนการและรูปแบบการดำเนินงาน เช่น สรุป Request / Responeของแต่ละเซอร์วิสโดยเฉพาะการคุยทางเชิงเฉพาะทางที่ BA ไม่สามารถให้ความช่วยเหลือได้

- ทีม ML (Machine Learning)

โดยปกติแล้วฝั่งของ Front-End Client จะไม่ได้ทำการติดต่อกับทีม ML บ่อยมากนัก เนื่องจากการใช้งาน Machine learning จะมีเพียงในบางฟีเจอร์เท่านั้นและหากมีการใช้งานจริง ทีม ML จะติดต่อประสานงานผ่านทางทีม Back-End เอง แต่สำหรับโครงงานครั้งนี้ผู้จัดทำโครงงานได้มีโอกาสได้ติดต่อประสานงานกับทีม ML จากฟีเจอร์ Voice command ที่จะกล่าวถึงในหัวข้อต่อ ๆ ไป

- ทีม Tester

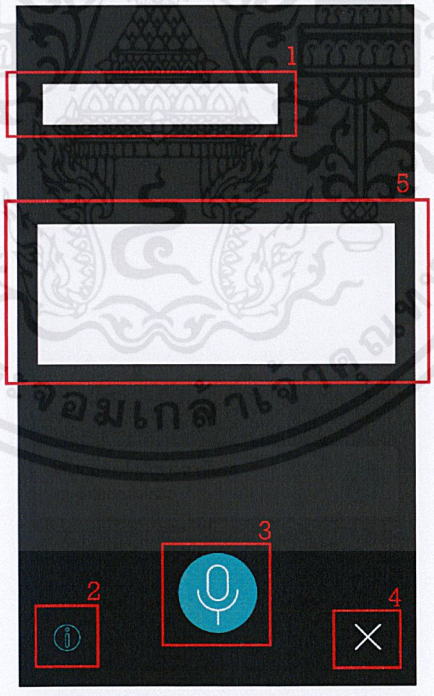
ทีม Tester คือทีมที่คอยตรวจสอบรายละเอียดชิ้นงานที่ทำส่งก่อนนำขึ้น Product จริง ทั้งในส่วนของเงื่อนไขการแสดงผล และรายละเอียดในส่วนยิบย่อยต่าง ๆ ที่ผู้พัฒนาอาจดูแลไม่ทั่วถึง โดยหากมีข้อผิดพลาด ทีม Tester จะแจ้งมาทาง Trello (อ้างถึงหัวข้อ 2.4.5)

3.1.2 สรุปความต้องการที่ได้รับ

- Voice Command

จากความต้องการที่ได้รวบรวมมาสำหรับฟีเจอร์ voice command สรุปได้ว่าฟีเจอร์นี้มีไว้สำหรับการทำรายการ โอน เติม จ่าย กับบัญชีที่เพิ่มเป็นรายการโปรดไว้แล้ว และสามารถเช็กยอดเงินในบัญชีและบัตรเครดิตได้ โดยมีช่องทางการใช้งานผ่าน Widget บนหน้า Home สำหรับการใช้งานในครั้งแรกจำเป็นต้องให้ Widget บนหน้า Home พาไปหน้า Setting และทำการเปิดการใช้งานจากหน้านั้น

การใช้งาน Voice Command จะอยู่บน Transparent Dialog ดังภาพที่ 3.2 โดยมีปุ่มสำคัญบนหน้า Dialog 3 ปุ่มคือ ปุ่มแนะนำการใช้งาน บริเวณล่างซ้าย ดังภาพกรอบหมายเลข 2 ของภาพที่ 3.2 ปุ่มไมโครโฟนกลางล่างของ Dialog ดังกรอบหมายเลข 3 ปุ่มกดปิดที่มุมล่างขวาดังกรอบหมายเลข 4 โดยปุ่มแนะนำการใช้งานจะไม่แสดงเมื่อมีการอัดเสียงอยู่หรือประมวลผล และมีพื้นที่ดังกรอบหมายเลข 1 ของภาพที่ 3.2 เป็นพื้นที่การบอกสถานะปัจจุบัน และมีกรอบหมายเลข 5 ในการแสดงผลเพิ่มเติม



ภาพที่ 3.2 ตำแหน่งขององค์ประกอบหลักในหน้า Voice command

ทุกครั้งที่มีการพูดบริเวณรอบปุ่มไมโครโฟนจะมี Animation เป็นวงกระเพื่อมตามเสียงพูด หากมีการพูดเสียงเบา วงกระเพื่อมจะแคบ 3 ระดับตามภาพ Animation ที่ได้รับ และเมื่อเสียงที่ได้รับดังขึ้นวงก็จะกระเพื่อมจะกว้างขึ้นตามน้ำหนักเสียง

นอกจากนี้ยังมีข้อกำหนดเพิ่มเติมเกี่ยวกับลิมิตความยาวของข้อความ ได้แก่ ข้อกำหนดความยาวของเสียงไม่ให้เกินจำนวนวินาทีที่กำหนดไว้ ถ้ามีเสียงหากเกินให้ทำการตัดข้อความทันที หรือหากไม่มีเสียงในช่วงกี่วินาทีแรกให้ทำการแสดงหน้าแนะนำการใช้งานขึ้นมา และหลังจากเงียบไป กี่วินาทีต่อมาหลังจากหยุดพูด ให้ทำการตัดไฟล์เสียงและส่งให้ Sever ทำการวิเคราะห์ทันที

โดยหลังจากที่ได้รับไฟล์เสียงมาแล้ว ให้ทำการส่งต่อไปยังระบบหลังบ้าน (Back End) เพื่อทำการแกะคำพูดและประมวลผลว่าผู้ใช้งานต้องการทำรายการใด โอน เติม จ่าย ให้ใคร จำนวนเท่าไหร หรือการเช็กยอดเงินบัญชีใด หลังจากประมวลผลเสร็จแล้วจะส่งข้อมูลกลับมาเพื่อแสดงผลและให้พาไปยังหน้าทำรายการ หรือพาไปที่หน้า Home Banking เพื่อทำการอ่านยอดเงินในบัญชีที่ต้องการ โดยหากมีการพบรายชื่อใกล้เคียงกัน 2 รายชื่อขึ้นไป จะมีการแสดงชื่อที่ใกล้เคียงกัน 2 รายชื่อนั้นให้ผู้ใช้เป็นคนเลือกก่อนทำรายการ

หากในระหว่างประมวลผลผู้ใช้ต้องการยกเลิก ก็ต้องสามารถกดซ้ำที่ปุ่มไมโครโฟนและทำการสั่งคำสั่งใหม่ได้ ทั้งนี้ในทุกขั้นตอนหากมีข้อความแสดงสถานะการทำงาน เช่น กำลังรอคำสั่ง ไฟล์เสียงผิดพลาด กำลังประมวลผล ก็ให้ทำการอ่านข้อความนั้นๆด้วย

- Transaction Suggestions

ฟีเจอร์ Transaction Suggestions คือฟีเจอร์ที่จะช่วยแนะนำรายชื่อที่เจ้าของบัญชีทำรายการใช้งานบ่อยที่สุดในช่วง ๆ หนึ่ง ไม่ว่าจะเป็นบุคคลธรรมดา บริษัท หรือหน่วยงาน เช่น การไฟฟ้านครหลวง เพื่อให้ผู้ใช้งานสามารถกดทำรายการได้โดยไม่ต้องเพิ่มเป็นรายการโปรด หรือกดพิมพ์เลขบัญชี

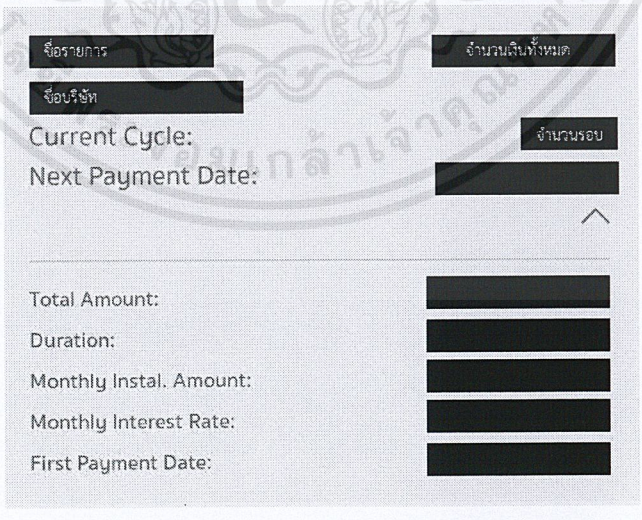
สำหรับการใช้งานฟีเจอร์นี้สามารถใช้งานได้ผ่าน Widget บนหน้า Home โดยจะเริ่มแนะนำรายชื่อใช้บ่อยเมื่อบัญชีนั้นมีการทำรายการกับบัญชีอื่น ๆ ซึ่งรายชื่อที่แนะนำขึ้นมานั้นจะไม่สามารถแก้ไขได้ แต่จะเปลี่ยนไปตามช่วงเวลาที่กำหนด มีความต้องการที่ได้รับและรวบรวมมาคือ ฟีเจอร์นี้จะแสดงข้อมูลมากสุด 7 รายการ โดยหากรายการใช้บ่อยเป็นบุคคลธรรมดา จะทำการแสดงรูปเป็นบุคคลธรรมดา หากเป็นการจ่ายบิลจะแสดงภาพขององค์กรที่ต้องจ่ายบิล และทุก ๆ รูปจะมีข้อความและชื่อเจ้าของบัญชี หรือองค์กรนั้น ๆ แสดงขึ้นมา

- Credit card 2019

Credit card 2019 เป็นชื่อเรียกรวมของการพัฒนาฟีเจอร์ในหมวดบัตรเครดิตที่มีการเพิ่มฟีเจอร์ใหม่เข้ามา 2 ฟีเจอร์ ได้แก่ Installment plans และ Get Cash เพื่อให้ลูกค้ามีการใช้งานที่ง่ายขึ้น

ฟีเจอร์ Installment plans เป็นฟีเจอร์การผ่อนชำระบัตรเครดิต โดยความต้องการที่ได้รับและรวบรวมมาสำหรับระบบ Front-end มีดังต่อไปนี้

ในขั้นตอนแรกผู้ใช้งานจะสามารถเลือกยอดค้างชำระที่ต้องการผ่อน (สามารถเลือกยอดค้างชำระได้หลายยอด) และเลือกแผนการผ่อนชำระได้ (สามารถเลือกได้แผนเดียว) โดยหากเลือกแผนการชำระแล้ว จะมีการแสดงยอดค้างชำระทั้งหมดที่เลือก แสดงยอดค้างชำระที่ผ่านการคำนวณอัตราดอกเบี้ยในแผนที่เลือกไว้ แสดงอัตราดอกเบี้ย แสดงระยะเวลาการผ่อนทั้งหมด แสดงวันเริ่มต้นการผ่อนและวันสิ้นสุดการผ่อน โดยหากมีการชำระตามช่วงเวลาที่กำหนดจะมีการแสดงประวัติไว้ในหน้าประวัติการชำระ (History) โดยในหน้าประวัติจะมีการแสดงยอดค้างชำระทั้งหมด ที่บริเวณด้านบนของหน้า และมี Item การชำระที่ต้องมีรายละเอียดของ รอบการชำระ รอบการชำระเงินครั้งต่อไป ระยะเวลาในการชำระ จำนวนเงินที่ต้องชำระต่อเดือน จำนวนดอกเบี้ย และวันชำระเงินครั้งแรก ดังภาพที่ 3.3 และ Item จะมีการเรียกมาแสดงทีละ 10 รายการ หากมีประวัติรายการที่มากกว่า 10 จะทำการโหลดเพิ่มหลังจากสไลด์จนถึง Item อันสุดท้าย และ ปรากฏหมายเหตุ (note) ที่ด้านล่างสุด



ภาพที่ 3.3 ตัวอย่างข้อมูลที่ต้องแสดงในหน้าประวัติการชำระ

ฟีเจอร์ Get Cash คือฟีเจอร์การโอนเงินเข้าบัญชีของตัวเองที่ต้องการ โดยตัดจากยอดวงเงินของบัตรเครดิตที่ได้ระบุไว้ มีจุดประสงค์คือต้องการให้ผู้ใช้งานสามารถใช้วงเงิน

บัตรเครดิตเป็นเงินในบัญชีได้และถอดเป็นเงินสดได้ เพื่อก้าวข้ามข้อจำกัดว่าการชำระเงินด้วยวงเงินบัตรเครดิตจะชำระได้เพียบร้านค้าที่รับบัตรเครดิตเท่านั้น

โดยในฟีเจอร์นี้มีความต้องการว่า หลังจากเลือกบัตรเครดิตแล้วสามารถเลือกได้ว่าต้องการโอนเงินจากบัตรเครดิตเข้าไปที่บัญชีใด สามารถกรอกจำนวนเงินที่ต้องการได้โดยต้องไม่เกินยอดวงเงินบัตรเครดิตที่เลือกและอยู่ระหว่างจำนวนเงินที่บริษัทกำหนด ต้องมีการใช้ e-mail เพื่อส่งเอกสารยืนยัน และจะสามารถไปสู่หน้าต่อไปก็ต่อเมื่อกดยอมรับในเงื่อนไขและข้อตกลงแล้ว ซึ่งก่อนระบบจะทำการโอนเงินเข้าสู่บัญชีจะมีให้เลือกแผนการผ่อนชำระสำหรับค่าใช้จ่ายนี้ เช่นเดียวกันกับในส่วนของ Get Cash

หากต้องการตรวจสอบประวัติการใช้ฟีเจอร์ Get Cash สามารถดูได้ที่หน้าประวัติเช่นด้วยกับ Installment plans โดยส่วนที่ต่างกันคือการจัดลำดับความสำคัญของข้อมูลที่จะแสดง และมีการเพิ่มส่วนของค่าธรรมเนียมเข้ามา (Fee)

3.2 วางแผนการพัฒนา

เนื่องจากหัวข้อโครงงานที่ได้รับในการทำโปรเจกต์สหกิจครั้งนี้มีความหลากหลาย และมีขั้นตอนการพัฒนาที่ต่างกันไปในแต่ละหัวข้อ ผู้จัดทำจึงแยกเป็นข้อย่อยตามหัวข้อที่ได้รับดังนี้

3.2.1 Voice Command

จากความต้องการทั้งหมดที่ได้รับมาจะเห็นได้ว่าเป็นฟีเจอร์ที่มีความซับซ้อนและจำเป็นต้องประสานงานกับหลายฝ่าย ทำให้ทางผู้จัดทำโครงงานได้จัดการวางแผนและกำหนดระยะเวลาการทำงานเป็นเวลา 3 Sprint หรือ 6 สัปดาห์ดังตารางที่ 3.2

หัวข้อ \ ระยะเวลา	ส.ค.				ก.ย.			
	1	2	3	4	1	2	3	4
ประชุมงาน		■	■	■	■	■		
ศึกษาความต้องการ	■			■				
วางแผนและออกแบบ	■		■		■			
พัฒนาฟีเจอร์	■	■	■	■	■	■		
ทดสอบการใช้งาน		■		■		■		
นำไปใช้งานจริง								

ตารางที่ 3.2 แผนการพัฒนาฟีเจอร์ Voice Command

3.2.2 Transaction Suggestions

และในการแสดงรายชื่อนั้นจะไม่มีการใช้หลักตรรกะ (Logic) ในการประมวลผลก่อนแสดงมากนัก การพัฒนาฟีเจอร์นี้จึงสามารถพัฒนาได้ในเวลาสั้น ๆ ซึ่งได้วางแผนไว้เป็นเวลา 1 Sprint หรือ 2 สัปดาห์ ดังตารางที่ 3.3

หัวข้อ \ ระยะเวลา	ก.ย.				ต.ค.			
	1	2	3	4	1	2	3	4
ประชุมงาน								
ศึกษาความต้องการ								
วางแผนและออกแบบ								
พัฒนาฟีเจอร์								
ทดสอบการใช้งาน								
นำไปใช้งานจริง								

ตารางที่ 3.3 แผนการพัฒนาฟีเจอร์ Transaction Suggestions

3.2.3 Credit card 2019

สำหรับในฟีเจอร์นี้ ผู้จัดทำได้มีส่วนร่วมพัฒนาในหน้าการแสดงประวัติของทั้ง Installment plans และ Get Cash จึงได้มีการวางแผนการทำงานไว้ที่ 2 Sprint หรือ 4 สัปดาห์ดังตารางที่ 3.4

หัวข้อ \ ระยะเวลา	ต.ค.				พ.ย.			
	1	2	3	4	1	2	3	4
ประชุมงาน								
ศึกษาความต้องการ								
วางแผนและออกแบบ								
พัฒนาฟีเจอร์								
ทดสอบการใช้งาน								
นำไปใช้งานจริง								

ตารางที่ 3.4 แผนการพัฒนาฟีเจอร์ Installment plans

3.3 เริ่มพัฒนา

การเริ่มพัฒนาในที่นี้จะกล่าวถึงลำดับขั้นตอนการพัฒนาฟีเจอร์หรือการเขียนโค้ด (Coding) ซึ่งประกอบไปด้วยขั้นตอนสำคัญในการ Code แต่ละส่วน รูปแบบการเขียน Code และเนื้อหาที่เกี่ยวข้องซึ่งจะเชื่อมโยงกับบทที่ 2 โดยจะไม่เน้นการนำ Code ที่ใช้เขียนจริงภายในแอปพลิเคชันจริงมาแสดงเนื่องด้วยนโยบายของบริษัท และสำหรับโค้ดส่วนที่นำมาใส่ในรายงานฉบับนี้จะเป็น open source และโค้ดส่วนที่เปิดเผยได้

3.3.1 Voice Command

ในส่วนที่ยากสำหรับการพัฒนาฟีเจอร์ Voice Command คือการสร้างเงื่อนไขในการแสดงผลต่าง ๆ ให้ตรงกับความต้องการ ซึ่งมีหลากหลายกรณีเพื่อให้มีผลลัพธ์เหมือนกับความต้องการที่ได้รับมา ดังหัวข้อ 3.2.1

- Widget

ขั้นตอนในการพัฒนาฟีเจอร์นี้เริ่มจากการสร้าง Widget บนหน้า Home ซึ่งจะมี 2 รูปแบบตามความต้องการที่ได้รับ ได้แก่ Widget ที่ยังไม่เปิดใช้งาน Voice Command และ Widget ที่เปิดใช้งาน Voice Command แล้ว สามารถทำได้โดยการปรับใช้ปุ่มและองค์ประกอบที่มีอยู่เดิมภายในแอปพลิเคชันมาประกอบและจัดวางให้ผลลัพธ์เหมือนกับความต้องการที่ได้รับมา

- Layout

การพัฒนาต่อมาภายใน Layout ซึ่งเป็นแบบ Transparent Layout ทำการจัดวางองค์ประกอบหลัก ๆ ให้ครบและใส่เงื่อนไขการแสดง ปุ่มต่าง ๆ และ Animation เช่น ปุ่มแนะนำข้อมูลการใช้งาน ที่จะไม่แสดงขึ้นมาเมื่อมีการบันทึกเสียงอยู่ และใช้ Text-To-Speech (TTS) ในการอ่านข้อความให้เป็นเสียง

- การบันทึกเสียง

สำหรับการบันทึกเสียงจำเป็นต้องได้รับการอนุญาต (Permissions) จากเจ้าของเครื่องก่อนทำการบันทึกเสียง โดยใช้คำสั่งดังภาพที่ 3.4 ในไฟล์ AndroidManifest.xml และใช้ MediaRecorder ซึ่งเป็น open class ส่วนหนึ่งของ Library android.media ในการบันทึกเสียงเป็นไฟล์สกุล mp3 และใช้ Realm Database (อ้างถึงหัวข้อ 2.6.2) ในการบันทึกที่อยู่ไฟล์ภายในเครื่อง (Path) และใช้ที่อยู่นั้นในการอัปโหลดไฟล์เสียงไปยัง Server

```
1  <uses-permission android:name="android.permission.RECORD_AUDIO"/>
2  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

ภาพที่ 3.4 คำสั่งขอ Permissions เข้าถึงการบันทึกเสียง

- Animation

Animation ที่ใช้งานในฟีเจอร์นี้จะเป็นการใช้ Animation ของ Lottie Library ที่กล่าวถึงไว้ในหัวข้อ 2.5.1 และมีการใช้งานทั้งหมด 4 จุด ได้แก่

1) ระหว่างทำการบันทึกเสียง ที่จะแสดงวงกระเพื่อม 3 ระดับตามเสียงที่ได้รับ

ในส่วนนี้มีต้องใช้เงื่อนไขในการแสดงผล ซึ่งทำได้ในระหว่างทำการบันทึกเสียงโดยการวัดระดับความดังทุก ๆ 0.5 วินาทีแล้วนำเข้าสู่เงื่อนไขว่า หากมีระดับช่วงความดังอยู่ในช่วงเดียวกันกับเมื่อ 0.5 วินาทีที่แล้ว วงกระเพื่อมจะไม่มีการเปลี่ยนขนาด แต่หากมีช่วงเสียงที่ต่างไปจะทำการเปลี่ยนขนาดของวงตามช่วง

2) ระหว่างการรอประมวลผลจะมีวง Loading ขึ้นมาโดยรอบปุ่มไมโครโฟน

ใช้การเช็กสถานะว่ากำลังส่งข้อความเสียงไปประมวลผลและยังไม่ได้รับข้อมูลกลับมาเพื่อแสดง Animation โดยรอบปุ่มไมโครโฟน ดังเช่นภาพที่ 3.5

ภาพที่ 3.5 ตัวอย่าง Animation ระหว่างรอโหลด

3) Animation ของหน้าข้อแนะนำในการใช้งาน

Animation การแสดงข้อมูลแนะนำการใช้งาน จะเป็นการสไลด์ข้อความทั้งหมดขึ้นมาจากด้านล่างของจอ โดยใช้หลักการว่าให้ตั้งค่า Default ของตำแหน่งข้อมูลไว้ด้านล่างจากนั้นเมื่อต้องการใช้งานจะสามารถดึงขึ้นมาจากข้างล่างได้ทันที

4) การแสดงรายการโปรดที่คล้ายกัน

หากมีรายชื่อที่คล้ายกันระบบจะทำการแสดงรายการที่คล้ายกันมากที่สุด 2 อันขึ้นมาโดยเป็นแบบจางแล้วค่อย ๆ เข้มขึ้น (Fade-in) และให้ค่อย ๆ จางหายไป (Fade-out) เมื่อใช้การใช้งานเสร็จสิ้น

- การรับส่งข้อมูลกับทาง Server

การส่งข้อมูลใช้การอัปโหลดข้อความเสียงส่งไปยังหลังบ้านเพื่อวิเคราะห์ว่าเป็นคำสั่งชนิดใด และในการรับข้อมูลจะนำมาใส่ Data class ที่เตรียมไว้ และตรวจสอบตาม Field ของข้อมูลที่ได้รับ เช่น ตรวจสอบว่าเป็นคำสั่ง Type ใด โอน เติม จ่าย หรือ เช็กยอดเงิน จากนั้นก็จะสามารถพาผู้ใช้งานไปที่หน้าทำรายการได้

3.3.2 Transaction Suggestions

สำหรับการพัฒนาในฟีเจอร์นี้ ผู้จัดทำโครงงานได้รับหน้าที่ของการเปิดโอนเงินและหน้าการเติมเงินหลังจากการกดเลือกรายชื่อแนะนำ

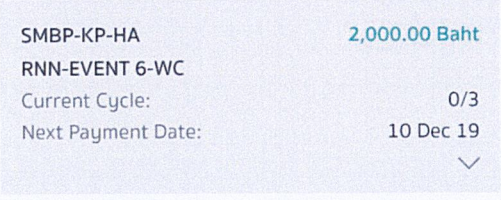
การใช้งานหน้าโอนเงินและเติมเงิน สามารถทำได้โดยใช้ตัวโค้ดที่มีอยู่แล้ว มาพัฒนาต่อยอด โดยมีความยากของการพัฒนาคือการใช้งานและศึกษาโค้ดที่มีอยู่ เนื่องจากหน้าการโอนเงินมีผู้พัฒนาหลายคน และเป็นโค้ดภาษา Java ที่ถูกใช้ในหลายฟีเจอร์ทั่วทั้งแอป หากมีการเปลี่ยนแปลงแล้วเกิดผลกระทบจะเป็นเรื่องที่ไม่พึงประสงค์ จึงต้องมีความระมัดระวังและต้องทดสอบอย่างรอบคอบ

3.3.3 Credit card 2019 (ประกอบไปด้วย Installment Plan และ Get Cash)

หน้าที่ที่ได้รับคือการช่วยพัฒนาหน้าดูประวัติการชำระเงินของทั้งสองฟีเจอร์ และมีการพัฒนาโดยใช้ Recyclerview เป็นตัวพื้นฐานหลัก

- Item ของประวัติการชำระ

เนื่องจากทั้ง 2 ฟีเจอร์นั้นภายใน 1 Item มีการรับข้อมูลจำนวนมากมาแสดงในรูปแบบที่คล้ายกัน จึงเลือกทำ Layout แบบ Dynamic ซึ่งจะเพิ่มความยืดหยุ่นในการประยุกต์ใช้ได้ดีกว่า นอกจากนี้จากความต้องการที่ได้รับมามีการระบุถึงความต้องการที่จะให้ใน 1 Item สามารถกดขยายเพื่อดูข้อมูลเพิ่มเติมได้ ในส่วนนี้สามารถพัฒนาโดยการใช้โค้ดที่มีอยู่ในตัวแอปพลิเคชันมาใช้งานได้



ภาพที่ 3.6 ตัวอย่าง Item ของประวัติการชำระใน Installment Plan

- การโหลดข้อมูลจาก Sever

ทุกครั้งที่มีการโหลดข้อมูล จะเป็นการโหลดไม่เกิน 10 Item ต่อครั้ง เช่น มี 5 Item ก็จะทำการแสดงขึ้นมา 7 และแสดงหมายเหตุที่ด้านล่างเพื่อเป็นการบอกจุดสิ้นสุด แต่หากมี Item ทั้งหมด 15 Item ระบบจะทำการดาวน์โหลด 10 Item มาแสดงก่อน และหากผู้ใช้งานได้เลื่อนลงสุดใน Item ที่ 10 แล้วระบบจะแสดง Progress Dialog ขึ้นมาแทนหมายเหตุระหว่างโหลดข้อมูลอีก 5 Item

ภาพที่ 3.7 ตัวอย่างของหมายเหตุที่จะแสดงเมื่อสิ้นสุดการแสดงข้อมูล

- การเก็บข้อมูล

ทุกครั้งที่มีการดาวน์โหลดข้อมูลใหม่ผ่านอินเทอร์เน็ต ระบบจะทำการบันทึกข้อมูลลงในฐานข้อมูล Local (อ้างถึงหัวข้อที่ 2.6.2) เพื่อความรวดเร็วในการแสดงผลข้อมูล ทำให้เมื่อมีการเปิดใช้งานหน้าประวัติการชำระและข้อมูลไม่มีการเพิ่มขึ้นมา จะไม่มีการดาวน์โหลดข้อมูลมาแสดง แต่หากตรวจพบจำนวนรายการที่ไม่เท่ากับที่บันทึกไว้ ก็จะทำการโหลดข้อมูลอีกครั้งตามเงื่อนไขที่กดหนด

3.4 ทดสอบการใช้งาน

เนื่องจากโครงสร้างของแอปพลิเคชัน เป็นชนิดแยกพัฒนาระหว่าง Android และ iOS ทำให้ต้องมีการทดสอบและเปรียบเทียบการใช้งานกันระหว่างทั้ง 2 ระบบปฏิบัติการอยู่เสมอ เพื่อให้แอปพลิเคชันผลลัพธ์มีรูปแบบการใช้งานที่เหมือนกันที่สุด โดยเมื่อพัฒนาถึงจุดหนึ่ง ๆ ที่ตกลงกันไว้แล้วจะมีการทำการทดสอบภายใน (Internal Test) หรือคือการนำโทรศัพท์สมาร์ตโฟนที่มีระบบปฏิบัติการ Android และ iOS มาทดลองใช้งานในฟีเจอร์นั้น ๆ พร้อมกัน เพื่อทดสอบหาจุดแตกต่างและแก้ไขก่อนส่งมอบงานจริง เพื่อลดภาระการแก้ Defect ที่จะเกิดขึ้น

โดยมีตัวอย่างการปรับแก้ เช่น เวลาในการแสดงผลที่อาจไม่เท่ากัน รูปแบบ Animation ที่ปรากฏขึ้นมา พื้นที่การกดปุ่มที่ไม่เท่ากัน ฯลฯ

3.5 ส่งมอบงาน

ในการส่งมอบงานคือการ Deploy ฟีเจอร์ที่ได้พัฒนาแล้วลงสู่แอปพลิเคชันทดลองเพื่อให้ทีม Tester ได้ทำการตรวจสอบและหาข้อผิดพลาด จากการลองใช้งานจริงและการทดสอบตามกรณี (Case) ต่าง ๆ ที่ได้กำหนดไว้ โดยมีการใช้งาน Jenkins (อ้างถึงหัวข้อ 2.4.6) เป็นเครื่องมือในการ Deploy โดยในโครงงานนี้ผู้จัดทำไม่มีหน้าที่ในการ Deploy ผ่าน Jenkins จึงไม่สามารถบอกเล่าถึงขั้นตอนอย่างละเอียดได้

และก่อนทำการรวมโค้ดของตนเข้าตัวโค้ดหลักของทีม ต้องผ่านการตรวจสอบโค้ด (Code Review) จากสมาชิกภายในทีมและทีมผู้ดูแลเวอร์ชั่นนั้น ๆ ผ่าน Gitlab (อ้างถึงหัวข้อ 2.4.3) ด้วยฟีเจอร์การ Mesh Request

หลังจาก Deploy ขึ้นพบข้อผิดพลาดที่ไม่พึงประสงค์ใด ๆ จะทำการใช้ JIRA (อ้างถึงหัวข้อ 2.4.4) เป็นเครื่องมือในการแจ้งปัญหา ในรูปแบบของการ เปิด Defect มาให้แก่ผู้พัฒนา โดยหากข้อผิดพลาดนั้นเป็นความเข้าใจผิดหรือเป็นข้อผิดพลาดที่ยอมรับได้ ก็สามารถกดปฏิเสธได้

และทุกครั้งที่ได้ส่งมอบงานแล้ว ผู้พัฒนาจะต้องอัปเดตข้อมูลของตนเองใน Trello (อ้างถึงหัวข้อ 2.4.5) เพื่อแจ้งแก่สมาชิกในทีมว่างานที่ตนได้รับมอบหมายสำเร็จลุล่วงแล้ว

บทที่ 4
ผลการดำเนินโครงงาน

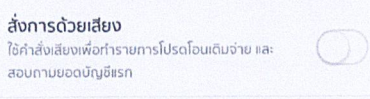
4.1 Voice Command

หน้าเริ่มต้นการใช้งาน Voice command จะอยู่ที่หน้าแรกเมื่อเปิดแอปพลิเคชันขึ้นมา

ภาพที่ 4.1 พร้อมใช้งานของฟีเจอร์ Voice command

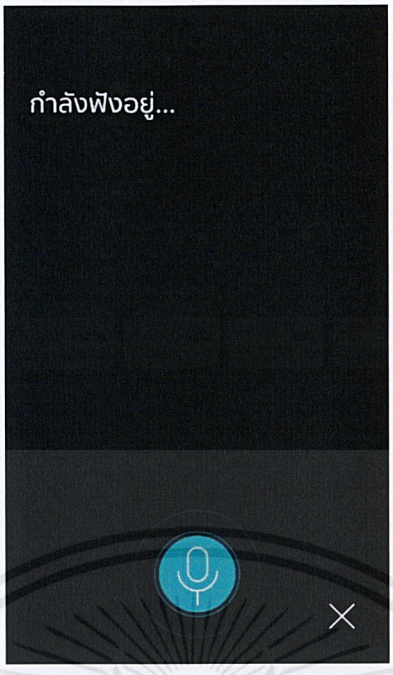
หากยังไม่เปิดการใช้งาน Widgets ที่แสดงจะเป็นดังภาพที่ 4.2 และเมื่อกดปุ่ม “เปิดการใช้งาน” แล้วจะพาไปยังหน้าตั้งค่าเพื่อให้เปิดการใช้งานระบบสั่งการด้วยเสียง ดังภาพที่ 4.3

ภาพที่ 4.2 Widgets Voice Command หากยังไม่เปิดการใช้งาน



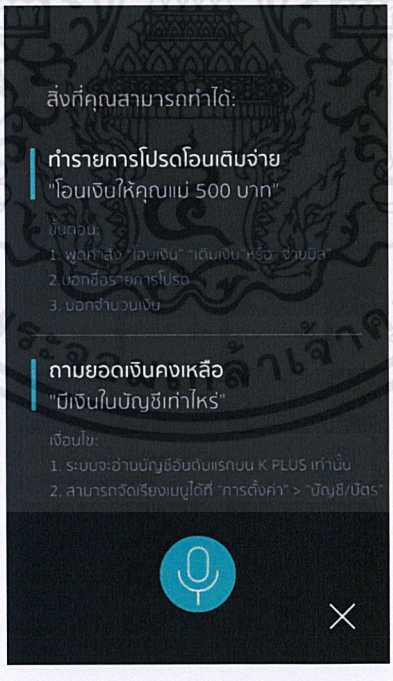
ภาพที่ 4.3 การเปิดใช้งานฟีเจอร์คำสั่งเสียงในหน้า Setting

เมื่อเปิดการใช้งานฟีเจอร์แล้วและกดปุ่มรูปไมโครโฟน จะปรากฏหน้ารับคำสั่งขึ้นมา และเริ่มฟังเสียงทันที ดังภาพที่ 4.4 โดยมี Animation กระเพื่อมตามความดัง-เบาของเสียง 3 ระดับ



ภาพที่ 4.4 หน้าต่างรับคำสั่งเสียง

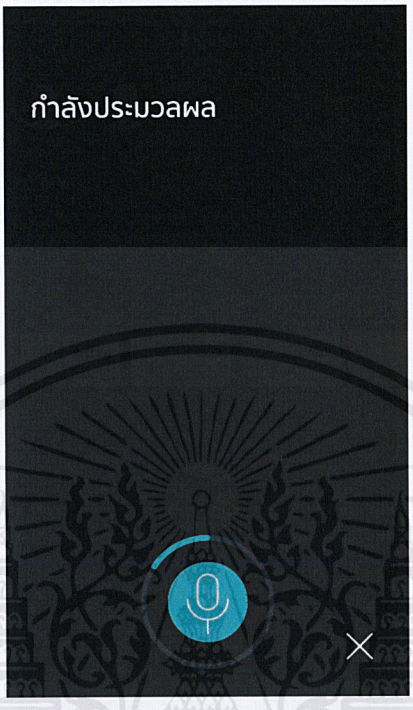
หากเปิดหน้าต่างรับคำสั่งแล้ว ไม่มีเสียงเข้ามาหรือไม่สามารถจับข้อมูลเสียงได้ ภายในช่วงวินาทีที่กำหนด จะมีข้อแนะนำการใช้งานเบื้องต้นปรากฏขึ้นมาดังภาพที่ 4.5



ภาพที่ 4.5 ข้อความแนะนำการใช้งานเบื้องต้น

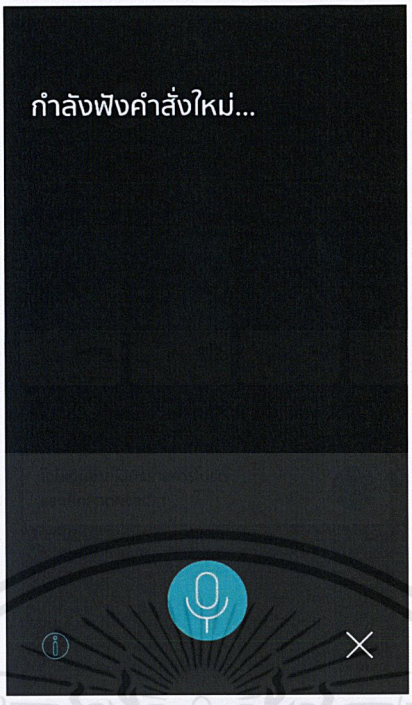
หากพูดคำสั่งจบแล้ว สามารถกดปุ่มไมโครโฟนเพื่อหยุดการบันทึกเสียงได้ จากนั้นระบบจะทำการส่งความข้อเสียงไปวิเคราะห์บน Sever เพื่อจำแนกชนิดของคำสั่ง โดยระหว่างการประมวลผล

จะมีหน้าแจ้งสถานะขึ้นมา ดังภาพที่ 4.6 และแสดง Animation Loading ขึ้นมารอบปุ่มไมโครโฟน จนกว่าจะได้รับข้อมูลกลับมา และหากไม่ระบบไม่สามาถจับเสียงต่อได้เกินจำนวนวินาทีที่กำหนด ก็จะหยุดการบันทึกเสียงและส่งความข้อเสียงไปวิเคราะห์บน Sever ได้เช่นเดียวกัน



ภาพที่ 4.6 หน้าต่างแสดงสถานะกำลังประมวลผล

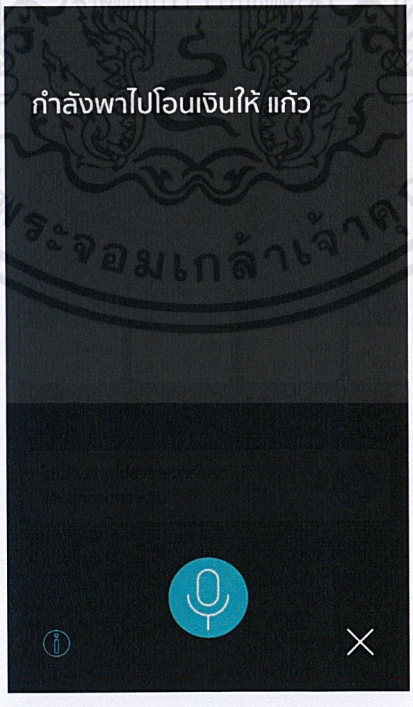
หากต้องการยกเลิกคำสั่งเสียงระหว่างประมวลผล สามารถทำได้โดยการกดที่ปุ่มไมโครโฟน จากนั้นจะปรากฏหน้ารอฟังคำสั่งใหม่ขึ้นมาดังภาพที่ 4.7 และในหน้ารอฟังคำสั่งนี้ จะปรากฏปุ่มตัว i ขึ้นมา หมายถึงการแสดงข้อมูลคำแนะนำการใช้งานเบื้องต้น ดังภาพที่ 4.5 และเมื่อกดปุ่มไมโครโฟนตรงกลางอีกครั้ง ก็จะสามารถเริ่มพูดคำสั่งใหม่ได้ทันที



ภาพที่ 4.7 หน้าต่างแสดงสถานะรอฟังคำสั่งใหม่

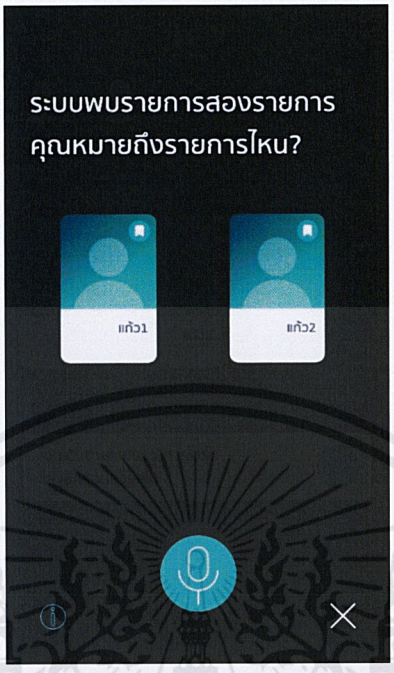
ในที่นี้จะยกตัวอย่างการโอนเงินให้รายการโปรดที่ชื่อ “แก้ว” โดยการพูดว่า “โอนเงินให้แก้ว 200 บาท” เมื่อระบบทำการประมวลผลและส่งข้อมูลกลับมาแล้ว จะมี 3 กรณีดังนี้

- **พบรายการโปรด 1 รายการ:** จะปรากฏหน้าต่างดังภาพที่ 4.8 ก่อนพาไปทำรายการตามคำสั่ง



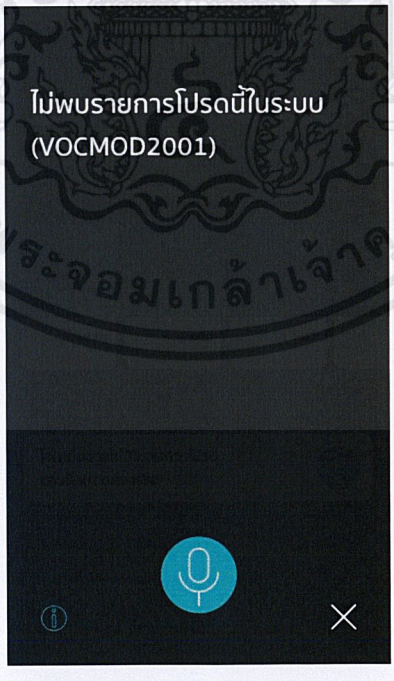
ภาพที่ 4.8 ตัวอย่างหน้าต่างแสดงสถานะหลังประมวลผลแล้ว

- **พบรายการโปรดที่มีชื่อคล้ายกัน 2 รายการขึ้นไป:** จะปรากฏหน้าต่างดังภาพที่ 4.9 และให้เลือกรายการที่ต้องการ



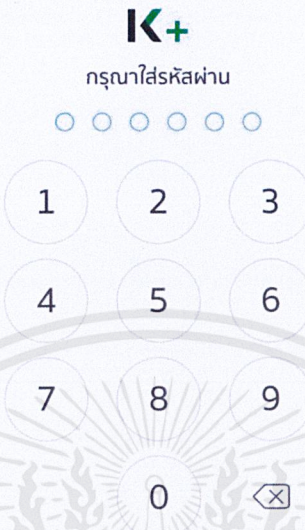
ภาพที่ 4.9 ตัวอย่างหน้าต่างแสดงสถานะพบรายการโปรดที่มีชื่อคล้ายกัน 2 รายการขึ้นไป

- **ไม่พบรายการโปรดที่มีชื่อตรงกัน:** จะปรากฏหน้าต่างดังภาพที่ 4.10



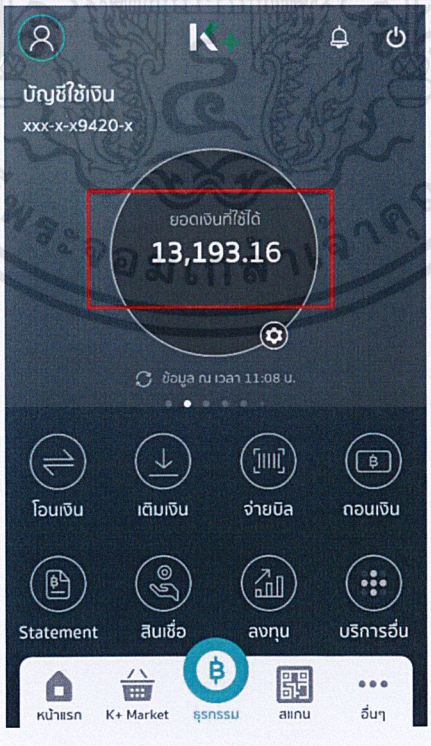
ภาพที่ 4.10 ตัวอย่างหน้าต่างแสดงสถานะไม่พบรายการโปรดที่มีชื่อตรงกัน

เมื่อรับคำสั่งเสียงและประมวลผลแล้ว ระบบจะพาไปสู่หน้าทำรายการ โอน-เติม-จ่าย ตามปกติ ซึ่งหลังผ่านหน้าแสดงสถานะหลังประมวลผลดังภาพที่ 4.8 แล้วหากยังไม่ได้ยืนยันตัวตน ระบบจะทำการพาไปยังหน้าใส่ Pin เพื่อยืนยันตัวตน ดังภาพที่ 4.11



ภาพที่ 4.11 หน้ายืนยันตัวตน

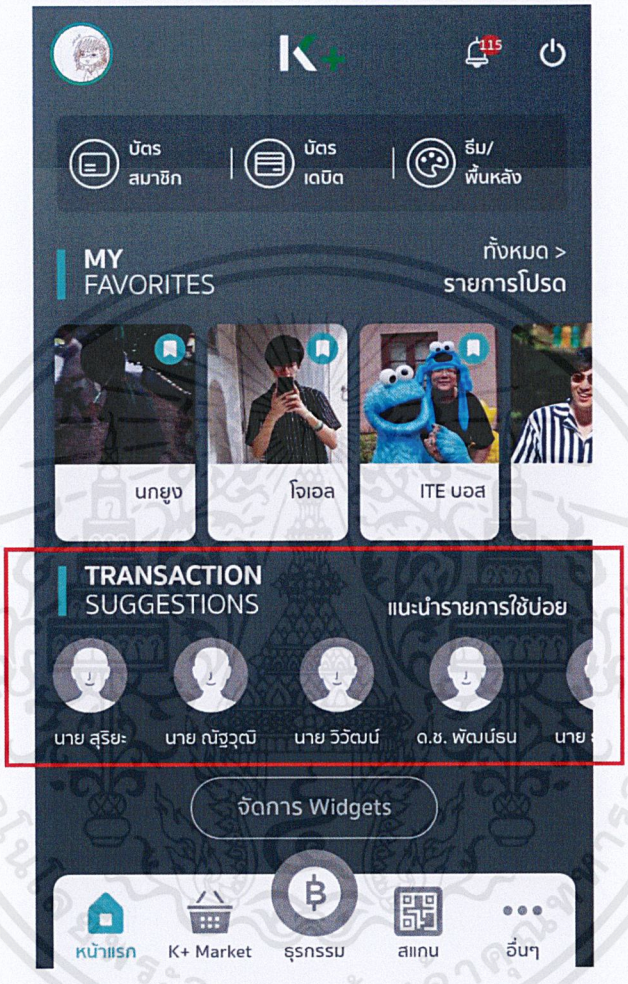
สำหรับการเช็กยอดเงินในบัญชี จะเป็นการพาไปยังหน้าธุรกรรมหลัก และอ่านจำนวนยอดเงินคงเหลือให้ ดังภาพที่ 4.12 ในบริเวณกรอบสีแดง



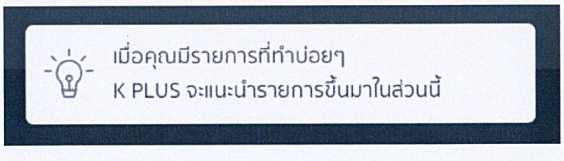
ภาพที่ 4.12 หน้าธุรกรรมหลัก

4.2 Transaction Suggestions

การใช้งาน Transaction Suggestions จะอยู่ที่หน้าแรกเมื่อเปิดแอปพลิเคชันขึ้นมาในบริเวณกรอบสีแดงของภาพที่ 4.13 และในกรณีที่มี Transaction ในบัญชีไม่พอจะแนะนำ ระบบจะแสดง Widgets ดังภาพที่ 4.14



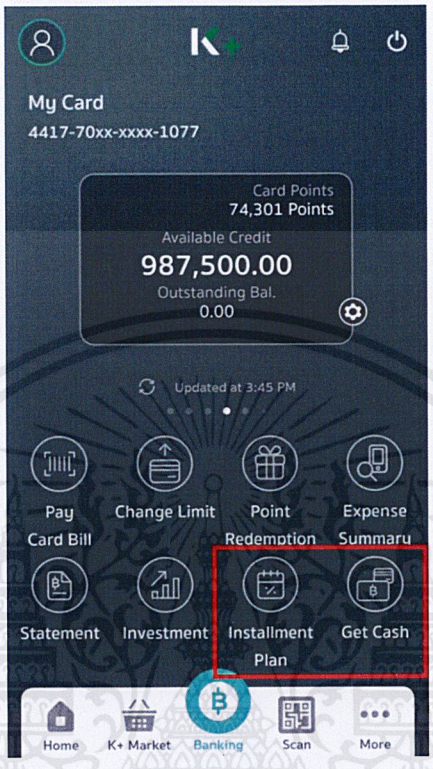
ภาพที่ 4.13 การใช้งาน Transaction Suggestions



ภาพที่ 4.14 Widgets ที่จะแสดงเมื่อบัญชีมี Transaction ไม่เพียงพอที่จะแนะนำ

4.3 Credit card 2019

สามารถเข้าใช้งานได้ที่หน้าธุรกรรมหลัก และเลือกหน้าบัตรเครดิตจะมีทางเข้าสู้ฟีเจอร์ 2 ตัว ดังในกรอบสีภาพแดงของภาพที่ 4.15 ซึ่งจะแยกแสดงผลลัพธ์การทำงานในหัวข้อย่อยต่อไปนี้



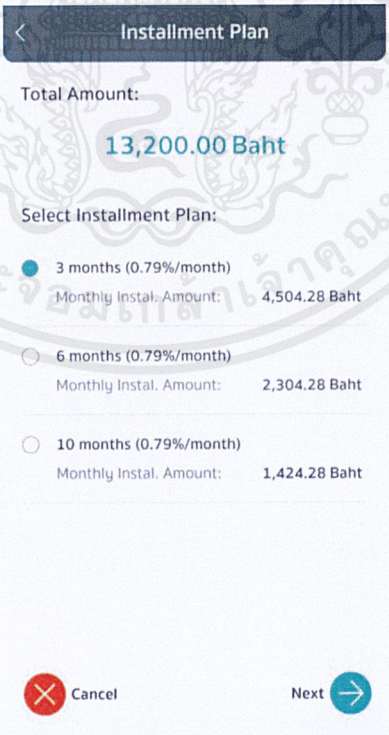
ภาพที่ 4.15 ไอคอนทางเข้าสู่ฟีเจอร์

4.3.1 Installment Plan

เมื่อเข้ามาหน้าแรก จะมีรายการค้างชำระแสดงขึ้นมาดังภาพ 4.16 (ฝั่งซ้าย) และสามารถเลือกรายการที่ต้องการได้ดังภาพ 4.16 (ฝั่งขวา) โดยจะมีการแสดงยอดรวมของรายการที่เลือกไว้ที่ด้านล่าง

ภาพที่ 4.16 แสดงยอดค้างชำระ

เมื่อทำการกดปุ่ม Next ระบบจะนำทางไปที่หน้าเลือกแผนการชำระเงิน โดยจะมีการแสดงดอกเบี้ยต่อเดือนของแผนที่เลือก และยอดที่ต้องชำระต่อเดือน ดังภาพที่ 4.17

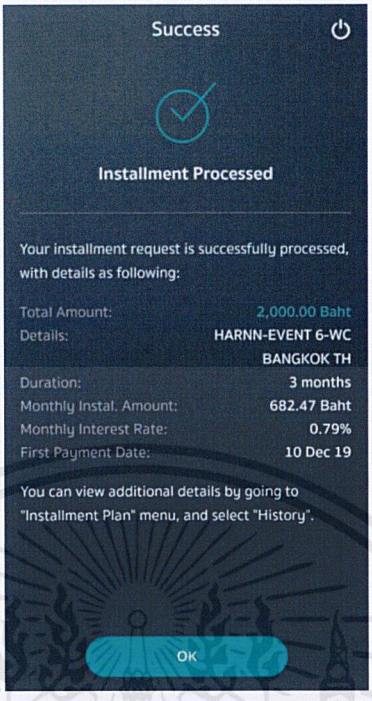


ภาพที่ 4.17 หน้าการเลือกแผนการชำระ

เมื่อทำการเลือกแผนการชำระแล้ว ระบบจะพาไปสู่หน้ายืนยันการเลือกแผนการชำระ โดยจะต้องกรอกอีเมลเพื่อส่งเอกสารให้ผู้ทำรายการ และกดยอมรับข้อตกลงจึงจะสามารถ ยืนยันการชำระเงินได้ ดังภาพที่ 4.18

ภาพที่ 4.18 หน้ายืนยันการเลือกแผนการชำระ

เมื่อจัดยืนยันการชำระแล้ว จะมีหน้าการเลือกแผนการชำระสำเร็จขึ้นมาดังภาพที่ 4.19



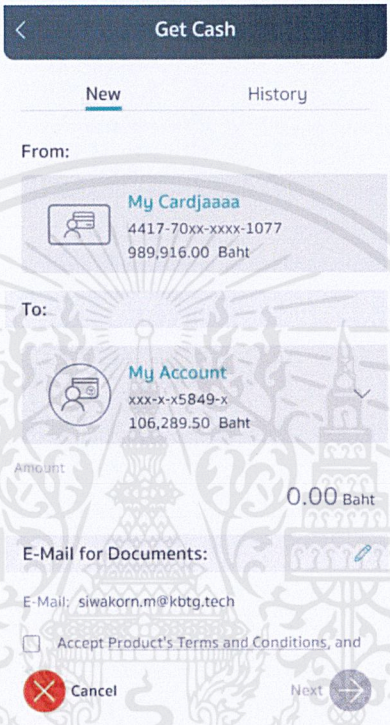
ภาพที่ 4.19 การเลือกแผนการชำระสำเร็จ

หากต้องการตรวจสอบประวัติการชำระเงิน สามารถตรวจสอบได้ในหน้าประวัติ (History) ดังภาพที่ 4.20 (ฝั่งซ้าย) และขยายเพื่อดูข้อมูลเพิ่มเติมได้ดังภาพ 4.20 (ฝั่งขวา)

ภาพที่ 4.20 หน้าแสดงประวัติการชำระ

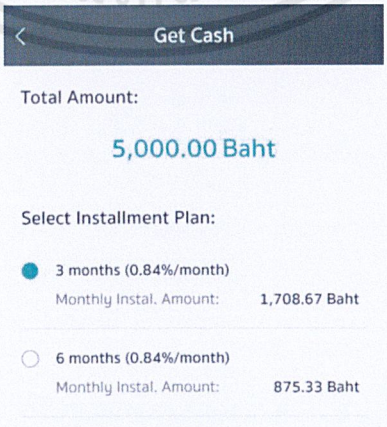
4.3.2 Get Cash

หน้าแรกของฟีเจอร์ Get Cash คือหน้าสร้างรายการใหม่จะสามารถเลือก Account ที่ต้องการโอนและจำนวนเงินได้ และเช่นเดียวกับฟีเจอร์ Installment Plan เพื่อที่จะไปสู่หน้าต่อไป ฟีเจอร์ Get Cash ก็จำเป็นต้องกรอกอีเมลเพื่อส่งเอกสารและกดยอมรับข้อตกลงเช่นเดียวกัน ดังภาพที่ 4.21



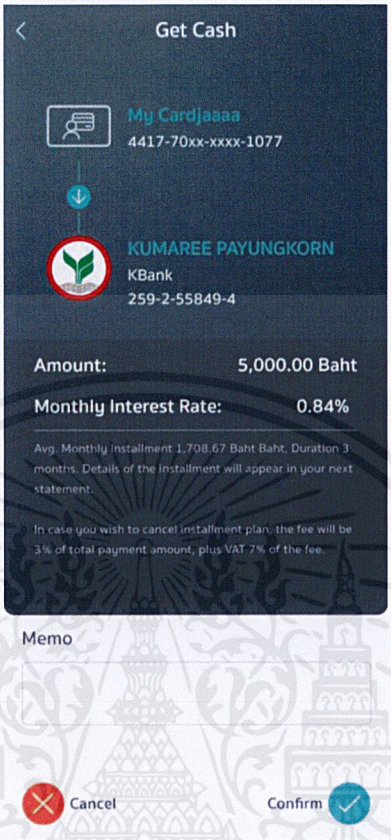
ภาพที่ 4.21 หน้าสร้างรายการใหม่ของ Get Cash

หน้าถัดมาคือการเลือกแผนการผ่อนชำระ ที่มีรูปแบบเหมือน Installment Plan

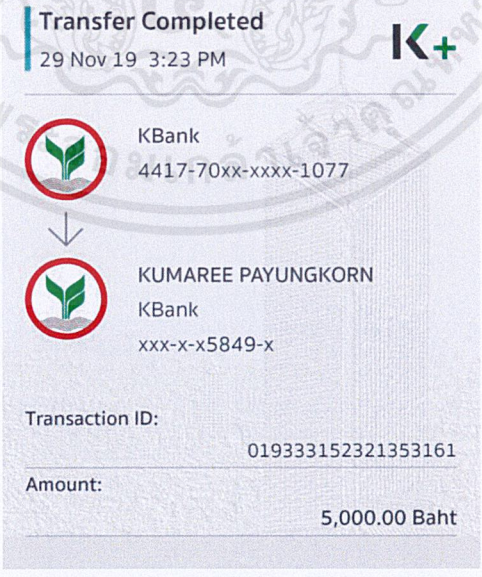


ภาพที่ 4.22 หน้าเลือกแผนผ่อนชำระของ Get cash

ต่อมาหลังเลือกแผนการชำระเงินแล้ว จะเป็นหน้ายืนยันการโอนดังภาพที่ 4.23 และจะได้รับสลิปใบเสร็จการโอนเงินดังภาพที่ 4.24

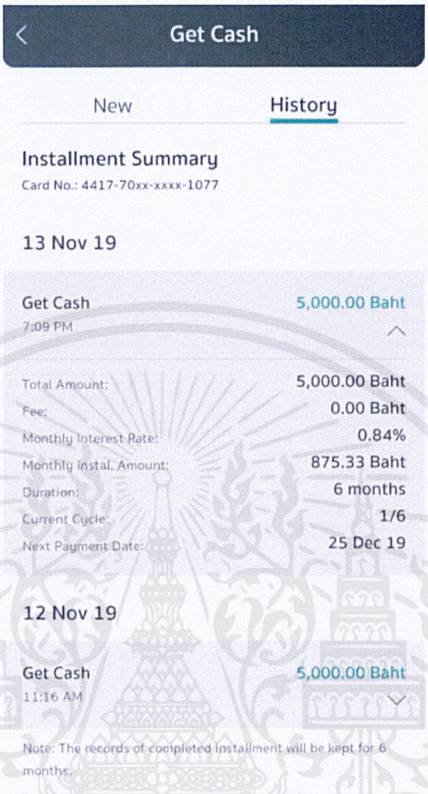


ภาพที่ 4.23 หน้ายืนยันการ Get Cash



ภาพที่ 4.24 สลิปที่ได้รับหลังจากการ Get Cash

สามารถตรวจสอบประวัตการโอนเงินได้จากหน้าประวัติ (History) ที่คล้ายกับ Installment Plan แต่แตกต่างกันที่ลำดับความสำคัญที่เรียงข้อมูล และข้อมูลที่แสดง ดังภาพที่ 4.25



ภาพที่ 4.25 หน้าประวัติของ Get Cash

บทที่ 5
สรุปผลโครงงานและข้อเสนอแนะ

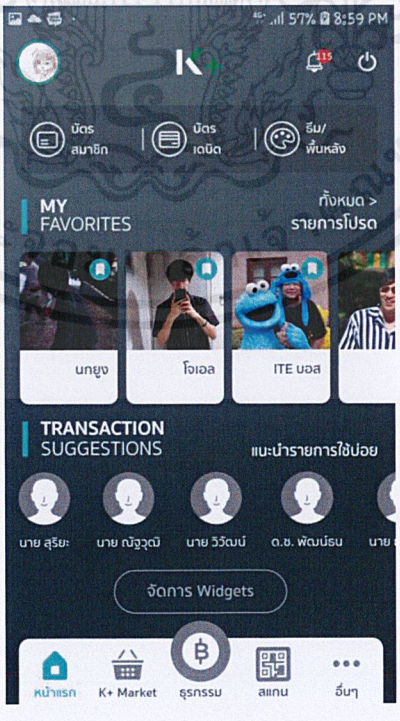
5.1 สรุปผลการดำเนินงาน

5.1.1 Voice Command

ฟีเจอร์การสั่งการด้วยเสียงที่รองรับการใช้งานในรูปแบบ โอน เติม จ่าย และเช็กยอดเงินในบัญชี และวงเงินบัตรเครดิตได้ สามารถส่งได้ตามกำหนดเวลา และทำได้ตามความต้องการทั้งหมด แต่เนื่องจากความสามารถของ Machine Learning ที่ถูกพัฒนาขึ้นนั้นยังให้ผลลัพธ์การทำงานไม่ดีเท่าที่ควร ทำให้โครงงานนี้ไม่สามารถปล่อยออกสู่ท้องตลาดได้ และยังไม่มีแผนการพัฒนาต่อในเร็ว ๆ นี้

5.1.2 Transaction Suggestions

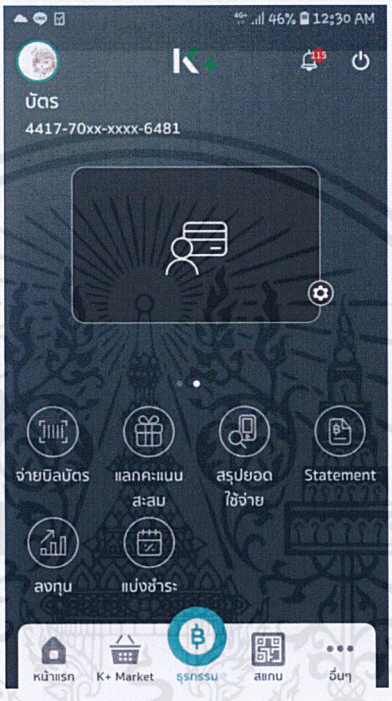
ฟีเจอร์ Transaction Suggestions สามารถแนะนำรายการใช้งานบ่อยของผู้ใช้งานได้ โดยในการพัฒนานั้นสามารถพัฒนาได้สำเสร็จตามกำหนดการ อีกทั้งยังสามารถรองรับในโทรศัพท์แอนดรอยด์ทุกรุ่น และในปัจจุบันฟีเจอร์ Transaction Suggestions มีการใช้งานอยู่จริงบนแอปพลิเคชัน K-PLUS มาตั้งแต่ช่วงกลางปี 2019



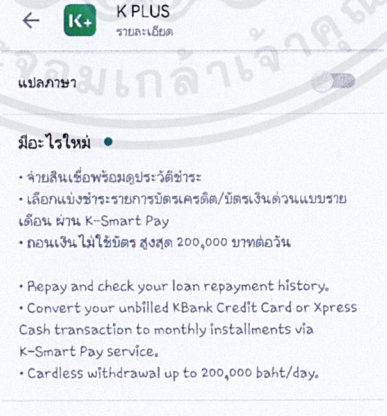
ภาพที่ 5.1 การใช้งานจริงของฟีเจอร์ Transaction Suggestions บน K-PLUS

5.1.3 Credit card 2019

การพัฒนาส่วนของ Credit card 2019 ประกอบไปด้วยฟีเจอร์ Installment Plan ที่สามารถแบ่งชำระยอดบัตรเครดิตกับธานคารได้ และฟีเจอร์ Get Cash ที่สามารถใช้บัตรเครดิตแทนเงินสด สามารถพัฒนาและส่งมอบงานได้ตามเป้าหมาย และสำหรับฟีเจอร์ Installment Plan ได้ใช้งานอยู่จริงภายในแอปพลิเคชัน K-PLUS ในหน้าบัตรเครดิต ตั้งแต่ช่วงเดือนธันวาคม 2019 และ Get Cash มีแผนที่จะปล่อยออกสู่ท้องตลาดภายในปี 2020



ภาพที่ 5.2 การใช้งาน Installment Plan บนแอปพลิเคชัน K-PLUS



ภาพที่ 5.3 การอัปเดตฟีเจอร์ Installment Plan ของ K-PLUS บน Play Store

5.2 ข้อเสนอแนะ

สถาบันและภาควิชาควรให้ความร่วมมือกันมากกว่าเดิมในการให้ข้อมูลที่ถูกต้องแก่นักศึกษา อาทิเช่น ลำดับขั้นตอนแรกยื่นเอกสาร กำหนดการ หรือข้อมูลควรทราบ พร้อมทั้งคอยเพิ่มเติมและอัปเดตข้อมูลอยู่เสมอ รวมถึงอาจารย์ภายในภาคควรมีการประสานงานกันเพื่อให้ข้อมูลที่ตรงกันแก่นักศึกษา และควรมีปฐมนิเทศที่แยกตามสายงานมากกว่าการปฐมนิเทศรวมเพื่อให้ได้ข้อมูลที่จำเป็นและตรงต่อความต้องการ

เอกสารอ้างอิง

[1] KBTG. (ม.ป.ป). “วิสัยทัศน์ขององค์กร”. สืบค้นเมื่อเดือน พฤศจิกายน 2562, จาก http://www.kbtg.tech/th

[2] Endtry Studio. (2018). “มาดูว่าระหว่าง Java กับ Kotlin มีอะไรแตกต่างกันบ้างแบบผ่าน ๆ”. สืบค้นเมื่อเดือน พฤศจิกายน 2562, จาก https://prongbang.github.io/kotlin/2018/07/28/java-vs-kotlin.html

[3] Sathittham Sangthong. (2012). “Activity Life Cycle วงจรการทำงานของแอคทิวิตี้”. สืบค้นเมื่อเดือน พฤศจิกายน 2562, จาก https://medium.com/sathittham/android-activity-life-cycle-วงจรการทำงานของแอคทิวิตี้

[4] Akexorcist. (2014). “Let's Fragment - วงจรชีวิตของ Fragment (Fragment Lifecycle)”. สืบค้นเมื่อเดือน พฤศจิกายน 2562, จาก http://www.akexorcist.com/2014/11/lets-fragment-lifecycle.html

[5] Nutti Saelor. (2017). “MVC MVP MVVM คืออะไร และต่างกันอย่างไร”. สืบค้นมื่อเดือน พฤศจิกายน 2562, จาก https://medium.com/@leelorz6/mvc-mvp-mvvm-คืออะไร และต่างกันอย่างไร

[6] Palm’s. (2018). “เริ่มต้นสร้าง Android Application พื้นฐานด้วย Android Studio (Lab 3SB04)”. สืบค้นมื่อเดือน พฤศจิกายน 2562, จาก https://medium.com/@palmz/เริ่มต้นสร้าง-android-application-พื้นฐานด้วย-android-studio-lab-3sb04-3fda43b07a1

[7] Nati Namvong. (2016). “Sketch มันของ Designer, Dev อย่างเรามันต้อง Zeplin”. สืบค้นมื่อเดือน พฤศจิกายน 2562, จาก https://stories.sellsuki.co.th/sketch-มันของ-designer-dev-อย่างเรามันต้อง -zeplin-734b483a437d

[8] mk. (2018). “GitLab ซอฟต์แวร์ CI/CD ชื่อดัง เปิดให้เชื่อมต่อกับ GitHub โดยตรงแล้ว”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก https://www.blognone.com/node/100920

[9] Sakul Montha. (2019). “Jira คืออะไร แล้ว Epic, Story, Task และ Sub-Task ต่างกันอย่างไร”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก https://medium.com/@iamgique/jira-คืออะไร-แล้ว-epic-story-task-แล้ว-sub-task-ต่างกันอย่างไร-a928d10bece0

[10] codebee. (2017). “วิธีใช้ trello โปรแกรมบริหารจัดการงานของดีและฟรี”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก https://www.codebee.co.th/labs/วิธีใช้-trello/

[11] P’Art. (2018). “มารันเทสแบบอัตโนมัติ ด้วย Jenkins กันเถอะ”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก https://www.qahive.com/2018/02/03/run-test-jenkins/

[12] W.Songak. (2017). “[ของดีเมือง Android] Lottie ไลบรารี่ที่เกิดมาเพื่อคนชอบความหวือหวา~”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก https://medium.com/ubon-startup/noobshare-ep-1-lottie-ไลบรารี่ที่เกิดมาเพื่อคนชอบความหวือหวา-6549c87cb6ce

[13] Phayao Boonon. (2018). “Mock ทุกสรรพสิ่งใน Java/Spring ด้วย Mockito”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก https://medium.com/@phayao/mock-ทุกสรรพสิ่งใน-java-spring-ด้วย-mockito-6d2a8d671f48

[14] Akexorcist. (2018). “Dagger 2 in Android [Part 1] - Dependency Injection แบบหล่อๆ ด้วย Dagger 2”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก http://www.akexorcist.com/2018/07/beautiful-dependency-inject-in-android-with-dagger-2-part-1.html

[15] Akexorcist. (2018). “Dagger 2 in Android [Part 1] - Dependency Injection แบบหล่อๆ ด้วย Dagger 2”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก http://www.akexorcist.com/2018/07/beautiful-dependency-inject-in-android-with-dagger-2-part-1.html

[16] nuuneoi. (2015). “Retrofit อัปเดตครั้งใหญ่ สู่เวอร์ชั่น 2.0 อย่างเป็นทางการ พร้อมฟีเจอร์และการเปลี่ยนแปลงเพียบ”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก https://inthecheesefactory.com/blog/retrofit-2.0/th

[17] Thanyavuth Akarasomcheep. (2018). “Agile คืออะไร เริ่มใช้งานอย่างไร”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก https://medium.com/fastwork-engineering/agile-คืออะไร-เริ่มใช้งานอย่างไร-ab749306d96e

[18] Unknown. (ม.ป.ป). “SDLC (Software Development Life Cycle) using Agile”. สืบค้นเมื่อเดือน ธันวาคม 2562, จาก http://www.riverpark.co.th/blog/agilesdlc.html

[19] Sakul Montha. (2019). “REST กับ RESTful API ต่างกันนะรู้ยัง”. สืบค้นเมื่อเดือน มกราคม 2563, จาก https://medium.com/@iamgique/restful-api-กับ-rest-api-ต่างกันนะรู้ยัง-2c70c42990e3

[20] Unknown. (2015). “มาทำความรู้จักกับ Database ชื่อที่ว่า Realm กันซักหน่อย”. สืบค้นเมื่อเดือน มกราคม 2563, จาก https://try-droid.blogspot.com/2015/12/android-realm-mobile-database.html

[21] Akexorcist. (2014). “เก็บค่าตัวแปรให้ถาวรแบบง่ายๆได้ด้วย Shared Preferences”. สืบค้นเมื่อเดือน มกราคม 2563, จาก http://www.akexorcist.com/2014/09/android-shared-preferences.html

[22] Minseo Chayabanjonglerd. (2017). “มาทำความรู้จัก Unit Test สำหรับ Android Developer กันเถอะ”. สืบค้นเมื่อเดือน มกราคม 2563, จาก https://medium.com/fungjai/มาทำความรู้จัก-unit-test-สำหรับ-android-developer-กันเถอะ-817ac642b44c

ประวัติผู้เขียน

หัวข้อโครงงาน การพัฒนาแอปพลิเคชันทางการเงินบนระบบปฏิบัติการแอนดรอยด์

ชื่อนามสกุล น.ส.วริศรา ศิริอักษร

รหัสนักศึกษา 59011201

คณะ วิศวกรรมศาสตร์

ภาควิชา วิศวกรรมคอมพิวเตอร์ **สาขาวิชา** วิศวกรรมสารสนเทศ

ประวัติส่วนตัว

วันเกิดปีเกิด 24 ธันวาคม พ.ศ. 2540

ชื่อเล่น อีฟ

อีเมล eve14072@gmail.com