

ເຄົ້າໂຄຮງໂຄຮງການຄອມພິວເຕອຮ່

ເວັບຄອມມູນືຂໍຂອງຄົນເລື່ອງແມວ

Community website for cat owners

ຈັດທຳໄດຍ

นางສາວສ້ອຍທອງແທ້ ອຸໝົນອົກ 653380346-6 Sec 3

นางສາວພິພູສຣ ມິຕຣເຈຣີຢູ່ຕັນ 653380208-8 Sec 4

นางສາວມະນຸ້ງຫຼາມ ຮມນັນທີພົງໝາ 653380323-8 Sec 4

นางສາວວາຣີນີ ອນຸສຸເຮັນທີ 653380341-6 Sec 4

นางສາວສີຣີຍາກ ອາຈຍາງຄຳ 653380348-2 Sec 4

รายงานນີ້ເປັນສ່ວນໜຶ່ງຂອງການສຶກກາວິຊາ CP353002 ລັກກາຮອກແບບພັດນາຂອ່າພົດ
ການຮຽນທີ 1 ປີການສຶກກາ 2567
ສາຂາວິຊາວິທາກາຮອກຄອມພິວເຕອຮ່ ວິທາຍາລັ້ຍກາຮອກຄອມພິວເຕອຮ່
ມາຮັດວຽກ

ชื่อโครงงาน

ชื่อภาษาไทย: เว็บคอมมูนิตี้ของคนเลี้ยงแมว

ชื่อภาษาอังกฤษ Community website for cat owners

รายชื่อผู้จัดทำ
นางสาวพิพัฒน์ มิตรเจริญรัตน์

นางสาวมณิสุขญาณ์ รมนันท์พงษา

นางสาววริณี อนุสุเรนทร์

นางสาวสร้อยทองแท้ อุ้ยนอก

นางสาวสิริยากร อาจายางคำ

อาจารย์ที่ปรึกษา
รศ. ดร.ปัญญาพล หอระตะ

ระดับการศึกษา
ปริญญาตรี

สาขาวิชา
วิทยาการคอมพิวเตอร์ วิทยาลัยการคณพิวเตอร์ มหาวิทยาลัยขอนแก่น

ปีการศึกษา
2567

บทคัดย่อ

เนื่องจากในปัจจุบันมีกลุ่มคนเลี้ยงแมวเพิ่มมากขึ้น จึงส่งผลให้เกิดปัญหามากมาย ไม่ว่าจะเป็น ปัญหาสุขภาพที่ผู้เลี้ยงอาจจะไม่รู้ว่าเกิดจากสาเหตุอะไร หรือจะเป็นอุบัติสัยที่เปลี่ยนไป หรือในเรื่องอาหาร การกินก็เป็นเรื่องที่สำคัญซึ่งผู้เลี้ยงไม่ค่อยได้ใส่ใจว่าแมวของคุณควรรับประทานแบบไหน จึงจะเหมาะสม เพราะผู้เลี้ยงส่วนใหญ่ไม่ค่อยเข้าใจารมณ์ของสัตว์เลี้ยง และบางที่ไม่รู้ว่าต้องฝึกอย่างไร

จากปัญหาข้างต้น ผู้จัดทำโครงงานได้เล็งเห็นความสำคัญจึงได้เกิดแนวความคิดในการพัฒนาเว็บไซต์ให้ตอบโจทย์สำหรับกลุ่มคนเลี้ยงแมวโดยเฉพาะ เพื่อเป็นพื้นที่ในการพูดคุยแลกเปลี่ยนประสบการณ์เกี่ยวกับแมวของตนเองหรือที่เคยได้พบเจอมาก่อนทั้งสำหรับผู้ที่เลี้ยงแมวหรือผู้ที่มีความสนใจ

กิตติกรรมประกาศ

รายงานนี้เป็นส่วนหนึ่งของรายวิชา CP353002 หลักการออกแบบพัฒนาซอฟต์แวร์ เพื่อให้ได้ศึกษาและเรียนรู้หลักในการพัฒนาและออกแบบซอฟต์แวร์ โดยได้รับความกรุณาอย่าง สูงจาก รศ. ดร.ปัญญา พล หอรัตน์ เป็นที่ปรึกษาและให้คำแนะนำแก่ไขและปรับปรุงข้อบกพร่องต่างๆ และทาง คณบุญจัดทำขอกราบขอบพระคุณมา ณ ที่นี่

ผู้จัดทำหวังว่า รายงานเล่มนี้จะเป็นประโยชน์กับผู้อ่าน ไม่มากก็น้อยหากมีข้อแนะนำ หรือข้อผิดพลาด ประการใด ผู้จัดทำขออน้อมรับไว้และขอภัยมา ณ ที่นี่

คณบุญจัดทำ

บทที่1

บทนำ

1.1 ที่มาและความสำคัญ

การเลี้ยงแมวกล้ายเป็นกิจกรรมยอดนิยมในสังคมปัจจุบัน โดยเฉพาะในกลุ่มคนรักสัตว์ แมวเป็นสัตว์เลี้ยงที่มีลักษณะนิสัยน่ารักและเป็นเพื่อนที่ดีต่อผู้คน อย่างไรก็ตาม ความรู้และประสบการณ์ในการดูแลแมวอาจแตกต่างกันไปในแต่ละบุคคล ทำให้เกิดความต้องการพื้นที่ออนไลน์ที่เจ้าของแมวสามารถแลกเปลี่ยนข้อมูลและแบ่งปันประสบการณ์กันได้ การมีบล็อกที่เป็นพื้นที่สำหรับการแบ่งปันเรื่องราวเหล่านี้จะช่วยสร้างชุมชนออนไลน์ของผู้รักแมวที่เข้มแข็งมากขึ้น

เว็บไซต์นี้จะทำหน้าที่เป็นศูนย์กลางสำหรับผู้ใช้ในการเผยแพร่บทความ แชร์เรื่องราวหรือภาพถ่ายของแมว ตลอดจนการให้คำแนะนำเกี่ยวกับการดูแลแมว นอกจากนี้ ยังเปิดโอกาสให้ผู้ใช้งานได้แลกเปลี่ยนความคิดเห็นและประสบการณ์กัน ซึ่งไม่เพียงแต่ช่วยสร้างชุมชนคนรักแมวให้แน่นแฟ้นยิ่งขึ้น แต่ยังส่งเสริมให้เกิดการเรียนรู้ร่วมกันในการดูแลแมวย่างมีประสิทธิภาพ

การพัฒนาเว็บไซต์เกี่ยวกับแมวจึงมีความสำคัญในการสร้างพื้นที่ที่ผู้ใช้สามารถแชร์เรื่องราวของพกพาได้อย่างอิสระ ทั้งยังเป็นแหล่งข้อมูลที่มีประโยชน์ในการดูแลแมว ทำให้ผู้เลี้ยงแมวสามารถเข้าถึงข้อมูลและคำแนะนำได้ง่ายขึ้น ผ่านการแบ่งปันจากประสบการณ์จริงของผู้ใช้งานคนอื่นๆ

1.2 วัตถุประสงค์

1.2.1 เพื่อสร้างพื้นที่สำหรับผู้เลี้ยงแมวในการแลกเปลี่ยนข้อมูล ความรู้ และเคล็ดลับในการดูแลแมว

1.2.2 เพื่อพัฒนาเว็บไซต์สำหรับการแบ่งปันเรื่องราวและประสบการณ์เกี่ยวกับแมวระหว่างผู้ใช้งาน

1.2.3 เพื่ออำนวยความสะดวกในการแชร์ข้อมูลสุขภาพแมว สายพันธุ์แมว อาหาร การดูแล และปัญหาที่พบบ่อยในการเลี้ยงแมว

1.3 เป้าหมายของโครงงาน

- 1.3.1 ให้พื้นที่สำหรับผู้เลี้ยงแมวมีใหม่ และผู้มีประสบการณ์ได้เข้ามาพูดคุยและปรึกษาปัญหาที่เกี่ยวข้องกับแมว
- 1.3.2 ทำให้เว็บไซต์สามารถค้นหาข้อมูลที่จำเป็นเกี่ยวกับแมวได้
- 1.3.3 ให้ผู้ใช้สามารถซื้อขาย และเปลี่ยน หรือแนะนำสินค้าและบริการที่เกี่ยวข้องกับแมวได้

1.4 ขอบเขตของโครงงาน

- 1.4.1 ระบบสมัครสมาชิกและໂປຣໄຟລ໌ຜູ້ໃຊ້
 1. ຜູ້ໃຊ້ສາມາດสร้างບັນຫຼືໄດ້
 2. ຜູ້ໃຊ້ສາມາດແກ້ໄຂຂໍ້ອມລວມຕົວໄດ້
- 1.4.2 ระบบແລກເປີ່ຍນຂໍ້ອມລ
 1. ຜູ້ໃຊ້ສາມາດສ້າງໂພສຕໍ່ໄດ້
 2. ຜູ້ໃຊ້ສາມາດເພີ່ມ ລບ ແລະ ແກ້ໄຂໂພສຕໍ່ຂອງຕົນເອງໄດ້
 3. ຜູ້ໃຊ້ສາມາດຄູ່ໂພສຕໍ່ຂອງຜູ້ອື່ນໄດ້
 4. ຜູ້ໃຊ້ສາມາດແສດງຄວາມຄິດເຫັນເກີ່ຽວກັບໂພສຕໍ່ໄດ້
 5. ຜູ້ໃຊ້ສາມາດລබຄວາມຄິດເຫັນທີ່ຕົນເອງໄດ້ແສດງຄວາມຄິດເຫັນໄປໄດ້
- 1.4.3 ระบบຂໍ້ອມລແມວຂອງຜູ້ໃຊ້
 1. ຜູ້ໃຊ້ສາມາດເພີ່ມຂໍ້ອມລແມວຂອງຕົນເອງໄດ້ຫລາຍຕົວ
 2. ຜູ້ໃຊ້ສາມາດແກ້ໄຂ ແລະ ລບຂໍ້ອມລແມວຂອງຕົນເອງໄດ້

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- 1.5.1 ຜູ້ໃຊ້ສາມາດແບ່ງປັນແລກເປີ່ຍນປະສົບການນີ້ໃນການເລື່ອງແມວໄດ້ຢ່າງສະດວກແລະເປັນຮະບບ
- 1.5.2 ປ້ອຍໃຫ້ຜູ້ໃຊ້ຈານມີພື້ນທີ່ໃນການເກີ່ຽວຮ່ວມເຮື່ອງຮາວແລະກາພຄ່າຍຂອງແມວໃນຮູບແບບອອນໄລນ໌ທີ່ສາມາດເຂົ້າຄຶງໄດ້ທຸກທີ່ທຸກເວລາ
- 1.5.3 ການພັນຫຼັກສະຖານທັງເທິດໂນໄລຍື ແລະ ກາຮອກແບບຮະບບໃໝ່ມີຄວາມສະດວກຕ່ອກການໃຊ້ຈານ ຮົມຄື່ງການພັນນະຮະບບຈັດການເນື້ອທານວິເປີ່ຍ

1.6 นิယายคัพท์

1.6.1 คอมมูนิตี้ หมายถึง การรวมกลุ่มของผู้ที่มีความสนใจหรือมีเป้าหมายร่วมกัน มุ่งเน้นการเชื่อมโยงผู้คนในวงกว้าง และเปิดโอกาสให้สามารถแชร์เนื้อหาได้ทุกประเภท วัตถุประสงค์หลักคือการให้ผู้คนสามารถแสดงตัวตน แชร์ชีวิตประจำวัน และสร้างปฏิสัมพันธ์กับคนอื่น ๆ โดยไม่มีความจำเป็นต้องมีความสนใจร่วมกัน[1]

1.6.2 เว็บไซต์ หมายถึง การจัดทำระบบนำเสนอด้วยเครือข่ายอินเทอร์เน็ต (Internet) และนำเสนอด้วยเครือข่ายอินเทอร์เน็ต สำหรับนำเสนอข้อมูลโดยใช้เครื่องคอมพิวเตอร์ เป็นการรวบรวมหน้าเว็บเพจ (Page) หลายหน้ามาทำการเชื่อมโยงกันผ่านทางไฮเปอร์ลิงก์ (Hyperlink)[2]

บทที่ 2

ทฤษฎีและเครื่องมือที่ใช้

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 ชุมชนออนไลน์ (Online Community) ต่างจาก Social Media อย่างไร?

ชุมชนออนไลน์ (Online Community) คือกลุ่มของผู้คนที่มาร่วมตัวกันในพื้นที่ออนไลน์เพื่อแลกเปลี่ยนความรู้ ประสบการณ์ และความคิดเห็นในหัวข้อหรือเรื่องราวที่มีความสนใจร่วมกัน ชุมชนออนไลน์มักมีการสร้างปฏิสัมพันธ์ที่ลึกซึ้งและยาวนาน โดยสมาชิกในชุมชนจะมีเป้าหมายหรือความสนใจร่วมกัน และมักจะมีการสื่อสารและแลกเปลี่ยนข้อมูลที่เฉพาะเจาะจง ส่วน Social Media คือแพลตฟอร์มออนไลน์ที่ถูกออกแบบมาเพื่อให้ผู้ใช้สามารถสร้างและแชร์เนื้อหา รวมถึงสร้างปฏิสัมพันธ์กับผู้ใช้อื่น ๆ แพลตฟอร์ม Social Media ที่เราๆ กันดี เช่น Facebook, Instagram, Twitter, และ TikTok มีจุดประสงค์หลักคือการเชื่อมโยงผู้คนและให้พากเข้าสามารถแชร์เรื่องราวชีวิตประจำวัน ภาพถ่าย วิดีโอ และความคิดเห็นในหัวข้อที่หลากหลาย

ชุมชนออนไลน์ (Online Community) มีวัตถุประสงค์ที่ชัดเจนในการรวมกลุ่มผู้ที่มีความสนใจหรือเป้าหมายร่วมกัน สมาชิกมักจะมีการสื่อสารที่เน้นการแลกเปลี่ยนความรู้และประสบการณ์ในหัวข้อเฉพาะ เช่น ชุมชนผู้รักการอ่าน ชุมชนนักพัฒนาโปรแกรม หรือชุมชนครูผู้สอน ส่วน Social Media มุ่งเน้นการเชื่อมโยงผู้คนในวงกว้าง และเปิดโอกาสให้ผู้ใช้สามารถแชร์เนื้อหาได้ทุกประเภท วัตถุประสงค์หลักคือการให้ผู้ใช้สามารถแสดงตัวตน แชร์ชีวิตประจำวัน และสร้างปฏิสัมพันธ์กับคนอื่น ๆ โดยไม่มีความจำกัดด้วยความสนใจร่วมกัน[1]

2.1.2 7C เพื่อการสื่อสารที่ดี และประสบความสำเร็จ

7C หรือ 7 C's of Communication คือการสื่อสารซึ่งมีองค์ประกอบ 7 อย่าง ซึ่งการสื่อสารนั้นไม่ได้จำกัดเพียงการพูดเท่านั้นแต่ยังเป็นเรื่องของการนำเสนอ การเขียน การออกความคิดเห็นและอื่นๆ อีกมากมาย เราสามารถใช้หลักการนี้ในการพัฒนาตัวเว็บไซต์ให้การสื่อสารมีประสิทธิภาพมากขึ้นได้ โดยเป็นหลักการ 7C ดังนี้

1. ชัดเจน (Clear)

ในการสื่อสารไม่ว่าจะด้วยการพูดหรือการเขียน ข้อมูลที่สื่อสารต้องมีความชัดเจน เข้าใจง่าย ชัดในเนื้อหาสาระ ใช้คำที่มีความหมายตรงตัว ไม่ต้องตีความจนอาจเกิดความเข้าใจผิด

2. ถูกต้อง (Correct)

ความถูกต้องของข้อมูล หมายถึงเป็นข้อเท็จจริง (fact) ตรวจสอบได้ สร้างความมั่นใจแก่ผู้รับข้อมูลว่า ไม่ได้ถูกหลอกให้หลงเชื่อ ใช้ภาษาได้อย่างถูกต้องเหมาะสม ถูกต้องทั้งไวยากรณ์และตัวสะกด

3. ครบถ้วน (Complete)

การสื่อสารควรเป็นการส่งข้อมูลซึ่งเป็นข้อเท็จจริงทั้งหมดที่ผู้รับข้อมูลควรทราบครบถ้วนนี้จะต่างกันไปในแต่สถานการณ์ ไม่ตักหล่นเนื้อหาสาระที่สำคัญ เป็นข้อมูลที่สร้างแรงจูงใจ

4. หนักแน่นมีสาระ (Concrete)

ข้อมูลควรมีความจำเพาะเจาะจง หนักแน่น มีสาระ ไม่คุลุเครื่องหรือกว้างจนเกินไป มีข้อเท็จจริง ไม่ใช่คำที่ลดความน่าเชื่อถือ จำเพาะเจาะจงในประเด็นที่สื่อสาร มีความเป็นรูปธรรมในเรื่องที่สื่อสาร

5. กระชับ (Concise)

ข้อมูลควรกระชับ ตรงประเด็น ไม่เยินเย้อ ชูเนื้อหาสาระหลักได้ชัดเจน ไม่กล่าววนเวียนซ้ำแล้วซ้ำอีก

6. สมเหตุสมผล (Coherent)

ข้อมูลที่สื่อสารควรมีตรรกะ มีความเป็นเหตุเป็นผล เชื่อมโยงและสอดคล้องสมพันธ์กับประเด็นหลัก สร้างมุมมองแนวคิดที่เป็นประโยชน์แก่ผู้รับเพื่อให้ได้รับการตอบสนองที่เป็นบวก มองโลกในแง่ดี เน้นไปในสิ่งที่เป็นไปได้

7. มีมารยาท (Courteous)

ข้อมูลที่สื่อสารควรมีลักษณะที่เป็นมิตร เปิดเผย ตรงไปตรงมา ไม่มีประเด็นซ่อนเร้นหรือเหน็บแนม ก้าวร้าว เลือกใช้คำพูดที่สุภาพ ให้เกียรติ ไม่นำอคติใด ๆ มาบิดเบือนข้อมูลให้ผิดไปจากข้อเท็จจริง คำนึงถึงความคิดเห็น มนุษย์ ทัศนคติ ของผู้รับข้อมูล ไม่หักห้ามหรือฝืนความรู้สึกนึกคิดของผู้รับ ใช้คำในเชิงบวก[2]

2.1.3 หลักการออกแบบเว็บขั้นพื้นฐานพร้อมองค์ประกอบและรูปแบบโครงสร้าง

หลักในการออกแบบเว็บไซต์

การออกแบบเว็บไซต์ที่ดีจะต้องคำนึงถึงหลายอย่างด้วยกัน โดยมี 9 ข้อที่ควรคำนึงถึงดังนี้

1. ความเรียบง่าย เว็บไซต์ที่ดีควรมีรูปแบบที่เรียบง่ายและไม่ซับซ้อน เพื่อให้ผู้ชมสามารถใช้งานเว็บไซต์ได้อย่างสะดวกมากขึ้น โดยเฉพาะพิภพกราฟฟิกทั้งหลาย จะต้องไม่ใช้ตัวอักษรที่เคลื่อนไหวอยู่ตลอดเวลา และไม่มีสีสันที่ดูแสงตาจนเกินไป
2. ความสม่ำเสมอ คือการเลือกใช้รูปแบบ กราฟฟิก โทนสี และการตกแต่งหรือการแสดงผลต่างๆ ในเว็บไซต์ให้เป็นรูปแบบเดียวกันหรือคล้ายคลึงกันตลอดทั้งเว็บ
3. ความเป็นเอกลักษณ์ เว็บไซต์ควรมีเอกลักษณ์เฉพาะตัว ที่สามารถบ่งบอกได้ว่าความเป็นบริษัท องค์กร หรือแบรนด์ต่างๆ
4. เนื้อหา โดยเนื้อหาที่นำมาลงในเว็บ ควรเป็นเนื้อหาที่มีความเกี่ยวข้องกับเว็บ หรืออาจเป็นเนื้อหาที่ได้สาระ มีประโยชน์ สามารถดึงดูดความสนใจของผู้คนได้ดี และที่สำคัญจะต้องมีความถูกต้อง สมบูรณ์และมีความทันสมัย
5. ระบบ ควรออกแบบให้สามารถใช้งานได้ง่ายและสะดวก สื่อความหมายต่างๆ และอธิบายได้อย่างชัดเจน รวมถึงต้องมีรูปแบบ และลำดับรายการที่มีความสม่ำเสมอ
6. ลักษณะเด่น ส่วนนี้จะถือเป็นหน้าตาของเว็บไซต์เพื่อใช้ในการดึงผู้ใช้งาน อาจออกแบบลักษณะเด่นของเว็บให้ตรงกับความชอบส่วนใหญ่ของกลุ่มเป้าหมาย
7. การใช้งานที่ไม่จำกัด การทำเว็บไซต์ให้รองรับการเข้าใช้งานจากหลายระบบ ไม่ว่าจะเป็นการเข้าใช้งานจากเครื่อง PC สมาร์ทโฟน หรือการใช้งานบนแพทฟอร์มต่างๆในการเข้าใช้งาน
8. คุณภาพในการออกแบบ จำเป็นต้องทำเว็บไซต์ให้มีคุณภาพมากที่สุด ไม่ว่าจะเป็นในเรื่องของการเรียบเรียงเนื้อหาอย่างรอบคอบ การตรวจสอบความถูกต้องและการทำให้เว็บไซต์มีความน่าเชื่อถือ
9. การเชื่อมโยงไปยังลิงค์ต่างๆ ซึ่งจะต้องเชื่อมโยงไปยังหน้าเว็บที่มีอยู่จริง และมีเนื้อหาที่เกี่ยวข้องกัน และควรหมั่นตรวจสอบอยู่เสมอ ว่าระบบการเชื่อมโยงยังคงทำงานได้ตามปกติและมีความถูกต้อง แม่นยำอยู่หรือไม่

ส่วนประกอบสำคัญบนหน้าเว็บเพจ

มีส่วนประกอบสำคัญอยู่ 3 ส่วน ได้แก่

1. ส่วนหัวของหน้า (Header)

อยู่ตอนบนสุดของหน้าและเป็นส่วนที่สำคัญที่สุด โดยจะต้องทำให้สามารถดึงดูดผู้ชมให้รู้สึกอย่างติดตามเนื้อหาในเว็บไซต์ต่อไป ซึ่งส่วนใหญ่ก็มักจะมีการใส่ภาพกราฟฟิคให้ดูสวยงาม สิ่งสำคัญหลักๆ เลย ก็คือ โลโก้ ชื่อเว็บไซต์และเมนูหลักที่สามารถลิงค์ไปยังเนื้อหาในหน้าเว็บเพจต่างๆ ได้

2. ส่วนของเนื้อหา (Body)

อยู่บริเวณตอนกลางของหน้าเว็บ โดยจะแสดงข้อมูลเกี่ยวกับเนื้อหาบนเว็บแบบคร่าวๆ ซึ่งก็จะมีข้อความ กราฟฟิค ตารางข้อมูลหรือวิดีโอประกอบอยู่ และหากมีเมนูแบบเฉพาะกลุ่มก็จะถูกจัดไว้ในหน้านี้ เช่น กัน และที่สำคัญเนื้อหาในส่วนนี้ควรจะมีความกระชับ เข้าใจง่าย มีการใช้รูปแบบตัวอักษรแบบเรียบง่ายและเป็นระเบียบ

3. ส่วนท้ายของหน้า (Footer)

อยู่ล่างสุดของหน้าเว็บ ซึ่งจะมีหรือไม่มีก็ได้ ส่วนนี้จะแสดงถึงข้อมูลต่างๆ เพิ่มเติมเข้าไป เช่น ข้อความที่แสดงถึงการเป็นลิขสิทธิ์ ข้อมูลเจ้าของเว็บไซต์ วิธีการติดต่อและคำแนะนำต่างๆ เกี่ยวกับการใช้งานเว็บไซต์อย่างถูกต้อง เป็นต้น

วิธีการเลือกใช้สีสำหรับการออกแบบเว็บไซต์

การเลือกใช้สีในการออกแบบเว็บไซต์มีความสำคัญเป็นอย่างมาก เพราะสีสามารถกำหนดอารมณ์ ความรู้สึก และกระตุ้นการรับรู้ทางด้านจิตใจของมนุษย์ได้ดี ดังนั้นสีที่ใช้จึงต้องมีความสอดคล้องกับเนื้อหาและจุดประสงค์ของเว็บ ว่าต้องการให้ผู้เข้าชมรู้สึกอย่างไรต่อเนื้อหาที่ได้อ่าน โดยรูปแบบของสีที่สายตาของมนุษย์สามารถมองเห็นได้ก็แบ่งออกเป็น 3 กลุ่มดังต่อไปนี้

1. สีโทนร้อน (Warm Colors) เป็นสีแห่งความอบอุ่น ปลอบโยนและกระตุ้นความสุขได้ดี ซึ่งจะทำให้ผู้เข้าชมรู้สึกมีชีวิตชีวาและมีแรงผลักดันมากขึ้น อีกทั้งยังช่วยดึงดูดให้ผู้ชมรู้สึกอย่างติดตามเนื้อหามากขึ้น

2. สีโทนเย็น (Cool Colors) เป็นสีแห่งความสุภาพและความอ่อนโยน ทำให้ผู้ชมรู้สึกผ่อนคลายและเพลิดเพลินมากขึ้น และยังสามารถใช้ในการนำเสนองานจากในระยะไกลได้อีกด้วย

3. สีโทนกลาง (Neutral Colors) สีเหล่านี้มักจะถูกนำไปสมกับสีอื่นๆ เพื่อให้เกิดสีที่เป็นกลางมากขึ้น และให้ความรู้สึกที่เป็นธรรมชาติ[4]

2.2 เครื่องมือที่ใช้

2.2.1 โปรแกรม Eclipse

Eclipse คือโปรแกรมที่ใช้สำหรับพัฒนาภาษาได้หลายภาษา ซึ่งโปรแกรม Eclipse เป็นโปรแกรมหนึ่งที่ใช้ในการพัฒนา Application Server ได้อย่างมีประสิทธิภาพ และเนื่องจาก Eclipse เป็นซอฟต์แวร์ Open Source ที่พัฒนาขึ้นเพื่อใช้โดยนักพัฒนาเอง ทำให้ความก้าวหน้าในการพัฒนาของ Eclipse เป็นไปอย่างต่อเนื่อง และรวดเร็ว[5]

2.2.2 integrated development environment (IDE)

เป็นโปรแกรมประยุกต์ซอฟต์แวร์ที่ช่วยให้โปรแกรมเมอร์พัฒนาซอฟต์แวร์ได้อย่างมีประสิทธิภาพ ช่วยเพิ่มประสิทธิภาพการทำงานของนักพัฒนาด้วยการผสมผสานความสามารถต่างๆ เข้าด้วยกัน เช่น การแก้ไข การสร้าง การทดสอบ การจัดแพ็คเกจซอฟต์แวร์ สำหรับแอปพลิเคชันที่ใช้งานง่าย เช่นเดียวกับที่นักเขียนใช้ตัวแก้ไขข้อความ นักบัญชีใช้สเปรดชีต นักพัฒนาซอฟต์แวร์ใช้ IDE เพื่อทำให้งานของพากขา่ง่ายขึ้น[6]

2.2.3 MySQL

คือ ระบบจัดการฐานข้อมูล หรือ Database Management System (DBMS) แบบข้อมูลเชิง สัมพันธ์ หรือ Relational Database Management System (RDBMS) ซึ่งเป็นระบบฐานข้อมูลที่จัดเก็บรวมข้อมูลในรูปแบบตาราง โดยมีการแบ่งข้อมูลออกเป็นแถว (Row) และในแต่ละแถวแบ่ง ออกเป็นคอลัมน์ (Column) เพื่อเข้มข้นระหว่างข้อมูลในตารางกับข้อมูลในคอลัมน์ที่กำหนด แทนการ เก็บข้อมูลที่แยกออกจากกัน โดยไม่มีความเข้มข้นระหว่างข้อมูลในตารางกับข้อมูลในคอลัมน์ที่กำหนด แทนการ เก็บข้อมูลที่แยกออกจากกัน โดยไม่มีความเข้มข้น ซึ่งประกอบด้วยข้อมูล (Attribute) ที่มี ความสัมพันธ์เข้มข้น (Relation) โดยใช้ RDBMS Tools สำหรับการควบคุมและจัดเก็บฐานข้อมูลที่ จำเป็น ทำให้นำไปประยุกต์ใช้งานได้ง่าย ช่วยเพิ่มประสิทธิภาพในการทำงานให้มีความยืดหยุ่นและ รวดเร็วได้มากยิ่งขึ้น รวมถึงเชื่อมโยงข้อมูล ที่จัดแบ่งกลุ่มข้อมูลแต่ละประเภทได้ตามต้องการ จึงทำให้ MySQL เป็นโปรแกรมระบบจัดฐานข้อมูลที่ได้รับความนิยมสูง[7]

2.2.5 ภาษา Java

Java เป็นภาษาการเขียนโปรแกรมที่ใช้กันอย่างแพร่หลายสำหรับการเขียนโค้ดเว็บแอปพลิเคชัน ซึ่งเป็นตัวเลือกที่ได้รับความนิยมในหมู่นักพัฒนามากกว่าสองทศวรรษ โดยมีการใช้งานแอปพลิเคชัน Java หลายล้านรายการในปัจจุบัน Java เป็นภาษาแบบหลายแพลตฟอร์ม เชิงวัตถุ และใช้เครือข่ายเป็นศูนย์กลางที่สามารถใช้เป็น

แพลตฟอร์มได้ในตัวเอง ซึ่งเป็นภาษาการเขียนโปรแกรมที่รวดเร็ว ปลอดภัย และเชื่อถือได้สำหรับการเขียนโค้ดทุกประเภทตั้งแต่แอปมือถือและซอฟต์แวร์ระดับองค์กร ไปจนถึงแอปพลิเคชัน Big Data และเทคโนโลยีฝั่งเซิร์ฟเวอร์[8]

2.2.6 ภาษา HTML

HTML ย่อมาจาก Hyper Text Markup Language คือภาษาคอมพิวเตอร์ที่ใช้ในการแสดงผลของเอกสารบน website หรือที่เราเรียกว่าเว็บเพจ HTML เป็นอีกภาษาหนึ่งที่ใช้เขียนโปรแกรมได้ เป็นภาษาประเภท Markup ใช้สำหรับการสร้างเว็บเพจ โครงสร้างของ HTML จะเป็นในรูปแบบของ Tag ต่างๆ และ Web Browser จะแปลความของ Tag แต่ละ Tag ออกแบบมาเป็นหน้าตาเว็บไซต์จากรูปแบบของภาษาสำหรับการสร้างหน้าเว็บที่มีลักษณะเป็นเอกสารแบบໄอเปอร์เท็กซ์ ซึ่งมีคุณสมบัติที่สามารถเข้ามายโยงข้อมูลต่างๆ ไปยังหน้าเว็บอื่นๆ ตามต้องการได้ทำให้การเข้ามายโยงข้อมูลในหน้าเว็บต่างๆ เป็นไปอย่างสะดวกและรวดเร็ว[9]

2.2.6 Spring Boot

Spring Boot คือ Open-Source Framework ที่ใช้สำหรับการออกแบบโครงสร้าง Service หรือการออกแบบซอฟต์แวร์อย่าง Microservice และ Spring Boot ยังมีจุดเด่นในการช่วยให้สามารถสร้างแอปพลิเคชันแบบ Stand-alone โดยตัว Spring Boot จะเข้ามาช่วยทำหน้าที่ในการจัดการกับตัวโค้ดให้เป็นระเบียบและรวดเร็วมากขึ้น ทำให้การทำงานของ Spring แอปพลิเคชันก็จะมีความรวดเร็วมากขึ้นด้วยเช่นกัน

นอกจากนี้ Spring Boot ยังมี Auto Configuration ที่ช่วยลดความยุ่งยากในการกำหนดค่าต่าง ๆ ลง และก็สามารถนำมาใช้งานได้เลยทันที อีกทั้ง Spring Boot เป็น Framework ที่มีการพัฒนาอยู่ตลอดเวลา[10]

บทที่ 3

วิธีการจัดทำโครงงาน

3.1 SOLID ที่นำมาประยุกต์กับระบบ

3.1.1 Single Responsibility Principle (SRP)

ทุกคลาสคร้มีหน้าที่เดียว เช่น CatService CommentService และ UserService มีความรับผิดชอบเฉพาะในการจัดการข้อมูลที่เกี่ยวข้องกับแมว คอมเมนต์ และผู้ใช้ตามลำดับ

3.1.2 Dependency Inversion Principle (DIP)

การประยุกต์ใช้ Spring's Dependency Injection เพื่อแยกความรับผิดชอบระหว่าง Service และ Repository ทำให้การทดสอบง่ายขึ้น และลดการพึ่งพาที่ตรงไปตรงมา เช่น ใน CatService เราใช้ Dependency Injection (DI) เพื่อ inject CatRepository ผ่าน Constructor โดยทำให้ CatService สามารถเรียกใช้เมธอดใน CatRepository ได้อย่างสะดวก

3.1.3 Repository Pattern

การประยุกต์ใช้ Repository Pattern ในการจัดการกับการเข้าถึงข้อมูล เช่น CatRepository CommentRepository เพื่อทำให้โค้ดของ Service คลีนและมีการจัดการข้อมูลอย่างมีระบบ

3.2 ฐานข้อมูลใน MySQL

3.2.1 Table User

Attribute	Type	Constraint	ตัวอย่างข้อมูล
id	long	PK	u01
username	String		kwong11
password	String		kw1111

email	String		k00@gmail.com
userPic	String		https://shorturl.at/57mHE

3.2.2 Table Cat

Attribute	Type	Constraint	ตัวอย่างข้อมูล
id	long	PK	c01
name	String		leo
breed	String		ເປົ່າມະເຈີຍ
age	String		1 ປີ
catPic	String		https://shorturl.at/57mHE
Id_user	String	FK	U01

3.2.3 Table Post

Attribute	Type	Constraint	ตัวอย่างข้อมูล
id	long	PK	p01
title	String		ແນວຂ້າວນ
content	String		ນ້ອງນໍາຮັກໄໝ໌

postPic	String		https://shorturl.at/57mHE
id_user	String	FK	u01

3.2.4 Table Comment

Attribute	Type	Constraint	ตัวอย่างข้อมูล
id	long	PK	c01
content	String		น่ารัก
id_post	String	FK	p01

ความสัมพันธ์ระหว่างตาราง

ตาราง users เชื่อมโยงกับตาราง posts ผ่าน user_id

ตาราง posts เชื่อมโยงกับตาราง user ผ่าน user_id

ตาราง cat เชื่อมโยงกับตาราง comments ผ่าน user_id

ตาราง comment เชื่อมโยงกับตาราง post ผ่าน post_id

3.3 การสร้างระบบเว็บโดยใช้ Spring boot

3.1.1 ตั้งค่าโปรเจกต์

สร้างโปรเจกต์ Spring Boot ด้วย Spring Initializr โดยเลือก dependencies เช่น Spring Web, Spring Data JPA, และ Spring Security

3.1.2 สร้าง Package Model โดยภายใน Package Model ประกอบไปด้วย

1. Class User

```
L4@Entity
L5@Table(name = "users")
L6public class User {
L7    @Id
L8    @GeneratedValue(strategy = GenerationType.AUTO)
L9    private long id;
L10   private String username;
L11   private String password;
L12   private String email;
L13   @Column(length = 10000)
L14   private String userPic;
L15
L16   @Column(name = "reset_password_token")
L17   private String resetPasswordToken;
L18
L19   @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
L20   private List<Cat> cats;
L21
L22   @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
L23   private List<Post> posts;
L24
L25   @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
L26   private List<Comment> comments;
L27
L28
L29   public User(String username, String password, String email, String userPic) {
L30       super();
L31       this.username = username;
L32       this.password = password;
L33       this.email = email;
L34       this.userPic = userPic;
L35   }
L36
L37   public User() {
L38       super();
L39       // TODO Auto-generated constructor stub
L40   }
L41
L42   public long getId() {
L43       return id;
L44   }
L45
L46   public void setId(long id) {
L47       this.id = id;
L48   }
L49
L50   public String getUsername() {
L51       return username;
L52   }
L53
L54   public void setUsername(String username) {
L55       this.username = username;
L56   }
L57
L58   public String getPassword() {
L59       return password;
L60   }
L61
L62   public void setPassword(String password) {
L63       this.password = password;
L64   }
L65
L66   public String getEmail() {
L67       return email;
L68   }
L69
L70   public void setEmail(String email) {
L71       this.email = email;
L72   }
L73
L74   public String getUserPic() {
L75       return userPic;
L76   }
```

```

public void setUserPic(String userPic) {
    this.userPic = userPic;
}

public String getResetPasswordToken() {
    return resetPasswordToken;
}

public void setResetPasswordToken(String resetPasswordToken) {
    this.resetPasswordToken = resetPasswordToken;
}

public List<Cat> getCats() {
    return cats;
}

public void setCats(List<Cat> cats) {
    this.cats = cats;
}

public List<Post> getPosts() {
    return posts;
}

public void setPosts(List<Post> posts) {
    this.posts = posts;
}

public List<Comment> getComments() {
    return comments;
}

public void setComments(List<Comment> comments) {
    this.comments = comments;
}

:4   @Override
:5   public String toString() {
:6       return "User [id=" + id + ", username=" + username + ", password=" + password + ", email=" + email
:7           + ", userPic=" + userPic + ", resetPasswordToken=" + resetPasswordToken + ", cats=" + cats + ", posts="
:8           + posts + ", comments=" + comments + "]";
:9   }
:0
:1}

```

ส่วนประกอบหลักในคลาส User

การประกาศ Entity และ Table

- คลาสนี้ถูกกำหนดเป็น @Entity เพื่อบอกให้ Spring Data JPA รู้ว่าเป็น Entity ที่จะทำการแมป ข้อมูล เข้ากับฐานข้อมูล และกำหนดตารางในฐานข้อมูลผ่าน @Table(name = "users")
- รหัสของผู้ใช้ที่ถูกกำหนดเป็นคีย์หลัก (@Id) และใช้การสร้างอัตโนมัติผ่าน @GeneratedValue (strategy = GenerationType.AUTO)

ฟิลด์ต่าง ๆ

- username, password, email เป็นข้อมูลพื้นฐานของผู้ใช้

- userPic ทำหน้าที่เก็บข้อมูลรูปโปรไฟล์ของผู้ใช้
- resetPasswordToken ใช้เก็บโทเคนสำหรับการรีเซ็ตรหัสผ่าน

ความสัมพันธ์ระหว่าง Entity

- มีการใช้ @OneToMany เพื่อบอกว่า ผู้ใช้ 1 คนสามารถมีหลายแมว (cats), โพสต์ (posts), และคอมเมนต์ (comments) ซึ่งแสดงถึงความสัมพันธ์แบบ One-to-Many
- ความสัมพันธ์เหล่านี้ถูกกำหนดผ่าน mappedBy = "user" เพื่อบอกว่าฝั่งตรงข้ามในคลาส Cat, Post, Comment จะเป็นตัวควบคุมความสัมพันธ์ (ฝั่ง "เจ้าของ")

Constructor

- คลาสนี้มีทั้ง constructor แบบมีพารามิเตอร์ (User(String username, String password, String email, String userPic)) และ constructor เปลาสำหรับ Spring Boot ในการสร้างออบเจกต์ โดยอัตโนมัติ

Getter และ Setter

- มีการสร้าง getter และ setter สำหรับฟิลด์ทุกตัว เพื่อให้ง่ายต่อการเข้าถึงและแก้ไขค่าของฟิลด์

เมธอด `toString()`

- เมธอดนี้ถูก override เพื่อใช้แสดงข้อมูลของผู้ใช้ในรูปแบบสตริง ซึ่งสามารถใช้ในการดีบัก หรือ แสดงผลในระบบได้

2. Class Cat

```
2@Entity
3@Table(name="cat")
4public class Cat {
5
6    @Id
7    @GeneratedValue(strategy=GenerationType.AUTO)
8    private long id;
9    private String name;
10   private String breed;
11   private int age;
12   @Column(length = 10000)
13   private String catPic;
14
15   @ManyToOne
16   @JoinColumn(name="user_id")
17   private User user;
18
19   public Cat(long id, String name, String breed, int age, String catPic) {
20       super();
21       this.id = id;
22       this.name = name;
23       this.breed = breed;
24       this.age = age;
25       this.catPic = catPic;
26   }
27
28   public Cat() {
29       super();
30       // TODO Auto-generated constructor stub
31   }
32
33   public long getId() {
34       return id;
35   }
36
37   public void setId(long id) {
38       this.id = id;
39   }
40
41   public String getName() {
42       return name;
43   }
44
45   public void setName(String name) {
46       this.name = name;
47   }
48
49   public String getBreed() {
50       return breed;
51   }
52
53   public void setBreed(String breed) {
54       this.breed = breed;
55   }
56
57   public int getAge() {
58       return age;
59   }
60
61   public void setAge(int age) {
62       this.age = age;
63   }
64
65   public String getCatPic() {
66       return catPic;
67   }
68
69   public void setCatPic(String catPic) {
70       this.catPic = catPic;
71   }
72
73   public User getUser() {
74       return user;
75   }
76}
```

```

7  public void setUser(User user) {
8      this.user = user;
9  }
0
1  @Override
2  public String toString() {
3      return "Cat [id=" + id + ", name=" + name + ", breed=" + breed + ", age=" + age + ", catPic=" + catPic + "]";
4  }
5
6 }

```

ส่วนประกอบหลักในคลาส Cat

การประกาศ Entity และ Table

- คลาสนี้ถูกกำหนดเป็น @Entity เพื่อบอกให้ Spring Data JPA รู้ว่าเป็น Entity
ที่จะทำการแมปข้อมูลเข้ากับฐานข้อมูล และกำหนดตารางในฐานข้อมูลผ่าน @Table(name = "cat")

ฟิลด์ต่าง ๆ

- id หมายถึง รหัสของแมว เป็นคีย์หลัก (@Id) และใช้การสร้างอัตโนมัติผ่าน @GeneratedValue(strategy = GenerationType.AUTO)
- name หมายถึง ชื่อของแมว
- breed หมายถึง สายพันธุ์ของแมว
- age หมายถึง อายุของแมว
- catPic หมายถึง รูปของแมวที่ถูกเก็บในฐานข้อมูลเป็นสตริง
โดยกำหนดขนาดสูงสุดของข้อมูลเป็น 10,000 ตัวอักษรผ่าน @Column(length = 10000)

ความสัมพันธ์กับ User

- ใช้ @ManyToOne เพื่อกำหนดความสัมพันธ์แบบ Many-to-One หมายความว่า
แมวหลายตัว สามารถมีเจ้าของคนเดียวได้
- ใช้ @JoinColumn(name = "user_id") เพื่อกำหนดคอลัมน์ user_id ในตาราง cat
ที่ใช้ในการเชื่อมต่อกับตาราง users

Constructor

- Constructor แบบมีพารามิเตอร์ (Cat(long id, String name, String breed, int age, String catPic)) ใช้สำหรับการสร้างของออบเจกต์ Cat ที่มีข้อมูลครบถ้วน
- Constructor เป็นว่าง (Cat()) ถูกสร้างไว้สำหรับการสร้างของออบเจกต์ว่าง ๆ ที่ Spring Boot อาจใช้

Getter และ Setter

- มีการสร้าง getter และ setter สำหรับฟิลด์ทุกด้าน เพื่อให้ง่ายต่อการเข้าถึงและแก้ไขค่าของฟิลด์

เมธอด `toString()`

- เมธอดนี้ถูก override เพื่อใช้แสดงข้อมูลของแมวในรูปแบบสตริง ซึ่งสามารถใช้ในการตีบักหรือแสดงผลในระบบได้

3. Class Post

```

19@Entity
20@Table(name="post")
21public class Post {
22
23    @Id
24    @GeneratedValue(strategy=GenerationType.AUTO)
25    private long id;
26    private String title;
27    @Column(length = 10000)
28    private String content;
29    private String postPic;
30
31    @ManyToOne
32    @JoinColumn(name = "user_id")
33    private User user;
34
35    @OneToMany(mappedBy = "post", cascade = CascadeType.ALL)
36    @JsonIgnore
37    private List<Comment> comments;
38
39    public Post(long id, String title, String content, String postPic) {
40        super();
41        this.id = id;
42        this.title = title;
43        this.content = content;
44        this.postPic = postPic;
45    }
46
47    public Post() {
48        super();
49        // TODO Auto-generated constructor stub
50    }
51
52    public long getId() {
53        return id;
54    }

```

```

56     public void setId(long id) {
57         this.id = id;
58     }
59
60     public String getTitle() {
61         return title;
62     }
63
64     public void setTitle(String title) {
65         this.title = title;
66     }
67
68     public String getContent() {
69         return content;
70     }
71
72     public void setContent(String content) {
73         this.content = content;
74     }
75
76     public String getPostPic() {
77         return postPic;
78     }
79
80     public void setPostPic(String postPic) {
81         this.postPic = postPic;
82     }
83
84     public User getUser() {
85         return user;
86     }
87
88     public void setUser(User user) {
89         this.user = user;
90     }
91
92     public List<Comment> getComments() {
93         return comments;
94     }
95
96     public void setComments(List<Comment> comments) {
97         this.comments = comments;
98     }
99
100    @Override
101    public String toString() {
102        return "Post [id=" + id + ", title=" + title + ", content=" + content + ", postPic=" + postPic + ", user=" + user + "]";
103    }
104}

```

อธิบายส่วนประกอบของคลาส Post

การประกาศ Entity และ Table

- คลาสนี้ถูกกำหนดเป็น @Entity เพื่อบอกให้ JPA รู้ว่าเป็น Entity ที่จะถูกแมปกับฐานข้อมูล และกำหนดตารางชื่อ post ผ่าน @Table(name = "post")

พิล็อตต่าง ๆ

- id หมายถึง รหัสโพสต์ เป็นคีย์หลัก (@Id) และใช้การสร้างอัตโนมัติผ่าน @GeneratedValue(strategy = GenerationType.AUTO)

- title หมายถึง ชื่อของโพสต์
- content หมายถึง เนื้อหาของโพสต์ กำหนดความยาวสูงสุด 10,000 ตัวอักษร
(@Column(length = 10000))
- postPic หมายถึง รูปภาพที่แนบมา กับโพสต์

ความสัมพันธ์กับ User

- ใช้ @ManyToOne เพื่อกำหนดความสัมพันธ์แบบ Many-to-One หมายความว่า โพสต์หลายโพสต์สามารถถูกสร้างโดยผู้ใช้คนเดียว
- ใช้ @JoinColumn(name = "user_id") เพื่อบรุ่งบอกลิ้ม์ user_id ในตาราง post จะใช้เชื่อมกับตาราง users

ความสัมพันธ์กับ Comment

- ใช้ @OneToMany(mappedBy = "post", cascade = CascadeType.ALL) เพื่อกำหนดความสัมพันธ์ แบบ One-to-Many ระหว่างโพสต์และคอมเม้นต์หมายถึงโพสต์แต่ละโพสต์สามารถมีหลายคอมเม้นต์
- ใช้ @JsonIgnore เพื่อไม่ให้ข้อมูลของคอมเม้นต์ถูกส่งออกมายัง JSON response เพื่อป้องกันปัญหา ในการเรียกข้อมูลแบบวนซ้ำ (circular reference)

Constructor

- Constructor แบบมีพารามิเตอร์ (Post(long id, String title, String content, String postPic)) ใช้สำหรับการสร้างออบเจกต์ Post ที่มีข้อมูลครบถ้วน
- Constructor เป็นว่าง (Post()) ถูกสร้างไว้เพื่อให้ Spring หรือ JPA สามารถสร้างออบเจกต์ได้ตามต้องการ

Getter และ Setter

- มีการสร้าง getter และ setter สำหรับฟิลด์ทุกตัว
เพื่อให้สามารถเข้าถึงและแก้ไขค่าของฟิลด์ได้ง่าย

เมธอด `toString()`

- เมธอดนี้ถูก override เพื่อแสดงข้อมูลของเพสต์ในรูปแบบสตริง
ซึ่งเป็นประโยชน์ในการดีบัก หรือแสดงข้อมูลในระบบ

4. Comment

```
17@Entity
18@Table(name="comment")
19public class Comment {
20
21    @Id
22    @GeneratedValue(strategy=GenerationType.AUTO)
23    private long id;
24    @Column(length = 10000)
25    private String content;
26
27    @ManyToOne
28    @JoinColumn(name = "user_id")
29    private User user;
30
31
32    @ManyToOne
33    @JoinColumn(name="post_id")
34    private Post post;
35
36
37    public Comment(long id, String content) {
38        super();
39        this.id = id;
40        this.content = content;
41    }
42
43    public Comment() {
44        super();
45        // TODO Auto-generated constructor stub
46    }
47
48    public long getId() {
49        return id;
50    }
51
52    public void setId(long id) {
53        this.id = id;
54    }
--
```

```

56     public String getContent() {
57         return content;
58     }
59
60     public void setContent(String content) {
61         this.content = content;
62     }
63
64     public User getUser() {
65         return user;
66     }
67
68     public void setUser(User user) {
69         this.user = user;
70     }
71
72     public Post getPost() {
73         return post;
74     }
75
76     public void setPost(Post post) {
77         this.post = post;
78     }
79
80     @Override
81     public String toString() {
82         return "Comment [id=" + id + ", content=" + content + ", user=" + user + ", post=" + post + "]";
83     }
84 }

```

อธิบายส่วนประกอบของคลาส Comment

การประกาศ Entity และ Table

- คลาสนี้ถูกกำหนดเป็น @Entity เพื่อบ่งบอกว่าเป็น Entity สำหรับจัดเก็บข้อมูลในฐานข้อมูล และกำหนดให้ แมปกับตารางชื่อ comment ผ่าน @Table(name = "comment").

ฟิลด์ต่าง ๆ

- Id หมายถึง รหัสคอมเมนต์ ซึ่งเป็นคีย์หลักของตาราง (@Id) และกำหนดให้สร้างอัตโนมัติตัวย @GeneratedValue(strategy = GenerationType.AUTO).
- content หมายถึง เนื้อหาของคอมเมนต์ ถูกกำหนดให้สามารถบรรจุข้อความที่มีความยาวสูงสุด 10,000 ตัวอักษร (@Column(length = 10000)).

ความสัมพันธ์กับ User

- ใช้ @ManyToOne เพื่อบอกว่าแต่ละคอมเมนต์สามารถสัมภาระได้โดยผู้ใช้คนเดียว
- ใช้ @JoinColumn(name = "user_id") เพื่อเชื่อมโยงฟิลด์ user_id ในตาราง comment กับ id ของตาราง users

ความสัมพันธ์กับ Post

- ใช้ @ManyToOne เพื่อบรุ่งบอกว่าคอมเม้นต์หลายคอมเม้นต์สามารถถูกโพสต์ในโพสต์เดียว
- ใช้ @JoinColumn(name = "post_id") เพื่อกำหนดว่าฟิลด์ post_id ในตาราง comment จะเชื่อมโยงกับ id ของตาราง post.

Constructor

- Constructor แบบมีพารามิเตอร์ (Comment(long id, String content))
ใช้สำหรับการสร้างคอมเม้นต์ ที่มีข้อมูลพร้อมใช้งาน.
- Constructor เเปล่า (Comment()) ใช้สำหรับการสร้างออบเจ็คต์เปล่าหรือถูกใช้งานโดย Spring/JPA.

Getter และ Setter

- มีการสร้าง getter และ setter สำหรับฟิลด์ทั้งหมด
ทำให้สามารถเข้าถึงและปรับเปลี่ยนข้อมูลในฟิลด์ ได้ง่าย

เมธอด toString()

- เมธอดนี้ถูก override เพื่อแสดงข้อมูลของคอมเม้นต์ในรูปแบบข้อความ
เป็นประโยชน์ในการดีบักหรือ แสดงข้อมูลในระบบ

3.3.2 สร้างแพ็คเกจ Repository ภายในแพ็คเกจ Repository ประกอบไปด้วย

1. class CatNotFoundException

```
1 package com.example.demo.repository;
2
3 public class CatNotFoundException extends RuntimeException{
4
5     public CatNotFoundException(Long id) {
6         super("Could not found cat "+id);
7     }
8 }
9 }
```

อธิบายส่วนประกอบของคลาส

- คลาสนี้สืบทอดจาก RuntimeException ซึ่งเป็น exception ที่ไม่จำเป็นต้องจับ (unchecked exception) ช่วยให้สามารถโยน exception ได้โดยไม่ต้องประกาศ throws ในเมธอดที่ใช้

Constructor

- Constructor รับพารามิเตอร์ Long id ซึ่งเป็น id ของแมวที่ไม่พบ
- ใช้ super("Could not found cat " + id) เพื่อเรียก constructor ของ RuntimeException พร้อมส่งข้อความที่กำหนดเองว่า "Could not found cat [id]" ซึ่งจะถูกเก็บใน exception object และแสดงเมื่อ exception นี้ถูกโยน

2. interface CatRepository

```
1 package com.example.demo.repository;
2
3 import java.util.List;
4 import org.springframework.data.repository.CrudRepository;
5 import org.springframework.stereotype.Repository;
6 import com.example.demo.model.Cat;
7 import com.example.demo.model.Comment;
8
9 @Repository
10 public interface CatRepository extends CrudRepository<Cat, Long>{
11 }
12
```

อธิบายส่วนประกอบของ CatRepository

ขยายจาก CrudRepository<Cat, Long>

- CrudRepository เป็น interface ที่ให้เมธอดพื้นฐานสำหรับการทำงานกับฐานข้อมูล.

- Cat คือ entity ที่จะจัดการ และ Long คือชนิดของ primary key ของ entity Cat

Annotation @Repository

- ระบุว่า CatRepository เป็น Spring Bean ประเภท Repository ซึ่งช่วยจัดการการเข้ามต่อ กับฐานข้อมูล.
- Spring จะทำการ inject และจัดการ lifecycle ของ CatRepository ให้อัตโนมัติ.

3. Class CommentNotFoundException

```
1 package com.example.demo.repository;
2
3 public class CommentNotFoundException extends RuntimeException{
4
5     public CommentNotFoundException(Long id) {
6         super("Could not found comment "+id);
7     }
8 }
```

อธิบายส่วนประกอบของ CommentNotFoundException

ขยายจาก RuntimeException

- คลาสนี้ขยายจาก RuntimeException ทำให้สามารถโยน exception ได้โดยไม่ต้องประกาศในเมธอด (unchecked exception).

Constructor

- รับพารามิเตอร์ Long id ซึ่งเป็น ID ของคอมเม้นต์ที่ไม่พบในระบบ.
- ใช้คำสั่ง super() เพื่อส่งข้อความแจ้งข้อผิดพลาดไปยัง RuntimeException ซึ่งในกรณีคือ "Could not found comment " + id.

4. interface CommentRepository

```
1 package com.example.demo.repository;
2
3 import java.util.List;
4 import org.springframework.data.repository.CrudRepository;
5 import org.springframework.stereotype.Repository;
6 import com.example.demo.model.Comment;
7
8 @Repository
9 public interface CommentRepository extends CrudRepository<Comment, Long>{
10
11     List<Comment> findByPostId(Long postId);
12
13 }
```

อธิบายส่วนประกอบของ CommentRepository

ขยายจาก CrudRepository<Comment, Long>

- CrudRepository มาพร้อมกับเมธอดสำหรับการทำ CRUD โดยไม่จำเป็นต้องเขียนโค้ดเพิ่มเติม.
- ประเภทของข้อมูลที่จัดการคือ Comment (จากโมเดล Comment) และใช้ Long เป็นชนิดของ ID.

เมธอดเพิ่มเติม findByPostId(Long postId)

- เมธอดนี้เพิ่งเข้ามาเพื่อใช้ค้นหาคอมเมนต์ทั้งหมดที่มีความสัมพันธ์กับโพสต์ที่ระบุด้วย postId.

5. Class PostNotFoundException

```
1 package com.example.demo.repository;
2
3 public class PostNotFoundException extends RuntimeException{
4
5     public PostNotFoundException(Long id) {
6         super("Could not found Post "+id);
7     }
8 }
```

อธิบายโครงสร้างของ PostNotFoundException

การสืบทอดจาก RuntimeException

- คลาสนี้สืบทอดจาก RuntimeException ทำให้เป็น exception แบบ unchecked ซึ่งไม่จำเป็นต้องประกาศ throws หรือจัดการด้วย try-catch เมื่อใช้งาน.

คอนสตรัคเตอร์ PostNotFoundException(Long id)

- รับค่าพารามิเตอร์ id ของโพสต์ที่ไม่สามารถค้นหาได้
- เรียกใช้คอนสตรัคเตอร์ของ RuntimeException โดยส่งข้อความที่อธิบายข้อผิดพลาด (ในที่นี่คือ "Could not found Post " + id)

6. Class PostRepository

```
1 package com.example.demo.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4 import org.springframework.stereotype.Repository;
5 import com.example.demo.model.Post;
6
7 @Repository
8 public interface PostRepository extends CrudRepository<Post, Long>{
9
10 }
```

อธิบายส่วนประกอบของ Class PostRepository

@Repository

- ทำหน้าที่ประกาศว่าอินเทอร์เฟซนี้เป็น Spring Repository ซึ่งเป็นคอมโพเนนต์ที่เกี่ยวข้องกับการจัดการข้อมูล (Data Access Layer).

CrudRepository<Post, Long>

- CrudRepository เป็นอินเทอร์เฟซของ Spring Data ที่มีเมธอดพื้นฐานสำหรับการทำงานกับฐานข้อมูล

<Post, Long> หมายความว่า

- Post คือ entity ที่จะถูกจัดการ.
- Long คือประเภทของ primary key ที่ใช้ใน entity Post.

PostRepository

- อินเทอร์เฟชนี้สามารถใช้เพื่อเรียกใช้เมธอดการจัดการข้อมูลที่มาพร้อมกับ CrudRepository โดยไม่ต้องเขียนโค้ดเพิ่มเติม

7. Interface UserRepository

```

1 package com.example.demo.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.example.demo.config.UserDto;
6 import com.example.demo.model.User;
7
8
9
10 public interface UserRepository extends JpaRepository<User, Long> {
11
12     User findByUsername(String username);
13
14     User save(UserDto userDto);
15
16     User findByResetPasswordToken(String token);
17 }
```

อธิบายส่วนประกอบของ UserRepository

extends JpaRepository<User, Long>

- JpaRepository

เป็นอินเทอร์เฟซที่ให้ฟังก์ชันการทำงานที่สมบูรณ์สำหรับการทำงานกับฐานข้อมูลใน Spring Data JPA.

<User, Long>

- User คือ entity ที่จะถูกจัดการ.
- Long คือประเภทของ primary key ที่ใช้ใน entity User.

メธอดใน UserRepository

User findByUsername(String username)

- เมธอดนี้ใช้เพื่อค้นหาผู้ใช้ตามชื่อผู้ใช้ (username) ซึ่งจะคืนค่าผู้ใช้ (User) ที่ตรงกัน.

User save(UserDto userDto)

- เมธอดนี้กำหนดให้รับ UserDto เป็นพารามิเตอร์และคืนค่า User ที่ถูกบันทึก. แต่ตามมาตรฐานของ JpaRepository เมธอดนี้จะต้องใช้ User แทน UserDto เพื่อให้สามารถบันทึกลงฐานข้อมูลได้ตรงๆ.

User findByResetPasswordToken(String token)

- เมื่อดันน์ให้เพื่อค้นหาผู้ใช้ตาม resetPasswordToken, ซึ่งจะช่วยในการรีเซ็ตรหัสผ่าน.

3.3.3 สร้างแพ็คเกจ Service ภายในแพ็คเกจ Service ประกอบไปด้วย

1. class CatService

```

13 @Service
14 public class CatService {
15
16     @Autowired
17     CatRepository catRepository;
18
19
20     public List<Cat> getCats(){
21         List<Cat> cats = (List<Cat>) catRepository.findAll();
22         return cats;
23     }
24
25     public Cat getCatById(Long id) {
26         return catRepository.findById(id).orElseThrow(()->new CatNotFoundException(id));
27     }
28
29     public void saveCat(Cat cat) {
30         catRepository.save(cat);
31     }
32
33     public Cat addCat(Cat cat) {
34         catRepository.save(cat);
35         return cat;
36     }
37
38     public void deleteCat(Long id) {
39         catRepository.deleteById(id);
40     }
41
42     public Cat updateCat(Long id, Cat cat) {
43         Cat existingCat = catRepository.findById(id).get();
44         existingCat.setName(cat.getName());
45         existingCat.setAge(cat.getAge());
46         existingCat.setBreed(cat.getBreed());
47         existingCat.setCatPic(cat.getCatPic());
48         return catRepository.save(existingCat);
49     }
50 }
```

อธิบายส่วนประกอบ CatService

@Service

- การใช้ annotation @Service ทำให้ Spring รู้ว่ามีคลาสที่ให้บริการ ซึ่งจะช่วยในการจัดการ dependency injection.

@Autowired

- Annotation นี้ใช้สำหรับการทำ dependency injection ให้ catRepository ซึ่งเป็น instance ของ CatRepository เพื่อให้สามารถใช้ในการเข้าถึงข้อมูลของเม瓦ได้.

getCats()

- เมื่อดันน์จะคืนค่ารายการของเม瓦ทั้งหมดจากฐานข้อมูล โดยใช้ findAll() ของ catRepository.

getCatById(Long id)

- เมธอดนี้ใช้เพื่อค้นหาข้อมูลของแมวตาม ID. หากไม่พบแมวที่มี ID ที่ระบุไว้ จะโยน CatNotFoundException.

saveCat(Cat cat)

- เมธอดนี้ใช้เพื่อบันทึกแมวใหม่ลงในฐานข้อมูล โดยเรียกใช้ save() ของ catRepository.

addCat(Cat cat)

- เมธอดนี้มีหน้าที่คล้ายกับ saveCat โดยจะบันทึกแมวและคืนค่าแมวที่บันทึกไป.

deleteCat(Long id)

- เมธอดนี้ใช้สำหรับลบแมวที่มี ID ที่ระบุไว้จากฐานข้อมูล โดยใช้ deleteById() ของ catRepository.

updateCat(Long id, Cat cat)

- เมธอดนี้ใช้ในการอัปเดตข้อมูลของแมว ค้นหาแมวที่มี ID ที่ระบุไว้ หากพบแมว จะทำการตั้งค่าชื่อ, อายุ, พันธุ์ และภาพแมวใหม่ตามข้อมูลที่ส่งเข้ามาสุดท้ายจะบันทึกการเปลี่ยนแปลงด้วย save().

2. class CommentService

```

12@Service
13public class CommentService {
14
15    @Autowired
16    CommentRepository commentRepository;
17
18    public List<Comment> getCommentsByPostId(Long postId) {
19        return commentRepository.findById(postId);
20    }
21    public List<Comment> getComments(){
22        List<Comment> comments = (List<Comment>) commentRepository.findAll();
23        return comments;
24    }
25
26    public Comment getCommentById(Long id) {
27        return commentRepository.findById(id).orElseThrow(()->new CommentNotFoundException(id));
28    }
29
30    public void saveComment(Comment c) {
31        commentRepository.save(c);
32    }
33
34    public Comment addComment(Comment c) {
35        commentRepository.save(c);
36        return c;
37    }
38
39    public void deleteComment(Long id) {
40        commentRepository.deleteById(id);
41    }
42
43    public Comment updateComment(Long id, Comment c) {
44        Comment existingComment = commentRepository.findById(id).get();
45        existingComment.setContent(c.getContent());
46        return commentRepository.save(existingComment);
47    }
48}

```

อธิบายส่วนประกอบ CommentService

@Service

- Annotation นี้บอกให้ Spring รู้ว่ามีคลาสบริการที่ใช้ในการดำเนินการธุรกิจ และสามารถถูกจัดการด้วย Spring Container.

@Autowired

- ใช้เพื่อทำการ inject CommentRepository เข้ามาใน CommentService เพื่อให้สามารถเรียกใช้ฟังก์ชันที่จัดการกับข้อมูลความคิดเห็นในฐานข้อมูลได้.

getCommentsByPostId(Long postId):

- เมื่อดันนี้ใช้เพื่อดึงความคิดเห็นทั้งหมดที่เกี่ยวข้องกับโพสต์ที่มี postId โดยเรียกใช้ findByPostId(postId) จาก CommentRepository.

getComments()

- เมื่อดันนี้จะคืนค่ารายการความคิดเห็นทั้งหมดจากฐานข้อมูล โดยเรียกใช้ findAll().

getCommentById(Long id)

- เมื่อต้องค้นหาข้อมูลความคิดเห็นตาม ID ถ้าไม่พบความคิดเห็นที่มี ID ดังกล่าว จะโยน CommentNotFoundException.

saveComment(Comment c)

- เมื่อต้องบันทึกความคิดเห็นใหม่หรืออัปเดตความคิดเห็นที่มีอยู่ในฐานข้อมูล.

addComment(Comment c)

- เมื่อต้องมีฟังก์ชันคล้ายกับ saveComment
โดยจะบันทึกความคิดเห็นและคืนค่าความคิดเห็นที่บันทึกไป.

deleteComment(Long id)

- เมื่อต้องใช้ในการลบความคิดเห็นที่มี ID ที่ระบุจากฐานข้อมูล โดยเรียกใช้ deleteById()
ของ commentRepository.

updateComment(Long id, Comment c)

- เมื่อต้องใช้ในการอัปเดตความคิดเห็น ค้นหาความคิดเห็นที่มี ID ที่ระบุไว้ ถ้าพบความคิดเห็น^{จะทำการตั้งค่าเนื้อหาของความคิดเห็นใหม่ตามข้อมูลที่ส่งเข้ามา สุดท้ายบันทึกความคิดเห็นที่อัปเดตด้วย save().}

3. class PostService

```

12@Service
13public class PostService {
14
15    @Autowired
16    private PostRepository postRepository;
17
18
19    public List<Post> getPosts() {
20        return (List<Post>) postRepository.findAll();
21    }
22
23    public Post getPostById(Long id) {
24        return postRepository.findById(id).orElseThrow(() -> new PostNotFoundException(id));
25    }
26
27    public void savePost(Post post) {
28        postRepository.save(post);
29    }
30
31    public Post addPost(Post post) {
32        postRepository.save(post);
33        return post;
34    }
35
36    public void deletePost(Long id) {
37        postRepository.deleteById(id);
38    }
39
40    public Post updatePost(Long id, Post post) {
41        Post existingPost = postRepository.findById(id).orElseThrow(() -> new PostNotFoundException(id));
42        existingPost.setTitle(post.getTitle());
43        existingPost.setContent(post.getContent());
44        existingPost.setPostPic(post.getPostPic());
45        return postRepository.save(existingPost);
46    }
47}

```

อธิบายส่วนประกอบ PostService

@Service

- Annotation นี้บอกให้ Spring รู้ว่าเป็นคลาสบริการที่ใช้ในการจัดการธุรกิจ และสามารถถูกจัดการด้วย Spring Container.

@Autowired

- ใช้เพื่อทำการ inject PostRepository เข้ามาใน PostService เพื่อให้สามารถเรียกใช้ฟังก์ชันที่จัดการกับข้อมูลโพสต์ในฐานข้อมูลได้.

getPosts()

- เมื่อดันนี้คืนค่ารายการโพสต์ทั้งหมดจากฐานข้อมูล โดยเรียกใช้ findAll() จาก postRepository.

getPostById(Long id)

- เมื่อดันนี้คืนหาข้อมูลโพสต์ตาม ID ถ้าไม่พบโพสต์ที่มี ID ดังกล่าว จะโยน PostNotFoundException.

savePost(Post post)

- เมื่อดันนี้ใช้เพื่อบันทึกโพสต์ใหม่หรืออัปเดตโพสต์ที่มีอยู่ในฐานข้อมูล.

addPost(Post post)

- เมื่อดันนี้ทำหน้าที่เหมือนกับ savePost โดยบันทึกโพสต์ใหม่และคืนค่าโพสต์ที่บันทึกไป.

deletePost(Long id)

- เมื่อดันนี้ใช้ในการลบโพสต์ที่มี ID ที่ระบุจากฐานข้อมูล โดยเรียกใช้ deleteById() ของ postRepository.

updatePost(Long id, Post post)

- เมื่อดันนี้ใช้ในการอัปเดตโพสต์ ทำหน้าที่คืนหาโพสต์ที่มี ID ที่ระบุไว้ ถ้าพบโพสต์ จะทำการตั้งค่าเนื้อหา (title, content, postPic) ใหม่ตามข้อมูลที่ส่งเข้ามาและ บันทึกโพสต์ที่อัปเดตด้วย save().

4. class UserService

```
14@Service
15public class UserService {
16
17    @Autowired
18    PasswordEncoder passwordEncoder;
19
20    private UserRepository userRepository;
21
22    public UserService(UserRepository userRepository) {
23        super();
24        this.userRepository = userRepository;
25    }
26
27    public User findByUsername(String username) {
28        return userRepository.findByUsername(username);
29    }
30
31    public User save(UserDto userDto) {
32        User user = new User(userDto.getUsername(), passwordEncoder.encode(userDto.getPassword()), userDto.getEmail(),
33            userDto.getUserPic());
34        return userRepository.save(user);
35    }
36
37    public User getByResetPasswordToken(String token) {
38        return userRepository.findByResetPasswordToken(token);
39    }
40
41    public void updatePassword(User user, String newPassword) {
42        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
43        String encodedPassword = passwordEncoder.encode(newPassword);
44        user.setPassword(encodedPassword);
45        user.setResetPasswordToken(null);
46        userRepository.save(user);
47    }
48
49    public List<User> getUsers() {
50        return (List<User>) userRepository.findAll();
51    }
52
53    public User getUserById(Long id) {
54        return userRepository.findById(id).orElseThrow();
55    }
56
57    public User addUser(User user) {
58        return userRepository.save(user);
59    }
60
61    public void deleteUser(Long id) {
62        userRepository.deleteById(id);
63    }
64
65    // พิมพ์ชื่อคลาสที่ต้องการ
66    public boolean checkPassword(User user, String currentPassword) {
67        return passwordEncoder.matches(currentPassword, user.getPassword());
68    }
69
70    public User updateUser(Long id, User updatedUser) {
71        // อ่านค่าจากupdatedUser
72        User existingUser = userRepository.findById(id).orElseThrow();
73        existingUser.setUsername(updatedUser.getUsername());
74        existingUser.setEmail(updatedUser.getEmail());
75        if (!updatedUser.getPassword().isEmpty()) {
76            BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
77            existingUser.setPassword(passwordEncoder.encode(updatedUser.getPassword()));
78        }
79        existingUser.setUserPic(updatedUser.getUserPic());
80        userRepository.save(existingUser);
81        return userRepository.save(existingUser);
82    }
83}
```

อธิบายส่วนประกอบ UserService

@Service

- Annotation นี้บอกให้ Spring รู้ว่าคลาสนี้เป็น Service Component ที่ใช้สำหรับการจัดการโลจิกรกิจ.

@Autowired

- ใช้สำหรับการ inject PasswordEncoder และ UserRepository เข้ามาใน UserService.

Constructor

- ตัว constructor ถูกใช้เพื่อ inject UserRepository เข้ามา.

findByUsername(String username)

- ค้นหาผู้ใช้จากชื่อผู้ใช้ที่นำมาโดยใช้ฟังก์ชันจาก UserRepository.

save(UserDto userDto)

- สร้างผู้ใช้ใหม่จากข้อมูลที่อยู่ใน UserDto รหัสผ่านจะถูกเข้ารหัสก่อนบันทึก จากนั้นบันทึกผู้ใช้ใหม่ลงในฐานข้อมูล.

getByResetPasswordToken(String token)

- ค้นหาผู้ใช้ตาม token สำหรับการรีเซ็ตรหัสผ่าน.

updatePassword(User user, String newPassword)

- อัปเดตรหัสผ่านของผู้ใช้ รหัสผ่านใหม่จะถูกเข้ารหัสและบันทึกลงในฐานข้อมูล.
- รีเซ็ต token สำหรับการรีเซ็ตรหัสผ่าน.

getUsers()

- คืนค่ารายการผู้ใช้ทั้งหมดจากฐานข้อมูล.

getUserById(Long id)

- ค้นหาผู้ใช้ตาม ID ถ้าไม่พบจะโยนข้อผิดพลาด.

addUser(User user)

- บันทึกผู้ใช้ใหม่ลงในฐานข้อมูล.

deleteUser(Long id)

- ลบผู้ใช้ที่มี ID ที่ระบุออกจากฐานข้อมูล.

checkPassword(User user, String currentPassword)

- ตรวจสอบรหัสผ่านปัจจุบันของผู้ใช้ โดยใช้ matches เพื่อเปรียบเทียบรหัสผ่านที่ป้อน กับรหัสผ่านที่เข้ารหัสในฐานข้อมูล.

updateUser(Long id, User updatedUser)

- อัปเดตข้อมูลผู้ใช้
- ค้นหาผู้ใช้ตาม ID
- อัปเดตชื่อผู้ใช้, อีเมล, รูปภาพผู้ใช้
- ถ้ารหัสผ่านใหม่ไม่ว่างเปล่า ก็ให้เข้ารหัสและอัปเดต

3.3.4 สร้างแพ็คเกจ Config ภายในแพ็คเกจ Config ประกอบไปด้วย

1. class CustomUserDetail

```

8 public class CustomUserDetails implements UserDetails {
9
10    private Long id;
11    private String username;
12    private String password;
13    private Collection<? extends GrantedAuthority> authorities;
14    private String email;
15    private String userPic;
16
17    public CustomUserDetails(String username, String password,
18        Collection<? extends GrantedAuthority> authorities, String email, String userPic) {
19        super();
20        this.username = username;
21        this.password = password;
22        this.authorities = authorities;
23        this.email = email;
24        this.userPic = userPic;
25    }
26
27    public Long getId() {
28        return id;
29    }
30
31    public void setId(Long id) {
32        this.id = id;
33    }
34
35    public String getUsername() {
36        return username;
37    }
38
39    public void setUsername(String username) {
40        this.username = username;
41    }
42
43    public String getPassword() {
44        return password;
45    }

```

```

public void setPassword(String password) {
    this.password = password;
}

public Collection<? extends GrantedAuthority> getAuthorities() {
    return authorities;
}

public void setAuthorities(Collection<? extends GrantedAuthority> authorities) {
    this.authorities = authorities;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getUserPic() {
    return userPic;
}

public void setUserPic(String userPic) {
    this.userPic = userPic;
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

35   @Override
36   public boolean isCredentialsNonExpired() {
37       return true;
38   }
39
40   @Override
41   public boolean isEnabled() {
42       return true;
43   }
44 }
```

อธิบายส่วนประกอบ CustomUserDetails

CustomUserDetails Constructor

- รับข้อมูลของผู้ใช้ เช่น ชื่อผู้ใช้ รหัสผ่าน สิทธิ์การเข้าถึง (authorities) อีเมล และรูปภาพผู้ใช้ โดยข้อมูลเหล่านี้จะถูกกำหนดให้กับฟิลด์ของคลาส

ฟิลด์

- Long id หมายเลข ID ของผู้ใช้

- String username หมายถึง ชื่อผู้ใช้
- String password หมายถึง รหัสผ่าน (เข้ารหัส)
- Collection<? extends GrantedAuthority> authorities หมายถึง สิทธิ์การเข้าถึงของผู้ใช้
- String email หมายถึง อีเมลของผู้ใช้
- String userPic หมายถึง รูปภาพของผู้ใช้

Getter และ Setter

- มี getter และ setter สำหรับทุกฟิลด์เพื่อให้สามารถเข้าถึงและปรับปรุงข้อมูลผู้ใช้ได้

UserDetails Interface Implementation

- isAccountNonExpired() คืนค่า true หมายความว่าบัญชีผู้ใช้ไม่หมดอายุ
- isAccountNonLocked() คืนค่า true หมายความว่าบัญชีผู้ใช้ไม่ได้ถูกล็อก
- isCredentialsNonExpired() คืนค่า true หมายความว่ารหัสผ่านของผู้ใช้ไม่หมดอายุ
- isEnabled() คืนค่า true หมายความว่าบัญชีผู้ใช้เปิดใช้งาน

2. class CustomUserDetailsService

```

16 @Service
17 public class CustomUserDetailsService implements UserDetailsService {
18
19     private UserRepository userRepository;
20
21     public CustomUserDetailsService(UserRepository userRepository) {
22         super();
23         this.userRepository = userRepository;
24     }
25
26     @Override
27     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
28
29         User user = userRepository.findByUsername(username);
30         if (user == null) {
31             throw new UsernameNotFoundException("Username or Password not found");
32         }
33         return new CustomUserDetails(user.getUsername(), user.getPassword(), authorities(),
34             user.getEmail(), user.getUserPic());
35     }
36
37     public Collection<? extends GrantedAuthority> authorities() {
38         return Arrays.asList(new SimpleGrantedAuthority("USER"));
39     }
40 }
```

อธิบายส่วนประกอบ CustomUserDetailsService

การประกาศคลาส

- คลาส CustomUserDetailsService ใช้ @Service เพื่อบ่งบอกว่าเป็นบริการที่จัดการเกี่ยวกับการตรวจสอบสิทธิ์ผู้ใช้
-

ฟิล์ด

- UserRepository userRepository หมายถึง ตัวแปรที่ใช้เข้าถึงฐานข้อมูลผู้ใช้

Constructor

- CustomUserDetailsService(UserRepository userRepository): ใช้สำหรับการฉีด (inject) UserRepository ผ่าน constructor

Method loadUserByUsername

- ใช้เพื่อโหลดผู้ใช้จากฐานข้อมูลตามชื่อผู้ใช้ (username)
- หากไม่พบผู้ใช้ จะโยน UsernameNotFoundException พร้อมข้อความว่า "Username or Password not found"
- หากพบผู้ใช้ จะสร้างและคืนค่า CustomUserDetails โดยส่งข้อมูลของผู้ใช้ เช่น ชื่อผู้ใช้ รหัสผ่าน สิทธิ์การเข้าถึง (authorities) อีเมล และรูปภาพผู้ใช้

Method authorities

- คืนค่า Collection<? extends GrantedAuthority> ซึ่งในที่นี้จะคืนค่า SimpleGrantedAuthority ที่กำหนดสิทธิ์การเข้าถึงเป็น "USER"

3. class SecurityConfig

```
14 @Configuration
15 @EnableWebSecurity
16 public class SecurityConfig {
17
18     @Autowired
19     CustomUserDetailsService customUserDetailsService;
20
21     @Bean
22     public static PasswordEncoder passwordEncoder() {
23         return new BCryptPasswordEncoder();
24     }
25
26     @Bean
27     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
28         http.csrf().disable()
29             .authorizeHttpRequests()
30                 .requestMatchers("/register", "/login").permitAll() // อนุญาต register และ login ให้ทุกคนได้เข้าถึง
31                 .requestMatchers("/*").authenticated() // วิ่ง home ต้องลงชื่อเข้าใช้
32                 .anyRequest().authenticated() // ทุกๆ request ต้องลงชื่อเข้าใช้
33             .and().formLogin()
34                 .loginPage("/login")
35                 .loginProcessingUrl("/login")
36                 .defaultSuccessUrl("/", true) // วนกันรอบๆ redirect ไปยัง /home
37             .permitAll()
38             .and().logout()
39                 .invalidateHttpSession(true)
40                 .clearAuthentication(true)
41                 .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
42                 .logoutSuccessUrl("/login/logout")
43             .permitAll();
44         return http.build();
45     }
46
47     @Autowired
48     public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
49         auth.userDetailsService(customUserDetailsService).passwordEncoder(passwordEncoder());
50     }
51 }
```

อธิบายส่วนประกอบ SecurityConfig

การประมวลผลคลาส

- ใช้ `@Configuration` เพื่อระบุว่ามีคือคลาสการตั้งค่าของ Spring
- ใช้ `@EnableWebSecurity` เพื่อเปิดใช้งานการรักษาความปลอดภัยของเว็บใน Spring Security

ฟิลด์

- `CustomUserDetailsService customUserDetailsService:` ตัวแปรที่ใช้สำหรับการฉีด `CustomUserDetailsService`

Bean สำหรับ PasswordEncoder

- สร้าง bean สำหรับการเข้ารหัสผ่านโดยใช้ `BCryptPasswordEncoder`

CSRF Protection

- ปิดการป้องกัน CSRF (Cross-Site Request Forgery) ด้วย `http.csrf().disable()`

Authorization Rules

- หน้า `/register` และ `/login` สามารถเข้าถึงได้โดยไม่ต้องล็อกอิน
- หน้า `/` ต้องล็อกอินก่อนจึงจะเข้าถึงได้
- ทุกหน้าต้องล็อกอินก่อนจึงจะเข้าถึงได้

Form Login

- กำหนดหน้าเข้าสู่ระบบเป็น `/login`
- กำหนด URL สำหรับการประมวลผลการล็อกอินเป็น `/login`
- หลังจากล็อกอินสำเร็จจะถูกเปลี่ยนเส้นทางไปยังหน้า `/`

Logout Configuration

- ล็อกเอาท์จะทำการลบ session และ clear การยืนยันตัวตน
- เมื่อออกจากระบบสำเร็จจะถูกเปลี่ยนเส้นทางไปยัง `/login?logout`

Global Authentication Configuration

- ใช้สำหรับการตั้งค่าการยืนยันตัวตนทั่วทั้งแอปพลิเคชัน โดยการใช้ `CustomUserDetailsService` และ `PasswordEncoder` ที่สร้างขึ้น

4. Class UserDto

```
3 public class UserDto {  
4  
5     private String username;  
6     private String password;  
7     private String email;  
8     private String userPic;  
9  
10    public UserDto(String username, String password, String email, String userPic) {  
11        super();  
12        this.username = username;  
13        this.password = password;  
14        this.email = email;  
15        this.userPic = userPic;  
16    }  
17  
18    public String getUsername() {  
19        return username;  
20    }  
21  
22    public void setUsername(String username) {  
23        this.username = username;  
24    }  
25  
26    public String getPassword() {  
27        return password;  
28    }  
29  
30    public void setPassword(String password) {  
31        this.password = password;  
32    }  
33  
34    public String getEmail() {  
35        return email;  
36    }  
37  
38    public void setEmail(String email) {  
39        this.email = email;  
40    }  
41  
42    public String getUserPic() {  
43        return userPic;  
44    }  
45  
46    public void setUserPic(String userPic) {  
47        this.userPic = userPic;  
48    }  
49  
50    @Override  
51    public String toString() {  
52        return "UserDto username=" + username + ", password=" + password + ", email=" + email  
53                + ", userPic=" + userPic + "]";  
54    }  
55}
```

อธิบายส่วนประกอบ UserDto

Fields (ตัวแปรภายใน)

- Username หมายถึง ชื่อผู้ใช้
- Password หมายถึง รหัสผ่าน
- Email หมายถึง อีเมลของผู้ใช้

- userPic หมายถึง รูปโปรไฟล์ของผู้ใช้

Constructor

- คอนสตรัคเตอร์นี้ใช้สำหรับสร้างอ็อบเจกต์ของ UserDto โดยกำหนดค่าทั้งสี่ฟิลด์ (username, password, email, userPic) เมื่อสร้างอ็อบเจกต์ใหม่

Getters และ Setters

- getUsername(), setUsername(String username) พังก์ชันสำหรับเข้าถึงและกำหนดค่า username
- getPassword(), setPassword(String password) พังก์ชันสำหรับเข้าถึงและกำหนดค่า password
- getEmail(), setEmail(String email) พังก์ชันสำหรับเข้าถึงและกำหนดค่า email
- getUserPic(), setUserPic(String userPic) พังก์ชันสำหรับเข้าถึงและกำหนดค่า userPic

เมธอด toString()

- เมธอดนี้ใช้สำหรับสร้างข้อความแสดงผลที่适合ดูกใน การพิมพ์ออกมา โดยแสดงค่าของ username, password, email, และ userPic
- อาจมีการใช้ในกรณีที่ต้องการแสดงข้อมูลของผู้ใช้ในรูปแบบสตริง

3.3.5 สร้างแพ็คเกจ Controller ภายในแพ็คเกจ Controller ประกอบไปด้วย

1. class CommentController

```
@Controller
public class CommentController {
    @Autowired
    UserService userService;

    @Autowired
    PostService postService;

    @Autowired
    CommentService commentService;

    @PostMapping("/addComment/{postId}")
    public String addComment(@PathVariable("postId") Long postId, @ModelAttribute Comment comment,
                           Principal principal) {
        // Get the current post and user
        Post currentPost = postService.getPostById(postId);
        User currentUser = userService.findByUsername(principal.getName());

        // Associate the comment with the post and user
        comment.setPost(currentPost);
        comment.setUser(currentUser);

        // Save the comment
        commentService.saveComment(comment);

        // Redirect back to the post view
        return "redirect:/viewPost/" + postId;
    }

    @GetMapping("/editComment/{id}")
    public String editComment(@PathVariable Long id, Model model) {
        Comment comment = commentService.getCommentById(id);
        model.addAttribute("comment", comment);
        return "editComment"; // Create an editComment.html template
    }

    @GetMapping("/deleteComment/{id}")
    public String deleteComment(@PathVariable Long id) {
        Comment comment = commentService.getCommentById(id);
        Long postId = comment.getPost().getId();
        commentService.deleteComment(id);
        return "redirect:/viewPost/" + postId;
    }
}
```

อธิบายส่วนประกอบ CommentController

Annotations

- @Controller: ทำให้คลาสนี้ทำงานที่เป็นคอนโทรลเลอร์ใน Spring MVC

- @Autowired: ใช้เพื่อให้ Spring สามารถจัด (inject) Dependencies เข้าไปในตัวแปรที่กำหนดไว้

Dependencies

- UserService: ใช้ในการเข้าถึงข้อมูลผู้ใช้
- PostService: ใช้ในการเข้าถึงข้อมูลโพสต์
- CommentService: ใช้ในการจัดการความคิดเห็น

Method: addComment

- @PostMapping("/addComment/{postId}"): กำหนดให้เมธอดนี้ทำงานเมื่อมีการส่ง HTTP POST request ไปยัง URL /addComment/{postId}

Parameters

- @PathVariable("postId") Long postId: ดึงค่า postId จาก URL เพื่อรับบุโพสต์ที่ต้องการแสดงความคิดเห็น
- @ModelAttribute Comment comment: ดึงข้อมูลความคิดเห็นจากแบบฟอร์มที่ผู้ใช้ส่งมา
- Principal principal: ใช้เพื่อดึงข้อมูลผู้ใช้ที่ล็อกอินอยู่ในปัจจุบัน

Logic

- ดึงข้อมูลโพสต์ปัจจุบันจาก PostService โดยใช้ postId
- ดึงข้อมูลผู้ใช้ปัจจุบันจาก UserService โดยใช้ชื่อผู้ใช้จาก Principal
- เชื่อมโยงความคิดเห็นกับโพสต์และผู้ใช้
- บันทึกความคิดเห็นโดยใช้ CommentService
- ทำการ redirect กลับไปที่หน้าดูโพสต์ที่แสดงความคิดเห็น

Method editComment

- @GetMapping("/editComment/{id}"): กำหนดให้เมธอดนี้ทำงานเมื่อมีการส่ง HTTP GET request ไปยัง URL /editComment/{id} โดย id คือ ID ของความคิดเห็นที่ต้องการแก้ไข

Parameters:

- @PathVariable Long id: ดึงค่า id ของความคิดเห็นจาก URL
- Model model: ใช้เพื่อส่งข้อมูลไปยัง view

Logic:

- commentService.getCommentById(id): ดึงความคิดเห็นจากฐานข้อมูลตาม id ที่ได้รับจาก URL
- model.addAttribute("comment", comment): เพิ่มความคิดเห็นที่ดึงมาใน Model เพื่อส่งไปให้ view ที่จะแสดงข้อมูลความคิดเห็น
- return "editComment": ส่งชื่อของเทมเพลต editComment.html ซึ่งจะแสดงฟอร์มสำหรับแก้ไขความคิดเห็น

Method deleteComment

Annotation:

- @GetMapping("/deleteComment/{id}"): กำหนดให้เมธอดนี้ทำงานเมื่อมีการส่ง HTTP GET request ไปยัง URL /deleteComment/{id} โดย id คือ ID ของความคิดเห็นที่ต้องการลบ

Parameters:

- @PathVariable Long id: ดึงค่า id ของความคิดเห็นจาก URL

Logic:

- commentService.getCommentById(id): ดึงความคิดเห็นจากฐานข้อมูลตาม id
- Long postId = comment.getPost().getId(): ดึง postId จากความคิดเห็นที่เกี่ยวข้องกับโพสต์ เพื่อใช้ในการ redirect กลับไปยังหน้าดูโพสต์หลังจากลบความคิดเห็น
- commentService.deleteComment(id): ลบความคิดเห็นออกจากฐานข้อมูล
- return "redirect:/viewPost/" + postId

2. class PostController

```
28 @Controller
29 public class PostController {
30
31     @Autowired
32     private UserDetailsService userDetailsService;
33
34     @Autowired
35     private UserService userService;
36
37
38     @Autowired
39     PostService postService;
40
41     @Autowired
42     CommentService commentService;
43
44
45     @GetMapping("/myProfile")
46     public String showProfile(@ModelAttribute Post post, Principal principal, Model model) {
47         User user = userService.findByUsername(principal.getName());
48         model.addAttribute("user", user);
49         return "myProfile";
50     }
51
52     @GetMapping("/addPost")
53     public String addPostForm(Model model) {
54         model.addAttribute("post", new Post());
55         return "addPost";
56     }
57
58
59     @PostMapping("/addPost")
60     public String addPost(@ModelAttribute Post post, Principal principal) {
61         // ရှိယူလိုက်ပါ။
62         User user = userService.findByUsername(principal.getName());
63
64         // တင်ယူလိုက်ပါ။
65         post.setUser(user);
66         postService.savePost(post);
67         return "redirect:/";
68     }
69
70     @GetMapping("/editPost/{id}")
71     public String editPostForm(@PathVariable Long id, Model model) {
72         Post post = postService.getPostById(id);
73         model.addAttribute("post", post);
74         return "editPost";
75     }
76
77     @PostMapping("/updatePost/{id}")
78     public String updatePost(@PathVariable Long id, @ModelAttribute Post post, Model model) {
79         postService.updatePost(id, post);
80         return "redirect:/myProfile";
81     }
82
83     @GetMapping("/deletePost/{id}")
84     public String deletePost(@PathVariable Long id) {
85         postService.deletePost(id);
86         return "redirect:/myProfile";
87     }
88
89
90     @GetMapping("/viewPost/{id}")
91     public String viewPost(@PathVariable Long id, Model model, Principal principal) {
92         Post post = postService.getPostById(id);
93         List<Comment> comments = commentService.getCommentsByPostId(id);
94         User user = post.getUser();
95         // ဖော်ပြန်လိုက်မယ့်အတွက် မျှမှန်ပေးခြင်းဖြစ်ပါ။
96         User myUser = userService.findByUsername(principal.getName());
97         model.addAttribute("myUser", myUser);
98
99         model.addAttribute("post", post);
100        model.addAttribute("comments", comments);
101        model.addAttribute("user", user);
102
103        return "viewPost";
104    }
```

ခုပ္ပန်အသေဆုံးသော PostController

Annotations

- @Controller: ทำให้คลาสนี้ทำงานที่เป็นคอนโทรลเลอร์ใน Spring MVC
- @Autowired: ใช้เพื่อให้ Spring สามารถ inject Dependencies เข้าไปในตัวแปรที่กำหนดไว้

Dependencies

- UserDetailsService: ใช้ในการจัดการข้อมูลผู้ใช้
- UserService: ใช้ในการเข้าถึงข้อมูลผู้ใช้
- PostService: ใช้ในการจัดการโพสต์
- CommentService: ใช้ในการจัดการความคิดเห็น

Method: showProfile

- @GetMapping("/myProfile"): แสดงหน้าโปรไฟล์ของผู้ใช้
- ดึงข้อมูลผู้ใช้จาก UserService โดยใช้อัตราผู้ใช้จาก Principal และส่งข้อมูลไปยังโมเดลเพื่อแสดงในวิว myProfile

Method: addPostForm

- @GetMapping("/addPost"): แสดงฟอร์มเพื่อสร้างโพสต์ใหม่
- สร้างอ็อบเจกต์ Post ใหม่และส่งไปยังโมเดลสำหรับการแสดงในวิว addPost

Method: addPost

- @PostMapping("/addPost"): รับการส่งข้อมูลโพสต์ใหม่จากฟอร์ม
- ดึงข้อมูลผู้ใช้ที่ล็อกอินอยู่ และตั้งค่าผู้ใช้ในโพสต์ จากนั้นบันทึกโพสต์โดยใช้ PostService
- ทำการ redirect ไปที่หน้าแรก (/)

Method: editPostForm

- @GetMapping("/editPost/{id}"): แสดงฟอร์มเพื่อแก้ไขโพสต์ที่มีอยู่

- ดึงข้อมูลโพสต์จาก PostService และส่งไปยังโมเดลสำหรับการแสดงในวิว editPost

Method: updatePost

- @PostMapping("/updatePost/{id}"): รับการส่งข้อมูลโพสต์ที่ถูกแก้ไข
- อัปเดตโพสต์ในฐานข้อมูลโดยใช้ PostService และทำการ redirect ไปที่หน้าโปรไฟล์ผู้ใช้ (/myProfile)

Method: deletePost

- @GetMapping("/deletePost/{id}"): ลบโพสต์ที่ระบุโดย id
- ใช้ PostService ในการลบโพสต์ และทำการ redirect ไปที่หน้าโปรไฟล์ผู้ใช้

Method: viewPost

- Annotation @GetMapping("/viewPost/{id}"):
 - ระบุว่าเมื่อคนนี้จะตอบสนองต่อคำขอ HTTP GET ที่ URL /viewPost/{id} โดย {id} คือพารามิเตอร์ที่แทนที่ด้วย ID ของโพสต์ที่ต้องการดู

Parameters

- @PathVariable Long id: รับค่า ID ของโพสต์จาก URL
- Model model: ใช้สำหรับเก็บข้อมูลที่จะส่งไปยังเทมเพลต Thymeleaf
- Principal principal ใช้เพื่อดึงข้อมูลผู้ใช้ที่ล็อกอินอยู่ (ในกรณีคือลงชื่อเข้าใช้)
- Post post = postService.getPostById(id); เรียกใช้บริการเพื่อดึงโพสต์ตาม ID ที่ได้รับ
- List<Comment> comments = commentService.getCommentsByPostId(id); เรียกใช้บริการเพื่อดึงความคิดเห็นที่เกี่ยวข้องกับโพสต์นั้น
- User user = post.getUser() ดึงข้อมูลผู้ใช้ที่เป็นเจ้าของโพสต์
- User myUser = userService.findByUsername(principal.getName()): ดึงข้อมูลผู้ใช้ที่ล็อกอินอยู่ โดยใช้ชื่อผู้ใช้จาก principal
- model.addAttribute("myUser", myUser); เพิ่มข้อมูลผู้ใช้ที่ล็อกอินอยู่ในโมเดล
- model.addAttribute("post", post); เพิ่มโพสต์ที่ดึงมาในโมเดล
- model.addAttribute("comments", comments); เพิ่มความคิดเห็นที่ดึงมาในโมเดล

- model.addAttribute("user", user); เพิ่มข้อมูลผู้ใช้ที่เป็นเจ้าของโพสต์ในโมเดล
- return "viewPost"; ส่งคืนชื่อของเทมเพลตที่ต้องการให้แสดงผล ใน viewPost)

3. class CatController

```

@Controller
public class CatController {

    @Autowired
    private UserService userService;

    @Autowired
    CatService catService;

    @GetMapping("/addCat")
    public String addCatForm(Model model) {
        model.addAttribute("cat", new Cat());
        return "addCat";
    }

    @PostMapping("/addCat")
    public String addCat(@ModelAttribute Cat cat, Principal principal) {
        User user = userService.findByUsername(principal.getName());
        cat.setUser(user);
        catService.saveCat(cat);
        return "redirect:/myProfile";
    }

    @GetMapping("/editCat/{id}")
    public String editCatForm(@PathVariable Long id, Model model) {
        Cat cat = catService.getCatById(id);
        model.addAttribute("cat", cat);
        return "editCat";
    }

    @PostMapping("/updateCat/{id}")
    public String updateCat(@PathVariable Long id, @ModelAttribute Cat cat, Model model) {
        catService.updateCat(id, cat);
        return "redirect:/myProfile";
    }

    @GetMapping("/deleteCat/{id}")
    public String deleteCat(@PathVariable Long id) {
        catService.deleteCat(id);
        return "redirect:/myProfile";
    }
}

```

```

    @GetMapping("/catDetail/{id}")
    public String viewCat(@PathVariable Long id, Model model, Principal principal) {
        User myUser = userService.findByUsername(principal.getName());
        Cat cat = catService.getCatById(id);
        User user = cat.getUser();
        model.addAttribute("cat", cat);
        model.addAttribute("user", user);

        if(myUser.getId()==user.getId()) {
            return "viewCatDetailbyUser";
        }else {
            return "viewCatDetail";
        }

    }
}

```

อธิบายการทำงานของ CatController

Dependencies

- UserService userService: ใช้สำหรับดึงข้อมูลผู้ใช้จากฐานข้อมูล โดยเฉพาะผู้ใช้ที่ล็อกอินอยู่ผ่าน Principal
- CatService catService: ใช้สำหรับจัดการข้อมูลแมว เช่น การเพิ่ม ลบ อัปเดต และดึงข้อมูลแมว จากฐานข้อมูล

Method: addCatForm

- @GetMapping("/addCat"): แสดงหน้า HTML ฟอร์มสำหรับเพิ่มแมวใหม่ โดยเพิ่ม Cat ใหม่เข้าไปในโมเดลเพื่อแสดงในฟอร์ม
- เพิ่มอีบเจกต์ Cat ใหม่เข้าไปในโมเดลและคืนค่าไปยังวิว addCat.html

Method: addCat

- @PostMapping("/addCat"): รับข้อมูลจากฟอร์มเพิ่มแมวใหม่
- ดึงข้อมูลผู้ใช้ที่ล็อกอินอยู่จาก Principal
- กำหนดผู้ใช้ให้กับแมวที่เพิ่งสร้าง และบันทึกแมวใหม่ลงในฐานข้อมูลผ่าน CatService
- หลังจากบันทึกสำเร็จ จะ redirect กลับไปที่หน้าโปรไฟล์ผู้ใช้ (/myProfile)

Method: editCatForm

- @GetMapping("/editCat/{id}"): แสดงฟอร์มสำหรับแก้ไขข้อมูลแมว โดยระบุ id
- ดึงข้อมูลแมวจาก CatService ตาม id และส่งในโมเดลเพื่อแสดงในฟอร์ม editCat.html

Method: updateCat

- @PostMapping("/updateCat/{id}"): รับข้อมูลที่แก้ไขจากฟอร์ม
- อัปเดตข้อมูลแมวในฐานข้อมูลผ่าน CatService โดยระบุ id
- หลังจากอัปเดตสำเร็จ จะ redirect กลับไปที่หน้าโปรไฟล์ผู้ใช้ (/myProfile)

Method: deleteCat

- @GetMapping("/deleteCat/{id}"): ลบข้อมูลแมวจากฐานข้อมูลตาม id
- ลบแมวตาม id ผ่าน CatService
- หลังจากลบสำเร็จ จะ redirect กลับไปที่หน้าโปรไฟล์ผู้ใช้ (/myProfile)

Method: viewCat

- @GetMapping("/catDetail/{id}"): แสดงรายละเอียดของแมวตาม id
- ดึงข้อมูลผู้ใช้ที่ล็อกอินอยู่จาก Principal
- ดึงข้อมูลแมวตาม id จาก CatService
- ดึงข้อมูลเจ้าของแมว
- ใส่ข้อมูลแมวและเจ้าของแมวลงในโมเดลเพื่อแสดงรายละเอียด
- เช็คว่าแมวที่ดูเป็นของผู้ใช้ที่ล็อกอินอยู่หรือไม่ ถ้าเป็นเจ้าของ จะส่งคืนเพลต viewCatDetailbyUser.html เพื่อให้ผู้ใช้แก้ไขหรือจัดการได้ แต่ถ้าไม่ใช่เจ้าของ จะส่งคืนเพลต viewCatDetail.html สำหรับแสดงรายละเอียดเท่านั้น

4. class UserController

```
25 @Controller
26 public class UserController {
27
28     @Autowired
29     private UserDetailsService userDetailsService;
30
31     @Autowired
32     private UserService userService;
33
34     @Autowired
35     PostService postService;
36
37     @GetMapping("/")
38     public String home(Model model, Principal principal) {
39         if (principal == null) {
40             return "redirect:/login"; // หากยังไม่ได้รับ token ให้ redirect ไปที่หน้า login
41         }
42         UserDetails user = userDetailsService.loadUserByUsername(principal.getName());
43         model.addAttribute("user", user);
44         List<User> userList = userService.getUsers().stream().filter(u -> u.getUsername() != null)
45             .collect(Collectors.toList());
46
47         model.addAttribute("users", userList);
48         List<Post> posts = postService.getPosts();
49         model.addAttribute("posts", posts);
50         return "home";
51     }
52
53     @GetMapping("/login")
54     public String login(Model model, UserDto userDto) {
55         model.addAttribute("user", userDto);
56         return "login";
57     }
58
59     @GetMapping("/register")
60     public String register(Model model, UserDto userDto) {
61         model.addAttribute("user", userDto);
62         return "register";
63     }
64
65     @PostMapping("/register")
66     public String registerSave(@ModelAttribute("user") UserDto userDto, Model model) {
67         User user = userService.findByUsername(userDto.getUsername());
68         if (user != null) {
69             model.addAttribute("Userexist", user);
70             return "register";
71         }
72         userService.save(userDto);
73         return "redirect:/register?success";
74     }
75
76     // View Profile
77     @GetMapping("/user/{id}")
78     public String getViewProfile(@PathVariable Long id, Model model, Principal principal) {
79         User myUser = userService.findByUsername(principal.getName());
80         User user = userService.getUserById(id);
81         model.addAttribute("user", user);
82         model.addAttribute("myUser", myUser);
83         return "viewUser";
84     }
85
86     @GetMapping("/editUser/{id}")
87     public String editUser(@PathVariable Long id, Model model) {
88         User user = userService.getUserById(id);
89         model.addAttribute("user", user);
90         return "editUser";
91     }
92 }
```

```

3 @PostMapping("/updateUser/{id}")
4 public String updateUser(@PathVariable Long id, @ModelAttribute User user,
5                         @ModelAttribute("currentPassword") String currentPassword, Principal principal, HttpSession session,
6                         Model model) {
7     User myUser = userService.findByUsername(principal.getName());
8
9     if (myUser.getId() == id) {
10         if (userService.checkPassword(myUser, currentPassword)) {
11             userService.updateUser(id, user);
12             session.setAttribute("user", userService.getUserById(id));
13             return "redirect:/myProfile";
14         } else {
15             model.addAttribute("passwordError", "Current password is incorrect.");
16             return "editUser";
17         }
18     } else {
19         return "error/403";
20     }
21 }
22
23

```

อธิบายส่วนประกอบของ UserController

Annotations

- @Controller แสดงว่าคลาสนี้เป็นคอนโทรลเลอร์ที่จัดการคำขอ HTTP
- @Autowired ใช้เพื่อให้ Spring สามารถฉีด (inject) Dependencies เข้าไปในตัวแปรที่กำหนดไว้

Dependencies

- UserDetailsService ใช้ในการโหลดข้อมูลผู้ใช้
- UserService ใช้ในการเข้าถึงข้อมูลผู้ใช้
- PostService ใช้ในการจัดการโพสต์

Method home

- @GetMapping("/") แสดงหน้าแรก
- หากผู้ใช้งานไม่ได้ล็อกอิน (principal เป็น null), จะ redirect ไปยังหน้าเข้าสู่ระบบ
- ดึงข้อมูลผู้ใช้ที่ล็อกอินอยู่และโพสต์ทั้งหมดจากบริการที่เกี่ยวข้อง และส่งข้อมูลไปยังโมเดลสำหรับการแสดงในวิว home

Method login

- `@GetMapping("/login")` แสดงหน้าเข้าสู่ระบบ
- สร้างอ็อบเจกต์ `UserDto` ใหม่และส่งไปยังโมเดลสำหรับการแสดงในวิว `Login`

Method: register

- `@GetMapping("/register")` แสดงฟอร์มลงทะเบียน
- ส่ง `UserDto` ไปยังโมเดลเพื่อแสดงในวิว `register`

Method: registerSave

- `@PostMapping("/register")` รับข้อมูลการลงทะเบียนที่ถูกส่งมาจากฟอร์ม
- ตรวจสอบว่าชื่อผู้ใช้มีอยู่ในระบบแล้วหรือไม่ หากมีจะแสดงข้อความว่า "User exists"
- หากไม่มี จะบันทึกผู้ใช้ใหม่และ redirect ไปยังหน้า `register` พร้อมกับพารามิเตอร์ `success`

Method getViewProfile

- `@GetMapping("/user/{id}")` แสดงโปรไฟล์ของผู้ใช้ที่ระบุโดย `id`
- ดึงข้อมูลผู้ใช้จาก `UserService` และส่งไปยังโมเดลเพื่อแสดงในวิว `viewUser`

Method editUser

- `@GetMapping("/editUser/{id}")` แสดงฟอร์มสำหรับการแก้ไขโปรไฟล์ของผู้ใช้ที่ระบุโดย `id`
- ดึงข้อมูลผู้ใช้และส่งไปยังโมเดลสำหรับการแสดงในวิว `editUser`

Method updateUser

- `@PostMapping("/updateUser/{id}")` รับการส่งข้อมูลจากฟอร์มแก้ไขโปรไฟล์
- ตรวจสอบว่าผู้ใช้ที่ล็อกอินอยู่มี `id` ตรงกับ `id` ที่ต้องการแก้ไขหรือไม่
- ตรวจสอบรหัสผ่านปัจจุบันของผู้ใช้ หากถูกต้องจะอัปเดตข้อมูลผู้ใช้และเก็บข้อมูลใน `HttpSession`

- หากรหัสผ่านปัจจุบันไม่ถูกต้อง จะแสดงข้อความผิดพลาดและกลับไปที่ฟอร์มแก้ไข
- หากไม่ใช่ผู้ใช้ที่ล็อกอิน จะ redirect ไปที่หน้าข้อผิดพลาด (403)

4. Application.properties

```

1 spring.application.name=MyProjectCat
2 spring.datasource.url=jdbc:mysql://localhost:3306/catcommudb
3 spring.datasource.username=user01
4 spring.datasource.password=s$create01
5 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
6 spring.jpa.show-sql: true
7 spring.jpa.properties.hibernate.formate_sql=true
8 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
9 spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
10 spring.main.allow-bean-definition-overriding=true
11 spring.jpa.hibernate.ddl-auto=update
12

```

3.3.6 สร้าง html file ในโฟลเดอร์ templates

1. addCat.html

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <title>Create a New Post</title>
5 <link
6   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
7   rel="stylesheet"
8   integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hk+ALEwIH"
9   crossorigin="anonymous">
10<style>
11 body {
12   background-color: #f4f4f4;
13   color: #333;
14   font-family: Arial, sans-serif;
15 }
16
17 .container {
18   max-width: 600px;
19   margin: 50px auto;
20   padding: 20px;
21   background-color: #ffffff;
22   border-radius: 8px;
23   box-shadow: 0 2px 10px #rgba(0, 0, 0, 0.1);
24 }
25
26 h2 {
27   text-align: center;
28   color: #0d3889;
29   margin-bottom: 30px;
30 }
31
32 .form-label {
33   font-weight: bold;
34 }

```

```

36 .btn-primary {
37   background-color: #0d3889;
38   border: none;
39 }
40
41 .btn-primary:hover {
42   background-color: #ffae34;
43   color: white;
44 }
45 </style>
46 </head>

47<body>
48  <div class="container">
49    <h2>Create a New Cat</h2>
50    <form th:action="@{/addCat}" th:object="${cat}" method="post">
51      <div class="mb-3">
52        <label for="name" class="form-label">Name:</label> <input
53          type="text" id="name" th:field="${name}" class="form-control"
54          required />
55      </div>
56      <div class="mb-3">
57        <label for="breed" class="form-label">Breed:</label> <input
58          type="text" id="breed" th:field="${breed}" class="form-control"
59          required />
60      </div>
61      <div class="mb-3">
62        <label for="age" class="form-label">Age(Year):</label> <input
63          type="number" id="age" th:field="${age}" class="form-control"
64          required min="0" step="0.01" />
65      </div>
66
67      <div class="mb-3">
68        <label for="catPic" class="form-label">Cat Picture URL:</label> <input
69          type="text" id="catPic" th:field="${catPic}" class="form-control" />
70      </div>
71      <div class="text-center">
72        <button type="submit" class="btn btn-primary">Save</button>
73      </div>
74    </form>
75  </div>
76
77  <script
78    src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
79    integrity="sha384-Yvpcryf0tY3LHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdslK1eN7N6jIeHz"
80    crossorigin="anonymous"></script>
81 </body>
82 </html>

```

2. addComment.html

```
1 <!DOCTYPE html>
2 @<html xmlns:th="http://www.thymeleaf.org">
3 @<head>
4     <title>Add Comment</title>
5     <link rel="stylesheet" href="/css/style.css"<!-- Link to your CSS --&gt;
6 &lt;/head&gt;
7 @&lt;body&gt;
8     &lt;h1&gt;Add Comment&lt;/h1&gt;
9
10    &lt;h2 th:text="${post.title}"&gt;&lt;/h2&gt;<!-- Display the post title --&gt;
11
12 @&lt;form th:action="@{/addComment/{id}(id=${post.id})}" method="post"&gt;
13     &lt;div&gt;
14         &lt;label for="content"&gt;Comment:&lt;/label&gt;
15         &lt;textarea id="content" name="content" rows="4" cols="50" required&gt;&lt;/textarea&gt;
16     &lt;/div&gt;
17     &lt;div&gt;
18         &lt;button type="submit"&gt;Add Comment&lt;/button&gt;
19     &lt;/div&gt;
20 &lt;/form&gt;
21
22    &lt;a th:href="@{/viewPost/{id}(id=${post.id})}"&gt;Back to Post&lt;/a&gt;<!-- Link back to the post --&gt;
23 &lt;/body&gt;
24 &lt;/html&gt;
25 |</pre>
```

3. addPost.html

```
1 <!DOCTYPE html>
2 @<html xmlns:th="http://www.thymeleaf.org">
3 @<head>
4     <title>Create a New Post</title>
5     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
6          integrity="sha384-QWTKZyjpPEjISv5WwR0FeRpok6YctnYmDr5pNLyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
7 @<style>
8     body {
9         background-color: #f4f4f4;
10        color: #333;
11        font-family: Arial, sans-serif;
12    }
13
14    .container {
15        max-width: 600px;
16        margin: 50px auto;
17        padding: 20px;
18        background-color: #ffffff;
19        border-radius: 8px;
20        box-shadow: 0 2px 10px #rgba(0, 0, 0, 0.1);
21    }
22
23    h2 {
24        text-align: center;
25        color: #0d3889;
26        margin-bottom: 30px;
27    }
28
29    .form-label {
30        font-weight: bold;
31    }
32
33    .btn-primary {
34        background-color: #0d3889;
35        border: none;
36    }
```

```

8     .btn-primary:hover {
9         background-color: #ffae34;
0         color: white;
1     }
2   
```

- 3 </head>
- 4 <body>
- 5 <div class="container">
- 6 <h2>Create a New Post</h2>
- 7 <form th:action="@{/addPost}" th:object="\${post}" method="post">
- 8 <div class="mb-3">
9 <label for="title" class="form-label">Title:</label>
0 <input type="text" id="title" th:field="*[title]" class="form-control" required/>
1 </div>
2 <div class="mb-3">
3 <label for="content" class="form-label">Content:</label>
4 <textarea id="content" th:field="*[content]" class="form-control" rows="5" required></textarea>
5 </div>
6 <div class="mb-3">
7 <label for="postPic" class="form-label">Post Picture URL:</label>
8 <input type="text" id="postPic" th:field="*[postPic]" class="form-control"/>
9 </div>
0 <div class="text-center">
1 <button type="submit" class="btn btn-primary">Save</button>
2 </div>
3 </form>
4 </div>
5
6 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
7 integrity="sha384-YvpcrYf0tY3LHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdsLK1eN7N6jTeHz" crossorigin="anonymous"></script>
8 </body>
9 </html>

4. editCat.html

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3   
```

- 4 <head>
- 5 <meta charset="UTF-8">
- 6 <title>Edit Post</title>
- 7 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WarU9i" data-bbox="111 500 846 515" style="display: block; margin: auto; width: fit-content; height: fit-content; border: 1px solid black; padding: 5px; border-radius: 5px; font-size: 10px; font-weight: bold; text-decoration: none; color: inherit; text-align: center; background-color: transparent; transition: all 0.3s ease;"/>
- 8 <style>
- 9 body {
10 background-color: #f8f9fa;
11 }
12 .container {
13 background-color: #ffffff;
14 padding: 20px;
15 border-radius: 8px;
16 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
17 }
18 h2 {
19 margin-bottom: 20px;
20 color: #333;
21 }
22 label {
23 font-weight: bold;
24 }
25 .form-group {
26 margin-bottom: 15px;
27 }
28 .btn-primary {
29 background-color: #007bff;
30 border-color: #007bff;
31 }
32 .btn-primary:hover {
33 background-color: #0056b3;
34 border-color: #004085;
35 }
36 .form-control {
37 border-radius: 0.25rem;
38 }
39 </style>

```

39 | </head>
40 |
41<body>
42    <div class="container mt-4">
43        <h2>Edit Cat</h2>
44        <form th:action="@{/updateCat/{id}(id=${cat.id})}" th:object="${cat}" method="post">
45            <div class="form-group">
46                <label for="name">Name:</label>
47                <input type="text" id="name" th:field="*{name}" class="form-control" placeholder="Enter cat name" />
48            </div>
49            <div class="form-group">
50                <label for="breed">Breed:</label>
51                <input id="breed" th:field="*{breed}" class="form-control" placeholder="Enter cat breed" />
52            </div>
53            <div class="form-group">
54                <label for="age">Age(year):</label>
55                <input id="age" th:field="*{age}" class="form-control" placeholder="Enter cat age" min="0" step="0.01" />
56            </div>
57            <div class="form-group">
58                <label for="catPic">Post Picture URL:</label>
59                <input type="text" id="catPic" th:field="*{catPic}" class="form-control" placeholder="Enter URL for cat picture" />
60            </div>
61            <button type="submit" class="btn btn-primary">Save</button>
62        </form>
63    </div>
64    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YVpcryYf0tY3LHB60NNkmXc5s9fDVZLESaAA55N"
65 </body>
66
67 </html>

```

5. editPost.html

```

1<!DOCTYPE html>
2<html xmlns:th="http://www.thymeleaf.org">
3
4<head>
5    <meta charset="UTF-8">
6    <title>Edit Post</title>
7    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRL
8<style>
9     body {
10         background-color: #f8f9fa;
11     }
12     .container {
13         background-color: #ffffff;
14         padding: 20px;
15         border-radius: 8px;
16         box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
17     }
18     h2 {
19         margin-bottom: 20px;
20         color: #333;
21     }
22     label {
23         font-weight: bold;
24     }
25     .form-group {
26         margin-bottom: 15px;
27     }
28     .btn-primary {
29         background-color: #007bff;
30         border-color: #007bff;
31     }
32     .btn-primary:hover {
33         background-color: #0056b3;
34         border-color: #004085;
35     }
36     .form-control {
37         border-radius: 0.25rem;
38     }

```

```

39     </style>
40 </head>
41
42<body>
43    <div class="container mt-4">
44      <h2>Edit Post</h2>
45      <form th:action="@{/updatePost/{id}(id=${post.id})}" th:object="${post}" method="post">
46        <div class="form-group">
47          <label for="title">Title:</label>
48          <input type="text" id="title" th:field="#{title}" class="form-control" placeholder="Enter post title" />
49        </div>
50        <div class="form-group">
51          <label for="content">Content:</label>
52          <textarea id="content" th:field="#{content}" class="form-control" rows="4" placeholder="Enter post content"></textarea>
53        </div>
54        <div class="form-group">
55          <label for="postPic">Post Picture URL:</label>
56          <input type="text" id="postPic" th:field="#{postPic}" class="form-control" placeholder="Enter URL for post picture" />
57        </div>
58        <button type="submit" class="btn btn-primary">Save</button>
59      </form>
60    </div>
61    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3LHB60NNkmXc5s9fdVZLESa" crossorigin="anonymous"></script>
62 </body>
63
64 </html>

```

6. editUser.html

```

1 <!DOCTYPE html>
2<html xmlns:th="http://www.thymeleaf.org">
3
4<head>
5 <meta charset="UTF-8">
6 <title>Edit Post</title>
7 <link
8   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
9   rel="stylesheet"
10  integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hlW+ALEwIH"
11  crossorigin="anonymous">
12<style>
13 body {
14   background-color: #f8f9fa;
15 }
16
17 .container {
18   background-color: #ffffff;
19   padding: 20px;
20   border-radius: 8px;
21   box-shadow: 0 4px 8px #rgba(0, 0, 0, 0.1);
22 }
23
24 h2 {
25   margin-bottom: 20px;
26   color: #333;
27 }
28
29 label {
30   font-weight: bold;
31 }
32
33 .form-group {
34   margin-bottom: 15px;
35 }

```

```

37 .btn-primary {
38     background-color: #007bff;
39     border-color: #007bff;
40 }
41
42 .btn-primary:hover {
43     background-color: #0056b3;
44     border-color: #004085;
45 }
46
47 .form-control {
48     border-radius: 0.25rem;
49 }
50 </style>
51 </head>
52
53<body>
54    <div class="container mt-4">
55        <h2>Edit Profile</h2>
56        <form th:action="@{/updateUser/{id}(id=${user.id})}"
57             th:object="${user}" method="post">
58            <div class="form-group">
59                <label for="username">Username:</label> <input type="text"
60                  id="username" th:field="#${username}" class="form-control"
61                  placeholder="Enter Username" />
62            </div>
63            <div class="form-group">
64                <label for="email">Email:</label> <input type="text" id="email"
65                  th:field="#${email}" class="form-control" placeholder="Enter Email" />
66            </div>
67            <div class="form-group">
68                <label for="currentPassword">Current Password:</label> <input
69                  type="password" id="currentPassword" name="currentPassword"
70                  class="form-control" placeholder="Enter Current Password" required />
71        </div>
72
73        <div class="form-group">
74            <label for="password">New Password:</label> <input type="password"
75              id="password" th:field="#${password}" class="form-control"
76              placeholder="Enter Password" />
77        </div>
78        <div class="alert alert-danger" th:if="${passwordError}" th:text="${passwordError}"></div>
79        <div class="form-group">
80            <label for="userPic">Profile Picture URL:</label> <input type="text"
81              id="userPic" th:field="#${userPic}" class="form-control"
82              placeholder="Enter URL for user picture" />
83        </div>
84        <button type="submit" class="btn btn-primary">Save</button>
85    </form>
86  </div>
87  <script
88    src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
89    integrity="sha384-YvpcryYf0tY3LHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdslK1eN7N6jIeHz"
90    crossorigin="anonymous"></script>
91 </body>
92
93 </html>

```

7. home.html

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <meta charset="UTF-8">
6   <title>Home Page</title>
7   <link
8     href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
9     rel="stylesheet"
10    integrity="sha384-QWTKZyjpPEjISv5WaRU9OfeRpok6YctnYmDr5pNlyT2bRjXh0JNhjY6hi4+ALEwIH"
11    crossorigin="anonymous">
12 </head>
13 <style>
14   h1 {
15     background-color: #ffae34;
16   }
17   body {
18     background-color: #0d3889;
19   }
20
21 .post-image {
22   width: 100%;
23   max-height: 250px; /* จำกัดความสูงสุด และไม่จำกัดความกว้างมาก */
24   object-fit: cover;
25 }
26
27 .card {
28   width: 100%;
29   max-width: 450px;
30   flex-grow: 1; /* ให้ card ขยายตามเนื้อหา */
31 }
32
33 .card-body {
34   display: flex;
35   flex-direction: column;
36   justify-content: space-between; /* กระจายเนื้อหาให้สมดุล */
37 }
38
39 .user-image {
40   width: 50px;
41   height: 50px;
42   border-radius: 50%; /* Makes the image circular */
43   object-fit: cover; /* Ensures the image covers the circular frame */
44   border: 2px solid #ccc; /* Adds a border around the circle */
45 }
46 </style>
47 </head>
48
49 <body>
50   <h1 class="text-center text-light ps-5">CATZY</h1>
51   <nav class="navbar navbar-expand-lg">
52     <div class="container justify-content-center">
53       <ul class="navbar-nav mr-auto">
54         <li class="nav-item"><a class="nav-link text-light" href="/">Home</a></li>
55         <li class="nav-item">
56           <li class="nav-item"><a class="nav-link text-light"
57             href="/myProfile">My Profile</a></li>
58           <li class="nav-item">
59             <li class="nav-item" ><a class="nav-link text-light"
60               href="/logout">Logout</a></li>
61           </ul>
62
63         </div>
64       </div>
65     </nav>
66     <div class="container text-light">
67       <div class="my-4">
68         <h2 class="my-4 text-center">All Posts</h2>
69         <p class="fw-bold">
70           Post Your Story --><a class="btn btn-warning" href="/addPost">Add
71           New Post</a>
72         </p>
73       </div>
```

```

73@     <div class="row">
74@         <div class="d-flex flex-wrap justify-content-start">
75@             <div th:each="post : ${posts}">
76@                 class="col-lg-3 col-md-4 col-sm-6 m-2">
77@                     <div class="card ">
78@                         <div class="card-footer d-flex align-items-center">
79@                              <a
82@                                 th:href="@{'/user/' + ${post.user.id}}"
83@                                 class="text-primary-emphasis"> <span
84@                                 th:text="${post.user.username}"></span>
85@                         </a>
86@                     </div>
87@                     
89@                     <div class="card-body">
90@                         <h5 th:text="${post.title}" class="card-title"></h5>
91@                         <p th:text="${post.content}" class="card-text"></p>
92@                         <a class="btn btn-warning"
93@                             th:href="@{/viewPost/{id}(id=${post.id})}">View Post</a>
94@                     </div>
95@                 </div>
96@             </div>
97@         </div>
98@     </div>
99@ </div>
l00@ 
l01@     <script
l02@         src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
l03@         integrity="sha384-YvpcrYf0tY3LHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdsLK1eN7N6jIeHz"
l04@         crossorigin="anonymous"></script>
l05 </body>
l06
l07 </html>

```

8. login.html

```

1  <!DOCTYPE html>
2  <html>
3
4@ <head>
5  <meta charset="UTF-8">
6  <title>Login</title>
7@ <style>
8  body {
9      font-family: 'Arial', sans-serif;
10     margin: 0;
11     padding: 0;
12     display: flex;
13     justify-content: center;
14     align-items: center;
15     height: 100vh;
16     background-color: #fffae34;
17 }
18
19 .form-container {
20     background-color: #fff;
21     width: 400px;
22     padding: 30px;
23     border-radius: 10px;
24     box-shadow: 0 4px 20px rgba(0, 0, 0, 0.2);
25 }
26
27
28 h2 {
29     color: #0d3889;
30     text-align: center;
31     margin-bottom: 20px;
32     font-size: 1.8em;
33 }

```

```
35 label {
36   display: block;
37   margin-bottom: 8px;
38   color: #0d3889;
39   font-weight: bold;
40 }
41
42 input {
43   width: 100%;
44   padding: 10px;
45   margin-bottom: 16px;
46   border: 1px solid #ccc;
47   border-radius: 4px;
48   box-sizing: border-box;
49   transition: border 0.3s;
50 }
51
52 input:focus {
53   border-color: #0d3889;
54   outline: none;
55 }
56
57 button {
58   background-color: #0d3889;
59   color: #fff;
60   padding: 10px 20px;
61   border: none;
62   border-radius: 4px;
63   cursor: pointer;
64   font-weight: bold;
65   transition: background-color 0.3s;
66   width: 100%;
67 }
68
69 button:hover {
70   background-color: #ffae34;
71 }
72
73 .form-footer {
74   text-align: center;
75   margin-top: 20px;
76   color: #888;
77 }
78
79 .form-footer a {
80   color: #0d3889;
81   text-decoration: underline;
82 }
83
84 .error-message, .logout-message {
85   background-color: #0d3889;
86   color: #fff;
87   padding: 10px;
88   border-radius: 3px;
89   text-align: center;
90   margin-bottom: 16px;
91   transition: opacity 0.3s;
92 }
93
94 .error-message.hidden, .logout-message.hidden {
95   opacity: 0;
96   height: 0;
97   overflow: hidden;
98 }
99 </style>
100 </head>
```

```

12 <body>
13   <div class="container">
14     <div class="form-container">
15
16       <div th:if="${param.error}" class="error-message">
17         <p>Invalid Username or Password</p>
18       </div>
19
20       <div th:if="${param.logout}" class="Logout-message">
21         <p>Logout Successful!</p>
22       </div>
23
24       <h2>Login</h2>
25       <form th:action="@{/login}" method="post" role="form"
26         th:object="${user}">
27
28         <label for="username">Username:</label> <input
29           th:field="*{username}" type="text" id="username" name="username"
30           placeholder="Enter your Username" required> <label
31           for="password">Password:</label> <input th:field="*{password}"
32           type="password" id="password" name="password"
33           placeholder="Enter your Password" required>
34
35         <button type="submit">Login</button>
36       </form>
37       <p class="form-footer">
38         Don't have an account? <a th:href="@{/register}">Register</a> here.
39       </p>
40
41     </div>

```

9. myProfile.html

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3
4   <head>
5     <meta charset="UTF-8">
6     <title>My Profile</title>
7     <link
8       href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
9       rel="stylesheet"
10      integrity="sha384-QwTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hlw+ALEwIH"
11      crossorigin="anonymous">
12
13   <style>
14     body {
15       background-color: #0d3889;
16       color: white;
17     }
18
19     .profile-container {
20       margin: 0 auto;
21       padding-top: 0px;
22     }
23
24     h1 {
25       font-size: 2rem;
26       color: white;
27       background-color: #ffae34;
28       text-align: center;
29       padding: 10px;
30     }
31
32     .profile-picture img {
33       border-radius: 50%;
34       border: 3px solid #ccc;
35     }

```

```
6 .user-info {
7   font-size: 1.2rem;
8   color: #ddd;
9   text-align: center;
0 }
1
2 h2 {
3   font-size: 1.5rem;
4   margin-top: 30px;
5   color: white;
6   text-align: center;
7 }
8
9 .post-image {
0   width: 100%;
1   max-height: 250px; /* จำกัดความสูงสูงสุด แต่ไม่บังคับความสูงเท่ากัน */
2   object-fit: cover;
3 }
4
5 .card {
6   width: 100%;
7   max-width: 450px;
8   flex-grow: 1; /* ให้ card ขยายตามเนื้อหา */
9 }
0
1 .card-body {
2   display: flex;
3   flex-direction: column;
4   justify-content: space-between; /* กระจายเนื้อหาให้สมดุล */
5 }
6
7 .card-title {
8   font-size: 1.2rem;
9   color: #333;
0   margin-bottom: 10px;
1 }
^
73 .card-text {
74   color: #666;
75   margin-bottom: 15px;
76 }
77
78 .user-image {
79   width: 50px;
80   height: 50px;
81   border-radius: 50%;
82   margin-right: 10px;
83   object-fit: cover;
84 }
85
86 .btn {
87   background-color: #ffae34;
88   font-weight: bold;
89   color: white;
90   border-radius: 0.25rem;
91   padding: 0.375rem 0.75rem;
92   font-size: 0.875rem;
93   text-align: center;
94 }
95
96 .btn:hover {
97   background-color: #ff8c00; /* เป็นสีเมืองแม่น้ำ */
98   color: white;
99   transition: background-color 0.3s ease, color 0.3s ease;
00 }
01
02 .cats-list {
03   list-style: none; /* Remove bullet points */
04   padding: 0; /* Remove default padding */
05   display: flex; /* Use flexbox for horizontal alignment */
06   flex-wrap: wrap; /* Allow wrapping to the next line if necessary */
07 }
^
```

```

~` 09 .cat-item {
10     display: flex; /* Use flex for item layout */
11     flex-direction: column; /* Arrange children in a column */
12     align-items: center; /* Center align the items */
13     margin-right: 20px; /* Space between items */
14     text-align: center; /* Center the text */
15 }
16
17 .cat-image {
18     width: 100px; /* Set the desired width */
19     height: auto; /* Maintain aspect ratio */
20     border-radius: 5px; /* Optional: round the corners */
21     margin-bottom: 5px; /* Space between image and name */
22 }
23
24 .cat-name {
25     font-size: 18px; /* Adjust font size as needed */
26     font-weight: bold; /* Make the name bold */
27 }
28
29 nav {
30     margin-left: 80px;
31     padding-top: 20px;
32 }
33
34 .row {
35     margin-left: 80px;
36     margin-right: 80px
37 }
38
39 .cat {
40     margin-left: 100px;
41     margin-right: 100px
42 }
43 </style>
44 </head>

@<body>
@  <div class="profile-container">
@    <h1>
@      Profile of <span th:text="${user.username}"></span>
@    </h1>
@    <nav aria-label="breadcrumb">
@      <ol class="breadcrumb">
@        <li class="breadcrumb-item"><a href="/">Home</a></li>
@        <li class="breadcrumb-item active text-light" aria-current="page">My
@          Profile</li>
@        </ol>
@      </nav>
@      
@      <div class="profile-picture text-center mb-4">
@        <img th:src ="${user.userPic}" alt="User Picture"
@             style="width: 150px; height: 150px; object-fit: cover;">
@      </div>
@      
@      <p class="user-info">
@        Username: <span th:text="${user.username}"></span>
@      </p>
@      <p class="user-info">
@        Email: <span th:text="${user.email}"></span>
@      </p>
@      <div class="my-4 text-center">
@        <p class="fw-bold">
@          <a class="btn btn-warning"
@              th:href ="@{/editUser/{id}(id=${user.id})}">Edit Profile</a>
@        </p>
@      </div>
@      <div class="my-4 text-center">
@        <p class="fw-bold">
@          <a class="btn btn-warning" th:href ="@{/addCat}">Add Cat</a>
@        </p>
@      </div>

```

```

<!-- Cats Section -->
<h2>Cats</h2>
<ul class="cats-list cat">
    <li th:each="cat : ${user.cats}" class="cat-item">
        
        <p th:text="${cat.name}" class="cat-name"></p>
        <a th:href="@{/catDetail/{id}(id=${cat.id})}" class="btn btn-link text-light">View Detail</a>
    </li>
</ul>

<!-- Posts Section -->
<h2>Posts</h2>
<div class="row">
    <div class="d-flex flex-wrap justify-content-start">
        <div th:each="post : ${user.posts}" class="col-lg-3 col-md-4 col-sm-6 m-2">
            <div class="card">
                <div class="card-footer d-flex align-items-center">
                     <a
                        th:href="@{'/user/' + ${post.user.id}}"
                        class="text-primary-emphasis"> <span
                        th:text="${post.username}"></span>
                </a>
            </div>
            
            <div class="card-body">
                <h5 th:text="${post.title}" class="card-title"></h5>
                <p th:text="${post.content}" class="card-text"></p>
                <div class="btn-group" role="group">
                    <a class="btn btn-success"
                        th:href="@{/viewPost/{id}(id=${post.id})}">View</a> <a
                        class="btn btn-warning"
                        th:href="@{/editPost/{id}(id=${post.id})}">Edit</a> <a
                        class="btn btn-danger"
                        th:href="@{/deletePost/{id}(id=${post.id})}"
                        onclick="return confirm('Are you sure you want to delete this post?');">Delete</a>
                </div>
            </div>
            </div>
        </div>
    </div>
</div>
<script
    src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-YvpcrYf0tY3LHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdsLK1eN7N6jIeHz"
    crossorigin="anonymous"></script>
</body>
</html>

```

10. register.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5 <meta charset="UTF-8">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>Registration</title>
8 <style>
9 body {
10   font-family: 'Arial', sans-serif;
11   margin: 0;
12   padding: 0;
13   display: flex;
14   justify-content: center;
15   align-items: center;
16   height: 100vh;
17   background-color: #ffae34;
18 }
19
20 .form-container {
21   background-color: #fff;
22   width: 400px;
23   padding: 30px;
24   border-radius: 12px;
25   box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
26 }
27
28 h2 {
29   color: #0d3889;
30   text-align: center;
31   margin-bottom: 24px;
32   font-size: 1.8em;
33 }
34
35 label {
36   display: block;
37   margin-bottom: 10px;
38   color: #0d3889;
39   font-weight: bold;
40 }
41
42 input {
43   width: 100%;
44   padding: 10px;
45   margin-bottom: 20px;
46   border: 1px solid #ccc;
47   border-radius: 6px;
48   box-sizing: border-box;
49   transition: border 0.3s ease;
50 }
51
52 input:focus {
53   border-color: #5e057e;
54   outline: none;
55 }
56
57 button {
58   background-color: #0d3889;
59   color: #fff;
60   padding: 12px;
61   border: none;
62   border-radius: 6px;
63   cursor: pointer;
64   font-weight: bold;
65   width: 100%;
66   font-size: 1em;
67   transition: background-color 0.3s ease;
68 }
69
70 button:hover {
71   background-color: #ffae34;
72 }
```

```

'4 .form-footer {
'5     text-align: center;
'6     margin-top: 20px;
'7     color: #888;
'8 }
'9
'10 .form-footer a {
'11     color: #0d3889;
'12     text-decoration: underline;
'13 }
'14
'15 .success-message {
'16     background-color: #5e057e;
'17     color: #ffff;
'18     padding: 12px;
'19     border-radius: 6px;
'20     text-align: center;
'21     margin-bottom: 20px;
'22 }
'23
'24 .error-message {
'25     color: red;
'26     text-align: center;
'27     margin-bottom: 16px;
'28 }
'29 </style>
'30 </head>
'31
'32 <body>
'33     <div class="form-container">
'34
'35         <div th:if="${param.success}" class="success-message">
'36             <p>Registration Successful!</p>
'37         </div>
'38
'39         <h2>Register</h2>
'40         <form th:action="@{/register}" method="post" role="form"
'41             th:object="${user}">
'42             <label for="username">Username:</label> <input th:field="*{username}"
'43                 type="text" id="username" name="username"
'44                 placeholder="Enter your Username" required> <label
'45                 for="email">Email:</label> <input th:field="*{email}" type="email"
'46                 id="email" name="email" placeholder="Enter your Email" required>
'47
'48             <label for="password">Password:</label> <input th:field="*{password}"
'49                 type="password" id="password" name="password"
'50                 placeholder="Enter your Password" required>
'51
'52             <button type="submit">Register</button>
'53         </form>
'54
'55         <div th:if="${Userexist}" class="error-message">
'56             <span>Username is Taken</span>
'57         </div>
'58
'59         <div class="form-footer">
'60             <p>
'61                 Already have an account? <a th:href="@{/Login1}">Login</a> here.
'62             </p>
'63         </div>
'64     </div>
'65 </body>
'66
'67 </html>

```

11. viewCatdetail.html

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3
4 <head>
5 <meta charset="UTF-8">
6 <title>View Post</title>
7 <link
8   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
9   rel="stylesheet"
10  integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
11  crossorigin="anonymous">
12 <style>
13 body {
14   background-color: #d3889;
15   color: white;
16 }
17
18 .card {
19   margin-bottom: 20px;
20   border-radius: 8px;
21   box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
22   background-color: #fff;
23   width: 100%; /* Full width for responsive design */
24   max-width: 800px; /* Maximum width for larger screens */
25   margin: 0 auto; /* Centering the card */
26 }
27
28 .card-img-top {
29   height: 400px; /* Increased height */
30   object-fit: cover;
31   border-top-left-radius: 8px;
32   border-top-right-radius: 8px;
33 }
34
35 .card-body {
36   padding: 20px; /* Added more padding for a spacious look */
37 }

38 .card-title {
39   font-size: 1.5rem;
40 }
41
42 .btn {
43   background-color: #ffae34;
44   color: white;
45   border-radius: 0.25rem;
46   padding: 0.375rem 0.75rem;
47 }
48
49 .btn:hover {
50   background-color: #ff8c00;
51   color: white;
52   transition: background-color 0.3s ease;
53 }
54
55 .comment-section {
56   margin-top: 20px;
57   max-width: 800px;
58   margin: 0 auto;
59 }
60
61 .comment {
62   max-width: 800px;
63   padding: 10px;
64   background-color: #ffae34;
65   border-radius: 8px;
66   margin: 0 auto;
67   margin-bottom: 15px;
68 }
69 }
```

```

71 .reply {
72   max-width: 800px;
73   margin-left: 30px;
74   padding: 5px;
75   background-color: #e0e0e0;
76   border-radius: 8px;
77   margin: 0 auto;
78 }
79
80 .profile-picture img {
81   border-radius: 50%;
82   border: 3px solid #ccc;
83   max-width: 150px;
84   max-height: 150px;
85 }
86
87 .user-info {
88   font-size: 1.2rem;
89   color: #ddd;
90   text-align: center;
91 }
92
93 h1 {
94   font-size: 2rem;
95   color: white;
96   background-color: #ffae34;
97   text-align: center;
98   padding: 10px;
99   max-width: 800px;
100  margin: 0 auto;
101 }
102
103 nav {
104   padding: 10px;
105   max-width: 800px;
106   margin: 0 auto;
107 }

108 </style>
109 </head>
110
111 <body>
112   <div class="container">
113     <h1 class="my-4">Cat Details</h1>
114     <nav aria-label="breadcrumb" class="p-2">
115       <ol class="breadcrumb">
116         <li class="breadcrumb-item"><a href="/">Home</a></li>
117         <li class="breadcrumb-item active text-light" aria-current="page">View
118           Cats</li>
119       </ol>
120     </nav>
121     <div class="card">
122       
124       <div class="card-body">
125         <h5 class="card-title">Name: <span th:text="${cat.name}"></span></h5>
126         <p class="card-text">Breed: <span th:text="${cat.breed}"></span></p>
127         <p class="card-text">Age: <span th:text="${cat.age}"></span> year</p>
128       </div>
129     </div>
130   </div>
131
132   <script
133     src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
134     integrity="sha384-YvpcrYf0tY3LHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdsLK1eN7N6jIeHz"
135     crossorigin="anonymous"></script>
136 </body>
137
138 </html>

```

12. viewCateDetailbyUser.html

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3
4 <head>
5 <meta charset="UTF-8">
6 <title>View Post</title>
7 <link
8   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
9   rel="stylesheet"
10  integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNLyT2bRjXh0JMhjY6hW+ALEwIH"
11  crossorigin="anonymous">
12 <style>
13 body {
14   background-color: #0d3889;
15   color: white;
16 }
17
18 .card {
19   margin-bottom: 20px;
20   border-radius: 8px;
21   box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
22   background-color: #fff;
23   width: 100%; /* Full width for responsive design */
24   max-width: 800px; /* Maximum width for larger screens */
25   margin: 0 auto; /* Centering the card */
26 }
27
28 .card-img-top {
29   height: 400px; /* Increased height */
30   object-fit: cover;
31   border-top-left-radius: 8px;
32   border-top-right-radius: 8px;
33 }
34
35 .card-body {
36   padding: 20px; /* Added more padding for a spacious look */
37 }
38
39 .card-title {
40   font-size: 1.5rem;
41 }
42 .btn {
43   background-color: #fffae34;
44   color: white;
45   border-radius: 0.25rem;
46   padding: 0.375rem 0.75rem;
47 }
48
49 .btn:hover {
50   background-color: #ff8c00;
51   color: white;
52   transition: background-color 0.3s ease;
53 }
54 .comment-section {
55   margin-top: 20px;
56   max-width: 800px;
57   margin: 0 auto;
58 }
59
60 .comment {
61   max-width: 800px;
62   padding: 10px;
63   background-color: #fffae34;
64   border-radius: 8px;
65   margin: 0 auto;
66   margin-bottom: 15px;
67 }
68
69 .reply {
70   max-width: 800px;
71   margin-left: 30px;
72   padding: 5px;
73   background-color: #e0e0e0;
74   border-radius: 8px;
75   margin: 0 auto;
76 }
```

```

78 .profile-picture img {
79   border-radius: 50%;
80   border: 3px solid #ccc;
81   max-width: 150px;
82   max-height: 150px;
83 }
84
85 .user-info {
86   font-size: 1.2rem;
87   color: #ddd;
88   text-align: center;
89 }
90
91 h1 {
92   font-size: 2rem;
93   color: white;
94   background-color: #ffae34;
95   text-align: center;
96   padding: 10px;
97   max-width: 800px;
98   margin: 0 auto;
99 }
100
101 nav {
102   padding: 10px;
103   max-width: 800px;
104   margin: 0 auto;
105 }
106 </style>
107 </head>
108
109 <body>
110   <div class="container">
111     <h1 class="my-4">Cat Details</h1>
112     <nav aria-label="breadcrumb" class="p-2">
113       <ol class="breadcrumb">
114         <li class="breadcrumb-item"><a href="/">Home</a></li>
115         <li class="breadcrumb-item active text-light" aria-current="page">View
116           Cats</li>
117         </ol>
118       </nav>
119       <div class="card">
120         
122         <div class="card-body">
123           <h5 class="card-title">Name: <span th:text="${cat.name}"></span></h5>
124           <p class="card-text">Breed: <span th:text="${cat.breed}"></span></p>
125           <p class="card-text">Age: <span th:text="${cat.age}"></span> year</p>
126         </div>
127         <div class="m-2" >
128           <a th:href="@{/editCat/{id}(id=${cat.id})}" class="btn btn-link text-primary">Edit</a>
129           <a th:href="@{/deleteCat/{id}(id=${cat.id})}" class="btn btn-link text-danger">Delete</a>
130         </div>
131       </div>
132     </div>
133
134     <script
135       src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
136       integrity="sha384-YvpcrYf0Y3lHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdslK1eN7N6jIeHz"
137       crossorigin="anonymous"></script>
138   </body>
139
140 </html>

```

13. viewPost.html

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3
4 <head>
5 <meta charset="UTF-8">
6 <title>View Post</title>
7 <link
8 href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
9 rel="stylesheet"
10 integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
11 crossorigin="anonymous">
12 <style>
13 body {
14   background-color: #0d3889;
15   color: white;
16 }
17
18 .card {
19   margin-bottom: 20px;
20   border-radius: 8px;
21   box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
22   background-color: #fff;
23   width: 100%; /* Full width for responsive design */
24   max-width: 800px; /* Maximum width for larger screens */
25   margin: 0 auto; /* Centering the card */
26 }
27
28 .card-img-top {
29   height: 400px; /* Increased height */
30   object-fit: cover;
31   border-top-left-radius: 8px;
32   border-top-right-radius: 8px;
33 }
34
35 .card-body {
36   padding: 20px; /* Added more padding for a spacious look */
37 }
38
39 .card-title {
40   font-size: 1.5rem;
41 }
42 .btn {
43   background-color: #ffae34;
44   color: white;
45   border-radius: 0.25rem;
46   padding: 0.375rem 0.75rem;
47 }
48
49 .btn:hover {
50   background-color: #ff8c00;
51   color: white;
52   transition: background-color 0.3s ease;
53 }
54 .comment-section {
55   margin-top: 20px;
56   max-width: 800px;
57   margin: 0 auto;
58 }
59
60 .comment {
61   max-width: 800px;
62   padding: 10px;
63   background-color: #ffae34;
64   border-radius: 8px;
65   margin: 0 auto;
66   margin-bottom: 15px;
67 }
68
69 .reply {
70   max-width: 800px;
71   margin-left: 30px;
72   padding: 5px;
73   background-color: #e0e0e0;
74   border-radius: 8px;
75   margin: 0 auto;
76 }
```

```

78 .profile-picture img {
79   border-radius: 50%;
80   border: 3px solid #ccc;
81   max-width: 150px;
82   max-height: 150px;
83 }
84
85 .user-info {
86   font-size: 1.2rem;
87   color: #ddd;
88   text-align: center;
89 }
90
91 h1 {
92   font-size: 2rem;
93   color: white;
94   background-color: #ffae34;
95   text-align: center;
96   padding: 10px;
97   max-width: 800px;
98   margin: 0 auto;
99 }
100
101 nav {
102   padding: 10px;
103   max-width: 800px;
104   margin: 0 auto;
105 }
106 </style>
107 </head>
108
109 ><body>
110   <div class="container">
111     <h1 class="my-4">Post Details</h1>
112     <nav aria-label="breadcrumb" class="p-2">
113       <ol class="breadcrumb">
114         <li class="breadcrumb-item"><a href="/">Home</a></li>
115         <li class="breadcrumb-item active text-light" aria-current="page">View
116           Post</li>
117         </ol>
118       </nav>
119     <!-- Post -->
120     <div class="card">
121       
123       <div class="card-body">
124         <h5 class="card-title" th:text="${post.title}"></h5>
125         <p class="card-text" th:text="${post.content}"></p>
126       </div>
127       <div class="card-footer d-flex align-items-center">
128          <a
132             th:href="@{'/user/' + ${user.id}}"
133             class="link-offset-2 link-underline link-underline-opacity-0 mb-0 text-primary-emphasis">
134             <span th:text="${user.username}"></span>
135           </a>
136         </div>
137       </div>
138
139     <!-- Comments Section -->
140     <div class="comment-section">
141       <h2>Comments</h2>
142       <form th:action="@{/addComment/{id}(id=${post.id})}"
143           th:object="${post}" method="post">
144         <div class="input-group mb-3">
145           <input type="text" id="content" th:field="*{content}"
146               class="form-control" placeholder="Enter Comment" />
147           <button class="btn btn-outline-secondary" type="submit"
148               id="button-addon2">Comment</button>
149         </div>
150       </form>
151       <div th:each="comment : ${comments}" class="comment mb-3">
152         <div class="d-flex align-items-start justify-content-between">
153           <div class="d-flex align-items-center">
154             
157             <div>
158               <p class="mb-1">
159                 <a th:href="@{'/user/' + ${comment.user.id}}"
160                   class="text-primary fw-bold"> <span
161                     th:text="${comment.user.username}"></span>
162                   </a>: <span th:text="${comment.content}"></span>
163               </p>
164             </div>
165           <div class="ms-auto">
166             <a th:if="${comment.user.id == myUser.id}" th:href="@{/deleteComment/{id}(id=${comment.id})}"
167               class="btn btn-link text-danger">Delete</a>
168           </div>
169         </div>
170       </div>
171     </div>
172   </div>

```

```

76@      <script
77        src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
78        integrity="sha384-YvpcrYf0tY3LHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdsLK1eN7N6jIeHz"
79        crossorigin="anonymous"></script>
30  </body>
31
32 </html>
```

```

## 14. viewUser.html

```

1 !DOCTYPE html
2 <html xmlns:th="http://www.thymeleaf.org">
3
4@<head>
5 <meta charset="UTF-8">
6 <title>My Profile</title>
7 <link
8 href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
9 rel="stylesheet"
10 integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNLyT2bRjXh0JMhjY6hW+ALEwIH"
11 crossorigin="anonymous">
12<style>
13 body {
14 background-color: #0d3889;
15 color: white;
16 }
17
18 .profile-container {
19 max-width: 800px;
20 margin: 0 auto;
21 padding-top: 20px;
22 }
23
24 h1 {
25 font-size: 2rem;
26 color: white;
27 background-color: #ffae34;
28 text-align: center;
29 padding: 10px;
30 }
31
32 .profile-picture img {
33 border-radius: 50%;
34 border: 3px solid #ccc;
35 max-width: 150px;
36 max-height: 150px;
37 }
```

```

```
1 .user-info {
2   font-size: 1.2rem;
3   color: #ddd;
4   text-align: center;
5 }
6
7 h2 {
8   font-size: 1.5rem;
9   margin-top: 30px;
10  color: white;
11  text-align: center;
12 }
13
14 .card {
15   border: none;
16   border-radius: 8px;
17   box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
18   margin-bottom: 20px;
19   background-color: #fff;
20   color: #333;
21 }
22
23 .card-img-top {
24   height: 200px;
25   object-fit: cover;
26   border-top-left-radius: 8px;
27   border-top-right-radius: 8px;
28 }
29
30 .card-body {
31   padding: 15px;
32   background-color: #f8f9fa;
33   border-radius: 0 0 8px 8px;
34   display: flex;
35   flex-direction: column;
36   justify-content: space-between;
37 }
38
39 .card-title {
40   font-size: 1.2rem;
41   color: #333;
42   margin-bottom: 10px;
43 }
44
45 .card-text {
46   color: #666;
47   margin-bottom: 15px;
48 }
49
50 .card-footer {
51   padding: 10px;
52   background-color: #f8f9fa;
53   text-align: center;
54   border-top: 1px solid #dee2e6;
55   border-radius: 0 0 8px 8px;
56 }
57
58 .user-image {
59   width: 50px;
60   height: 50px;
61   border-radius: 50%;
62   margin-right: 10px;
63   object-fit: cover;
64 }
65
66 .btn {
67   background-color: #ffae34;
68   font-weight: bold;
69   color: white;
70   border-radius: 0.25rem;
71   padding: 0.375rem 0.75rem;
72   font-size: 0.875rem;
73   text-align: center;
74 }
```

```

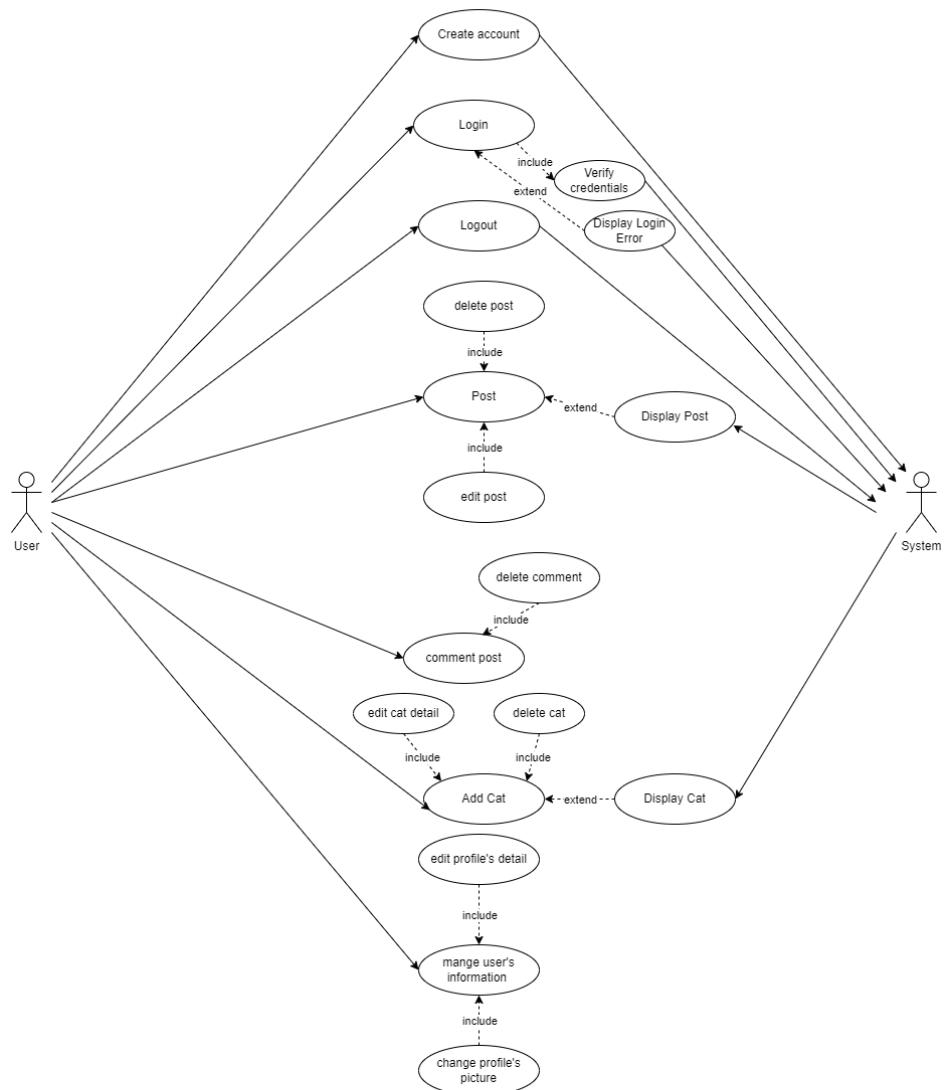
4 .btn:hover {
5   background-color: #ff8c00; /* เปลี่ยนสีเมื่อคลิก */
6   color: white;
7   transition: background-color 0.3s ease, color 0.3s ease;
8 }
9
10 .cats-list {
11   list-style: none; /* Remove bullet points */
12   padding: 0; /* Remove default padding */
13   display: flex; /* Use flexbox for horizontal alignment */
14   flex-wrap: wrap; /* Allow wrapping to the next line if necessary */
15 }
16
17 .cat-item {
18   display: flex; /* Use flex for item layout */
19   flex-direction: column; /* Arrange children in a column */
20   align-items: center; /* Center align the items */
21   margin-right: 20px; /* Space between items */
22   text-align: center; /* Center the text */
23 }
24
25 .cat-image {
26   width: 100px; /* Set the desired width */
27   height: auto; /* Maintain aspect ratio */
28   border-radius: 5px; /* Optional: round the corners */
29   margin-bottom: 5px; /* Space between image and name */
30 }
31
32 .cat-name {
33   font-size: 18px; /* Adjust font size as needed */
34   font-weight: bold; /* Make the name bold */
35 }
36 </style>
37 </head>
38
39 <body>
40   <div class="profile-container">
41     <h1>
42       Profile of <span th:text="${user.username}"></span>
43     </h1>
44     <nav aria-label="breadcrumb" class="p-2">
45       <ol class="breadcrumb">
46         <li class="breadcrumb-item"><a href="/">Home</a></li>
47         <li class="breadcrumb-item active text-light" aria-current="page">Profile</li>
48       </ol>
49     </nav>
50     <!-- User Picture -->
51     <div class="profile-picture text-center mb-4">
52       
53     </div>
54
55     <!-- User Information -->
56     <p class="user-info">
57       Username: <span th:text="${user.username}"></span>
58     </p>
59     <p th:if="${myUser.id == user.id}" class="user-info" >
60       Email: <span th:text="${user.email}"></span>
61     </p>
62
63     <div class="my-4 text-center">
64       <p class="fv-bold">
65         <a th:if="${myUser.id == user.id}" class="btn btn-warning" th:href="@{/editUser/{id}(id=${user.id})}">Edit Profile</a>
66       </p>
67     </div>
68
69     <h2>Cats</h2>
70     <ul class="cats-list cat">
71       <li th:each="cat : ${user.cats}" class="cat-item">
72         
73         <p th:text="${cat.name}" class="cat-name"></p>
74         <a th:href="@{/catDetail/{id}(id=${cat.id})}" class="btn btn-link text-light">View Detail</a>
75       </li>
76     </ul>
77
78     <!-- Posts Section -->
79     <h2>Posts</h2>
80     <div class="row justify-content-center">
81       <div th:each="post : ${user.posts}" class="col-md-6 col-lg-4 mb-4">
82         <div class="card h-100">
83           
85           <div class="card-body">
86             <h5 th:text="${post.title}" class="card-title"></h5>
87             <p th:text="${post.content}" class="card-text"></p>
88             <div class="d-flex justify-content-between">
89               <a th:href="@{/viewPost/{id}(id=${post.id})}" class="btn btn-primary">View Post</a>
90               <a th:if="${myUser.id == user.id}" th:href="@{/editPost/{id}(id=${post.id})}" class="btn btn-warning">Edit Post</a>
91               <a th:if="${myUser.id == user.id}" th:href="@{/deletePost/{id}(id=${post.id})}"
92                 class="btn btn-danger" onclick="return confirm('Are you sure you want to delete this post?');">Delete Post</a>
93             </div>
94           </div>
95         </div>
96       </div>
97     </div>
98   </div>
99
100   <script>
101     src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
102     integrity="sha384-Ypcryf0tY3LHB60NNkmXc5s9fDVZLESAAS5NDz0xhy9GkcIdslKleN7N6jIeHz"
103     crossorigin="anonymous"></script>
104 </body>
105 </html>

```

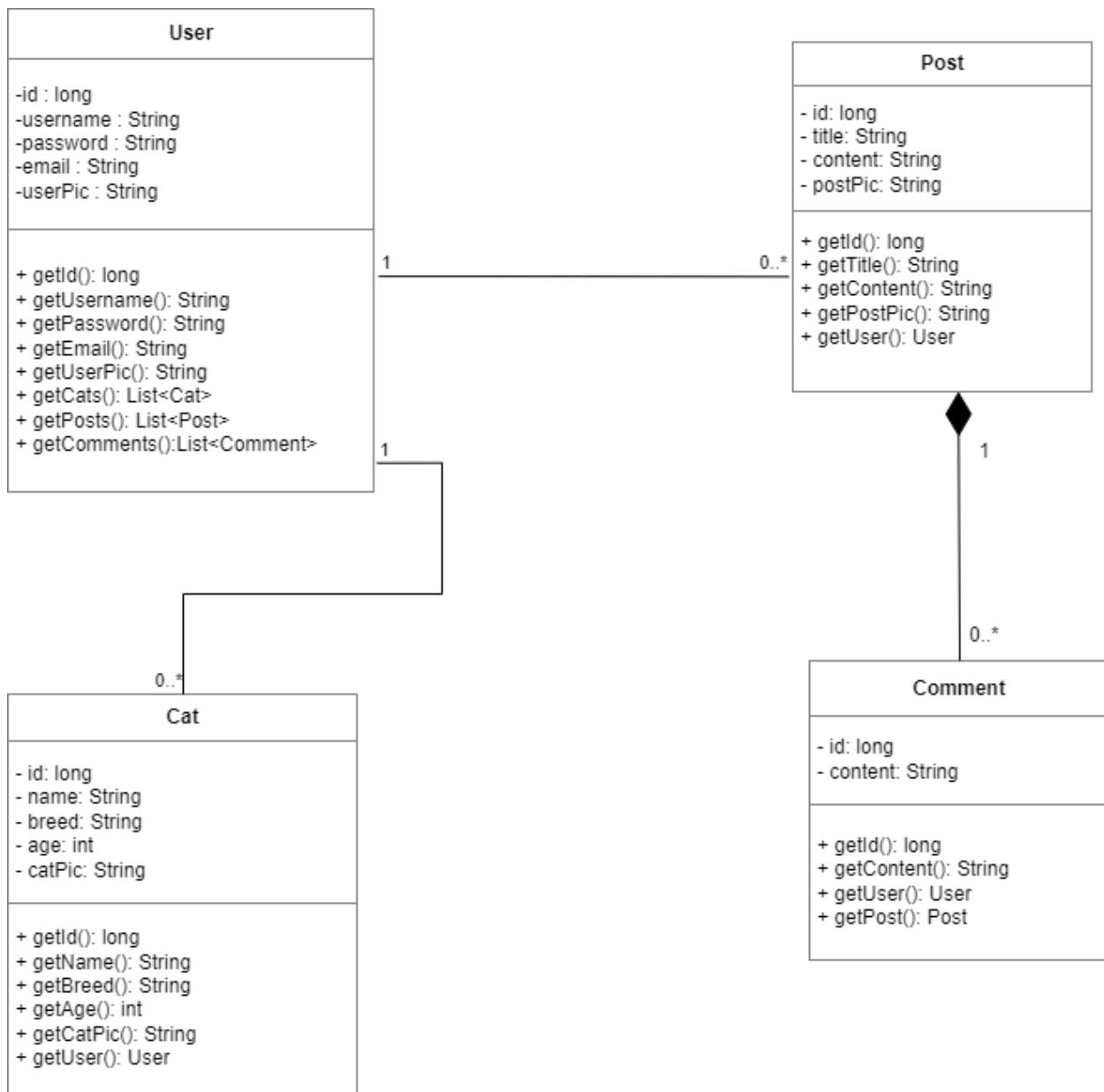
บทที่ 4

ภาพรวมการทำงานของระบบ

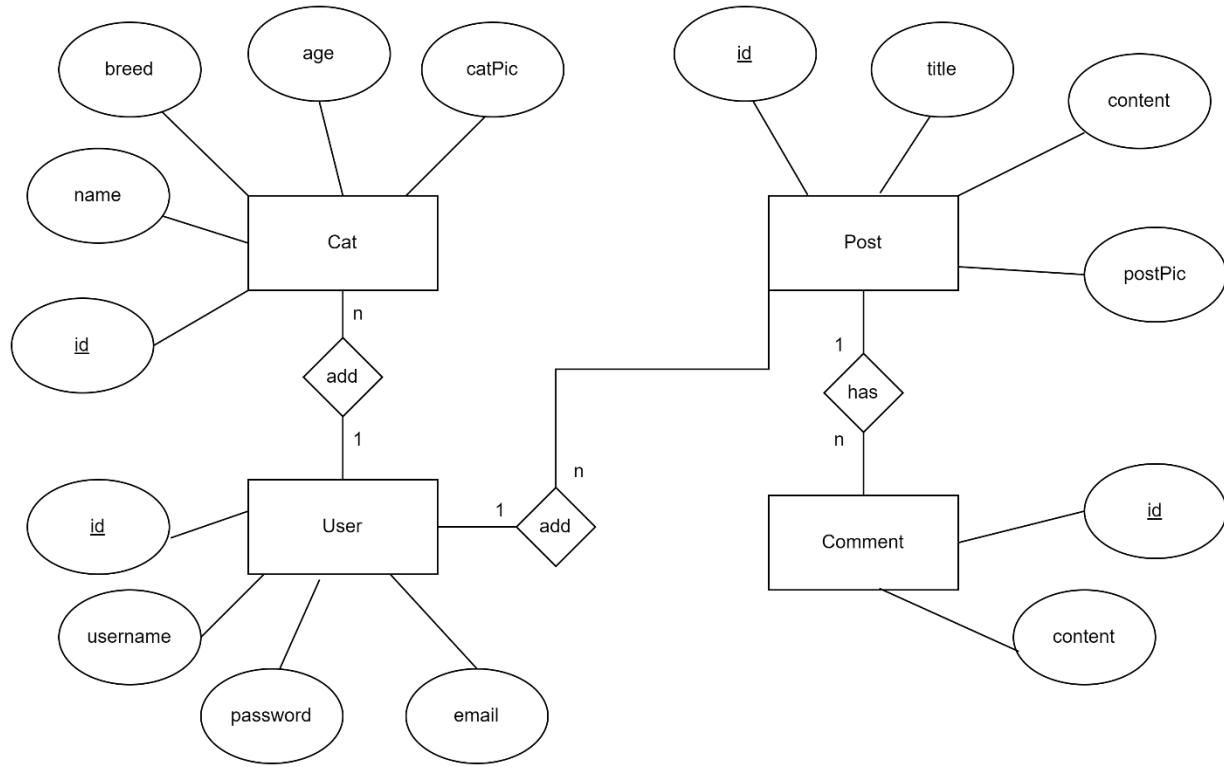
4.1 Use case diagram



4.2 Class diagram



4.3 ER diagram



4.4 User Interface

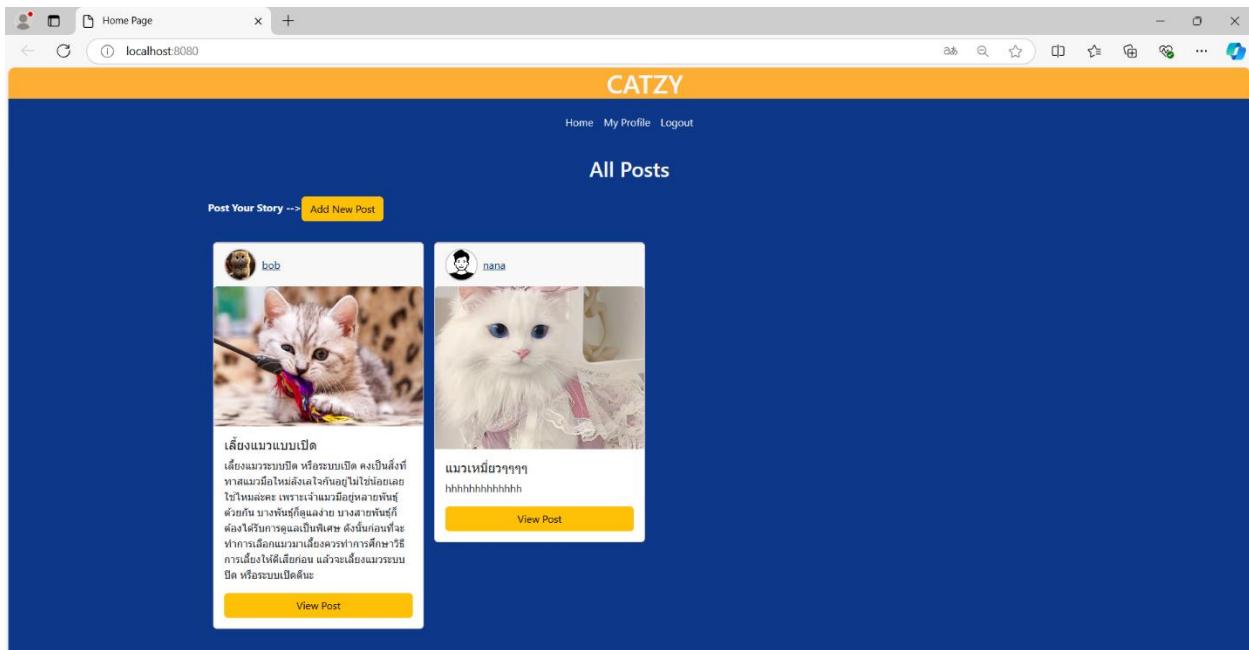
4.4.1 หน้า Login

The screenshot shows a web browser window with the URL `localhost:8080/login` in the address bar. The main content is a login form titled "Login". It contains two input fields: "Username" and "Password", both with placeholder text "Enter your Username" and "Enter your Password" respectively. Below the inputs is a blue "Login" button. At the bottom of the form, there is a link "Don't have an account? [Register here.](#)". The background of the browser window is orange.

4.4.2 หน้า Register

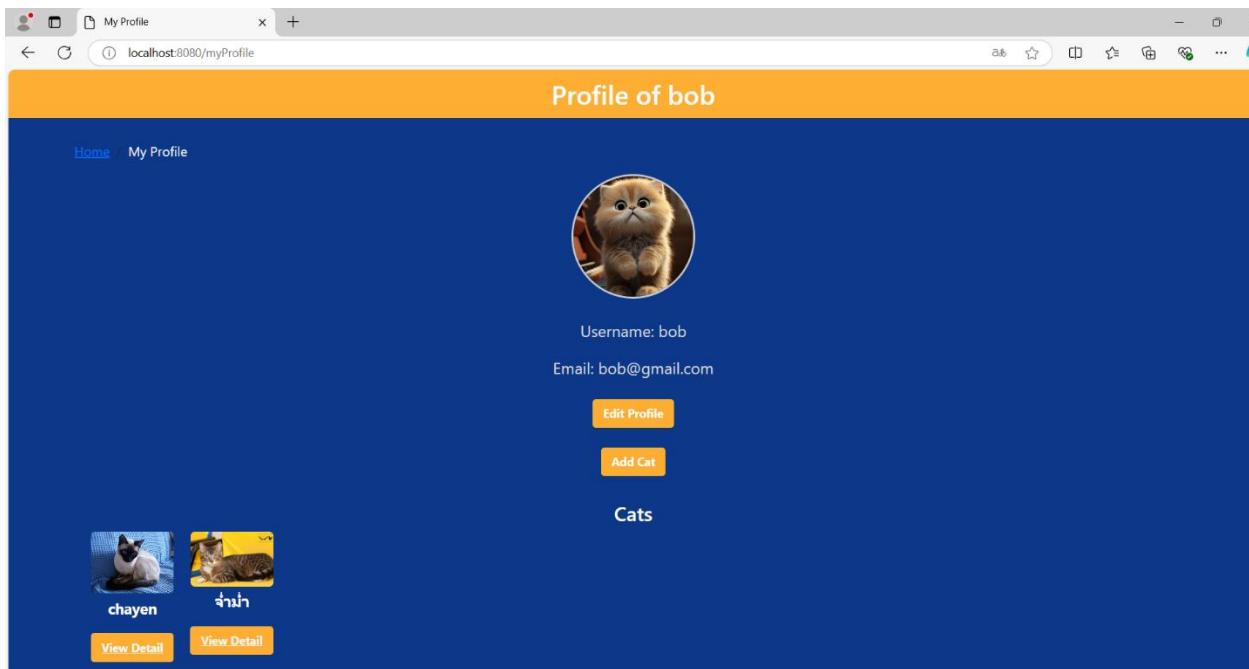
The screenshot shows a web browser window with the URL `localhost:8080/register` in the address bar. The main content is a register form titled "Register". It contains three input fields: "Username", "Email", and "Password", all with placeholder text "Enter your Username", "Enter your Email", and "Enter your Password" respectively. Below the inputs is a blue "Register" button. At the bottom of the form, there is a link "Already have an account? [Login here.](#)". The background of the browser window is orange.

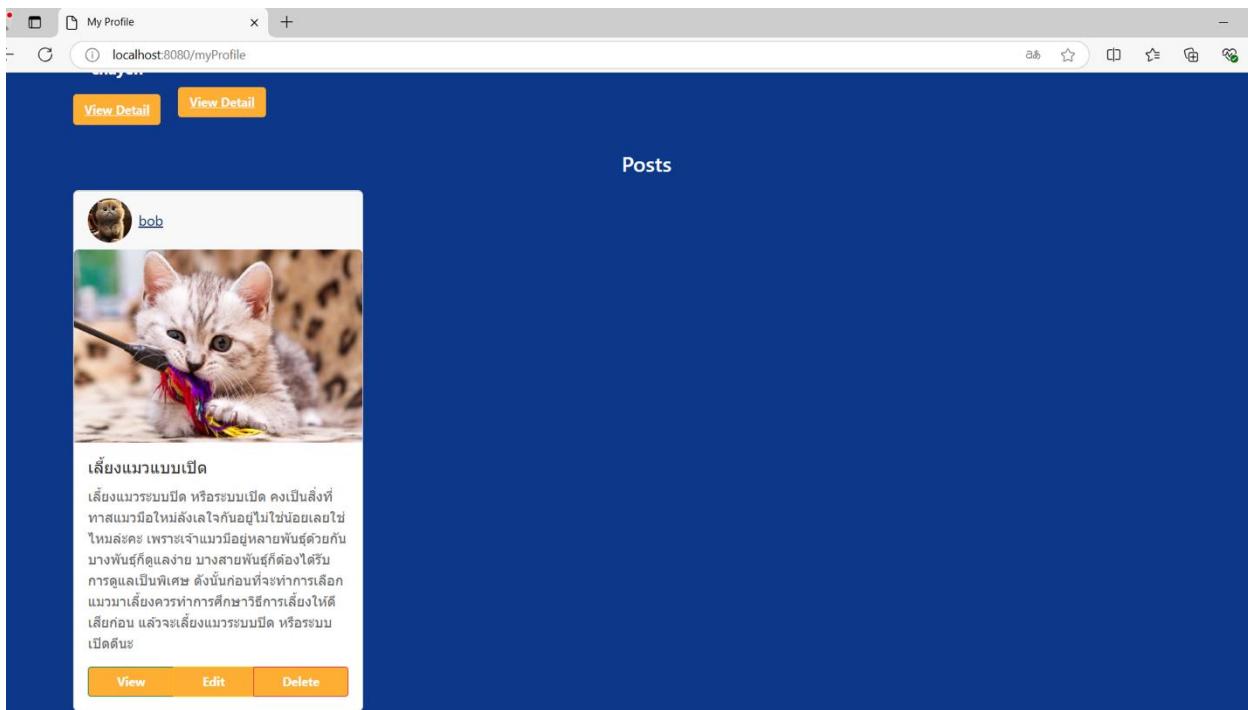
4.4.3 หน้า Home



จะเห็นทุกโพสต์ที่ User ทุกคนได้โพสต์

4.4.4 หน้า Profile





4.4.5 หน้าฟอร์มแก้ไข Post ของตนเอง

Edit Post

Title:
ເລື່ອມແນວມາປຶກ

Content:
ເລື່ອມແນວມາປຶກ ທີ່ໄດ້ຮັບມາປຶກ ດັ່ງນີ້ແລ້ວໃຫຍ່ໄດ້ກັບອຸນາໄນ້ນີ້ແນວຍໃຫ້ໄວ້ສະເໜີ ເພື່ອໄຈເມນວມືອ່ງຄາພິທັນຖຸວັນ ນາງພັນຖຸກິໂລງໄສຮັນກາຊຸແລ້ວເປັນ
ທີ່ເສຍ ດັ່ງນີ້ກອນທີ່ຈະໄດ້ການເລື່ອມແນວມາເສີ່ງຄວາມທີ່ການວິຊາວິຊາການເສີ່ງໄລ້ເລີຍກອນ ແລ້ວຈະເສີ່ງມາຮຽນມີດີ ທີ່ຈະຮັບມາປຶກ

Post Picture URL:
https://www.buzzpetsfood.com/wp-content/uploads/2021/02/shutterstock_1235997691.jpg

Save

4.4.6 អនុវត្តកិច្ច Profile

localhost:8080/editUser/1

Edit Profile

Username: lingling Kwong

Email: ling@gmail.com

Current Password: *****

New Password: *****

Profile Picture URL: <https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcQjr-6m4fJFTmH4slqOHIMjbQBTMw4JhG-CcVscjg5N-fDXF6ku>

Save

localhost:8080/myProfile

Profile of lingling Kwong

[Home](#) / [My Profile](#)



Username: lingling Kwong
Email: ling@gmail.com

[Edit Profile](#)
[Add Cat](#)

[Cats](#)
[Posts](#)

4.4.7 หน้าฟอร์มเพิ่มข้อมูลแมว

Create a New Cat

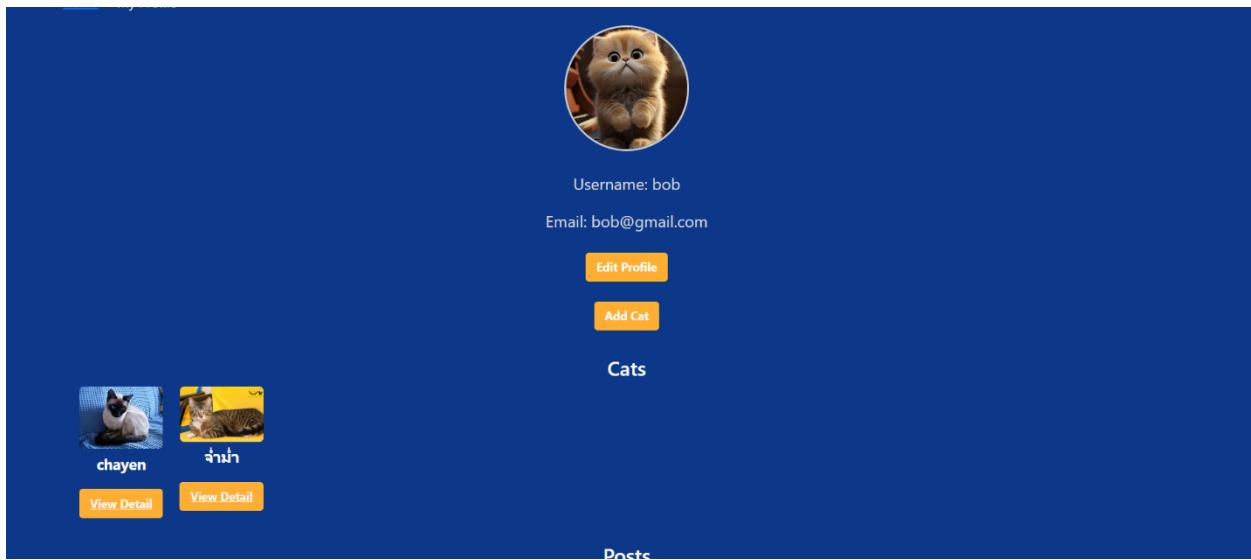
Name: จ่าแมว

Breed: สลัด

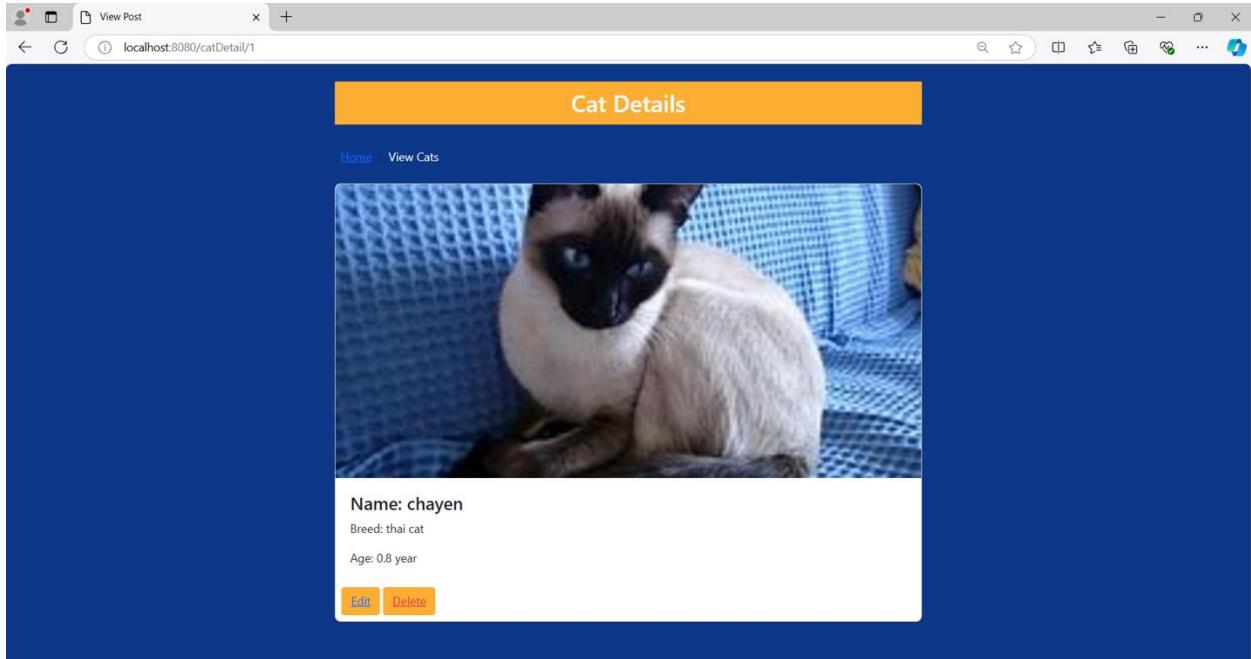
Age(Year): 0.5

Cat Picture URL: <https://i.pinimg.com/originals/84/8d/88/848d88ed938ea3340422150cf5a48c>

Save

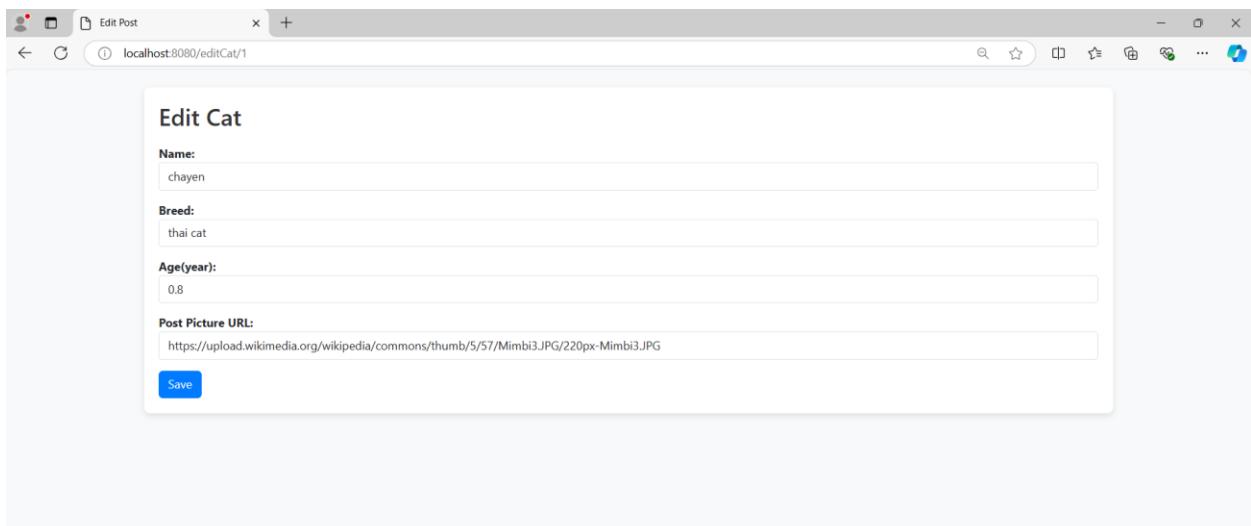


4.4.8 หน้าดูข้อมูลแมว

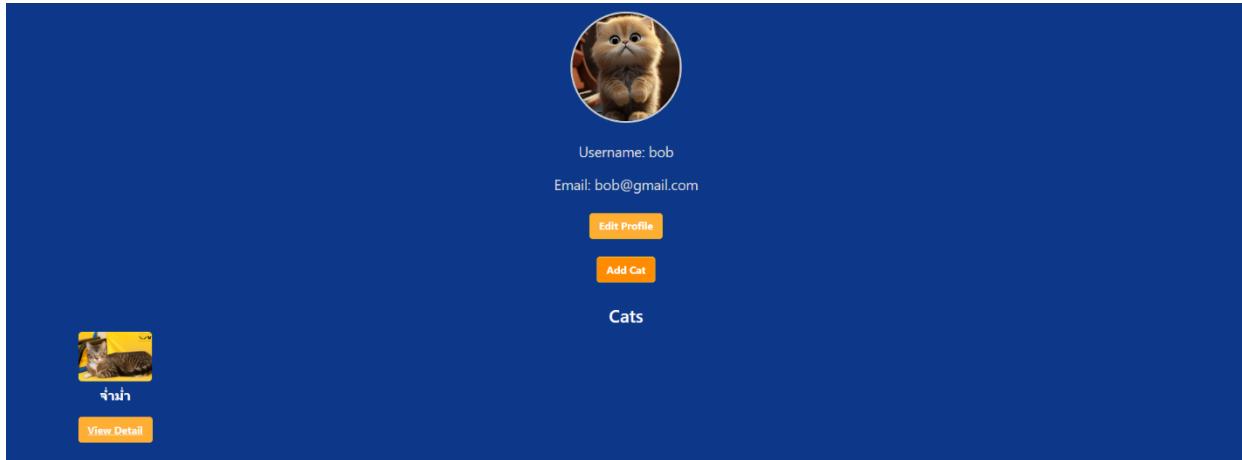


4.4.9 ของตัวเองสามารถแก้ไขและลบได้

เมื่อแก้ไข

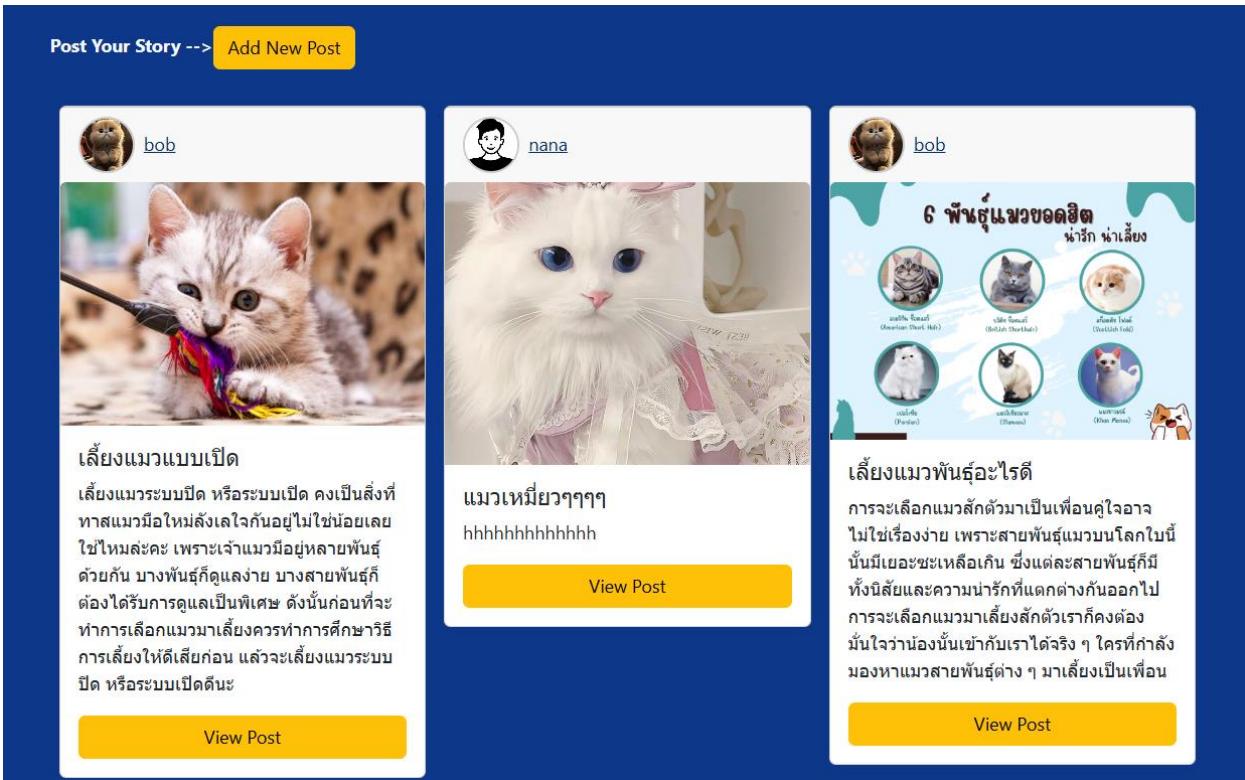


เมื่อคลิก



4.4.10 หน้าฟอร์มสำหรับเพิ่ม Post

A screenshot of a web browser window showing a "Create a New Post" form. The title bar says "Create a New Post". The form itself has a light gray background and a white header bar with the title "Create a New Post". It contains three input fields: "Title:" with an empty text input, "Content:" with an empty text input, and "Post Picture URL:" with an empty text input. At the bottom right of the form is a blue "Save" button. The browser's address bar shows "localhost:8080/addPost". The overall interface is clean and modern.



4.4.11 หน้า Post Detail

The image shows a detailed view of a post from the application:

Post Details

Post Content: A photo of a kitten playing with a colorful feather toy. The caption reads: "เลี้ยงแมวแบบเปิด" (Open cat breeding) and "เลี้ยงแมวแบบเปิด หรือระบบเปิด คือ เป็นสิ่งที่ทางแมวนี้มีให้แล้ว ไม่ต้องซื้อ ไม่ต้องขาย ในทุนลักษณะ เพราะเจ้าแมวนี้อยู่ห่างจากพันธุ์ดั้งเดิม นางพันธุ์ดูแลง่าย นางสายพันธุ์ดี ต้องได้รับการดูแลเป็นพิเศษ ดังนั้นก่อนที่จะทำการเลือกแมวมาเลี้ยงควรทำการศึกษาวิธีการเลี้ยงให้ดีเสียก่อน และจะเลี้ยงแมวระบบเปิด หรือระบบเปิดดีนั้น".

Comments:

- bob: 55555
- bob: 55555

สามารถดู comment ต่างๆ และลบ comment ได้หาก comment นั้นเป็นของตนเอง

Comments

hhhhhhhhhhhhh

Comment



[nana](#): 5555555555



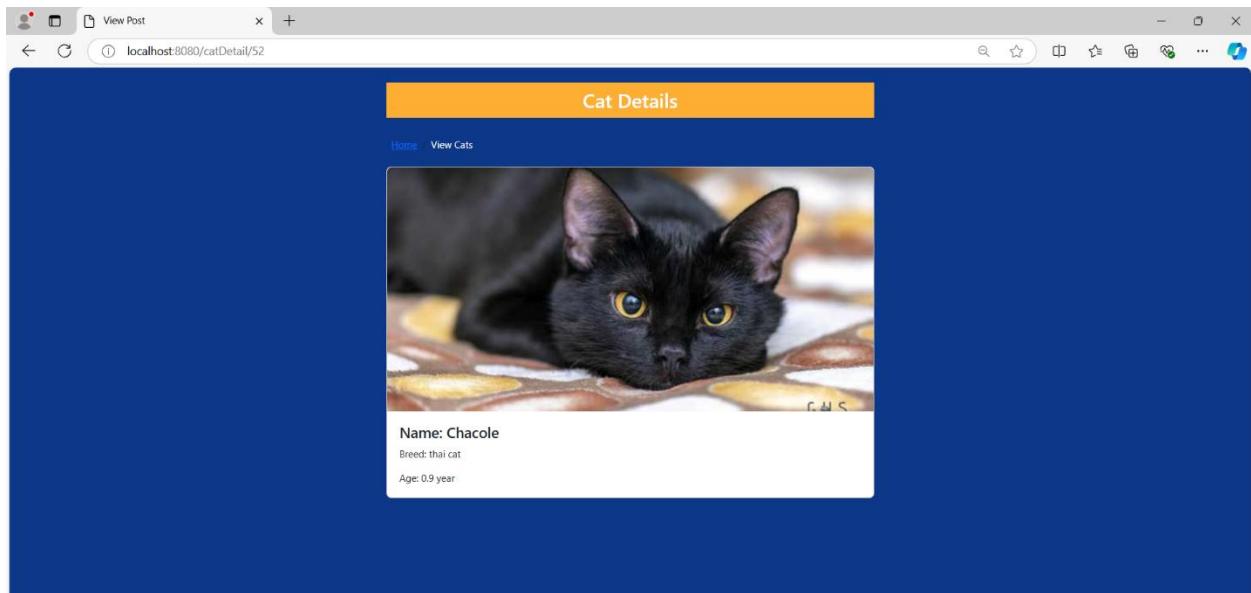
[bob](#): นำรักก็อก

Delete

4.4.12 หน้า Profile ของผู้ใช้คนอื่น

The screenshot shows a web browser window with a dark blue header bar. The title bar says "My Profile" and the address bar shows "localhost:8080/user/2". The main content area has a yellow header bar with the text "Profile of nana". Below it, there's a navigation bar with "Home" and "Profile".
The profile section features a circular profile picture of a person, followed by the text "Username: nana".
The "Cats" section contains a small thumbnail image of a black cat and the name "Chacole". There is also a "View Detail" button.
The "Posts" section displays a larger image of a white cat with blue eyes, along with some text:
แมวเหมียวๆๆๆ
hhhhhhhhhhhhh
View Post

4.4.13 หน้าข้อมูลแมวของผู้ใช้คนอื่น



บทที่ 5

สรุปผล

จากการจัดทำเว็บคอมมูนิตี้ของคนเลี้ยงแมว(Community website for cat owners) ผู้จัดทำได้จัดทำโครงงานจนได้ผลการดำเนินงานซึ่งสามารถสรุปผลและให้ข้อเสนอแนะได้ดังนี้

5.1 วัตถุประสงค์

5.1.1 เพื่อช่วยการแลกเปลี่ยนข้อมูล ความรู้ และเคล็ดลับในการดูแลแมวเป็นเรื่องที่เข้าถึงได้ง่ายที่ขึ้น

5.1.2 เพื่อช่วยให้การแบ่งปันเรื่องราวและประสบการณ์เกี่ยวกับแมวสามารถเข้าถึงได้ง่ายขึ้น

5.2 สรุปผล

ปัจจุบันมีแพลตฟอร์มในการพูดคุยแลกเปลี่ยนสาระ และความรู้มากมาย แต่สำหรับการพูดคุยแลกเปลี่ยนความรู้เฉพาะเรื่องนั้นยังไม่มากนัก ในบางครั้งการพูดคุย หรือโพสเกี่ยวกับเรื่องที่เราสนใจ อาจถือเป็นเรื่องไร้สาระ สำหรับผู้ที่มีความชอบหรือความสนใจต่างกัน นำมาซึ่งความไม่พอใจ อันเป็นเหตุให้เกิดการทะเลาะวิวาททางโลกออนไลน์ได้ โครงงานนี้จึงจัดทำขึ้นเพื่อร่วมกลุ่มผู้ที่มีความสนใจในเรื่องของแมวเพื่อพูดคุย แลกเปลี่ยนข้อมูล ความรู้ และช่วยให้คำแนะนำเกี่ยวกับแมว เพื่อยุติเหตุการณ์ดังกล่าวที่อาจเกิดขึ้นได้

5.4 ปัญหาอุปสรรค และ แนวทางแก้ไข

ปัญหาที่พบคือข้อจำกัดด้านเครื่องมือและความรู้ของผู้จัดทำที่ยังต้องมีการฝึกใช้งานเครื่องมือและภาษาเพิ่มเติม

5.5 ข้อเสนอแนะ

เว็บคอมมูนิตี้ของคนเลี้ยงแมวเป็นสิ่งที่ต้องดูแลอย่างต่อเนื่อง ผู้พัฒนาต้องปรับปรุงฟีเจอร์ใหม่ ๆ และอัปเดตฟังก์ชันเพื่อให้เว็บไซต์ให้ทันสมัยอยู่เสมอ

เอกสารอ้างอิง

[1] 1belief. (2567, 9 23). หลักการ ออกแบบเว็บ ขั้นพื้นฐาน พร้อมองค์ประกอบและรูปแบบโครงสร้าง. Retrieved from 1belief Web Service: <https://www.1belief.com/article/website-design/>

Amazon. (2567, 9 23). *IDE* (สภาพแวดล้อมสำหรับการพัฒนาแบบเบ็ดเสร็จ) คืออะไร. Retrieved from Amazon WEb Service: <https://aws.amazon.com/th/what-is/ide/>

Amazon. (2567, 9 23). Java คืออะไร. Retrieved from Amazon Web Service: <https://aws.amazon.com/th/what-is/java/>

Amazon. (2567, 9 23). SQL และ MySQL แตกต่างกันอย่างไร. Retrieved from Amazon Web Service: <https://aws.amazon.com/th/compare/the-difference-between-sql-and-mysql/>

aosoft. (2567, 9 23). *Eclipse คืออะไร?* Retrieved from aosoft Web Service:
<https://wwwaosoft.co.th/article/312/Eclipse->

¹ See also the discussion in *Journal of International Accounting Auditing and Taxation*, Vol. 16, No. 1, 2007, pp. 1-16.

businessplus. (2007, ๓ ๒๘). 7-C: WEB ที่สอนด้วยวิธีสอนแบบผู้สอน. Retrieved from businessplus.
<https://www.businessplus.co.th/Activities/นำสาร-hrm-c021/7-c-เพื่อการสื่อสารที่ดี-และประสบความสำเร็จ-v6090>

devhub. (2567, 9 23). *HTML គីអូខ្លួន (Introduction)*. Retrieved from Devhub Web Service:
<https://devhub.in.th/learn/html/what-is-html>

Jo. (2567, 9 23). *Spring Boot คืออะไร?* . Retrieved from Talance Hiring Guide:
<https://www.talance.tech/blog/talance-hiring-guide-spring-boot-developer>

Labz, S. (2567, 9 23). ชุมชนออนไลน์ (Online Community) ต่างจาก Social Media อย่างไร? Retrieved from Starfish Labz web: <https://www.starfishlabz.com/blog/1735-ชุมชนออนไลน์-online-community-ต่างจาก-social-media-อย่างไร>

เกียรติพงษ์, อ. (2567, 9 23). web นิยามศัพท์ที่พบในงานเว็บไซต์ (Website). Retrieved from iok2u.com:
<https://www.iok2u.com/article/information-technology/web-vocabulary>