เครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล

น.ส.อมรรัตน์ ใจมูล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2562

A Tool for Analyzing Impact to Hibernate Source Code and Test Cases for Database Schema Changes

Miss Amornrat Jaimoon

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2019

หัวข้อวิทยานิพนธ์	เครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล
โดย	น.ส.อมรรัตน์ ใจมูล
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

..... กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.ภัทรชัย ลลิตโรจน์วงศ์)

อมรรัตน์ ใจมูล : เครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล. (A Tool for Analyzing Impact to Hibernate Source Code and Test Cases for Database Schema Changes) อ.ที่ปรึกษาหลัก : รศ. ดร.ธาราทิพย์ สุวรรณศาสตร์

การออกแบบฐานข้อมูลเป็นส่วนสำคัญของการพัฒนาซอฟต์แวร์ เพราะมีการจัดการความสอดคล้องของข้อมูลและทำให้มั่นใจได้ว่าข้อมูลไม่ได้อยู่ในสภาพที่ไม่ถูกต้อง โดยในแต่ละรอบของการพัฒนาซอฟต์แวร์จะมีเปอร์เซ็นต์ของความซับซ้อนและขนาดของฐานข้อมูลที่มีแนวโน้มเพิ่มขึ้นอย่างมาก ซึ่งการเปลี่ยนแปลงสคีมาฐานข้อมูลจะส่งผลกระทบต่อซอร์สโค้ดที่เกี่ยวข้องรวมไปถึงกรณีทดสอบที่มักจะนำไปสู่ความล้มเหลวของกระบวนการซอฟต์แวร์ได้

นักวิจัยบางคนวิเคราะห์ผลกระทบจากการเปลี่ยนแปลงโดยใช้เทคนิคต่าง ๆ เช่น การจัดการตัวกำหนดค่า การตรวจสอบความสัมพันธ์ในรูปแบบย้อนกลับ และการแบ่งส่วนของโปรแกรม เป็นการวิเคราะห์ผลกระทบเพื่อระบุปัจจัยที่ทำให้เกิดการเปลี่ยนแปลงขึ้น แต่อย่างไรก็ตามการศึกษาเหล่านั้นไม่ได้เน้นถึงผลกระทบที่มีต่อซอร์สโค้ดและกรณีทดสอบ ดังนั้นงานวิจัยนี้นำเสนอเครื่องมือที่ใช้ในการวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล นอกจากนี้เครื่องมือยังสามารถแสดงให้ผู้ใช้ได้ทราบถึงตำแหน่งของผลกระทบที่เกิดขึ้นในซอร์สโค้ดและกรณีทดสอบรวมไปถึงตำแหน่งของซอร์สโค้ดและกรณีทดสอบที่ได้รับการปรับปรุงแก้ไขบนซอร์สโค้ดและการณีทดสอบที่ได้รับผลกระทบอีกด้วย สุดท้ายนี้เครื่องมือที่สร้างขึ้นจะถูกนำไปทดสอบกับกรณีศึกษา 3 กรณี ซึ่งพบว่าซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบจากการเปลี่ยนแปลงสคีมาฐานข้อมูลสามารถนำกลับไปใช้งานต่อได้อย่างสมบูรณ์เนื่องจากได้รับการปรับปรุงแก้ไขจากเครื่องมือที่สร้างขึ้น

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2562 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6070990421 : MAJOR SOFTWARE ENGINEERING

KEYWORD: Object-Relational Mapping (ORM), Hibernate ORM, Database Schema, Software Testing, Test Case

Amornrat Jaimoon : A Tool for Analyzing Impact to Hibernate Source Code and Test Cases for Database Schema Changes. Advisor: Assoc. Prof. TARATIP SUWANNASART, Ph.D.

Database design is an important part of software development because it manages the data consistency and makes sure that the data is not in an invalid state. In each software development cycle, the percentages of complexity and size of the database are likely to increase dramatically. Changes in a database schema impact related source code and test cases, which often lead to software process failure.

Some studies analyzed the impact of these changes using several techniques such as configuration management, traceability relationships, and program slicing to identify factors that caused changes. However, these studies do not emphasize the impact on source code and test cases. Therefore, this research proposes a tool to analyze impacts on Hibernate source code and test cases caused by changing the database schema. In addition, this tool can show the users about the location of the impact that occurs in the source code and test cases including the location of modification on affected source code and test cases. Finally, this tool will be tested with three case studies. The result shows that the affected Hibernate source code and the test cases can be fully reused as it has been modified by this tool.

Field of Study: Software Engineering Student's Signature .....

Academic Year: 2019 Advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความช่วยเหลือและความกรุณาอย่างสูงจาก รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่เสียสละเวลาให้คำแนะนำปรึกษา ชี้แนะแนวทางการทำวิจัย ตลอดจนคอยตรวจทานปรับปรุงแก้ไขข้อบกพร่องต่าง ๆ ที่เกิดขึ้นด้วยความเอาใจใส่อย่างดียิ่ง อีกทั้งยังให้ความรู้ในด้านต่าง ๆ ทั้งด้านวิชาการ ด้านประสบการณ์ชีวิต ผู้วิจัยตระหนักถึงความตั้งใจจริงและความทุ่มเทของอาจารย์จนทำให้ผู้วิจัยสามารถดำเนินงานวิจัยจนประสบความสำเร็จ และขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้

ขอขอบพระคุณรองศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี ประธานกรรมการสอบ ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์ และผู้ช่วยศาสตราจารย์ ดร.ภัทรชัย ลลิตโรจน์วงศ์ กรรมการสอบวิทยานิพนธ์ ที่ได้กรุณาสละเวลาและให้คำแนะนำเกี่ยวกับการทำวิจัย เพื่อให้วิทยานิพนธ์ฉบับนี้มีความสมบูรณ์และสำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณเหล่าคณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ได้มอบความรู้ทางด้านวิชาการและด้านต่าง ๆ ที่ล้วนแล้วแต่เป็นประโยชน์อย่างยิ่ง อีกทั้งรวมถึงบุคลากรทุกท่านในภาควิชาฯ ที่คอยให้ข้อมูล ให้คำแนะนำและความช่วยเหลือในระหว่างที่ผู้วิจัยกำลังศึกษาตลอดจนสอบวิทยานิพนธ์สำเร็จลุล่วง

ขอขอบคุณเพื่อน ๆ พี่ ๆ น้อง ๆ ทั้งในภาควิชาฯและที่ได้รู้จักเป็นการส่วนตัว ที่ได้ให้คำแนะนำช่วยเหลือ และเป็นแรงสนับสนุนแก่ผู้วิจัยเสนอมา

ขอขอบพระคุณบิดามารดาและพี่สาวของผู้วิจัย ที่คอยให้คำปรึกษาแนะนำ และคอยให้กำลังใจเสมอมาซึ่งเป็นสิ่งที่มีค่าที่สุดแก่ผู้วิจัย

อนึ่ง ผู้วิจัยหวังว่า วิทยานิพนธ์ฉบับนี้จะมีประโยชน์อยู่ไม่น้อย สำหรับข้อบกพร่องต่าง ๆ ที่อาจจะเกิดขึ้นนั้น ผู้วิจัยขอน้อมรับผิดเพียงผู้เดียว และยินดีที่จะรับฟังคำแนะนำจากทุกท่าน เพื่อเป็นประโยชน์ในการพัฒนางานวิจัยต่อไป

อมรรัตน์ ใจมูล

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ปัจจุบันการเขียนโปรแกรมเชิงวัตถุและระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management Systems) ได้ถูกใช้งานร่วมกันอย่างแพร่หลาย แต่เนื่องจากฐานข้อมูลเชิงสัมพันธ์ไม่สามารถเก็บข้อมูลเชิงวัตถุได้โดยตรง จึงมีความจำเป็นที่จะต้องเชื่อมต่อเทคโนโลยีทั้งสองเข้าด้วยกัน การแก้ปัญหาสองแบบที่ได้รับความนิยมอย่างแพร่หลายคือใช้การแปลงความสัมพันธ์เชิงวัตถุ (Object-Relational Mapping) หรือโออาร์เอ็ม (ORM) และอีกวิธีการคือการใช้งานระบบจัดการฐานข้อมูลเชิงวัตถุ (Object-Oriented Database Management Systems) แทนที่ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ซึ่งวิธีการนี้ก็ยังไม่ได้รับความนิยมมากนัก ปัจจุบันแนวคิดของการแปลงความสัมพันธ์เชิงวัตถุได้รับความนิยมเพิ่มสูงขึ้นทำให้โปรแกรมที่พัฒนาตามแนวความคิดนี้มีอย่างแพร่หลาย เช่น ไฮเบอร์เนต (Hibernate) เอ็นไฮเบอร์เนต (NHibernate) และ ไอบาทิส (IBatis) เป็นต้น โดยที่ยกตัวอย่างมาทั้งหมดเป็นเครื่องมือการแม็พแบบวัตถุเชิงสัมพันธ์ด้วยภาษาจาวา (Java) ที่กำลังได้รับความนิยมสำหรับใช้งานเป็น โอ-อาร์แม็พเปอร์ (O/R Mapper) ทั้งนี้ในงานวิจัยนี้สนใจแนวคิดของไฮเบอร์เนต เนื่องจากมีความสามารถและประสิทธิภาพค่อนข้างสูง อีกทั้งยังเป็นซอฟต์แวร์ประเภทโอเพนซอร์ส (Open Source) ซึ่งสามารถนำไปพัฒนาได้อย่างอิสระและไม่มีข้อจำกัดทางด้านลิขสิทธิ์จึงมีการใช้งานกันกว้างขวาง รวมไปถึงการพัฒนาโปรแกรมประยุกต์สามารถนำวิธีการไฮเบอร์เนตโอ-อาร์เอ็มเข้ามาใช้ในการช่วยจัดการฐานข้อมูลได้ ซึ่งจะกล่าวถึงรายละเอียดของไฮเบอร์เนตในทฤษฎีที่เกี่ยวข้องเป็นลำดับถัดไป

ในส่วนของการทดสอบซอฟต์แวร์ (Software Testing) ถือเป็นขั้นตอนที่สำคัญในวงจรการพัฒนาซอฟต์แวร์ (Software Development Life Cycle) เพื่อช่วยให้เกิดความชัดเจนว่าซอฟต์แวร์ที่พัฒนานั้นถูกต้องตรงตามความต้องการของผู้ใช้ มีประสิทธิภาพ และมีความน่าเชื่อถือ ดังนั้นขั้นตอนการทดสอบซอฟต์แวร์จึงเป็นขั้นตอนที่ต้องใช้ทั้งแรงงาน ระยะเวลา และงบประมาณที่ค่อนข้างสูง เนื่องจากกระบวนการทดสอบซอฟต์แวร์จะต้องทำการสร้างกรณีทดสอบ (Test Case) ที่ครอบคลุมกับระบบที่พัฒนามากที่สุด กรณีทดสอบจึงถือเป็นสิ่งจำเป็นสำหรับผู้ทดสอบเพื่อใช้ในกิจกรรมการทดสอบซอฟต์แวร์ โดยการออกแบบกรณีทดสอบสามารถแบ่งออกเป็น 2 ประเภทคือ แบบแบล็กบอกซ์ (Black-box) ที่จะเน้นเรื่องของฟังก์ชันการทำงานของซอฟต์แวร์โดยมองข้ามกลไกการทำงานภายในของซอฟต์แวร์ และมุ่งเน้นไปที่ผลลัพธ์ (Output) ที่ได้มาหลังจาก

การตอบสนองของซอฟต์แวร์เพียงอย่างเดียว และแบบไวท์บอกซ์ (White-box) ที่จะเน้นไปที่กลไกการทำงานภายในของซอฟต์แวร์โดยทำการตรวจสอบการทำงานของซอร์สโค้ด (Source code)

นอกจากนี้ฐานข้อมูล (Database) ได้เข้ามามีบทบาทและเป็นองค์ประกอบสำคัญส่วนหนึ่งในการพัฒนาซอฟต์แวร์ รวมไปถึงแอปพลิเคชันฐานข้อมูลที่ช่วยให้ผู้ใช้สามารถติดต่อกับฐานข้อมูลได้อย่างสะดวกมากยิ่งขึ้น ในขณะเดียวกันซอฟต์แวร์อาจมีการพัฒนาอย่างต่อเนื่องทำให้ซอฟต์แวร์ที่พัฒนาขึ้นมีขนาดใหญ่และซับซ้อนมากยิ่งขึ้น ฐานข้อมูลที่ถูกใช้ร่วมกันกับซอฟต์แวร์ย่อมมีขนาดใหญ่และซับซ้อนมากตามไปด้วย อีกทั้งหากมีการเปลี่ยนแปลงเกิดขึ้นภายในโครงร่างฐานข้อมูลหรือสคีมาฐานข้อมูล (Database Schema) อาทิเช่น การเพิ่มฟิลด์ (Field) การแก้ไขฟิลด์ หรือการลบฟิลด์ในตาราง (Table) เป็นต้น การเปลี่ยนแปลงดังกล่าวอาจส่งผลกระทบให้กับงานส่วนต่างๆ ของซอฟต์แวร์ ตลอดจนส่งผลกระทบถึงซอร์สโค้ดและกรณีทดสอบได้ในการสร้างกรณีทดสอบนั้นพบว่ามีเครื่องมือหลากหลายเครื่องมือที่สามารถช่วยสร้างกรณีทดสอบขึ้นมาจากขอบเขตของข้อมูลที่รับเข้ามา แต่ไม่ได้คำนึงถึงสคีมาของฐานข้อมูลที่ใช้ในการสร้างกรณีทดสอบ ซึ่งหากเกิดการเปลี่ยนแปลงสคีมาฐานข้อมูลย่อมส่งผลกระทบต่อซอร์สโค้ดและกระทบต่อกรณีทดสอบได้ เพราะเมื่อมีการเปลี่ยนแปลงซอร์สโค้ดอาจส่งผลกระทบต่อกรณีทดสอบด้วยเมื่อกรณีทดสอบนั้นถูกสร้างขึ้นมาจากไวท์บอกซ์ เพราะเหตุนี้จึงทำให้ผู้ทดสอบไม่สามารถทราบได้ว่ากรณีทดสอบใดบ้างที่ได้รับผลกระทบต่อการเปลี่ยนแปลงที่เกิดขึ้น และเมื่อผู้ทดสอบนำกรณีทดสอบที่ได้รับผลกระทบไปใช้งาน ก็อาจจะทำให้เกิดความผิดพลาดและความไม่ถูกต้องต่อการทดสอบ ส่งผลให้การทดสอบผิดพลาด ไม่ตรงตามความต้องการของลูกค้า และนำมาสู่การใช้แรงงาน และงบประมาณในการแก้ไขและพัฒนาซอฟต์แวร์จากการศึกษางานวิจัยที่เกี่ยวข้องกับการวิเคราะห์ผลกระทบเมื่อมีการแก้ไขสคีมาฐานข้อมูลพบว่า มีงานวิจัยที่ได้นำเสนอเกี่ยวกับแนวคิดการวิเคราะห์ผลกระทบต่อซอร์สโค้ดและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล [1] ซึ่งเป็นการวิเคราะห์หาข้อมูลจากล็อกไฟล์ (Log File) เพื่อหาการเปลี่ยนแปลงสคีมาฐานข้อมูลที่เกิดขึ้น โดยนำไปเปรียบเทียบกับเอกสารสคีมาของฐานข้อมูลก่อนการเปลี่ยนแปลง แล้ววิเคราะห์หาซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบ หากมีผลกระทบเกิดขึ้นจะทำการแสดงตำแหน่งที่เกิดผลกระทบในซอร์สโค้ดพร้อมทั้งปรับปรุงแก้ไขกรณีทดสอบ แต่แนวคิดดังกล่าวยังไม่ครอบคลุมไปถึงการแก้ไขซอร์สโค้ดจึงส่งผลให้การปรับปรุงแก้ไขกรณีทดสอบเพียงอย่างเดียวไม่สามารถทำงานได้อย่างถูกต้องเนื่องจากซอร์สโค้ดที่เชื่อมต่อกับกรณีทดสอบยังไม่ได้ถูกปรับปรุงแก้ไข ทั้งนี้ผู้ใช้ยังต้องนำเข้าไฟล์ล็อกและเอกสารสคีมาของฐานข้อมูลก่อนการเปลี่ยนแปลงเข้าสู่เครื่องมือด้วยตัวเอง เพื่อทำการหาสคีมาฐานข้อมูลที่เปลี่ยนแปลง ทำผู้ใช้ใช้เวลาอย่างมากในการเตรียมไฟล์ข้อมูล และผู้ใช้จะต้องมีความรู้ความเข้าใจพื้นฐานภาษาเอสคิวแอลในระดับเบื้องต้น เพื่อนำออกและนำเข้าเอกสารสคีมา

ฐานข้อมูลเข้าสู่เครื่องมือ ดังนั้นงานวิจัยนี้จึงนำเสนอแนวคิดและเครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบเมื่อมีการเปลี่ยนแปลงสคีมาฐานข้อมูล โดยเชื่อมต่อฐานข้อมูลของโปรแกรมอย่างอัตโนมัติเพื่อตรวจสอบการเปลี่ยนแปลงของสคีมาฐานข้อมูลที่เกิดขึ้น และวิเคราะห์หาผลกระทบต่อซอร์สโค้ดซึ่งเป็นการมุ่งเน้นการวิเคราะห์ผ่านกลไกการทำงานภายในของซอฟต์แวร์และกรณีทดสอบ หากมีผลกระทบเกิดขึ้นกับซอร์สโค้ดและกรณีทดสอบ เครื่องมือจะปรับปรุงซอร์สโค้ดในเบื้องต้นและปรับปรุงกรณีทดสอบเพื่อให้ข้อมูลในกรณีทดสอบที่ปรับปรุงมีความสอดคล้องกับข้อมูลภายในสคีมาฐานข้อมูลที่เกิดการเปลี่ยนแปลง และช่วยให้ผู้ทดสอบลดเวลาในการค้นหาผลกระทบและแก้ไขกรณีทดสอบใหม่ด้วยตนเองและสามารถนำกรณีทดสอบไปใช้งานได้

1.2 วัตถุประสงค์

1. เพื่อพัฒนาเครื่องมือการวิเคราะห์ผลกระทบของซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบที่ได้รับผลกระทบจากการเปลี่ยนแปลงของสคีมาฐานข้อมูล
2. เพื่อแก้ไขซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบจากการเปลี่ยนแปลงของสคีมาฐานข้อมูล จากการเปลี่ยนแปลงอินพุตของความต้องการเชิงฟังก์ชัน

1.3 ขอบเขตการวิจัย

1. เครื่องมือสามารถนำเข้าไฟล์ได้ 3 ชนิดดังต่อไปนี้
   1.1. ไฟล์กำหนดค่า (Configuration) ในรูปแบบไฟล์เอกซ์เอ็มแอล
   1.2. ไฟล์กรณีทดสอบในรูปแบบไฟล์เอกซ์เซล
   1.3. ไฟล์ซอร์สโค้ดในรูปแบบไฟล์จาวา ซึ่งประกอบด้วยไฟล์ 2 รูปแบบดังนี้
      1) เพอซิสเทินคลาส (Persistent Class)
      2) คอนโทรลเลอร์คลาส (Controller Class)
2. กรณีทดสอบจะสอดคล้องกับซอร์สโค้ดที่อยู่ในคอนโทรลเลอร์คลาสเท่านั้น
3. กรณีทดสอบประกอบด้วยข้อมูลดังต่อไปนี้
   3.1. หมายเลขกรณีทดสอบ
   3.2. เส้นทางเดิน
   3.3. ชื่อฟิลด์
   3.4. ชนิดของข้อมูล
   3.5. ขนาดของข้อมูล
   3.6. ค่าของข้อมูล
   3.7. ผลลัพธ์ของข้อมูล

4. ฐานข้อมูลที่เชื่อมต่อกับซอร์สโค้ดไฮเบอร์เนตต้องเป็น MySQL เท่านั้น
5. สคริปต์ไฟล์สคีมาฐานข้อมูลที่นำออกและนำเข้าต้องอยู่ในรูปแบบ .sql เท่านั้น
6. การเปลี่ยนแปลงของสคีมาฐานข้อมูลครอบคลุมทั้งหมด 7 รูปแบบดังนี้
   6.1. การแก้ไขชื่อตาราง
   6.2. การลบตาราง
   6.3. การแก้ไขชื่อฟิลด์
   6.4. การแก้ไขประเภทข้อมูลของฟิลด์
   6.5. การแก้ไขขนาดข้อมูลของฟิลด์
   6.6. แก้ไขเปลี่ยนแปลงคีย์หลักของฟิลด์
   6.7. การลบฟิลด์
7. เครื่องมือจะไม่รองรับตัวแปรที่ไม่เกี่ยวข้องกับสคีมาฐานข้อมูลในไฟล์ซอร์สโค้ด
8. ผลลัพธ์ของเครื่องมือสามารถปรับปรุงไฮเบอร์เนตซอร์สโค้ดและกรณีทดสอบได้
9. กรณีมีผลกระทบเกิดขึ้นและเครื่องมือไม่สามารถแก้ไขซอร์สโค้ดได้เนื่องจากมีข้อผิดพลาดเกิดขึ้นกับซอร์สโค้ดคอนโทรลเลอร์ เครื่องมือจะไม่แก้ไขซอร์สโค้ดแต่จะแสดงรายงานสรุปผลให้ผู้ใช้ได้รับทราบ
10. สามารถนำออกไฟล์ซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบที่ได้รับการปรับปรุงแก้ไขได้
11. ทดสอบเครื่องมือที่พัฒนาด้วยระบบงานอย่างน้อย 3 ระบบ

1.4 ขั้นตอนและวิธีดำเนินงานวิจัย

1. สำรวจและศึกษาเครื่องมือทดสอบ
2. ศึกษางานที่เกี่ยวข้อง
3. ศึกษารูปแบบสคีมาฐานข้อมูล และการกำหนดเงื่อนไขข้อบังคับของฐานข้อมูล
4. กำหนดขอบเขตการเปลี่ยนแปลงของสคีมาฐานข้อมูล
5. ออกแบบเครื่องมือสนับสนุน
   5.1. ออกแบบส่วนต่อประสานกับผู้ใช้
   5.2. ออกแบบโครงสร้างการจัดเก็บข้อมูล
   5.3. ออกแบบโครงสร้างเครื่องมือ และกำหนดข้อมูลนำเข้าที่นำมาใช้พัฒนาเครื่องมือ

6. กำหนดขอบเขตความสามารถของเครื่องมือในการแก้ไขปรับปรุงซอร์สโค้ดและกรณีทดสอบ

7. พัฒนาเครื่องมือสำหรับวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบจากการเปลี่ยนแปลงสคีมาฐานข้อมูลตามที่ได้ออกแบบไว้

8. ทดสอบเครื่องมือที่ถูกพัฒนาตามขอบเขตที่กำหนดไว้

9. สรุปและประเมินผลการทดสอบ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

เครื่องมือสนับสนุนที่ถูกพัฒนาขึ้นจากวิธีที่นำเสนอสามารถช่วยลดเวลาในการตรวจหาผลกระทบต่อซอร์สโค้ดและกรณีทดสอบจากการเปลี่ยนแปลงของสคีมาฐานข้อมูล

1.6 บทความวิชาการที่ได้รับการตีพิมพ์

งานวิจัยนี้ได้รับคัดเลือกและตีพิมพ์เป็นบทความวิชาการเรื่อง “Impact Analysis of Database Schema Changes on Hibernate Source Code and Test Cases” โดย อมรรัตน์ ใจมูล และ ธาราทิพย์ สุวรรณศาสตร์ ในการประชุมวิชาการ “2019 The 3rd International Conference on Software and e-Business (ICSEB 2019)” ระหว่างวันที่ 9-10 ธันวาคม 2562 ณ มหาวิทยาลัยวาเซดะ (Waseda University) ประเทศญี่ปุ่น

บทที่ 2

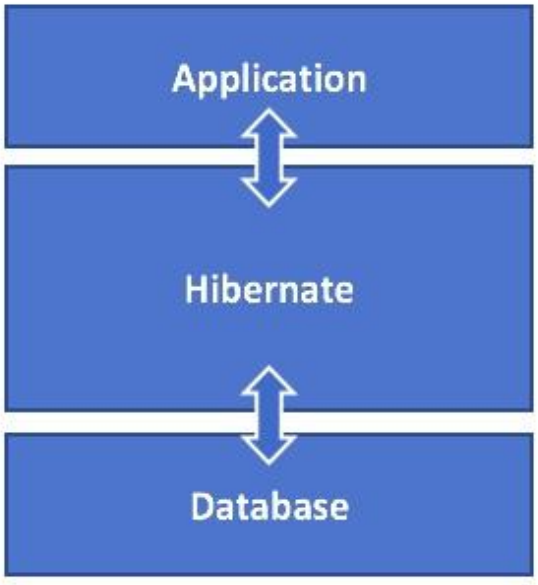
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1. งานวิจัยที่เกี่ยวข้อง

วิทยานิพนธ์นี้ได้อ้างอิงถึงทฤษฎีที่เกี่ยวข้องเพื่อนำมาใช้ประโยชน์ในขั้นตอนการวิจัย ซึ่งประกอบด้วยทฤษฎีของไฮเบอร์เนต การทดสอบซอฟต์แวร์ การทดสอบแบบไวท์บอกซ์ กรณีทดสอบ และสคีมาฐานข้อมูล รายละเอียดดังนี้

2.1.1. ไฮเบอร์เนต

ไฮเบอร์เนต [2] เป็นรูปแบบของจาวาอีกรูปแบบหนึ่งที่ใช้ในการจัดการข้อมูลแบบโมเดลเชิงวัตถุและเชิงสัมพันธ์กับฐานข้อมูล เพื่ออำนวยความสะดวกในการเชื่อมต่อข้อมูลจากฐานข้อมูลอัตโนมัติกลับไปกลับมาได้ รวมไปถึงการเข้าถึงข้อมูล และการเรียกค้นข้อมูล ซึ่งจะช่วยให้โปรแกรมเมอร์สามารถทำงานได้อย่างมีประสิทธิภาพ โดยโครงสร้างการทำงานของไฮเบอร์เนตจะทำหน้าที่เป็นตัวกลางในการสื่อสารระหว่างชั้นของแอปพลิเคชัน (Application Layer) และชั้นของฐานข้อมูล (Database Layer) ดังรูปที่ 2-1



รูปที่ 2-1 โครงสร้างไฮเบอร์เนต

การทำงานของไฮเบอร์เนตจะแบ่งเป็น 3 ส่วนหลักๆ คือ

1) การเชื่อมต่อฐานข้อมูลด้วยไฟล์กำหนดค่า (Configuration FIle) ในไฟล์นี้จะเป็นส่วนที่ใช้ในการเชื่อมต่อกับฐานข้อมูลซึ่งโปรแกรมเมอร์จะต้องทำการระบุสคีมาฐานข้อมูลที่ต้องการใช้และระบุรหัสผู้ใช้งานเข้ามา เพื่อให้เครื่องมือสามารถเชื่อมต่อกับฐานข้อมูลได้อย่างอัตโนมัติ

2) การเชื่อมต่อข้อมูลตารางฐานข้อมูลและฟิลด์ในจาวาซอร์สโค้ดด้วยไฟล์ซอร์สโค้ดเพอซิสเทิน เป็นการเชื่อมต่อข้อมูลตารางฐานข้อมูลที่ต้องการเรียกใช้ โดยมีการเชื่อมต่อชนิดของข้อมูลระหว่างซอร์สโค้ดไฮเบอร์เนตและฟิลด์ในสคีมาฐานข้อมูลแสดงดังตารางที่ 2-1 และเชื่อมต่อตาราง ฟิลด์ จากฐานข้อมูลเข้ากับคลาส และแอตทริบิวต์ (Attribute) โดยเรียกใช้ผ่านคำสั่งแอนโนเทชัน (Annotation) ดังแสดงในรูปที่ 2-2

3) โปรแกรมหลัก (Main Program) หรือที่เรียกกันว่าคอนโทรลเลอร์ (Controller) เป็นส่วนของโปรแกรมหลักที่จะนำค่าของข้อมูลที่ได้จากการเชื่อมต่อจากฐานข้อมูลในไฟล์เพอซิสเทินไปใช้งาน โดยจะมีการเรียกใช้ไฟล์เพอซิสเทินด้วยการนำเข้าด้วยคำสั่ง Import ดังรูปที่ 2-3 ที่แสดงซอร์สโค้ดคอนโทรลเลอร์ที่นำเข้าไฟล์เพอซิสเทิน Note ด้วยคำสั่ง import

ตารางที่ 2-1 การเชื่อมต่อชนิดของฟิลด์ในซอร์สโค้ดไฮเบอร์เนตและสคีมาฐานข้อมูล [3]

Hibernate Code	MYSQL
integer	INTEGER
long	BIGINT
short	SMALLINT
float	FLOAT
double	DOUBLE
big_decimal	NUMERIC
character	CHAR(1)
string	VARCHAR

ตารางที่ 2-1 การเชื่อมต่อชนิดของฟิลด์ในซอร์สโค้ดไฮเบอร์เนตและฐานข้อมูล [3] (ต่อ)

Hibernate Code	MYSQL
byte	TINYINT
boolean	BIT
date	DATE
time	TIME
binary	VARBINARY (or BLOB)
text	CLOB
clob	CLOB
blob	BLOB

```
1  package com.example.easynotes.model;
2
3  import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
11
12 @Entity
13 @Table(name = "notes")
14 @EntityListeners(AuditingEntityListener.class)
15 @JsonIgnoreProperties(value = {"createdAt", "updatedAt"},
16         allowGetters = true)
17 public class Note {
18     @Id
19     @GeneratedValue(strategy = GenerationType.IDENTITY)
20     @Column(name = "NOTEID")
21     private Long id;
22
23     @Column(name = "POST_ID")
24     private Long post_id;
25
26     @Column(name = "TITLE")
27     @NotBlank
28     private String title;
29
30     @Column(name = "CONTENT")
31     @NotBlank
32     private String content;
```

รูปที่ 2-2 ไฮเบอร์เนตในรูปแบบเพอซิสเทิน

```
1  package com.example.easynotes.controller;
2
3  import com.example.easynotes.exception.ResourceNotFoundException;
4  import com.example.easynotes.model.Note;
5  import com.example.easynotes.repository.NoteRepository;
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.boot.SpringApplication;
8  import org.springframework.http.ResponseEntity;
9  import org.springframework.web.bind.annotation.*;
10
11 import javax.validation.Valid;
12 import java.util.List;
13 import java.util.Optional;
14
15 @RestController
16 @RequestMapping("/api")
17 public class NoteController implements java.io.Serializable  {
18
19     @Autowired
20     NoteRepository noteRepository;
21
22     @GetMapping("/notes")
23     public List<Note> getAllNotes()
24     {
```

รูปที่ 2-3 ไฮเบอร์เนตในรูปแบบคอนโทรลเลอร์

2.1.2. การทดสอบซอฟต์แวร์

การทดสอบซอฟต์แวร์ [4] หมายถึง กระบวนการในการใช้งานหรือประเมินค่าซอฟต์แวร์โดยการทดลองใช้ซอฟต์แวร์อย่างมีแนวทาง โดยใช้ความรู้ทางด้านเทคนิค เพื่อให้สามารถระบุหรือค้นหาความผิดพลาด (Error) ของซอฟต์แวร์ที่อาจจะซ่อนอยู่ให้ปรากฏออกมา และสามารถระบุถึงแนวทางการเกิดปัญหา พร้อมสมมติฐานของความผิดพลาดที่เกิดขึ้นได้ การทดสอบซอฟต์แวร์สามารถทำได้ทั้งการทำด้วยมือ หรือทำอย่างอัตโนมัติ เพื่อที่จะตรวจสอบว่าซอฟต์แวร์เป็นไปตามความต้องการของซอฟต์แวร์ (Software Requirements) หรือเพื่อที่จะระบุความแตกต่างระหว่างผลลัพธ์ที่ได้จริงจากซอฟต์แวร์ การทดสอบซอฟต์แวร์แบ่งได้เป็น 4 ระดับดังนี้

1) การทดสอบหน่วยโปรแกรม (Unit Testing) เป็นการทดสอบหน่วยโปรแกรมหลังจากที่เขียนโปรแกรมในหน่วยนั้นเสร็จแล้ว

2) การทดสอบแบบบูรณาการ (Integration Testing) เป็นการทดสอบซอฟต์แวร์โดยนำแต่ละหน่วยโปรแกรม ที่ผ่านการทดสอบหน่วยโปรแกรมมาแล้วมาทำงานร่วมกัน

3) การทดสอบระบบ (System Testing) เป็นการทดสอบระบบหรือโปรแกรมโดยดูภาพรวมของการทำงาน ว่ามีการตอบสนองความต้องการทั้งในส่วนของฟังก์ชันการทำงานและประสิทธิภาพการทำงาน ว่าสอดคล้องกับลักษณะของความต้องการของซอฟต์แวร์ (Requirement Specification) หรือไม่

4) การทดสอบการตรวจรับ (Acceptance Testing) เป็นกระบวนการทดสอบระบบขั้นตอนสุดท้ายเพื่อให้แน่ใจว่า ระบบที่พัฒนาพร้อมที่จะใช้งานได้จริง ตรงตามกระบวนการทางธุรกิจ (Business Process) และความต้องการของผู้ใช้งานที่ได้กำหนดไว้ โดยผลลัพธ์การทดสอบจะต้องเป็นไปตามเงื่อนไขความสมบูรณ์ของระบบที่ควรจะเป็นและสามารถยอมรับได้ (Acceptance Criteria) ซึ่งได้ร่วมกันกำหนดขึ้นระหว่างผู้ใช้งานระบบกับทีมงานพัฒนาระบบรวมถึงส่วนงานอื่นๆ ที่เกี่ยวข้อง

2.1.3. การทดสอบแบบไวท์บอกซ์

การทดสอบแบบไวท์บอกซ์ [4] เป็นเทคนิคที่ใช้ทดสอบซอฟต์แวร์ หรือที่เรียกว่าการทดสอบเชิงโครงสร้าง ซึ่งวิธีการนี้จะพิจารณากลไกภายในของระบบหรือทางเดินในโปรแกรมโดยจะมุ่งเน้นไปที่โครงสร้างการทำงานภายในของโปรแกรม นักพัฒนา (Developer) จะทราบถึงรายละเอียดการทำงานภายในของโปรแกรม และผู้ทดสอบจะต้องออกแบบกรณีทดสอบให้ขึ้นกับการทำงานภายในของโปรแกรม เพื่อให้การทดสอบเป็นไปตามผลลัพธ์ที่คาดหวังของวัตถุประสงค์เชิงธุรกิจ (Business Requirement) และบ่งชี้ให้เห็นถึงการทำงานที่ถูกต้อง ทั้งนี้เพื่อให้ได้ซอฟต์แวร์ที่มีประสิทธิภาพตรงตามความต้องการของผู้ใช้

2.1.4. กรณีทดสอบ

กรณีทดสอบ เป็นชุดข้อมูลทดสอบที่ใช้ทดสอบโปรแกรมหรือการทำงานของโปรแกรมขึ้นอยู่กับวัตถุประสงค์ของระบบหรือความต้องการเชิงธุรกิจ โดยการออกแบบกรณีทดสอบสามารถแบ่งออกเป็น 2 ประเภทคือ แบบแบล็กบอกซ์ ที่จะเน้นเรื่องของฟังก์ชันการทำงานของซอฟต์แวร์โดยมองข้ามกลไกการทำงานภายในของซอฟต์แวร์ และมุ่งเน้นไปที่ผลลัพธ์ (Output) ที่ได้มาหลังจากการตอบสนองของซอฟต์แวร์เพียงอย่างเดียว และแบบไวท์บอกซ์ ที่จะเน้นไปที่กลไกการทำงานภายในของซอฟต์แวร์โดยตรวจสอบการทำงานของซอร์สโค้ด และครอบคลุมการทดสอบในด้านทางเดินการไหลของข้อมูล

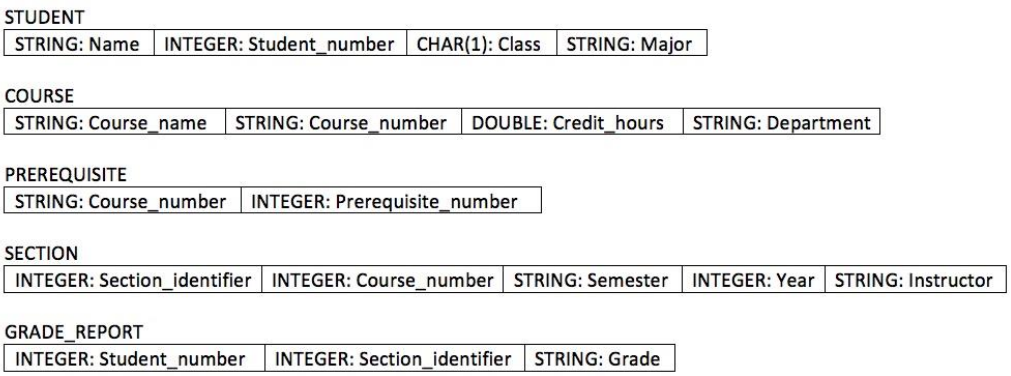
กรณีทดสอบมีหลากหลายรูปแบบ ทั้งนี้ขึ้นอยู่กับวัตถุประสงค์ของระบบเพื่อให้เป็นไปตามแนวทางที่วางแผนไว้ โดยชนิดของการทดสอบนั้นจะก่อให้เกิดความแตกต่างของกรณีทดสอบที่นำมาทดสอบโปรแกรม หรืออาจจะออกแบบกรณีทดสอบให้เกิดความยืดหยุ่น เพื่อเป็นไปตามวัตถุประสงค์ของแต่ละระบบงาน ซึ่งออกแบบโดยผู้ทดสอบโปรแกรม (Tester) โดยทั่วไปแล้วกรณีทดสอบประกอบด้วยโครงสร้างดังต่อไปนี้

- หมายเลขกรณีทดสอบ (Test Case ID)
- เงื่อนไขก่อนเริ่มทำการทดสอบ (Preconditions)

- ข้อมูลนำเข้าที่ใช้ทดสอบ (Input)
- ประเภทของกรณีทดสอบ แบ่งออกเป็น 2 ประเภทคือ กรณีที่ถูกต้อง (Valid) และกรณีที่ไม่ถูกต้อง (Invalid)
- ผลลัพธ์ที่คาดหวัง (Expected Output)

2.1.5. สคีมาฐานข้อมูล

สคีมาฐานข้อมูล [5] คือ โครงสร้างของฐานข้อมูลหรือนิยามข้อมูล ซึ่งเป็นโครงร่างที่ได้จากการออกแบบฐานข้อมูล โดยมีการกำหนด ชื่อตารางฐานข้อมูล ชื่อแอตทริบิวต์ ชนิดของข้อมูล ขนาดของข้อมูล ความสัมพันธ์ระหว่างข้อมูล และคุณสมบัติของข้อมูล ดังแสดงในรูปที่ 2-4



รูปที่ 2-4 แผนภาพสคีมาฐานข้อมูล

จากรูปที่ 2-4 แสดงตัวอย่างโครงสร้างของฐานข้อมูลของระบบฐานข้อมูลมหาวิทยาลัย ที่ซึ่งประกอบไปด้วย ตาราง ข้อมูลนักเรียน (STUDENT) ที่ประกอบด้วย ชื่อนักศึกษา รหัสนักศึกษา ชั้นปี และสาขาวิชาหลัก ตารางข้อมูลวิชาเรียน (COURSE) ที่ประกอบด้วย รหัสวิชา ชื่อวิชา จำนวนชั่วโมง และคณะที่รับผิดชอบ ตารางข้อมูลวิชาเรียนที่ต้องผ่านก่อน (PREREQUISITE) ประกอบด้วย รหัสวิชา และรหัสวิชาที่ต้องเรียนก่อน ตารางข้อมูลตอนเรียน (SECTION) ที่ประกอบด้วย รหัสตอนเรียน รหัสวิชา ปีการศึกษา ปี และผู้สอน และตารางข้อมูลรายงานผลการเรียน (GRADE_REPORT) ที่ประกอบด้วย รหัสนักศึกษา รหัสตอนเรียน และเกรดที่ได้รับ โดยแต่ละแอตทริบิวต์จะระบุประเภทข้อมูลเช่นชื่อนักศึกษาที่จัดเก็บด้วยประเภทข้อมูลที่เป็นสายอักขระ (String) รหัสนักศึกษาที่จัดเก็บด้วยชนิดของข้อมูลที่เป็นตัวเลข (Integer) และชั้นปีที่จัดเก็บด้วยประเภทข้อมูลที่เป็นตัวอักษร (Char(1)) เป็นต้น

2.2. งานวิจัยที่เกี่ยวข้อง

วิทยานิพนธ์นี้ได้อ้างอิงถึงงานวิจัยอื่นที่เกี่ยวข้องเพื่อนำมาใช้ประโยชน์ในขั้นตอนการวิจัย โดยประกอบด้วย งานวิจัยดังนี้

2.2.1. งานวิจัย " A Tool for Test Case Impact Analysis of Database Schema Changes using Use Cases"

งานวิจัยโดย Jiratchaya Jainae และ Taratip Suwannasart (2014:186-189) [6] นำเสนอเครื่องมือสำหรับใช้วิเคราะห์ผลกระทบต่อกรณีทดสอบจากการเปลี่ยนแปลงสคีมาโดยใช้รายละเอียดยูสเคสเป็นเครื่องมือในการสร้างกรณีทดสอบใหม่ขึ้นมาเพื่อแทนกรณีทดสอบเดิม โดยงานวิจัยนี้มีข้อมูลนำเข้าคือชุดคำสั่งเอสคิวแอล จากนั้นวิเคราะห์หาความเปลี่ยนแปลงที่เกิดขึ้นจากสคีมาฐานข้อมูล แล้วนำแอตทริบิวต์ที่ได้จากสคีมาฐานข้อมูลมาตรวจสอบความต้องกันหากการค้นหาและเปรียบเทียบรายละเอียดของยูสเคสเดิมได้รับผลกระทบจากข้อมูลใหม่ในสคีมาฐานข้อมูลที่ได้รับการเปลี่ยนแปลง โดยการเปรียบเทียบจากลักษณะคำสั่งดังต่อไปนี้ ลักษณะของการลบ (DROP) ลักษณะของการเพิ่ม (ADD) และลักษณะของการแก้ไข (CHANGE) เครื่องมือจะวิเคราะห์กรณีทดสอบเป็นลำดับต่อไป ซึ่งในการวิเคราะห์ผลกระทบที่มีต่อกรณีทดสอบนั้นจะเปรียบเทียบข้อมูลทดสอบของกรณีทดสอบกับค่าเงื่อนไขที่ได้จากการเปรียบเทียบรายละเอียดยูสเคส โดยเครื่องมือจะนำเสนอผลลัพธ์ของกรณีทดสอบที่ไม่ได้รับผลกระทบ และผลลัพธ์ของกรณีทดสอบที่ได้รับผลกระทบ จากนั้นจึงสร้างกรณีทดสอบขึ้นมาใหม่

จากงานวิจัยนี้ผู้วิจัยได้นำแนวคิดและลักษณะการเปลี่ยนแปลงที่ส่งผลกระทบต่อฐานข้อมูลมาประยุกต์ใช้เพื่อนำไปวิเคราะห์ผลกระทบต่อกรณีทดสอบของเครื่องมือได้

2.2.2. งานวิจัย "Impact Analysis to Database Schema and Test Cases from Inputs of Functional Requirement Changes"

งานวิจัยโดย Apirak Kampeera และ Taratip Suwannasart (2016:449-453) [7] นำเสนอวิธีการวิเคราะห์หาผลกระทบต่อสคีมาฐานข้อมูลและกรณีทดสอบจากการเปลี่ยนแปลงของข้อมูลนำเข้าของความต้องการเชิงฟังก์ชัน โดยนำเข้าเอกสารความต้องการเชิงฟังก์ชันสองเวอร์ชัน จากนั้นวิเคราะห์หาผลกระทบต่อสคีมาฐานข้อมูลจากการเปลี่ยนแปลงของข้อมูลนำเข้า แล้วสร้างชุดคำสั่งเอสคิวแอล เพื่อปรับปรุงสคีมาฐานข้อมูลที่ได้รับผลกระทบ จากนั้นวิเคราะห์หาผลกระทบของกรณีทดสอบที่สัมพันธ์กับความต้องการเชิงฟังก์ชัน โดยใช้ตารางการตามรอยความต้องการ ซึ่งเป็นตารางที่แสดงความสัมพันธ์ระหว่างความต้องการและกรณีทดสอบ ดังนั้นถ้าหากข้อมูลนำเข้า

ของความต้องการเชิงฟังก์ชัน เปลี่ยนกรณีทดสอบจะถูกปรับปรุงแก้ไขตามด้วย ผลลัพธ์ที่ได้จากงานวิจัยนี้คือ ผลกระทบที่เกิดจากการเปลี่ยนแปลงข้อมูลนำเข้าของความต้องการเชิงฟังก์ชัน

จากงานวิจัยนี้ ผู้วิจัยได้นำกระบวนการวิเคราะห์และการปรับปรุงกรณีทดสอบมาประยุกต์ใช้และใช้เป็นแนวทางในการพัฒนาเครื่องมือเพื่อสร้างและปรับปรุงกรณีทดสอบได้

2.2.3. งานวิจัย "A Tool for Impact Analysis of Test Cases Based on Changed of A Web Application"

งานวิจัยโดย Surasak Phetmanee และ Taratip Suwannasart (2014:497-500) [8] นำเสนอเครื่องมือการวิเคราะห์ผลกระทบของกรณีทดสอบจากการเปลี่ยนแปลงเว็บแอปพลิเคชัน 2 เวอร์ชัน โดยเริ่มเปรียบเทียบจากไฟล์เอชทีเอ็มแอล (HTML) และไฟล์เอกซ์เอ็มแอลเวอร์ชันเดิมและเวอร์ชันที่ได้รับการแก้ไข โดยเปรียบเทียบในขอบเขตของการเปลี่ยนแปลง ชื่อตัวแปร ชนิดของข้อมูล ค่าตัวแปร ค่าแท็กของตัวแปร ลำดับของตัวแปร การเชื่อมโยง และจำนวนของตัวแปร จากนั้นเครื่องมือจะวิเคราะห์ผลกระทบและปรับปรุงกรณีทดสอบรวมไปถึงสร้างกรณีทดสอบใหม่ให้ครอบคลุมการทดสอบขั้นสมมูล

จากงานวิจัยนี้ผู้วิจัยสามารถนำความรู้เรื่องการวิเคราะห์ผลกระทบของกรณีทดสอบมาประยุกต์ใช้และใช้เป็นแนวทางในการพัฒนาเครื่องมือได้

2.2.4. งานวิจัย "Identify Impact of Database Schema on Application"

งานวิจัยโดย A. Karahasanovic (2001:93-104) [9] นำเสนอเครื่องมือเพื่อระบุผลกระทบที่เกิดจากการเปลี่ยนแปลงสคีมาฐานข้อมูลบนแอปพลิเคชันเชิงวัตถุด้วยเครื่องมือ SEMT (Schema Evolution Management Tools) เพื่อลดเวลาในการจัดการการเปลี่ยนแปลงฐานข้อมูลและลดข้อผิดพลาดที่เกิดขึ้น โดยเงื่อนไขการเปลี่ยนแปลงของงานวิจัยนี้คือการ เพิ่ม ลบ แก้ไขฟิลด์ และเมท็อด ในสคีมาฐานข้อมูล ผลการวิเคราะห์ของงานวิจัยนี้ได้นำเสนอให้เห็นผลกระทบที่เกิดในระดับฟิลด์ เมทอด และคลาส โดยแสดงผลในรูปแบบของกราฟเท่านั้น แต่ไม่ได้แสดงให้ผู้อ่านได้ทราบถึงส่วนที่ได้รับผลกระทบใดๆ ในแอปพลิเคชัน

จากงานวิจัยนี้ผู้วิจัยสามารถนำความรู้เรื่องการเปลี่ยนแปลงสคีมาฐานข้อมูลบนแอปพลิเคชันเชิงวัตถุมาประยุกต์ใช้และใช้เป็นแนวทางในการพัฒนาเครื่องมือได้

2.2.5. งานวิจัย "A Two-folded Impact Analysis of Schema Change on Database Applications"

งานวิจัยโดย S.K. Gardikiotis และ N. Malevris (2009:109-123) [10] นำเสนอการวิเคราะห์ผลกระทบที่เกิดจากการเปลี่ยนแปลงสคีมาฐานข้อมูล โดยแบ่งเป็น 2 ส่วนคือส่วนที่กระทบซอร์สโค้ด และส่วนที่กระทบกรณีทดสอบ ผู้วิจัยได้สร้างเครื่องมือชื่อ DATA ขึ้นมาเพื่อบอกผลกระทบโดยวิเคราะห์ผลกระทบต่อซอร์สโค้ดโดยใช้คอนโทรลโฟลว์กราฟ (Control Flow Graph) และแสดงผลกระทบต่อกรณีทดสอบในรูปแบบกราฟ ซึ่งผู้วิจัยได้กำหนดเงื่อนไขการเปลี่ยนแปลงของงานวิจัยนี้ไว้เพียง 2 รูปแบบคือการเพิ่ม และลบฟิลด์ในตารางฐานข้อมูลเท่านั้น และผลลัพธ์ของเครื่องมือนี้มีเพียงการแสดงผลกระทบในรูปแบบกราฟเท่านั้น ไม่ได้ปรับปรุงหรือสร้างกรณีทดสอบใหม่ขึ้นมา

จากงานวิจัยนี้ผู้วิจัยสามารถนำความรู้เรื่องการวิเคราะห์ผลกระทบต่อซอร์สโค้ดและกรณีทดสอบมาประยุกต์ใช้และใช้เป็นแนวทางในการพัฒนาเครื่องมือได้

2.2.6. งานวิจัย "เครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล"

งานวิจัยโดย Chanwit Sriarpanon และ Taratip Suwannasart (2014) [1] นำเสนอเครื่องมือวิเคราะห์ผลกระทบต่อกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล โดยมุ่งเน้นการวิเคราะห์ผ่านกลไกการทำงานภายในซอฟต์แวร์ ด้วยข้อมูลนำเข้า 4 ไฟล์ ได้แก่ ไฟล์ล็อก ไฟล์เอกสารสคีมาฐานข้อมูลก่อนการเปลี่ยนแปลง ซอร์สโค้ดแบบฝังตัว และกรณีทดสอบ เครื่องมือจะวิเคราะห์การเปลี่ยนแปลงที่เกิดขึ้นกับสคีมาฐานข้อมูลผ่านไฟล์ล็อกดังแสดงในรูปที่ 2-5 โดยไฟล์ล็อกนี้อยู่ในรูปแบบของไฟล์เอกสาร (Text File) ซึ่งประกอบไปด้วย ชื่อตารางที่เกิดการเปลี่ยนแปลง ชื่อฟิลด์ที่เกิดการเปลี่ยนแปลง และประเภทของการเปลี่ยนแปลง เพื่อนำมาใช้เปรียบเทียบความตรงกันของข้อมูลระหว่างไฟล์ล็อกกับไฟล์เอกสารสคีมาฐานข้อมูลก่อนเกิดการเปลี่ยนแปลงที่อยู่ในรูปแบบเอกสารเอสคิวแอลสคริปต์ ดังแสดงในรูปที่ 2-6 หลังจากนั้นจะวิเคราะห์กลไกการทำงานภายในของซอฟต์แวร์เพื่อตรวจสอบหาผลกระทบต่อซอร์สโค้ด และกรณีทดสอบเป็นลำดับต่อไป โดยการวิเคราะห์หาผลกระทบต่อซอร์สโค้ดจะหา Statement และตัวแปรที่ได้รับผลกระทบ ซึ่งผู้ใช้ต้องนำเข้าไฟล์แมปปิ้ง (Mapping File) ที่สร้างมาจากซอร์สโค้ดดังแสดงในตารางที่ 2-2 เพื่อบอกถึงรายละเอียดความสัมพันธ์ระหว่างซอร์สโค้ดกับหมายเลขบรรทัดของซอร์สโค้ด ซึ่งอยู่ในรูปแบบของไฟล์ซีเอสวี (CSV File) โดยผลลัพธ์ที่ได้คือหมายเลขบรรทัดที่ได้รับผลกระทบจากการเปลี่ยนแปลงสคีมาฐานข้อมูลเมื่อได้หมายเลขบรรทัดที่ได้รับผลกระทบจะนำมาเปรียบเทียบกับทางเดินการไหลในกรณีทดสอบ เพื่อหากรณีทดสอบได้รับผลกระทบ

ดังแสดงในตารางที่ 2-3 ด้วยผลกระทบ 5 รูปแบบดังนี้ (1) ผลกระทบต่อกรณีทดสอบการณีแก้ไขชื่อฟิลด์ (2) ผลกระทบต่อกรณีทดสอบกรณีแก้ไขขนาดของฟิลด์แบบเพิ่มขนาด (3) ผลกระทบต่อกรณีทดสอบกรณีแก้ไขประเภทข้อมูลของฟิลด์ (4) ผลกระทบต่อกรณีทดสอบกรณีลบฟิลด์ และ (5) ผลกระทบต่อกรณีทดสอบกรณีเพิ่มฟิลด์ หากพบว่าหากเกิดผลกระทบต่อกรณีทดสอบ จะส่งผลให้กรณีทดสอบนั้นไม่สามารถใช้งานต่อได้ งานวิจัยนี้จึงพัฒนาเครื่องมือปรับปรุงกรณีทดสอบเพื่อให้กรณีทดสอบนั้นมีความสอดคล้องกับข้อมูลภายในสคีมาฐานข้อมูลที่เกิดการเปลี่ยนแปลงและสามารถนำไปใช้งานต่อไปได้

```
ALTER TABLE `member` CHANGE `password` `password` VARCHAR(5) NOT NULL
ALTER TABLE `member` CHANGE `firstname` `firstname` CHAR(50) NOT NULL
ALTER TABLE `member` CHANGE `cash_balance` `cash_bal` INT(11) NOT NULL
ALTER TABLE `member`  ADD `description` VARCHAR(50) NOT NULL
ALTER TABLE `member` DROP `lastname`
```

รูปที่ 2-5 ตัวอย่างไฟล์ล็อก

```
CREATE TABLE IF NOT EXISTS `member` (
`mid` varchar(10) NOT NULL,
`password` varchar(10) NOT NULL,
`firstname` varchar(50) NOT NULL,
`lastname` varchar(50) NOT NULL,
`address` varchar(100) NOT NULL,
`email` varchar(100) NOT NULL,
`cash_balance` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

รูปที่ 2-6 ตัวอย่างเอกสารสคีมาฐานข้อมูลก่อนการเปลี่ยนแปลง

ตารางที่ 2-2 ตัวอย่างซอร์สโค้ดแบบฝังตัวใรภาษาจาวา

Line of code number	Statement
34	private static void memberLogIn() throws SQLException, IOException
35	{
36	String mmid1, pass1, fname1, lname1;
37	String lname2 = readEntry("Account#: ");
38	String pass2 = readEntry("Password: ");
39	Try
40	{
41	#sql {select mid, password, firstname, lastname into :mmid1, :pass1, :fname1, lname1 from member where lastname = :lname2 and password = :pass2};
42	}

ตารางที่ 2-3 ตัวอย่างกรณีทดสอบที่ได้รับผลกระทบจากการเปลี่ยนแปลงสคีมาฐานข้อมูล

Test case ID	TC1		
Path	34-35-36-37-38-39-40-41-42		
Input			
Name	Type	Size	Value
lastname	STRING	10	Sriarpanon
password	STRING	10	aaaaaaaaaa
Expected output	Member login success		

บทที่ 3

การวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ

ในบทนี้จะอธิบายถึงวิธีการวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล เพื่อที่จะนำไปประยุกต์ใช้ในการออกแบบและพัฒนาเครื่องมือสนับสนุนต่อไป โดยมีรายละเอียดดังนี้

3.1. ภาพรวมการทำงานของเครื่องมือ

รูปที่ 3-1 แสดงถึงภาพรวมการทำงานของเครื่องมือที่งานวิจัยนี้ได้นำเสนอ ซึ่งประกอบไปด้วย 5 ขั้นตอนหลัก ดังนี้

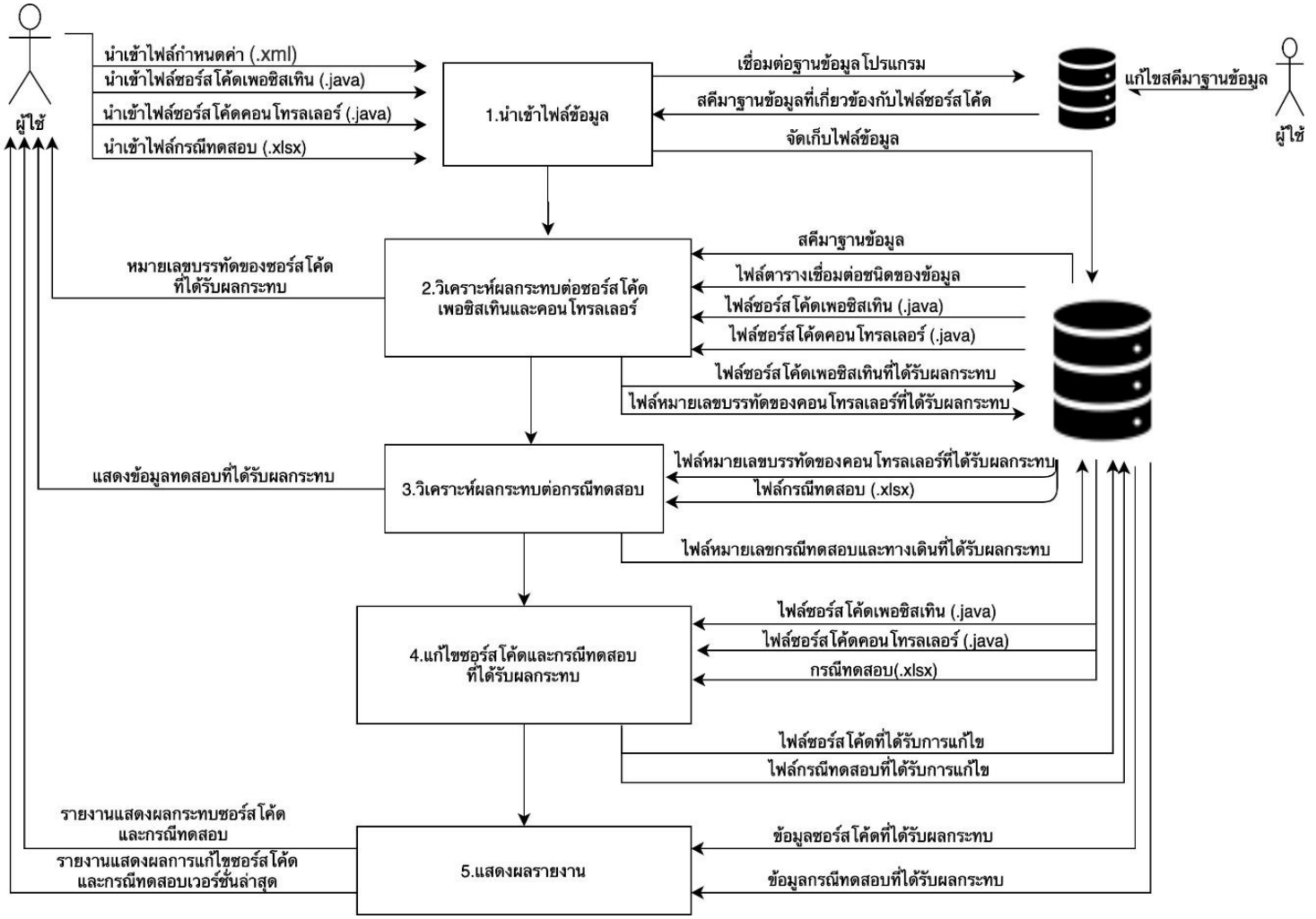
(1) นำเข้าไฟล์ข้อมูลตั้งต้นที่ต้องการตรวจสอบ 4 ไฟล์ ได้แก่ ไฟล์กำหนดค่าไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบ หลังจากนั้นเครื่องมือจะเชื่อมต่อฐานข้อมูลของโปรแกรมที่จะนำมาทดสอบผ่านไฟล์กำหนดค่าและสกัดสคีมาฐานข้อมูลที่เกี่ยวข้องกับไฟล์เพอซิสเทินจากฐานข้อมูลโปรแกรมมาจัดเก็บในฐานข้อมูลของเครื่องมือ

(2) วิเคราะห์หาผลกระทบต่อซอร์โค้ดเพอซิสเทินและซอร์โค้ดคอนโทรลเลอร์ เพื่อเปรียบเทียบความตรงกันของข้อมูล โดยใช้ไฟล์ตารางเชื่อมต่อชนิดของข้อมูลเข้ามาช่วยตรวจสอบความตรงกันในสคีมาฐานข้อมูลและไฟล์ซอร์สโค้ดเพอซิสเทิน หากข้อมูลไม่ตรงกันหมายความว่าเกิดการเปลี่ยนแปลงกับสคีมาฐานข้อมูล ในกรณีที่มีผลกระทบเกิดขึ้น เครื่องมือจะจัดเก็บหมายเลขบรรทัดของซอร์สโค้ดคอนโทรลเลอร์ที่ได้รับผลกระทบไว้ในฐานข้อมูลของเครื่องมือ

(3) วิเคราะห์หาผลกระทบต่อกรณีทดสอบ ขั้นตอนนี้เครื่องมือจะตรวจสอบความตรงกันของไฟล์หมายเลขบรรทัดที่ได้รับผลกระทบกับเส้นทางเดินการทดสอบของไฟล์กรณีทดสอบ หากตรงกันหมายถึงกรณีทดสอบนั้นได้รับผลกระทบจากการเปลี่ยนแปลงสคีมา เครื่องมือจะจัดเก็บหมายเลขกรณีทดสอบและทางเดินที่ได้รับผลกระทบเอาไว้ เพื่อนำไปปรับปรุงแก้ไขในขั้นตอนต่อไป

(4) การแก้ไขซอร์สโค้ดและกรณีทดสอบ เครื่องมือจะนำไฟล์ที่ได้รับผลกระทบทั้งหมดมาตรวจสอบประเภทของการเปลี่ยนแปลง เพื่อที่จะแก้ไขได้ถูกต้อง โดยเครื่องมือจะปรับปรุงแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทินก่อนเป็นอันดับแรก จากนั้นจะแก้ไขไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และแก้ไขไฟล์กรณีทดสอบตามลำดับ และจัดเก็บไฟล์ลงไปในฐานข้อมูลของเครื่องมือเพื่อให้ผู้ใช้งานสามารถนำออกไฟล์ไปใช้งานต่อได้

(5) แสดงข้อมูลโดยสรุปรายละเอียดการเปลี่ยนแปลงและการแก้ไขทั้งหมดให้ผู้ใช้ทราบในกระบวนการแสดงผลรายงานดังรายละเอียดต่อไปนี้



รูปที่ 3-1 แผนภาพการทำงานของเครื่องมือ

3.1.1. การนำเข้าไฟล์ข้อมูล

ผู้ใช้นำเข้าไฟล์ข้อมูลไฟล์ข้อมูลตั้งต้นที่ต้องการตรวจสอบ 4 ไฟล์ได้แก่ ไฟล์กำหนดค่าไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และนำเข้าไฟล์กรณีทดสอบ เพื่อนำไปวิเคราะห์หาผลกระทบที่เกิดขึ้นต่อซอร์สโค้ดและกรณีทดสอบ โดยไฟล์กรณีทดสอบที่นำเข้าสู่เครื่องมือจะต้องมีความสอดคล้องกับไฟล์ซอร์สโค้ดคอนโทรลเลอร์ในรูปแบบของการทดสอบทางเดินการไหลของข้อมูล โดยไฟล์กรณีทดสอบจะประกอบไปด้วย หมายเลขกรณีทดสอบ เส้นทางเดิน ชื่อฟิลด์ ชนิดของข้อมูล ขนาดของข้อมูล ค่าของข้อมูล และผลลัพธ์ของข้อมูล เมื่อนำเข้าไฟล์ข้อมูลเสร็จแล้ว เครื่องมือจะเชื่อมต่อฐานข้อมูลโปรแกรมโดยอัตโนมัติผ่านไฟล์กำหนดค่า ซึ่งภายในไฟล์กำหนดค่าจะประกอบไปด้วยตำแหน่งที่อยู่ของเว็บ (URL) ชื่อผู้ใช้งาน (Username) และรหัสผ่าน (Password) จากนั้นตรวจสอบข้อมูลชื่อตาราง หากพบชื่อตารางที่ตรงกัน เครื่องมือจะสกัดสคีมาฐานข้อมูลจากตารางที่เกี่ยวข้อง เพื่อมาจัดเก็บในฐานข้อมูลของเครื่องมือ

3.1.2. การวิเคราะห์ผลกระทบต่อซอร์สโค้ดเพอซิสเทินและซอร์สโค้ดคอนโทรลเลอร์

ขั้นตอนนี้เป็นขั้นตอนการวิเคราะห์ผลกระทบต่อซอร์สโค้ดเพอซิสเทินและซอร์สโค้ดคอนโทรลเลอร์ โดยเครื่องมือจะวิเคราะห์หาผลกระทบต่อซอร์สโค้ดเพอซิสเทินก่อนเป็นอันดับแรก โดยสกัดข้อมูลตาราง แอตทริบิวต์ ประเภทข้อมูลของแอตทริบิวต์ และค่าคีย์ของแอตทริบิวต์จากไฟล์ซอร์สโค้ดเพอซิสเทินและสคีมาฐานข้อมูลก่อน จากนั้นจะตรวจสอบและเปรียบเทียบความตรงกันของข้อมูล โดยเครื่องมือจะวิเคราะห์ความเปลี่ยนแปลงที่เกิดขึ้นในสคีมาฐานข้อมูล 7 รูปแบบดังนี้

1. ชื่อตารางมีการเปลี่ยนแปลง
2. ชื่อแอตทริบิวต์มีการเปลี่ยนแปลง
3. ประเภทข้อมูลของแอตทริบิวต์มีการเปลี่ยนแปลง
4. ค่าขนาดข้อมูลของแอตทริบิวต์มีการเปลี่ยนแปลง
5. คีย์ของแอตทริบิวต์มีการเปลี่ยนแปลง
6. การลบตาราง
7. การลบแอตทริบิวต์

เมื่อผลการวิเคราะห์ความตรงกันของข้อมูลแสดงให้เห็นว่าข้อมูลไม่ตรงกัน ตัวอย่างเช่น ประเภทข้อมูลของแอตทริบิวต์ในสคีมาฐานข้อมูล และประเภทข้อมูลของแอตทริบิวต์ที่อยู่ในซอร์สโค้ดเพอซิสเทินไม่ตรงกัน ดังแสดงในรูปที่ 3-2 และรูปที่ 3-3 หมายความว่าเกิดการเปลี่ยนแปลงเกิดขึ้นในสคีมาฐานข้อมูลโดยเปลี่ยนประเภทข้อมูลจาก int ไปเป็นประเภทข้อมูล float เครื่องมือจะจัดเก็บชื่อแอตทริบิวต์ ประเภทข้อมูลของแอตทริบิวต์ และชื่อฟังก์ชันที่ได้รับผลกระทบของไฟล์

ซอร์สโค้ดเพอซิสเทินไว้ในฐานข้อมูลของเครื่องมือ จากนั้นจะนำชื่อฟังก์ชันที่ได้รับผลกระทบของไฟล์ซอร์สโค้ดเพอซิสเทิน ซึ่งได้แก่ ชื่อฟังก์ชัน getPrice() และชื่อฟังก์ชัน setPrice() ไปตรวจสอบหาความตรงกันของชื่อฟังก์ชันในซอร์สโค้ดคอนโทรลเลอร์ หากผลการวิเคราะห์พบว่ามีชื่อฟังก์ชันที่ตรงกันในไฟล์ซอร์สโค้ดคอนโทรลเลอร์ ดังแสดงในรูปที่ 3-4 เครื่องมือจะจัดเก็บหมายเลขบรรทัดของซอร์สโค้ดคอนโทรลเลอร์ไว้ในฐานข้อมูลของเครื่องมือเพื่อใช้ในขั้นตอนการ วิเคราะห์ผลกระทบต่อกรณีทดสอบเป็นลำดับต่อไป ในส่วนของการแสดงผลลัพธ์จากการวิเคราะห์หาผลกระทบต่อซอร์สโค้ดเพอซิสเทินและคอนโทรลเลอร์ เครื่องมือจะแสดงโดยการไฮไลท์ด้วยสีเพื่อบอกถึงตำแหน่งที่เกิดผลกระทบในซอร์สโค้ดผ่านทางหน้าจอให้ผู้ใช้ได้รับทราบ

```
31 CREATE TABLE `note` (
32   `NOTEid` bigint(20) NOT NULL,
33   `content` varchar(255) DEFAULT NULL,
34   `created_at` datetime NOT NULL,
35   `title` varchar(200) DEFAULT NULL,
36   `updated_at` datetime NOT NULL,
37   `price` float NOT NULL,
38   `post_id` bigint(20) DEFAULT NULL,
39   `enable` bit(1) NOT NULL,
40   `remark` char(1) NOT NULL,
41   `test1` smallint(6) NOT NULL,
42   `test2` double NOT NULL
43 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

รูปที่ 3-2 ตัวอย่างประเภทข้อมูลของแอตทริบิวต์ในสคีมาฐานข้อมูลที่มีการเปลี่ยนแปลง

```
34⊖    @Column(name = "PRICE")
35     @NotBlank
36     private int price;
37
39⊕    private boolean enable;
40
42⊕    private char remark;
43
45⊕    private short test1;
46
48⊕    private double test2;
49
53⊕    private Date createdAt;
54
58⊕    private Date updatedAt;
59
67⊕    public Long getId() {
70
71⊕    public void setId(Long id) {
74
75⊕    public Long getPostId() {
78
79⊕    public void setPostId(Long post_id) {
82
83⊖    public int getPrice() {
84         return price;
85     }
86
87⊖    public void setPrice(int price) {
88         this.price = price;
89     }
```

รูปที่ 3-3 ตัวอย่างประเภทข้อมูลของแอตทริบิวต์ในซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบ

```
81⊖    @GetMapping("/notesCalPrice/{id}")
82     public int CalPrice(@PathVariable(value = "id") Long noteId)
83     {
84         System.out.println("getNoteString");
85         Optional<Note> noteList2 = noteRepository.findById(noteId);
86         int nPrice = noteList2.get().getPrice();
87         if(nPrice > 1001 && nPrice < 2000)
88         {
89             noteList2.get().setPrice(nPrice-100);
90         }else if(nPrice > 2000)
91         {
92             noteList2.get().setPrice(nPrice-200);
93         }
94         return noteList2.get().getPrice();
95     }
```

รูปที่ 3-4 ตัวอย่างไฟล์ซอร์สโค้ดคอนโทรลเลอร์ที่ได้รับผลกระทบ

ตารางที่ 3-1 แสดงรายการของการเปลี่ยนแปลงทั้งหมดในเครื่องมือที่ส่งผลกระทบต่อซอร์สโค้ดเพอซิสเทินและซอร์สโค้ดคอนโทรลเลอร์ ตัวอย่างเช่น หากมีการเปลี่ยนแปลงประเภทข้อมูลของแอตทริบิวต์เกิดขึ้นในสคีมาฐานข้อมูลจาก int ไปเป็น float ดังรูปที่ 3-2 จะทำให้ส่งผลกระทบต่อไฟล์ซอร์สโค้ดเพอซิสเทินและไฟล์ซอร์สโค้ดคอนโทรลเลอร์ได้ เนื่องจากประเภทข้อมูลของแอตทริบิวต์นั้นๆ จะถูกกำหนดไว้ในไฟล์ซอร์สโค้ดเพอซิสเทินและไฟล์ซอร์สโค้ดคอนโทรลเลอร์ ดังแสดงในรูปที่ 3-3 (หมายเลขบรรทัดที่ 36 83 และ 87) และรูปที่ 3-4 (หมายเลขบรรทัดที่ 86)

ตารางที่ 3-1 ผลกระทบของซอร์สโค้ดต่อการเปลี่ยนแปลงที่เกิดขึ้น

การเปลี่ยนแปลง	กระทบ/ไม่กระทบ ซอร์สโค้ด เพอซิสเทิน	กระทบ/ไม่กระทบ ซอร์สโค้ด คอนโทรลเลอร์
ชื่อตารางมีการเปลี่ยนแปลง	กระทบ	ไม่กระทบ
ชื่อแอตทริบิวต์มีการเปลี่ยนแปลง	กระทบ	ไม่กระทบ
ประเภทข้อมูลของแอตทริบิวต์มีการเปลี่ยนแปลง	กระทบ	กระทบ
ค่าขนาดข้อมูลของแอตทริบิวต์มีการเปลี่ยนแปลง	กระทบ	ไม่กระทบ
คีย์ของแอตทริบิวต์มีการเปลี่ยนแปลง	กระทบ	ไม่กระทบ
การลบตาราง	กระทบ	กระทบ
การลบแอตทริบิวต์	กระทบ	กระทบ

3.1.3. การวิเคราะห์ผลกระทบต่อกรณีทดสอบ

ในขั้นตอนนี้เครื่องมือจะวิเคราะห์ผลกระทบการเปลี่ยนแปลงสคีมาฐานข้อมูลกับกรณีทดสอบ โดยเปรียบเทียบหมายเลขบรรทัดของซอร์สโค้ดคอนโทรลเลอร์ที่ได้รับผลกระทบและประเภทข้อมูลของแอตทริบิวต์กับไฟล์กรณีทดสอบว่ามีความตรงกันหรือไม่ ซึ่งจากรูปที่ 3-4 แสดงถึงซอร์สโค้ดคอนโทรลเลอร์ที่พบว่ามีหมายเลขบรรทัดที่ 86 89 และ 92 ได้รับผลกระทบจากการเปลี่ยนแปลงสคีมาฐานข้อมูล และหมายเลขบรรทัดของซอร์สโค้ดคอนโทรลเลอร์ที่ได้รับผลกระทบนี้ตรงกันกับหมายเลขทางเดินการไหลของข้อมูลในไฟล์กรณีทดสอบที่แสดงในตารางที่ 3-2 จึงสามารถสรุปได้ว่ากรณีทดสอบนั้นได้รับผลกระทบ หลังจากนั้นเครื่องมือจะจัดเก็บหมายเลขกรณีทดสอบ ประเภทข้อมูลของแอตทริบิวต์และทางเดินการไหลของข้อมูลที่ได้รับผลกระทบไว้ในฐานข้อมูลของเครื่องมือ และจะแสดงหมายเลขตำแหน่งที่เกิดผลกระทบให้ผู้ใช้ได้รับทราบผ่านทางหน้าจอ

ตารางที่ 3-3 แสดงรายการของการเปลี่ยนแปลงทั้งหมดในเครื่องมือที่ส่งผลกระทบต่อกรณีทดสอบ ตัวอย่างเช่น การเปลี่ยนแปลงประเภทของข้อมูลแอตทริบิวต์ในสคีมาฐานข้อมูล จะทำให้เกิดผลกระทบต่อกรณีทดสอบเนื่องจากหมายเลขทางเดินการไหลของข้อมูลในกรณีทดสอบมีความสอดคล้องกับหมายเลขบรรทัดในไฟล์ซอร์สโค้ดคอนโทรลเลอร์ หากหมายเลขบรรทัดที่ปรากฏในหมายเลขทางเดินการไหลของข้อมูลได้รับผลกระทบ แสดงว่าหมายเลขกรณีทดสอบนั้นย่อมได้รับผลกระทบ

ตารางที่ 3-2 ตัวอย่างไฟล์กรณีทดสอบที่ได้รับผลกระทบจากการเปลี่ยนแปลงสคีมาฐานข้อมูล

Test Case ID	TC1		
Path	81-82-83-84-85-86-87-88-89-90-94-95		
Input			
Name	**Type**	**Size**	**Value**
Price	int	6	1,500
Expected Output	1,400		

ตารางที่ 3-3 ผลกระทบของกรณีทดสอบต่อการเปลี่ยนแปลงที่เกิดขึ้น

การเปลี่ยนแปลง	กระทบ/ไม่กระทบ กรณีทดสอบ
ชื่อตารางมีการเปลี่ยนแปลง	ไม่กระทบ
ชื่อแอตทริบิวต์มีการเปลี่ยนแปลง	ไม่กระทบ
ประเภทข้อมูลของแอตทริบิวต์มีการเปลี่ยนแปลง	กระทบ
ค่าขนาดข้อมูลของแอตทริบิวต์มีการเปลี่ยนแปลง	ไม่กระทบ
คีย์ของแอตทริบิวต์มีการเปลี่ยนแปลง	ไม่กระทบ
การลบตาราง	กระทบ
การลบแอตทริบิวต์	กระทบ

3.1.4. การปรับปรุงแก้ไขซอร์สโค้ดและกรณีทดสอบ

ในขั้นตอนการปรับปรุงแก้ไขซอร์สโค้ดและกรณีทดสอบนี้ การปรับปรุงแก้ไขจะขึ้นอยู่กับกรณีของการเปลี่ยนแปลงที่เกิดขึ้นดังตารางที่ 3-4 ซึ่งตารางนี้จะแสดงถึงรายการของการแก้ไขซอร์สโค้ดเพอซิสเทิน การแก้ไขซอร์สโค้ดคอนโทรลเลอร์ และการแก้ไขกรณีทดสอบสำหรับการเปลี่ยนแปลงที่เกิดขึ้นในสคีมาฐานข้อมูลทั้งหมดที่โปรแกรมสามารถรองรับได้ โดยที่เครื่องมือจะตรวจสอบว่าเป็นรายการแก้ไขประเภทใดก่อนเป็นอันดับแรก จากนั้นจึงทำการระบุว่าจะแก้ไขหรือไม่แก้ไข ตัวอย่างเช่น เกิดการแก้ไขประเภทข้อมูลของแอตทริบิวต์ในสคีมาฐานข้อมูล เครื่องมือจะทำการแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทิน แก้ไขไฟล์ซอร์สโค้ดเพอซิสเทิน และแก้ไขกรณีทดสอบเนื่องจากได้รับผลกระทบจากกระบวนการก่อนหน้า

ตารางที่ 3-4 การแก้ไขซอร์สโค้ดและกรณีทดสอบต่อการเปลี่ยนแปลงที่เกิดขึ้น

การเปลี่ยนแปลง	แก้ไข/ไม่แก้ไข เพอซิสเทินคลาส	แก้ไข/ไม่แก้ไข คอนโทรลเลอร์คลาส	แก้ไข/ไม่แก้ไข กรณีทดสอบ
ชื่อตารางมีการเปลี่ยนแปลง	ไม่แก้ไข	ไม่แก้ไข	ไม่แก้ไข
ชื่อแอตทริบิวต์มีการเปลี่ยนแปลง	แก้ไข	ไม่แก้ไข	ไม่แก้ไข
ประเภทข้อมูลของแอตทริบิวต์มีการเปลี่ยนแปลง	แก้ไข	แก้ไข	แก้ไข
ค่าขนาดข้อมูลของแอตทริบิวต์มีการเปลี่ยนแปลง	แก้ไข	ไม่แก้ไข	ไม่แก้ไข
คีย์ของแอตทริบิวต์มีการเปลี่ยนแปลง	แก้ไข	ไม่แก้ไข	ไม่แก้ไข
การลบตาราง	ไม่แก้ไข	ไม่แก้ไข	ไม่แก้ไข
การลบแอตทริบิวต์	แก้ไข	แก้ไข	แก้ไข

จากข้อมูลในตารางที่ 3-4 การแก้ไขหรือไม่แก้ไขซอร์สโค้ดและกรณีทดสอบ จะขึ้นอยู่กับผลกระทบที่เกิดขึ้นจากการเปลี่ยนแปลงข้อมูลในสคีมาฐานข้อมูล ตัวอย่างเพิ่มเติมดังนี้

(1) หากมีการเปลี่ยนแปลงชื่อตารางในสคีมาฐานข้อมูล เครื่องมือจะไม่แก้ไขไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์โค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบ เนื่องจากเครื่องมือไม่สามารถเข้าถึงข้อมูลตารางจากโปรแกรมทดสอบได้ตั้งแต่ขั้นตอนการนำไฟล์เข้าไฟล์ข้อมูลตั้งต้น ดังนั้นเครื่องมือจึงไม่ได้ทำการวิเคราะห์หาผลกระทบที่เกิดขึ้นและส่งผลให้ไม่สามารทำการปรับปรุงแก้ไขได้

(2) หากมีการเปลี่ยนแปลงชื่อของแอตทริบิวต์ในสคีมาฐานข้อมูล เครื่องมือจะแก้ไขเฉพาะซอร์สโค้ดเพอซิสเทินเท่านั้น เพราะไฟล์ซอร์สโค้ดเพอซิสเทินเป็นไฟล์ที่ทำหน้าที่เชื่อมต่อระหว่างแอตทริบิวต์ในสคีมาฐานข้อมูลกับแอตทริบิวต์ในซอร์สโค้ดจาวา ส่วนไฟล์ซอร์สโค้ดคอนโทรลเลอร์และกรณีทดสอบที่ไม่ได้รับผลกระทบ เครื่องมือจึงไม่ได้ปรับปรุงแก้ไข

(3) หากมีการเปลี่ยนแปลงประเภทข้อมูลของแอตทริบิวต์ในสคีมาฐานข้อมูล เครื่องมือจะแก้ไขทั้ง 3 ไฟล์ คือไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และกรณีทดสอบที่ได้รับผลกระทบ เนื่องจากทั้ง 3 ไฟล์มีความสอดคล้องกันตามลำดับดังนี้ ไฟล์ซอร์สโค้ดสอดคล้องกับไฟล์ซอร์สโค้ดคอนโทรลเลอร์เนื้องจากที่การใช้งานแอตทริบิวต์ผ่านชื่อตัวแปร และชื่อฟังก์ชันไฟล์ซอร์สโค้ดคอนโทรลเลอร์สอดคล้องกับไฟล์กรณีทดสอบ เนื่องจากหมายเลขบรรทัดของซอร์สโค้ดในไฟล์คอนโทรลเลอร์จะปรากฏอยู่ในหมายเลขทางเดินการไหลของข้อมูลในกรณีทดสอบ

(4) หากมีการเปลี่ยนแปลงค่าขนาดข้อมูลของแอตทริบิวต์ในสคีมาฐานข้อมูล เครื่องมือจะแก้ไขเฉพาะซอร์สโค้ดเพอซิสเทินเท่านั้น เพราะไฟล์ซอร์สโค้ดเพอซิสเทินเป็นไฟล์ที่ทำหน้าที่เชื่อมต่อระหว่างค่าขนาดข้อมูลของแอตทริบิวต์ในสคีมาฐานข้อมูลกับค่าขนาดข้อมูลของแอตทริบิวต์ในซอร์สโค้ดจาวา ส่วนไฟล์ซอร์สโค้ดคอนโทรลเลอร์และกรณีทดสอบที่ไม่ได้รับผลกระทบ เครื่องมือจึงไม่ได้ปรับปรุงแก้ไข

(5) หากมีการเปลี่ยนแปลงคีย์ของแอตทริบิวต์ในสคีมาฐานข้อมูลจากฟิลด์หนึ่งไปเป็นอีกฟิลด์หนึ่ง เครื่องมือจะแก้ไขเฉพาะซอร์สโค้ดเพอซิสเทินเท่านั้น เพราะไฟล์ซอร์สโค้ดเพอซิสเทินเป็นไฟล์ที่ทำหน้าที่เชื่อมต่อระหว่างคีย์ข้อมูลของแอตทริบิวต์ในสคีมาฐานข้อมูลกับคีย์ข้อมูลของแอตทริบิวต์ในซอร์สโค้ดจาวา ส่วนไฟล์ซอร์สโค้ดคอนโทรลเลอร์และกรณีทดสอบที่ไม่ได้รับผลกระทบ เครื่องมือจึงไม่ได้ปรับปรุงแก้ไข

(6) หากมีการลบตารางในสคีมาฐานข้อมูล เครื่องมือจะไม่แก้ไขไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์โค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบ เนื่องจากเครื่องมือไม่สามารถเข้าถึงข้อมูลตารางจากโปรแกรมทดสอบได้ตั้งแต่ขั้นตอนการนำไฟล์เข้าไฟล์ข้อมูลตั้งต้น ดังนั้นเครื่องมือจึงไม่ได้ทำการวิเคราะห์หาผลกระทบที่เกิดขึ้นและส่งผลให้ไม่สามารทำการปรับปรุงแก้ไขได้

(7) หากมีการลบแอตทริบิวต์ในสคีมาฐานข้อมูล เครื่องมือจะแก้ไขทั้ง 3 ไฟล์ คือไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และกรณีทดสอบที่ได้รับผลกระทบ หากแอตทริบิวต์นั้นถูกใช้งานในทั้ง 3 ไฟล์ แต่หากแอตทริบิวต์ที่ถูกลบ ไม่ได้ถูกใช้งานในไฟล์ซอร์สโค้ดเพอซิสเทิน เครื่องมือจะไม่ทำการแก้ไขทั้ง 3 ไฟล์ เนื่องจากแอตทริบิวต์นั้นไม่ได้ถูกใช้งานในโปรแกรมทดสอบ

ในกรณีที่เครื่องมือได้รับคำสั่งจากผู้ใช้งานเพื่อให้ปรับปรุงแก้ไขไฟล์ข้อมูลที่ได้รับผลกระทบ เครื่องมือจะแก้ไขซอร์สโค้ดเพอซิสเทินและซอร์สโค้ดคอนโทรลเลอร์ก่อนเป็นอันดับแรก หลังจากนั้นจึงปรับปรุงแก้ไขกรณีทดสอบ โดยมีรายละเอียดดังต่อไปนี้

1) การปรับปรุงแก้ไขซอร์สโค้ดเพอซิสเทินและซอร์สโค้ดคอนโทรลเลอร์

เครื่องมือจะสกัดข้อมูลข้อมูลตาราง ชื่อแอตทริบิวต์ ประเภทข้อมูลของแอตทริบิวต์ ขนาดของแอตทริบิวต์ และค่าคีย์ของแอตทริบิวต์จากไฟล์ซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบและไฟล์หมายเลขบรรทัดของคอนโทรลเลอร์ เพื่อนำมาตรวจสอบประเภทของการเปลี่ยนแปลงกับไฟล์ซอร์สโค้ดเพอซิสเทินและไฟล์ซอร์สโค้ดคอลโทรลเลอร์ตั้งต้น โดยการเปลี่ยนแปลงในแต่ละประเภทจะทำการปรับปรุงแก้ไขที่แตกต่างกัน ตัวอย่างเช่น หากมีการเปลี่ยนแปลงประเภทข้อมูลของแอตทริบิวต์จาก int ไปเป็น float เครื่องมือจะแก้ไขประเภทข้อมูลของแอตทริบิวต์ในไฟล์ซอร์สโค้ดเพอซิสเทินจากรูปที่ 3-3 มาเป็นดังรูปที่ 3-5 และแก้ไขไฟล์ซอร์สโค้ดคอนโทรลเลอร์จากรูปที่ 3-4 มาเป็นดังรูปที่ 3-6 เมื่อแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทินและไฟล์ซอร์สโค้ดคอนโทรลเลอร์เสร็จ เครื่องมือจะจัดเก็บไฟล์ซอร์สโค้ดที่ได้รับการแก้ไขไว้ในฐานข้อมูลของเครื่องมือ

```
34⊖     @Column(name = "PRICE")
35      @NotBlank
36      private float price;
37⊖
38
39⊖     public float getPrice() {
40          return price;
41      }
42
43⊖     public void setPrice(float price) {
44          this.price = price;
45      }
46
```

รูปที่ 3-5 ตัวอย่างซอร์โค้ดเพอซิสเทินที่ได้รับการแก้ไข

```
81⊖     @GetMapping("/notesCalPrice/{id}")
82      public float CalPrice(@PathVariable(value = "id") Long noteId)
83      {
84          System.out.println("---getNoteString----");
85          Optional<Note> noteList2 =  noteRepository.findById(noteId);
86          float nPrice = noteList2.get().getPrice();
87          if(nPrice > 1001 && nPrice <2000)
88          {
89              noteList2.get().setPrice(nPrice - 100); //get discount 100
90          }else if (nPrice > 2000)
91          {
92              noteList2.get().setPrice(nPrice - 200); //get discount 200
93          }
94          return noteList2.get().getPrice();
95      }
```

รูปที่ 3-6 ตัวอย่างซอร์โค้ดคอนโทรลเลอร์ที่ได้รับการแก้ไข

2) การปรับปรุงแก้ไขกรณีทดสอบ

ในการปรับปรุงแก้ไขกรณีทดสอบ เครื่องมือจะนำเข้าไฟล์หมายเลขบรรทัดของคอนโทรลเลอร์ที่ได้รับผลกระทบ และไฟล์กรณีทดสอบตั้งต้น เพื่อตรวจสอบประเภทของการเปลี่ยนแปลง โดยจะตรวจสอบประเภทของการเปลี่ยนแปลง 4 ประเภท ได้แก่ การแก้ไขชื่อแอตทริบิวต์ การแก้ไขประเภทแอตทริบิวต์ การแก้ไขขนาดแอตทริบิวต์ และทางเดินของกรณีทดสอบตรงกับทางเดินได้รับผลกระทบ ซึ่งการเปลี่ยนแปลงในแต่ละประเภทจะมีการปรับปรุงแก้ไขที่แตกต่างกัน ตัวอย่างเช่น หากตรวจสอบพบว่าเป็นกรณีการแก้ไขประเภทแอตทริบิวต์ เครื่องมือจะแก้ไขจากประเภทข้อมูล int เป็นประเภทข้อมูล float จากนั้นจะแก้ไขค่าข้อมูลและค่าผลลัพธ์ที่คาดหวังตามประเภทของข้อมูลที่เปลี่ยนแปลง ดังแสดงในตารางที่ 3-5 เมื่อแก้ไขไฟล์กรณีทดสอบเสร็จจะจัดเก็บไฟล์กรณีทดสอบที่ได้รับการแก้ไขไว้ในฐานข้อมูลของเครื่องมือ

ตารางที่ 3-5 ตัวอย่างกรณีทดสอบที่ได้รับการแก้ไข

Test Case ID	TC1		
Path	81-82-83-84-85-86-87-88-89-90-94-95		
Input			
Name	**Type**	**Size**	**Value**
Price	float	6	1,500.50
Expected Output	1,400.50		

3.1.5. การแสดงผลรายงาน

ขั้นตอนนี้เป็นการแสดงผลรายงานสรุปผลให้กับผู้ใช้ได้รับทราบผ่านทางหน้าจอหลังจากการทำงานเสร็จสิ้นในแต่ละกระบวนการ ได้แก่

(1) หลังจากเสร็จสิ้นกระบวนการวิเคราะห์หาผลกระทบที่เกิดขึ้นต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ เครื่องมือจะแสดงผลกระทบต่อซอร์สโค้ดเพอซิสเทิน ผลกระทบต่อซอร์สโค้ดคอนโทรลเลอร์ และผลกระทบต่อกรณีทดสอบด้วยการไฮไลท์สีในตำแหน่งที่ได้รับผลกระทบ หากไฟล์ข้อมูลใดไม่ได้รับผลกระทบ เครื่องมือจะทำการแสดงข้อมูลแต่ไม่ปรากฏการไฮไลท์สี

(2) หลังจากเสร็จสิ้นกระบวนการปรับปรุงแก้ไขผลกระทบที่เกิดขึ้นต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ โดยเครื่องมือจะแสดงไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์และกรณีทดสอบ ด้วยการไฮไลท์สีในตำแหน่งที่ได้รับการปรับปรุงแก้ไขแก่ผู้ใช้ได้รับททราบผ่านทางหน้าจอ แต่หากไม่ได้รับการปรับปรุงแก้ไข เครื่องมือจะแสดงข้อความแจ้งให้ผู้ใช้ได้รับทราบว่าไม่มีไฟล์ข้อมูลที่ได้รับการปรับปรุงแก้ไข

บทที่ 4

การออกแบบและพัฒนาเครื่องมือ

ในส่วนนี้จะกล่าวถึงการพัฒนาเครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล ประกอบด้วยการออกแบบเครื่องมือ สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ และการพัฒนาส่วนต่อประสานกับผู้ใช้ของแต่ละหน้าจอการทำงาน ซึ่งรายละเอียดมีดังต่อไปนี้

4.1. การออกแบบของเครื่องเครื่องมือ

การออกแบบเครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูลสามารถแสดงได้ด้วยแผนภาพยูสเคส (Use Case Diagram) แผนภาพกิจกรรม (Activity Diagram) แผนภาพคลาส (Class Diagram) แผนภาพลำดับ (Sequence Diagram) และโครงสร้างฐานข้อมูล (Entity Relationship Diagram) ซึ่งมีรายละเอียดดังต่อไปนี้

4.1.1. แผนภาพยูสเคส

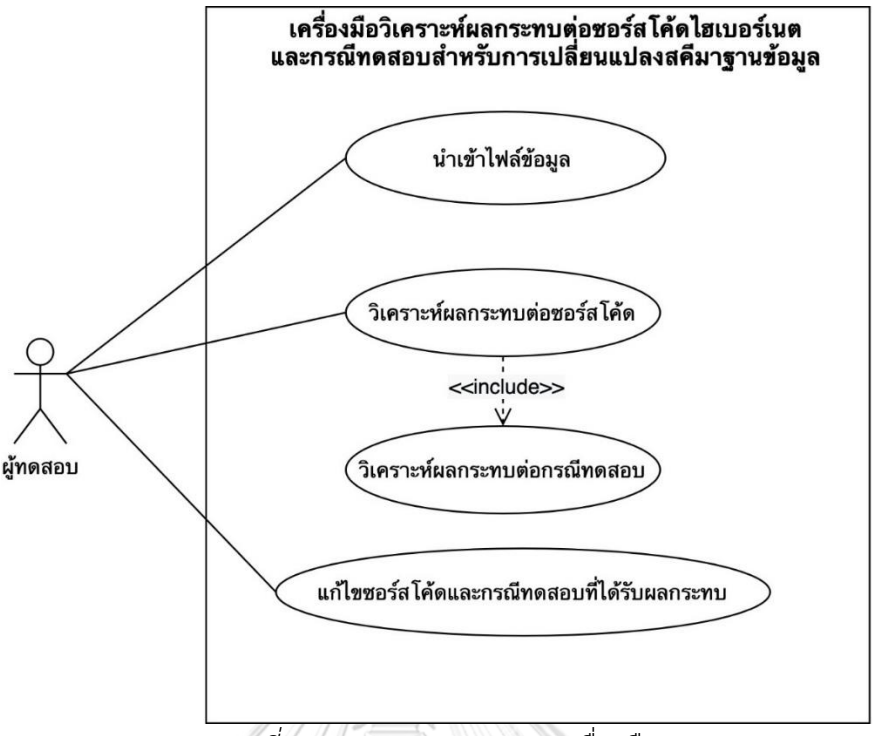
แผนภาพยูสเคสของเครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล ดังแสดงในรูปที่ 4-1 รายละเอียดของแต่ละยูสเคสสามารถดูเพิ่มเติมได้ที่ภาคผนวก ก โดยแผนภาพยูสเคสสามารถแบ่งออกเป็น 4 ยูสเคสหลัก ดังต่อไปนี้

1. ยูสเคสนำเข้าไฟล์ข้อมูล เป็นการนำเข้าไฟล์ข้อมูลโดยผู้ใช้เพื่อจัดเก็บในเครื่องมือจะนำไฟล์ข้อมูลไปประมวลผลต่อเพื่อใช้ในการวิเคราะห์หาผลกระทบต่อซอร์สโค้ดและกรณีทดสอบในลำดับถัดไป

2. ยูสเคสวิเคราะห์ผลกระทบต่อซอร์สโค้ด เป็นการวิเคราะห์หาการเปลี่ยนแปลงที่เกิดขึ้นในสคีมาฐานข้อมูลเมื่อเปรียบเทียบกับไฟล์ซอร์สโค้ดเพอซิสเทินและไฟล์ซอร์สโค้ดคอนโทรลเลอร์

3. ยูสเคสวิเคราะห์ผลกระทบต่อกรณีทดสอบ เป็นการวิเคราะห์หากรณีทดสอบที่ได้รับผลกระมบจากการเปลี่ยนแปลงที่เกิดขึ้นกับสคีมาฐานข้อมูล

4. ยูสเคสแก้ไขซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบ เป็นการปรับปรุงแก้ไขข้อมูลซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบ โดยเครื่องมือจะแสดงข้อมูลซอร์สโค้ดและกรณีทดสอบที่แก้ไขเสร็จแล้วผ่านยูสเคสการแสดงผลรายงาน แต่หากไม่มีผลกระทบเกิดขึ้นเครื่องมือจะไม่แสดงผลรายงานแต่จะแสดงข้อความแจ้งเตือนให้ผู้ใช้ได้ทราบว่าไม่สามารถแก้ไขข้อมูลได้เนื่องจากไม่มีผลกระทบเกิดขึ้น



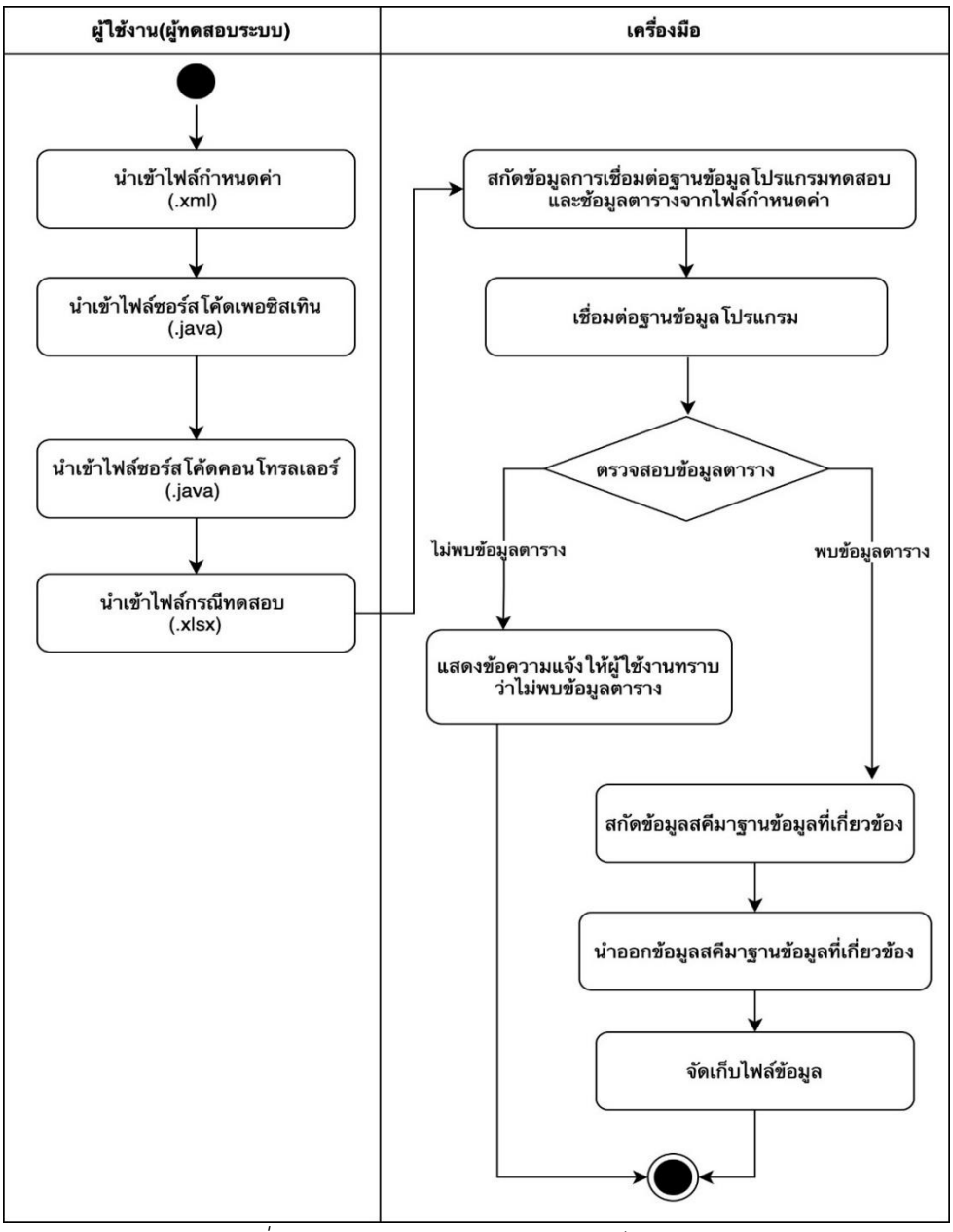
รูปที่ 4-1 แผนภาพยูสเคสของเครื่องมือ

4.1.2. แผนภาพกิจกรรม

แผนภาพกิจกรรมของเครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล ระกอบด้วยแผนภาพ 4 กิจกรรมหลัก โดยมีรายละเอียดดังนี้

1. แผนภาพกิจกรรมการนำเข้าไฟล์ข้อมูล

แผนภาพกิจกรรมการนำเข้าไฟล์ข้อมูล แสดงในรูปที่ 4-2 เป็นขั้นตอนแรกที่ผู้ใช้นำเข้าข้อมูลสำหรับโปรแกรมทดสอบโดยเริ่มต้นจากผู้ใช้นำเข้า 4 ไฟล์ได้แก่ ไฟล์กำหนดค่าไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบ หลังจากนั้นเครื่องมือจะทำหน้าที่ตรวจสอบความถูกต้องของข้อมูลนำเข้า หากข้อมูลมีความถูกต้อง เครื่องมือจะสกัดข้อมูลการเชื่อมต่อของโปรแกรมทดสอบและข้อมูลตารางจากไฟล์กำหนดค่าและไฟล์เพอซิสเทิน เพื่อใช้ในการเชื่อมต่อกับฐานข้อมูลของโปรแกรมทดสอบ ภายหลังจากการเชื่อมต่อฐานข้อมูลหากเครื่องมือตรวจสอบแล้วไม่พบข้อมูลชื่อตารางที่ตรงกันกับชื่อตารางในสคีมาฐานข้อมูลของโปรแกรมทดสอบ เครื่องมือจะแสดงข้อความแจ้งให้ผู้ใช้ได้ทราบว่าไม่พบข้อมูลตาราง แต่หากทำการตรวจสอบแล้วพบข้อมูลตารางที่ตรงกัน เครื่องมือจะสกัดสคีมาฐานข้อมูลของตารางที่เกี่ยวข้องและนำออกข้อมูลมาบันทึกและจัดเก็บในฐานข้อมูลของเครื่องมือ



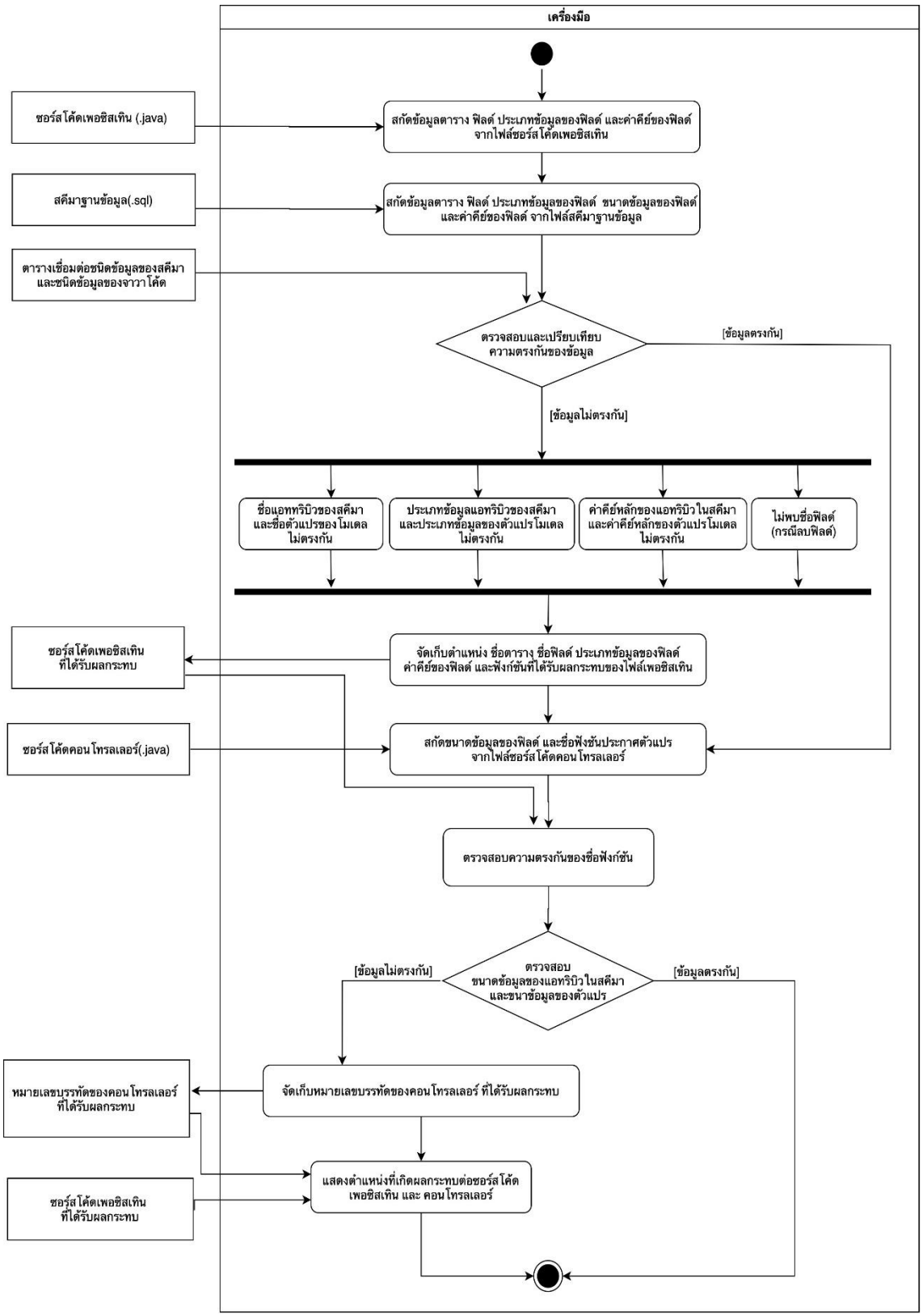
รูปที่ 4-2 แผนภาพกิจกรรมการนำเข้าไฟล์ข้อมูล

2. แผนภาพกิจกรรมการวิเคราะห์ผลกระทบต่อซอร์สโค้ด

แผนภาพกิจกรรมการวิเคราะห์ผลกระทบต่อซอร์สโค้ด ดังแสดงในรูปที่ 4-3 เครื่องมือจะดึงข้อมูลไฟล์ซอร์สโค้ดเพอซิสเทินและสคีมาฐานข้อมูลที่จัดเก็บไว้มาสกัดข้อมูลตาราง ฟิลด์ ประเภทของฟิลด์ และค่าคีย์ของฟิลด์ หลังจากนั้นเครื่องมือจะตรวจสอบและเปรียบเทียบความตรงกันของข้อมูลระหว่างข้อมูลที่สกัดมาจากซอร์สโค้ดเพอซิสเทินและข้อมูลที่สกัดมาจากสคีมาฐานข้อมูลของโปรแกรมทดสอบ หากพบว่าข้อมูลตรงกัน หมายความว่าข้อมูลสคีมาฐานข้อมูลไม่มีการเปลี่ยนแปลง แต่หากพบว่าข้อมูลไม่ตรงกัน หมายความว่าเกิดการเปลี่ยนแปลงเกิดขึ้นในสคีมาฐานข้อมูล เครื่องมือจะจัดเก็บชื่อแอตทริบิวต์ ประเภทข้อมูลของแอตทริบิวต์ และฟังก์ชันที่ได้รับผลกระทบของไฟล์ซอร์สโค้ดเพอซิสเทินไว้ในฐานข้อมูลของเครื่องมือ จากนั้นจะนำฟังก์ชันที่ได้รับผลกระทบของไฟล์ซอร์สโค้ดเพอซิสเทินไปเปรียบเทียบกับซอร์สโค้ดคอนโทรลเลอร์เพื่อตรวจสอบหาความตรงกันของชื่อฟังก์ชัน และขนาดของแอตทริบิวต์ที่เปลี่ยนแปลงไป หากผลการวิเคราะห์พบว่ามีชื่อฟังก์ชันที่ตรงกันในไฟล์ซอร์สโค้ดคอนโทรลเลอร์ เครื่องมือจะจัดเก็บหมายเลขบรรทัดของซอร์สโค้ดคอนโทรลเลอร์ไว้ในฐานข้อมูลของเครื่องมือเพื่อใช้ในขั้นตอนการวิเคราะห์ผลกระทบต่อกรณีทดสอบเป็นลำดับต่อไป หลังจากการวิเคราะห์หาผลกระทบต่อซอร์สโค้ดเพอซิสเทินและคอนโทรลเลอร์ เครื่องมือจะแสดงผลรายงานให้ผู้ใช้ได้ทราบโดยการทำเครื่องหมายด้วยสีเพื่อบอกถึงตำแหน่งที่เกิดผลกระทบในซอร์สโค้ดผ่านทางหน้าจอ

3. แผนภาพกิจกรรมการวิเคราะห์ผลกระทบต่อกรณีทดสอบ

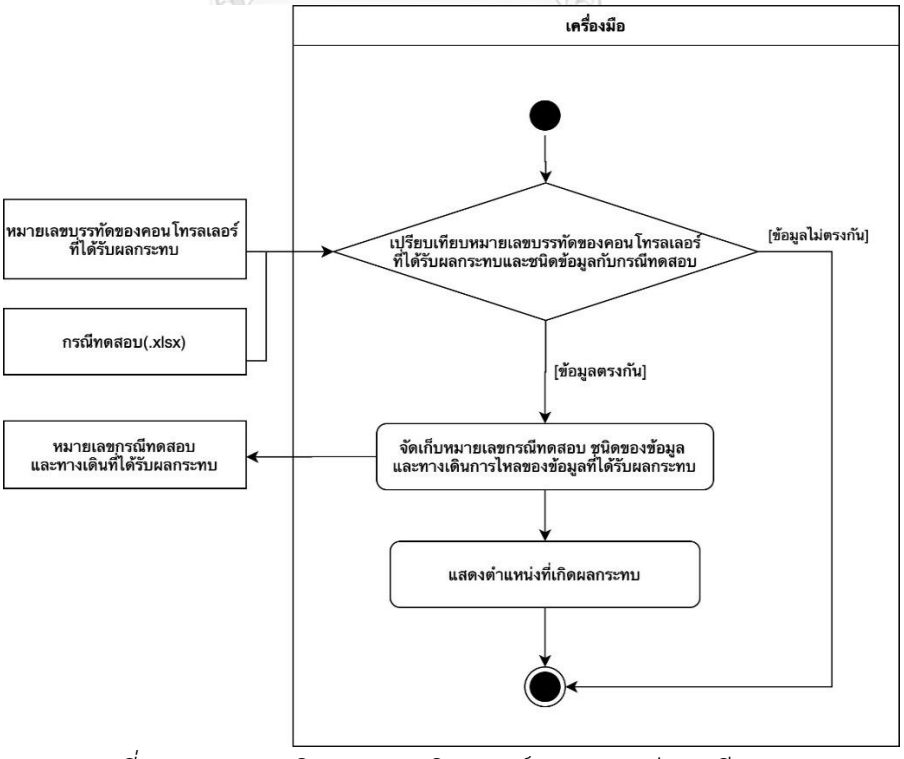
แผนภาพกิจกรรมการวิเคราะห์ผลกระทบต่อกรณีทดสอบ ดังแสดงในรูปที่ 4-4 แสดงถึงขั้นตอนของเครื่องมือที่วิเคราะห์หาผลกระทบต่อกรณีทดสอบ โดยเครื่องมือจะทำการเปรียบเทียบหมายเลขบรรทัดของซอร์สโค้ดคอนโทรลเลอร์ที่ได้รับผลกระทบและประเภทข้อมูลของแอตทริบิวต์กับไฟล์กรณีทดสอบว่ามีความตรงกันหรือไม่ กรณีที่พบว่าหมายเลขบรรทัดของซอร์สโค้ดคอนโทรลเลอร์ตรงกันกับหมายเลขทางเดินการไหลของข้อมูลในไฟล์กรณีทดสอบ แสดงว่ากรณีทดสอบนั้นได้รับผลกระทบ เครื่องมือจะจัดเก็บหมายเลขกรณีทดสอบ ประเภทข้อมูลของแอตทริบิวต์และทางเดินการไหลของข้อมูลที่ได้รับผลกระทบไว้ในฐานข้อมูลของเครื่องมือ และจะแสดงหมายเลขตำแหน่งที่เกิดผลกระทบให้ผู้ใช้ได้รับทราบผ่านทางหน้าจอ



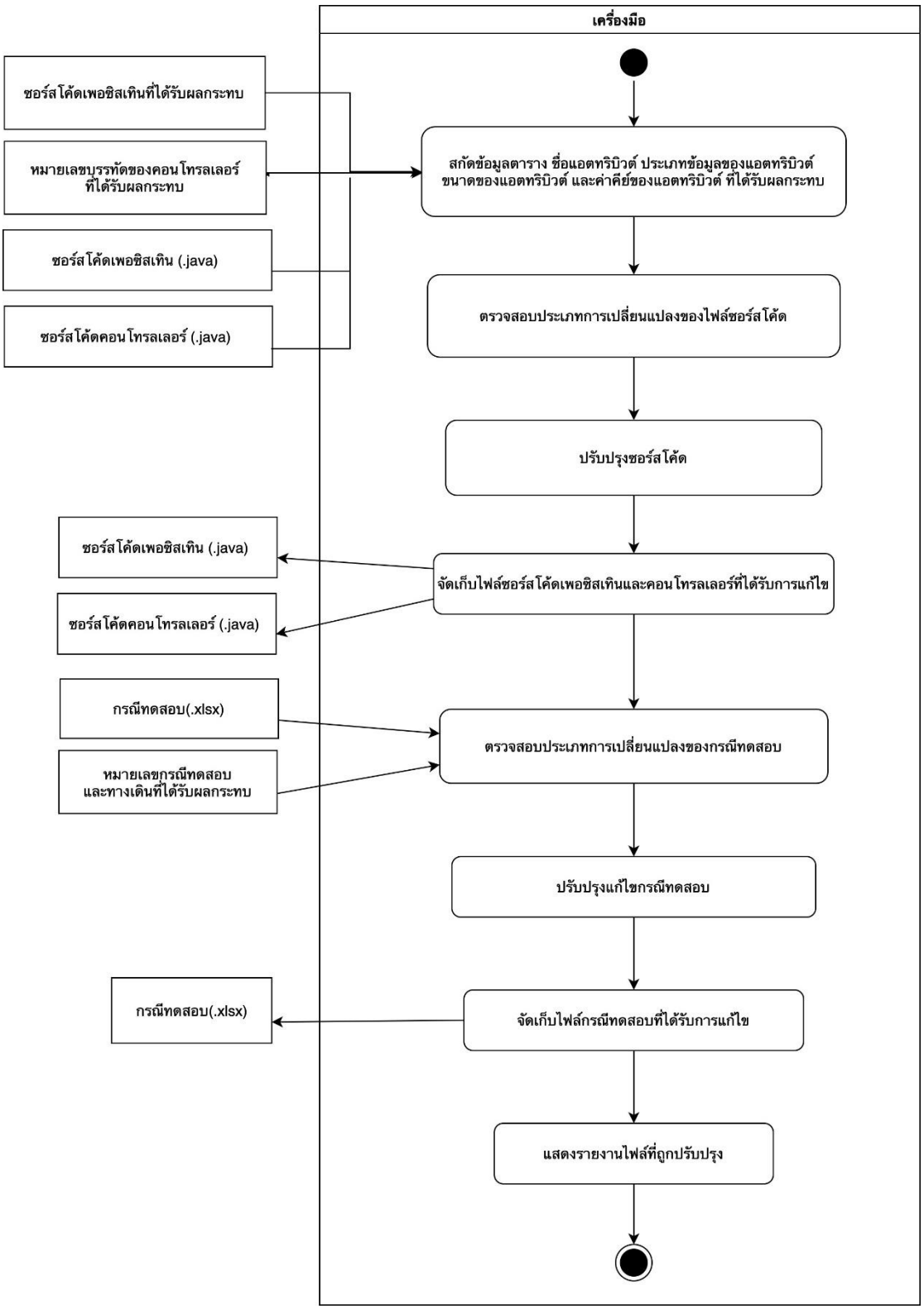
รูปที่ 4-3 แผนภาพกิจกรรมการวิเคราะห์ผลกระทบต่อซอร์สโค้ด

4. แผนภาพกิจกรรมการปรับปรุงแก้ไขซอร์สโค้ดและกรณีทดสอบ

แผนภาพกิจกรรมการการปรับปรุงแก้ไขซอร์สโค้ดและกรณีทดสอบ ดังแสดงในรูปที่ 4-5 แสดงขั้นตอนการปรับปรุงแก้ไขซอร์สโค้ดและกรณีทดสอบ โดยการทำงานของเครื่องมือจะปรับปรุงแก้ไขซอร์สโค้ดเพอซิสเทินและซอร์สโค้ดคอนโทรลเลอร์ก่อนเป็นอันดับแรก หลังจากนั้นจึงปรับปรุงแก้ไขกรณีทดสอบ เริ่มต้นจากเครื่องมือนำไฟล์ซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบและไฟล์หมายเลขบรรทัดของคอนโทรลเลอร์ที่ได้รับผลกระทบ มาสกัดข้อมูลตาราง ชื่อแอตทริบิวต์ ประเภทข้อมูลของแอตทริบิวต์ ขนาดของแอตทริบิวต์ และค่าคีย์ของแอตทริบิวต์ เพื่อนำมาตรวจสอบประเภทของการเปลี่ยนแปลงกับไฟล์ซอร์สโค้ดเพอซิสเทินและไฟล์ซอร์สโค้ดคอลโทรลเลอร์ตั้งต้น โดยการเปลี่ยนแปลงในแต่ละประเภทจะทำการปรับปรุงแก้ไขที่แตกต่างกัน เมื่อทำการแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทินและไฟล์ซอร์สโค้ดคอนโทรลเลอร์เสร็จ จะจัดเก็บไฟล์ซอร์สโค้ดที่ได้รับการแก้ไขไว้ในฐานข้อมูลของเครื่องมือ ขั้นตอนต่อไปเครื่องมือจะทำการปรับปรุงแก้ไขกรณีทดสอบ โดยนำเข้าไฟล์หมายเลขบรรทัดของคอนโทรลเลอร์ที่ได้รับผลกระทบและไฟล์กรณีทดสอบตั้งต้น เพื่อตรวจสอบประเภทของการเปลี่ยนแปลง โดยจะตรวจสอบประเภทของการเปลี่ยนแปลง 4 ประเภท ได้แก่ การแก้ไขชื่อแอตทริบิวต์ การแก้ไขประเภทแอตทริบิวต์ การแก้ไขขนาดแอตทริบิวต์ และทางเดินของกรณีทดสอบตรงกับทางเดินได้รับผลกระทบ เมื่อทำการแก้ไขไฟล์กรณีทดสอบแล้วเสร็จ จะจัดเก็บไฟล์กรณีทดสอบที่ได้รับการแก้ไขไว้ในฐานข้อมูลของเครื่องมือ และแสดงรายงานไฟล์ที่ถูกปรับปรุงให้ผู้ใช้ได้รับทราบผ่านทางหน้าจอ



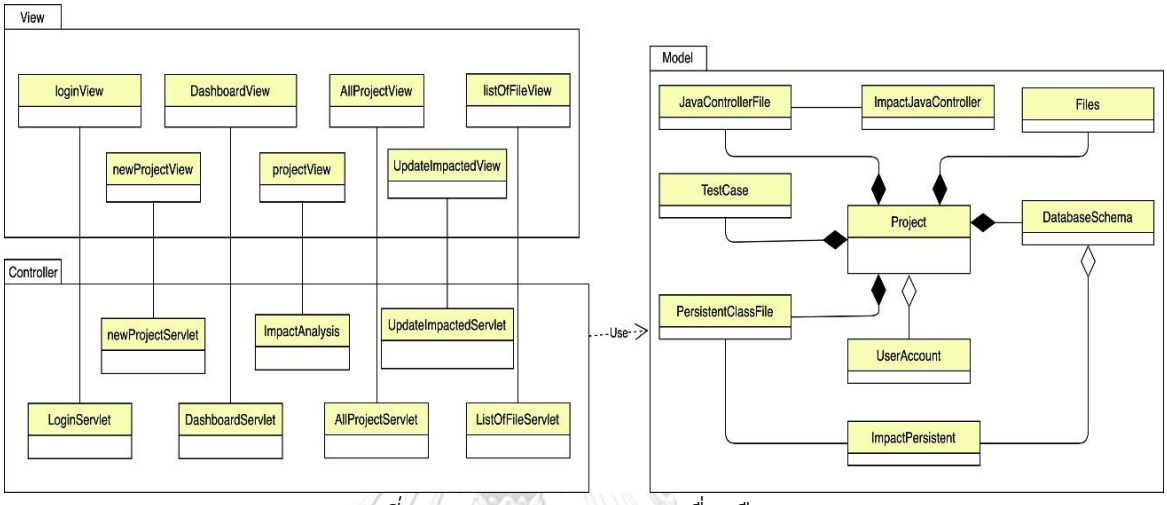
รูปที่ 4-4 แผนภาพกิจกรรมการวิเคราะห์ผลกระทบต่อกรณีทดสอบ



รูปที่ 4-5 แผนภาพกิจกรรมการปรับปรุงแก้ไขซอร์สโค้ดและกรณีทดสอบ

4.1.3. แผนภาพคลาส

แผนภาพคลาสของเครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล แสดงได้ดังรูปที่ 4-6 ซึ่งประกอบด้วย 3 แพ็คเกจหลัก ซึ่งมีรายละเอียดดังต่อไปนี้

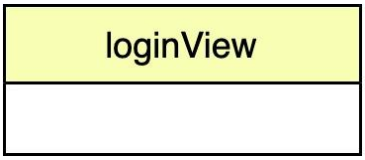


รูปที่ 4-6 แผนภาพคลาสของเครื่องมือ

1. แพ็คเกจ view

แพ็คเกจ view ประกอบด้วยคลาสที่ทำหน้าที่เป็นคลาสกลุ่มวิว (View) ตามรูปแบบเอ็มวีซี รายละเอียดของแต่ละคลาสมีดังต่อไปนี้

1.1. คลาส loginView เป็นคลาสส่วนต่อประสานกับผู้ใช้สำหรับการลงชื่อเข้าใช้งาน โดยจะมีฟังก์ชันการตรวจสอบข้อมูล รูปที่ 4-7 แสดงรายละเอียดของคลาส loginView

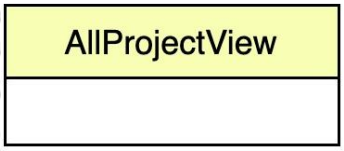


รูปที่ 4-7 คลาส loginView

1.2. คลาส DashboardView เป็นคลาสส่วนต่อประสานที่ทำหน้าที่ในการแสดงข้อมูลของโปรแกรม รวมไปถึงข้อมูลของผู้พัฒนาโปรแกรม รูปที่ 4-8 แสดงรายละเอียดของคลาส DashboardView

รูปที่ 4-8 คลาส DashboardView

1.3. คลาส AllProjectView เป็นคลาสส่วนต่อประสานกับผู้ใช้สำหรับแสดงรายการโปรเจคทั้งหมดที่ผู้ใช้งานได้ทำการสร้างไว้เพื่อให้ผู้ใช้สามารถเลือกโปรเจคเพื่อนำไปวิเคราะห์ผลกระทบในกระบวนการถัดไปได้ รูปที่ 4-9 แสดงรายละเอียดของคลาส AllProjectView

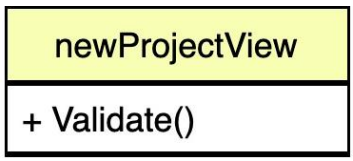


รูปที่ 4-9 คลาส AllProjectView

1.4. คลาส listOfFileView เป็นคลาสส่วนต่อประสานกับผู้ใช้สำหรับแสดงรายการไฟล์ทั้งหมดในโปรแกรม โดยผู้ใช้สามารถนำออกไฟล์ข้อมูลผ่านหน้าจอได้ รูปที่ 4-10 แสดงรายละเอียดของคลาส listOfFileView

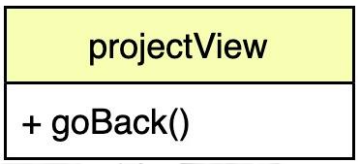
รูปที่ 4-10 คลาส listOfFileView

1.5. คลาส newProjectView เป็นคลาสส่วนต่อประสานกับผู้ใช้สำหรับการสร้างโปรเจค โดยให้ผู้ใช้กรอกข้อมูลชื่อโปรเจค และนำเข้าข้อมูล 4 ไฟล์ ได้แก่ ไฟล์กำหนดค่า ไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบ นอกจากนี้คลาส newProjectView มีการดำเนินการสำหรับการตรวจสอบความถูกต้องของไฟล์นำเข้า4ไฟล์ รูปที่ 4-11 แสดงรายละเอียดของคลาส newProjectView



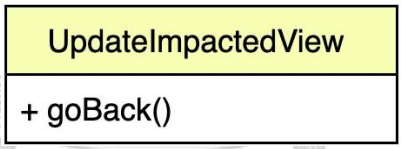
รูปที่ 4-11 คลาส newProjectView

1.6. คลาส projectView เป็นคลาสส่วนต่อประสานกับผู้ใช้สำหรับการแสดงรายละเอียดไฟล์นำเข้าทั้ง 4 ไฟล์ โดยผู้ใช้สามารถร้องขอให้มีการวิเคราะห์ผลกระทบต่อซอร์สโค้ดและกรณีทดสอบ รูปที่ 4-12 แสดงรายละเอียดของคลาส projectView



รูปที่ 4-12 คลาส projectView

1.7. คลาส UpdateImpactedView เป็นคลาสส่วนต่อประสานกับผู้ใช้สำหรับการแสดงรายละเอียดไฟล์ที่ได้ทำการปรับปรุงแก้ไขเสร็จสิ้น โดยผู้ใช้สามารถร้องขอให้มีการแก้ไขปรับปรุงซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบ และสามารถดาวน์โหลดไฟล์ซอร์สโค้ดและกรณีทดสอบที่แก้ไขเสร็จสิ้นได้ รูปที่ 4-13 แสดงรายละเอียดของคลาส UpdateImpactedView

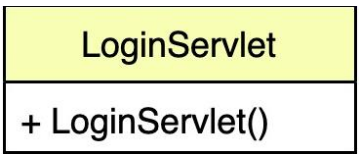


รูปที่ 4-13 คลาส *UpdateImpactedView*

2. แพ็คเกจ Controller

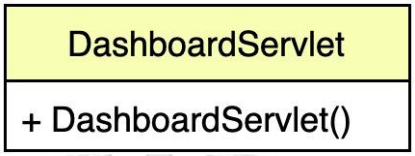
แพ็คเกจ controller ประกอบด้วยคลาสที่ทำหน้าที่เป็นคลาสกลุ่มคอลโทลเลอร์(Controller) ตามรูปแบบเอ็มวีซี โดยแต่ละคลาสมีรายละเอียดดังต่อไปนี้

2.1. คลาส LoginServlet เป็นคลาสชนิดควบคุมที่มีหน้าที่ในการประมวลผลโดยรับคำสั่งมาจากคลาส loginView นอกจากนี้คลาส LoginServlet ยังมีการดำเนินการสำหรับการตรวจสอบผู้ที่มาลงชื่อเข้าใช้งานเครื่องมือ โดยจะทำการตรวจสอบจากชื่อผู้ใช้งาน (username) และรหัสผ่าน (password) ว่ามีความถูกต้องตรงกันหรือไม่ จากนั้นจึงดำเนินการออกจากเครื่องมือ รูปที่ 4-14 แสดงรายละเอียดของคลาส LoginServlet



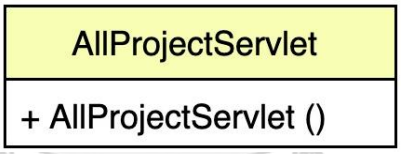
รูปที่ 4-14 คลาส LoginServlet

2.2. คลาส DashboardServlet เป็นคลาสชนิดควบคุมที่ทำหน้าที่ในการประมวลผลดึงข้อมูลผู้ใช้งานจากคลาส UserAccount ส่งไปแสดงยังเว็บเบราว์เซอร์ผ่านคลาส DashboardView รูปที่ 4-15 แสดงรายละเอียดของคลาส DashboardServlet



รูปที่ 4-15 คลาส DashboardServlet

2.3. คลาส AllProjectServlet เป็นคลาสชนิดควบคุมที่มีหน้าที่ในการประมวลผลดึงข้อมูลโปรเจคและข้อมูลผู้ใช้งานส่งไปแสดงยังเว็บเบราว์เซอร์ผ่านคลาส AllProjectView รูปที่ 4-16 แสดงรายละเอียดของคลาส AllProjectServlet

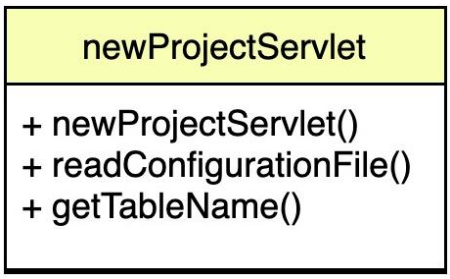


รูปที่ 4-16 คลาส AllProjectServlet

2.4. คลาส listOfFileServlet เป็นคลาสชนิดควบคุมที่มีหน้าที่ในการประมวลผลดึงข้อมูลไฟล์จากคลาส Files ส่งไปแสดงยังเว็บเบราว์เซอร์ผ่านคลาส listOfFileView นอกจากนี้คลาส listOfFileServlet ยังมีการดำเนินการสำหรับดาวน์โหลดข้อมูลไฟล์ลงเครื่องคอมพิวเตอร์ของผู้ใช้ รูปที่ 4-17 แสดงรายละเอียดของคลาส listOfFileServlet

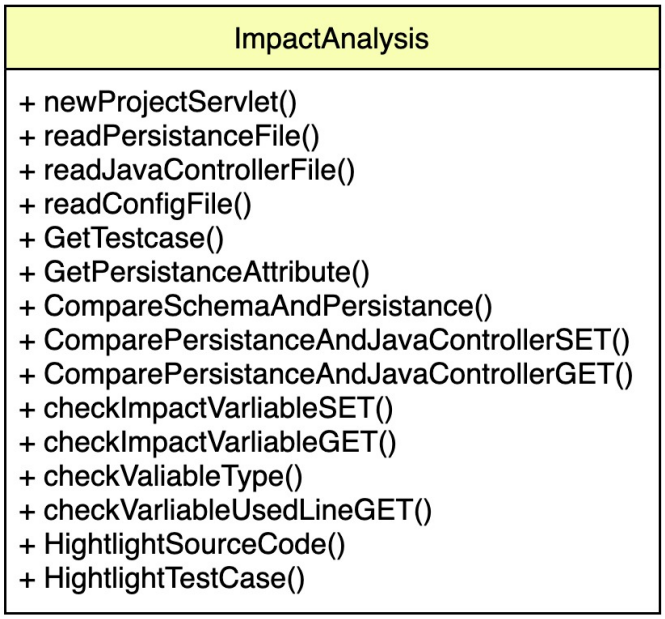
รูปที่ 4-17 คลาส listOfFileServlet

2.5. คลาส newProjectServlet เป็นคลาสชนิดควบคุมที่มีหน้าที่ในการประมวลผลซึ่งรับคำสั่งมาจากคลาส newProjectView โดยมีการดำเนินการสำหรับอ่านไฟล์กำหนดค่าเพื่อสกัดข้อมูลสำหรับการเชื่อมต่อฐานข้อมูลของโปรแกรมทดสอบ การดำเนินการสกัดชื่อตารางจากไฟล์ซอร์สโค้ดเพอซิสเทินเพื่อใช้ในการสกัดข้อมูลตารางที่เกี่ยวข้องในฐานข้อมูลของโปรแกรมทดสอบ และการดำเนินการบันทึกข้อมูลโปรเจคและข้อมูลไฟล์นำเข้าเข้าทั้ง 4 ไฟล์เข้าสู่ฐานข้อมูลของเครื่องมือ รูปที่ 4-18 แสดงรายละเอียดของคลาส newProjectServlet



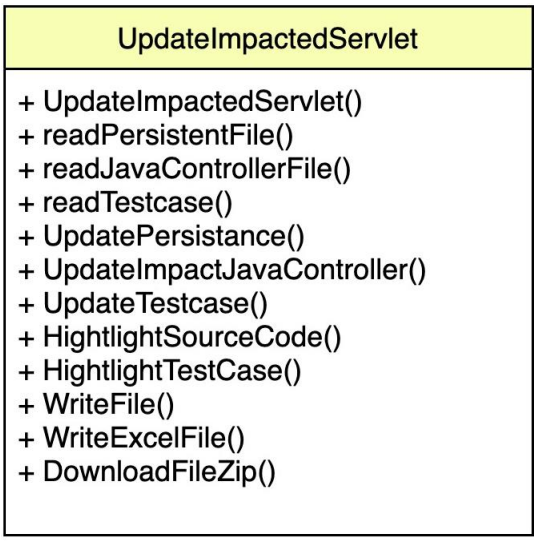
รูปที่ 4-18 คลาส newProjectServlet

2.6. คลาส ImpactAnalysis เป็นคลาสชนิดควบคุมที่มีหน้าที่ในการประมวลผลดึงข้อมูลจากคลาส Files และ DatabaseShema ส่งไปแสดงยังเว็บเบราว์เซอร์ผ่านคลาส projectView นอกจากนี้ยังเป็นคลาสหลักที่ใช้ทำงานในส่วนของการวิเคราะห์หาผลกระทบต่อซอร์สโค้ดและกรณีทดสอบ โดยเครื่องมือจะแสดงตำแหน่งที่เกิดผลกระทบของซอร์สโค้ดและกรณีทดสอบด้วยการระบายสี (Highlight) รวมถึงแสดงตารางรายละเอียดในคลาส projectView รูปที่ 4-19 แสดงรายละเอียดของคลาส ImpactAnalysis



รูปที่ 4-19 คลาส ImpactAnalysis

2.7. คลาส UpdateImpactedServlet เป็นคลาสชนิดควบคุมที่มีหน้าที่ในการปรับปรุงแก้ไขซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบจากการเปลี่ยนแปลงของสคีมาฐานข้อมูล โดยหลังจากการปรับปรุงแก้ไขแล้วเสร็จ จะดำเนินการบันทึกข้อมูลไฟล์ที่ได้รับการแก้ไขเข้าสู่ฐานข้อมูลของเครื่องมือ โดยเครื่องมือจะแสดงตำแหน่งที่ได้ทำการปรับปรุงแก้ไขของซอร์สโค้ดและกรณีทดสอบด้วยการไฮไลท์สี ผ่านคลาส UpdateImpactedView รูปที่ 4-20 แสดงรายละเอียดของคลาส UpdateImpactedServlet

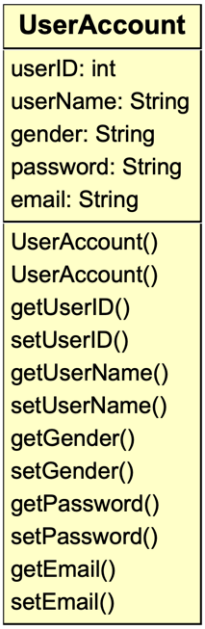


รูปที่ 4-20 UpdateImpactedServlet

3. แพ็คเกจ model

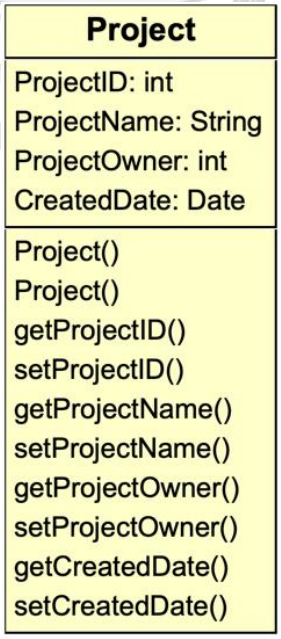
แพ็คเกจ model ประกอบด้วยคลาสที่ทำหน้าที่เป็นคลาสกลุ่มโมเดล (Model) ตามรูปแบบเอ็มวีซี โดยแต่ละคลาสมีรายละเอียดดังต่อไปนี้

3.1. คลาส UserAccount เป็นคลาสชนิดเอนทิตีที่ทำหน้าเป็นโครงสร้างสำหรับจัดเก็บข้อมูลของผู้ใช้ซึ่งจะมีการระบุ รหัสอ้างอิงผู้ใช้ ชื่อผู้ใช้ รหัสผ่าน เพศ และ อีเมล โดยมีการดำเนินการทั่วไปคือการสร้าง และเรียกดูข้อมูล รูปที่ 4-21 แสดงรายละเอียดของคลาส UserAccount



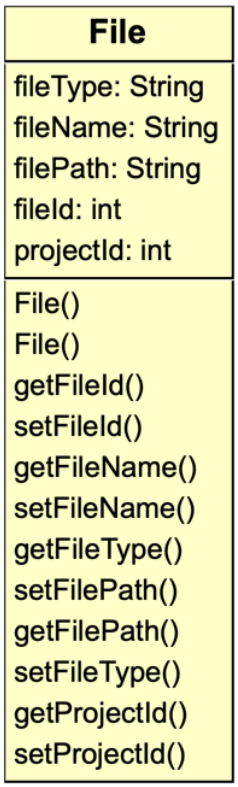
รูปที่ 4-21 รายละเอียดของคลาส UserAccount

3.2. คลาส Project เป็นคลาสชนิดเอนทิตีที่ทำหน้าเป็นโครงสร้างสำหรับจัดเก็บข้อมูลของโครงการ โดยจะมีการระบุ รหัสโครงการ ชื่อโครงการ รหัสผู้ใช้งาน และวันที่สร้างโครงการ นอกจากนี้คลาส Project ยังมีการดำเนินการทั่วไปคือการสร้าง และเรียกดูข้อมูล รูปที่ 4-22 แสดงรายละเอียดของคลาส Project



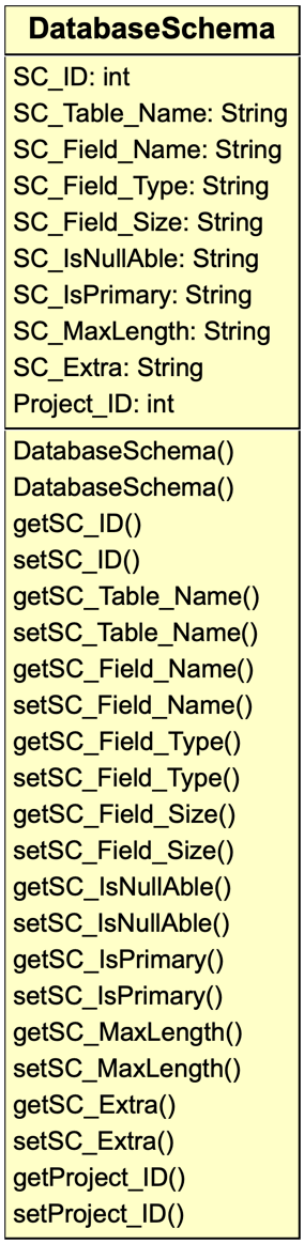
รูปที่ 4-22 คลาส Project

3.3. คลาส Files เป็นคลาสชนิดเอนทิตีที่ทำหน้าเป็นโครงสร้างสำหรับจัดเก็บข้อมูลของไฟล์ โดยจะมีการระบุ รหัสไฟล์ ชื่อไฟล์ ชนิดของไฟล์ เส้นทางเดิน รหัสโครงการ วันที่สร้างไฟล์ รหัสอ้างอิงผู้ใช้ที่สร้างไฟล์ และสถานะการปรับปรุงแก้ไขไฟล์ นอกจากนี้คลาส Files ยังมีการดำเนินการทั่วไปคือการสร้าง และเรียกดูข้อมูล รูปที่ 4-23 แสดงรายละเอียดของคลาส Files



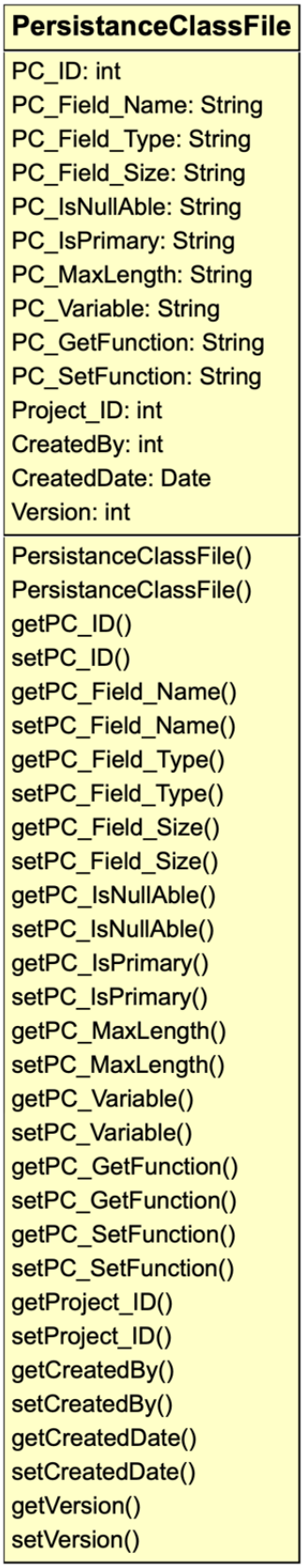
รูปที่ 4-23 รายละเอียดของคลาส Files

3.4. คลาส DatabaseSchema เป็นคลาสชนิดเอนทิตีที่ทำหน้าเป็นโครงสร้างสำหรับจัดเก็บสคีมาฐานข้อมูลของตารางที่เกี่ยวข้องกับโปรแกรมทดสอบ โดยจะมีการระบุ รหัสสคีมาฐานข้อมูล ชื่อตาราง ชื่อแอตทริบิวต์ ชนิดของฟิลด์ ขนาดของฟิลด์ คีย์หลัก และรหัสโครงการ รูปที่ 4-24 แสดงรายละเอียดของคลาส DatabaseSchema



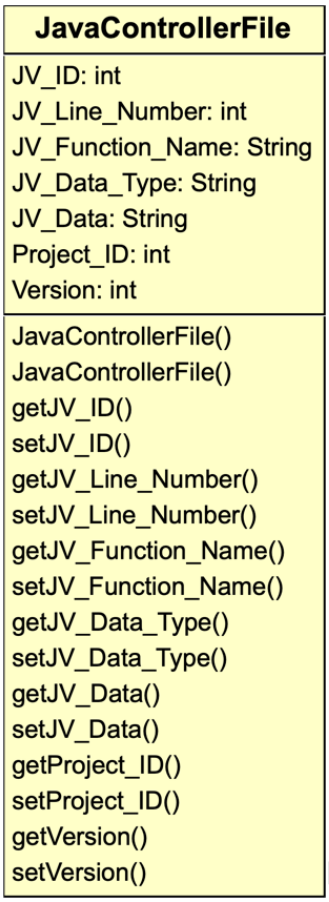
รูปที่ 4-24 คลาส DatabaseSchema

3.5. คลาส PersistentClassFile เป็นคลาสชนิดเอนทิตีที่ทำหน้าเป็นโครงสร้างสำหรับจัดเก็บข้อมูลของไฟล์ซอร์สโค้ดเพอซิสเทิน โดยจะมีการระบุ รหัสเพอซิสเทิน ชื่อแอตทริบิวต์ ชนิดของฟิลด์ ขนาดของฟิลด์ ค่าว่าง คีย์หลัก ชื่อตัวแปร ชื่อฟังก์ชันเรียกดูข้อมูล ชื่อฟังก์ชันกำหนดค่าข้อมูล รหัสโครงการ รหัสอ้างอิงผู้ใช้ วันที่สร้าง และหมายเลขเวอร์ชันการจัดเก็บของไฟล์ นอกจากนี้คลาส PersistentClassFile ยังมีการดำเนินการทั่วไปคือการสร้าง และเรียกดูข้อมูล รูปที่ 4-25 แสดงรายละเอียดของคลาส PersistentClassFile



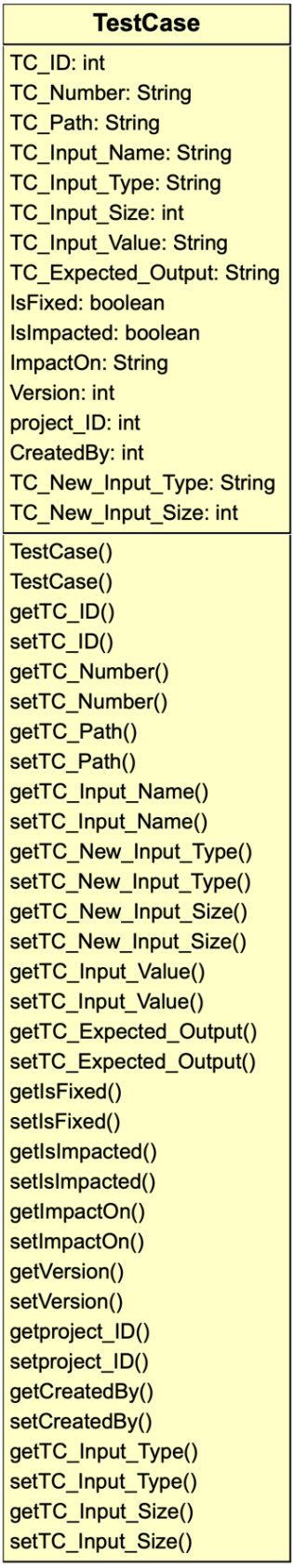
รูปที่ 4-25 คลาส PersistentClassFile

3.6. คลาส JavaControllerFile เป็นคลาสชนิดเอนทิตีที่ทำหน้าเป็นโครงสร้างสำหรับจัดเก็บข้อมูลของไฟล์ซอร์สโค้ดคอนโทรลเลอร์ โดยจะมีการระบุ รหัสคอนโทรลเลอร์ หมายเลขบรรทัด ชนิดของฟิลด์ ขนาดของฟิลด์ รหัสโครงการ และหมายเลขเวอร์ชันการจัดเก็บไฟล์ซอร์สโค้ด รูปที่ 4-26 แสดงรายละเอียดของคลาส JavaControllerFile



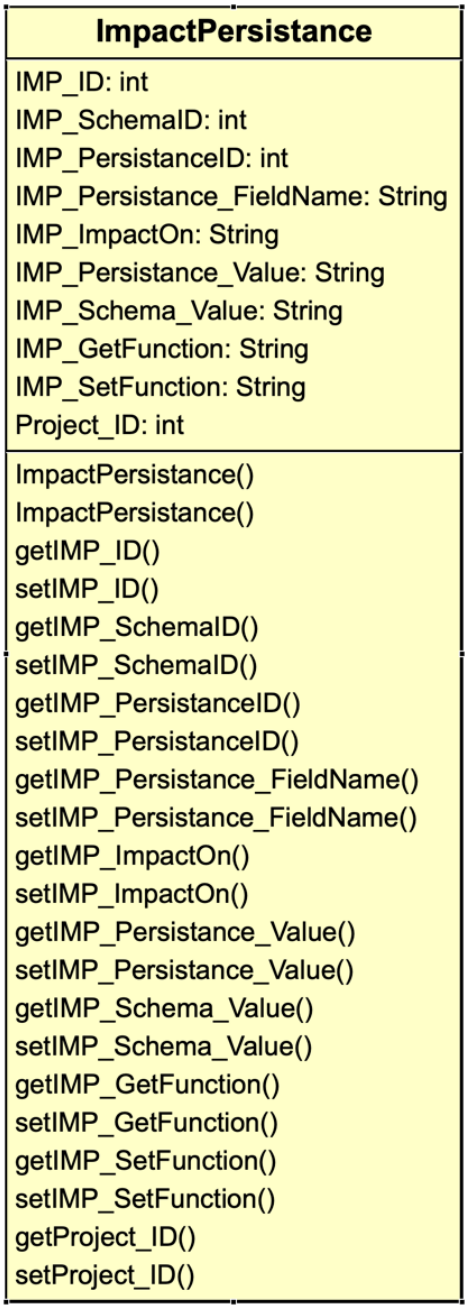
รูปที่ 4-26 คลาส JavaControllerFile

3.7. คลาส TestCase เป็นคลาสชนิดเอนทิตีที่ทำหน้าเป็นโครงสร้างสำหรับจัดเก็บข้อมูลของกรณีทดสอบ โดยจะมีการระบุ รหัสกรณีทดสอบ หมายเลขกรณีทดสอบ เส้นทางเดินของกรณีทดสอบ ชื่อข้อมูลนำเข้า ชนิดของข้อมูลนำเข้า ขนาดของข้อมูลนำเข้า ค่าของข้อมูลนำเข้า ผลลัพธ์ที่คาดหวัง การปรับปรุงแก้ไขกรณีทดสอบ ผลกระทบต่อกรณีทดสอบ ข้อมูลที่กรณีทดสอบถูกกระทบ หมายเลขเวอร์ชันของไฟล์ รหัสโครงการ รหัสอ้างอิงผู้ใช้ ค่าใหม่ชนิดของข้อมูลนำเข้า และค่าใหม่ขนาดของข้อมูลนำเข้า รูปที่ 4-27 แสดงรายละเอียดของคลาส TestCase



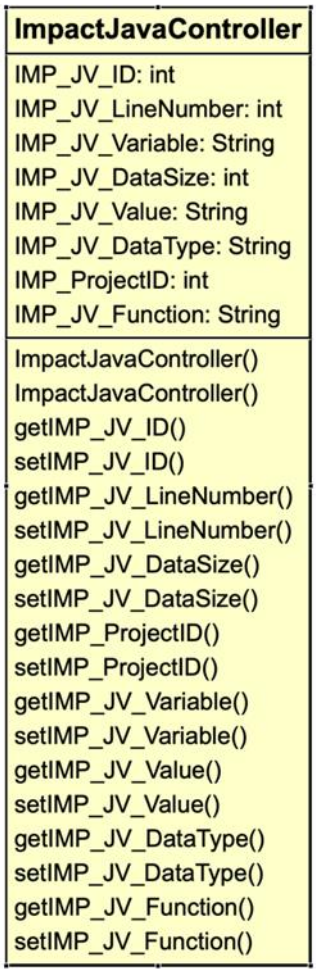
รูปที่ 4-27 คลาส TestCase

3.8. คลาส ImpactPersistent เป็นคลาสชนิดเอนทิตีที่ทำหน้าเป็นโครงสร้างสำหรับจัดเก็บข้อมูลของไฟล์ซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบ โดยจะมีการระบุ รหัสผลกระทบเพอซิสเทิน รหัสสคีมาฐานข้อมูล รหัสเพอซิสเทิน ชื่อฟิลด์ที่ได้รับผลกระทบ ผลกระทบที่เกิดขึ้นต่อเพอซิสเทิน ค่าข้อมูลในเพอซิสเทิน ค่าข้อมูลในสคีมาฐานข้อมูล ชื่อฟังก์ชันเรียกดูข้อมูล ชื่อฟังก์ชันกำหนดค่าข้อมูล และรหัสโครงการ รูปที่ 4-28 แสดงรายละเอียดของคลาส ImpactPersistent



รูปที่ 4-28 คลาส ImpactPersistent

3.9. คลาส ImpactJavaController เป็นคลาสชนิดเอนทิตีที่ทำหน้าเป็นโครงสร้างสำหรับจัดเก็บข้อมูลของไฟล์ซอร์สโค้ดคอนโทรลเลอร์ที่ได้รับผลกระทบ โดยจะมีการระบุ รหัสผลกระทบคอนโทรลเลอร์ หมายเลขบรรทัด ชื่อตัวแปร ขนาดของข้อมูล ค่าของข้อมูล ชนิดของข้อมูล รหัสโครงการ และชื่อฟังก์ชันที่ได้รับผลกระทบ รูปที่ 4-29 แสดงรายละเอียดของคลาส ImpactJavaController



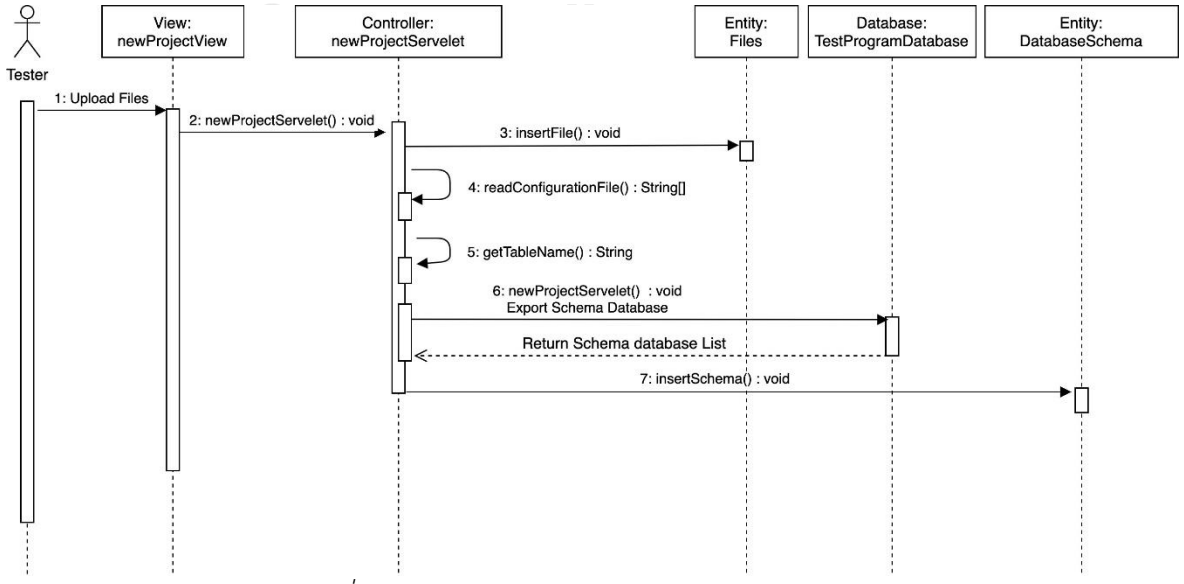
รูปที่ 4-29 คลาส ImpactJavaController

4.1.4. แผนภาพลำดับ

แผนภาพลำดับของฟังก์ชันหลักของเครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูลประกอบด้วยภาพลำดับ 4 แผนภาพหลักดังต่อไปนี้

1. แผนภาพลำดับการนำเข้าไฟล์ข้อมูล

สำหรับแผนภาพลำดับการนำเข้าไฟล์ข้อมูลเป็นการนำเข้าไฟล์ข้อมูลตั้งต้นที่ต้องการตรวจสอบ 4 ไฟล์ได้แก่ ไฟล์กำหนดค่า ไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์และไฟล์กรณีทดสอบ เพื่อนำไปวิเคราะห์หาผลกระทบที่เกิดขึ้นต่อซอร์สโค้ดและกรณีทดสอบ โดยเริ่มจากผู้ใช้นำเข้าไฟล์ข้อมูล 4 ไฟล์ผ่านส่วนต่อประสานในคลาส newProjecView ข้อความจะถูกส่งต่อไปยังคลาสควบคุม newProjectServlet เพื่อใช้งานการดำเนินการ newProjectServlet() หลังจากนั้นเครื่องมือจะส่งข้อความไปยังคลาส DBUtils เพื่อเรียกใช้การดำเนินการ insertFile() ในการจัดเก็บไฟล์ข้อมูลลงไปในฐานข้อมูลของเครื่องมือ ในส่วนของการสกัดข้อมูลสคีมาฐานข้อมูลที่เกี่ยวข้องกับโปรแกรมทดสอบ จะเริ่มหลังจากจัดเก็บไฟล์เสร็จสมบูรณ์ โดยทำการอ่านไฟล์กำหนดค่าผ่านการดำเนินการ readConfigurationFile() และ getTableName() ในคลาส newProjectServlet เมื่อได้ข้อมูลการเชื่อมต่อกับฐานข้อมูลของโปรแกรมทดสอบเครื่องมือจะเชื่อมต่อไปยังฐานข้อมูลของโปรแกรมที่ใช้ทดสอบผ่านการดำเนินการ newProjectServlet() จากนั้นจะได้รับผลลัพธ์คือข้อมูลสคีมาฐานข้อมูลของตารางที่เกี่ยวข้องส่งกลับมายังคลาสควบคุม และเครื่องมือจะจัดเก็บข้อมูลลงไปในฐานข้อมูลเครื่องมือโดยเรียกใช้การดำเนินการ insertSchema() ในคลาส DBUtils ดังแสดงในรูปที่ 4-30



รูปที่ 4-30 แผนภาพลำดับการนำเข้าไฟล์ข้อมูล

2. แผนภาพลำดับการวิเคราะห์ผลกระทบต่อซอร์สโค้ด

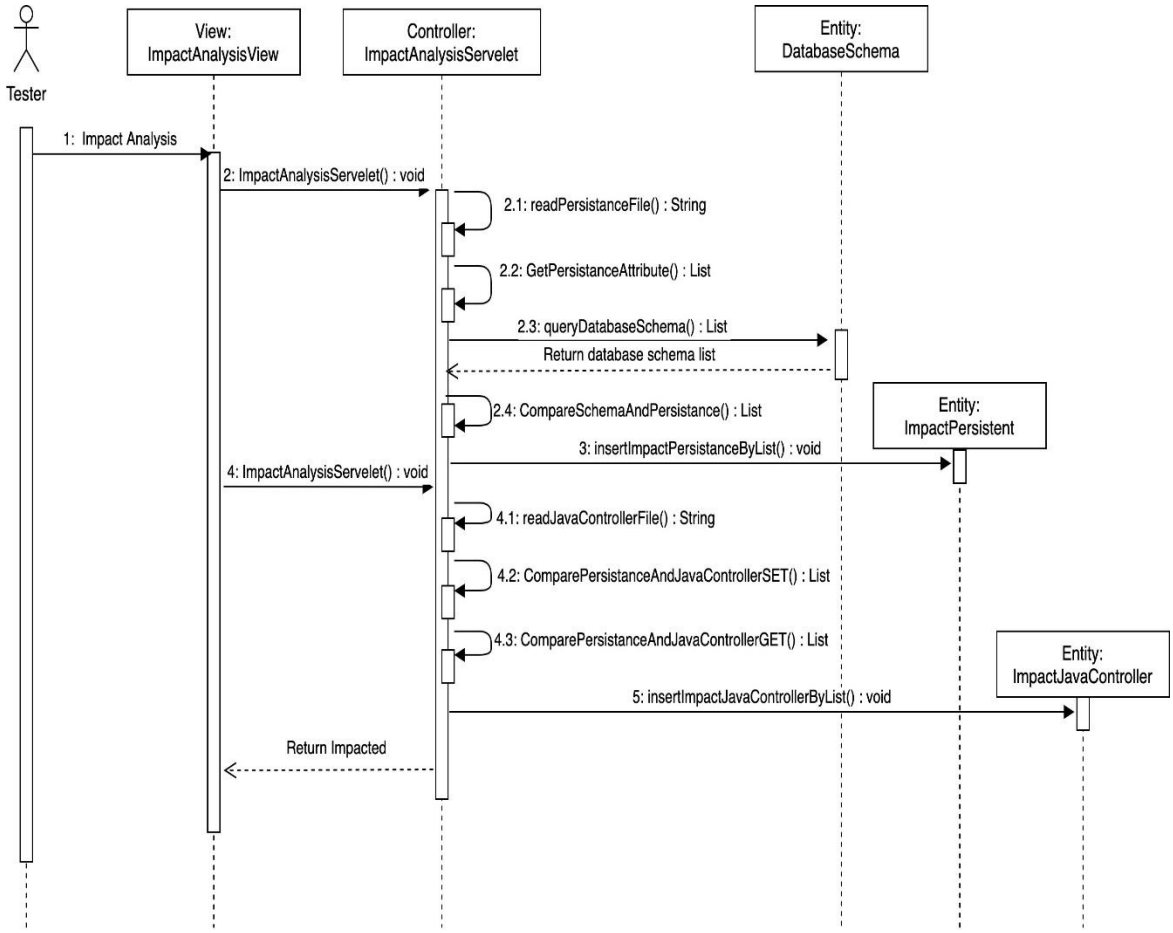
แผนภาพลำดับการวิเคราะห์ผลกระทบต่อซอร์สโค้ด เป็นแผนภาพที่แสดงถึงการวิเคราะห์ผลกระทบต่อซอร์โค้ดเพอซิสเทินและซอร์สโค้ดคอนโทรลเลอร์ โดยเริ่มจากผู้ใช้เรียกใช้การดำเนินการ ImpactAnalysisServelet() ในคลาส ImpactAnalysisServelet เพื่อวิเคราะห์หาผลกระทบในซอร์สโค้ดเพอซิสเทิน จากนั้นจะเริ่มการอ่านไฟล์ซอร์สโค้ดเพอซิสเทินและสกัดข้อมูลแอตทริบิวต์ทั้งหมดผ่านการดำเนินการ readPersistanceFile() และ GetPersistanceAttribute() เมื่อได้ข้อมูลแอตทริบิวต์ทั้งหมด เครื่องมือจะทำการดึงข้อมูลสคีมาฐานข้อมูลจากคลาส DBUtils ผ่านการดำเนินการ queryDatabaseSchema() เพื่อให้ได้ผลลัพธ์ส่งกลับมายังคลาส ImpactAnalysis-Servelet และทำการวิเคราะห์หาผลกระทบต่อซอร์สโค้ดในลำดับต่อไป หลังจากที่ทำการวิเคราะห์ผลกระทบต่อซอร์สโค้ดเพอซิสเทินเสร็จสมบูรณ์ เครื่องมือจะทำการจัดเก็บข้อมูลแอตทริบิวต์ที่ได้รับผลกระทบทั้งหมดลงในฐานข้อมูลของเครื่องมือโดยเรียกใช้ผ่านการดำเนินการ insertImpact-PersistanceByList() ในคลาส DBUtils

เมื่อทำการวิเคราะห์ผลกระทบต่อซอร์สโค้ดเพอซิสเทินแล้วเสร็จ ลำดับต่อมาเครื่องมือจะทำการวิเคราะห์หาผลกระทบต่อซอร์สโค้ดคอนโทรลเลอร์ โดยเริ่มจากการอ่านไฟล์ซอร์สโค้ดคอนโทรลเลอร์ผ่านการดำเนินการ readPersistanceFile() เพื่อสกัดชื่อข้อมูลแอตทริบิวต์และชื่อฟังก์ชัน เพื่อทำการเปรียบเทียบหาผลกระทบซอร์สโค้ดคอนโทรลเลอร์ผ่านการดำเนินการComparePersistanceAndJavaControllerSET() และComparePersistance-AndJavaControllerGET() ในคลาส ImpactAnalysisServelet ตามลำดับ หลังจากนั้นเครื่องมือจะทำการจัดเก็บข้อมูลหมายเลขบรรทัดของไฟล์ซอร์สโค้ดคอนโทรเลอร์ที่ได้รับผลกระทบทั้งหมดลงในฐานข้อมูลของเครื่องมือโดยเรียกใช้ผ่านการดำเนินการinsertImpactJavaControllerByList() ในคลาส DBUtils และส่งผลลัพธ์กลับไปยังคลาส ImpactAnalysisView เพื่อแสดงผลผ่านส่วนต่อประสานให้ผู้ใช้ได้รับทราบ แสดงดังรูปที่ 4-31

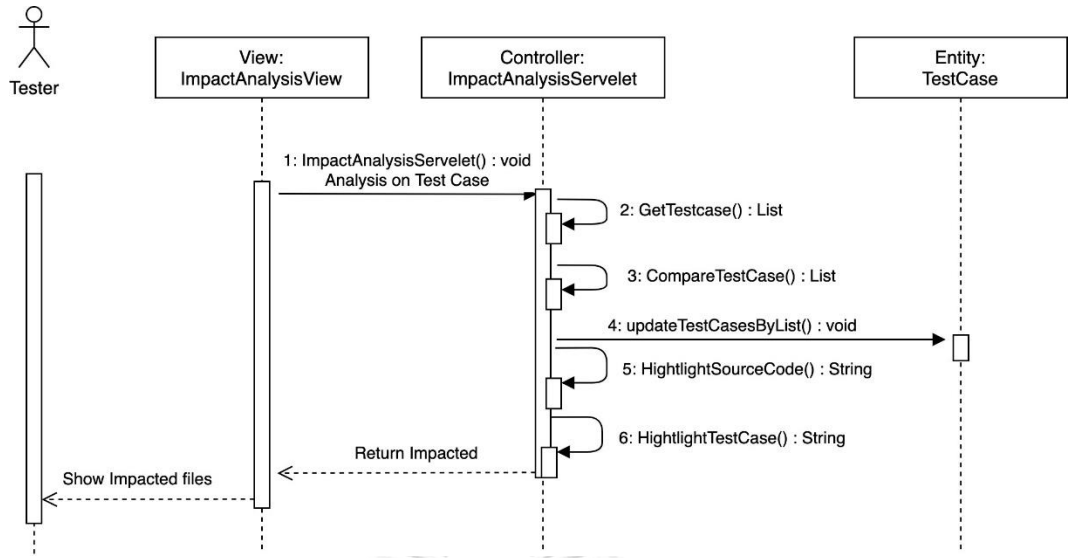
3. แผนภาพลำดับการวิเคราะห์ผลกระทบต่อกรณีทดสอบ

แผนภาพลำดับการวิเคราะห์ผลกระทบต่อกรณีทดสอบ เป็นแผนภาพที่แสดงถึงการวิเคราะห์เพื่อหาผลกระทบต่อกรณีทดสอบโดยเริ่มจากอ่านข้อมูลกรณีทดสอบจากไฟล์กรณีทดสอบผ่านการดำเนินการ GetTestcase() ในคลาส ImpactAnalysisServelet หลังจากนั้นนำผลลัพธ์ที่ได้จากการวิเคราะห์หาผลกระทบต่อซอร์สโค้ด ซึ่งก็คือหมายเลขบรรทัดของซอร์สโค้ดที่ได้รับผลกระทบมาเปรียบเทียบกับเส้นทางเดินในแต่ละกรณีทดสอบผ่านการดำเนินการ CompareTestCase() หากพบว่ามีความตรงกัน นั่นหมายถึงกรณีทดสอบนั้นได้รับผลกระทบ เครื่องมือจะทำการตรวจสอบหาผลกระทบต่อแอตทริบิวต์เป็นลำดับถัดไป เมื่อได้รายการกรณีทดสอบที่ได้รับผลกระทบทั้งหมดแล้ว

เครื่องมือจะส่งข้อความเรียกการดำเนินการ updateTestCasesByList() เพื่อบันทึกข้อมูลลงในฐานข้อมูลของเครื่องมือ และส่งข้อความเรียกการดำเนินการ HightlightSourceCode() และ HightlightTestcase() เพื่อระบายสีส่วนที่ได้รับผลกระทบและส่งผลลัพธ์กลับไปยังส่วนต่อประสาน ImpactAnalysisView ให้ผู้ใช้ได้ทราบ แสดงดังรูปที่ 4-32



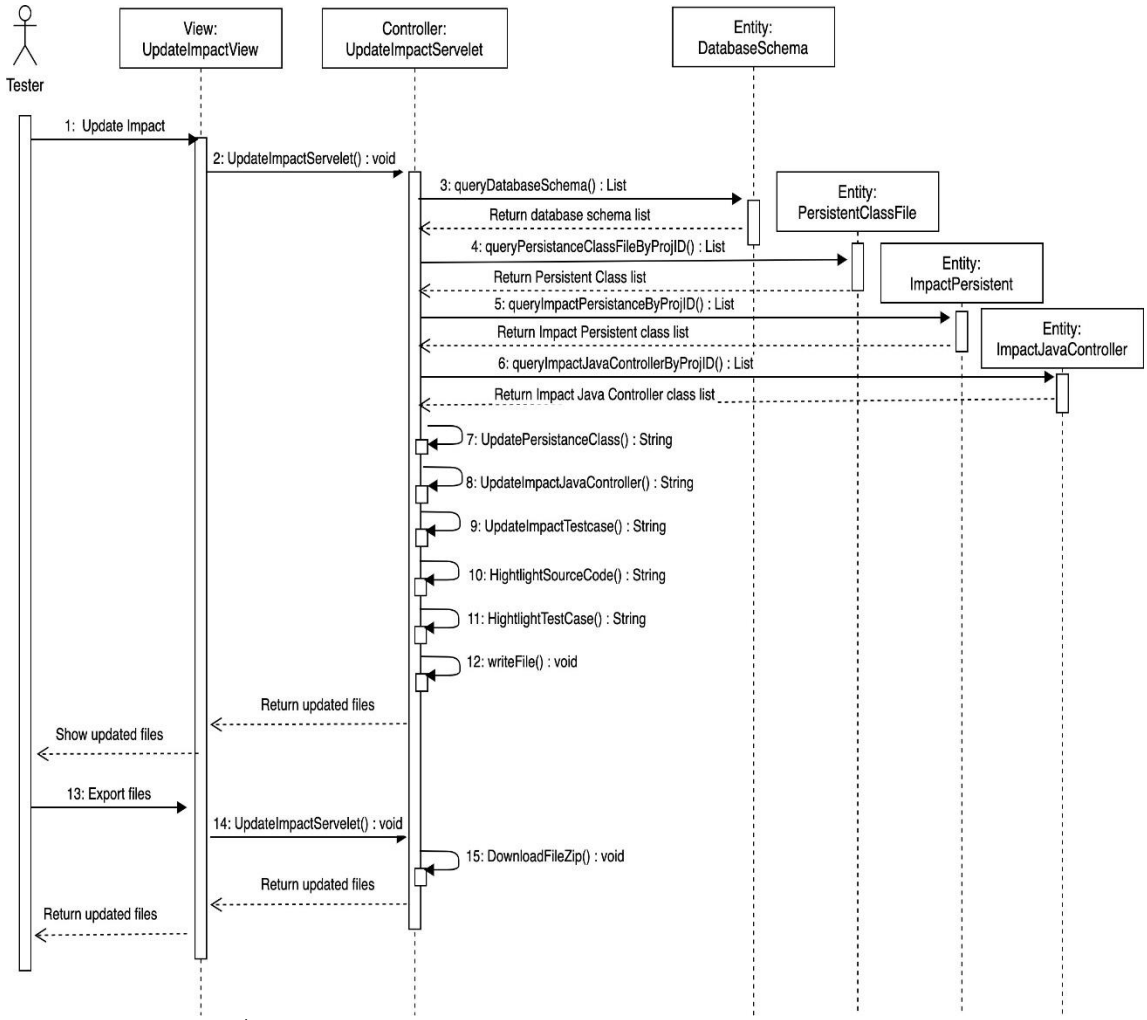
รูปที่ 4-31 แผนภาพลำดับการวิเคราะห์ผลกระทบต่อซอร์สโค้ด



รูปที่ 4-32 แผนภาพลำดับการวิเคราะห์ผลกระทบต่อกรณีทดสอบ

4. แผนภาพลำดับการแก้ไขไฟล์ซอร์สโค้ดและกรณีทดสอบ

แผนภาพลำดับการแก้ไขไฟล์ซอร์สโค้ดและกรณีทดสอบ ดังแสดงในรูปที่ 4-33 จะแสดงขั้นตอนการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดและไฟล์กรณีทดสอบที่ได้รับผลกระทบจากการเปลี่ยนแปลงสคีมาฐานข้อมูล โดยเริ่มจากผู้ใช้กดเลือกให้มีการปรับปรุงแก้ไขผ่านส่วนต่อประสาน UpdateImpactView จากนั้นเครื่องมือจะทำการส่งข้อความไปยังคลาส UpdateImpactServelet ผ่านการดำเนินการ UpdateImpactServelet() ในขั้นตอนถัดไปเครื่องมือจะทำการดึงข้อมูลสคีมาฐานข้อมูล ข้อมูลไฟล์ซอร์สโค้ดเพอซิสเทิน ข้อมูลแอตทริบิวต์จากไฟล์ซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบ และข้อมูลแอตทริบิวต์และหมายแลขบรรทัดจากไฟล์ซอร์สโค้ดคอนโทรลเลอร์ที่ได้รับผลกระทบผ่านการดำเนินการ queryDatabaseSchema(), queryPersistanceClass-FileByProjID(), queryImpactPersistanceByProjID(), queryImpact-JavaControllerByProjID() ตามลำดับ เมื่อได้ผลลัพธ์ส่งกลับมายังคลาส UpdateImpactServelet เครื่องมือจะทำการปรับปรุงแก้ไขซอร์สโค้ดเพอซิสเทินก่อนเป็นอันดับแรก ต่อด้วยซอร์สโค้ดคอนโทรลเลอร์ และกรณีทดสอบตามลำดับผ่านการดำเนินการ UpdatePersistanceClass() UpdateImpactJavaController() และ UpdateImpactTestcase() ในขั้นตอนต่อไปเครื่องมือจะทำการส่งระบายสีข้อมูลที่ได้ทำการปรับปรุงแก้ไขผ่านการดำเนินการ HightlightSourceCode() และ HightlightTestCase() จากนั้นก็ทำการเขียนไฟล์ที่ปรับปรุงเสร็จเรียบร้อยแล้ว ทำการบันทึกข้อมูลลงในฐานข้อมูลของเครื่องมือด้วยเวอร์ชันใหม่ผ่านการดำเนินการ writeFile() และส่งผลลัพธ์กลับไปยังส่วนต่อประสาน UpdateImpactView เพื่อแสดงผลให้ผู้ใช้ได้ทราบ หากผู้ใช้เลือกคำร้องขอนำออกไฟล์ข้อมูล เครื่องมือจะส่งข้อความผ่านการดำเนินการ DownloadFileZip() และไฟล์ทั้งหมดที่ถูกแก้ไขจะถูกนำออกจากเครื่องมือไปยังคอมพิวเตอร์ของผู้ใช้

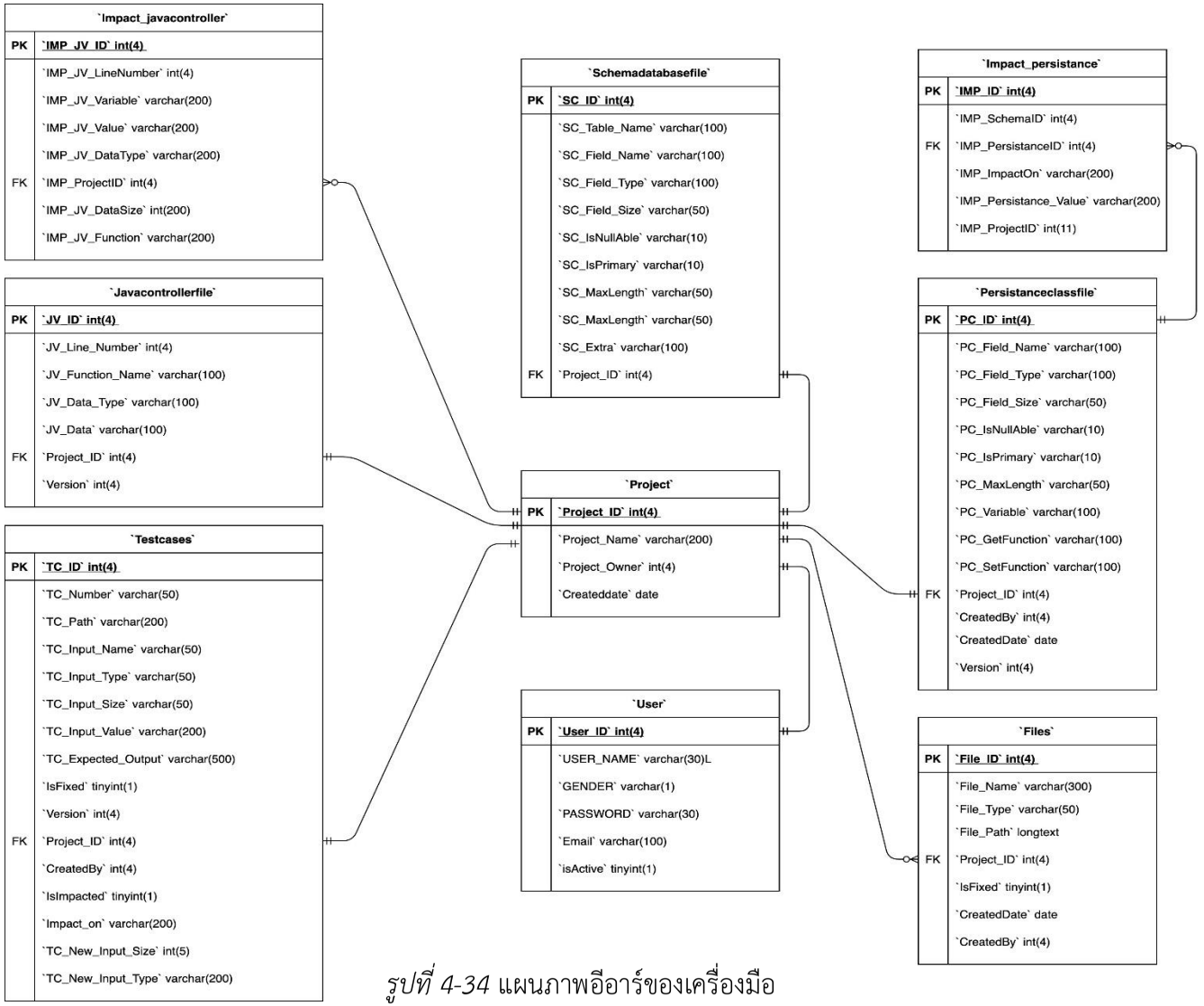


รูปที่ 4-33 แผนภาพลำดับการแก้ไขไฟล์ซอร์สโค้ดและกรณีทดสอบ

4.1.5. โครงสร้างฐานข้อมูล

ผู้วิจัยได้นำแผนภาพคลาสตามที่ได้อธิบายไว้ก่อนหน้านี้แล้ว มาวิเคราะห์และออกแบบเป็นโครงสร้างของฐานข้อมูล แสดงด้วยแผนภาพอีอาร์ดังรูปที่ 4-34 ซึ่งมีรายละเอียดดังต่อไปนี้

1. เอนทิตี UserAccount เป็นเอนทิตี ที่ใช้สำหรับเก็บข้อมูลผู้ใช้ในเครื่องมือ
2. เอนทิตี Project เป็นเอนทิตีที่ใช้สำหรับเก็บข้อมูลโครงการ
3. เอนทิตี Files เป็นเอนทิตีที่ใช้สำหรับเก็บข้อมูลไฟล์นำเข้าและนำออกทั้งหมดในเครื่องมือ
4. เอนทิตี DatabaseSchema เป็นเอนทิตีที่ใช้สำหรับเก็บข้อมูลสคีมาฐานข้อมูลของโปรแกรมทดสอบ
5. เอนทิตี PersistentClassFile เป็นเอนทิตีที่ใช้สำหรับเก็บสคีมาของข้อมูลจากไฟล์ซอร์สโค้ดเพอซิสเทิน
6. เอนทิตี JavaControllerFile เป็นเอนทิตีที่ใช้สำหรับเก็บสคีมาของข้อมูลจากไฟล์ซอร์สโค้ดคอนโทรลเลอร์
7. เอนทิตี TestCase เป็นเอนทิตีที่ใช้สำหรับเก็บข้อมูลกรณีทดสอบ
8. เอนทิตี ImpactPersistent เป็นเอนทิตีที่ใช้สำหรับเก็บข้อมูลไฟล์ซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบจากการเปลี่ยนแปลงสคีมาฐานข้อมูล
9. เอนทิตี ImpactJavaController เป็นเอนทิตีที่ใช้สำหรับเก็บข้อมูลไฟล์ซอร์สโค้ดคอนโทรลเลอร์ที่ได้รับผลกระทบจากการเปลี่ยนแปลงสคีมาฐานข้อมูล



รูปที่ 4-34 แผนภาพอีอาร์ของเครื่องมือ

4.2. สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ

สภาพแวดล้อมสภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือมีรายละเอียดดังต่อไปนี้

4.2.1. ฮาร์ดแวร์ (Hardware)

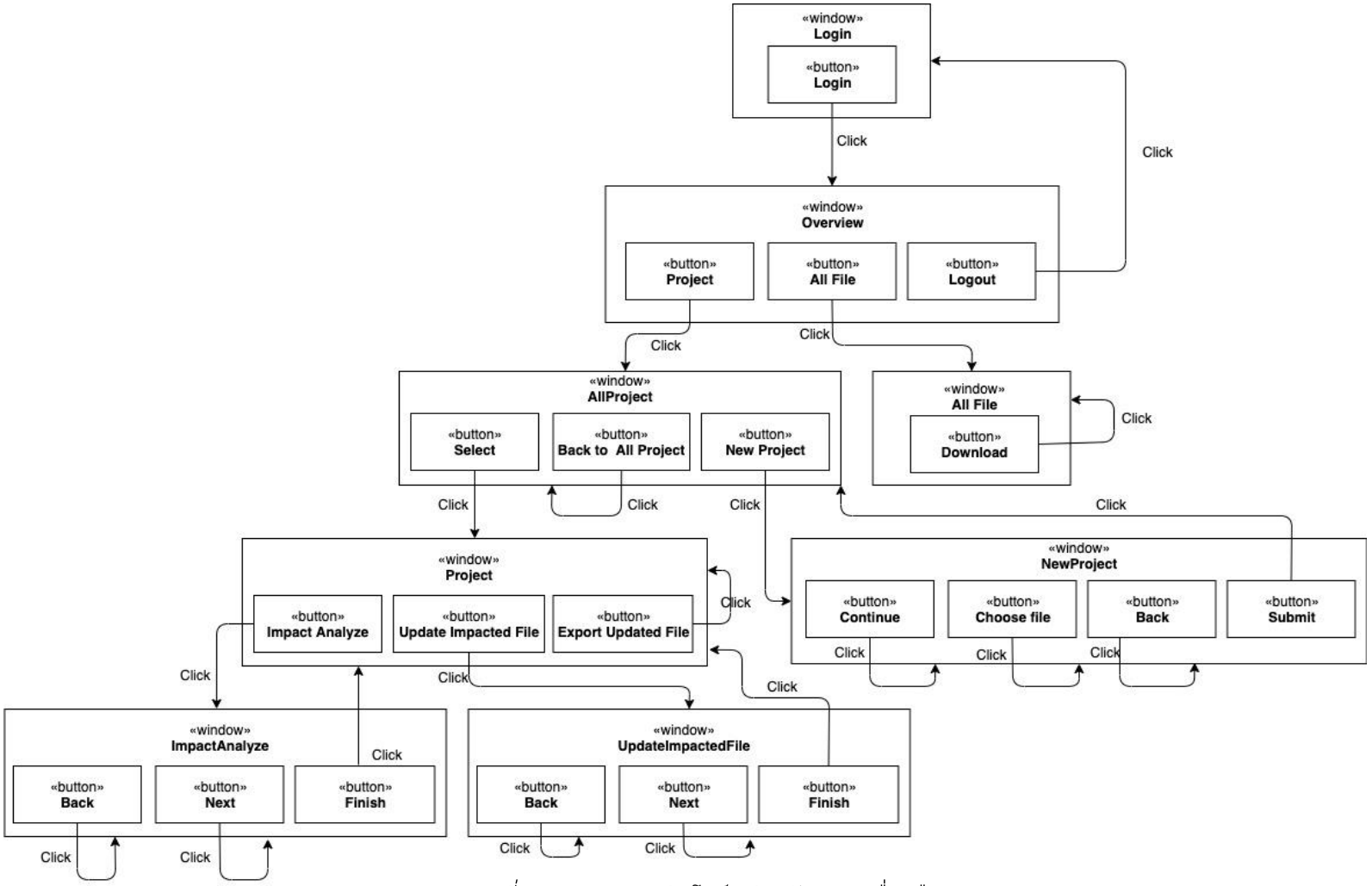
1. เครื่องคอมพิวเตอร์แบบพกพา หน่วยประมวลผลอินเทลคอร์ไอเซเว่น 1.6 กิกะเฮิรตซ์ (1.6 GHz Intel Core i7)
2. หน่วยความจำของคอมพิวเตอร์หรือแรม (Ram) 8.0 กิกะไบต์ (8 GB)
3. ฮาร์ดดิสก์ (Hard disk) 1 เทระไบต์ (1 TB)

4.2.2. ซอฟต์แวร์ (Software)

1. ระบบปฏิบัติการ (Operating System) แม็คโอเอส (macOS)
2. ภาษาที่ใช้พัฒนาคือภาษาจาวา(Java)เวอร์ชัน 8 ขึ้นไป
3. เครื่องมือที่ใช้พัฒนาอิคลิปส์เจทูอีอี เวอร์ชัน 4.1 (Eclipse J2EE)
4. เฟรมเวิร์คที่ใช้พัฒนาคือ สปริงบูต (Spring Boot)
5. ฐานข้อมูลเอสคิวแอล เซิฟเวอร์ เวอร์ชัน 2016 (Microsoft SQL Server 2016)
6. เว็บเบราว์เซอร์กูเกิลโครม (Google Chrome) เวอร์ชัน 74 ขึ้นไป

4.3. โครงสร้างส่วนต่อประสานกับผู้ใช้ของเครื่องมือ

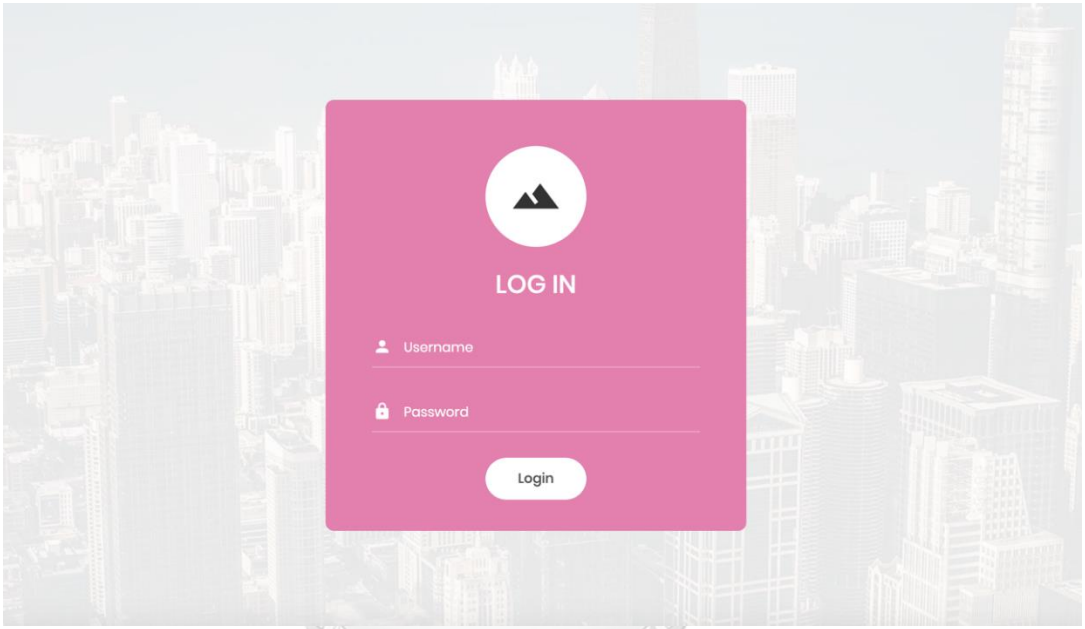
โครงสร้างของส่วนต่อประสานกับผู้ใช้ถูกอธิบายด้วยแผนภาพวินโดว์เนวิเกชัน (Window Navigation Diagram) ซึ่งได้อธิบายส่วนต่อประสานทั้งหมดของเครื่องมือ และความสัมพันธ์ระหว่างส่วนต่อประสานด้วยกัน แสดงดังรูปที่ 4-35



รูปที่ 4-35 แผนภาพวินโดว์เนวิเกชันของเครื่องมือ

แผนภาพวินโดว์เนวิเกชันที่ใช้อธิบายส่วนประกอบของส่วนต่อประสานกับผู้ใช้ของเครื่องมือสนับสนุน ซึ่งจะประกอบไปด้วย หน้าต่าง เมนู โดยที่แต่ละส่วนประกอบมีรายละเอียดดังต่อไปนี้

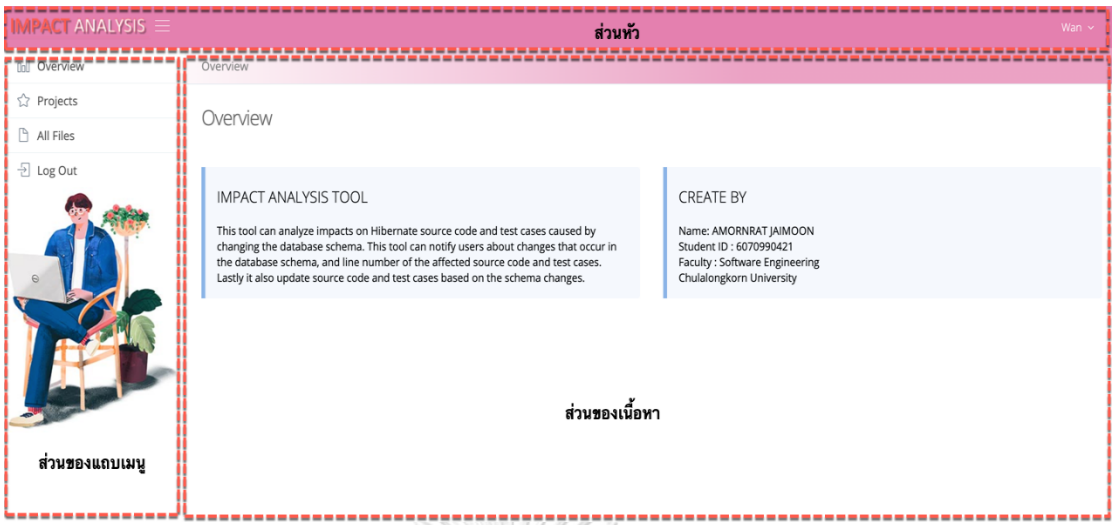
4.2.1. หน้าต่างลงชื่อเข้าใช้งาน เป็นหน้าต่างเริ่มต้นของเครื่องมือเพื่อเข้าใช้งาน โดยมีส่วนประกอบดังต่อไปนี้ กล่องข้อความสำหรับกรอกชื่อผู้ใช้งาน กล่องข้อความสำหรับกรอกรหัสผ่านของผู้ใช้งานและปุ่ม “Login” เพื่อเข้าใช้งานเครื่องมือ แสดงดังรูปที่ 4-36



รูปที่ 4-36 หน้าต่างลงชื่อเข้าใช้งาน

4.2.2. หน้าต่างหลัก เป็นหน้าต่างแรกที่ปรากฏขึ้นมาหลังจากที่ผู้ใช้ลงชื่อเข้าใช้งานสำเร็จ แสดงดังรูปที่ 4-37 ซึ่งโครงสร้างของหน้าต่างประกอบไปด้วย 3 องค์ประกอบหลัก ๆ ดังนี้

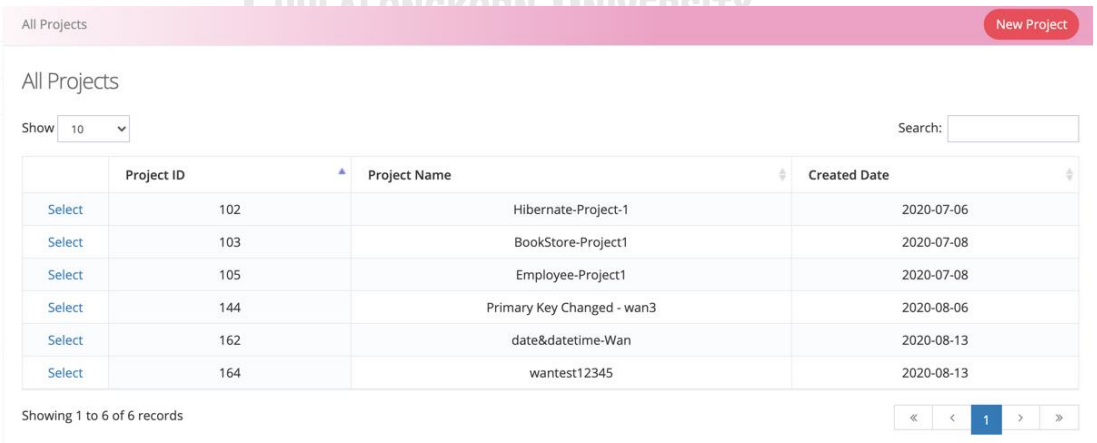
1. ส่วนหัว (Header) เป็นส่วนที่อยู่บนสุดของหน้าต่าง ประกอบด้วย ชื่อเครื่องมือ และชื่อของผู้ใช้งาน
2. ส่วนของแถบเมนู (Menu) ประกอบด้วยเมนูการใช้งานต่างๆ ดังนี้ เมนูภาพรวมของเครื่องมือ เมนูโครงการ เมนูรายการของไฟล์ และเมนูออกจากระบบ
3. ส่วนของเนื้อหาหลัก (Main Content) เป็นส่วนที่แสดงเนื้อหาของเว็บไซต์ โดยเนื้อหาของหน้าแรกจะแสดงข้อมูลของโปรแกรม และชื่อผู้พัฒนาโปรแกรม



รูปที่ 4-37 หน้าต่างหลัก

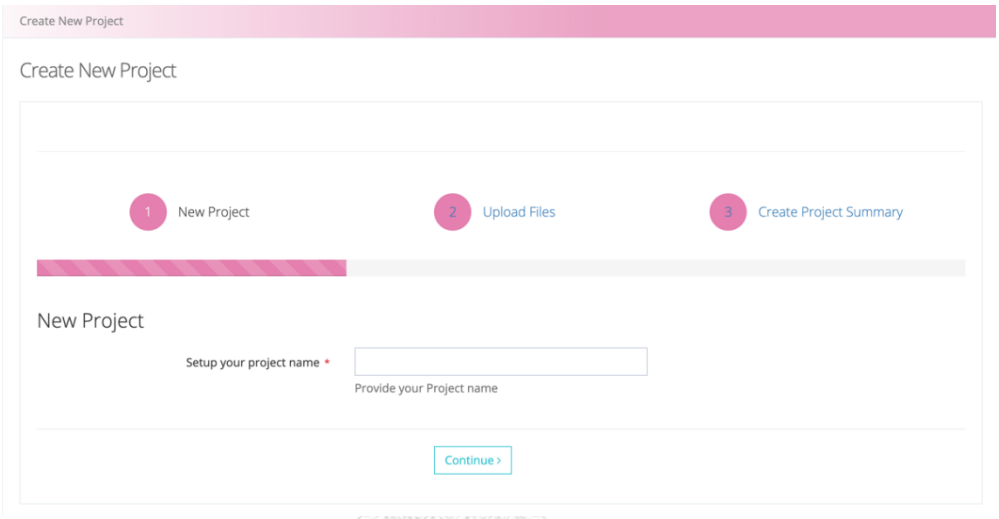
4.2.3. หน้าต่างโครงการ แสดงดังรูปที่ 4-38 เป็นหน้าต่างที่แสดงรายการของโครงการที่ผู้ใช้ได้สร้างขึ้นมาเรียบร้อยแล้วในฐานข้อมูลของเครื่องมือ โดยสามารถค้นหาโครงการจากการระบุเงื่อนไขการค้นหาได้ดังนี้ รหัสโครงการ ชื่อโครงการ และวันที่สร้างโครงการ เมื่อทำการระบุเงื่อนไขการค้นหาในกล่องข้อความ “Search” เครื่องมือจะทำการค้นหาและดึงข้อมูลของโครงการที่ตรงกันมาแสดงในส่วนของพื้นที่แสดงผลลัพธ์ด้านล่างโดยอัตโนมัติ หากทำการล้างค่าเงื่อนไขการค้นหาในกล่องข้อความ เครื่องมือจะแสดงข้อมูลรายการโครงการทั้งหมดอีกครั้ง.oพื้นที่แสดงผลลัพธ์ด้านล่างสำหรับการดำเนินการในหน้าต่างนี้ มีทั้งหมด 2 การดำเนินการหลัก ดังนี้

1. การเพิ่มโครงการใหม่ สามารถทำได้โดยคลิกปุ่ม “New Project”
2. การเลือกโครงการเพื่อทำการวิเคราะห์หาผลกระทบต่อซอร์สโค้ดและกรณีทดสอบ สามารถคลิกที่ลิงค์ “Select”

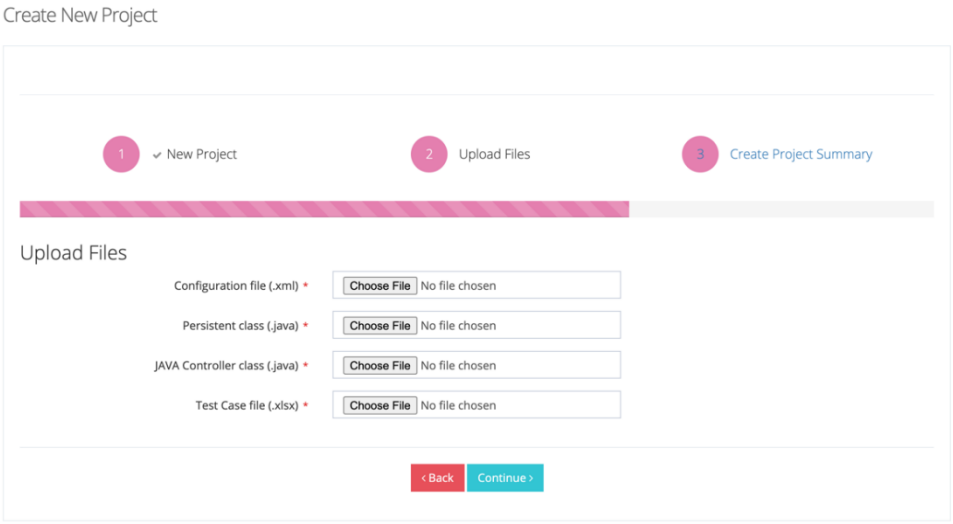


รูปที่ 4-38 หน้าต่างโครงการ

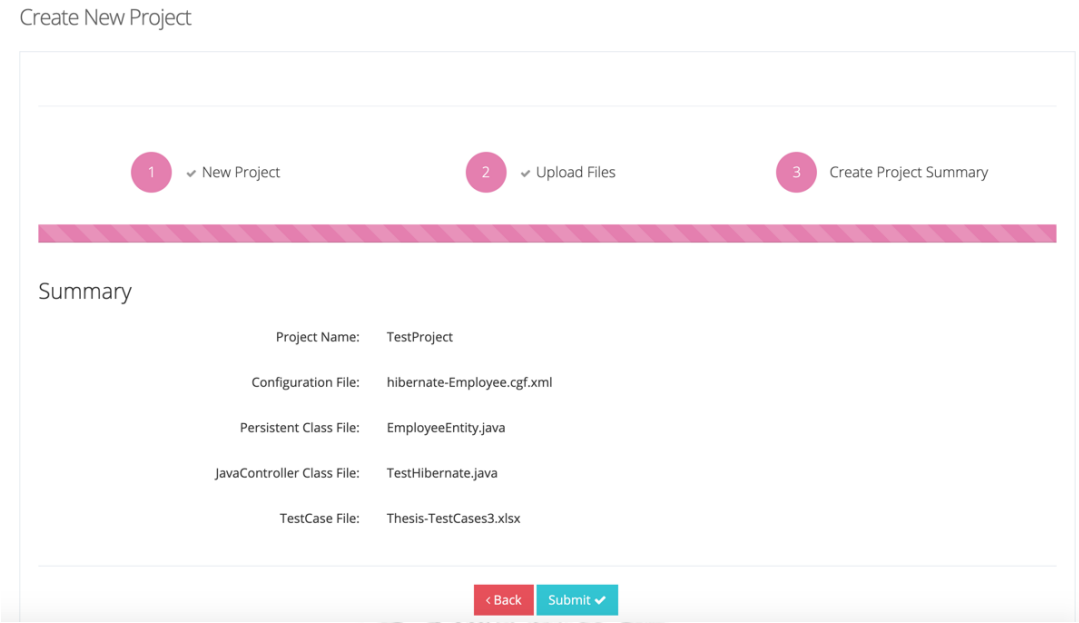
4.2.4. หน้าต่างสำหรับเพิ่มโครงการใหม่ หน้าต่างนี้จะแสดงหลังจากที่ผู้ใช้คลิกที่ปุ่ม “New Project” โดยจะแสดงหน้าต่างให้ผู้ใช้ระบุชื่อโครงการ แสดงดังรูปที่ 4-39 หลังจากผู้ใช้ระบุข้อมูลโครงการและคลิกปุ่ม “Continue” เครื่องมือจะแสดงหน้าต่างสำหรับอัพโหลดไฟล์นำเข้า 4 ไฟล์ แสดงดังรูปที่ 4-40 หลังจากนั้นหากผู้ใช้คลิกที่ปุ่ม “Back” เครื่องมือจะย้อนกลับไปยังหน้าต่างระบุชื่อโครงการ แต่หากผู้ใช้คลิกที่ปุ่ม “Continue” จะแสดงหน้าต่างสรุปข้อมูลของโครงการ แสดงดังรูปที่ 4-41 เพื่อให้ผู้ใช้ได้ตรวจสอบข้อมูลที่ระบุไปทั้งหมด เมื่อผู้ใช้กดปุ่ม “Submit” เครื่องมือจะบันทึกข้อมูลโครงการพร้อมไฟล์ข้อมูลนำเข้าทั้ง 4 ไฟล์เข้าสู่ฐานข้อมูลของเครื่องมือ



รูปที่ 4-39 หน้าต่างเพิ่มโครงการใหม่สำหรับระบุชื่อโครงการ



รูปที่ 4-40 หน้าต่างเพิ่มโครงการใหม่สำหรับระบุไฟล์นำเข้า



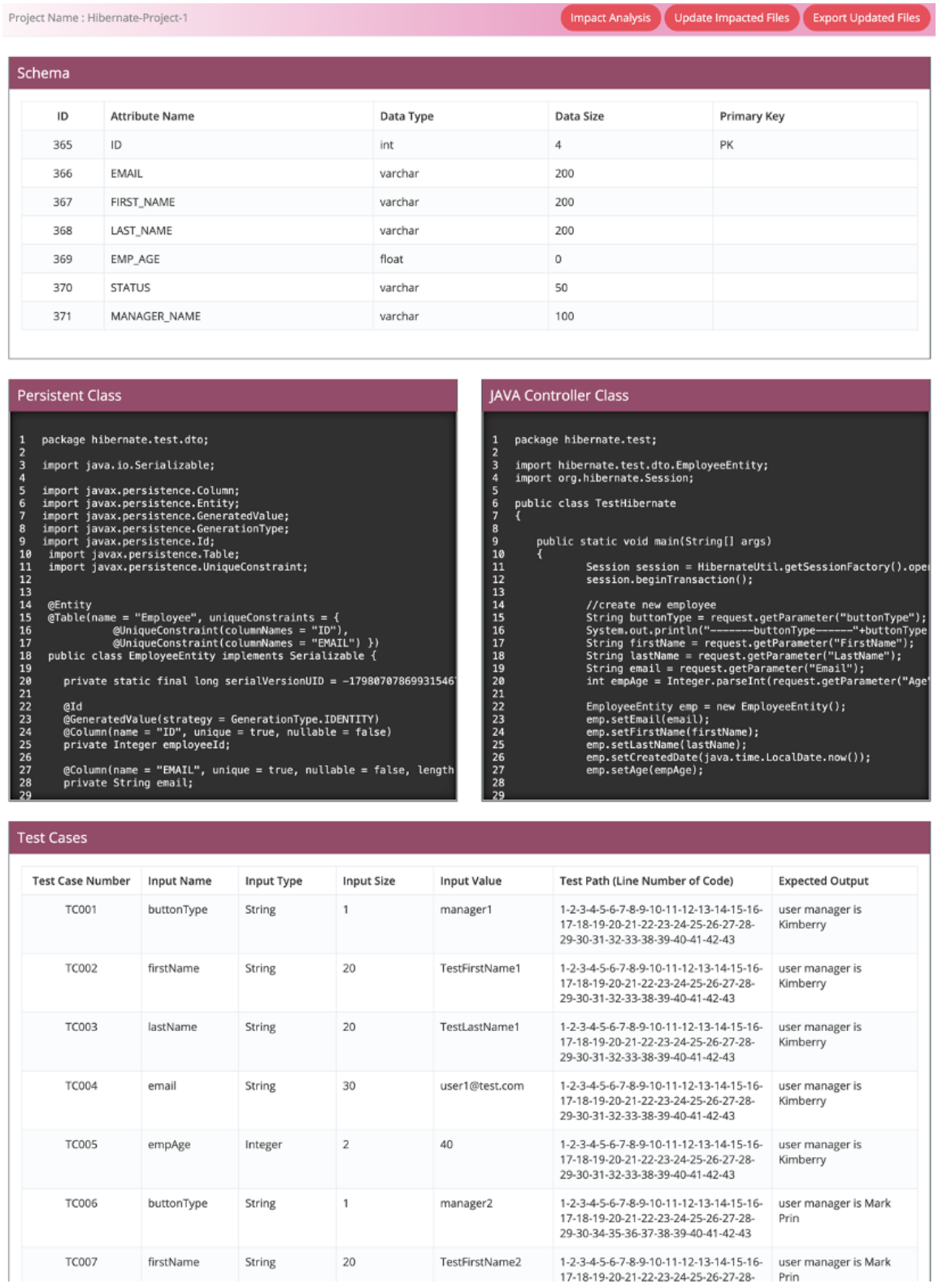
รูปที่ 4-41 หน้าต่างเพิ่มโครงการใหม่สำหรับนำส่งข้อมูลโครงการ

4.2.5. หน้าต่างแสดงรายละเอียดโครงการ แสดงดังรูปที่ 4-42 เป็นหน้าต่างที่แสดงข้อมูลทั้งหมดของโครงการ เช่น ชื่อโครงการ ข้อมูลสคีมาฐานข้อมูล ข้อมูลซอร์สโค้ดเพอซิสเทิน ข้อมูลซอร์สโค้ดคอนโทรลเลอร์ และข้อมูลกรณีทดสอบ โดยหน้าต่างนี้มีฟังก์ชันการใช้งานหลักทั้งหมด 3 ฟังก์ชัน คือ การวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ การปรับปรุงแก้ไขซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ และการนำออกไฟล์ที่ได้รับการปรับปรุงแก้ไข

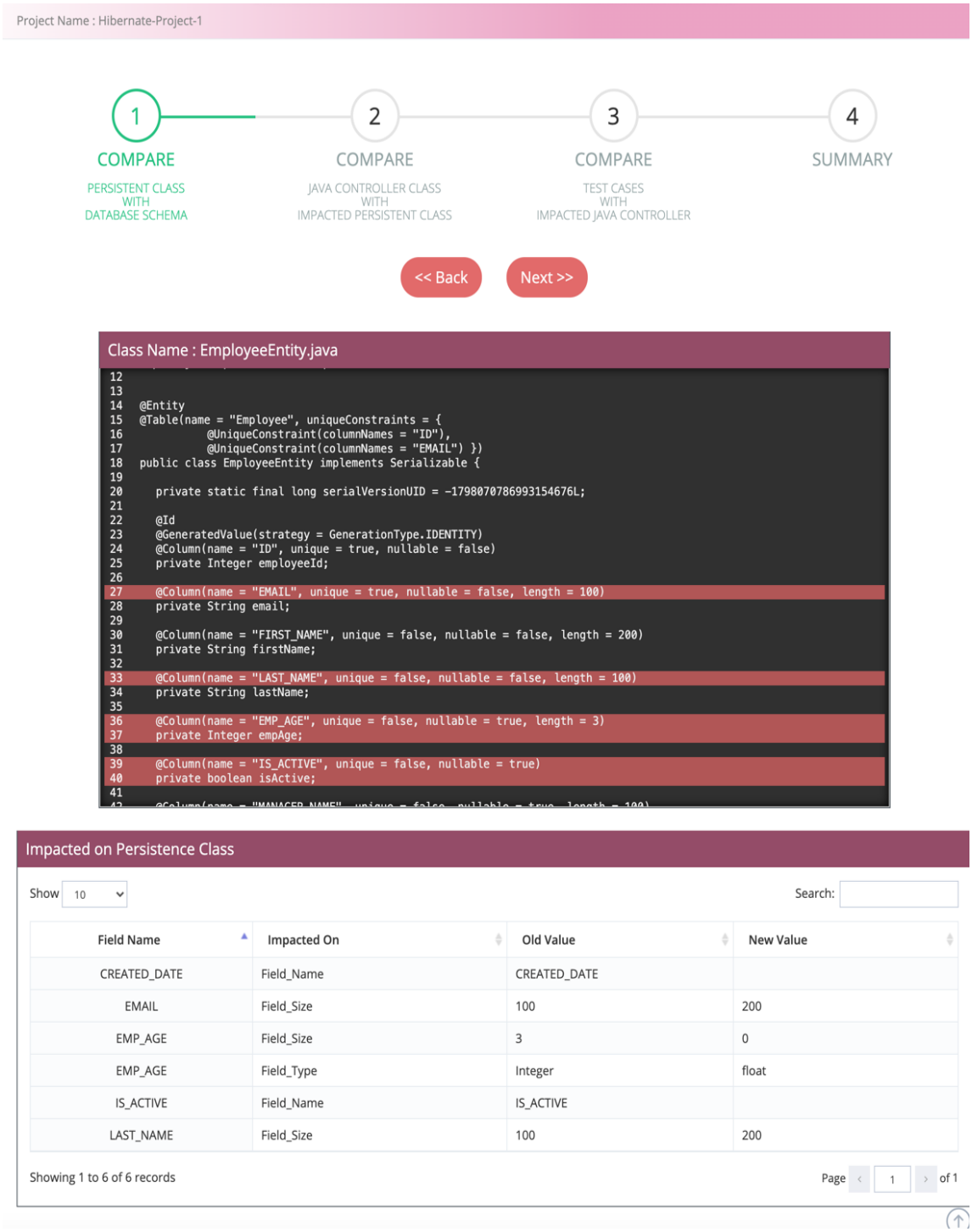
4.2.6. หน้าต่างวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ โดยหน้าต่างนี้จะแสดงหลังจากที่ผู้ใช้คลิกปุ่ม “Impact Analysis” เพื่อทำการวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ ซึ่งฟังก์ชันการวิเคราะห์ผลกระทบของหน้าต่างนี้จะแบ่งออกเป็น 4 ฟังก์ชันการทำงาน ดังรายละเอียดต่อไปนี้

1. การวิเคราะห์ผลกระทบต่อซอร์สโค้ดเพอซิสเทิน โดยหลังจากที่ผู้ใช้คลิกปุ่ม “Impact Analysis” เครื่องมือจะทำการวิเคราะห์ผลกระทบระหว่างสคีมาฐานข้อมูลกับซอร์สโค้ดเพอซิสเทินก่อนเป็นอันดับแรก หากพบว่ามีผลกระทบเกิดขึ้น เครื่องมือจะทำการไฮไลท์สีแดงบนซอร์สโค้ดเพื่อแสดงให้ผู้ใช้ได้ทราบถึงตำแหน่งที่ได้รับผลกระทบในซอร์สโค้ดเพอซิสเทิน พร้อมกับแสดงตารางรายละเอียดของข้อมูลที่ไม่ตรงกันระหว่างข้อมูลในสคีมาฐานข้อมูลและข้อมูลในซอร์สโค้ดเพอซิสเทิน ดังแสดงในรูปที่ 4-43 แต่หากไม่พบว่ามีผลกระทบเกิดขึ้น เครื่องมือจะแสดงซอร์สโค้ด

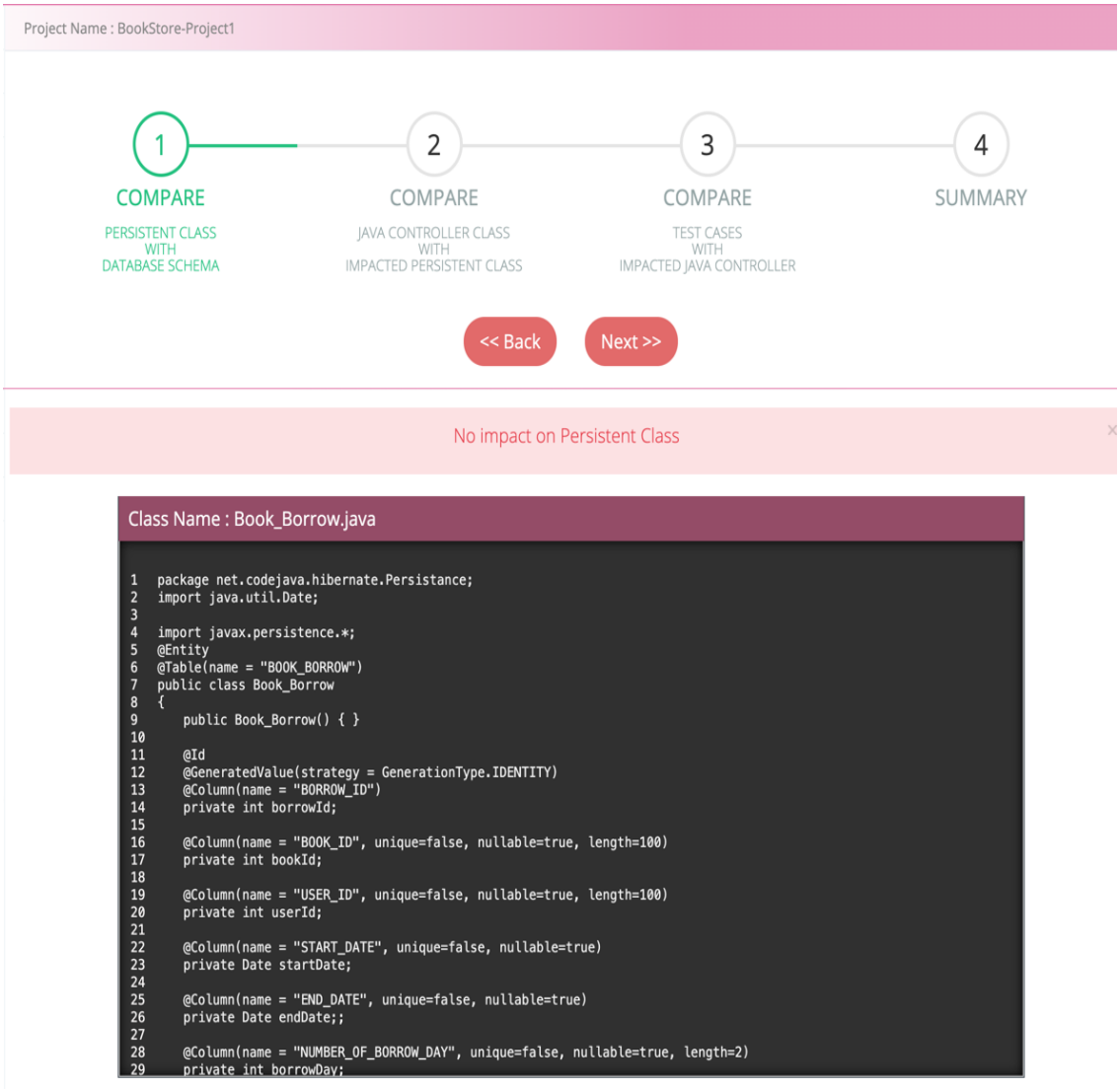
เพอซิสเทินที่ไม่มีไฮไลท์สีแดง พร้อมทั้งแสดงข้อความว่า “No impact on Persistent Class” ดังแสดงในรูป 4-44



รูปที่ 4-42 หน้าต่างแสดงรายละเอียดโครงการ



รูปที่ 4-43 หน้าต่างวิเคราะห์ผลกระทบต่อซอร์สโค้ดเพอซิสเทินกรณีเจอผลกระทบ



รูปที่ 4-44 หน้าต่างวิเคราะห์ผลกระทบต่อซอร์สโค้ดเพอซิสเทินกรณีไม่เจอผลกระทบ

2. การวิเคราะห์ผลกระทบต่อซอร์สโค้ดคอนโทรลเลอร์ หน้าต่างนี้จะแสดงหลังจากที่ผู้ใช้คลิกปุ่ม “Next” โดยเครื่องมือจะทำการวิเคราะห์ผลกระทบระหว่างซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบกับซอร์สโค้ดคอนโทรลเลอร์ หากพบว่ามีผลกระทบเกิดขึ้น เครื่องมือจะไฮไลท์สีแดงบนซอร์สโค้ดคอนโทรลเลอร์เพื่อแสดงให้ผู้ใช้ได้ทราบถึงตำแหน่งที่ได้รับผลกระทบในซอร์สโค้ดคอนโทรลเลอร์ พร้อมกับแสดงตารางรายละเอียดของหมายเลขบรรทัดที่ได้รับผลกระทบดังแสดงในรูปที่ 4-45 แต่หากไม่พบว่ามีผลกระทบเกิดขึ้นกับซอร์สโค้ดคอนโทรลเลอร์ เครื่องมือจะแสดข้อความว่า “No impact on Java Controller” ดังแสดงในรูป 4-46

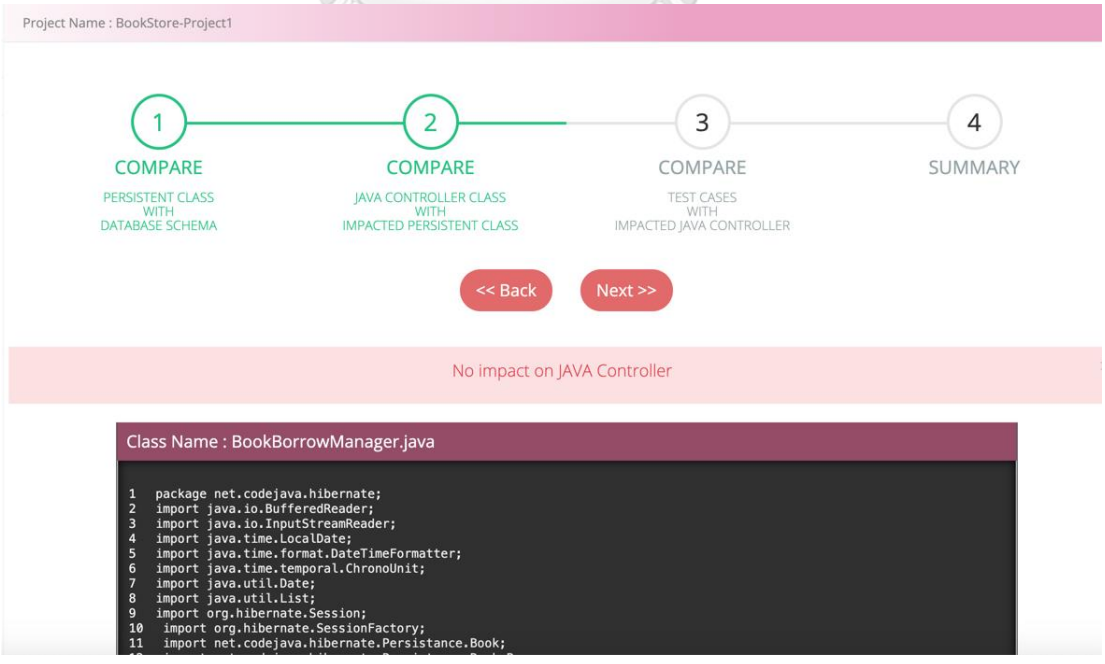
Class Name : TestHibernate.java

```
1   package hibernate.test;
2
3   import hibernate.test.dto.EmployeeEntity;
4   import org.hibernate.Session;
5
6   public class TestHibernate
7   {
8
9       public static void main(String[] args)
10      {
11              Session session = HibernateUtil.getSessionFactory().openSession();
12              session.beginTransaction();
13
14              //create new employee
15              String buttonType = request.getParameter("buttonType");
16              System.out.println("-------buttonType------"+buttonType);
17              String firstName = request.getParameter("FirstName");
18              String lastName = request.getParameter("LastName");
19              String email = request.getParameter("Email");
20              int empAge = Integer.parseInt(request.getParameter("Age"));
21
22              EmployeeEntity emp = new EmployeeEntity();
23              emp.setEmail(email);
24              emp.setFirstName(firstName);
25              emp.setLastName(lastName);
26              emp.setCreatedDate(java.time.LocalDate.now());
27              emp.setAge(empAge);
28
29
```

Impacted on JAVA Controller Class

Line Number	Variable Name	Function Name	Data Type	Data Value
20	empAge		int	Integer.parseInt(request.getParameter("Age"))
26		setCreatedDate		java.time.LocalDate.now(
27	empAge	setAge		

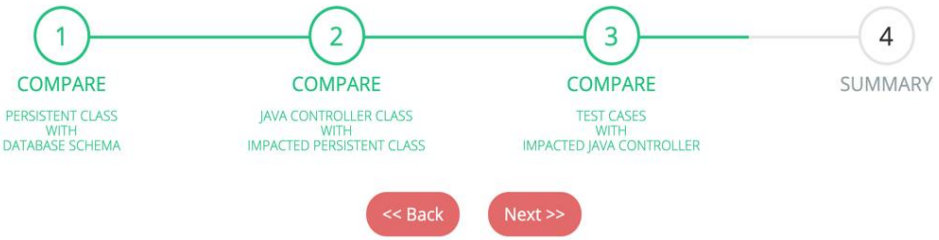
รูปที่ 4-45 หน้าต่างวิเคราะห์ผลกระทบต่อซอร์สโค้ดคอนโทรลเลอร์กรณีเจอผลกระทบ



รูปที่ 4-46 หน้าต่างวิเคราะห์ผลกระทบต่อซอร์สโค้ดคอนโทรลเลอร์กรณีไม่เจอผลกระทบ

3. การวิเคราะห์ผลกระทบต่อกรณีทดสอบ หน้าต่างนี้จะแสดงหลังจากที่ผู้ใช้คลิกปุ่ม “Next” โดยเครื่องมือจะวิเคราะห์ผลกระทบระหว่างหมายเลขบรรทัดที่ได้รับผลกระทบในซอร์สโค้ดคอนโทรลเลอร์กับหมายเลขเส้นทางเดินในกรณีทดสอบ หากพบว่ามีผลกระทบเกิดขึ้น เครื่องมือจะทำการไฮไลท์บนกรณีทดสอบเพื่อแสดงให้ผู้ใช้ได้ทราบถึงตำแหน่งที่ได้รับผลกระทบ ดังแสดงในรูปที่ 4-47 แต่หากไม่พบว่ามีผลกระทบเกิดขึ้นกับกรณีทดสอบเครื่องมือจะแสดข้อความว่า “No impact on Test Case” ดังแสดงในรูป 4-48

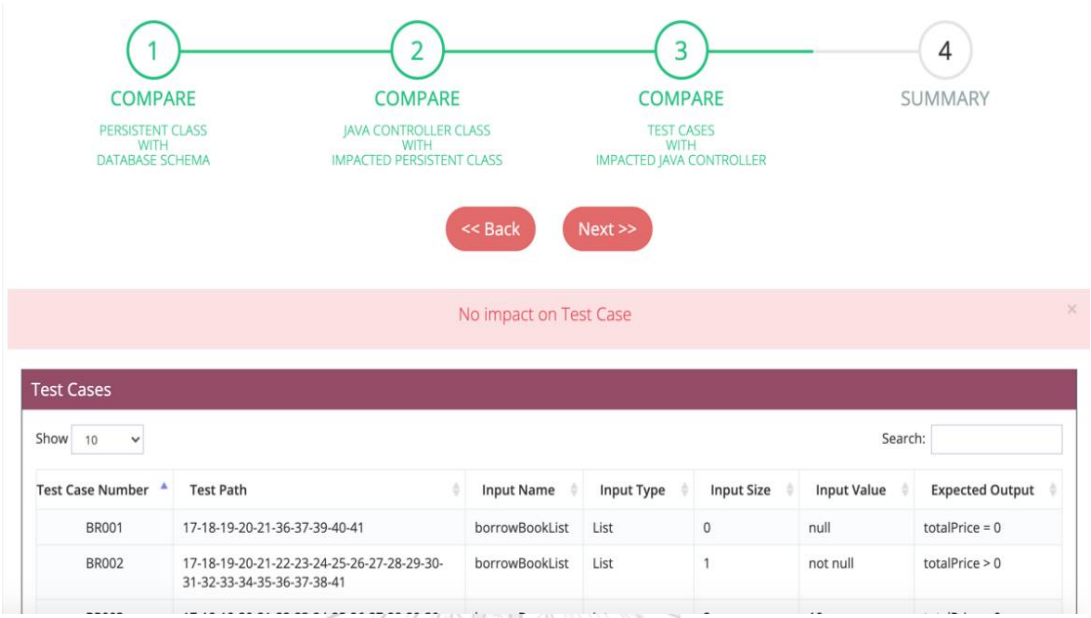
4. การสรุปผลการวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ หน้าต่างนี้จะแสดงหลังจากที่ผู้ใช้คลิกปุ่ม “Next” โดยเครื่องมือจะแสดงผลกระทบที่เกิดขึ้นทั้งหมดของ 3 ไฟล์ คือไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบ หากไฟล์ไหนได้รับผลกระทบ เครื่องมือจะไฮไลท์สีบนซอร์สโค้ดเพื่อแสดงตำแหน่งที่ได้รับผลกระทบให้ผู้ใช้ได้ทราบ ดังแสดงในรูปที่ 4-49 แต่หากไม่มีผลกระทบเกิดขึ้น เครื่องมีจะแสดงไฟลีข้อมูลทั้ง 3 โดยไม่มีการไฮไลท์สี ดังแสดงในรูปที่ 4-50 และหากผู้ใช้คลิกปุ่ม “Back” เครื่องมือจะทำการย้อนกลับไปแสดงหน้าต่างก่อนหน้า



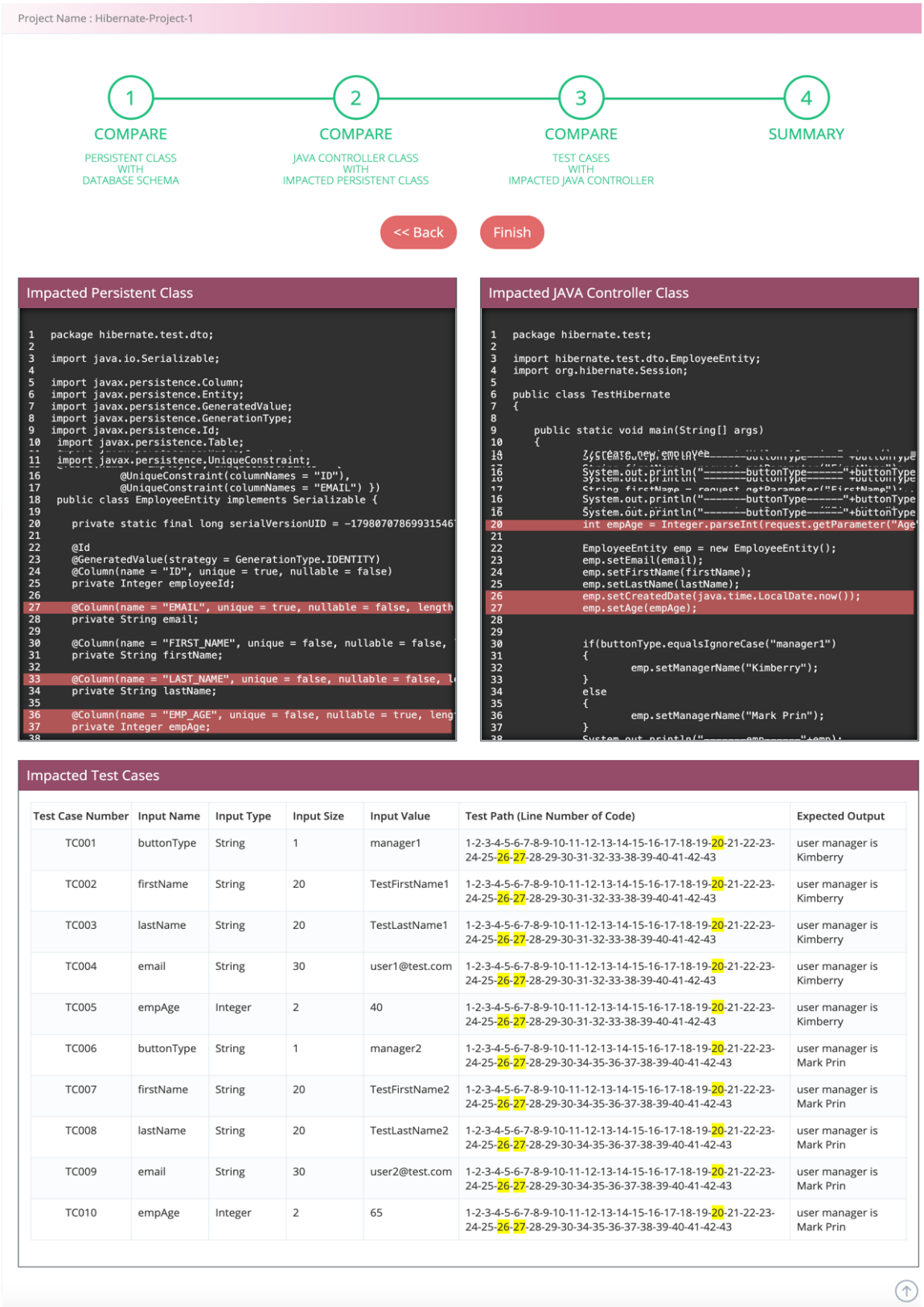
Impacted Test Cases

Test Case Number	Input Name	Input Type	Input Size	Input Value	Test Path (Line Number of Code)	Expected Output
TC001	buttonType	String	1	manager1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC002	firstName	String	20	TestFirstName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC003	lastName	String	20	TestLastName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC004	email	String	30	user1@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC005	empAge	Integer	2	40	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC006	buttonType	String	1	manager2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC007	firstName	String	20	TestFirstName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC008	lastName	String	20	TestLastName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC009	email	String	30	user2@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC010	empAge	Integer	2	65	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin

รูปที่ 4-47 หน้าต่างวิเคราะห์ผลกระทบต่อกรณีทดสอบกรณีเจอผลกระทบ

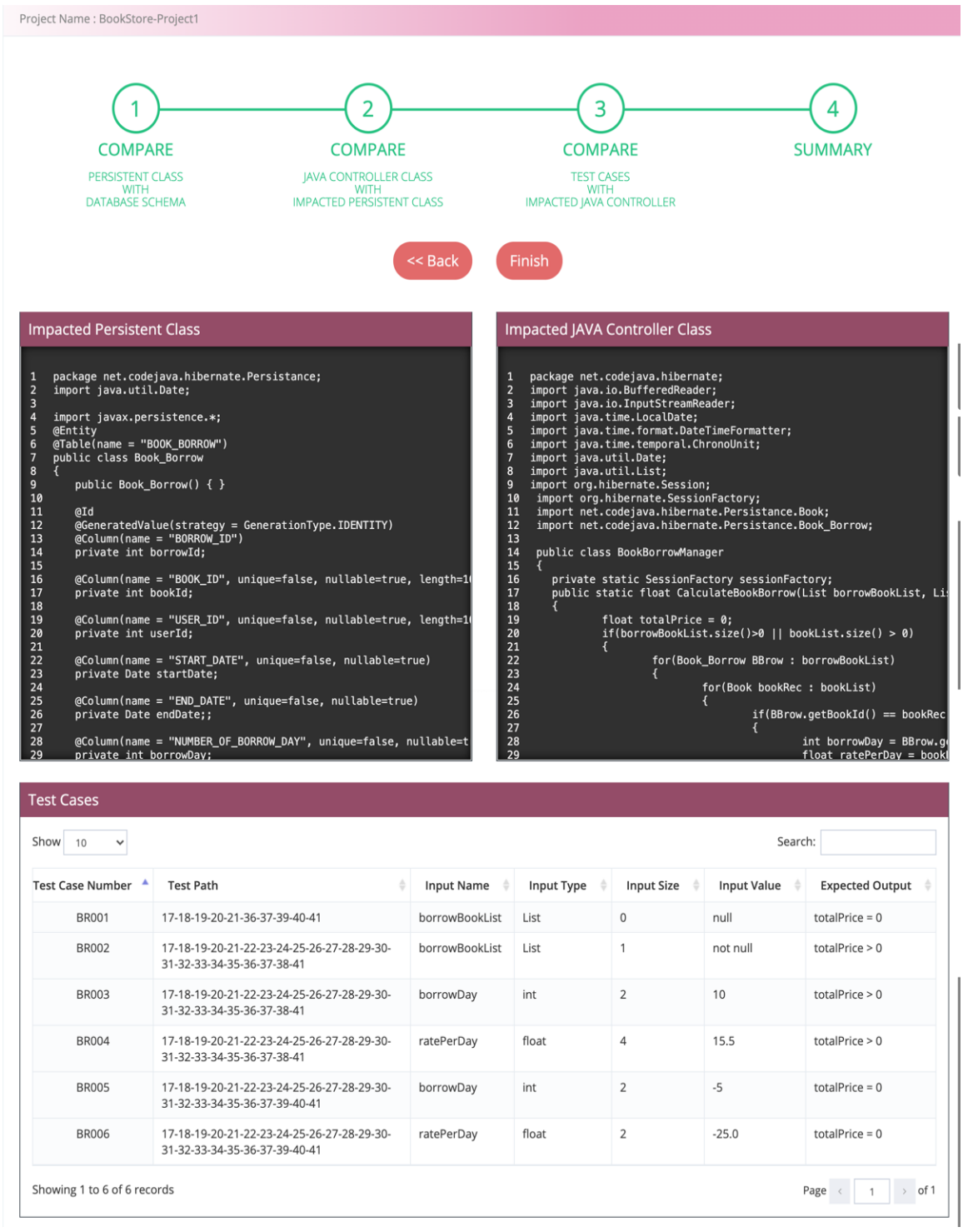


รูปที่ 4-48 หน้าต่างวิเคราะห์ผลกระทบต่อกรณีทดสอบกรณีไม่เจอผลกระทบ



Test Case Number	Input Name	Input Type	Input Size	Input Value	Test Path (Line Number of Code)	Expected Output
TC001	buttonType	String	1	manager1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC002	firstName	String	20	TestFirstName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC003	lastName	String	20	TestLastName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC004	email	String	30	user1@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC005	empAge	Integer	2	40	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC006	buttonType	String	1	manager2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC007	firstName	String	20	TestFirstName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC008	lastName	String	20	TestLastName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC009	email	String	30	user2@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC010	empAge	Integer	2	65	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin

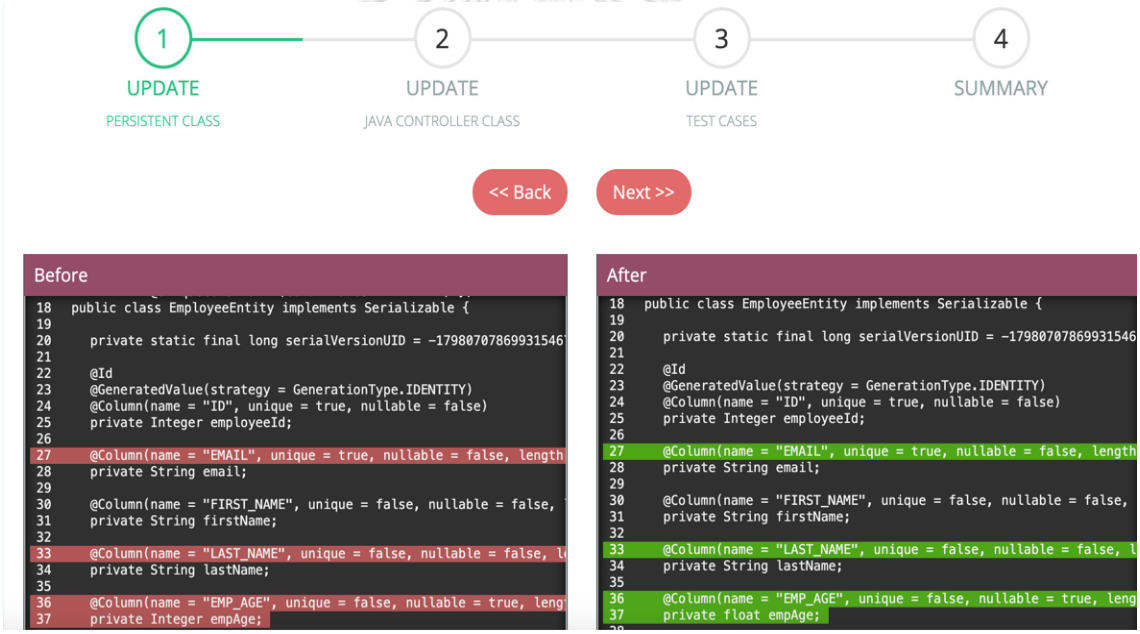
รูปที่ 4-49 หน้าต่างสรุปผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบกรณีมีผลกระทบ



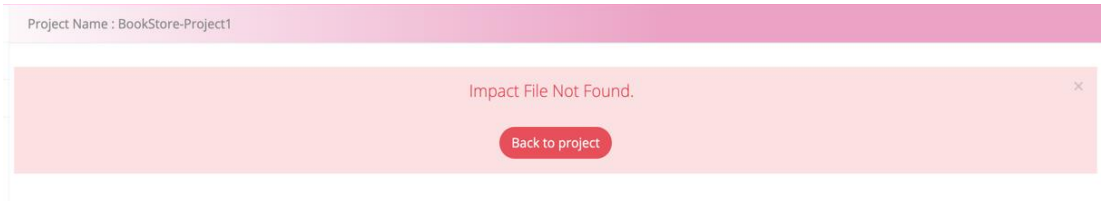
รูปที่ 4-50 หน้าต่างสรุปผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบกรณีไม่มีผลกระทบ

4.2.7. หน้าต่างปรับปรุงแก้ไขซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ โดยหน้าต่างนี้จะแสดงหลังจากที่ผู้ใช้คลิกปุ่ม “Update Impacted Files” เพื่อปรับปรุงแก้ไขไฟล์ซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบที่ได้รับผลกระทบ ซึ่งฟังก์ชันการวิเคราะห์ผลกระทบของหน้าต่างนี้จะแบ่งออกเป็น 4 ฟังก์ชันการทำงาน ดังรายละเอียดต่อไปนี้

1. การปรับปรุงซอร์สโค้ดเพอซิสเทิน เป็นการดำเนินการแรกหลังจากการวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตเสร็จเรียบร้อยแล้ว หาพบว่าซอร์สโค้ดเพอซิสเทินได้รับผลกระทบ เครื่องมือจะทำการปรับปรุงแก้ไขซอร์สโค้ดและไฮไลท์สีทั้งก่อนแก้ไข และหลังแก้ไข เพื่อแสดงให้ผู้ใช้ได้ทราบถึงตำแหน่งที่ได้ทำการแก้ไข ดังแสดงในรูปที่ 4-51 แต่หากผู้ใช้คลิกปุ่ม “Update Effected Files” โดยที่ไม่มีผลกระทบเกิดขึ้นกับไฟล์เพอซิสเทิน เครื่องมือจะแสดงข้อความ “Impact File Not Found” ดังรูปที่ 4-52 และจะมีปุ่ม “Back to project” แสดงขึ้นมาเพื่อกลับไปสู่หน้ารายละเอียดของโครงการ

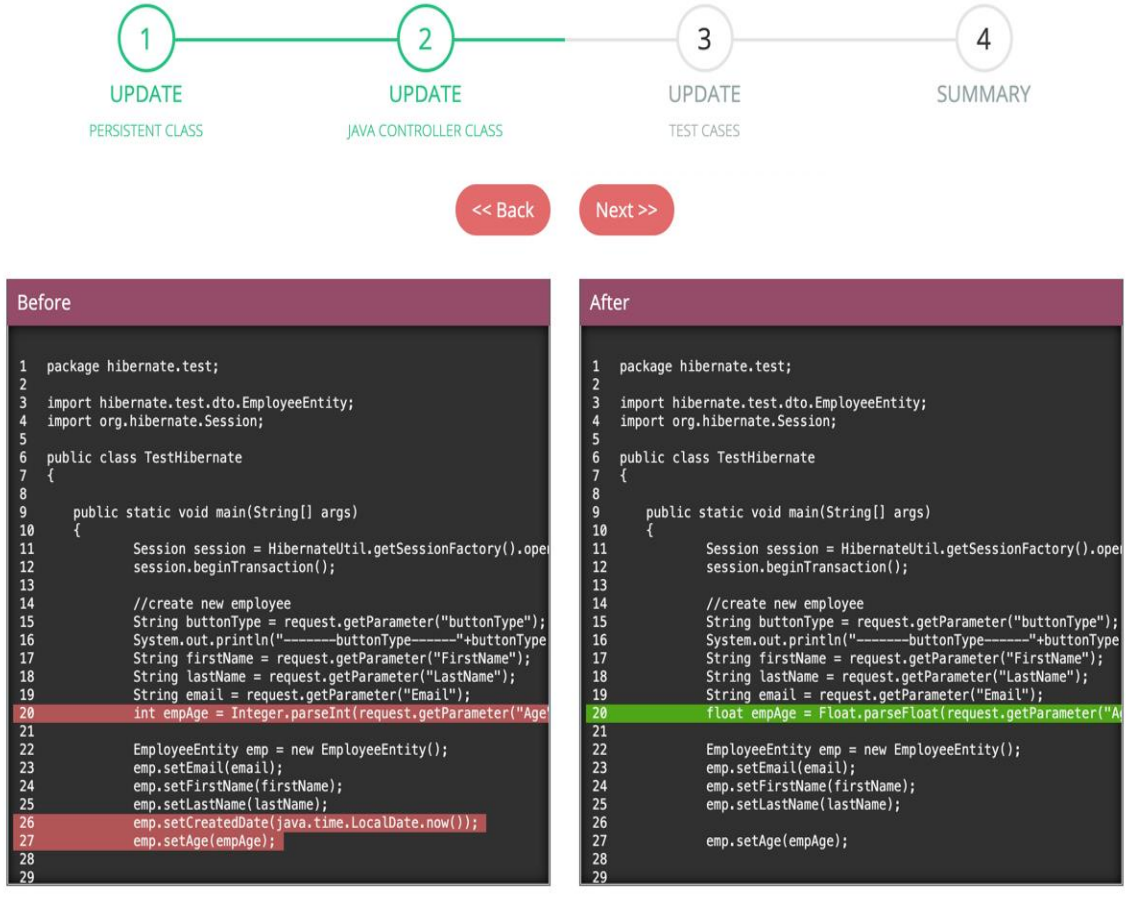


รูปที่ 4-51 หน้าต่างปรับปรุงแก้ไขซอร์สโค้ดเพอซิสเทิน



รูปที่ 4-52 หน้าต่างปรับปรุงแก้ไขซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบกรณีไม่เจอผลกระทบ

2. การปรับปรุงซอร์สโค้ดคอนโทรลเลอร์ เป็นการดำเนินการหลังจากที่ผู้ใช้คลิกปุ่ม “Next” โดยเครื่องมือจะปรับปรุงแก้ไขไฟล์ซอร์โค้ดคอนโทรลเลอร์ที่ได้รับผลกระทบ จากนั้นจะไฮไลท์สีทั้งก่อนแก้ไข และหลังแก้ไข เพื่อแสดงให้ผู้ใช้ได้ทราบถึงตำแหน่งที่ได้ทำการแก้ไข ดังแสดงในรูปที่ 4-53



รูปที่ 4-53 หน้าต่างปรับปรุงแก้ไขซอร์สโค้ดคอนโทรลเลอร์

3. การปรับปรุงกรณีทดสอบ เป็นการดำเนินการหลังจากที่ ผู้ใช้คลิกปุ่ม “Next” โดยเครื่องมือจะปรับปรุงแก้ไขกรณีทดสอบที่ได้รับผลกระทบ จากนั้นจะไฮไลท์สีทั้งก่อนแก้ไขและหลังแก้ไข เพื่อแสดงให้ผู้ใช้ได้ทราบถึงตำแหน่งที่ได้ทำการแก้ไข ดังแสดงในรูปที่ 4-54

4. การสรุปผลการปรับปรุงซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ หน้าต่างนี้จะแสดงหลังจากที่ผู้ใช้คลิกปุ่ม “Next” จากหน้าต่างก่อนหน้า โดยเครื่องมือจะแสดงไฟล์ที่ได้รับการปรับปรุงแก้ไขทั้งหมด 3 ไฟล์ คือไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบ และทำการไฮไลท์สีเพื่อแสดงตำแหน่งที่ได้รับการปรับปรุงแก้ไขให้ผู้ใช้ได้ทราบ ดังแสดงในรูปที่ 4-55 หากผู้ใช้ต้องการที่จะจบกระบวนการนี้ ผู้ใช้สามมารถคลิกที่ปุ่ม “Finish” เพื่อกลับไปยังหน้าต่างแสดงรายการโครงการ หรือคลิกที่ปุ่ม “Back” เพื่อทำการย้อนกลับไปยังหน้าต่างก่อนหน้าได้

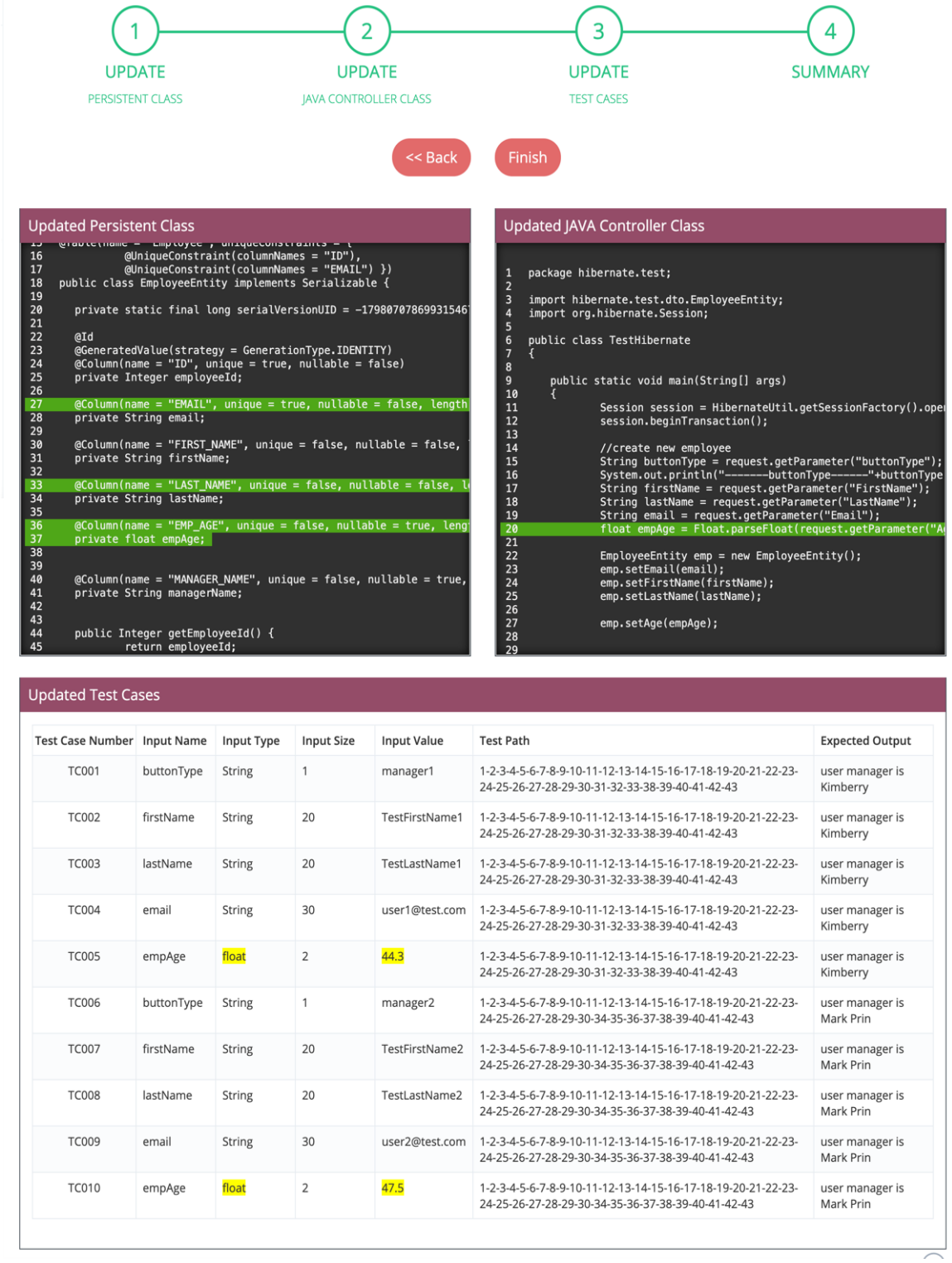
Before

Test Case Number	Input Name	Input Type	Input Size	Input Value	Test Path	Expected Output
TC001	buttonType	String	1	manager1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC002	firstName	String	20	TestFirstName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC003	lastName	String	20	TestLastName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC004	email	String	30	user1@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC005	empAge	Integer	2	40	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC006	buttonType	String	1	manager2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC007	firstName	String	20	TestFirstName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC008	lastName	String	20	TestLastName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC009	email	String	30	user2@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC010	empAge	Integer	2	65	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin

After

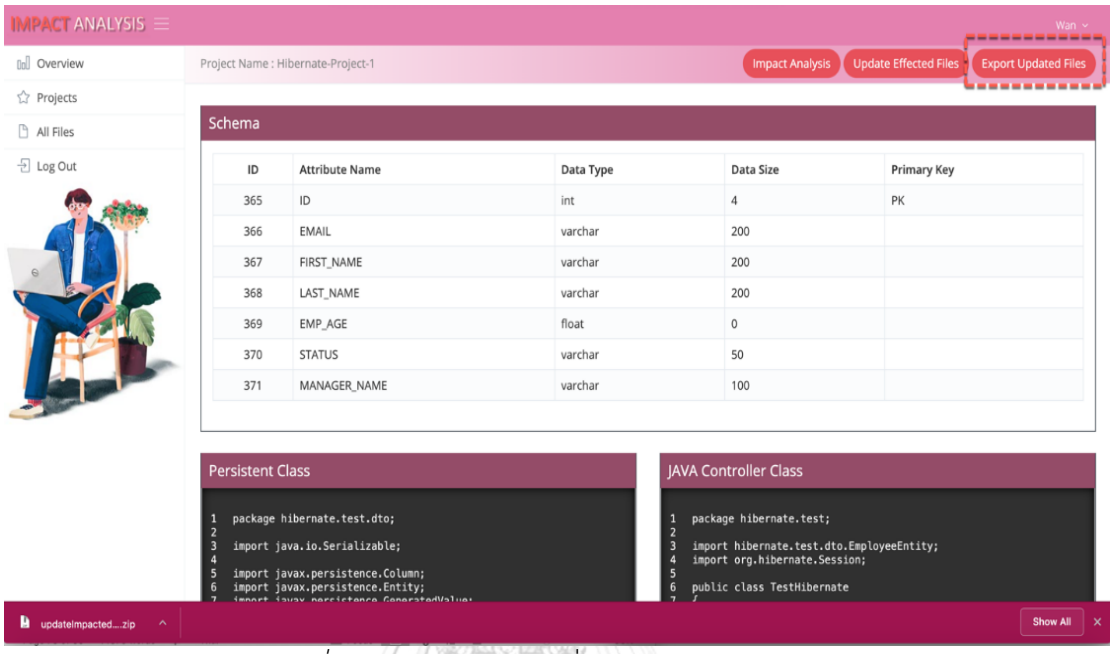
Test Case Number	Input Name	Input Type	Input Size	Input Value	Test Path	Expected Output
TC001	buttonType	String	1	manager1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC002	firstName	String	20	TestFirstName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC003	lastName	String	20	TestLastName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC004	email	String	30	user1@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC005	empAge	float	2	44.3	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC006	buttonType	String	1	manager2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC007	firstName	String	20	TestFirstName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC008	lastName	String	20	TestLastName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC009	email	String	30	user2@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC010	empAge	float	2	47.5	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin

รูปที่ 4-54 หน้าต่างปรับปรุงแก้ไขกรณีทดสอบ



รูปที่ 4-55 หน้าต่างสรุปผลการปรับปรุงแก้ไขซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ

4.2.8. ปุ่มนำออกไฟล์ที่ได้รับการปรับปรุงแก้ไข เมื่อคลิกที่ปุ่มนี้ เครื่องมือจะทำการรวบรวมไฟล์ซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบที่ได้รับการปรับปรุงแก้ไขและดาวน์โหลดเข้าสู่เครื่องคอมพิวเตอร์ของผู้ใช้ ดังรูปที่ 4-56



รูปที่ 4-56 ปุ่มนำออกไฟล์ที่ได้รับการปรับปรุงแก้ไข

4.2.9. หน้าต่างแสดงรายการไฟล์ แสดงดังรูปที่ 4-57 เป็นหน้าต่างที่แสดงรายการของไฟล์ทั้งหมดที่นำเข้ามาในเครื่องมือ รวมไปถึงไฟล์ซอร์สโค้ดและกรณีทดสอบที่ได้รับการปรับปรุงแก้ไขเสร็จเรียบร้อยแล้ว เพื่อให้ผู้ใช้ได้ทำการนำออกไฟล์ได้อีกครั้งหลังจากจบกระบวนการวิเคราะห์หาผลกระทบและปรับปรุงแก้ไขซอร์สโดไฮเบอร์เนตและกรณีทดสอบ โดยการคลิกที่ปุ่ม “Download” โดยเครื่องมือจะทำการนำออกไฟล์มาเก็บไว้ยังเครื่องคอมพิวเตอร์ของผู้ใช้

All Files

Show 10 Search:

Download	Project Id	File Id	File Name	File Type
Download	102	325	hibernate-Employee.cgf.xml	Configuration File
Download	102	326	EmployeeEntity.java	Persistance Class
Download	102	327	TestHibernate.java	Java Controller Class
Download	102	328	Thesis-TestCases3.xlsx	Test Case File
Download	102	484	Updated-EmployeeEntity.java	Persistance Class
Download	102	485	Updated-TestHibernate.java	Java Controller Class
Download	102	486	Updated-Thesis-TestCases3.xlsx	Test Case File
Download	103	332	hibernate.cfg.xml	Configuration File

รูปที่ 4-57 หน้าต่างสรุปผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบกรณีไม่มีผลกระทบ

บทที่ 5
การทดสอบเครื่องมือ

ในบทนี้จะกล่าวถึงการทดสอบเครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล ซึ่งจะกล่าวถึงสภาพแวดล้อมที่ใช้ในการทดสอบการทำงานของเครื่องมือและผลการทดสอบเครื่องมือโดยมีรายละเอียดดังต่อไปนี้

5.1. สภาพแวดล้อมที่ใช้ทดสอบ

สภาพแวดล้อมที่ใช้ทดสอบเครื่องมือ มีรายละเอียดดังต่อไปนี้

5.1.1. ฮาร์ดแวร์ (Hardware)

1. เครื่องคอมพิวเตอร์แบบพกพา หน่วยประมวลผลอินเทลคอร์ไอเซเว่น 1.6 กิกะเฮิรตซ์ (1.6 GHz Intel Core i7)
2. หน่วยความจำของคอมพิวเตอร์หรือแรม (Ram) 8.0 กิกะไบต์ (8 GB)
3. ฮาร์ดดิสก์ (Hard disk) 1 เทระไบต์ (1 TB)

5.1.2. ซอฟต์แวร์ (Software)

1. ระบบปฏิบัติการ (Operating System) แม็คโอเอส (macOS)
2. อีคลิปส์เจทูอีอี เวอร์ชัน 4.1 (Eclipse J2EE)
3. จาวาเจดีเค เวอร์ชัน 8 (Java JDK 8)
4. ฐานข้อมูลเอสคิวแอล เซิฟเวอร์ เวอร์ชัน 2016 (Microsoft SQL Server 2016)
5. เว็บเบราว์เซอร์กูเกิลโครม (Google Chrome) เวอร์ชัน 74

5.2. การทดสอบเครื่องมือ

การทดสอบเครื่องมือเป็นกระบวนการที่สำคัญอย่างหนึ่ง เพื่อตรวจสอบความถูกต้องของการทำงานของเครื่องมือว่ามีความสามารถในการวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบได้หรือไม่ อีกทั้งยังมีทดสอบความสามารถในการปรับปรุงแก้ไขซอร์สโดไฮเบอร์เนตและกรณีทดสอบในกรณีที่มีผลกระทบเกิดขึ้น โดยการทดสอบนี้จะกระทำกับระบบจำลองการทำงานที่สร้างขึ้นเพื่อทดสอบเครื่องมือนี้โดยเฉพาะ ซึ่งระบบจำลองนั้นประกอบไปด้วย 3 ระบบ คือ ระบบยืมหนังสือ ระบบจัดการพนักงาน และระบบสั่งซื้อสินค้า ดังรายละเอียดต่อไปนี้

5.2.1. กรณีทดสอบที่ 1 ระบบยืมหนังสือ

กรณีศึกษาระบบยืมหนังสือ เป็นระบบที่ใช้ในร้านเช่าหนังสือเพื่อบันทึกข้อมูลการยืมหนังสือของลูกค้าแต่ละคน โดยระบบนี้สามารถเพิ่มข้อมูลหนังสือ เพิ่มโปรโมชันการยืมหนังสือ บันทึกข้อมูลการยืมหนังสือ และคำนวณเงินค่ายืมหนังสือตามจำนวนวันที่ลูกค้าเลือก ซึ่งตัวอย่างกรณีศึกษานี้จะใช้ทดสอบเฉพาะฟังก์ชันการคำนวณเงินค่ายืมหนังสือ ด้วยการเปลี่ยนแปลงสคีมาฐานข้อมูลเพียงรูปแบบเดียว คือการเปลี่ยนคีย์หลักของตารางฐานข้อมูล เพื่อตรวจสอบว่าเครื่องมือสามารถวิเคราะห์หาผลกระทบต่อซอร์สโค้ดได้และปรับปรุงแก้ไขค่าคีย์ข้อมูลให้ถูกต้องตามการเปลี่ยนแปลงที่เกิดขึ้นในสคีมาฐานข้อมูล

การทดสอบเริ่มจากนำเข้าไฟล์ข้อมูล 4 ไฟล์คือ ไฟล์กำหนดค่า ไฟล์ซอร์สโค้ดเพอซิสเทินไฟล์ซอสโค้ดคอนโทรลเลอร์ และกรณีทดสอบ (รายละเอียดข้อมูลตามที่อธิบายในภาคผนวก ข) โดยมีการเปลี่ยนแปลงของสคีมาฐานข้อมูลคือ เปลี่ยนคีย์หลักของตารางจากแอตทริบิวต์ BOOK_ID ไปเป็นแอตทริบิวต์ BORROW_ID ดังแสดงในตารางที่ 5-1 และ 5-2

ตารางที่ 5-1 รายละเอียดก่อนการเปลี่ยนคีย์หลักของตารางในสคีมาฐานข้อมูลโปรแกรมทดสอบ

ชื่อแอตทริบิวต์	ชนิดของแอตทริบิวต์	ประเภทการเปลี่ยนแปลง	คีย์หลัก
BORROW_ID	Int(6)	-	
BOOK_ID	Int(6)	-	PRI

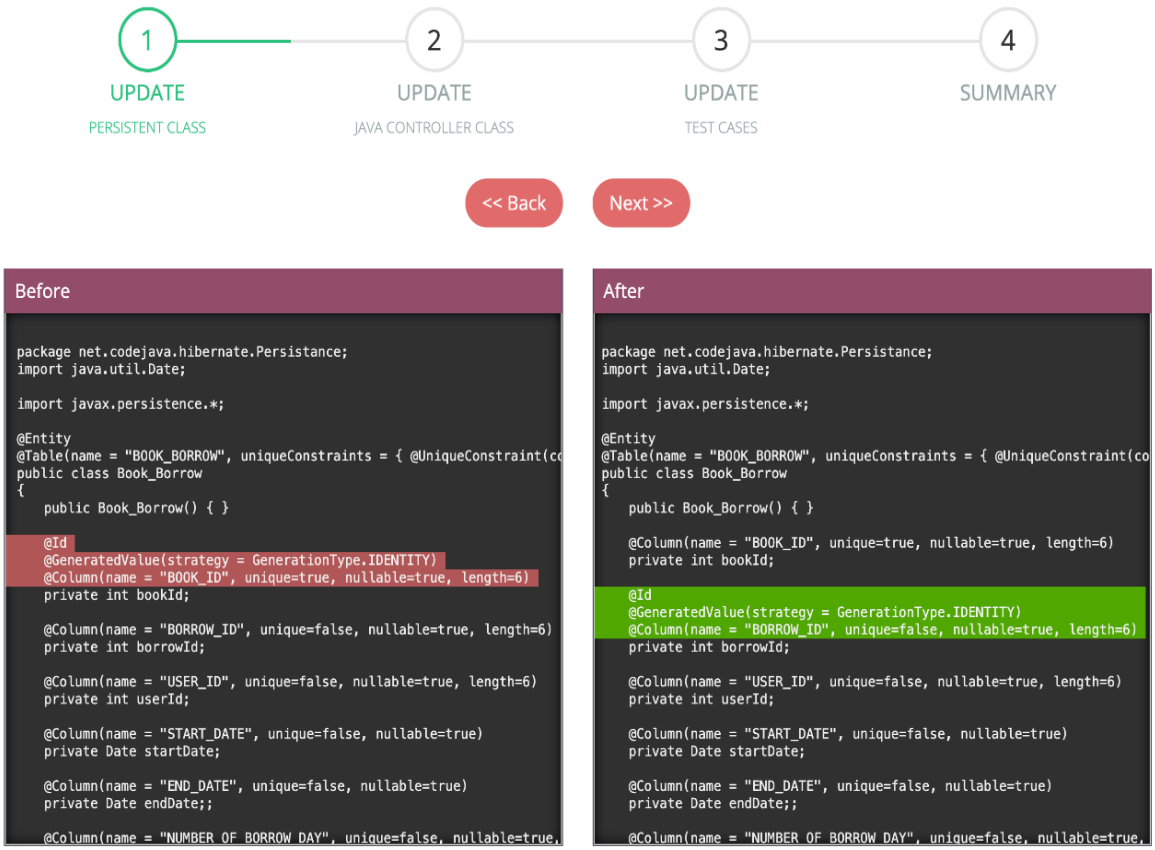
ตารางที่ 5-2 รายละเอียดหลังการเปลี่ยนคีย์หลักของตารางในสคีมาฐานข้อมูลโปรแกรมทดสอบ

ชื่อแอตทริบิวต์	ชนิดของแอตทริบิวต์	ประเภทการเปลี่ยนแปลง	คีย์หลัก
BORROW_ID	Int(6)	แก้ไข	PRI
BOOK_ID	Int(6)	แก้ไข	

การทดสอบการเปลี่ยนแปลงดังกล่าวได้ส่งผลกระทบต่อไฟล์ซอร์สโค้ดเพอซิสเทินไฟล์เดียวเท่านั้น ดังนั้นเครื่องมือจึงต้องแสดงผลการวิเคราะห์โดยไฮไลท์ตรงตำแหน่งของค่าคีย์หลักที่มีการเปลี่ยนแปลงในซอร์สโค้ดเพอซิสเทิน ส่วนซอร์สโค้ดคอนโทรลเลอร์และกรณีทดสอบที่ไม่ได้รับผลกระทบจึงไม่ได้ทำการไฮไลท์ แสดงดังรูปที่ 5-3

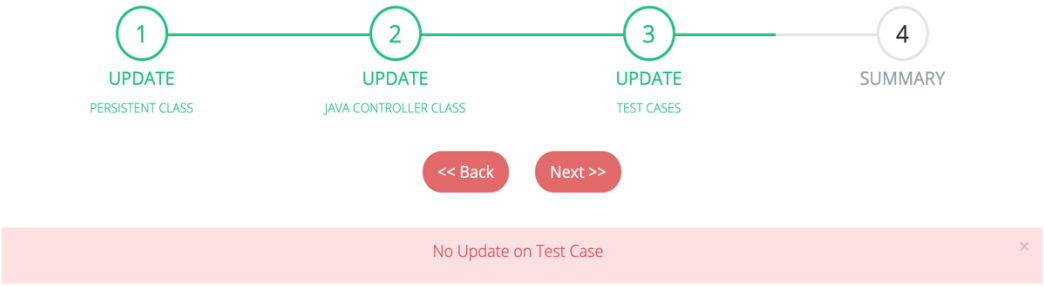
รูปที่ 5-1 ผลการวิเคราะห์ผลกระทบต่อซอร์สโค้ดและกรณีทดสอบของระบบยืมหนังสือ

ต่อมาเมื่อทราบว่ามีผลกระทบเกิดขึ้นกับไฟล์ซอร์สโค้ดเพอร์ซิสเทิน ผู้ใช้จะกดปุ่มปรับปรุงแก้ไขไฟล์ที่ได้รับผลกระทบเพื่อให้เครื่องมือเริ่มทำการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทิน โดยเครื่องมือจะแสดงซอร์สโค้ดทั้งก่อนทำการแก้ไข และหลังทำการแก้ไขให้กับผู้ใช้ได้เห็นผ่านทางหน้าจอ แสดงในรูปที่ 5-2 หากผู้ใช้กดปุ่มถัดไป เครื่องมือจะแสดงข้อความว่าไม่พบไฟล์ซอร์สโค้ดคอนโทรลเลอร์และไฟล์กรณีทดสอบที่จะทำการปรับปรุงแก้ไข เนื่องจากสองไฟล์นี้ไม่ได้รับผลกระทบจากการเปลี่ยนแปลงสคีมาฐานข้อมูล แสดงดังรูปที่ 5-3 และ5-4



รูปที่ 5-2 ผลการปรับปรุงแก้ไขซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบของระบบยืมหนังสือ

รูปที่ 5-3 ข้อความกรณีไม่พบไฟล์ซอร์สโค้ดคอนโทรลเลอร์ที่จะทำการปรับปรุงแก้ไข



รูปที่ 5-4 ข้อความกรณีไม่พบไฟล์กรณีทดสอบที่จะทำการปรับปรุงแก้ไข

สุดท้ายหลังจากที่เครื่องมือทำการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทินเสร็จเรียบร้อยแล้ว ผู้ใช้สามารถนำออกไฟล์ซอร์สโค้ดที่ได้รับการปรับปรุงแก้ไขเพื่อนำไปให้นักพัฒนาโปรแกรมใช้งานต่อได้โดยที่ไม่มีผลกระทบเกิดขึ้นต่อกรณีทดสอบ

5.2.2. กรณีทดสอบที่ 2 ระบบจัดการพนักงาน

ระบบจัดการพนักงาน เป็นระบบที่ใช้ในการจัดเก็บข้อมูลของพนักงานในบริษัท โดยระบบนี้สามารถ เพิ่มข้อมูลพนักงานใหม่ จัดสรรทีมโปรเจคให้กับพนักงาน จัดสรรหัวหน้าให้กับพนักงาน จัดการข้อมูลสวัสดิการของพนักงาน และจัดการข้อมูลเงินเดือนของพนักงาน ซึ่งตัวอย่างกรณีศึกษานี้จะใช้ฟังก์ชันของการจัดสรรหัวหน้าให้กับพนักงานมาทดสอบ โดยการเปลี่ยนแปลง 3 รูปแบบในโครงสร้างสคีมาฐานข้อมูล ดังนี้ แก้ไขประเภทข้อมูลของฟิลด์ แก้ไขขนาดข้อมูลของฟิลด์ และลบฟิลด์ เพื่อตรวจสอบว่าเครื่องมือสามารถวิเคราะห์หาผลกระทบที่เกิดขึ้นในไฟล์ซอร์สโค้ดและกรณีทดสอบได้ จากนั้นเครื่องมือจะต้องทำการปรับปรุงแก้ไขข้อมูลให้ถูกต้องตามการเปลี่ยนแปลงที่เกิดขึ้น

อย่างไรก็ตามการทดสอบจะเริ่มจากการนำเข้าข้อมูลตั้งต้นทั้ง 4 ไฟล์ ได้แก่ไฟล์กำหนดค่าไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบ หลังจากนั้นเครื่องมือจะทำการสกัดข้อมูลการเชื่อมต่อฐานข้อมูลของโปรแกรมทดสอบและทำการสกัดข้อมูลตารางที่เกี่ยวข้องเพื่อนำมาจัดเก็บในฐานข้อมูลของเครื่องมือ (รายละเอียดข้อมูลสคีมาฐานข้อมูลไฟล์กำหนดค่า ไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบที่เกี่ยวข้องในกรณีทดสอบนี้ จะแสดงรายละเอียดในภาคผนวก ข) โดยมีการเปลี่ยนแปลงของสคีมาฐานข้อมูลแสดงในตารางที่ 5-3

ตารางที่ 5-3 รายละเอียดการเปลี่ยนแปลงของข้อมูลในสคีมาฐานข้อมูลของระบบจัดการพนักงาน

ชื่อฟิลด์	การเปลี่ยนแปลง	ค่าเดิม	ค่าใหม่
EMAIL	แก้ไขขนาดข้อมูลของฟิลด์	100	200
LAST_NAME	แก้ไขขนาดข้อมูลของฟิลด์	100	200
EMP_AGE	แก้ไขประเภทข้อมูลของฟิลด์	Int	Float
IS_ACTIVE	ลบฟิลด์		
CREATED_DATE	ลบฟิลด์		

จากตารางที่ 5-3 แสดงรายละเอียดการเปลี่ยนแปลงที่เกิดขึ้นในสคีมาฐานข้อมูล ประกอบด้วย 5 รายการการเปลี่ยนแปลง ได้แก่ (1) การแก้ไขขนาดข้อมูลของฟิลด์ EMAIL จาก 100 ไปเป็น 200 (2) การแก้ไขขนาดข้อมูลของฟิลด์ LAST_NAME จาก 100 ไปเป็น 200 (3) แก้ไขประเภทข้อมูลของฟิลด์ EMP_AGE จาก int ไปเป็น float (4) การลบฟิลด์ IS_ACTIVE (5) การลบฟิลด์ CREATED_DATE

จากผลการวิเคราะห์ผลกระทบของเครื่องมือพบว่ามีผลกระทบเกิดขึ้นกับไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และกรณีทดสอบ ดังแสดงในตารางที่ 5-4

ตารางที่ 5-4 รายละเอียดผลกระทบต่อไฟล์ข้อมูลนำเข้าของระบบจัดการพนักงาน

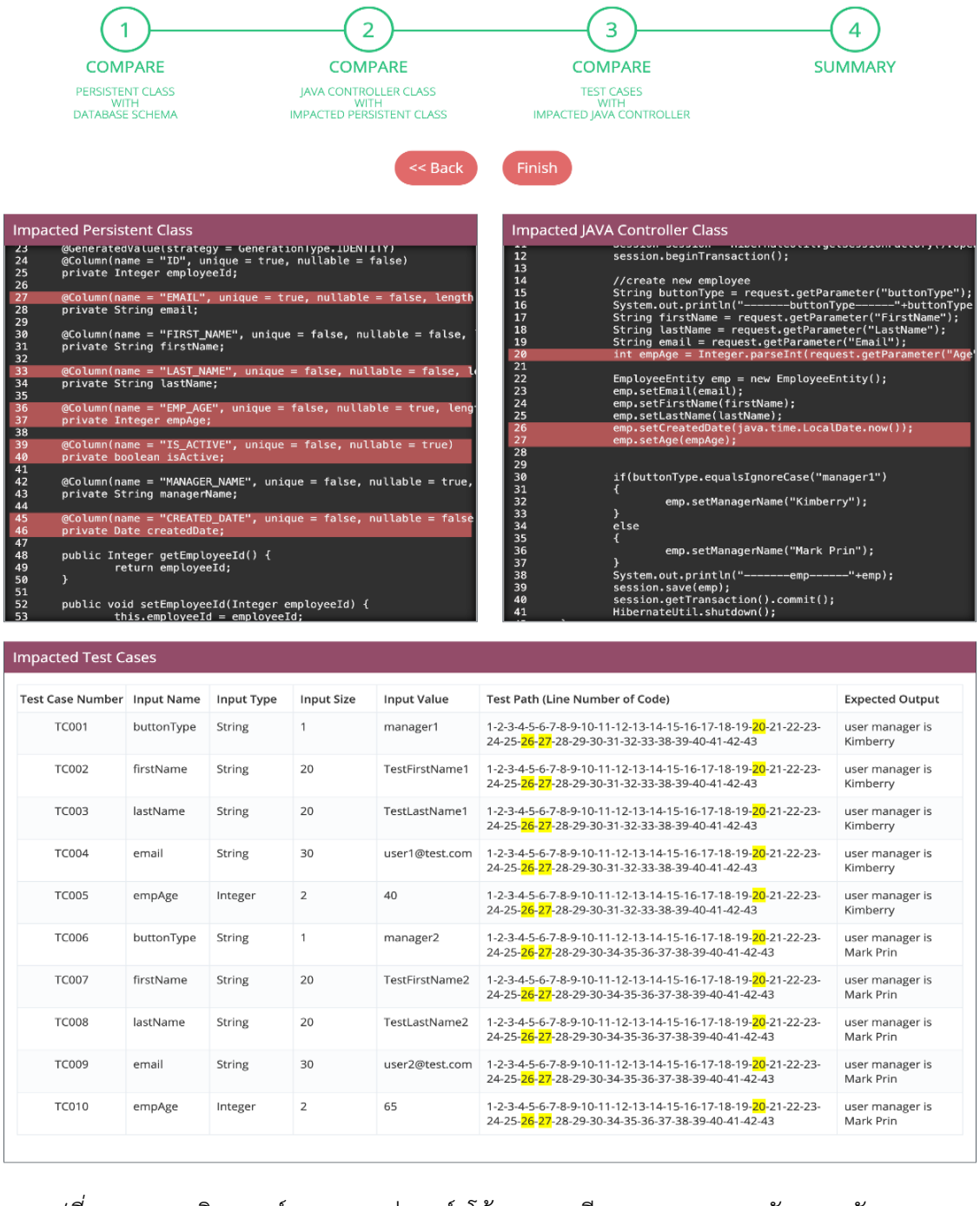
ชื่อไฟล์ข้อมูล	ผลกระทบ	รายละเอียด
ไฟล์ซอร์สโค้ดเพอซิสเทิน	กระทบต่อขนาดของฟิลด์ EMAIL	กระทบเนื่องจากเปลี่ยนแปลง-ขนาดของฟิลด์
	กระทบต่อขนาดของฟิลด์ LAST NAME	กระทบเนื่องจากเปลี่ยนแปลงขนาดของฟิลด์
	กระทบต่อขนาดของฟิลด์ EMP AGE	กระทบเนื่องจากเปลี่ยนแปลงขนาดของฟิลด์
	กระทบต่อชนิดของข้อมูลฟิลด์ EMP_AGE	กระทบเนื่องจากเปลี่ยนแปลงชนิดของฟิลด์
	กระทบต่อชื่อฟิลด์ CREATED_DATE	กระทบเนื่องจากลบฟิลด์
	กระทบต่อชื่อฟิลด์ IS_ACTIVE	กระทบเนื่องจากลบฟิลด์
ไฟล์ซอร์สโค้ดคอนโทรลเลอร์	กระทบต่อหมายเลขบรรทัด 20	กระทบจากตัวแปร empAge ที่ถูกกำหนดค่าไว้ในฟังก์ชัน setAge()
	กระทบต่อหมายเลขบรรทัด 26	กระทบจากฟังก์ชัน setCreateDate()

ตารางที่ 5-4 แสดงรายละเอียดผลกระทบต่อไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบ โดยไฟล์ซอร์สโค้ดเพอซิสเทินได้รับผลกระทบทั้งหมด 6 รายการ ได้แก่ (1) ผลกระทบต่อขนาดของฟิลด์ EMAIL (2) ผลกระทบต่อขนาดของฟิลด์ LAST_NAME (3) ผลกระทบต่อขนาดของฟิลด์ EMP_AGE (4) ผลกระทบต่อชนิดของข้อมูลฟิลด์ EMP_AGE (5) ผลกระทบต่อชื่อฟิลด์ CREATED_DATE (6) ผลกระทบต่อชื่อฟิลด์ IS_ACTIVE ส่วนไฟล์ซอร์สโค้ดคอนโทรลเลอร์ได้รับผลกระทบทั้งหมด 3 รายการ ได้แก่ (1) ผลกระทบต่อหมายเลขบรรทัด 20 เนื่องจากตัวแปร empAge ที่ถูกกำหนดค่าไว้ในฟังก์ชัน setAge() ได้รับผลกระทบจากซอร์สโค้ดเพอซิสเทิน (2) กระทบต่อหมายเลขบรรทัด 26 เนื่องจากฟังก์ชัน setCreateDate() ได้รับผลกระทบจากซอร์สโค้ดเพอซิสเทิน (3) ผลกระทบต่อหมายเลขบรรทัด 27 เนื่องจากฟังก์ชัน setAge() ได้รับผลกระทบจากซอร์สโค้ดเพอซิสเทิน และสุดท้ายไฟล์กรณีทดสอบได้รับผลกระทบทั้งหมด 10 รายการ เนื่องจากหมายเลขบรรทัด 20, 26 และ 27 ปรากฏอยู่ทุกหมายเลขทางเดินทดสอบ ดังนั้นการทดสอบการเปลี่ยนแปลงในสคีมาฐานข้อมูลได้ส่งผลกระทบต่อไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และไฟล์กรณีทดสอบ ดังแสดงในรูปที่ 5-5

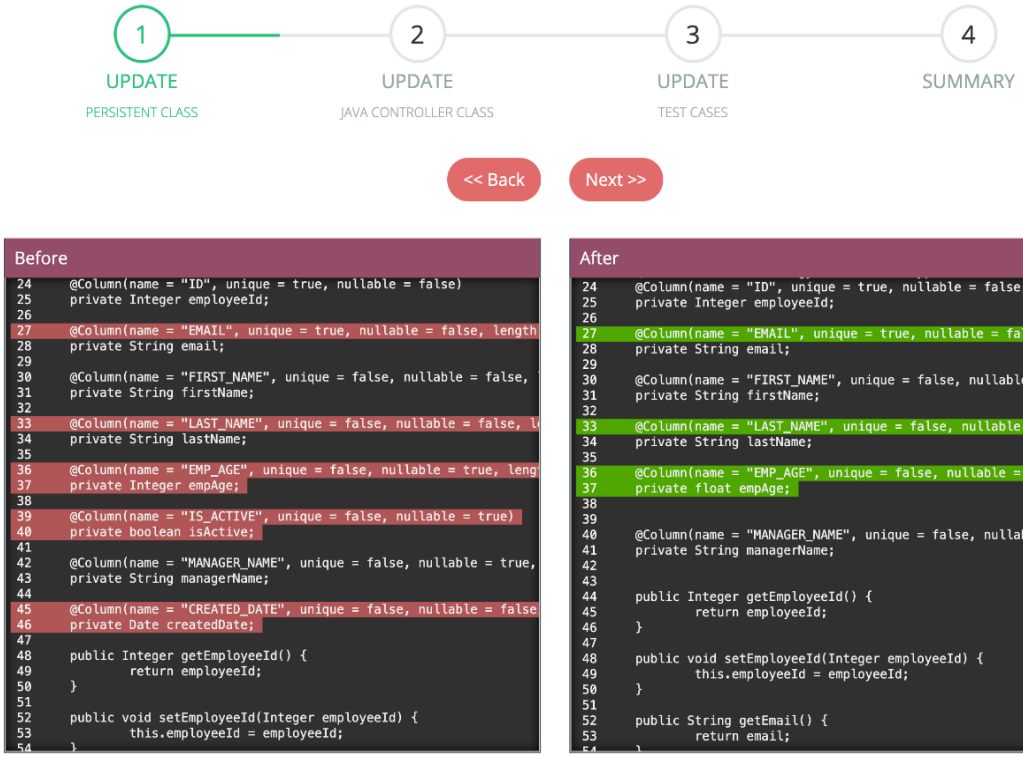
ต่อมาเมื่อผู้ใช้ทราบว่ามีผลกระทบเกิดขึ้นกับไฟล์ซอร์สโค้ดเพอร์ซิสเทิน ผู้ใช้จะกดปุ่มปรับปรุงแก้ไขไฟล์ที่ได้รับผลกระทบเพื่อให้เครื่องมือเริ่มทำการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทิน โดยเครื่องมือจะแสดงซอร์สโค้ดทั้งก่อนทำการแก้ไข และหลังทำการแก้ไขให้กับผู้ใช้ได้เห็นผ่านทางหน้าจอ ดังแสดงในรูปที่ 5-6 ถึง 5-8 โดยแต่ละรายการเปลี่ยนแปลงที่เกิดขึ้นจะมีวิธีการแก้ไขที่แตกต่างกัน ดังแสดงให้เห็นในตารางที่ 5-5

ตารางที่ 5-5 รายละเอียดการปรับปรุงแก้ไขไฟล์ข้อมูลของระบบจัดการพนักงาน

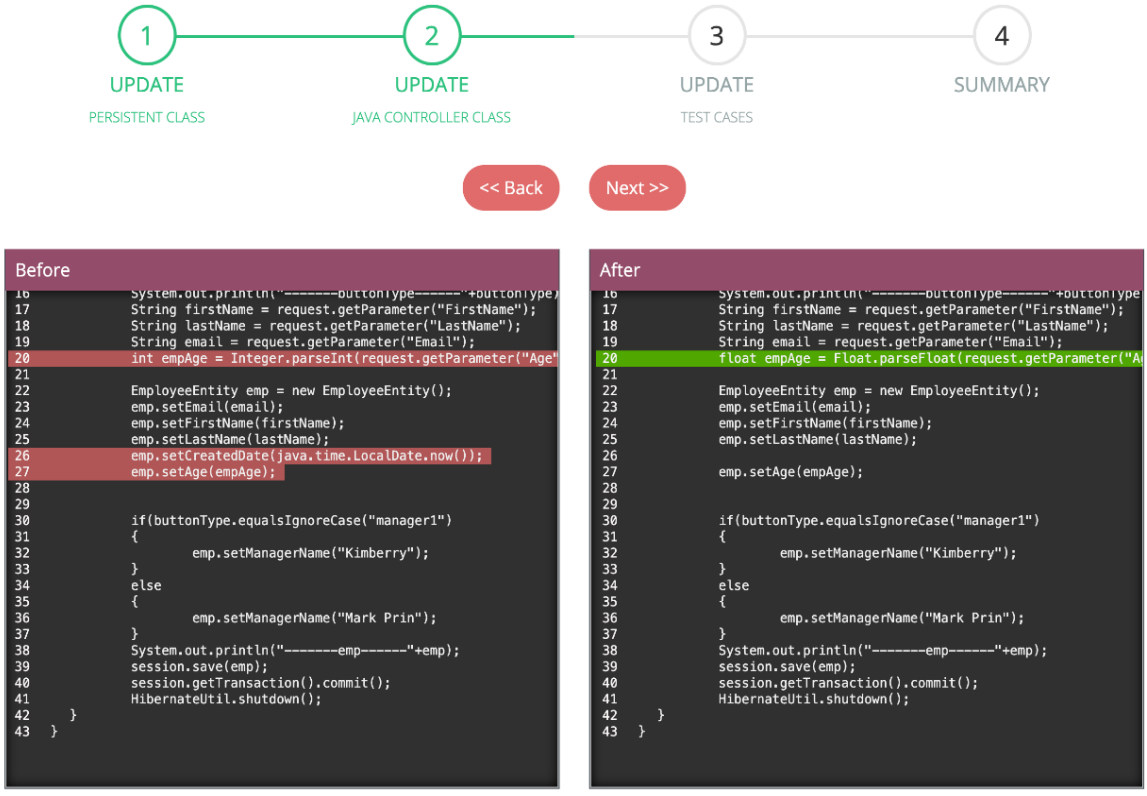
ชื่อไฟล์ข้อมูล	การเปลี่ยนแปลง	การปรับปรุงแก้ไข
ไฟล์ซอร์สโค้ดเพอซิสเทิน	เปลี่ยนแปลงขนาดของฟิลด์	แก้ไขขนาดของฟิลด์ให้ถูกต้อง
	เปลี่ยนแปลงชนิดของข้อมูลฟิลด์	แก้ไขชนิดของฟิลด์ให้ถูกต้อง
	ลบฟิลด์	ลบข้อมูลฟิลด์และฟังก์ชันที่เกี่ยวข้อง
ไฟล์ซอร์สโค้ดคอนโทรลเลอร์	เปลี่ยนแปลงชนิดของข้อมูลฟิลด์	แก้ไขชนิดของฟิลด์ให้ถูกต้อง
	ลบฟิลด์	ลบโค้ดที่มีชื่อตัวแปรและฟังก์ชันที่เกี่ยวข้อง
ไฟล์กรณีทดสอบ	เปลี่ยนแปลงขนาดของฟิลด์	แก้ไขขนาดของฟิลด์ให้ถูกต้อง
	เปลี่ยนแปลงชนิดของข้อมูลฟิลด์	แก้ไขชนิดของฟิลด์ให้ถูกต้อง
	ลบฟิลด์	ลบตัวแปรที่ถูกลบ(ถ้ามีตัวแปรปรากฏอยู่)



รูปที่ 5-5 ผลการวิเคราะห์ผลกระทบต่อซอร์สโค้ดและกรณีทดสอบของระบบจัดการพนักงาน



รูปที่ 5-6 ผลการปรับปรุงแก้ไขซอร์สโค้ดเพอซิสเทินของระบบจัดการพนักงาน



รูปที่ 5-7 ผลการปรับปรุงแก้ไขซอร์สโค้ดคอนโทรลเลอร์ของระบบจัดการพนักงาน

Before

Test Case Number	Input Name	Input Type	Input Size	Input Value	Test Path	Expected Output
TC001	buttonType	String	1	manager1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC002	firstName	String	20	TestFirstName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC003	lastName	String	20	TestLastName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC004	email	String	30	user1@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC005	empAge	Integer	2	40	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC006	buttonType	String	1	manager2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC007	firstName	String	20	TestFirstName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC008	lastName	String	20	TestLastName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC009	email	String	30	user2@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC010	empAge	Integer	2	65	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin

After

Test Case Number	Input Name	Input Type	Input Size	Input Value	Test Path	Expected Output
TC001	buttonType	String	1	manager1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC002	firstName	String	20	TestFirstName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC003	lastName	String	20	TestLastName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC004	email	String	30	user1@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC005	empAge	float	2	30.1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC006	buttonType	String	1	manager2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC007	firstName	String	20	TestFirstName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC008	lastName	String	20	TestLastName2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC009	email	String	30	user2@test.com	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin
TC010	empAge	float	2	37.6	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	user manager is Mark Prin

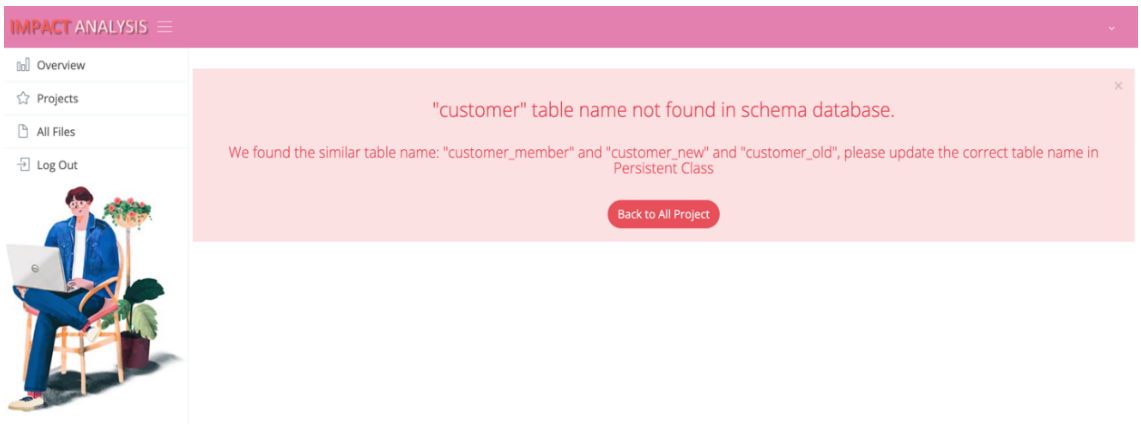
รูปที่ 5-8 ผลการปรับปรุงแก้ไขซอร์กรณีทดสอบของระบบจัดการพนักงาน

สุดท้ายหลังจากที่เครื่องมือทำการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทินเสร็จเรียบร้อยแล้ว ผู้ใช้สามารถนำออกไฟล์ซอร์สโค้ดที่ได้รับการปรับปรุงแก้ไขเพื่อนำไปให้นักพัฒนาโปรแกรมใช้งานต่อได้โดยที่ไม่มีผลกระทบเกิดขึ้นต่อกรณีทดสอบ

5.2.3. กรณีทดสอบที่ 3 ระบบสั่งซื้อสินค้า

ระบบสั่งซื้อสินค้า เป็นระบบที่ใช้สำหรับสั่งซื้อสินค้าออนไลน์ โดยระบบสามารถจัดการข้อมูลสินค้า จัดการข้อมูลคำสั่งซื้อสินค้า จัดการข้อมูลลูกค้า คำนวณราคาการสั่งซื้อสินค้า จัดการข้อมูลการจ่ายเงิน รวมไปถึงจัดการข้อมูลใบเสร็จของลูกค้า ตัวอย่างกรณีศึกษานี้จะใช้ฟังก์ชันการจัดการข้อมูลการจ่ายเงินมาทดสอบ โดยการเปลี่ยนชื่อตารางในสคีมาฐานข้อมูล เพื่อตรวจสอบว่าเครื่องมือสามารถวิเคราะห์ได้ว่าไม่พบข้อมูลตาราง และแสดงชื่อตารางที่ใกล้เคียงกันให้ผู้ใช้หหได้รับทราบ

เมื่อผู้ใช้สร้างโครงการใหม่และนำเข้าไฟล์ข้อมูลตั้งต้นทั้ง 4 ไฟล์ (สามารถดูรายละเอียดเพิ่มเติมได้ในภาคผนวก ข) เครื่องมือจะทำการตรวจสอบหาข้อมูลชื่อตารางที่สกัดมาจากไฟล์ซอร์สโค้ดเพอซิสเทินในสคีมาฐานข้อมูลของระบบสั่งซื้อสินค้า ในกรณีที่มีการเปลี่ยนแปลงชื่อตารางนั้น เครื่องมือจะไม่สามารถทำการปรับปรุงแก้ไขให้ได้ เนื่องจากเครื่องมือไม่สามารถรับรู้ได้ว่าชื่อตารางเปลี่ยนจากชื่อเดิมเป็นชื่ออะไร แต่เครื่องมือจะทำการแสดงข้อความให้ผู้ใช้ได้ทราบว่าไม่พบข้อมูลของตารางที่ต้องการ และแสดงรายการชื่อตารางที่ใกล้เคียงเพื่อให้ผู้ใช้สามารถปรับปรุงแก้ไขชื่อตารางเป็นตารางที่ถูกต้อง ก่อนนำเข้าสู่เครื่องมือเพื่อวิเคราะห์หาผลกระทบในลำดับต่อไป ดังแสดงในรูปที่ 5-9



รูปที่ 5-9 ผลการวิเคราะห์ซอร์สโค้ดไฮเบอร์เนตของระบบสั่งซื้อสินค้า

5.3. ผลการทดสอบเครื่องมือ

จากการทดสอบเครื่องมือโดยใช้ไฟล์ซอร์สโค้ดและกรณีทดสอบที่สร้างขึ้นเฉพาะ ได้รับผลการทดสอบดังตารางที่ 5-6 สามารถดูรายละเอียดเพิ่มเติมได้ในภาคผนวก ข.

ตารางที่ 5-6 ผลการทดสอบการวิเคราะห์ผลกระทบต่อซอร์สโค้ดและกรณีทดสอบ

ชื่อระบบ	รายการการเปลี่ยนแปลงของสคีมาฐานข้อมูล	จำนวนบรรทัดของซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบ	จำนวนบรรทัดของซอร์สโค้ดคอนโทรลเลอร์ที่ได้รับผลกระทบ	จำนวนกรณีทดสอบที่ได้รับผลกระทบ
ระบบยืมหนังสือ	เปลี่ยนคีย์หลักของตารางฐานข้อมูล	3	0	0
ระบบจัดการพนักงาน	แก้ไขประเภทข้อมูลของฟิลด์	8	2	2
	แก้ไขขนาดข้อมูลของฟิลด์	2	0	0
	ลบฟิลด์	16	1	1
ระบบสั่งซื้อสินค้า	เปลี่ยนชื่อตารางในสคีมาฐานข้อมูล	1	0	0

ตารางที่ 5-6 แสดงผลการทดสอบการวิเคราะห์ผลกระทบต่อซอร์สโค้ดและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล ซึ่งข้อมูลในตารางประกอบไปด้วย ชื่อระบบ รายการการเปลี่ยนแปลงของสคีมาฐานข้อมูล จำนวนบรรทัดของซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบ จำนวนบรรทัดของซอร์สโดคอนโทรลเลอร์ที่ได้รับผลกระทบ และกรณีทดสอบที่ได้รับผลกระทบ โดยระบบทดสอบทั้ง 3 ระบบ ได้มีการปรับปรุงแก้ไขที่แตกต่างกัน ตัวอย่างเช่น ระบบยืมหนังสือที่มีการเปลี่ยนคีย์หลักของฟิลด์ หลังจากเครื่องมือทำการวิเคราะห์ผลกระทบแล้วพบว่า มีจำนวนหมายเลขบรรทัดของซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบทั้งหมด 3 บรรทัด แต่ไม่มีจำนวนซอร์สโค้ดคอนโทรลเลอร์และกรณีทดสอบที่ได้รับผลกระทบ เนื่องจากมีการกำหนดคีย์หลักของตารางที่ไฟล์ซอร์สโค้ดเพอซิสเทินเท่านั้น ดังนั้นเครื่องมือจึงทำการแก้ไขเฉพาะไฟล์ซอร์สโค้ดเพอซิสเทินให้กลับมาใช้งานได้เพียงไฟล์เดียว

ในส่วนของระบบทดสอบที่ 2 ระบบการจัดการพนักงานที่มีการแก้ไขข้อมูล 3 ประเภท ได้แก่ การแก้ไขประเภทข้อมูลของฟิลด์ การแก้ไขขนาดของข้อมูลของฟิลด์ และการลบฟิลด์ มีจำนวนหมายเลขบรรทัดของซอร์สโค้ดเพอซิสเทินที่ได้รับผลกระทบทั้งหมด 8 บรรทัด 2 บรรทัด และ16 บรรทัดตามลำดับ โดยมีแค่ 2 รายการการเปลี่ยนแปลงเท่านั้นที่มีผลกระทบต่อซอร์สโค้ดคอนโทรลเลอร์และกรณีทดสอบ เนื่องจากการแก้ไขขนาดของข้อมูลเป็นการเพิ่มขนาดข้อมูลเพิ่มขึ้นจาก 100 เป็น 200 ตัวอักษรทำให้ไม่ส่งผลกระทบต่อข้อมูลที่มีอยู่แล้วในซอร์สโค้ดคอนโทรลเลอร์และกรณีทดสอบ ดังนั้นเครื่องมือจะทำการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบทั้งหมดเพื่อให้ผู้ใช้สามารถนำไปใช้งานต่อได้อย่างถูกต้อง

ระบบทดสอบสุดท้ายระบบสั่งซื้อสินค้าที่มีการเปลี่ยนแปลงชื่อตารางในสคีมาฐานข้อมูล พบว่ามีผลกระทบต่อไฟล์ซอร์สโค้ดเพอซิสเทินเพียงไฟล์เดียว แต่เครื่องมือจะไม่ทำการแก้ไขไฟล์ข้อมูล เนื่องจากไฟล์ซอร์สโค้ดเพอซิสเทินเป็นไฟล์ที่เชื่อมต่อข้อมูลตารางกับสคีมาฐานข้อมูล หากมีการเปลี่ยนแปลงชื่อตารางเกิดขึ้น เครื่องมือจะไม่สามารถรู้ได้ว่าชื่อตารางได้เปลี่ยนไปเป็นชื่ออะไร ดังนั้นเครื่องมือจะทำการแสดงข้อความผ่านทางหน้าจอให้ผู้ใช้ทราบว่าไม่พบชื่อตารางที่ต้องการ และจะแสดงชื่อตารางที่ใกล้เคียงเพื่อให้ผู้ใช้สามารถนำไปปรับปรุงแก้ไขเองได้

ดังนั้นจึงสรุปผลการทดสอบได้ว่า เครื่องมือสามารถวิเคราะห์หาผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบได้อย่างถูกต้องตามที่ออกแบบไว้ โดยเครื่องมือสามารถรองรับการเปลี่ยนแปลงได้ทั้งหมด 7 รูปแบบคือ การแก้ไขชื่อตาราง การลบตาราง การแก้ไขชื่อฟิลด์ การแก้ไขประเภทข้อมูลของฟิลด์ การแก้ไขขนาดข้อมูลของฟิลด์ การแก้ไขเปลี่ยนแปลงคีย์หลักของฟิลด์ และการลบฟิลด์ หลังจากนั้นหากพบว่ามีผลกระทบเกิดขึ้นต่อซอร์สโค้ดหรือกรณีทดสอบ เครื่องมือจะปรับปรุงแก้ไขไฟล์ข้อมูลที่ได้รับผลกระทบ นอกจากนี้ยังสามารถนำออกไฟล์ที่ได้รับการปรับปรุงแก้ไขออกจากเครื่องมือเพื่อให้ผู้ใช้สามารถนำไปใช้งานต่อไปได้อย่างถูกต้อง

บทที่ 6

สรุปผลการวิจัยและข้อเสนอแนะ

จากการศึกษาวิจัยและพัฒนาเครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนต และกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล สามารถสรุปผลวิจัย ข้อจำกัดของเครื่องมือ และแนวทางในการพัฒนาต่อ โดยมีรายละเอียดดังนี้

6.1. สรุปผลการวิจัย

งานวิจัยนี้นำเสนอเครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล โดยรองรับการเปลี่ยนแปลงที่เกิดขึ้นในสคีมาฐานข้อมูล 7 รูปแบบ คือ การแก้ไขชื่อตาราง การลบตาราง การแก้ไขชื่อฟิลด์ การแก้ไขประเภทข้อมูลของฟิลด์ การแก้ไขขนาดข้อมูลของฟิลด์ การแก้ไขเปลี่ยนแปลงคีย์หลักของฟิลด์ และการลบฟิลด์ ทั้งนี้ผู้วิจัยได้ทดสอบเครื่องมือกับการทำงานจริงใน 3 กรณีศึกษาและพบว่าเครื่องมือสามารถวิเคราะห์หาผลกระทบที่เกิดขึ้นต่อไฟล์ซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบได้อย่างถูกต้องตามวิธีการที่ได้นำเสนอไว้ โดยหลังจากการวิเคราะห์ผลกระทบที่เกิดขึ้น เครื่องมือสามารถทำการปรับปรุงแก้ไขซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบที่ได้รับผลกระทบจากการเปลี่ยนแปลงของสคีมาฐานข้อมูล พร้อมทั้งสามารถนำออกไฟล์ข้อมูลที่ได้รับการปรับปรุงแก้ไขเพื่อนำไปใช้งานจริงได้

6.2. ข้อจำกัดของเครื่องมือ

เครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล มีข้อจำกัดดังต่อไปนี้

1. เครื่องมือสามารถรองรับการเปลี่ยนแปลงที่เกิดขึ้นในสคีมาฐานข้อมูล 7 รูปแบบคือ การแก้ไขชื่อตาราง การลบตาราง การแก้ไขชื่อฟิลด์ การแก้ไขประเภทข้อมูลของฟิลด์ การแก้ไขขนาดข้อมูลของฟิลด์ การแก้ไขเปลี่ยนแปลงคีย์หลักของฟิลด์ และการลบฟิลด์ เท่านั้น
2. เครื่องมือจะไม่รองรับกรณีที่มีการเพิ่มและลบคีย์หลักของฟิลด์
3. ไฟล์กำหนดค่าที่ใช้ในโปรแกรมทดสอบจะต้องเชื่อมต่อกับฐานข้อมูลที่รองรับด้วย MySQL
4. ไฟล์กรณีทดสอบที่นำมาวิเคราะห์ผลกระทบจะต้องสอดคล้องกับซอร์สโค้ดที่อยู่ในคอนโทรลเลอร์คลาสเท่านั้น
5. เครื่องมือสามารถวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตได้ครั้งละ1คลาสเท่านั้น

6.3. แนวทางการพัฒนาต่อ

เครื่องมือวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบสำหรับการเปลี่ยนแปลงสคีมาฐานข้อมูล สามารถนำไปต่อยอดพัฒนาเพิ่มเติมให้รองรับฟังก์ชันเพิ่มเติมและแก้ไขข้อจำกัดได้ดังต่อไปนี้

1. sพัฒนาเครื่องมือให้รองรับการวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตได้ครั้งละมากกว่า 1 คลาส
2. พัฒนาเครื่องมือให้รองรับการเปลี่ยนแปลงที่เกิดขึ้นกับสคีมาฐานข้อมูลให้มากกว่า 7 รูปแบบ เช่น การเพิ่มคีย์หลักของตาราง การลบคีย์หลักของตาราง และการเพิ่มฟิลด์ เป็นต้น
3. พัฒนาเครื่องมือให้สามารถจัดเก็บไฟล์ที่ได้รับการปรับปรุงในรูปแบบเวอร์ชันได้
4. พัฒนาเครื่องมือให้สามารถรองรับซอร์สโค้ดรูปแบบอื่นๆที่ไม่ใช่ซอร์สโค้ดไฮเบอร์เนตได้
5. พัฒนาเครื่องมือให้รองรับการแจ้งเตือนให้ผู้ใช้งานได้รับทราบ กรณีมีการเพิ่มแอตทริบิวต์ใหม่เข้ามาในสคีมาฐานข้อมูล

บรรณานุกรม

1. Suwannasart, C.S.a.T., *A Tool for Analyzing Impacts to Source Code and Test Cases for Database Schema Changes.* 2014.
2. *Hibernate (framework)*.
3. *Hibernate – Mapping Types*.
4. *Software and systems engineering — Software testing*. ISO/IEC/IEEE. 2013.
5. Jorgensen, P.C., *Software Testing: A Craftsman's Approach*. 2014.
6. Suwannasart, J.J.a.T., *A Tool for test case impact analysis of database schema changes using use cases.* 2014.
7. Suwannasart, A.K.a.T., *Impact Analysis to Database Schema and Test Cases from Inputs of Functional Requirement Changes.* 2016: p. 449-453.
8. Suwannasart, S.P.a.T., *A Tool for Impact Analysis of Test Cases Based on Changed of A Web Application.* 2014: p. 497-500.
9. Karahasanovic, A., *Identify Impact of Database Schema on Application.* 2001. **1**: p. 93-104.
10. Malevris, S.K.G.a.N., *A Two-folded Impact Analysis of schema change on Database Applications.* 2009. **6**: p. 109-123.

ภาคผนวก ก

รายละเอียดยูสเคสของเครื่องมือ

ในภาคผนวก ก จะแสดงรายละเอียดยูสเคสแต่ละรายการอย่างละเอียด ดังต่อไปนี้

ตารางที่ ก-1 รายละเอียดยูสเคสนำเข้าไฟล์ข้อมูล

หมายเลขยูสเคส	UC-01
ชื่อยูสเคส	นำเข้าไฟล์ข้อมูล
รายละเอียดยูสเคส	เพื่อให้ผู้ใช้นำเข้าไฟล์ข้อมูลตั้งต้นเข้าสู่เครื่องมือ
ผู้กระทำ	ผู้ใช้
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	ผู้ใช้ลงชื่อเข้าใช้งานสำเร็จ
ขั้นตอน	1. ผู้ใช้คลิกปุ่ม ‘New Project’ 2. ผู้ใช้ป้อนข้อมูลชื่อโครงการ 3. ผู้ใช้คลิกปุ่ม “Next” เพื่อไปยังหน้าจอนำเข้าไฟล์ข้อมูล 4. ผู้ใช้นำเข้าข้อมูล 4 ไฟล์ได้แก่ 4.1. ไฟล์กำหนดค่า 4.2. ไฟล์ซอร์สโค้ดเพอซิสเทิน 4.3. ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ 4.4. ไฟล์กรณีทดสอบ 5. ผู้ใช้คลิกปุ่ม “Next” เพื่อไปยังหน้าจอสรุปรายละเอียดของข้อมูลโครงการ 6. ผู้ใช้คลิกปุ่ม “Submit” เพื่อบันทึกข้อมูลโครงการเข้าสู่ฐานข้อมูล
เงื่อนไขภายหลัง	ข้อมูลโครงการถูกบันทึกเข้าสู่ฐานข้อมูล

ตารางที่ ก-1 รายละเอียดยูสเคสวิเคราะห์ผลกระทบของซอร์สโค้ด

หมายเลขยูสเคส	UC-02
ชื่อยูสเคส	วิเคราะผลกระทบของซอร์สโค้ด
รายละเอียดยูสเคส	เพื่อวิเคราะห์ผลกระทบที่เกิดขึ้นต่อไฟล์ซอร์สโค้ดเพอซิสเทินและไฟล์ซอร์สโค้ดคอนโทรลเลอร์
ผู้กระทำ	ผู้ใช้
ความสัมพันธ์	Include: วิเคราะห์ผลกระทบต่อกรณีทดสอบ, แสดงผลรายงาน

ตารางที่ ก-2 รายละเอียดยูสเคสวิเคราะห์ผลกระทบของซอร์สโค้ด (ต่อ)

เงื่อนไขก่อนหน้า	นำเข้าข้อมูลไฟล์ตั้งต้นทั้งหมดเสร็จเรียบร้อยแล้ว
ขั้นตอน	1. ผู้ใช้เลือกโครงการเพื่อวิเคราะห์หาผลกระทบต่อไฟล์ซอร์สโค้ดไฮเบอร์เนต 2. ระบบแสดงข้อมูลโครงการและรายละเอียดของไฟล์นำเข้าทั้ง 4 ไฟล์ 3. ผู้ใช้คลิกปุ่ม “Impact Analyze” 4. ตรวจสอบว่ามีผลกระทบต่อซอร์สโค้ดเพอซิสเทินหรือไม่ 4.1.1.ถ้ามี 4.1.1.1. เครื่องมือแสดงรายงานผลกระทบผ่านทางหน้าจอพร้อมทั้งไฮไลท์ตำแหน่งที่ได้รับผลกระทบ 4.1.1.2. บันทึกข้อมูลผลกระทบเข้าสู่ฐานข้อมูลของเครื่องมือ 4.1.2.ถ้าไม่มี 4.1.2.1. เครื่องมือแสดงข้อความให้ผู้ใช้ได้รับทราบว่าไม่มีผลกระทบเกิดขึ้นต่อซอร์สโค้ดเพอซิสเทิน 4.1.2.2. เครื่องมือแสดงรายงานข้อมูลซอร์สโค้ดเพอซิสเทินโดยที่ไม่มีไฮไลท์สีปรากฏบนหน้าจอ 5. ผู้ใช้คลิกปุ่ม “Next” 6. ตรวจสอบว่ามีผลกระทบต่อซอร์สโค้ดคอนโทรลเลอร์หรือไม่ 6.1.1.ถ้ามี 6.1.1.1. เครื่องมือแสดงผลกระทบผ่านทางหน้าจอพร้อมทั้งไฮไลท์ตำแหน่งที่ได้รับผลกระทบ 6.1.1.2. บันทึกข้อมูลผลกระทบเข้าสู่ฐานข้อมูลของเครื่องมือ 6.1.2.ถ้าไม่มี 6.1.2.1. เครื่องมือแสดงข้อความให้ผู้ใช้ได้รับทราบว่าไม่มีผลกระทบเกิดขึ้นต่อซอร์สโค้ดคอนโทรลเลอร์ 6.1.2.2. เครื่องมือแสดงรายงานข้อมูลซอร์สโค้ดคอนโทรลเลอร์โดยที่ไม่มีไฮไลท์สีปรากฏบนหน้าจอ
เงื่อนไขภายหลัง	เครื่องมือแสดงรายงานผลกระทบผ่านทางหน้าจอ

ตารางที่ ก-2 รายละเอียดยูสเคสวิเคราะห์ผลกระทบต่อกรณีทดสอบ

หมายเลขยูสเคส	UC-03
ชื่อยูสเคส	วิเคราะห์ผลกระทบต่อกรณีทดสอบ
รายละเอียดยูสเคส	เพื่อวิเคราะห์ผลกระทบที่เกิดขึ้นต่อไฟล์กรณีทดสอบ
ผู้กระทำ	ผู้ใช้
ความสัมพันธ์	Include: แสดงผลรายงาน
เงื่อนไขก่อนหน้า	วิเคราะห์ผลกระทบต่อไฟล์ซอร์สโค้ดเพอซิสเทินและไฟล์ซอร์สโค้ดคอนโทรลเลอร์เสร็จเรียบร้อยแล้ว
ขั้นตอน	1. ผู้ใช้คลิกปุ่ม “Next” 2. ตรวจสอบว่ามีผลกระทบต่อกรณีทดสอบหรือไม่ 2.1. ถ้ามี 2.1.1. เครื่องมือแสดงรายงานผลกระทบผ่านทางหน้าจอพร้อมทั้งไฮไลท์ตำแหน่งที่ได้รับผลกระทบ 2.1.2. บันทึกข้อมูลผลกระทบเข้าสู่ฐานข้อมูลของเครื่องมือ 2.2. ถ้าไม่มี 2.2.1. เครื่องมือแสดงข้อความให้ผู้ใช้ได้รับทราบว่าไม่มีผลกระทบเกิดขึ้นต่อกรณีทดสอบ 2.2.2. เครื่องมือแสดงรายงานข้อมูลซอร์สโค้ดคอนโทรลเลอร์โดยที่ไม่มีไฮไลท์สีปรากฏบนหน้าจอ
เงื่อนไขภายหลัง	เครื่องมือแสดงรายงานผลกระทบผ่านทางหน้าจอ

ตารางที่ ก-3 รายละเอียดยูสเคสแก้ไขซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบ

หมายเลขยูสเคส	UC-04
ชื่อยูสเคส	แก้ไขซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบ
รายละเอียดยูสเคส	เพื่อแก้ไขซอร์สโค้ดเพอซิสเทิน ซอร์สโค้ดคอนโทรลเลอร์ และกรณีทดสอบที่ได้รับผลกระทบ
ผู้กระทำ	ผู้ใช้
ความสัมพันธ์	Extend: แสดงผลรายงาน
เงื่อนไขก่อนหน้า	วิเคราะห์ผลกระทบต่อไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และกรณีทดสอบเสร็จเรียบร้อยแล้ว

ตารางที่ ก-4 รายละเอียดยูสเคสแก้ไขซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบ (ต่อ)

ขั้นตอน	1. ผู้ใช้คลิกปุ่ม “Update Impacted files” 2. ตรวจสอบว่ามีผลกระทบต่อซอร์สโค้ดเพอซิสเทินหรือไม่ 2.1. ถ้ามี 2.1.1. ปรับปรุงแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทิน 2.1.2. บันทึกไฟล์ซอร์สโค้ดเพอซิสเทินที่ได้รับการแก้ไขเข้าสู่ฐานข้อมูล 2.1.3. เครื่องมือแสดงรายงานซอร์สโค้ดเพอซิสเทินก่อนและหลังการปรับปรุงแก้ไขบนหน้าจอ 2.2. ถ้าไม่มี 2.2.1. เครื่องมือแสดงข้อความแจ้งให้ผู้ใช้ได้รับทราบว่าไม่สามารถปรับปรุงแก้ไขได้เนื่องจากไม่มีผลกระทบเกิดขึ้นต่อซอร์สโค้ดเพอซิสเทิน 3. ผู้ใช้คลิกปุ่ม “Next” 4. ตรวจสอบว่ามีผลกระทบต่อซอร์สโค้ดคอนโทรลเลอร์หรือไม่ 4.1. ถ้ามี 4.1.1.ปรับปรุงแก้ไขไฟล์ซอร์สโค้ดคอนโทรลเลอร์ 4.1.2. บันทึกไฟล์ซอร์สโค้ดคอนโทรลเลอร์ที่ได้รับการแก้ไขเข้าสู่ฐานข้อมูล 4.1.3. เครื่องมือแสดงรายงานซอร์สโค้ดคอนโทรลเลอร์ก่อนและหลังการปรับปรุงแก้ไขบนหน้าจอ 4.2. ถ้าไม่มี 4.2.1. เครื่องมือแสดงข้อความแจ้งให้ผู้ใช้ได้รับทราบ ว่าไม่สามารถปรับปรุงแก้ไขได้เนื่องจากไม่มีผลกระทบเกิดขึ้นต่อซอร์สโค้ดคอนโทรลเลอร์ 5. ผู้ใช้คลิกปุ่ม “Next” 6. ตรวจสอบว่ามีผลกระทบต่อกรณีทดสอบหรือไม่ 6.1. ถ้ามี 6.1.1.ปรับปรุงแก้ไขไฟล์กรณีทดสอบ 6.1.2.บันทึกไฟล์กรณีทดสอบที่ได้รับการแก้ไขเข้าสู่ฐานข้อมูล

ตารางที่ ก-4 รายละเอียดยูสเคสแก้ไขซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบ (ต่อ)

ขั้นตอน	6.1.3.เครื่องมือแสดงรายงานกรณีทดสอบที่ได้รับการปรับปรุงแก้ไขบนหน้าจอ 6.2. ถ้าไม่มี 6.2.1.เครื่องมือแสดงข้อความแจ้งให้ผู้ใช้ได้รับทราบ ว่าไม่สามารถปรับปรุงแก้ไขได้เนื่องจากไม่มีผลกระทบเกิดขึ้นต่อกรณีทดสอบ
เงื่อนไขภายหลัง	บันทึกไฟล์ข้อมูลที่ได้รับการปรับปรุงแก้ไขเข้าสู่ฐานข้อมูล พร้อมทั้งแสดงผลการปรับปรุงแก้ไขผ่านทางหน้าจอ

ภาคผนวก ข
ตัวอย่างข้อมูลทดสอบเครื่องมือ

บทนี้จะกล่าวถึงรายละเอียดของตัวอย่างกรณีทดสอบ 3 กรณีทดสอบ ที่ใช้ทดสอบกับเครื่องมือ โดยมีรายละเอียดดังต่อไปนี้

1. กรณีศึกษาระบบยืมหนังสือ

เป็นระบบที่ใช้ในร้านเช่าหนังสือเพื่อบันทึกข้อมูลการยืมหนังสือของลูกค้าแต่ละคน โดยระบบนี้สามารถเพิ่มข้อมูลหนังสือ เพิ่มโปรโมชันการยืมหนังสือ บันทึกข้อมูลการยืมหนังสือ และคำนวณเงินค่ายืมหนังสือตามจำนวนวันที่ลูกค้าเลือก ซึ่งตัวอย่างกรณีศึกษานี้จะใช้ทดสอบเฉพาะฟังก์ชันการคำนวณเงินค่ายืมหนังสือ ด้วยการเปลี่ยนแปลงสคีมาฐานข้อมูลเพียงรูปแบบเดียว คือการเปลี่ยนคีย์หลักของตารางฐานข้อมูล เพื่อตรวจสอบว่าเครื่องมือสามารถวิเคราะห์หาผลกระทบต่อซอร์สโค้ดได้และปรับปรุงแก้ไขค่าคีย์ข้อมูลให้ถูกต้องตามการเปลี่ยนแปลงที่เกิดขึ้นในสคีมาฐานข้อมูล โดยมีรายละเอียดดังต่อไปนี้

1.1. การนำเข้าไฟล์ข้อมูลตั้งต้น

เริ่มจากผู้ใช้นำเข้าไฟล์ข้อมูลทั้งหมด 4 ไฟล์ได้แก่ ไฟล์กำหนดค่า ไฟล์ซอร์สโค้ดเพอซิสเทินไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และนำเข้าไฟล์กรณีทดสอบ เพื่อนำสกัดไปวิเคราะห์หาผลกระทบที่เกิดขึ้นต่อซอร์สโค้ดและกรณีทดสอบ รายละเอียดดังรูปที่ ข-1 ถึง ข-4

เมื่อนำเข้าไฟล์ข้อมูลเสร็จแล้ว เครื่องมือจะเชื่อมต่อฐานข้อมูลโปรแกรมโดยอัตโนมัติผ่านไฟล์กำหนดค่า ซึ่งภายในไฟล์กำหนดค่าจะประกอบไปด้วยตำแหน่งที่อยู่ของเว็บ (URL) ชื่อผู้ใช้งาน (Username) และรหัสผ่าน (Password) จากนั้นตรวจสอบข้อมูลชื่อตาราง หากพบเจอชื่อตารางที่ตรงกัน เครื่องมือจะสกัดสคีมาฐานข้อมูลจากตารางที่เกี่ยวข้อง ดังรูปที่ ข-5 เพื่อมาจัดเก็บในฐานข้อมูลของเครื่องมือ ดังแสดงในรูปที่ ข-6


```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "
3  <hibernate-configuration>
4    <session-factory>
5          <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
6          <property name="connection.url">jdbc:mysql://localhost:3307/bookstore</property>
7          <property name="connection.username">root</property>
8          <property name="connection.password"></property>
9          <property name="show_sql">true</property>
10         <mapping class="hibernate.test.dto.EmployeeEntity" />
11     </session-factory>
12 </hibernate-configuration>
```

รูปที่ ข-1 ไฟล์กำหนดค่าของระบบยืมหนังสือ

```java
package net.codejava.hibernate.Persistance;
import java.util.Date;

@Entity
@Table(name = "BOOK_BORROW", uniqueConstraints = { @UniqueConstraint(columnNames = "BORROW_ID")})
public class Book_Borrow
{
    public Book_Borrow() { }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "BOOK_ID", unique=true, nullable=true, length=6)
    private int bookId;

    @Column(name = "BORROW_ID", unique=false, nullable=true, length=6)
    private int borrowId;

    @Column(name = "USER_ID", unique=false, nullable=true, length=6)
    private int userId;

    @Column(name = "START_DATE", unique=false, nullable=true)
    private Date startDate;

    @Column(name = "END_DATE", unique=false, nullable=true)
    private Date endDate;;

    @Column(name = "NUMBER_OF_BORROW_DAY", unique=false, nullable=true, length=2)
    private int borrowDay;


    public int getId() {
        return borrowId;
    }

    public void setId(int borrowId) {
        this.borrowId = borrowId;
    }

    public int getBookId() {
        return bookId;
    }

    public void setBookId(int bookId) {
        this.bookId = bookId;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public Date getStartDate() {
        return startDate;
    }

    public void setStartDate(Date startDate) {
        this.startDate = startDate;
    }

    public Date getEndDate() {
        return endDate;
    }

    public void setEndDate(Date endDate) {
        this.endDate = endDate;
    }

    public int getBorrowDay() {
        return borrowDay;
    }

    public void setBorrowDay(int borrowDay) {
        this.borrowDay = borrowDay;
    }

}
```

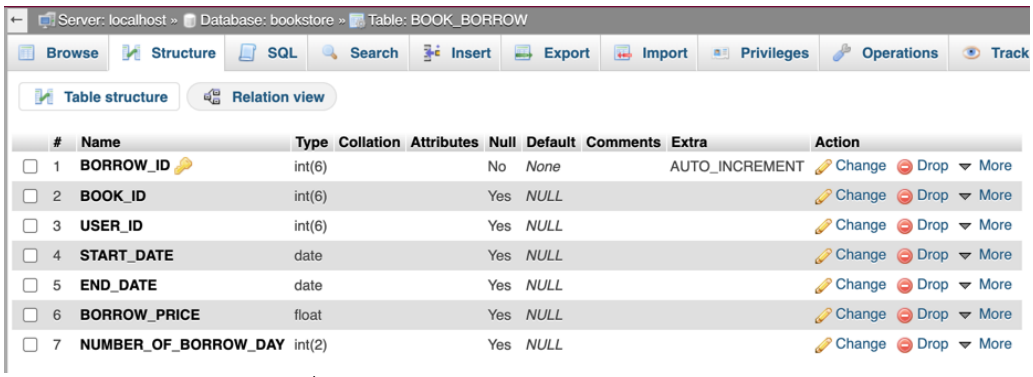
รูปที่ ข-2 ไฟล์ซอร์สโค้ดเพอซิสเทินของระบบยืมหนังสือ

```java
package net.codejava.hibernate;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.Date;
import java.util.List;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import net.codejava.hibernate.Persistance.Book;
import net.codejava.hibernate.Persistance.Book_Borrow;

public class BookBorrowManager
{
    private static SessionFactory sessionFactory;
    public static float CalculateBookBorrow(List<Book_Borrow> borrowBookList, List<Book> bookList)
    {
        float totalPrice = 0;
        if(borrowBookList.size()>0 || bookList.size() > 0)
        {
            for(Book_Borrow BBrow : borrowBookList)
            {
                for(Book bookRec : bookList)
                {
                    if(BBrow.getBookId() == bookRec.getId())
                    {
                        int borrowDay = BBrow.getBorrowDay();
                        float ratePerDay = bookRec.getPricePerDay();
                        float price =  borrowDay * ratePerDay;
                        totalPrice += price;
                    }
                }
            }
            System.out.println("--totalPrice--"+totalPrice);
        }
        if(totalPrice > 0)
            return totalPrice;
        else
            return 0;
    }

    public static void CreateBookBorrow(int bookID, int userID, Date startDate, Date endDate)
    {
        Book_Borrow bookBR = new Book_Borrow();
        bookBR.setBookId(bookID);
        bookBR.setUserId(userID);
        bookBR.setStartDate(startDate);
        bookBR.setEndDate(endDate);
        long borrowDay = ChronoUnit.DAYS.between(startDate.toInstant(), endDate.toInstant());
        System.out.println("--borrowDay--"+borrowDay);
        bookBR.setBorrowDay(Integer.parseInt(String.valueOf(borrowDay)));

        Session session = sessionFactory.openSession();
        session.beginTransaction();
        session.save(bookBR);
        session.getTransaction().commit();
        session.close();

    }
}
```

รูปที่ ข-3 ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ของระบบยืมหนังสือ

TC_Number	TC_Path	TC_Input_Name	TC_Input_Type	TC_Input_Size	TC_Input_Value	TC_Expected_Output
BR001	17-18-19-20-21-36-37-39-40-41	borrowBookList	List<Book_Borrow>	0	null	totalPrice = 0
BR002	17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-41	borrowBookList	List<Book_Borrow>	1	not null	totalPrice > 0
BR003	17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-41	borrowDay	int	2	10	totalPrice > 0
BR004	17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-41	ratePerDay	float	4	15.5	totalPrice > 0
BR005	17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-39-40-41	borrowDay	int	2	-5	totalPrice = 0
BR006	17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-39-40-41	ratePerDay	float	2	-25	totalPrice = 0

รูปที่ ข-4 ไฟล์กรณีทดสอบของระบบยืมหนังสือ



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	BORROW_ID	int(6)			No	None		AUTO_INCREMENT	Change Drop More
2	BOOK_ID	int(6)			Yes	NULL			Change Drop More
3	USER_ID	int(6)			Yes	NULL			Change Drop More
4	START_DATE	date			Yes	NULL			Change Drop More
5	END_DATE	date			Yes	NULL			Change Drop More
6	BORROW_PRICE	float			Yes	NULL			Change Drop More
7	NUMBER_OF_BORROW_DAY	int(2)			Yes	NULL			Change Drop More

รูปที่ ข-5 สคีมาฐานข้อมูลของระบบยืมหนังสือ

Schema

ID	Attribute Name	Data Type	Data Size	Primary Key
497	BORROW_ID	int	6	PRI
498	BOOK_ID	int	6	
499	USER_ID	int	6	
500	START_DATE	date	0	
501	END_DATE	date	0	
502	NUMBER_OF_BORROW_DAY	int	2	

Persistent Class

```
1  package net.codejava.hibernate.Persistance;
2  import java.util.Date;
3
4  import javax.persistence.*;
5
6  @Entity
7  @Table(name = "BOOK_BORROW", uniqueConstraints = { @UniqueConstrain
8  public class Book_Borrow
9  {
10     public Book_Borrow() { }
11
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     @Column(name = "BOOK_ID", unique=true, nullable=true, length=6)
15     private int bookId;
16
17     @Column(name = "BORROW_ID", unique=false, nullable=true, length
18     private int borrowId;
19
20     @Column(name = "USER_ID", unique=false, nullable=true, length=6
21     private int userId;
22
23     @Column(name = "START_DATE", unique=false, nullable=true)
24     private Date startDate;
25
26     @Column(name = "END_DATE", unique=false, nullable=true)
27     private Date endDate;;
28
29     @Column(name = "NUMBER_OF_BORROW_DAY", unique=false, nullable=t
```

JAVA Controller Class

```
1  package net.codejava.hibernate;
2  import java.io.BufferedReader;
3  import java.io.InputStreamReader;
4  import java.time.LocalDate;
5  import java.time.format.DateTimeFormatter;
6  import java.time.temporal.ChronoUnit;
7  import java.util.Date;
8  import java.util.List;
9  import org.hibernate.Session;
10  import org.hibernate.SessionFactory;
11  import net.codejava.hibernate.Persistance.Book;
12  import net.codejava.hibernate.Persistance.Book_Borrow;
13
14  public class BookBorrowManager
15  {
16     private static SessionFactory sessionFactory;
17     public static float CalculateBookBorrow(List borrowBookList, Li
18     {
19             float totalPrice = 0;
20             if(borrowBookList.size()>0 || bookList.size() > 0)
21             {
22                     for(Book_Borrow BBrow : borrowBookList)
23                     {
24                             for(Book bookRec : bookList)
25                             {
26                                     if(BBrow.getBookId() == bookRec
27                                     {
28                                             int borrowDay = BBrow.g
29                                             float ratePerDay = bookI
```

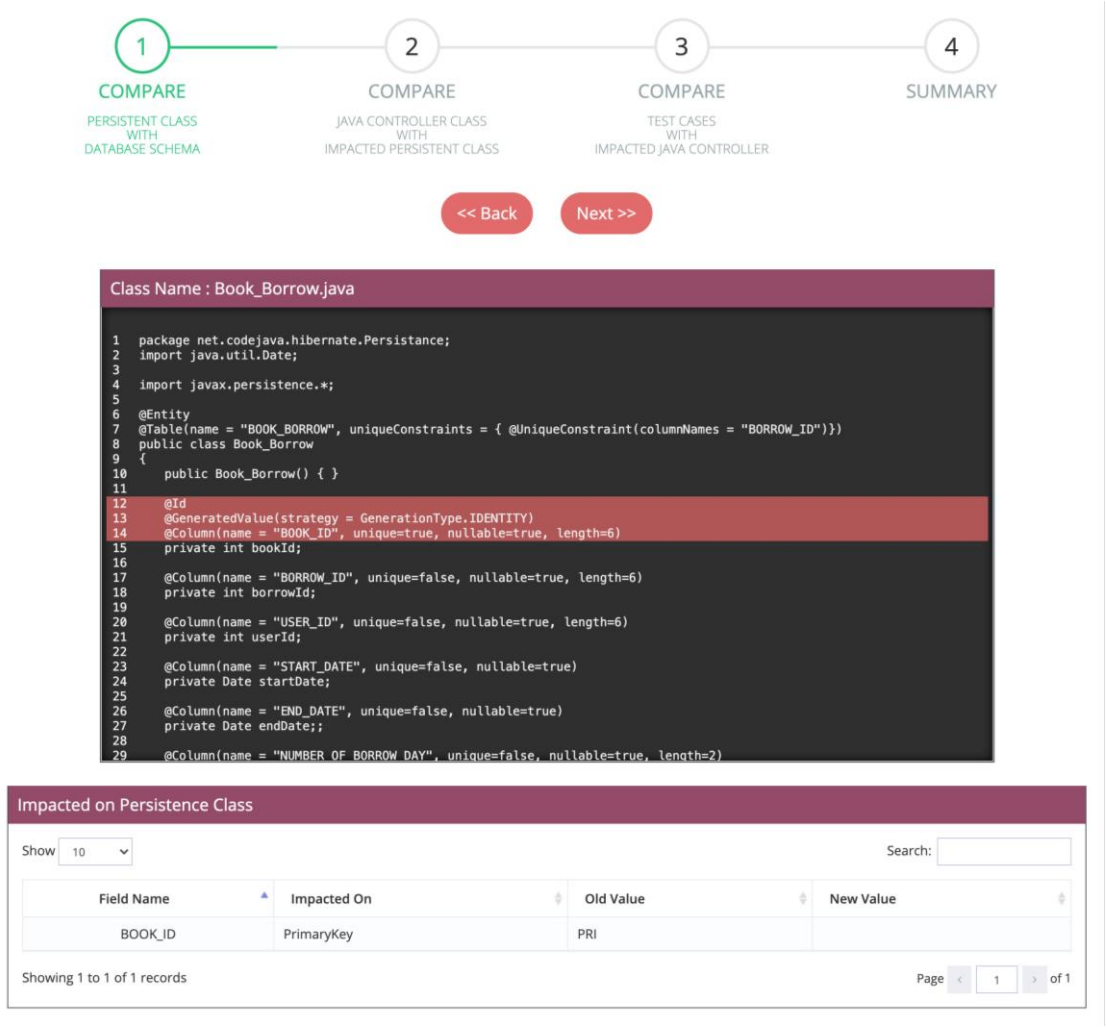
Test Cases

Test Case Number	Input Name	Input Type	Input Size	Input Value	Test Path (Line Number of Code)	Expected Output
BR001	borrowBookList	List	0	null	17-18-19-20-21-36-37-39-40-41	totalPrice = 0
BR002	borrowBookList	List	1	not null	17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-41	totalPrice > 0
BR003	borrowDay	int	2	10	17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-41	totalPrice > 0
BR004	ratePerDay	float	4	15.5	17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-41	totalPrice > 0
BR005	borrowDay	int	2	-5	17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-39-40-41	totalPrice = 0
BR006	ratePerDay	float	2	-25.0	17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-39-40-41	totalPrice = 0

รูปที่ ข-6 ไฟล์ข้อมูลตั้งต้นของระบบยืมหนังสือ

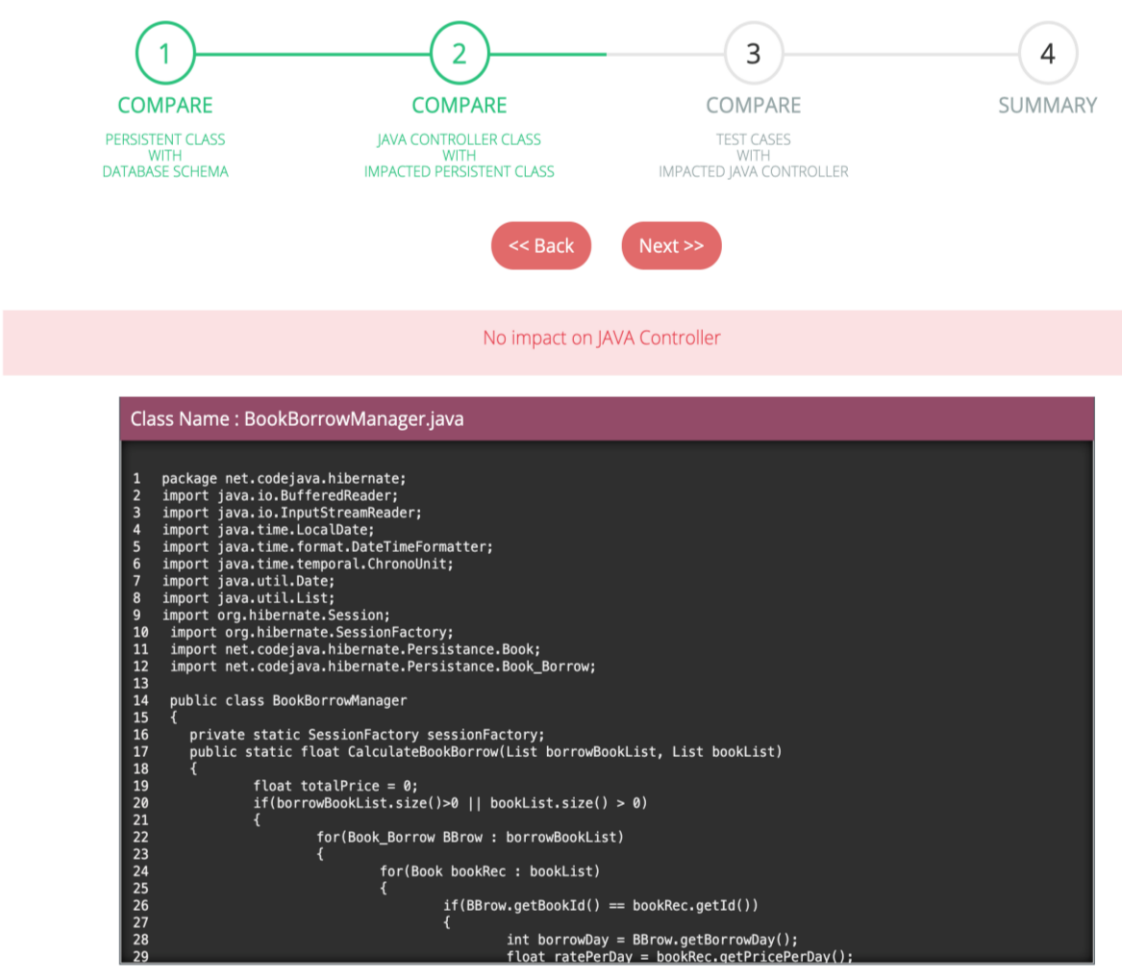
1.2. การวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ

ขั้นตอนที่สอง เครื่องมือจะวิเคราะห์หาผลกระทบต่อซอร์สโค้ดเพอซิสเทิน โดยทำการเปรียบเทียบข้อมูลจากสคีมาฐานข้อมูลของระบบยืมหนังสือกับข้อมูลที่สกัดมาจากไฟล์ซอร์สโค้ดเพอซิสเทิน แสดงดังรูปที่ ข-7 ซึ่งจากรูปพบว่าไฟล์ซอร์สโค้ดเพอร์ซิสเทินได้รับผลกระทบ เครื่องมือจึงได้ทำการไฮไลท์เพื่อให้ผู้ใช้ได้ทราบถึงตำแหน่งที่ได้รับผลกระทบ พร้อมทั้งบอกรายละเอียดในตารางแสดงผลกระทบเพื่ออธิบายเพิ่มเติมให้ผู้ใช้ได้ทราบว่าข้อมูลถูกกระทบในส่วนไหน จากนั้นผู้ใช้จะทำการกดปุ่ม Next เพื่อตรวจสอบผลกระทบที่จะเกิดขึ้นต่อซอร์สโค้ดคอนโทรลเลอร์ต่อไป



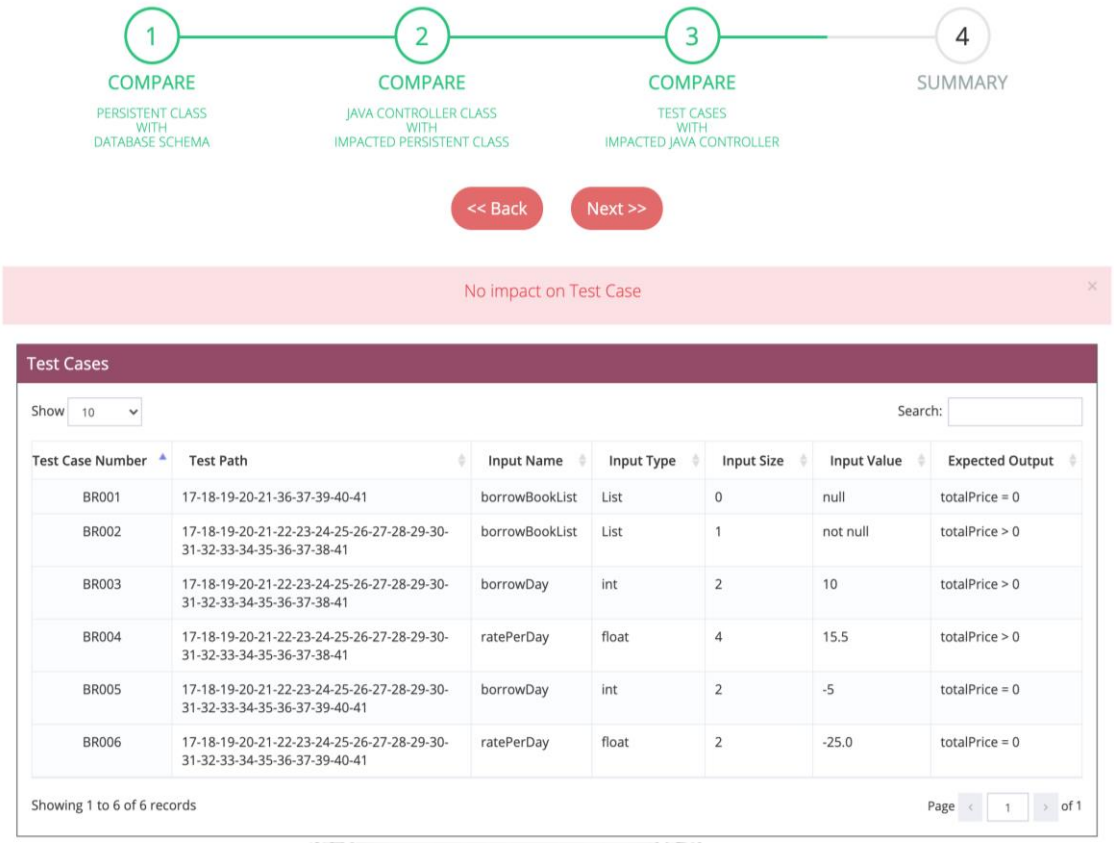
รูปที่ ข-7 ผลการวิเคราะห์ผลกระทบจากไฟล์ซอร์สโค้ดเพอซิสเทินของระบบยืมหนังสือ

หลังจากที่เครื่องมือตรวจสอบแล้วพบว่า การเปลี่ยนแปลงคีย์หลักของตาราง ไม่ได้ส่งผลกระทบต่อซอร์สโค้ดคอนโทรลเลอร์ ดังนั้นเครื่องมือจึงแสดงข้อความเพื่อบอกให้ผู้ใช้ทราบทางหน้าจอ ดังแสดงในรูปที่ ข-8



รูปที่ ข--8 ผลการวิเคราะห์ผลกระทบจากไฟล์ซอร์สโค้ดคอนโทรลเลอร์ของระบบยืมหนังสือ

เมื่อซอร์สโค้ดคอนโทรลเลอร์ไม่ได้รับผลกระทบ ดังนั้นไฟล์กรณีทดสอบจึงไม่ได้รับผลกระทบเช่นกัน เนื่องจากไฟล์กรณีทดสอบที่นำเข้ามามีความสอดคล้องกับซอร์สโค้ดคอนโทรลเลอร์ เครื่องมือจึงแสดงข้อความเพื่อบอกให้ผู้ใช้ทราบทางหน้าจอ ดังแสดงในรูปที่ ข-9



รูปที่ ข-9 ผลการวิเคราะห์ผลกระทบจากไฟล์กรณีทดสอบของระบบยืมหนังสือ

ดังนั้นจากผลการวิเคราะห์ผลกระทบต่อไฟล์ซอร์สโค้ดและกรณีทดสอบที่ได้ สรุปได้ว่าหากมีการแก้ไขคีย์หลักของตารางในสคีมาฐานข้อมูล จะส่งผลกระทบต่อไฟล์ซอร์สโค้ดเพอซิสเทินไฟล์เดียวเท่านั้น โดยไฟล์ซอร์สโค้ดคอนโทรลเลอร์และไฟล์กรณีทดสอบจะไม่ได้รับผลกระทบ ดังแสดงในรูปที่ ข-10

รูปที่ ข-10 สรุปผลการวิเคราะห์ผลกระทบซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบของระบบยืมหนังสือ

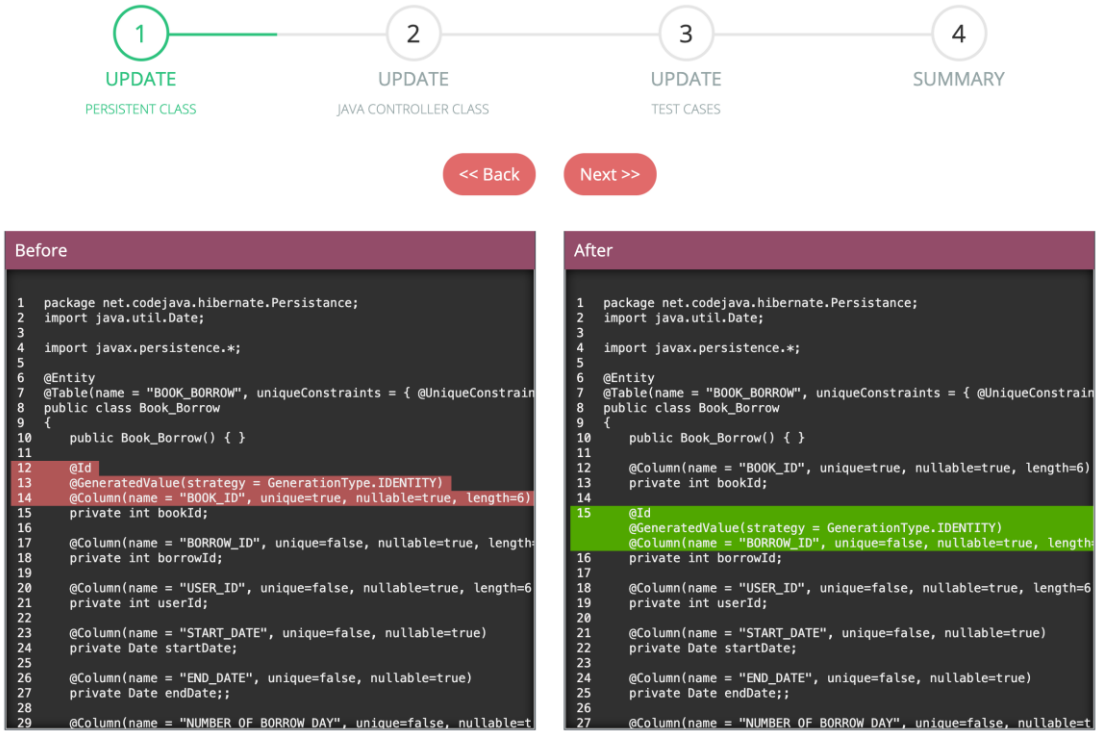
1.3. การปรับปรุงแก้ไขซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ

ขั้นตอนต่อมาเป็นขั้นตอนการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบ โดยผู้ใช้จะทำการกดปุ่ม Update Impacted Files เพื่อเริ่มการทำงาน โดยจะทำการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทิน ดังนี้

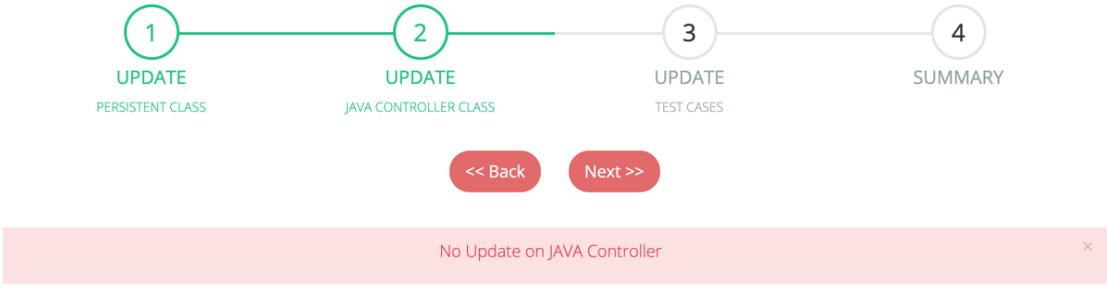
(1.) นำซอร์สโค้ดที่ประกาศคีย์หลักออกจากฟิลด์ BOOK_ID

(2.) เพิ่มซอร์สโค้ดที่ใช้สำหรับประกาศคีย์หลักให้กับฟิลด์ BORROW_ID

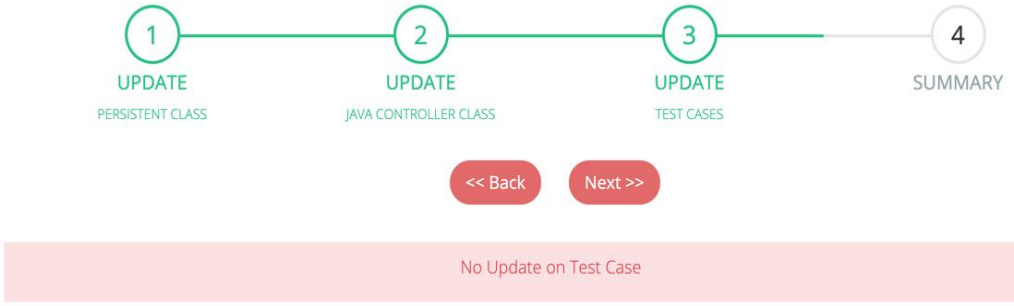
จากรูปที่ ข-11 เป็นผลลัพธ์ที่ได้จากเครื่องมือหลังจากการปรับปรุงแก้ไขซอร์สโค้ดเพอซิสเทินโดยเครื่องมือจะทำการแสดงข้อมูลก่อนและหลังการแก้ไข พร้อมทั้งไฮไลท์สีเพื่อแสดงให้ผู้ใช้ทราบถึงตำแหน่งที่ได้รับผลกระทบและตำแหน่งที่ได้รับการแก้ไข และเนื่องจากไฟล์ซอร์สโค้ดคอนโทรลเลอร์และกรณีทดสอบไม่ได้รับผลกระทบจากการเปลี่ยนแปลงสคีมาฐานข้อมูล เครื่องมือจึงไม่ทำการปรับปรุงแก้ไขให้ แต่จะสดงข้อความให้ผู้ใช้ได้ทราบผ่านทางหน้าจอ ดังแสดงในรูปที่ *ก-12 และ ข-13 ตามลำดับ*



รูปที่ ข-11 ผลการปรับปรุงแก้ไขของไฟล์ซอร์สโค้ดเพอซิสเทินของระบบยืมหนังสือ



รูปที่ ข-12 ข้อความกรณีไม่พบซอร์สโค้ดคอนโทรลเลอร์ที่จะทำการปรับปรุงแก้ไข



รูปที่ ข-13 ข้อความกรณีไม่พบกรณีทดสอบที่จะทำการปรับปรุงแก้ไข*ของระบบยืมหนังสือ*

2. กรณีศึกษาระบบจัดการพนักงาน

ระบบจัดการพนักงาน เป็นระบบที่ใช้ในการจัดเก็บข้อมูลของพนักงานในบริษัท โดยระบบนี้สามารถ เพิ่มข้อมูลพนักงานใหม่ จัดสรรทีมโปรเจคให้กับพนักงาน จัดสรรหัวหน้าให้กับพนักงาน จัดการข้อมูลสวัสดิการของพนักงาน และจัดการข้อมูลเงินเดือนของพนักงาน ซึ่งตัวอย่างกรณีศึกษานี้จะใช้ฟังก์ชันของการจัดสรรหัวหน้าให้กับพนักงานมาทดสอบ โดยการเปลี่ยนแปลง 3 รูปแบบในโครงสร้างสคีมาฐานข้อมูล ดังนี้ แก้ไขประเภทข้อมูลของฟิล แก้ไขขนาดข้อมูลของฟิลด์ และลบฟิลด์ เพื่อตรวจสอบว่าเครื่องมือสามารถวิเคราะห์หาผลกระทบที่เกิดขึ้นในไฟล์ซอร์สโค้ดและกรณีทดสอบได้ โดยมีรายละเอียดดังต่อไปนี้

2.1. การนำเข้าไฟล์ข้อมูลตั้งต้น

เริ่มจากผู้ใช้นำเข้าไฟล์ข้อมูลทั้งหมด 4 ไฟล์ได้แก่ ไฟล์กำหนดค่า ไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และนำเข้าไฟล์กรณีทดสอบ เพื่อนำไปวิเคราะห์หาผลกระทบที่เกิดขึ้นต่อซอร์สโค้ดและกรณีทดสอบ รายละเอียดดังรูปที่ ข-14 ถึง ข-19

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3307/hibernatetest</property>
        <property name="connection.username">root</property>
        <property name="connection.password"></property>
        <property name="show_sql">true</property>
        <mapping class="hibernate.test.dto.EmployeeEntity" />
    </session-factory>
</hibernate-configuration>
```

รูปที่ ข-14 ไฟล์กำหนดค่าของระบบจัดการพนักงาน

```java
package hibernate.test.dto;

import java.io.Serializable;


@Entity
@Table(name = "Employee", uniqueConstraints = {
        @UniqueConstraint(columnNames = "ID"),
        @UniqueConstraint(columnNames = "EMAIL") })
public class EmployeeEntity implements Serializable {

    private static final long serialVersionUID = -1798070786993154676L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "EMPID", unique = true, nullable = false)
    private Integer employeeId;

    @Column(name = "EMAIL", unique = true, nullable = false, length = 100)
    private String email;

    @Column(name = "FIRST_NAME", unique = false, nullable = false, length = 200)
    private String firstName;

    @Column(name = "LAST_NAME", unique = false, nullable = false, length = 100)
    private String lastName;

    @Column(name = "EMP_AGE", unique = false, nullable = true, length = 3)
    private Integer empAge;

    @Column(name = "IS_ACTIVE", unique = false, nullable = true)
    private boolean isActive;

    @Column(name = "MANAGER_NAME", unique = false, nullable = true, length = 100)
    private String managerName;

    @Column(name = "CREATED_DATE", unique = false, nullable = false)
    private Date createdDate;

    public Integer getEmployeeId() {
        return employeeId;
    }
```

รูปที่ ข-15 ไฟล์ซอร์สโค้ดเพอซิสเทินของระบบจัดการพนักงาน

```java
package hibernate.test;

import hibernate.test.dto.EmployeeEntity;

public class TestHibernate
{

    public static void main(String[] args)
    {
        Session session = HibernateUtil.getSessionFactory().openSession();
        session.beginTransaction();

        //create new employee
        String buttonType = request.getParameter("buttonType");
        System.out.println("-------buttonType------"+buttonType);
        String firstName = request.getParameter("FirstName");
        String lastName = request.getParameter("LastName");
        String email = request.getParameter("Email");
        int empAge = Integer.parseInt(request.getParameter("Age"));

        EmployeeEntity emp = new EmployeeEntity();
        emp.setEmail(email);
        emp.setFirstName(firstName);
        emp.setLastName(lastName);
        emp.setCreatedDate(java.time.LocalDate.now());
        emp.setAge(empAge);


        if(buttonType.equalsIgnoreCase("manager1")
        {
            emp.setManagerName("Kimberry");
        }
        else
        {
            emp.setManagerName("Mark Prin");
        }
        System.out.println("-------emp------"+emp);
        session.save(emp);
        session.getTransaction().commit();
        HibernateUtil.shutdown();
    }
}

```

รูปที่ ข-16 ไฟล์ซอร์สโค้ดคอนโทรลเลอร์*ของระบบจัดการพนักงาน*

TC_Number	TC_Path	TC_Input_Name	TC_Input_Type	TC_Input_Size	TC_Input_Value	TC_Expected_Output
TC001	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	buttonType	String	1	manager1	user manager is Kimberry
TC002	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	firstName	String	20	TestFirstName1	user manager is Kimberry
TC003	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	lastName	String	20	TestLastName1	user manager is Kimberry
TC004	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	email	String	30	user1@test.com	user manager is Kimberry
TC005	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	empAge	Integer	2	40	user manager is Kimberry
TC006	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	buttonType	String	1	manager2	user manager is Mark Prin
TC007	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	firstName	String	20	TestFirstName2	user manager is Mark Prin
TC008	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	lastName	String	20	TestLastName2	user manager is Mark Prin
TC009	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	email	String	30	user2@test.com	user manager is Mark Prin
TC010	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-34-35-36-37-38-39-40-41-42-43	empAge	Integer	2	65	user manager is Mark Prin

รูปที่ ข-17 ไฟล์กรณีทดสอบ*ของระบบจัดการพนักงาน*

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	EMPID	int(4)			No	*None*		AUTO_INCREMENT
2	EMAIL	varchar(200)	latin1_swedish_ci		No	*None*		
3	FIRST_NAME	varchar(200)	latin1_swedish_ci		No	*None*		
4	LAST_NAME	varchar(200)	latin1_swedish_ci		No	*None*		
5	EMP_AGE	float			Yes	*NULL*	adding month in this field	
6	STATUS	varchar(50)	latin1_swedish_ci		Yes	*NULL*		
7	MANAGER_NAME	varchar(100)	latin1_swedish_ci		Yes	*NULL*		

รูปที่ ข-18 สคีมาฐานข้อมูลของระบบจัดการพนักงาน

Project Name : Employee-Project1

Impact Analysis | Update Impacted Files | Export Updated Files

Schema

ID	Attribute Name	Data Type	Data Size	Primary Key
378	EMPID	int	4	PRI
379	EMAIL	varchar	200	
380	FIRST_NAME	varchar	200	
381	LAST_NAME	varchar	200	
382	EMP_AGE	float	0	
383	STATUS	varchar	50	
384	MANAGER_NAME	varchar	100	

Persistent Class

```
1  package hibernate.test.dto;
2
3  import java.io.Serializable;
4
5  import javax.persistence.Column;
6  import javax.persistence.Entity;
7  import javax.persistence.GeneratedValue;
8  import javax.persistence.GenerationType;
9  import javax.persistence.Id;
10  import javax.persistence.Table;
11  import javax.persistence.UniqueConstraint;
12
13
14  @Entity
15  @Table(name = "Employee", uniqueConstraints = {
16            @UniqueConstraint(columnNames = "ID"),
17            @UniqueConstraint(columnNames = "EMAIL") })
18  public class EmployeeEntity implements Serializable {
19
20     private static final long serialVersionUID = -17980707869931546
21
22     @Id
23     @GeneratedValue(strategy = GenerationType.IDENTITY)
24     @Column(name = "ID", unique = true, nullable = false)
25     private Integer employeeId;
26
27     @Column(name = "EMAIL", unique = true, nullable = false, length
28     private String email;
29
```

JAVA Controller Class

```
1  package hibernate.test;
2
3  import hibernate.test.dto.EmployeeEntity;
4  import org.hibernate.Session;
5
6  public class TestHibernate
7  {
8
9      public static void main(String[] args)
10     {
11             Session session = HibernateUtil.getSessionFactory().ope
12             session.beginTransaction();
13
14             //create new employee
15             String buttonType = request.getParameter("buttonType");
16             System.out.println("-------buttonType------"+buttonType
17             String firstName = request.getParameter("FirstName");
18             String lastName = request.getParameter("LastName");
19             String email = request.getParameter("Email");
20             int empAge = Integer.parseInt(request.getParameter("Age
21
22             EmployeeEntity emp = new EmployeeEntity();
23             emp.setEmail(email);
24             emp.setFirstName(firstName);
25             emp.setLastName(lastName);
26             emp.setCreatedDate(java.time.LocalDate.now());
27             emp.setAge(empAge);
28
29
```

Test Cases

Test Case Number	Input Name	Input Type	Input Size	Input Value	Test Path (Line Number of Code)	Expected Output
TC001	buttonType	String	1	manager1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC002	firstName	String	20	TestFirstName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry
TC003	lastName	String	20	TestLastName1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-38-39-40-41-42-43	user manager is Kimberry

รูปที่ ข-19 ไฟล์ข้อมูลตั้งต้นของระบบจัดการพนักงาน

2.2. การวิเคราะห์ผลกระทบต่อซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ

ขั้นตอนที่สอง เครื่องมือจะวิเคราะห์หาผลกระทบต่อซอร์สโค้ดเพอซิสเทิน โดยทำการเปรียบเทียบข้อมูลจากสคีมาฐานข้อมูลของระบบจัดการพนักงานกับข้อมูลที่สกัดมาจากไฟล์ซอร์สโค้ดเพอซิสเทิน แสดงดังรูปที่ ข-20 ซึ่งจากรูปพบว่าไฟล์ซอร์สโค้ดเพอร์ซิสเทินได้รับผลกระทบ โดยได้รับผลกระทบทั้งหมด 6 รายการ ได้แก่ (1) ผลกระทบต่อขนาดของฟิลด์ EMAIL (2) ผลกระทบต่อขนาดของฟิลด์ LAST NAME (3) ผลกระทบต่อขนาดของฟิลด์ EMP AGE (4) ผลกระทบต่อชนิดของข้อมูลฟิลด์ EMP_AGE (5) ผลกระทบต่อชื่อฟิลด์ CREATED_DATE (6) ผลกระทบต่อชื่อฟิลด์ IS_ACTIVE เครื่องมือจึงได้ทำการไฮไลท์เพื่อให้ผู้ใช้ได้ทราบถึงตำแหน่งที่ได้รับผลกระทบ พร้อมทั้งบอกรายละเอียดในตารางแสดงผลกระทบเพื่ออธิบายเพิ่มเติมให้ผู้ใช้ได้ทราบว่าข้อมูลถูกกระทบในส่วนไหน จากนั้นผู้ใช้จะทำการกดปุ่ม Next เพื่อตรวจสอบผลกระทบที่จะเกิดขึ้นต่อซอร์สโค้ดคอนโทรลเลอร์ต่อไป

Class Name : EmployeeEntity.java

```
21
22      @Id
23      @GeneratedValue(strategy = GenerationType.IDENTITY)
24      @Column(name = "ID", unique = true, nullable = false)
25      private Integer employeeId;
26
27      @Column(name = "EMAIL", unique = true, nullable = false, length = 100)
28      private String email;
29
30      @Column(name = "FIRST_NAME", unique = false, nullable = false, length = 200)
31      private String firstName;
32
33      @Column(name = "LAST_NAME", unique = false, nullable = false, length = 100)
34      private String lastName;
35
36      @Column(name = "EMP_AGE", unique = false, nullable = true, length = 3)
37      private Integer empAge;
38
39      @Column(name = "IS_ACTIVE", unique = false, nullable = true)
40      private boolean isActive;
41
42      @Column(name = "MANAGER_NAME", unique = false, nullable = true, length = 100)
43      private String managerName;
44
45      @Column(name = "CREATED_DATE", unique = false, nullable = false)
46      private Date createdDate;
47
48      public Integer getEmployeeId() {
49              return employeeId;
50      }
```

Impacted on Persistence Class

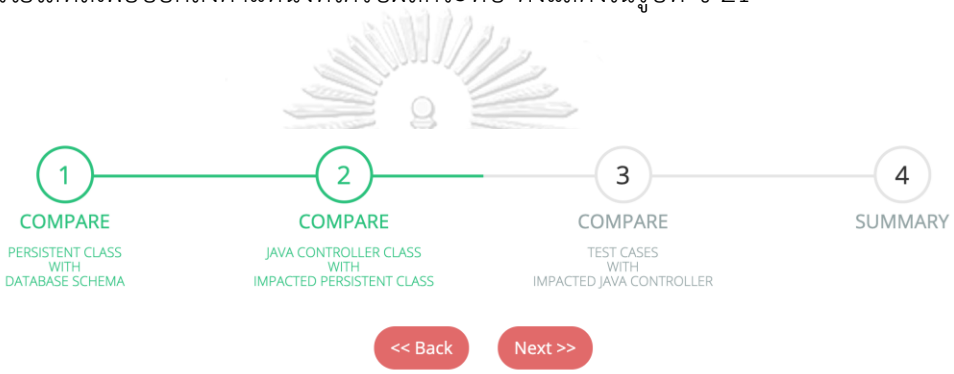
Show 10 Search:

Field Name	Impacted On	Old Value	New Value
CREATED_DATE	Field_Name	CREATED_DATE	
EMAIL	Field_Size	100	200
EMP_AGE	Field_Size	3	0
EMP_AGE	Field_Type	Integer	float
IS_ACTIVE	Field_Name	IS_ACTIVE	
LAST_NAME	Field_Size	100	200

Showing 1 to 6 of 6 records Page 1 of 1

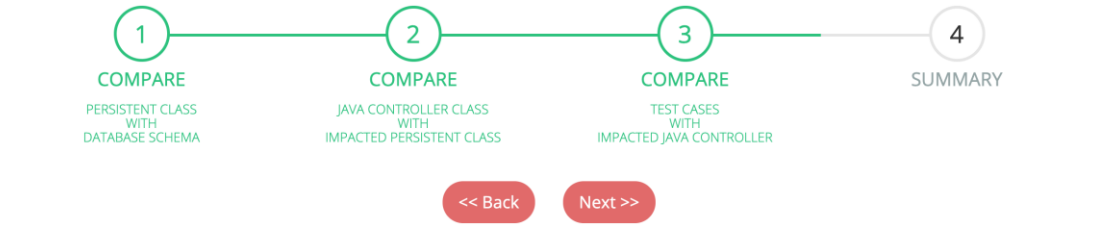
รูปที่ ข-20 ผลการวิเคราะห์ผลกระทบจากไฟล์ซอร์สโค้ดเพอซิสเทินของระบบจัดการพนักงาน

หลังจากที่เครื่องมือตรวจสอบแล้วพบว่ามีผลกระทบต่อซอร์สโค้ดเพอซิสเทิน เครื่องมือจะทำการตรวจสอบไฟล์ของซอร์สโค้ดคอนโทรลเลอร์เป็นลำดับถัดมา ซึ่งพบว่าไฟล์ซอร์สโค้ดคอนโทรลเลอร์ได้รับผลกระทบทั้งหมด 3 รายการ ได้แก่ (1) ผลกระทบต่อหมายเลขบรรทัด 20 เนื่องจากตัวแปร empAge ที่ถูกกำหนดค่าไว้ในฟังก์ชัน setAge() ได้รับผลกระทบจากซอร์สโค้ดเพอซิสเทิน (2) กระทบต่อหมายเลขบรรทัด 26 เนื่องจากฟังก์ชัน setCreateDate() ได้รับผลกระทบจากซอร์สโค้ดเพอซิสเทิน (3) ผลกระทบต่อหมายเลขบรรทัด 27 เนื่องจากฟังก์ชัน setAge() ได้รับผลกระทบจากซอร์สโค้ดเพอซิสเทิน ดังนั้นเครื่องมือจึงแสดงผลกระทบทั้งหมดผ่านทางหน้าจอด้วยการไฮไลท์สีเพื่อบอกถึงตำแหน่งที่ได้รับผลกระทบ ดังแสดงในรูปที่ ข-21



รูปที่ ข-21 ผลการวิเคราะห์ผลกระทบจากไฟล์ซอร์สโค้ดคอนโทรลเลอร์ของระบบจัดการพนักงาน

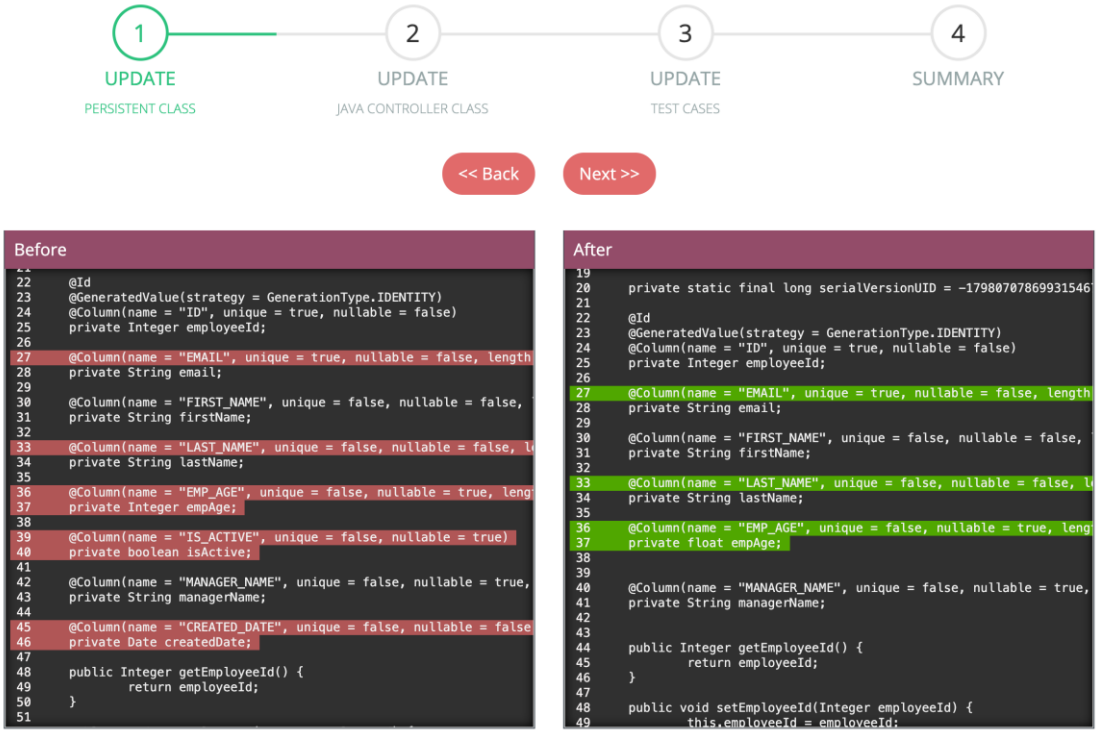
เมื่อซอร์สโค้ดคอนโทรลเลอร์ได้รับผลกระทบ ดังนั้นไฟล์กรณีทดสอบจึงได้รับผลกระทบเช่นกัน เนื่องจากไฟล์กรณีทดสอบที่นำเข้ามามีความสอดคล้องกับซอร์สโค้ดคอนโทรลเลอร์ โดยไฟล์กรณีทดสอบได้รับผลกระทบทั้งหมด 10 รายการ เนื่องจากหมายเลขบรรทัด 20, 26 และ 27 ปรากฏอยู่ทุกหมายเลขทางเดินทดสอบ ดังแสดงในรูปที่ ข-22



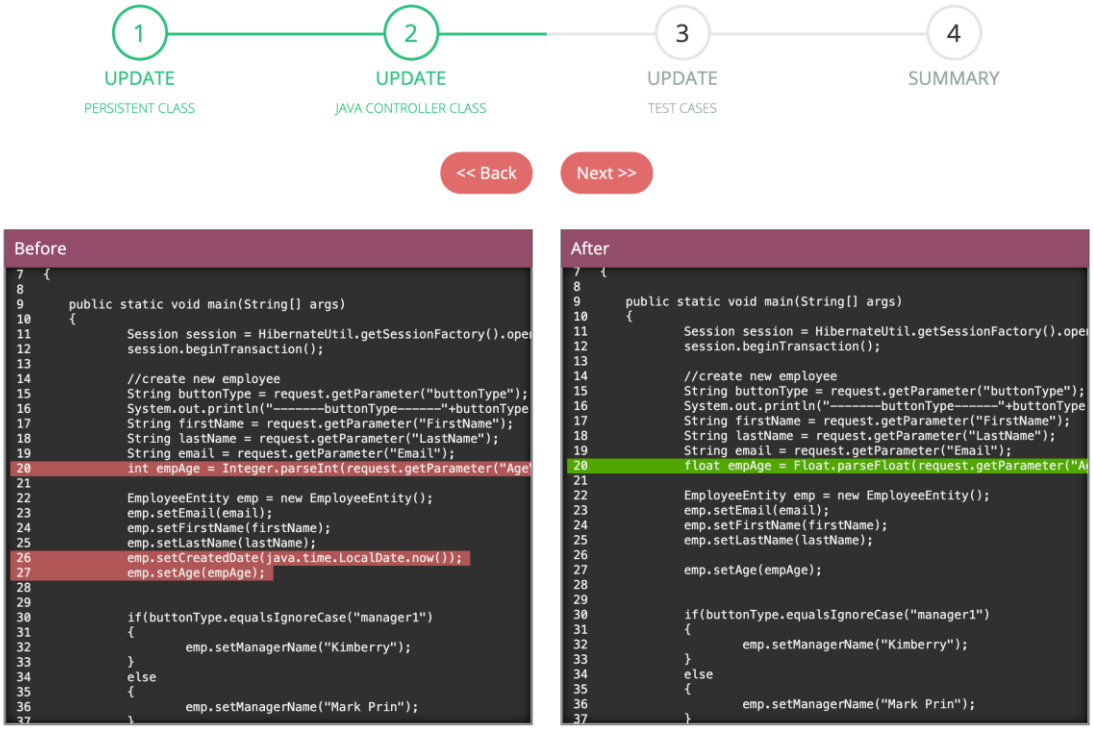
รูปที่ ข-22 ผลการวิเคราะห์ผลกระทบจากไฟล์กรณีทดสอบของระบบจัดการพนักงาน

2.3. การปรับปรุงแก้ไขซอร์สโค้ดไฮเบอร์เนตและกรณีทดสอบ

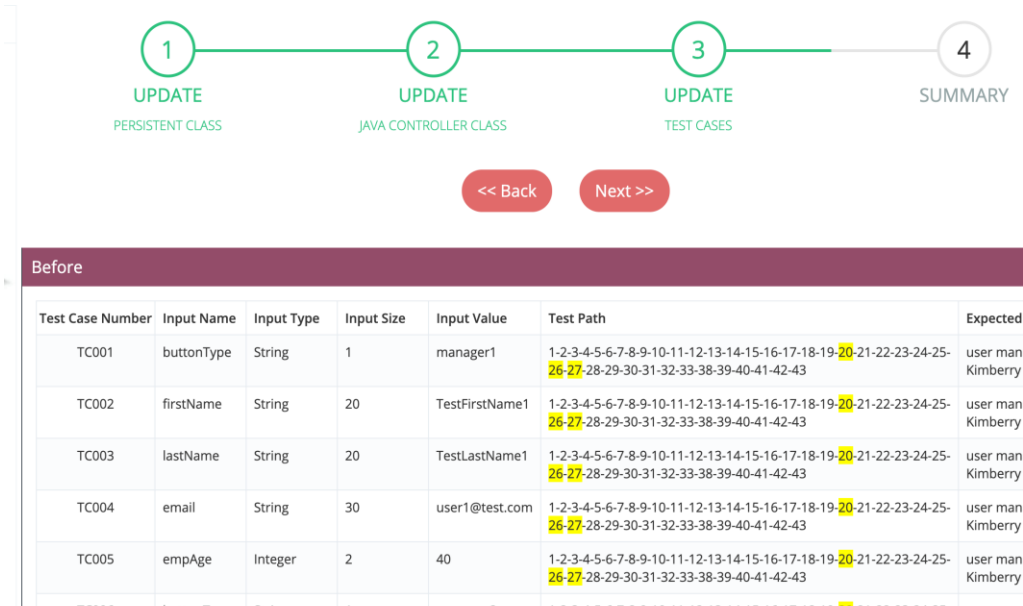
ขั้นตอนต่อมาเป็นขั้นตอนการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดและกรณีทดสอบที่ได้รับผลกระทบ ผู้ใช้จะทำการกดปุ่ม Update Impacted Files เพื่อเริ่มการทำงาน โดยจะทำการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทินก่อนเป็นอันดับแรก จากนั้นจะปรับปรุงแก้ไขไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และแก้ไขกรณีทดสอบเป็นลำดับสุดท้าย โดยเครื่องมือจะทำการแสดงข้อมูลก่อนและหลังการแก้ไข พร้อมทั้งไฮไลท์สีเพื่อแสดงให้ผู้ใช้ทราบถึงตำแหน่งที่ได้รับผลกระทบและตำแหน่งที่ได้รับการแก้ไข ดังแสดงในรูปที่ ข-23 ถึง ข-25 ตามลำดับ โดยหลังจากที่ผู้ใช้ได้ปรับปรุงแก้ไขไฟล์ซอร์สโค้ดและกรณีทดสอบเสร็จเรียบร้อยแล้ว ผู้ใช้สามารถนำออกไฟล์ไฟล์ข้อมูลในรูปแบบซิปไฟล์ เพื่อนำไปใช้งานต่อได้



รูปที่ ข-23 ผลการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดเพอซิสเทินของระบบจัดการพนักงาน



รูปที่ ข-24 ผลการปรับปรุงแก้ไขไฟล์ซอร์สโค้ดคอนโทรลเลอร์ของระบบจัดการพนักงาน



รูปที่ ข-25 ผลการปรับปรุงแก้ไขกรณีทดสอบของระบบจัดการพนักงาน

3. กรณีศึกษาระบบสั่งซื้อสินค้า

ระบบสั่งซื้อสินค้า เป็นระบบที่ใช้สำหรับสั่งซื้อสินค้าออนไลน์ โดยระบบสามารถ จัดการข้อมูลสินค้า จัดการข้อมูลคำสั่งซื้อสินค้า จัดการข้อมูลลูกค้า คำนวณราคาการสั่งซื้อสินค้า จัดการข้อมูลการจ่ายเงิน รวมไปถึงจัดการข้อมูลใบเสร็จของลูกค้า ตัวอย่างกรณีศึกษานี้จะใช้ฟังก์ชันการจัดการข้อมูลการจ่ายเงินมาทดสอบโดยการเปลี่ยนชื่อตารางในสคีมาฐานข้อมูล เพื่อตรวจสอบว่าเครื่องมือสามารถวิเคราะห์ได้ว่าไม่พบข้อมูลตาราง และแสดงชื่อตารางที่ใกล้เคียงกันให้ผู้ใช้ได้รับทราบ โดยมีรายละเอียดดังต่อไปนี้

3.1. การนำเข้าไฟล์ข้อมูลตั้งต้น

เริ่มจากผู้ใช้นำเข้าไฟล์ข้อมูลทั้งหมด 4 ไฟล์ได้แก่ ไฟล์กำหนดค่า ไฟล์ซอร์สโค้ดเพอซิสเทิน ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ และนำเข้าไฟล์กรณีทดสอบ เพื่อนำไปวิเคราะห์หาผลกระทบที่เกิดขึ้นต่อซอร์สโค้ดและกรณีทดสอบ รายละเอียดดังรูปที่ ข-26 ถึง ข-30

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3307/shoppingonline</property>
        <property name="connection.username">root</property>
        <property name="connection.password"></property>
        <property name="show_sql">true</property>
        <mapping class="hibernate.test.dto.EmployeeEntity" />
    </session-factory>
</hibernate-configuration>
```

รูปที่ ข-26 ไฟล์กำหนดค่า*ของระบบสั่งซื้อสินค้า*

```java
@Entity
@Table(name = "customer")
public class CustomerEntity implements Serializable {


    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "customer_id", unique = true, nullable = false, length = 3)
    private Integer customer_id;

    @Column(name = "customer_name", unique = false, nullable = false, length = 50)
    private String customer_name;

    @Column(name = "customer_lastname", unique = false, nullable = false, length = 50)
    private String customer_lastname;

    @Column(name = "customer_address", unique = false, nullable = false, length = 100)
    private String customer_address;

    @Column(name = "customer_tel", unique = false, nullable = false, length = 10)
    private String customer_tel;

    @Column(name = "customer_paid", unique = false, nullable = false, length = 1)
    private Double customer_paid;

    public Integer getCustomer_id() {
        return customer_id;
    }

    public void setCustomer_id(Integer customer_id) {
        this.customer_id = customer_id;
    }
```

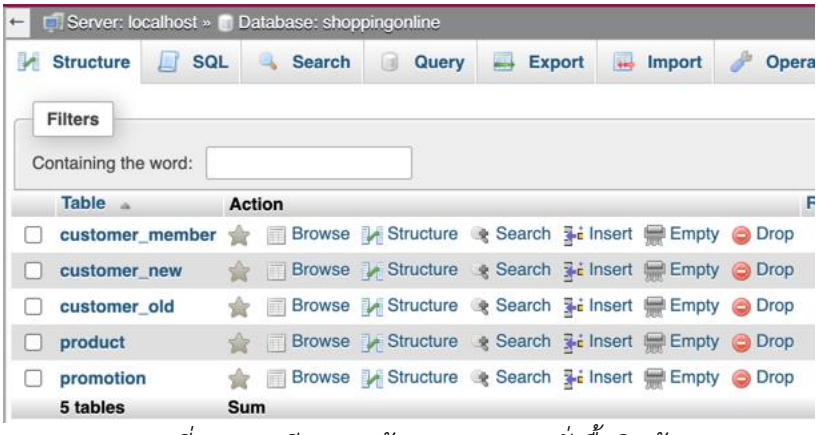
รูปที่ ข-27 ไฟล์ซอร์สโค้ดเพอซิสเทิน*ของระบบสั่งซื้อสินค้า*

```java
package hibernate.test;

import hibernate.test.dto.CustomerEntity;

public class CheckPaidController
{

    public static void main(String[] args)
    {
        String customerId = request.getParameter("customer_id");
        checkPaid(customerId);
        HibernateUtil.shutdown();
    }

    public static void checkPaid(String customerId)
    {
        Query q = em.createNativeQuery("SELECT a.customer_id, a.customer_name, a.customer_paid "
                + "FROM Customer a WHERE a.customer_id ="+customerId);

        List<CustomerEntity[]> customers = q.getResultList();

        if(customers != null){
            String paidStatus = "";
            for (CustomerEntity[] c : customers){
                if(c.getCustomer_paid())
                    paidStatus = "Paid";
                else
                    paidStatus = "Not Paid";
                System.out.println("Customer: "+ c.getCustomer_name());
                System.out.println("Status: "+ paidStatus);
            }
        }else
        {
            System.out.println("Not found customer");
        }
    }
}
```

รูปที่ ข-28 ไฟล์ซอร์สโค้ดคอนโทรลเลอร์ของระบบสั่งซื้อสินค้า

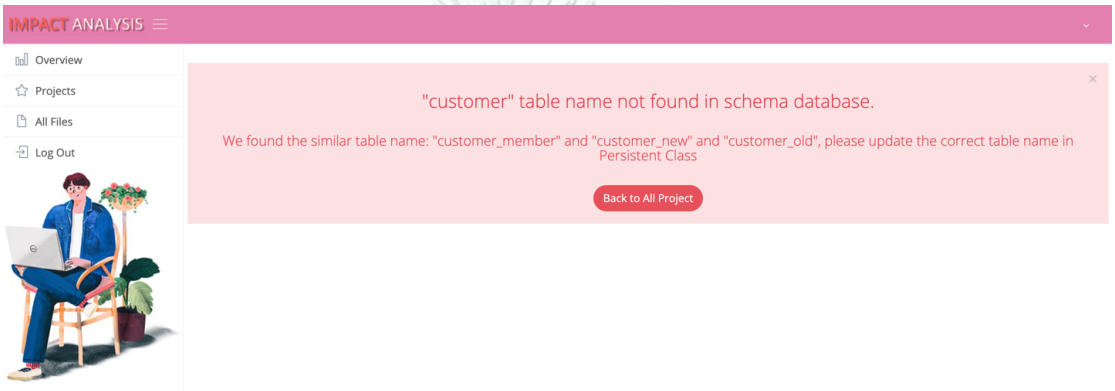
TC_Number	TC_Path	TC_Input_Name	TC_Input_Type	TC_Input_Size	TC_Input_Value	TC_Expected_Output
TC01	1-2-3-4-5-6-7-8-9-10-11-12-16-17-18-19-20-21-22-23-24-25-28-29-30-31-35-36-37-13-14	customerId	String	3	111	Status: Paid
TC02	1-2-3-4-5-6-7-8-9-10-11-12-16-17-18-19-20-21-22-23-26-27-28-29-30-31-35-36-37-13-15	customerId	String	3	222	Status: Not Paid
TC03	1-2-3-4-5-6-7-8-9-10-11-12-16-17-18-19-20-21-31-32-33-34-35-36-37-13-16	customerId	String	3	333	Not found customer

รูปที่ ข-29 ไฟล์กรณีทดสอบของระบบสั่งซื้อสินค้า



รูปที่ ข-30 สคีมาฐานข้อมูลของระบบสั่งซื้อสินค้า

เมื่อผู้ใช้สร้างโครงการใหม่และนำเข้าไฟล์ข้อมูลตั้งต้นทั้ง 4 ไฟล์ เครื่องมือจะตรวจสอบหาข้อมูลชื่อตารางที่สกัดมาจากไฟล์ซอร์สโดเพอซิสเทินในสคีมาฐานข้อมูลของระบบสั่งซื้อสินค้า ในกรณีที่มีการเปลี่ยนแปลงชื่อตารางนั้น เครื่องมือจะไม่สามารถทำการปรับปรุงแก้ไขให้ได้ เนื่องจากเครื่องมือไม่สามารถรับรู้ได้ว่าชื่อตารางเปลี่ยนจากชื่อเดิมเป็นชื่ออะไร แต่เครื่องมือจะทำการแสดงข้อความให้ผู้ใช้ได้ทราบว่าไม่พบข้อมูลของตารางที่ต้องการ และแสดงรายการชื่อตารางที่ใกล้เคียง เพื่อให้ผู้ใช้สามารถปรับปรุงแก้ไขชื่อตารางเป็นตารางที่ถูกต้อง ก่อนนำเข้าสู่เครื่องมือเพื่อวิเคราะห์หาผลกระทบในลำดับต่อไป ดังแสดงในรูปที่ ข-31



รูปที่ ข-31 ผลการวิเคราะห์ซอร์สโค้ดไฮเบอร์เนตของระบบสั่งซื้อสินค้า

ประวัติผู้เขียน

ชื่อ-สกุล	นางสาวอมรรัตน์ ใจมูล
วัน เดือน ปี เกิด	27 ตุลาคม 2533
สถานที่เกิด	จังหวัดเชียงใหม่
วุฒิการศึกษา	ปริญญาตรี หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยเชียงใหม่ ปีการศึกษา 2555
ที่อยู่ปัจจุบัน	75 หมู่ 5 ตำบลสันทราย อำเภอพร้าว จังหวัดเชียงใหม่