เอกสารประกอบการสอน

ENGCE301 - การออกแบบและพัฒนาซอฟต์แวร์

Software Design and Development

สัปดาห์ที่ 3

Agile Requirements & Use Cases + HTML/CSS Layout

สำหรับนักศึกษาวิศวกรรมคอมพิวเตอร์ ชั้นปีที่ 3

ภาคการศึกษาที่ 2/2568

สารบัญ

1. ทบทวนสัปดาห์ที่แล้ว: Requirements Engineering

ในสัปดาห์ที่ผ่านมา เราได้เรียนรู้เกี่ยวกับ Requirements Engineering
ซึ่งเป็นกระบวนการสำคัญในการพัฒนาซอฟต์แวร์ โดยมีเนื้อหาหลักดังนี้

สิ่งที่เราได้เรียนรู้

- Requirements Elicitation (การเก็บความต้องการ) - เทคนิคการสัมภาษณ์ผู้มีส่วนได้ส่วนเสีย การสังเกต และ Brainstorming
- Functional Requirements (ความต้องการเชิงหน้าที่) - สิ่งที่ระบบต้องทำ เช่น ผู้ใช้สามารถเข้าสู่ระบบได้
- Non-Functional Requirements (ความต้องการที่ไม่ใช่เชิงหน้าที่) - คุณภาพของระบบ เช่น Performance, Security, Usability
- การระบุ Stakeholders และ Scope ของโปรเจกต์
- การสร้าง Software Requirements Specification (SRS) เบื้องต้น
- การใช้ HTML5 Semantic Tags (header, nav, main, footer, section, article) เพื่อสร้างโครงสร้างหน้าเว็บ

การเชื่อมโยงกับสัปดาห์นี้

ในสัปดาห์นี้ เราจะนำความต้องการที่เก็บมาได้ มาแปลงเป็น User Stories และ Use Cases ในแนวทาง Agile ซึ่งเป็นรูปแบบที่นิยมใช้ในอุตสาหกรรมซอฟต์แวร์ยุคใหม่
พร้อมกับพัฒนาทักษะการออกแบบหน้าเว็บด้วย CSS Layout

2. ชื่อบทเรียนและวัตถุประสงค์การสอน

หน่วยเรียน: Agile Requirements & Use Cases + HTML/CSS Layout

ชื่อบทเรียน

1.1 Agile Requirements: User Stories & Acceptance Criteria

เรียนรู้การเขียน User Story ตามหลัก INVEST และการกำหนด Acceptance Criteria เพื่อให้ทีมพัฒนาเข้าใจความต้องการได้ชัดเจน

1.2 Use Case Modeling & Scenario Development

ออกแบบ Use Case Diagram เพื่อแสดงปฏิสัมพันธ์ระหว่าง Actor กับระบบ และเขียน Use Case Scenario อย่างละเอียด

1.3 HTML/CSS Layout Fundamentals

เรียนรู้หลักการจัด Layout ด้วย CSS Box Model, Selectors และการใช้ Typography, Color, Spacing ให้สวยงามและใช้งานง่าย

วัตถุประสงค์การสอน

เมื่อจบบทเรียนนี้ นักศึกษาจะสามารถ:

บทเรียนที่ 1.1

1. เขียน User Story ตามหลัก INVEST พร้อม Acceptance Criteria ที่ชัดเจนและวัดผลได้
2. อธิบายความแตกต่างระหว่าง Use Case และ User Story และเลือกใช้ได้อย่างเหมาะสม
3. แปลง Traditional Requirements จาก SRS ให้เป็น Product Backlog ในรูปแบบ Agile

บทเรียนที่ 1.2

4. ออกแบบ Use Case Diagram และระบุ Actor, Use Case หลักของระบบซอฟต์แวร์ได้อย่างถูกต้อง
5. เขียน Use Case Scenario (Main Flow และ Alternative Flow) เพื่ออธิบายการทำงานของระบบ
6. ใช้เครื่องมือ diagrams.net (draw.io) ในการวาด Use Case Diagram อย่างมืออาชีพ

บทเรียนที่ 1.3

7. เข้าใจและใช้ CSS Box Model (margin, border, padding, content) เพื่อควบคุมขนาดและระยะห่างของ elements
8. ใช้ CSS Selectors (element, class, id, descendant) เพื่อจัดรูปแบบหน้าเว็บได้อย่างมีประสิทธิภาพ

9. สร้าง Layout พื้นฐานด้วย CSS โดยใช้ Typography, Color Palette และ Spacing ให้สวยงามและใช้งานง่าย

CLO ที่เกี่ยวข้อง

- **CLO2 - การวิเคราะห์และระบุความต้องการ**
- **CLO3 - การออกแบบระบบและส่วนติดต่อผู้ใช้**
- **CLO4 - การพัฒนาและทดสอบซอฟต์แวร์**
- **CLO6 - การทำงานเป็นทีมและจริยธรรมวิชาชีพ**

3. บทเรียนที่ 1.1: Agile Requirements - User Stories & Acceptance Criteria

3.1 ทำไมต้องใช้ Agile Requirements?

ในโลกของการพัฒนาซอฟต์แวร์สมัยใหม่ ความต้องการของผู้ใช้เปลี่ยนแปลงอย่างรวดเร็ว การเขียน Requirements แบบดั้งเดิมที่มีรายละเอียดหลายสิบหน้าอาจไม่เหมาะสมอีกต่อไป Agile จึงนำเสนอวิธีการใหม่ในการจัดการความต้องการผ่าน User Stories

ข้อดีของ User Stories

- **เข้าใจง่าย - เขียนด้วยภาษาธรรมดาที่ทุกคนในทีมเข้าใจ**
- **มุ่งเน้นผู้ใช้ - เน้นที่คุณค่าที่ผู้ใช้จะได้รับ ไม่ใช่แค่ฟีเจอร์**
- **ยืดหยุ่น - สามารถปรับเปลี่ยนได้ง่ายตามความต้องการที่เปลี่ยนไป**
- **ส่งเสริมการสื่อสาร - เป็นจุดเริ่มต้นของการสนทนาระหว่างทีม**

3.2 โครงสร้างของ User Story

User Story มีรูปแบบมาตรฐานดังนี้:

> ***As a [type of user], I want [an action] so that [a benefit/value]***

ตัวอย่างที่ 1: ระบบ E-Learning

> As a student, I want to bookmark video lectures so that I can easily return to important topics for exam preparation

ตัวอย่างที่ 2: ระบบจัดการโปรเจกต์

> As a project manager, I want to assign tasks to team members with deadlines so that everyone knows their responsibilities and project stays on schedule

ตัวอย่างที่ 3: ระบบ E-Commerce

> As a customer, I want to save items to a wishlist so that I can purchase them later when I have budget

3.3 หลัก INVEST สำหรับ User Story ที่ดี

Bill Wake ได้เสนอหลักการ INVEST เพื่อช่วยให้เราเขียน User Story ที่มีคุณภาพ:

หลัก	คำอธิบาย
Independent	Story ควรเป็นอิสระจากกัน ไม่ขึ้นต่อกัน เพื่อให้พัฒนาได้แยกกัน
Negotiable	ไม่ใช่สัญญาที่ตายตัว แต่เป็นจุดเริ่มต้นของการคุยกัน สามารถปรับเปลี่ยนได้
Valuable	ต้องมีคุณค่าต่อผู้ใช้หรือ Stakeholder ชัดเจน ไม่ใช่แค่ฟีเจอร์ทางเทคนิค
Estimable	สามารถประเมินขนาดงานได้ ถึงแม้จะคร่าวๆ เพื่อวางแผน Sprint
Small	ควรเล็กพอที่จะทำเสร็จใน 1 Sprint (1-2 สัปดาห์) ถ้าใหญ่เกินไปให้แบ่งย่อย
Testable	ต้องทดสอบได้ว่าทำเสร็จหรือยัง ผ่าน Acceptance Criteria ที่กำหนดไว้

3.4 Acceptance Criteria คืออะไร?

Acceptance Criteria คือเงื่อนไขที่ User Story จะต้องผ่าน เพื่อถือว่าเสร็จสมบูรณ์ มักเขียนในรูปแบบ Given-When-Then หรือเป็น Checklist

ตัวอย่าง Acceptance Criteria

User Story:

> As a student, I want to submit assignments online so that I can turn in work from anywhere

Acceptance Criteria (Checklist Format):

- Student can upload file (PDF, DOCX) up to 10MB
- System shows confirmation message after successful upload
- Student can see submission timestamp
- Student cannot submit after deadline
- Professor receives email notification when student submits

Acceptance Criteria (Given-When-Then Format):

> Scenario: Successful file upload Given I am a logged-in student And I have selected a valid PDF file less than 10MB When I click the 'Submit Assignment' button Then the file should be uploaded to the system And I should see a confirmation message 'Assignment submitted successfully' And the submission timestamp should be displayed

3.5 Workshop: แปลง Traditional Requirements เป็น User Stories

ตัวอย่าง Traditional Requirement:

> The system shall provide a search function that allows users to search for products by name, category, or price range, and display results in a paginated format.

แปลงเป็น User Stories:

Story 1:

> As a customer, I want to search for products by name so that I can quickly find what I'm looking for

Story 2:

> As a customer, I want to filter products by category so that I can browse specific types of items

Story 3:

> As a customer, I want to set a price range filter so that I can find products within my budget

Story 4:

> As a customer, I want to see search results in pages of 20 items so that the page loads quickly and I can browse easily

4. บทเรียนที่ 1.2: Use Case Modeling & Scenario Development

4.1 Use Case คืออะไร?

Use Case เป็นเทคนิคในการอธิบายว่าระบบจะทำงานอย่างไร โดยมุ่งเน้นที่การโต้ตอบระหว่าง Actor (ผู้ใช้งานหรือระบบภายนอก) กับระบบ เพื่อบรรลุเป้าหมายบางอย่าง

องค์ประกอบหลักของ Use Case

- **Actor - บุคคลหรือระบบที่โต้ตอบกับระบบของเรา (อยู่นอกระบบ)**
  - ตัวอย่าง: ลูกค้า, ผู้ดูแลระบบ, ระบบชำระเงิน
- **Use Case - ฟังก์ชันหรือบริการที่ระบบให้กับ Actor**
  - ตัวอย่าง: เข้าสู่ระบบ, สั่งซื้อสินค้า, ดูรายงาน
- **System Boundary - ขอบเขตของระบบ แบ่งระหว่างสิ่งที่อยู่ในและนอกระบบ**
- **Relationship - ความสัมพันธ์ระหว่าง Use Cases**
  - Include: Use Case หนึ่งจำเป็นต้องใช้อีก Use Case (เช่น 'สั่งซื้อสินค้า' include 'ตรวจสอบสต็อก')
  - Extend: Use Case เพิ่มเติมที่อาจเกิดขึ้นในบางกรณี (เช่น 'สั่งซื้อสินค้า' extend 'ใช้คูปองส่วนลด')

4.2 Use Case vs User Story

ลักษณะ	Use Case	User Story
ระดับรายละเอียด	ละเอียด มีขั้นตอนชัดเจน	กระชับ เน้นคุณค่า
เป้าหมาย	ภาพรวมระบบ สถาปัตยกรรม	การพัฒนาในแต่ละ Sprint
ความยืดหยุ่น	เป็นทางการ แก้ไขยาก	ยืดหยุ่น เปลี่ยนแปลงง่าย
เหมาะกับ	Waterfall, ระบบซับซ้อน	Agile, Scrum, Startup

สรุป: Use Case และ User Story ไม่ได้เป็นคู่แข่งกัน แต่ใช้เสริมกัน Use Case ดีสำหรับ high-level view ของระบบ ส่วน User Story ดีสำหรับ sprint planning และการพัฒนาแบบ Agile

4.3 การเขียน Use Case Diagram

Use Case Diagram เป็นแผนภาพที่แสดงความสัมพันธ์ระหว่าง Actors และ Use Cases โดยใช้สัญลักษณ์มาตรฐาน UML

สัญลักษณ์ใน Use Case Diagram

สัญลักษณ์	คำอธิบาย
Stick Figure	แสดง Actor (ผู้ใช้หรือระบบภายนอก)
Oval/Ellipse	แสดง Use Case (ฟังก์ชันของระบบ)
Rectangle	แสดง System Boundary (ขอบเขตของระบบ)
Line	แสดงความสัมพันธ์ระหว่าง Actor กับ Use Case
<<include>>	Use Case หนึ่งจำเป็นต้องใช้อีก Use Case (เส้นประ)
<<extend>>	Use Case ขยายความสามารถของอีก Use Case (เส้นประ)

ตัวอย่าง Use Case Diagram: ระบบ E-Learning

ลองพิจารณาระบบ E-Learning ที่มี Actors และ Use Cases ดังนี้:

Actors:

- นักศึกษา (Student)
- อาจารย์ (Instructor)
- ผู้ดูแลระบบ (Admin)

Use Cases:

- เข้าสู่ระบบ (Login)
- ดูรายวิชา (View Courses) - Student & Instructor
- ส่งงาน (Submit Assignment) - Student
- ตรวจงาน (Grade Assignment) - Instructor
- สร้างรายวิชา (Create Course) - Instructor
- จัดการผู้ใช้ (Manage Users) - Admin

Note: ใน Lab เราจะใช้ diagrams.net (draw.io) เพื่อวาด Use Case Diagram จริง

4.4 การเขียน Use Case Scenario

Use Case Scenario คือการอธิบายขั้นตอนการทำงานของ Use Case อย่างละเอียด โดยแบ่งเป็น Main Flow (เส้นทางหลัก) และ Alternative Flows (เส้นทางเลือก)

โครงสร้างของ Use Case Scenario

1. Use Case Name

ชื่อของ Use Case ที่ชัดเจน

2. Actors

ระบุ Primary Actor และ Secondary Actors (ถ้ามี)

3. Preconditions

เงื่อนไขที่ต้องเป็นจริงก่อนเริ่ม Use Case

4. Main Flow

ขั้นตอนหลักที่เกิดขึ้นเมื่อทุกอย่างเป็นไปตามปกติ

5. Alternative Flows

ขั้นตอนเลือกที่อาจเกิดขึ้น (เช่น กรณีผิดพลาด, เลือกทางอื่น)

6. Postconditions

สถานะของระบบหลังจากจบ Use Case

ตัวอย่าง Use Case Scenario: ส่งงาน (Submit Assignment)

Use Case Name	Submit Assignment (ส่งงาน)
Primary Actor	Student (นักศึกษา)
Preconditions	• Student has logged into the system • Assignment deadline has not passed • Student has prepared assignment file
Main Flow	10. Student navigates to course assignments page 11. System displays list of assignments

	12. Student selects assignment to submit 13. System displays upload interface 14. Student clicks 'Choose File' button 15. Student selects file from computer 16. System validates file (type, size) 17. Student clicks 'Submit' button 18. System uploads file and saves submission 19. System displays confirmation with timestamp 20. System sends email notification to instructor
Alternative Flow 1	**Invalid File Type** 21. At step 7, if file type is invalid (not PDF/DOCX) 22. System displays error message 'Invalid file type. Please upload PDF or DOCX only' 23. Return to step 5
Alternative Flow 2	**File Too Large** 24. At step 7, if file size exceeds 10MB 25. System displays error message 'File too large. Maximum size is 10MB' 26. Return to step 5
Alternative Flow 3	**Past Deadline** 27. At step 3, if current time is past deadline 28. System displays message 'Submission deadline has passed. Contact instructor for late submission' 29. Use case ends
Postconditions	• Assignment file is stored in system database

	• Submission timestamp is recorded • Instructor is notified via email • Assignment status changed to 'Submitted'

5. บทเรียนที่ 1.3: HTML/CSS Layout Fundamentals

5.1 CSS Box Model - หัวใจของ Layout

CSS Box Model เป็นแนวคิดพื้นฐานที่สุดในการออกแบบ Layout ทุก element บนหน้าเว็บถูกจัดการเป็น 'กล่อง' ที่ประกอบด้วย 4 ส่วน:

- **Content - เนื้อหาจริงของ element (ข้อความ, รูปภาพ)**
- **Padding - ระยะห่างระหว่าง content กับ border (พื้นที่ว่างด้านใน)**
- **Border - เส้นขอบรอบ element**
- **Margin - ระยะห่างระหว่าง element กับ element อื่น (พื้นที่ว่างด้านนอก)**

ตัวอย่าง CSS Box Model

```
.box {   width: 300px;   padding: 20px;          /* ระยะห่างด้านใน */
border: 2px solid #333;  /* เส้นขอบ */   margin: 15px;          /*
ระยะห่างด้านนอก */ }
```

ความกว้างรวม = width + padding-left + padding-right + border-left + border-right

ในตัวอย่าง: 300 + 20 + 20 + 2 + 2 = 344px

5.2 CSS Selectors - เลือก Element ให้เป็น

CSS Selectors ช่วยให้เราเลือก HTML elements เพื่อจัดรูปแบบได้อย่างแม่นยำ

Selector	ตัวอย่าง CSS	คำอธิบาย
Element	`p { color: blue; }`	เลือก element ทั้งหมด
Class	`.highlight { ... }`	เลือกตาม class attribute
ID	`#header { ... }`	เลือกตาม id (ไม่ซ้ำ)
Descendant	`div p { ... }`	p ที่อยู่ใน div (ลูกหลาน)
Child	`div > p { ... }`	p ที่เป็นลูกโดยตรง
Pseudo-class	`a:hover { ... }`	เมื่อเมาส์ชี้ที่ลิงก์

5.3 Typography - ศิลปะของตัวอักษร

Typography ที่ดีช่วยให้เว็บไซต์อ่านง่าย สวยงาม และสื่อสารได้อย่างมีประสิทธิภาพ

หลักการเลือกฟอนต์

- **Sans-serif (Arial, Helvetica) - ดูสะอาด ทันสมัย เหมาะกับหน้าจอคอมพิวเตอร์**
- **Serif (Times New Roman, Georgia) - ดูเป็นทางการ เหมาะกับเอกสารยาว**
- **Monospace (Courier, Monaco) - เหมาะกับโค้ดและตัวเลข**

ตัวอย่าง Typography CSS

```
body {   font-family: 'Arial', sans-serif;   font-size: 16px;
line-height: 1.6;  /* ระยะห่างบรรทัด */   color: #333; }  h1 {
font-size: 2.5em;  /* 40px */   font-weight: 700;   letter-
spacing: -0.5px;   margin-bottom: 20px; }  p {   margin-
bottom: 15px;   text-align: justify; }
```

5.4 Color Palette - เลือกสีให้ลงตัว

การเลือกสีที่ดีช่วยสร้างอารมณ์และความรู้สึกให้กับเว็บไซต์ ควรมี Color Palette ที่สอดคล้องกัน

หลักการเลือกสี

- **Primary Color - สีหลักของแบรนด์ (1 สี)**
- **Secondary Color - สีเสริมเพื่อสร้างความหลากหลาย (1-2 สี)**
- **Neutral Colors - สีเทา, ขาว, ดำสำหรับพื้นหลังและข้อความ**
- **Accent Color - สีที่ใช้เน้นจุดสำคัญ (ปุ่ม, ลิงก์)**

ตัวอย่าง Color Palette

สี	รหัสสี (Hex)	การใช้งาน
Navy Blue	#2E5090	หัวข้อหลัก, ปุ่มสำคัญ
Blue	#4472C4	ลิงก์, หัวข้อรอง
Green	#70AD47	สถานะสำเร็จ, ปุ่มยืนยัน
Red	#C0504D	ข้อผิดพลาด, คำเตือน
Light Gray	#F4F4F4	พื้นหลัง, กรอบ

สี	รหัสสี (Hex)	การใช้งาน
Dark Gray	#333333	ข้อความหลัก, Footer

5.5 Spacing & Layout - พื้นที่ว่างที่มีค่า

White space (พื้นที่ว่าง) ไม่ใช่พื้นที่ที่สูญเปล่า
แต่เป็นองค์ประกอบสำคัญที่ช่วยให้เว็บอ่านง่ายและดูสะอาดตา

หลักการใช้ Spacing

- ใช้ระบบ spacing ที่สม่ำเสมอ (เช่น 4px, 8px, 16px, 24px, 32px)
- เพิ่ม padding ให้กับ container เพื่อสร้างพื้นที่หายใจ
- ใช้ margin เพื่อแยกกลุ่มเนื้อหาที่ต่างกัน
- ตั้ง line-height อย่างน้อย 1.5 เพื่อให้อ่านง่าย

ตัวอย่าง Layout แบบง่าย

HTML:

```
<div class="container">   <header class="header">     <h1>My
Website</h1>     <nav>       <a href="#">Home</a>       <a
href="#">About</a>       <a href="#">Contact</a>     </nav>
</header>      <main class="content">     <h2>Welcome</h2>
<p>This is the main content area.</p>   </main>      <footer
class="footer">     <p>&copy; 2024 My Website</p>   </footer>
</div>
```

CSS:

```
.container {   max-width: 1200px;   margin: 0 auto;  /* จัดกึ่งกลาง
*/   padding: 20px; }  .header {   background-color: #2E5090;
color: white;   padding: 20px;   margin-bottom: 30px; }
.header nav a {   color: white;   margin-right: 20px;   text-
decoration: none; }  .content {   padding: 30px;   background-
color: #f4f4f4;   margin-bottom: 30px; }  .footer {   text-
align: center;   padding: 20px;   background-color: #333;
color: white; }
```

6. Workshop & Lab Activities (3 ชั่วโมง)

ในช่วงปฏิบัติการ นักศึกษาจะได้ฝึกทักษะที่เรียนมาผ่านกิจกรรมต่อไปนี้

Activity 1: สร้าง Product Backlog (45 นาที)

วัตถุประสงค์:

แปลง SRS ที่เขียนในสัปดาห์ที่แล้วให้เป็น User Stories และจัดทำ Product Backlog บน Trello หรือ Jira

ขั้นตอน:

30. ทบทวน Functional Requirements จาก SRS
31. แปลงแต่ละ requirement เป็น User Story ตามหลัก INVEST
32. เขียน Acceptance Criteria สำหรับแต่ละ User Story (3-5 criteria)
33. สร้าง Board บน Trello หรือ Jira และเพิ่ม User Stories เข้าไป
34. จัดลำดับความสำคัญของ Stories (Priority: High, Medium, Low)

ผลลัพธ์ที่คาดหวัง:

- Product Backlog ที่มีอย่างน้อย 10-15 User Stories
- แต่ละ Story มี Acceptance Criteria ที่ชัดเจน
- Stories ถูกจัดลำดับความสำคัญแล้ว

Activity 2: วาด Use Case Diagram (45 นาที)

วัตถุประสงค์:

ออกแบบ Use Case Diagram สำหรับระบบโดยใช้ diagrams.net (draw.io)

ขั้นตอน:

35. เปิด diagrams.net และเลือก template UML Use Case Diagram
36. ระบุ Actors ทั้งหมดของระบบ (อย่างน้อย 2-3 actors)
37. ระบุ Use Cases หลัก (8-12 use cases)
38. เชื่อมความสัมพันธ์ระหว่าง Actors กับ Use Cases
39. เพิ่ม Include/Extend relationships ถ้าจำเป็น
40. Export เป็นไฟล์ PNG หรือ PDF

ผลลัพธ์ที่คาดหวัง:

- Use Case Diagram ที่สมบูรณ์และถูกต้องตามมาตรฐาน UML
- แสดง Actors และ Use Cases ครอบคลุมฟังก์ชันหลักของระบบ

Activity 3: เขียน Use Case Scenario (30 นาที)

วัตถุประสงค์:

เลือก Use Case สำคัญ 2-3 Use Cases มาเขียน Scenario แบบละเอียด

ขั้นตอน:

41. เลือก Use Case ที่มีความซับซ้อนปานกลาง
42. เขียน Main Flow โดยละเอียด (5-10 steps)
43. เขียน Alternative Flows สำหรับกรณีผิดปกติ (2-3 flows)
44. ระบุ Preconditions และ Postconditions

ผลลัพธ์ที่คาดหวัง:

- Use Case Scenarios ที่สมบูรณ์สำหรับ 2-3 Use Cases หลัก
- เขียนในรูปแบบที่ชัดเจน เข้าใจง่าย

Activity 4: พัฒนา Landing Page ด้วย CSS (60 นาที)

วัตถุประสงค์:

ปรับปรุง Landing Page ที่สร้างในสัปดาห์ที่แล้วด้วย CSS Layout ให้สวยงามและใช้งานง่าย

ขั้นตอน:

45. สร้างไฟล์ styles.css และเชื่อมกับ HTML
46. กำหนด Color Palette (3-4 สี) ที่เหมาะกับธีมของโปรเจกต์
47. จัด Typography (font-family, font-size, line-height)
48. ใช้ Box Model เพื่อควบคุม spacing (margin, padding)
49. สร้าง Navigation Bar ที่ทำงานได้
50. จัด Layout ของ sections ต่างๆ ให้สวยงาม
51. สร้างหน้า About หรือ Features เพิ่มเติม
52. Commit และ Push ขึ้น Git repository

ผลลัพธ์ที่คาดหวัง:

- Landing Page ที่มี CSS styling ที่สวยงามและสอดคล้องกัน
- มี Navigation Bar ที่ใช้งานได้
- มีหน้าเพิ่มเติม (About/Features) อย่างน้อย 1-2 หน้า
- โค้ดถูก commit และ push ขึ้น Git repository

การส่งงาน

ส่งผ่านระบบ LMS ภายใน 1 สัปดาห์ (ก่อนเรียนสัปดาห์ถัดไป)

53. ลิงก์ Trello/Jira Board (ตั้งเป็น Public หรือเชิญอาจารย์)
54. Use Case Diagram (PNG/PDF)
55. Use Case Scenarios (Word/PDF)
56. ลิงก์ Git Repository (ต้องมี commit ใหม่จากสัปดาห์นี้)

7. เตรียมความพร้อมสู่สัปดาห์ถัดไป

ในสัปดาห์ที่ 4 เราจะก้าวไปสู่หัวข้อที่น่าตื่นเต้น:

Software Design Principles & UX/UI + Responsive HTML/CSS

หัวข้อที่จะเรียน

- **SOLID Principles (Introduction) - หลักการออกแบบซอฟต์แวร์ที่ดี**
  - Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion
- **Coupling & Cohesion - วัดคุณภาพการออกแบบระบบ**
- **User-Centred Design (UCD) - การออกแบบที่มุ่งเน้นผู้ใช้**
  - สร้าง Personas และ User Journey Maps
- **Responsive Design - ออกแบบให้รองรับหลายขนาดหน้าจอ**
  - CSS Flexbox และ Grid Layout
  - Media Queries สำหรับ Mobile, Tablet, Desktop

สิ่งที่ควรเตรียม

- ทบทวน HTML/CSS ที่เรียนมาในสัปดาห์นี้
- ศึกษาเพิ่มเติมเกี่ยวกับ Flexbox และ Grid จาก MDN Web Docs
- สังเกต UI/UX ของเว็บไซต์ที่ใช้บ่อย ว่ามีจุดเด่นอย่างไร
- เตรียม Term Project ให้พร้อม เพื่อปรับเป็น Responsive ในสัปดาห์หน้า

แหล่งเรียนรู้เพิ่มเติม

- MDN Web Docs - https://developer.mozilla.org
- CSS Tricks - https://css-tricks.com
- Agile User Stories - Mountain Goat Software
- UML Use Case Diagrams - UML Diagrams.org

สิ้นสุดเอกสารประกอบการสอนสัปดาห์ที่ 3