

**วิทยาลัยเทคโนโลยีภาคตะวันออก (อี.เทค)**

**แผนการจัดการเรียนรู้ฐานสมรรถนะตามมาตรฐานอาชีพ**

**หลักสูตรประกาศนียบัตรวิชาชีพชั้นสูง พุทธศักราช 2567**

**สาขาวิชาเทคโนโลยีสารสนเทศ**

**กลุ่มอาชีพซอฟต์แวร์และการประยุกต์**

**ประเภทวิชาอุตสาหกรรมดิจิทัลและเทคโนโลยีสารสนเทศ**

**รหัสวิชา 31901-2004 วิชา การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End**

**จัดทำโดย**

**นายเอกรัตน์ อุไรโรจน์**

คำนำ

การจัดทำแผนการจัดการเรียนรู้หลักสูตรประกาศนียบัตรวิชาชีพ (ปวช.) และหลักสูตรประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.) พุทธศักราช 2567 กำหนดหลักเกณฑ์การใช้หลักสูตร ให้เชื่อมโยงกับมาตรฐานอาชีพ เป็นหัวใจสำคัญต่อการพัฒนากระบวนการจัดการเรียนการสอนเป็นอย่างมาก วิทยาลัยเทคโนโลยีภาคตะวันออก (อี เทค) เห็นความสำคัญกระบวนการใช้มาตรฐานอาชีพที่อ้างอิงสมรรถนะมาตรฐาน ตามความต้องการของสถานประกอบการ วิทยาลัยกำหนดนโยบายสนับสนุนให้ครู ผู้สอน จัดทำแผนการจัดการเรียนรู้ตามหลักเกณฑ์การใช้หลักสูตรให้สอดคล้องกับสมรรถนะและเชื่อมโยงกับมาตรฐานอาชีพามที่อ้างอิงการจัดทำแผนการจัดการเรียนรู้วิชา การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End หน่วยกิต 3 วิชานี้ประกอบด้วยผังการกำหนดหน่วยสมรรถนะจำนวน 5 หน่วย หน่วยสมรรถนะย่อยจำนวน 8 หน่วย เชื่อมโยงมาตรฐาน สถาบันคุณวุฒิวิชาชีพ อาชีพ นักพัฒนาระบบ ระดับ 4

การกำหนดคะแนนคุณภาพผลงานนักเรียน การประเมินผลการเรียนทฤษฎี และฝึกปฏิบัติงาน แผนการจัดการเรียนรู้หลักสูตร พุทธศักราช 2567 นี้ ผู้จัดทำแผนการจัดการเรียนรู้นี้จาก การค้นคว้ารวบรวม เอกสารประกอบการสอน จากหนังสือ เว็บไซด์ต่างๆ และจากประสบการณ์ ความเชี่ยวชาญที่ทำการสอนมาหลายปี เมื่อนักเรียนนักศึกษาเรียนจบวิชานี้แล้ว สามารถปฏิบัติงาน นักพัฒนาระบบ ได้ตามมาตรฐานอาชีพสถาบันคุณวุฒิวิชาชีพ ระดับ 4

ผู้จัดทำแผนการจัดการเรียนรู้หลักสูตรฐานสมรรถนะ หวังว่าเอกสารเล่มนี้จะเป็นประโยชน์ต่อผู้เรียน ที่จะค้นคว้าศึกษาทำให้มีความรู้ และคณะผู้จัดมีความมั่นใจผู้จัดทำแผนการจัดการเรียนรู้หลักสูตรฐานสมรรถนะนี้ จะเป็นแนวทางการจัดการเรียนการสอนตามหลักสูตรประกาศนียบัตรวิชาชีพ 2567 ซึ่งจะเป็นประโยชน์แก่ผู้สอน ผู้เรียน รวมถึงผู้ที่สนใจเป็นอย่างยิ่ง

ผู้สอน เอกรัตน์ อุไรโรจน์

เดือน ตุลาคม พ.ศ. 2567

**สารบัญ**

**หน้า**

คำนำ

สารบัญ

รายวิชา x

ตารางวิเคราะห์หน่วยการเรียนรู้ฐานสมรรถนะตามมาตรฐานอาชีพ x

ตารางรายการสอน x

ตารางวิเคราะห์สมรรถนะเพื่อกำหนดพฤติกรรมบ่งชี้ x

กรอบโครงสร้างชุดการสอน X

ชุดการสอนที่.......เรื่อง/งาน....................... x

**วิชา การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End**

หลักสูตรประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)

ประเภทวิชาอุตสาหกรรมดิจิทัลและเทคโนโลยีสารสนเทศ กลุ่มอาชีพ ซอฟต์แวร์และการประยุกต์

สาขาวิชา เทคโนโลยีสารสนเทศ รหัส 31901-2004 ชื่อวิชา การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End

หน่วยกิต 3 ทฤษฎี 2 ชั่วโมง/สัปดาห์ ปฏิบัติ 2 ชั่วโมง/สัปดาห์

อ้างอิงมาตรฐาน

มาตรฐานอาชีพ สถาบันคุณวุฒิวิชาชีพ อาชีพ นักพัฒนาระบบ ระดับ 4

ผลลัพธ์การเรียนรู้ระดับรายวิชา

พัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End เพื ่อออกแบบส่วนติดต่อผู้ใช้ตามหลักการ ด้วยความละเอียด รอบคอบ รับผิดชอบ การสื่อสารและการทำงานเป็นทีม

จุดประสงค์รายวิชา เพื่อให้

1. เข้าใจการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End
2. มีทักษะในการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End
3. มีเจตคติและกิจนิสัยที่ดีในการปฏิบัติงานด้วยความละเอียดรอบคอบ รับผิดชอบ การสื่อสาร

การคิดเชิงนวัตกรรมและการทำงานเป็นทีม

1. มีความสามารถประยุกต์ใช้หลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ในงานอาชีพ

สมรรถนะรายวิชา

1. ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ
2. พัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการและกระบวนการ
3. ทดสอบและแก้ไขข้อผิดพลาดของซอฟต์แวร์ตามหลักการและกระบวนการ
4. จัดทำคู่มือการใช้งานตามหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End
5. ประยุกต์ใช้หลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ในงานอาชีพ

คำอธิบายรายวิชา

ศึกษาและปฏิบัติเกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ความหมาย ความสำคัญ ประโยชน์ของการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End เลือกใช้ภาษาและเทคโนโลยีสมัยใหม่ในการพัฒนา (Vue/React/Angular/ฯลฯ) การทำงานร่วนกันเป็นทีม (Version control System) ดำเนินการพัฒนาโปรแกรมแบบ Integration ตกแต่งเอกสารด้วย UI Component Library/CSS Framework เขียนโปรแกรมเชื่อมต่อ RESTful API/Web Services เขียนโปรแกรมติดต่อฐานข้อมูลแบบ SQL/NoSQL การทำ Authentication ในรูปแบบ JSON Web Token/Session หาจุดผิดพลาด ตามบันทึกข้อผิดพลาดแก้ไขข้อผิดพลาดของโปรแกรม ทดสอบการแก้ไขข้อผิดพลาดของโปรแกรมอ่าน Functional/Program Specification/UML เขียนโปรแกรมตาม Functional/Program Specification/ UML ออกแบบการทดสอบ Integration Test ดำเนินการทดสอบโปรแกรมแบบ Integration Test จัดทำรายงาน ศึกษาการใช้งานโปรแกรมที่พัฒนาขึ้น จัดทำคู่มือการใช้งานโปรแกรม ตรวจสอบความถูกต้องของคู่มือการใช้งานโปรแกรมการส่งมอบซอฟต์แวร์เพื่อให้สามารถใช้งานได้

**ตารางวิเคราะห์หน่วยการเรียนรู้ฐานสมรรถนะตามมาตรฐานอาชีพ**

แผนการจัดการเรียนรู้ฐานสมรรถนะตามมาตรฐานอาชีพ ประเภทวิชาอุตสาหกรรมดิจิทัลและเทคโนโลยีสารสนเทศ กลุ่มอาชีพซอฟต์แวร์และการประยุกต์ สาขาวิชาเทคโนโลยีสารสนเทศ ปีการศึกษา2567

**การวิเคราะห์เนื้อหารายวิชา**

รหัส 31901-2004 ชื่อวิชาการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End หน่วยกิต 3 นก.

ตารางที่ 2. การวิเคราะห์สมรรถนะและสมรรถนะย่อย อ้างอิงมาตรฐานอาชีพ สถาบันคุณวุฒิวิชาชีพ อาชีพ นักพัฒนาระบบ ระดับ 4

|  |  |  |  |
| --- | --- | --- | --- |
| **ที่** | **สมรรถนะ**  **(UOC)** | **สมรรถนะย่อย**  **(EOC)** | **เกณฑ์การประเมิน**  **(PC)** |
| 1 | ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ | เข้าใจหลักการ MVC ใน ASP.NET Core | 1. ความรู้ |
|  |  |  | 2. ทักษะ |
|  |  |  | 3. เจตคติ |
|  |  |  | 4. ประยุกต์ใช้ |
|  | พัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการและกระบวนการ | ออกแบบและสร้างฐานข้อมูล SQL Server | 1. ความรู้ |
|  |  |  | 2. ทักษะ |
|  |  |  | 3. เจตคติ |
|  |  |  | 4. ประยุกต์ใช้ |

|  |  |  |  |
| --- | --- | --- | --- |
| **ที่** | **สมรรถนะ**  **(UOC)** | **สมรรถนะย่อย**  **(EOC)** | **เกณฑ์การประเมิน**  **(PC)** |
| 1 |  | เข้าใจหลักการ MVC ใน ASP.NET Core | 1. ความรู้ |
|  |  |  | 2. ทักษะ |
|  |  |  | 3. เจตคติ |
|  |  |  | 4. ประยุกต์ใช้ |
|  |  | พัฒนาส่วนแสดงผล | 1. ความรู้ |
|  |  |  | 2. ทักษะ |
|  |  |  | 3. เจตคติ |
|  |  |  | 4. ประยุกต์ใช้ |
|  | ทดสอบและแก้ไขข้อผิดพลาดของซอฟต์แวร์ตามหลักการและกระบวนการ | ทดสอบการค้นหาและแสดงผล | 1. ความรู้ |
|  |  |  | 2. ทักษะ |
|  |  |  | 3. เจตคติ |
|  |  |  | 4. ประยุกต์ใช้ |
|  | 11309 จัดทำคู่มือการใช้งานตามหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End | 11309.01 ศึกษาการใช้งานโปรแกรมที่พัฒนาขึ้น | 1. ความรู้ |
|  |  |  | 2. ทักษะ |
|  |  |  | 3. เจตคติ |
|  |  |  | 4. ประยุกต์ใช้ |

|  |  |  |  |
| --- | --- | --- | --- |
| **ที่** | **สมรรถนะ**  **(UOC)** | **สมรรถนะย่อย**  **(EOC)** | **เกณฑ์การประเมิน**  **(PC)** |
| 1 |  | 11309.03 ตรวจสอบความถูกต้องของคู่มือการใช้งานโปรแกรม | 1. ความรู้ |
|  |  |  | 2. ทักษะ |
|  |  |  | 3. เจตคติ |
|  |  |  | 4. ประยุกต์ใช้ |
|  | ประยุกต์ใช้หลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ในงานอาชีพ | ออกแบบและพัฒนาโครงงานในรายวิชา | 1. ความรู้ |
|  |  |  | 2. ทักษะ |
|  |  |  | 3. เจตคติ |
|  |  |  | 4. ประยุกต์ใช้ |
|  |  | นำเสนอโครงงานในรายวิชา | 1. ความรู้ |
|  |  |  | 2. ทักษะ |
|  |  |  | 3. เจตคติ |
|  |  |  | 4. ประยุกต์ใช้ |

**1. ขอบเขตการปฎิบัติ (Range : R)**

* คอมพิวเตอร์
* Visual Studio 2022 IDE
* SQL Server 2022
* Microsoft Word

**2. หลักฐานที่ต้องการ (Evidence : EV)**

แนวคิด แบบประเมินการปฎิบัติงานจริง หรือ ข้อสอบ หรืออื่นๆ

**3. แนวทางการประเมิน (Assessment guidance : AG)**

แนวคิด การทดสอบหรือตั้งสถานีทดสอบมาตรฐาน

**ตารางรายการสอน**

|  |  |  |  |
| --- | --- | --- | --- |
| **ที่** | **รายการ** | **จำนวนชั่วโมง** | **สัปดาห์ที่** |
| **1.**  **2.**  **3**  **4**  **5** | **สมรรถนะที่1** ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ  สมรรถนะย่อยที่1 เข้าใจหลักการ MVC ใน ASP.NET Core  **สมรรถนะที่2** พัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการและกระบวนการ  สมรรถนะย่อยที่1 สร้างโครงสร้างโปรเจค ASP.NET Core MVC  สมรรถนะย่อยที่2 ออกแบบและสร้างฐานข้อมูล SQL Server  สมรรถนะย่อยที่3 พัฒนาส่วนแสดงผล  **สมรรถนะที่3**. ทดสอบและแก้ไขข้อผิดพลาดของซอฟต์แวร์ตามหลักการและกระบวนการ  สมรรถนะย่อยที่1 ทดสอบการค้นหาและแสดงผล  **สมรรถนะที่4** จัดทำคู่มือการใช้งานตามหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End  สมรรถนะย่อยที่1 ศึกษาการใช้งานโปรแกรมที่พัฒนาขึ้น  สมรรถนะย่อยที่2 ตรวจสอบความถูกต้องของคู่มือการใช้งานโปรแกรม  **สมรรถนะที่5** ประยุกต์ใช้หลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ในงานอาชีพ  สมรรถนะย่อยที่1 ออกแบบและพัฒนาโครงงานในรายวิชา  สมรรถนะย่อยที่2 นำเสนอโครงงานในรายวิชา | 4  4  4  20  4  8  4  8  4 | 1  2  3  4-8  9  10-11  12  13-14  15 |

|  |  |  |
| --- | --- | --- |
|  | **ชุดการสอน**  **หน่วยสมรรถนะ : UOC 1**ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ | **หน่วยที่1** |
| **รหัสวิชาชื่อวิชา**31901-2004  **หน่วยกิต 3** | **สัปดาห์ที่...** |
| **สมรรถนะย่อย : EOC** เข้าใจหลักการ MVC ใน ASP.NET Core | **ทฤษฎี 4 ชม.**  **ปฏิบัติ 0 ชม.** |

**1. ผลลัพธ์การเรียนรู้** .......แนวคิดจบวิชานี้เป้าหมายสามารถทำอะไรได้...................................................................

**2. อ้างอิงมาตรฐานอาชีพ** .........................................................................................................................................

**3. หน่วยสมรรถนะ : UOC 1** ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์

**หน่วยสมรรถนะย่อย : EOC1.1** **EOC** เข้าใจหลักการ MVC ใน ASP.NET Core

**4. เกณฑ์การประเมิน (Performance Criteria : PC)**

* 1. ความรู้: อธิบายหลักการและโครงสร้างของ MVC ใน ASP.NET Core ได้อย่างถูกต้อง
  2. ทักษะ: สามารถสร้างและจัดการโครงสร้าง MVC ในโปรเจค ASP.NET Core ได้
  3. เจตคติ: ตระหนักถึงความสำคัญของการใช้ MVC ในการพัฒนาเว็บแอปพลิเคชัน
  4. ประยุกต์การใช้งาน: สามารถประยุกต์ใช้ MVC ในการพัฒนาเว็บแอปพลิเคชันขนาดเล็กได้

**5. ขอบเขตการปฏิบัติ (Range: R)**

* 1. เครื่องคอมพิวเตอร์ที่มีการติดตั้ง Visual Studio และ .NET Core SDK
  2. เอกสารประกอบการสอนเกี่ยวกับ MVC ใน ASP.NET Core
  3. ระบบอินเทอร์เน็ตสำหรับการดาวน์โหลดแพ็คเกจและการค้นคว้าเพิ่มเติม

**6. จุดประสงค์เชิงพฤติกรรม**

* 1. ความรู้:
* อธิบายความหมายและหลักการของ MVC ได้
* ระบุหน้าที่ของแต่ละส่วนใน MVC (Model, View, Controller) ได้
* อธิบายวิธีการทำงานของ Routing ใน ASP.NET Core MVC ได้
  1. ทักษะ:
* สร้างโปรเจค ASP.NET Core MVC ใหม่ได้
* สร้างและจัดการ Controller, Model, และ View ได้อย่างถูกต้อง
* ใช้ Data Annotation ในการกำหนดกฎการตรวจสอบข้อมูลได้
  1. เจตคติ:
* เห็นประโยชน์ของการแยกส่วนการทำงานตามหลักการ MVC
* มีความกระตือรือร้นในการเรียนรู้และทดลองใช้ MVC ในการพัฒนาเว็บแอปพลิเคชัน

6.4 ประยุกต์การใช้งาน:

* สามารถพัฒนาเว็บแอปพลิเคชันอย่างง่ายโดยใช้ ASP.NET Core MVC ได้
* สามารถเชื่อมต่อฐานข้อมูลและทำ CRUD operations ผ่าน MVC pattern ได้

**7. เนื้อหา**

1. Model-View-Controller คืออะไร
   1. Model-View-Controller (MVC) เป็นรูปแบบการออกแบบสถาปัตยกรรมซอฟต์แวร์ที่ใช้กันอย่างแพร่หลาย

ในการพัฒนาแอปพลิเคชันเว็บ ASP.NET Core ได้นำรูปแบบนี้มาใช้เพื่อช่วยให้นักพัฒนาสามารถสร้างแอปพลิเคชันที่มีโครงสร้างดี ง่ายต่อการบำรุงรักษา และขยายได้

* 1. องค์ประกอบของ MVC
     1. Model เป็นส่วนที่จัดการข้อมูลและตรรกะทางธุรกิจของแอปพลิเคชันประกอบด้วยคลาสที่แทนข้อมูลและกฎทางธุรกิจไม่ขึ้นกับส่วน View หรือ Controller
     2. View รับผิดชอบในการแสดงผลข้อมูลให้กับผู้ใช้ใน ASP.NET Core MVC, Views มักจะเป็นไฟล์ .cshtml ที่ใช้ Razor syntax
     3. Controller ทำหน้าที่เป็นตัวกลางระหว่าง Model และ View จัดการการร้องขอของผู้ใช้ ประมวลผลข้อมูลจาก Model และส่งข้อมูลไปยัง View ในASP.NET Core, Controllers เป็นคลาสที่สืบทอดมาจาก ControllerBase หรือ Controller
  2. การทำงานของ MVC ใน ASP.NET Core

ผู้ใช้ส่งคำขอผ่านเบราว์เซอร์Routing ใน ASP.NET Core จะส่งคำขอไปยัง Controller ที่เหมาะสมController ประมวลผลคำขอ โดยอาจมีการเรียกใช้ Model เพื่อดึงหรือจัดการข้อมูลController เลือก View ที่เหมาะสมและส่งข้อมูลไปยัง View View แสดงผลข้อมูลและส่งกลับเป็น HTML ไปยังเบราว์เซอร์ของผู้ใช้

* 1. ข้อดีของการใช้ MVC
* แยกส่วนความรับผิดชอบ (Separation of Concerns)
* ง่ายต่อการทดสอบ (Testability)
* ความยืดหยุ่นในการพัฒนา

1. โครงสร้างพื้นฐานและการทำงาน

SQL Server เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS) ที่มีองค์ประกอบหลายอย่างเพื่อจัดการและควบคุมข้อมูลที่เก็บในฐานข้อมูลได้อย่างมีประสิทธิภาพ โครงสร้างหลักของ SQL Server ประกอบด้วยองค์ประกอบดังนี้

* 1. โครงสร้างของ SQL Server:
     1. Tables: โครงสร้างหลักในการเก็บข้อมูล ตารางเป็นโครงสร้างพื้นฐานในการจัดเก็บข้อมูลใน SQL Server ข้อมูลจะถูกจัดเก็บในรูปแบบของแถว (rows) และคอลัมน์ (columns) โดยแต่ละคอลัมน์จะกำหนดประเภทข้อมูล เช่น int, varchar, datetime เป็นต้น ตารางหนึ่งๆ ในฐานข้อมูลสามารถกำหนดความสัมพันธ์กับตารางอื่นๆ ได้โดยใช้คีย์ต่างๆ เช่น Primary Key และ Foreign Key
     2. Stored Procedures:(โปรแกรมจัดเก็บสำเร็จรูป)Stored Procedures เป็นชุดคำสั่ง SQL ที่ถูกจัดเก็บไว้ใน SQL Server ซึ่งสามารถนำมาใช้ซ้ำได้ โดยมีประโยชน์ในการลดความซับซ้อนของการประมวลผลข้อมูล และเพิ่มประสิทธิภาพในการจัดการกับข้อมูล Stored Procedures สามารถรองรับการใช้เงื่อนไข (conditions), การวนซ้ำ (loops), และตัวแปร (variables) ภายในคำสั่งได้ อีกทั้งยังช่วยปรับปรุงความปลอดภัยของระบบเพราะการเข้าถึงข้อมูลสามารถควบคุมได้โดยไม่ต้องเปิดเผยโครงสร้างข้อมูลที่แท้จริง
     3. Indexes: คือโครงสร้างข้อมูลที่ใช้เพื่อเพิ่มความเร็วในการค้นหาและดึงข้อมูลจากตารางในฐานข้อมูล โดยดัชนีจะถูกสร้างขึ้นจากหนึ่งหรือหลายคอลัมน์ของตาราง การใช้งานดัชนีที่เหมาะสมจะช่วยให้การค้นหาข้อมูลทำได้รวดเร็วและมีประสิทธิภาพมากขึ้น อย่างไรก็ตาม การสร้างดัชนีที่มากเกินไปอาจส่งผลกระทบต่อประสิทธิภาพในการเพิ่ม แก้ไข หรือลบข้อมูล เนื่องจากต้องมีการอัปเดตดัชนีอยู่ตลอดเวลา
  2. สถาปัตยกรรมของ ASP.NET Core: เป็นแพลตฟอร์มสำหรับการพัฒนาเว็บแอปพลิเคชันที่ถูกออกแบบมาให้ทำงานบนหลายแพลตฟอร์ม (cross-platform) และมีโครงสร้างที่ยืดหยุ่นและทันสมัย สถาปัตยกรรมหลักของ ASP.NET Core มีองค์ประกอบสำคัญที่ช่วยในการพัฒนาแอปพลิเคชันที่มีความสามารถในการปรับตัวได้ดี, ง่ายต่อการทดสอบ, และมีประสิทธิภาพสูง ดังนี้
     1. MVC (Model-View-Controller) เป็นสถาปัตยกรรมแบบแยกส่วนที่ช่วยจัดการโครงสร้างของแอปพลิเคชัน โดยแบ่งออกเป็นสามส่วนหลัก:
* Model: ส่วนของแอปพลิเคชันที่จัดการข้อมูลและธุรกิจลอจิก (business logic) โดย Model จะเป็นตัวแทนของข้อมูลที่ได้รับจากฐานข้อมูลและใช้ในการประมวลผลต่างๆ เช่น การคำนวณหรือการตรวจสอบ
* View: ส่วนที่แสดงผลข้อมูลให้กับผู้ใช้งาน เป็นส่วนที่รับผิดชอบในการจัดการการแสดงข้อมูลที่ได้รับจาก Model ไปสู่ผู้ใช้งาน ผ่านการสร้างหน้าเว็บ (HTML) หรืออินเทอร์เฟซอื่นๆ
* Controller: ตัวกลางระหว่าง View และ Model มีหน้าที่รับคำขอ (request) จากผู้ใช้ผ่าน View และทำการประมวลผลโดยการเรียกใช้ Model เพื่อดึงข้อมูล จากนั้นส่งข้อมูลที่ได้ไปยัง View เพื่อแสดงผล
  + 1. Middleware คือส่วนประกอบที่ถูกใช้เพื่อจัดการคำขอ (request) และคำตอบ (response) ในแอปพลิเคชัน ASP.NET Core เมื่อผู้ใช้ส่งคำขอเข้ามาในแอปพลิเคชัน คำขอนั้นจะผ่าน Middleware หลายตัว ซึ่งจะทำงานเป็นขั้นตอนก่อนที่คำขอนั้นจะไปถึงส่วนที่ประมวลผลจริงๆ เช่น Controller Middleware แต่ละตัวสามารถ:
* จัดการคำขอ (เช่น การตรวจสอบสิทธิ์)
* เปลี่ยนแปลงคำขอหรือคำตอบ
* สร้างคำตอบเองหรือส่งคำขอไปยัง Middleware ตัวถัดไป
  + 1. Dependency Injection (DI) เป็นแนวคิดที่ช่วยในการจัดการ dependencies ของแอปพลิเคชัน โดยที่ ASP.NET Core นำ DI มาใช้เป็นส่วนหนึ่งของสถาปัตยกรรมหลัก ซึ่งช่วยในการควบคุมการสร้างและการใช้งานของ objects ที่เป็น dependencies ในแอปพลิเคชัน เช่น บริการ (services) หรือ classes อื่นๆ ที่แอปพลิเคชันต้องการใช้งาน การใช้ DI ช่วยให้โค้ดมีความยืดหยุ่นและทดสอบง่ายขึ้น โดยการแยกการสร้างและการใช้งาน objects ออกจากกัน ตัวอย่างของการใช้ Dependency Injection ได้แก่:
* Service Registration: แอปพลิเคชันจะทำการลงทะเบียนบริการต่างๆ ใน Startup.cs เช่นการสร้าง service ในรูปแบบของ Singleton, Scoped, หรือ Transient
* Service Injection: ASP.NET Core จะทำการ inject service ที่ลงทะเบียนไว้เข้าไปยัง Controller, Middleware, หรือส่วนประกอบอื่นๆ ของแอปพลิเคชันที่ต้องการใช้งาน

1. การเชื่อมต่อและการดึงข้อมูล

ในการพัฒนาแอปพลิเคชันด้วย ASP.NET Core การเชื่อมต่อกับฐานข้อมูลเป็นขั้นตอนสำคัญที่ช่วยให้สามารถจัดการข้อมูลได้อย่างมีประสิทธิภาพ โดยการเชื่อมต่อและดึงข้อมูลใน ASP.NET Core สามารถทำได้ผ่านหลายวิธี หนึ่งในวิธีหลักคือการใช้ Connection String และ Entity Framework Core ซึ่งเป็นเครื่องมือที่รองรับการทำงานกับฐานข้อมูลเชิงสัมพันธ์ผ่านการแมปข้อมูลในรูปแบบของวัตถุ (ORM)

* 1. Connection String คือข้อมูลที่ใช้ระบุรายละเอียดการเชื่อมต่อกับฐานข้อมูล เช่น ชื่อเซิร์ฟเวอร์, ชื่อฐานข้อมูล, และข้อมูลรับรอง (credentials) ของผู้ใช้ โดยทั่วไปแล้ว Connection String จะถูกเก็บไว้ในไฟล์การตั้งค่าของแอปพลิเคชัน เช่น appsettings.json
  2. Entity Framework Core เป็นเฟรมเวิร์กแบบ ORM ที่ได้รับความนิยมใน .NET สำหรับการทำงานกับฐานข้อมูลเชิงสัมพันธ์ เช่น SQL Server, PostgreSQL, และ MySQL EF Core ช่วยให้ผู้พัฒนาสามารถทำงานกับข้อมูลในฐานข้อมูลได้ในลักษณะของวัตถุ (Object-Oriented) โดยไม่ต้องเขียนคำสั่ง SQL โดยตรง EF Core ทำหน้าที่เป็นตัวกลางที่แปลงคำสั่งในโค้ดที่เป็นวัตถุ (C#) ให้กลายเป็นคำสั่ง SQL ที่ทำงานกับฐานข้อมูลได้ โดยผู้พัฒนาสามารถทำงานกับข้อมูลในลักษณะของ "Entities" ซึ่งแต่ละ Entity จะเป็นตัวแทนของตารางในฐานข้อมูล
  3. ORM (Object-Relational Mapping) คือเทคนิคในการแมป (mapping) ข้อมูลระหว่างฐานข้อมูลเชิงสัมพันธ์ (Relational Database) และวัตถุในภาษาการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming, OOP) การใช้ ORM ช่วยให้ผู้พัฒนาไม่จำเป็นต้องเขียนคำสั่ง SQL โดยตรง และสามารถจัดการข้อมูลในฐานข้อมูลในลักษณะของวัตถุได้อย่างสะดวกใน .NET หนึ่งใน ORM ที่ได้รับความนิยมมากที่สุดคือ Entity Framework Core ซึ่งทำให้การเชื่อมต่อฐานข้อมูลและการจัดการข้อมูลสามารถทำได้อย่างยืดหยุ่นและสะดวก ผู้พัฒนาสามารถสร้างโครงสร้างฐานข้อมูลจากโค้ด (Code-First) หรือสร้างโครงสร้างของวัตถุจากฐานข้อมูลที่มีอยู่แล้ว (Database-First)

**8. คู่มือ / Data**

1. คู่มือการใช้งาน Visual Studio สำหรับการพัฒนา ASP.NET Core MVC
2. เอกสารอธิบายโครงสร้างและการทำงานของ MVC ใน ASP.NET Core
3. ตัวอย่างโค้ดสำหรับการสร้าง Model, View, และ Controller พื้นฐาน

**9. เอกสารอ้างอิง**

1. Microsoft. (2024). Introduction to ASP.NET Core MVC. https://docs.microsoft.com/en-us/aspnet/core/mvc/overview
2. Freeman, A. (2023). Pro ASP.NET Core MVC. Apress.
3. Smith, S. (2024). ASP.NET Core in Action. Manning Publications.
4. Sanderson, S. (2023). ASP.NET Core Application Development: Building an Application in Four Sprints. Microsoft Press.
5. ASP.NET Core MVC Tutorial. (2024). https://www.tutorialsteacher.com/core/aspnet-core-introduction10. ใบงาน/ใบมอบหมายงาน

…………………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………………………

แผนผังกำหนดหน่วยโมดูลฐานสมรรถนะ

ตารางรายการสอน

|  |  |  |
| --- | --- | --- |
| รายการ | จำนวนชั่วโมง | สัปดาห์ที่ |
| **สมรรถนะที่ 1** ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ | | |
| สรรถนะย่อยที่ 1.1 เข้าใจหลักการ MVC ใน ASP.NET Core | 4 | 1 |
| **สมรรถนะที่ 2** พัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการและกระบวนการ | | |
| 2.1 สร้างโครงสร้างโปรเจค ASP.NET Core MVC | 4 | 2 |
| 2.2 ออกแบบและสร้างฐานข้อมูล SQL Server | 4 | 3 |
| 2.3 พัฒนาส่วนแสดงผล | 20 | 4-8 |
| **สมรรถนะที่ 3**. ทดสอบและแก้ไขข้อผิดพลาดของซอฟต์แวร์ตามหลักการและกระบวนการ | | |
| 3.1 ทดสอบการค้นหาและแสดงผล | 4 | 9 |
| **สมรรถนะที่ 4.**จัดทำคู่มือการใช้งานตามหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End | | |
| 4.1 ศึกษาการใช้งานโปรแกรมที่พัฒนาขึ้น | 8 | 10-11 |
| 4.2 ตรวจสอบความถูกต้องของคู่มือการใช้งานโปรแกรม | 4 | 12 |
| **สมรรถนะที่ 5.**ประยุกต์ใช้หลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ในงานอาชีพ | | |
| 5.1 ออกแบบและพัฒนาโครงงานในรายวิชา | 8 | 13-14 |
| 5.2 นำเสนอโครงงานในรายวิชา | 4 | 15 |
| รวม | 60 | 15 |

หน่วยสมรรถนะที่ 1 ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ที่** | **สมรรถนะย่อย  (EOC)** | **ขอบเขตการปฏิบัติงาน  (R)** | **เกณฑ์การประเมิน (PC)** | **เครื่องมืออุปกรณ์** | **หลักฐาน (EV)** | |
| **1.3** | **ศึกษาการใช้งานโปรแกรมที่พัฒนาขึ้น** | 1. ความถูกต้องของการทำงาน  2. การออกแบบโครงสร้าง MVC  3. การออกแบบ UI/UX (10%) | 1. ระบุการทำงานของโปรแกรม  2. ระบุการใช้งานโปรแกรม  3. ระบุข้อจำกัดการใช้งานโปรแกรม | - เครื่องคอมพิวเตอร์  - visual studio code | - แบบบรรจุภัณฑ์ / ตัวละครเกม | ประเมินผลการ  ทดสอบภาคปฏิบัติ  (ใบงาน) ผ่าน  เกณฑ์ร้อยละ 80 |
| **ขั้นตอนการประเมิน (AG)** | |
| จัดห้องทดสอบภาคปฏิบัติโดยการ  เตรียมใบงานหรือสถานการณ์จำลอง | |

**ชุดการสอนตามหน่วยสมรรถนะ**

**รหัสวิชา** 31901-2004 **ชื่อวิชา** การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End **หน่วยกิต** 4 รหัสหน่วย

**หน่วยสมรรถนะ** ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ

**หน่วยสมรรถนะย่อย** เข้าใจหลักการ MVC ใน ASP.NET Core

**เกณฑ์การประเมิน**

1. แบบทดสอบความเข้าใจเกี่ยวกับหลักการ MVC
2. โปรเจคจำลองโดยใช้ ASP.NET Core MVC
3. การนำเสนอและอธิบายโครงสร้าง MVC ที่สร้างขึ้น

**บทนำ**

Model-View-Controller คืออะไร

1. Model-View-Controller (MVC) เป็นรูปแบบการออกแบบสถาปัตยกรรมซอฟต์แวร์ที่ใช้กันอย่างแพร่หลาย

ในการพัฒนาแอปพลิเคชันเว็บ ASP.NET Core ได้นำรูปแบบนี้มาใช้เพื่อช่วยให้นักพัฒนาสามารถสร้างแอปพลิเคชันที่มีโครงสร้างดี ง่ายต่อการบำรุงรักษา และขยายได้

1. องค์ประกอบของ MVC
   1. Model เป็นส่วนที่จัดการข้อมูลและตรรกะทางธุรกิจของแอปพลิเคชันประกอบด้วยคลาสที่แทนข้อมูลและกฎทางธุรกิจไม่ขึ้นกับส่วน View หรือ Controller
   2. View รับผิดชอบในการแสดงผลข้อมูลให้กับผู้ใช้ใน ASP.NET Core MVC, Views มักจะเป็นไฟล์ .cshtml ที่ใช้ Razor syntax
   3. Controller ทำหน้าที่เป็นตัวกลางระหว่าง Model และ View จัดการการร้องขอของผู้ใช้ ประมวลผลข้อมูลจาก Model และส่งข้อมูลไปยัง View ในASP.NET Core, Controllers เป็นคลาสที่สืบทอดมาจาก ControllerBase หรือ Controller
2. การทำงานของ MVC ใน ASP.NET Core

ผู้ใช้ส่งคำขอผ่านเบราว์เซอร์Routing ใน ASP.NET Core จะส่งคำขอไปยัง Controller ที่เหมาะสมController ประมวลผลคำขอ โดยอาจมีการเรียกใช้ Model เพื่อดึงหรือจัดการข้อมูลController เลือก View ที่เหมาะสมและส่งข้อมูลไปยัง View View แสดงผลข้อมูลและส่งกลับเป็น HTML ไปยังเบราว์เซอร์ของผู้ใช้

1. ข้อดีของการใช้ MVC

* แยกส่วนความรับผิดชอบ (Separation of Concerns)
* ง่ายต่อการทดสอบ (Testability)
* ความยืดหยุ่นในการพัฒนา

1. โครงสร้างพื้นฐานและการทำงาน

SQL Server เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS) ที่มีองค์ประกอบหลายอย่างเพื่อจัดการและควบคุมข้อมูลที่เก็บในฐานข้อมูลได้อย่างมีประสิทธิภาพ โครงสร้างหลักของ SQL Server ประกอบด้วยองค์ประกอบดังนี้

* 1. โครงสร้างของ SQL Server:
     1. Tables: โครงสร้างหลักในการเก็บข้อมูล ตารางเป็นโครงสร้างพื้นฐานในการจัดเก็บข้อมูลใน SQL Server ข้อมูลจะถูกจัดเก็บในรูปแบบของแถว (rows) และคอลัมน์ (columns) โดยแต่ละคอลัมน์จะกำหนดประเภทข้อมูล เช่น int, varchar, datetime เป็นต้น ตารางหนึ่งๆ ในฐานข้อมูลสามารถกำหนดความสัมพันธ์กับตารางอื่นๆ ได้โดยใช้คีย์ต่างๆ เช่น Primary Key และ Foreign Key
     2. Stored Procedures:(โปรแกรมจัดเก็บสำเร็จรูป)Stored Procedures เป็นชุดคำสั่ง SQL ที่ถูกจัดเก็บไว้ใน SQL Server ซึ่งสามารถนำมาใช้ซ้ำได้ โดยมีประโยชน์ในการลดความซับซ้อนของการประมวลผลข้อมูล และเพิ่มประสิทธิภาพในการจัดการกับข้อมูล Stored Procedures สามารถรองรับการใช้เงื่อนไข (conditions), การวนซ้ำ (loops), และตัวแปร (variables) ภายในคำสั่งได้ อีกทั้งยังช่วยปรับปรุงความปลอดภัยของระบบเพราะการเข้าถึงข้อมูลสามารถควบคุมได้โดยไม่ต้องเปิดเผยโครงสร้างข้อมูลที่แท้จริง
     3. Indexes: คือโครงสร้างข้อมูลที่ใช้เพื่อเพิ่มความเร็วในการค้นหาและดึงข้อมูลจากตารางในฐานข้อมูล โดยดัชนีจะถูกสร้างขึ้นจากหนึ่งหรือหลายคอลัมน์ของตาราง การใช้งานดัชนีที่เหมาะสมจะช่วยให้การค้นหาข้อมูลทำได้รวดเร็วและมีประสิทธิภาพมากขึ้น อย่างไรก็ตาม การสร้างดัชนีที่มากเกินไปอาจส่งผลกระทบต่อประสิทธิภาพในการเพิ่ม แก้ไข หรือลบข้อมูล เนื่องจากต้องมีการอัปเดตดัชนีอยู่ตลอดเวลา
  2. สถาปัตยกรรมของ ASP.NET Core: เป็นแพลตฟอร์มสำหรับการพัฒนาเว็บแอปพลิเคชันที่ถูกออกแบบมาให้ทำงานบนหลายแพลตฟอร์ม (cross-platform) และมีโครงสร้างที่ยืดหยุ่นและทันสมัย สถาปัตยกรรมหลักของ ASP.NET Core มีองค์ประกอบสำคัญที่ช่วยในการพัฒนาแอปพลิเคชันที่มีความสามารถในการปรับตัวได้ดี, ง่ายต่อการทดสอบ, และมีประสิทธิภาพสูง ดังนี้
     1. MVC (Model-View-Controller) เป็นสถาปัตยกรรมแบบแยกส่วนที่ช่วยจัดการโครงสร้างของแอปพลิเคชัน โดยแบ่งออกเป็นสามส่วนหลัก:
* Model: ส่วนของแอปพลิเคชันที่จัดการข้อมูลและธุรกิจลอจิก (business logic) โดย Model จะเป็นตัวแทนของข้อมูลที่ได้รับจากฐานข้อมูลและใช้ในการประมวลผลต่างๆ เช่น การคำนวณหรือการตรวจสอบ
* View: ส่วนที่แสดงผลข้อมูลให้กับผู้ใช้งาน เป็นส่วนที่รับผิดชอบในการจัดการการแสดงข้อมูลที่ได้รับจาก Model ไปสู่ผู้ใช้งาน ผ่านการสร้างหน้าเว็บ (HTML) หรืออินเทอร์เฟซอื่นๆ
* Controller: ตัวกลางระหว่าง View และ Model มีหน้าที่รับคำขอ (request) จากผู้ใช้ผ่าน View และทำการประมวลผลโดยการเรียกใช้ Model เพื่อดึงข้อมูล จากนั้นส่งข้อมูลที่ได้ไปยัง View เพื่อแสดงผล
  + 1. Middleware คือส่วนประกอบที่ถูกใช้เพื่อจัดการคำขอ (request) และคำตอบ (response) ในแอปพลิเคชัน ASP.NET Core เมื่อผู้ใช้ส่งคำขอเข้ามาในแอปพลิเคชัน คำขอนั้นจะผ่าน Middleware หลายตัว ซึ่งจะทำงานเป็นขั้นตอนก่อนที่คำขอนั้นจะไปถึงส่วนที่ประมวลผลจริงๆ เช่น Controller Middleware แต่ละตัวสามารถ:
* จัดการคำขอ (เช่น การตรวจสอบสิทธิ์)
* เปลี่ยนแปลงคำขอหรือคำตอบ
* สร้างคำตอบเองหรือส่งคำขอไปยัง Middleware ตัวถัดไป
  + 1. Dependency Injection (DI) เป็นแนวคิดที่ช่วยในการจัดการ dependencies ของแอปพลิเคชัน โดยที่ ASP.NET Core นำ DI มาใช้เป็นส่วนหนึ่งของสถาปัตยกรรมหลัก ซึ่งช่วยในการควบคุมการสร้างและการใช้งานของ objects ที่เป็น dependencies ในแอปพลิเคชัน เช่น บริการ (services) หรือ classes อื่นๆ ที่แอปพลิเคชันต้องการใช้งาน การใช้ DI ช่วยให้โค้ดมีความยืดหยุ่นและทดสอบง่ายขึ้น โดยการแยกการสร้างและการใช้งาน objects ออกจากกัน ตัวอย่างของการใช้ Dependency Injection ได้แก่:
* Service Registration: แอปพลิเคชันจะทำการลงทะเบียนบริการต่างๆ ใน Startup.cs เช่นการสร้าง service ในรูปแบบของ Singleton, Scoped, หรือ Transient
* Service Injection: ASP.NET Core จะทำการ inject service ที่ลงทะเบียนไว้เข้าไปยัง Controller, Middleware, หรือส่วนประกอบอื่นๆ ของแอปพลิเคชันที่ต้องการใช้งาน

1. การเชื่อมต่อและการดึงข้อมูล

ในการพัฒนาแอปพลิเคชันด้วย ASP.NET Core การเชื่อมต่อกับฐานข้อมูลเป็นขั้นตอนสำคัญที่ช่วยให้สามารถจัดการข้อมูลได้อย่างมีประสิทธิภาพ โดยการเชื่อมต่อและดึงข้อมูลใน ASP.NET Core สามารถทำได้ผ่านหลายวิธี หนึ่งในวิธีหลักคือการใช้ Connection String และ Entity Framework Core ซึ่งเป็นเครื่องมือที่รองรับการทำงานกับฐานข้อมูลเชิงสัมพันธ์ผ่านการแมปข้อมูลในรูปแบบของวัตถุ (ORM)

* 1. Connection String คือข้อมูลที่ใช้ระบุรายละเอียดการเชื่อมต่อกับฐานข้อมูล เช่น ชื่อเซิร์ฟเวอร์, ชื่อฐานข้อมูล, และข้อมูลรับรอง (credentials) ของผู้ใช้ โดยทั่วไปแล้ว Connection String จะถูกเก็บไว้ในไฟล์การตั้งค่าของแอปพลิเคชัน เช่น appsettings.json
  2. Entity Framework Core เป็นเฟรมเวิร์กแบบ ORM ที่ได้รับความนิยมใน .NET สำหรับการทำงานกับฐานข้อมูลเชิงสัมพันธ์ เช่น SQL Server, PostgreSQL, และ MySQL EF Core ช่วยให้ผู้พัฒนาสามารถทำงานกับข้อมูลในฐานข้อมูลได้ในลักษณะของวัตถุ (Object-Oriented) โดยไม่ต้องเขียนคำสั่ง SQL โดยตรง EF Core ทำหน้าที่เป็นตัวกลางที่แปลงคำสั่งในโค้ดที่เป็นวัตถุ (C#) ให้กลายเป็นคำสั่ง SQL ที่ทำงานกับฐานข้อมูลได้ โดยผู้พัฒนาสามารถทำงานกับข้อมูลในลักษณะของ "Entities" ซึ่งแต่ละ Entity จะเป็นตัวแทนของตารางในฐานข้อมูล
  3. ORM (Object-Relational Mapping) คือเทคนิคในการแมป (mapping) ข้อมูลระหว่างฐานข้อมูลเชิงสัมพันธ์ (Relational Database) และวัตถุในภาษาการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming, OOP) การใช้ ORM ช่วยให้ผู้พัฒนาไม่จำเป็นต้องเขียนคำสั่ง SQL โดยตรง และสามารถจัดการข้อมูลในฐานข้อมูลในลักษณะของวัตถุได้อย่างสะดวกใน .NET หนึ่งใน ORM ที่ได้รับความนิยมมากที่สุดคือ Entity Framework Core ซึ่งทำให้การเชื่อมต่อฐานข้อมูลและการจัดการข้อมูลสามารถทำได้อย่างยืดหยุ่นและสะดวก ผู้พัฒนาสามารถสร้างโครงสร้างฐานข้อมูลจากโค้ด (Code-First) หรือสร้างโครงสร้างของวัตถุจากฐานข้อมูลที่มีอยู่แล้ว (Database-First)

**ชุดการสอนตามหน่วยสมรรถนะ**

**รหัสวิชา** 31901-2004 **ชื่อวิชา** การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End **หน่วยกิต** 4 รหัสหน่วย

**หน่วยสมรรถนะ** พัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการและกระบวนการ

**หน่วยสมรรถนะย่อย** สร้างโครงสร้างโปรเจค ASP.NET Core MVC

**เกณฑ์การประเมิน**

1. สร้างโครงสร้างโปรเจค ASP.NET Core MVC ได้ถูกต้องตามมาตรฐาน
2. กำหนด Model, View, และ Controller ได้อย่างเหมาะสมกับความต้องการของระบบ

**บทนำ**

**การสร้างโครงสร้างโปรเจค ASP.NET Core MVC**

การสร้างโปรเจค ASP.NET Core MVC เป็นขั้นตอนสำคัญสำหรับผู้ที่ต้องการพัฒนาแอปพลิเคชันเว็บที่ใช้เทคโนโลยี .NET Core ของ Microsoft โดยในขั้นตอนต่อไปนี้จะอธิบายวิธีการติดตั้ง .NET Core SDK และ Visual Studio, การสร้างโปรเจค ASP.NET Core MVC ใหม่, และการทำความเข้าใจโครงสร้างไฟล์และโฟลเดอร์ในโปรเจค

1. การติดตั้ง .NET Core SDK และ Visual Studio

1.1 การติดตั้ง .NET Core SDK

.NET Core SDK เป็นชุดเครื่องมือสำหรับการพัฒนาแอปพลิเคชันด้วย .NET โดยคุณสามารถดาวน์โหลดและติดตั้งได้จากเว็บไซต์ทางการของ Microsoft:

ไปที่ Download .NET

เลือกเวอร์ชันล่าสุดของ .NET SDK (ปัจจุบันคือ .NET 7 หรือ .NET 8)

ดาวน์โหลดและทำการติดตั้งตามขั้นตอน

1.2 การติดตั้ง Visual Studio

Visual Studio เป็นเครื่องมือพัฒนาโปรแกรมที่มีการรองรับการพัฒนาแอปพลิเคชัน ASP.NET Core:

ไปที่ Download Visual Studio

เลือก Visual Studio Community ซึ่งเป็นเวอร์ชันฟรี

ในขั้นตอนการติดตั้ง ให้เลือก workload ที่ชื่อว่า ASP.NET and web development

คลิก Install และรอจนกว่าการติดตั้งจะเสร็จสมบูรณ์

2. การสร้างโปรเจค ASP.NET Core MVC ใหม่

หลังจากติดตั้ง .NET Core SDK และ Visual Studio แล้ว คุณสามารถสร้างโปรเจค ASP.NET Core MVC ได้ตามขั้นตอนต่อไปนี้:

2.1 เปิด Visual Studio

เปิด Visual Studio

เลือก Create a new project

2.2 เลือกเทมเพลต ASP.NET Core Web App (Model-View-Controller)

ในหน้าต่าง Create a new project, ค้นหา ASP.NET Core Web App (Model-View-Controller) จากรายการเทมเพลตที่มีให้

เลือกแล้วคลิก Next

2.3 ตั้งชื่อโปรเจค

ตั้งชื่อโปรเจคของคุณ (เช่น MyMvcApp)

เลือกโฟลเดอร์ที่ต้องการบันทึกโปรเจค จากนั้นคลิก Next

2.4 เลือก .NET Core และเวอร์ชัน .NET

เลือก .NET 7.0 หรือเวอร์ชันที่คุณต้องการ

คลิก Create

3. โครงสร้างไฟล์และโฟลเดอร์ในโปรเจค

หลังจากที่คุณสร้างโปรเจคสำเร็จแล้ว โครงสร้างไฟล์และโฟลเดอร์ของ ASP.NET Core MVC จะมีลักษณะดังนี้:

* การติดตั้ง .NET Core SDK และ Visual Studio
* การสร้างโปรเจค ASP.NET Core MVC ใหม่
* โครงสร้างไฟล์และโฟลเดอร์ในโปรเจค