วิทยาลัยสงฆ์พุทธโสธร มหาวิทยาลัยมหาจุฬาลงกรณราชวิทยาลัย

คู่มือปฏิบัติงานตามมาตรฐานกำหนดตำแหน่ง (สายปฏิบัติการวิชาชีพ มจร.)

คู่มือการพัฒนาระบบสารสนเทศด้วยเครื่องมือ Visual Studio Code

เอกสารประกอบการประเมินเพื่อแต่งตั้งให้ดำรงตำแหน่งระดับชำนาญการ ตำแหน่งนักวิชาการคอมพิวเตอร์

๑ ข้อมูลเอกสารและสัดส่วนการมีส่วนร่วมในผลงาน (Document Metadata)

ส่วนต้น

ชื่อคู่มือปฏิบัติงาน:	คู่มือการพัฒนาระบบสารสนเทศด้วยเครื่องมือ Visual Studio Code ของวิทยาลัยสงฆ์พุทธโสธร
ผู้จัดทำ / ตำแหน่ง:	พระมหาประสพสุข ปิยภาณี (สุขล้วน) นักวิชาการคอมพิวเตอร์
ระดับตำแหน่งที่เสนอขอ:	ระดับชำนาญการ (สายปฏิบัติการวิชาชีพ กลุ่มงานคอมพิวเตอร์และเทคโนโลยีสารสนเทศ)
สัดส่วนการจัดทำผลงาน:	จัดทำคนเดียว ๑๐๐% (Single Author) จากการปฏิบัติงานจริงตามภาระงานที่ได้รับมอบหมาย
ความสอดคล้องกับเกณฑ์:	สอดคล้องตามประกาศ มจร. เรื่อง หลักเกณฑ์และวิธีการแต่งตั้งบุคลากรสายปฏิบัติการวิชาชีพ พ.ศ. ๒๕๕๘

๒ บทที่ ๑: บทนำ (Introduction & Scope)

โครงสร้าง มจร.

๑.๑ ความเป็นมาและความสำคัญของกระบวนงาน:
วิทยาลัยสงฆ์พุทธโสธร มุ่งยกระดับการให้บริการวิชาการและกิจการพระพุทธศาสนาสู่ระบบดิจิทัล (Digital Campus) อาทิ ระบบแจ้งซ่อม IT Helpdesk, ระบบจองห้องปฏิบัติการคอมพิวเตอร์, ระบบติดตามคำขอ และศูนย์ดาวน์โหลด การพัฒนาซอฟต์แวร์ภายในมหาวิทยาลัยจำเป็นต้องมี **สภาพแวดล้อมการเขียนโค้ดมาตรฐาน (Integrated Development Environment - IDE)** ที่มีความรวดเร็ว ปลอดภัย และสนับสนุนการทำงานร่วมกันเป็นทีม Visual Studio Code จึงถูกเลือกเป็น IDE หลักของหน่วยงาน

๑.๒ วัตถุประสงค์ของคู่มือ:

1. เพื่อสร้างมาตรฐานการติดตั้ง กำหนดค่า และใช้งาน Visual Studio Code ในการพัฒนาเว็บแอปพลิเคชันของมหาวิทยาลัย
2. เพื่อเพิ่มประสิทธิภาพและลดระยะเวลาในการเขียนโค้ด (Coding), ทดสอบ (Testing) และแก้ไขจุดบกพร่อง (Debugging)
3. เพื่อใช้เป็นแนวปฏิบัติงานสำหรับบุคลากรด้านคอมพิวเตอร์ ทั้งผู้ปฏิบัติงานปัจจุบันและผู้สืบทอดตำแหน่ง (Successor)

๑.๓ ขอบเขตของคู่มือ (Scope of Manual):

- **จุดเริ่มต้น:** เมื่อนักวิชาการคอมพิวเตอร์ได้รับมอบหมายให้พัฒนา ปรับปรุง หรือบำรุงรักษาระบบสารสนเทศ
- **กระบวนการภายใน:** ครอบคลุมการตั้งค่าสภาพแวดล้อม, ติดตั้ง Extension, การเขียนโค้ดตามเกณฑ์ Clean Code, การ Debug ด้วย Xdebug, การควบคุมเวอร์ชันด้วย Git และการตรวจทานความปลอดภัย
- **จุดสิ้นสุด:** สิ้นสุดเมื่อระบบผ่านการทดสอบอัตโนมัติ (Unit Test) และพร้อมส่งมอบขึ้นเซิร์ฟเวอร์ปฏิบัติการจริง
- **กลุ่มผู้ใช้เป้าหมาย:** นักวิชาการคอมพิวเตอร์, โปรแกรมเมอร์, และผู้ดูแลระบบสารสนเทศของ มจร.

๑.๔ คำจำกัดความและคำย่อ (Definitions):

VS Code: ตัวแก้ไขซอร์สโค้ดแบบ Open Source พัฒนาโดย Microsoft

Extension: ส่วนขยายที่เสริมความสามารถให้โปรแกรม

Linting: การวิเคราะห์โค้ดเพื่อหาข้อผิดพลาดทางไวยากรณ์อัตโนมัติ

Git: ระบบควบคุมการเปลี่ยนแปลงเวอร์ชันของซอร์สโค้ด (VCS)

๓ ประโยชน์ที่คาดว่าจะได้รับ (Expected Benefits & Value Creation)

- **ต่อบุคลากรผู้ปฏิบัติงาน**
  มีคู่มือขั้นตอนชัดเจน ลดข้อผิดพลาด (Syntax Error) ลดเวลาตั้งค่าเครื่องทำงานลงกว่า 70%
- **ต่อหน่วยงานวิทยาลัยสงฆ์**
  โค้ดมีรูปแบบมาตรฐานเดียวกัน (Standard Coding Convention) ส่งมอบงานและบำรุงรักษาระบบได้ต่อเนื่อง
- **ต่อมหาวิทยาลัย มจร.**
  ระบบสารสนเทศมีความมั่นคงปลอดภัย ปฏิบัติตาม พ.ร.บ. ไซเบอร์ฯ และคุ้มครองข้อมูลส่วนบุคคล (PDPA)

บทที่ ๒ หน่วยงานและหน้าที่ & บทที่ ๓ หลักเกณฑ์และมาตรฐานวิชาการ

๔ บทที่ ๒: โครงสร้างหน่วยงาน บทบาทหน้าที่ และสายการบังคับบัญชา

๒.๑ โครงสร้างส่วนงานและสายบังคับบัญชา:
วิทยาลัยสงฆ์พุทธโสธร จัดโครงสร้างการบริหารงานแบ่งเป็น สำนักงานวิทยาลัย และสำนักงานวิชาการ โดยงานเทคโนโลยีสารสนเทศ สังกัด **สำนักงานวิชาการ** ปฏิบัติหน้าที่สนับสนุนการเรียนการสอน วิจัย บริการวิชาการ และพัฒนานวัตกรรมดิจิทัล รายงานตรงต่อผู้อำนวยการสำนักงานวิชาการ และรองผู้อำนวยการฝ่ายวิชาการ

๒.๒ บทบาทหน้าที่ตามมาตรฐานกำหนดตำแหน่ง (นักวิชาการคอมพิวเตอร์):

- ศึกษา วิเคราะห์ ออกแบบ พัฒนา ติดตั้ง และบำรุงรักษาระบบเทคโนโลยีสารสนเทศ
- สร้างเครื่องมือดิจิทัล เว็บไซต์ แอปพลิเคชัน และฐานข้อมูลของมหาวิทยาลัย
- บริหารจัดการความมั่นคงปลอดภัยสารสนเทศ การสำรองข้อมูล และแก้ปัญหาขัดข้อง

๒.๓ หน้าที่ที่ได้รับมอบหมายจริง (ภาระงานที่สอดคล้องกับคู่มือ):

ภาระงานหลัก	หลักฐานคำสั่ง / เอกสารอ้างอิง	ผลผลิต (Outputs)
พัฒนาระบบบริการ IT Sothorn	คำสั่งแต่งตั้งคณะทำงานพัฒนาระบบฯ	เว็บแอปพลิเคชัน 6 โมดูลหลัก
ดูแลระบบ Helpdesk & จองห้อง	คำสั่งมอบหมายงานประจำปีการศึกษา	ระบบแจ้งซ่อม + ปฏิทินจองห้อง
ความมั่นคงปลอดภัยและการสำรอง	แผนบริหารความเสี่ยงด้าน IT	ระบบ Audit Log + Backup 3-2-1

๕ บทที่ ๓: หลักเกณฑ์ ระเบียบ กฎหมาย และมาตรฐานวิชาการที่เกี่ยวข้อง

Compliance & Standards

๓.๑ กฎหมายและระเบียบที่ต้องถือปฏิบัติเคร่งครัด:

- **พ.ร.บ. ว่าด้วยการกระทำความผิดเกี่ยวกับคอมพิวเตอร์ (ฉบับที่ ๒) พ.ศ. ๒๕๖๐:** จัดเก็บข้อมูลจราจรคอมพิวเตอร์ (Log files) ไม่น้อยกว่า 90 วัน
- **พ.ร.บ. คุ้มครองข้อมูลส่วนบุคคล พ.ศ. ๒๕๖๒ (PDPA):** ห้าม Hardcode รหัสผ่าน / API Token ลงในซอร์สโค้ดเด็ดขาด ต้องใช้ไฟล์ `.env`
- **ข้อบังคับ มจร. ว่าด้วยการใช้เทคโนโลยีสารสนเทศและการสื่อสาร:** การใช้ซอฟต์แวร์ที่มีลิขสิทธิ์ถูกต้อง (VS Code ได้รับอนุญาตตามสัญญา MIT License)

๓.๒ มาตรฐานวิชาการด้านการพัฒนาซอฟต์แวร์:

- **OWASP Top 10 Web Application Security:** ป้องกัน SQL Injection, XSS, CSRF โดยใช้ Prepared Statements และ Input Sanitization
- **Clean Code Principles:** ตั้งชื่อตัวแปร ฟังก์ชันที่มีความหมาย แยกไฟล์ตามสถาปัตยกรรม Modular Design (แยก Modules / Views / Config)
- **Version Control Standard:** บันทึกประวัติการพัฒนาผ่าน Git ด้วย Semantic Commit Messages เช่น `feat:`, `fix:`, `refactor:`

ข้อกำหนด / เกณฑ์มาตรฐาน	ขั้นตอนในคู่มือปฏิบัติงาน	จุดควบคุมคุณภาพ (Control Point)	หลักฐานตรวจสอบ
การคุ้มครองความลับและสิทธิ์ (PDPA)	กำหนดค่า `.gitignore` ละเว้น `.env` และไดเรกทอรีข้อมูลลับ	สแกน Git Status ก่อน Commit ตรวจจับไฟล์รหัสผ่าน	ไฟล์ .gitignore, Git Log
มาตรฐานความปลอดภัยซอร์สโค้ด	ใช้ Linter & Static Analysis ตรวจสอบช่องโหว่ขณะพิมพ์โค้ด	การแจ้งเตือน Error/Warning แบบเรียลไทม์ใน Problems Panel	ภาพหน้าจอ Problems Panel
มาตรฐานการทดสอบระบบ	รัน Unit Test อัตโนมัติผ่าน Terminal ในตัวของ VS Code	ทดสอบผ่านครบ 100% (87/87 ข้อ) ก่อน Deploy ขึ้นเซิร์ฟเวอร์	รายงานผล tests/run.php

๖ ชุดส่วนขยายมาตรฐานของวิทยาลัยสงฆ์พุทธโสธร (Recommended Extension Stack)

PHP Intelephense
ระบบ Auto-completion, Code Navigation และตรวจสอบ Type อย่างแม่นยำ
Core Engine

Tailwind IntelliSense
แนะนำคลาส CSS, พรีวิวสี และเตือน Class ผิดรูปแบบทันที
UI Styling

GitLens
ตรวจสอบประวัติ Commit ย้อนหลัง ใครแก้ไข บรรทัดไหน เมื่อใด
Audit Trail

PHP Debug (Xdebug)
ตั้ง Breakpoint ตรวจสอบตัวแปร และติดตาม Call Stack ทีละสเต็ป
Troubleshooting

บทที่ ๔: ขั้นตอนการปฏิบัติงานพัฒนาระบบด้วย VS Code (ขั้นตอนที่ ๑ - ๕)

๗ ผังกระบวนการพัฒนาระบบ (Development Workflow / SIPOC Model)

Process Diagram

[1. รับมอบหมายงาน] → [2. เตรียม Workspace & .env] → [3. ติดตั้ง Extension & Linter] → [4. พัฒนาโค้ด (Coding & Refactor)] → [5. Debugging (Xdebug)] → [6. Automated Unit Testing] → [7. Git Commit & Deploy]

๘ ขั้นตอนที่ ๑: ติดตั้งและตั้งค่า Workspace

1. ดาวน์โหลด VS Code จากเว็บไซต์ทางการ (Visual Studio Code Official)
2. เปิดโฟลเดอร์โปรเจกต์ `File → Open Folder...`
3. สร้างโฟลเดอร์ `.vscode/` สำหรับเก็บค่าคอนฟิกเฉพาะโปรเจกต์
4. ตั้งค่าการจัดรูปแบบโค้ดอัตโนมัติเมื่อกดบันทึก (Format on Save)

.vscode/settings.json

```
{
  "editor.tabSize": 4,
  "editor.formatOnSave": true,
  "editor.wordWrap": "on",
  "files.encoding": "utf8mb4",
  "files.eol": "\n",
  "php.validate.executablePath": "/bin/php"
}
```

๙ ขั้นตอนที่ ๒: บริหารจัดการความปลอดภัย (.env)

เพื่อปฏิบัติตามมาตรฐาน OWASP และ พ.ร.บ. PDPA:

1. ห้ามระบุรหัสผ่านฐานข้อมูลในไฟล์ PHP เด็ดขาด
2. สร้างไฟล์ `.env.example` เป็นแม่แบบไม่มีรหัสผ่าน
3. กำหนด `.gitignore` เพื่อตัดไฟล์ `.env` ออกจาก Git

.gitignore (Security Policy)

```
.env
*.log
uploads/documents/*
!uploads/documents/.gitkeep
.DS_Store
```

๑๐ ขั้นตอนที่ ๓: การเขียนโค้ดและ Refactoring

- **IntelliSense:** ใช้ปุ่ม `Ctrl+Space` เพื่อเรียกคำแนะนำฟังก์ชันและคลาสของระบบ มจร.
- **Rename Symbol:** กด F2 เพื่อเปลี่ยนชื่อตัวแปรหรือฟังก์ชันทั้งโปรเจกต์อย่างปลอดภัย
- **Go to Definition:** กด F12 เพื่อเปิดไฟล์ต้นทางของฟังก์ชัน เช่น `clean()`, `logVisitor()`
- **Multi-Cursor:** กด `Alt+Click` เพื่อพิมพ์แก้ไขโค้ดพร้อมกันหลายจุด เพิ่มความรวดเร็ว

๑๑ ขั้นตอนที่ ๔: การตั้งค่าและใช้งาน Debugger (Xdebug)

การติดตามข้อบกพร่องในระดับ Runtime:

1. ติดตั้งส่วนขยาย **PHP Debug (Felix Becker)**
2. สร้างไฟล์คอนฟิก `.vscode/launch.json`
3. คลิกที่แถบข้างเลขบรรทัด (Gutter) เพื่อวาง **Breakpoint (จุดสีแดง)**
4. กด F5 เพื่อเริ่มรันโหมด Debugging

.vscode/launch.json

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Listen for Xdebug",
      "type": "php",
      "request": "launch",
      "port": 9003
    }
  ]
}
```

๑๒ ขั้นตอนที่ ๕: การทดสอบอัตโนมัติ (Testing) และการส่งงานผ่าน Git

Quality Assurance

การรัน Unit Test ผ่าน Integrated Terminal:
กด `Ctrl+`` เพื่อเปิด Terminal ภายใน VS Code และรันชุดทดสอบเพื่อป้องกันข้อผิดพลาดก่อนส่งมอบงาน:

Terminal Command

```
$ php tests/run.php
=== Ticket transitions === PASS
=== Booking overlap === PASS
=== Production Readiness & Security === PASS
--- Passed: 87  Failed: 0
```

การตรวจสอบการเปลี่ยนแปลงผ่าน Source Control:

1. กด `Ctrl+Shift+G` เพื่อดูไฟล์ที่มีการเปลี่ยนแปลง (Diff View)
2. ตรวจสอบ Diff เทียบแบบ Side-by-Side ป้องกันโค้ดแปลกปลอม
3. พิมพ์ข้อความ Commit ตามแบบแผน เช่น `fix: resolve mobile overflow in news.php`
4. กด **Commit & Push** ขึ้น Remote Repository

๑๓ บทที่ ๕: ปัญหา อุปสรรค และแนวทางแก้ไขที่ผู้จัดทำสั่งสมจากประสบการณ์

Lesson Learned

ปัญหาที่พบจากการปฏิบัติงานจริง	สาเหตุที่ตรวจพบ	แนวทางแก้ไขที่ผู้จัดทำดำเนินการ	ผลลัพธ์ (ก่อน-หลัง)
ภาษาไทยเพี้ยน / สระลอย (Character Encoding)	เปิดไฟล์เดิมที่บันทึกเป็น TIS-620 หรือ ANSI ปะปนกับ UTF-8	กำหนดค่า `files.encoding: utf8mb4` และตั้งค่า Auto Guess Encoding ใน settings.json	**แก้ไขสำเร็จ 100%** การแสดงผลภาษาไทยถูกต้อง
Line Ending ขัดแย้งระหว่าง OS (CRLF vs LF)	พัฒนาบน Windows (CRLF) แต่นำไปรันบน Production Server (Linux/FreeBSD - LF)	ตั้งค่า `files.eol: " "` ใน VS Code และสร้างไฟล์ `.gitattributes` บังคับ `* text=auto eol=lf`	**ขจัดปัญหาสิ้นเชิง** รันสคริปต์ Bash ได้ไร้ Error
Xdebug ไม่หยุดทำงานที่ Breakpoint	พอร์ตชนกัน (Port 9000 vs 9003) หรือ Path Mapping บน Web Server ไม่ตรงกับโฟลเดอร์เครื่องพัฒนา	แก้ไข `php.ini` ให้ใช้ xdebug.mode=debug และ xdebug.client_port=9003 พร้อมระบุ pathMappings ใน launch.json	**ลดเวลา Debug** จากเดิม 2 ชม. เหลือไม่เกิน 15 นาที
Extension ตีกันและหน่วยความจำสูง	ติดตั้งส่วนขยายมากเกินไป ทำให้ VS Code ช้าและเกิดอาการค้าง	สร้างไฟล์ `.vscode/extensions.json` แนะนำเฉพาะส่วนขยายที่จำเป็น และปิดส่วนขยายที่ไม่เกี่ยวข้อง	**ประหยัด RAM** เพิ่มความเร็วในการโหลดขึ้น 50%

๑๔ ข้อเสนอแนวคิดการพัฒนางานในระยะต่อไป (Future Work & Proposal)

ชุดที่ ๒ ข้อเสนอพัฒนา

๑. การประยุกต์ใช้ AI-Assisted Coding (Copilot / Gemini Code Assist): เสนอจัดทำกรอบแนวปฏิบัติการใช้ AI Extension เสริมใน VS Code เพื่อช่วยเขียนเอกสาร Function Header, ร่างคำสั่ง SQL และตรวจสอบช่องโหว่ความปลอดภัย โดยมีแนวทางระวังไม่ให้รั่วไหลของข้อมูลส่วนบุคคลตาม PDPA

๒. การทำ Automated CI/CD ร่วมกับ Tasks ใน VS Code: เสนอพัฒนาสคริปต์ `tasks.json` สำหรับกดปุ่มลัด `Ctrl+Shift+B` เพื่อตรวจสอบ Linter, รันชุดทดสอบความปลอดภัย และอัปโหลดไฟล์อัตโนมัติ (Automated Deployment) สู่เซิร์ฟเวอร์ มจร. ลดข้อผิดพลาดของบุคลากร

๑๕ Checklist ตรวจสอบคุณภาพงานพัฒนาระบบก่อน Deploy (Pre-flight Checklist)

1. ด้านความปลอดภัย (Security)
- [✓] ไม่มีรหัสผ่าน/Token ในโค้ด
- [✓] ไฟล์ .env ถูกแยกใน .gitignore
- [✓] Input ทุกจุดผ่านฟังก์ชัน clean()

2. ด้านความถูกต้อง (Correctness)
- [✓] ปราศจาก Syntax Warning ใน VS Code
- [✓] รัน Unit Test ผ่านครบ 87/87 ข้อ
- [✓] ทดสอบการแสดงผลบนจอมือถือ (Responsive)

3. ด้านการควบคุมเวอร์ชัน (Git)
- [✓] เขียนข้อความ Commit มีความหมาย
- [✓] ไม่อัปโหลดไฟล์ขยะระบบ (.DS_Store)
- [✓] ตรวจสอบ Diff ครบถ้วนก่อนส่งมอบ

๑๖ บรรณานุกรม (References)

๑. มหาวิทยาลัยมหาจุฬาลงกรณราชวิทยาลัย. (๒๕๕๘). *ประกาศหลักเกณฑ์และวิธีการแต่งตั้งบุคลากรสายปฏิบัติการวิชาชีพให้ดำรงตำแหน่งระดับชำนาญการ*. พระนครศรีอยุธยา: กองกลาง สำนักงานอธิการบดี.
๒. มหาวิทยาลัยมหาจุฬาลงกรณราชวิทยาลัย. (๒๕๖๐). *ข้อบังคับมหาวิทยาลัยมหาจุฬาลงกรณราชวิทยาลัย ว่าด้วยเทคโนโลยีสารสนเทศและการสื่อสาร*.
๓. Microsoft Corporation. (2026). *Visual Studio Code Documentation and Best Practices*. Retrieved from https://code.visualstudio.com/docs
๔. Open Web Application Security Project (OWASP). (2025). *OWASP Top 10 Web Application Security Risks*. Retrieved from https://owasp.org/