ดิสคอร์ดบอทแสดงตารางการแข่งขันเกม

Discord Esports Schedule Bot

นายพชรกฤต หอมลำดวน 664230042

หมู่เรียน 66/46

โครงงานนี้เป็นส่วนหนึ่งของการศึกษารายวิชา 7204903

โครงงานด้านเทคโนโลยีสารสนเทศ 2

สาขาวิชาเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยราชภัฏนครปฐม

ภาคเรียนที่ 1 ปีการศึกษา 2568

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบัน ชุมชนเกมมิ่ง (Gaming Community) และวงการ Esports ได้เติบโตขึ้นอย่างก้าวกระโดด โดยมีแพลตฟอร์ม Discord เป็นศูนย์กลางหลักในการรวมกลุ่มพูดคุยและติดตามข่าวสารของเหล่าแฟนเกม อย่างไรก็ตาม การติดตามตารางการแข่งขัน Esports (เช่น Valorant, CS2, League of Legends, Dota 2) ยังคงเป็นไปอย่างกระจัดกระจาย ผู้ใช้จำเป็นต้องออกจากแอปพลิเคชัน Discord เพื่อไปค้นหาข้อมูลจากหลายแหล่งที่แตกต่างกัน เช่น VLR.gg, HLTV.org หรือ Liquipedia ซึ่งสร้างความไม่สะดวก สิ้นเปลืองเวลา และทำให้การสนทนาในชุมชนขาดความต่อเนื่อง รูปแบบดั้งเดิมที่ผู้ใช้ต้องคอยสอบถามกันเอง หรือคอยวางลิงก์ตารางแข่งจากภายนอก มีข้อจำกัดและปัญหาหลายประการ เช่น ข้อมูลที่ได้มาอาจไม่ถูกต้อง ไม่ครบถ้วน หรือไม่ใช่วันเวลาที่ตรงกับไทม์โซนของผู้ใช้ เพื่อแก้ไขปัญหาดังกล่าวและยกระดับประสบการณ์การใช้งานของชุมชน Esports ภายใน Discord ผู้จัดทำจึงมีความสนใจที่จะพัฒนาระบบที่นำเทคโนโลยี Bot เข้ามาประยุกต์ใช้

1.2 แนวคิดในการแก้ไขปัญหา

เพื่อแก้ไขปัญหาที่เกิดขึ้น โครงการนี้จึงนำเสนอแนวคิดในการพัฒนา "ระบบบอทสำหรับติดตามตารางแข่งขัน Esports ผ่านแอปพลิเคชัน Discord" (Esports Schedule Bot) โดยใช้ Discord Bot เป็นช่องทางหลักในการสื่อสารและให้บริการข้อมูลแนวคิดหลักคือการสร้างบอทที่ทำงานบนแพลตฟอร์ม Py-cord (หรือไลบรารีที่เกี่ยวข้อง) โดยเน้นการออกแบบส่วนติดต่อผู้ใช้ (User Interface) ที่ทันสมัยและเป็นมิตร แทนที่การใช้คำสั่งแบบพิมพ์ (Slash Command) แบบดั้งเดิม บอทจะใช้ระบบ "แผงควบคุมถาวร" (Persistent Control Panel) ที่ประกอบด้วยปุ่ม (Buttons) และเมนูเลือก (Select Menus) ทำให้ผู้ใช้สามารถโต้ตอบกับบอทได้ง่ายเพียงแค่คลิกโดยบอทจะทำหน้าที่เชื่อมต่อกับฐานข้อมูลกลางผ่าน Pandascore API เพื่อดึงข้อมูลการแข่งขัน, ข้อมูลนักแข่ง และข้อมูลทีมที่แม่นยำและเป็นปัจจุบัน จากนั้นระบบจะแสดงผลข้อมูลในรูปแบบ Embed ที่สวยงามและอ่านง่าย โดยคำตอบทั้งหมดจะเป็นแบบ "ส่วนตัว" (Ephemeral) เพื่อไม่ให้รบกวนการสนทนาในช่องแชทหลัก

1.3 วัตถุประสงค์ของระบบ

1.3.1 เพื่อพัฒนาระบบบอทสำหรับแจ้งเตือนและค้นหาตารางการแข่งขัน Esports ผ่านแอปพลิเคชัน Discord

1.3.2 เพื่อพัฒนาระบบค้นหาข้อมูลโดยเชื่อมต่อกับ Pandascore API ทำให้ผู้ใช้สามารถค้นหาตารางแข่ง, ข้อมูลนักแข่ง และข้อมูลทีมได้อย่างแม่นยำ

1.3.3 เพื่อออกแบบและพัฒนาส่วนติดต่อผู้ใช้ (User Interface) แบบถาวรโดยใช้ Discord UI Components (Buttons, Selects, Modals)

1.3.4 เพื่อเพิ่มประสิทธิภาพและความสะดวกสบายในการเข้าถึงข้อมูล Esports ให้กับผู้ใช้งานในชุมชน Discord

1.4 ขอบเขตการศึกษา

1.4.1 ขอบเขตของระบบ

1.4.1.1 ผู้ดูแลระบบ

สามารถใช้คำสั่ง /setup เพื่อสร้างแผงควบคุมถาวรในช่องแชทที่กำหนด

1.4.1.2 ผู้ใช้งานระบบ

1) สามารถโต้ตอบกับบอทผ่านแผงควบคุมถาวร (Buttons) และเมนูเลือก (Selects)

2) สามารถใช้ปุ่ม "ดูตารางแข่ง" เพื่อค้นหาแมตช์ตามช่วงเวลา (วันนี้, พรุ่งนี้, อาทิตย์นี้) และตามเกมที่เลือก

3) สามารถใช้ปุ่ม "ค้นหานักแข่ง" เพื่อกรอกชื่อนักแข่ง และรับตารางการแข่งขันของนักแข่งคนนั้น

4) สามารถใช้ปุ่ม "ค้นหาทีม" เพื่อกรอกชื่อทีม และรับตารางการแข่งขันของทีมนั้น

5) ได้รับการตอบกลับเป็นข้อความส่วนตัว (Ephemeral) เพื่อไม่ให้รบกวนผู้ใช้อื่นในช่อง

1.4.2 ฮาร์ดแวร์ที่ใช้ในการพัฒนา

1.4.2.1 โน้ตบุ๊ก เอซุส (Asus) รุ่น เก้า เจน อินเทล® คอร์™ ไอห้า-เก้าสามศูนย์ศูนย์เอช (9th Gen Intel® Core™ i5-9300H) เป็นเครื่องหลักสำหรับพัฒนาและทดสอบ ระบบ

1.4.3 ซอฟต์แวร์ที่ใช้ในการพัฒนา

1.4.3.1 ระบบปฏิบัติการ Microsoft Windows 10

1.4.3.2 โปรแกรมแก้ไขโค้ด (Code Editor): Visual Studio Code

1.4.3.3 ภาษาโปรแกรม: Python

1.4.3.4 ไลบรารีและเฟรมเวิร์ก: Py-cord (discord.py), Requests, python-dotenv

1.4.3.5 ฐานข้อมูล (Database): SQLite (สำหรับทำแคชข้อมูลการแข่งขัน และเก็บการตั้งค่าเซิร์ฟเวอร์)

1.4.3.6 ระบบจัดการเวอร์ชัน (VCS): Git และ GitHub

1.4.3.7 Gemini: ใช้เป็นเครื่องมือช่วยในการให้คำปรึกษาและสนับสนุนการพัฒนา (Development Support Tool) เช่น ช่วยวิเคราะห์และแก้ไขข้อบกพร่อง (Debugging), เสนอแนวทางในการปรับปรุงโค้ด (Refactoring) และช่วยในการร่างเอกสารโครงงาน

1.4.4 บริการ API ที่ใช้

1.4.4.1 Discord Bot API: ใช้สำหรับสร้าง Chatbot, รับ-ส่งข้อความ, และสร้างส่วนติดต่อผู้ใช้ (UI Components)

1.4.4.2 Pandascore API: ใช้สำหรับค้นหาข้อมูลตารางการแข่งขัน, รายละเอียดนักแข่ง และข้อมูลทีม Esports

1.5 ประโยชน์ที่ได้คาดว่าจะได้รับ

1.5.1 ลดขั้นตอนและเวลาสำหรับผู้ใช้ในการค้นหาตารางการแข่งขัน Esports

1.5.2 เพิ่มความสะดวกสบายให้ผู้ใช้สามารถเข้าถึงข้อมูลได้โดยตรงจากภายในแอปพลิเคชัน Discord โดยไม่ต้องสลับไปใช้เบราว์เซอร์

1.5.3 สร้างศูนย์รวมข้อมูล (Information Hub) ที่มีความแม่นยำและเป็นปัจจุบันสำหรับชุมชน Esports

1.5.4 เพิ่มปฏิสัมพันธ์ (Engagement) ภายในเซิร์ฟเวอร์ Discord และสร้างประสบการณ์ใช้งาน (User Experience) ที่ดีให้กับสมาชิก

บทที่ 2

หลักการและทฤษฎีที่เกี่ยวข้อง

ในการดำเนินงานโครงงาน "ระบบบอทสำหรับติดตามตารางแข่งขัน Esports ผ่านแอปพลิเคชัน Discord" ผู้จัดทำได้ศึกษาทฤษฎี หลักการทำงาน และเทคโนโลยีต่างๆ ที่เกี่ยวข้อง เพื่อใช้เป็นพื้นฐานในการออกแบบและพัฒนาซอฟต์แวร์ให้สามารถทำงานได้ตรงตามวัตถุประสงค์ที่กำหนดไว้ โดยมีเทคโนโลยีและองค์ความรู้ที่เกี่ยวข้องดังต่อไปนี้

2.1 ระบบงานเดิม

ในการใช้งานแพลตฟอร์ม Discord สำหรับชุมชน Esports ในปัจจุบัน การเข้าถึงข้อมูลตารางการแข่งขันยังคงต้องอาศัยกระบวนการที่ทำด้วยตนเอง (Manual Process) ซึ่งมีขั้นตอนดังนี้

1) การค้นหาจากแหล่งภายนอก: ผู้ใช้ที่ต้องการทราบตารางแข่ง จำเป็นต้องออกจากแอปพลิชัน Discord เพื่อไปเปิดเบราว์เซอร์และค้นหาข้อมูลจากเว็บไซต์เฉพาะทางต่างๆ เช่น VLR.gg (สำหรับ Valorant), HLTV.org (สำหรับ CS2), หรือ Liquipedia (สำหรับทุกเกม) ซึ่งข้อมูลมีการกระจัดกระจาย

2) การสอบถามในช่องแชท: ผู้ใช้มักจะพิมพ์คำถามลงในช่องแชท เช่น "วันนี้มีแข่งกี่โมง?" และต้องรอให้สมาชิกคนอื่นที่ทราบข้อมูลมาตอบ

3) การแบ่งปันลิงก์: ผู้ใช้ที่พบข้อมูลจะคัดลอกลิงก์จากเว็บไซต์ภายนอกมาวางในช่องแชท ซึ่งสร้างความไม่สะดวกในการอ่าน และทำให้ข้อความสำคัญอื่นๆ ถูกดันหายไป

กระบวนการเหล่านี้สิ้นเปลืองเวลา, ขาดความแม่นยำ, สร้างความไม่ต่อเนื่องในการสนทนา และสร้างภาระให้กับผู้ใช้ในการค้นหาข้อมูลด้วยตนเอง

2.2 ระบบงานอื่นที่เกี่ยวข้อง

จากการศึกษา พบว่ามีการพัฒนาระบบที่เกี่ยวข้องกับการให้บริการข้อมูล Esports ในหลายรูปแบบ แต่ยังมีช่องว่างที่โครงงานนี้สามารถเข้าไปแก้ไขได้ ดังนี้

1) เว็บไซต์และแอปพลิเคชันเฉพาะทาง (เช่น Strafe, Juked, VLR): แอปพลิเคชันเหล่านี้เป็นระบบที่ยอดเยี่ยมในการรวบรวมข้อมูลตารางแข่ง, ผลการแข่ง และสถิติ อย่างไรก็ตาม ระบบ

เหล่านี้เป็นแพลตฟอร์มภายนอก (External Platforms) ผู้ใช้ยังคงต้องสลับแอปพลิเคชันไปมา และไม่สามารถดึงข้อมูลมาใช้ประกอบการสนทนาใน Discord ได้ทันที

2) บอท Discord ทั่วไป (เช่น MEE6, Dyno): เป็นบอทสำหรับการจัดการเซิร์ฟเวอร์ (Moderation) หรือสร้างความบันเทิงทั่วไป บอทเหล่านี้ใช้ระบบคำสั่งแบบพิมพ์ (Slash Command) เป็นหลัก และไม่ได้ถูกออกแบบมาเพื่อเชื่อมต่อกับ API ข้อมูล Esports โดยเฉพาะ

3) บอท Esports แบบดั้งเดิม: มีผู้พัฒนาบอท Esports จำนวนหนึ่งที่ใช้ภาษา Python หรือ Node.js แต่บอทส่วนใหญ่มักใช้การโต้ตอบด้วยคำสั่งพิมพ์ เช่น /schedule valorant ซึ่งผู้ใช้ต้องเรียนรู้คำสั่งเอง, ขาดความยืดหยุ่น (เช่น การเลือกช่วงเวลา) และการแสดงผลไม่เป็นมิตรต่อผู้ใช้เท่าที่ควร

โดยสรุป ยังไม่มีระบบที่ผนวกรวมข้อมูล Esports เข้ากับการใช้งาน Discord ได้อย่างสมบูรณ์ ด้วยส่วนติดต่อผู้ใช้ (UI) ที่เป็นมิตรและใช้งานง่าย ซึ่งเป็นเป้าหมายหลักของโครงงานนี้

2.3 องค์ความรู้และเทคโนโลยีที่เกี่ยวข้อง

2.3.1 ภาษาไพทอน (Python)

เป็นภาษาโปรแกรมระดับสูง (High-level Programming Language) ที่ได้รับความนิยมสูง ถูกออกแบบมาให้มีโครงสร้างไวยากรณ์ (Syntax) ที่เรียบง่าย อ่านและทำความเข้าใจได้ง่าย ภาษาไพทอนมีระบบนิเวศของไลบรารี (Library Ecosystem) ที่กว้างขวาง ซึ่งในโครงงานนี้ได้นำมาใช้เป็นภาษาหลักในการพัฒนา และใช้ไลบรารีที่เกี่ยวข้องได้แก่ Py-cord (สำหรับการพัฒนาบอท), Requests (สำหรับการเชื่อมต่อ API) และ python-dotenv (สำหรับการจัดการความปลอดภัยของ API Key)

2.3.2 ดิสคอร์ด บอท เอพีไอ และไลบรารี ไพ-คอร์ด (Discord Bot API and Py-cord library) เป็นเทคโนโลยีหลักที่ใช้ในการสร้างบอทและส่วนติดต่อผู้ใช้

2.3.2.1 Discord Bot API: คือช่องทางการสื่อสารที่ Discord เปิดให้นักพัฒนาสามารถสร้างแอปพลิเคชัน (บอท) เพื่อโต้ตอบกับผู้ใช้และเซิร์ฟเวอร์ได้ การสื่อสารหลักใช้โปรโตคอล WebSocket (Gateway API) เพื่อรับ "เหตุการณ์" (Events) ต่างๆ เช่น การกดปุ่ม และใช้ REST API ในการส่งคำสั่งกลับไปยัง Discord เช่น การส่งข้อความตอบกลับ

2.3.2.2 ไลบรารี Py-cord: เป็นไลบรารี (Wrapper) สำหรับภาษาไพทอน ที่ช่วยจัดการความซับซ้อนในการเชื่อมต่อกับ Discord Bot API ทำให้นักพัฒนาสามารถเขียนโค้ดเพื่อตอบสนองต่อ Events ต่างๆ ได้ง่ายขึ้น

2.3.2.3 ส่วนติดต่อผู้ใช้ (Discord UI Components): โครงงานนี้ได้ใช้ฟีเจอร์ UI Components ของ Py-cord อย่างเต็มรูปแบบ เพื่อสร้างประสบการณ์การใช้งานที่ดี ได้แก่ Buttons (ปุ่มกด), Select Menus (เมนูเลือก) และ Modals (หน้าต่าง Pop-up)

2.3.3 แพนด้าสคอร์ เอพีไอ (Pandascore API) เป็นบริการ API ภายนอก (Third-party API) ที่ให้บริการข้อมูลด้าน Esports แบบครบวงจร โดยรวบรวมข้อมูลการแข่งขัน, ข้อมูลทีม, และข้อมูลผู้เล่นจากเกมชั้นนำ (Valorant, CS2, LoL, Dota 2) การทำงานเป็นแบบ RESTful ซึ่งผู้พัฒนาส่ง HTTP Request (เช่น GET) พร้อม API Key เพื่อขอข้อมูลในรูปแบบ JSON (JavaScript Object Notation) ซึ่งง่ายต่อการนำไปประมวลผลต่อในโปรแกรม

2.3.4 ฐานข้อมูล เอสคิวแอลไลต์ (SQLite Database) เป็นระบบจัดการฐานข้อมูลแบบเชิงสัมพันธ์ (RDBMS) ที่มีจุดเด่นคือการทำงานแบบ Serverless โดยจัดเก็บฐานข้อมูลทั้งหมดไว้ในไฟล์เดียว ในโครงงานนี้ SQLite ถูกนำมาใช้ 2 วัตถุประสงค์หลักคือ:

2.3.4.1 การทำแคช (Caching): ใช้ตาราง matches เพื่อเก็บข้อมูลตารางแข่งล่วงหน้า และอัปเดตเป็นรอบ (ทุก 15 นาที) เพื่อให้บอทตอบสนองได้รวดเร็วและประหยัดโควต้า API

2.3.4.2 การเก็บการตั้งค่า (Configuration Storage): ใช้ตาราง guild_settings เพื่อบันทึกว่าในแต่ละเซิร์ฟเวอร์ (Guild) ได้ตั้งค่าให้บอททำงานในช่องแชทใด ทำให้บอทรองรับการทำงานในหลายเซิร์ฟเวอร์พร้อมกันได้

2.3.5 เจมิไน (Gemini) เป็นเครื่องมือช่วยในการให้คำปรึกษาและสนับสนุนการพัฒนา (Development Support Tool) โดยมีบทบาทในการช่วยวิเคราะห์ข้อผิดพลาด (Debugging), เสนอแนวทางในการปรับปรุงโค้ด (Code Refactoring) และช่วยในการร่างเอกสารโครงงาน (Documentation)

บทที่ 3

วิธีการดำเนินงาน

บทนี้เป็นการอธิบายแผนการดำเนินงานและเทคนิคที่ใช้ในการพัฒนาโครงงาน "ระบบบอทสำหรับติดตามตารางแข่งขัน Esports ผ่านแอปพลิเคชัน Discord" เพื่อให้ได้ผลลัพธ์ตามวัตถุประสงค์ที่กำหนดไว้ โดยมีขั้นตอนการดำเนินงานดังนี้

3.1 การศึกษาเบื้องต้น

จากการศึกษาระบบงานเดิม (Manual Process) ในการติดตามการแข่งขัน Esports บนแพลตฟอร์ม Discord พบปัญหาหลักคือ ความไม่สะดวกในการเข้าถึงข้อมูล ผู้ใช้ต้องสลับแอปพลิเคชันเพื่อค้นหาข้อมูลจากเว็บไซต์ภายนอก (เช่น VLR.gg, HLTV) ซึ่งข้อมูลกระจัดกระจายและใช้เวลาค้นหานาน

ดังนั้น จึงมีแนวคิดในการแก้ปัญหาโดยการพัฒนาระบบบอทอัตโนมัติ (Esports Schedule Bot) ที่ทำหน้าที่เป็นศูนย์กลางข้อมูล (Information Hub) ภายใน Discord โดยตรง ผู้ใช้สามารถโต้ตอบผ่านส่วนติดต่อผู้ใช้ (UI) แบบกราฟิก (ปุ่มและเมนู) เพื่อดึงข้อมูลตารางแข่งที่อัปเดตและแม่นยำมาแสดงผลได้ทันที

3.2 การกำหนดความต้องการของระบบ

การกำหนดความต้องการของระบบ (Requirements Specification) เป็นการกำหนดขอบเขตและทรัพยากรที่จำเป็นในการพัฒนา ดังนี้

3.2.1 ขอบเขตของระบบ แบ่งตามบทบาทของผู้ใช้งาน 2 ระดับ:

3.2.1.1 ผู้ดูแลระบบ

1) สามารถใช้คำสั่ง /setup เพื่อสร้างและติดตั้ง "แผงควบคุมถาวร" ในช่องแชทที่กำหนด

3.2.1.2 ผู้ใช้งานทั่วไป

1) สามารถโต้ตอบกับ "แผงควบคุมถาวร" ผ่านปุ่มกด

2) สามารถใช้ฟังก์ชัน "ดูตารางแข่ง" โดยกรองตามช่วงเวลา (วันนี้, พรุ่งนี้, อาทิตย์นี้) และกรองตามเกม (Valorant, CS2, LoL, Dota 2)

3) สามารถใช้ฟังก์ชัน "ค้นหานักแข่ง" โดยเลือกเกม, กรอกชื่อ, และยืนยันนักแข่งที่ถูกต้องจากเมนูผลการค้นหา

4) สามารถใช้ฟังก์ชัน "ค้นหาทีม" โดยเลือกเกม, กรอกชื่อ, และยืนยันทีมที่ถูกต้องจากเมนูผลการค้นหา

5) การตอบกลับของบอททั้งหมดจะเป็นแบบส่วนตัว (Ephemeral) เพื่อไม่ให้รบกวนการสนทนาหลัก

3.2.2 ฮาร์ดแวร์ที่ใช้กับระบบงาน

3.2.2.1 เครื่องคอมพิวเตอร์สำหรับพัฒนา (Development): ใช้สำหรับเขียนโค้ด, ทดสอบระบบเบื้องต้น, และใช้ Git

3.2.2.2 เซิร์ฟเวอร์คลาวด์ (Production): ใช้บริการ Render (PaaS) สำหรับรันบอทให้ทำงานออนไลน์ 24 ชั่วโมง

3.2.2.3 อุปกรณ์ผู้ใช้งาน (Client): คอมพิวเตอร์ หรือ สมาร์ทโฟน ที่ติดตั้งแอปพลิเคชัน Discord

3.2.3 ซอฟต์แวร์ที่ใช้กับระบบงาน

3.2.3.1 ระบบปฏิบัติการ: Microsoft Windows (สำหรับการพัฒนา) และ Linux (สำหรับเซิร์ฟเวอร์ Render)

3.2.3.2 โปรแกรมแก้ไขโค้ด: Visual Studio Code

3.2.3.3 ภาษาโปรแกรม: Python 3

3.2.3.4 ไลบรารีหลัก: Py-cord (สำหรับ Discord Bot), Requests (สำหรับยิง API), python-dotenv (สำหรับจัดการ Key)

3.2.3.5 ระบบฐานข้อมูล: SQLite (สำหรับเก็บแคชข้อมูลแมตช์ และการตั้งค่าเซิร์ฟเวอร์)

3.2.3.6 ระบบจัดการเวอร์ชัน: Git และ GitHub

3.2.3.7 API ภายนอก: Discord Bot API และ Pandascore API

3.2.3.8 บริการโฮสติ้ง: Render

3.2.3.9 เครื่องมือสนับสนุน: Gemini (AI ช่วยในการเขียนโค้ดและแก้ไขบัค)

3.3 การออกแบบระบบ

การออกแบบระบบ ประกอบไปด้วยการออกแบบสถาปัตยกรรม, ฐานข้อมูล และส่วนติดต่อผู้ใช้

3.3.1 การออกแบบระบบ (System Architecture)

ในการออกแบบระบบได้ใช้แผนภาพกระแสข้อมูล (Data Flow Diagram - DFD) เพื่ออธิบายการไหลของข้อมูลในระบบ

3.3.1.1 แผนภาพกระแสข้อมูลระดับภาพรวม (Context Diagram) แผนภาพ Context Diagram (DFD Level 0) แสดงภาพรวมของระบบ โดยมี Process "ระบบบอท Esports" (Process 0) อยู่ตรงกลาง และเชื่อมต่อกับ External Entities 4 ส่วน:

1) User (ผู้ใช้): ส่งข้อมูล "คำขอ" (การกดปุ่ม, เลือกเมนู) และรับ "ผลลัพธ์" (Embed ตารางแข่ง)

2) Administrator (ผู้ดูแล): ส่งข้อมูล "คำสั่งตั้งค่า" (/setup)

3) Discord API: เป็นช่องทางหลักในการรับ-ส่งข้อมูลและ Events ทั้งหมดระหว่างผู้ใช้และระบบบอท

4) Pandascore API: ระบบส่ง "คำขอข้อมูล" (API Request) และรับ "ข้อมูล Esports" (JSON Response) กลับมา

3.3.1.2 แผนภาพกระแสข้อมูลระดับที่ 1 (Data Flow Diagram Level 1) แผนภาพ DFD Level 1 แสดงกระบวนการย่อย 3 ส่วนหลักภายในระบบ:

1) Process 1.0 (จัดการแคชข้อมูล): เป็นกระบวนการเบื้องหลัง (Background Task) ที่ทำงานอัตโนมัติ เชื่อมต่อกับ Pandascore API เพื่อดึงข้อมูลแมตช์ และเขียนข้อมูลลงใน Data Store D1 (matches)

2) Process 2.0 (จัดการคำขอผู้ใช้): เป็นกระบวนการหลักที่รอรับ Event จาก User (ผ่าน Discord API)

กรณี "ดูตารางแข่ง": จะอ่านข้อมูลจาก Data Store D1 (matches)

กรณี "ค้นหานักแข่ง/ทีม": จะส่ง Request สดไปยัง Pandascore API

ส่ง "ผลลัพธ์" กลับไปให้ User (ผ่าน Discord API)

3) Process 3.0 (จัดการคำสั่งผู้ดูแล): รับคำสั่ง /setup จาก Administrator (ผ่าน Discord API) และเขียนข้อมูลการตั้งค่าลงใน Data Store D2 (guild_settings)

3.3.2 การออกแบบฐานข้อมูล (Database Design)

3.3.2.1 แผนภาพแสดงความสัมพันธ์ของข้อมูล (Entity-Relationship Diagram)

3.3.2.2 พจนานุกรมข้อมูล (Data Dictionary)

พจนานุกรมข้อมูล (Data Dictionary) คือ รายละเอียดค าอธิบายข้อมูล ต่างๆ ในฐานข้อมูล เช่น ล าดับ (No) คุณสมบัติ (Attribute) ค าอธิบาย (Description) ขนาด (Size) ประเภท (Type) ประเภทคีย์ (Key Type) ซึ่งพจนานุกรมขอ้มูลของระบบมีข้อมูล ดังต่อไปนี้

ตารางที่ 3.1 ตารางเก็บข้อมูลแมตช์

ลำดับ (No)	คุณสมบัติ (Attribute)	คำอธิบาย (Description)	ขนาด (Width)	ประเภท (Type)	ประเภทคีย์ (Key Type)
1	match_id	ID แมตช์	-	INT	PK
2	game_slug	ชื่อเกม	-	TEXT	-
3	league_name	ชื่อลีก	-	TEXT	-
4	team1_name	ชื่อทีมที่ 1	-	TEXT	-
5	Team2_name	ชื่อทีมที่ 2	-	TEXT	-
6	begin_at	เวลาเริ่ม	-	TEXT	-
7	stream_url	ลิ้งค์เข้าชม	-	TEXT	-

ตารางที่ 3.2 ตารางเก็บข้อมูลช่องแชท

ลำดับ (No)	คุณสมบัติ (Attribute)	คำอธิบาย (Description)	ขนาด (Width)	ประเภท (Type)	ประเภทคีย์ (Key Type)
1	guild_id	ID เซิร์ฟเวอร์	-	INT	PK
2	dedicated_channel_id	ID ของช่องแชท	-	INT	-

3.3.3 การออกแบบส่วนติดต่อกับผู้ใช้

3.3.3.1 ออกแบบผลลัพธ์ (Output Design) การแสดงผลลัพธ์หลักจะใช้ Discord Embeds ซึ่งเป็นการ์ดข้อมูลที่สวยงามและอ่านง่าย แบ่งเป็น 3 ประเภท:

1) Embed ตารางแข่ง: แสดงหัวข้อ (ชื่อเกม, ช่วงเวลา) และรายการแมตช์ (ทีม vs ทีม, เวลา, ลิงก์ชม)

2) Embed ค้นหานักแข่ง: แสดงรูปโปรไฟล์นักแข่ง, ชื่อนักแข่ง, และรายการแมตช์ ของทีม

3) Embed ค้นหาทีม: แสดงโลโก้ทีม, ชื่อทีม, และรายการแมตช์ของทีม

หมายเหตุ: ผลลัพธ์ทั้งหมดนี้จะถูกส่งแบบ Ephemeral (เห็นเฉพาะผู้ใช้)

3.3.3.2 ออกแบบรายงาน (Report Design) ในบริบทของบอทนี้ "รายงาน" หมายถึง แผงควบคุมถาวร (Main Control Panel) ที่สร้างจากคำสั่ง /setup ซึ่งเป็น Embed สาธารณะที่คงอยู่ในช่องแชท ประกอบด้วย:

1) ส่วนอธิบายวิธีการใช้งานบอท

2) ปุ่มควบคุม 3 ปุ่ม (ดูตารางแข่ง, ค้นหานักแข่ง, ค้นหาทีม)

3.3.3.3 ออกแบบส่วนนำเข้า (Input Design)

ระบบรับข้อมูลเข้าจากผู้ใช้ผ่าน UI Components 3 รูปแบบหลัก:

1) Buttons (ปุ่ม):

1.1) ปุ่มในแผงควบคุมหลัก (3 ปุ่ม)

1.2) ปุ่มเลือกช่วงเวลา ("วันนี้", "พรุ่งนี้", "อาทิตย์นี้")

2) Select Menus (เมนูเลือก):

2.1) เมนูเลือกเกม (Valorant, CS2, LoL, Dota 2)

2.2) เมนูยืนยันผลการค้นหา (แสดงรายชื่อนักแข่ง/ทีม ที่ค้นเจอ)

3) Modals (หน้าต่าง Pop-up):

3.1) หน้าต่างสำหรับกรอกชื่อนักแข่ง

3.2) หน้าต่างสำหรับกรอกชื่อทีม

3.4 การพัฒนาระบบ

ผู้พัฒนาระบบได้มีการออกแบบขั้นตอนการพัฒนาระบบ ดังต่อไปนี้

3.4.1 ศึกษาข้อมูลและเตรียมเครื่องมือ: ศึกษาเอกสารของ Py-cord และ Pandascore API, สมัคร API Key, และตั้งค่าสภาพแวดล้อม (Python, Git, VS Code)บุคคลที่เกี่ยวข้องกับระบบงาน

3.4.2 กำหนดโครงสร้างโปรเจกต์: วางโครงสร้างไฟล์ (main.py, database.py, pandascore_api.py, config.py) และตั้งค่าการจัดการข้อมูลลับ (.env, .gitignore)

3.4.3 พัฒนาส่วน API และ ฐานข้อมูล: พัฒนาฟังก์ชันใน pandascore_api.py (สำหรับดึงข้อมูล) และ database.py (สำหรับสร้างและจัดการตาราง)

3.4.4 พัฒนาตรรกะหลักและ UI: พัฒนาส่วน main.py สร้างคลาสสำหรับ UI Components (Views, Buttons, Selects, Modals) และฟังก์ชันสร้าง Embeds

3.4.5 พัฒนาระบบเบื้องหลัง: พัฒนา Background Task (update_matches_cache) สำหรับอัปเดตแคช และคำสั่ง /setup สำหรับผู้ดูแล

3.4.6 พัฒนาระบบเบื้องหลัง: พัฒนา Background Task (update_matches_cache) สำหรับอัปเดตแคช และคำสั่ง /setup สำหรับผู้ดูแล

3.4.7 ทดสอบระบบโดยใช้งานจริงในกลุ่มผู้ใช้จำลอง

3.4.8 ปรับปรุงและจัดทำเอกสารคู่มือการใช้งาน

3.5 การทดสอบระบบ

หลังจากพัฒนาระบบเสร็จสมบูรณ์ ผู้พัฒนาได้ดำเนินการทดสอบระบบในหลายระดับ เพื่อประเมินความถูกต้องและประสิทธิภาพการทำงาน

3.5.1 การทดสอบแต่ละส่วน ผู้จัดทำได้แบ่งการทดสอบเป็น 3 ระดับ

1) Unit Test: ทดสอบฟังก์ชันย่อยๆ ว่าทำงานถูกต้องหรือไม่ เช่น การเชื่อมต่อ API, การอ่าน/เขียนฐานข้อมูล

2) Integration Test: ทดสอบกระบวนการทำงานจริงของผู้ใช้ เช่น ทดสอบ Flow การค้นหาทีม (กรอก "T1" และต้องได้เมนูเลือก T1/T1 Academy กลับมา)

3) Deployment Test: ทดสอบระบบบนเซิร์ฟเวอร์ Render เพื่อให้แน่ใจว่าบอทออนไลน์ได้ 24/7, Background Task ทำงานอัตโนมัติ, และอ่าน API Keys ได้ถูกต้อง

3.5.2 การทดสอบแบบเพิ่มเติม

การติดตั้งระบบประกอบด้วยการนำซอร์สโค้ดอัปโหลดขึ้น GitHub, เชื่อมต่อ Repository เข้ากับ Render, และตั้งค่า Environment Variables (API Keys) เพื่อ Deploy บอท. จากนั้น, ใช้ลิงก์ OAuth2 ที่กำหนดสิทธิ์แล้วเชิญบอทเข้าเซิร์ฟเวอร์ และสุดท้ายผู้ดูแลใช้คำสั่ง /setup เพื่อสร้างแผงควบคุมในช่องแชทที่ต้องการ

3.5.3 การทดสอบระบบรวม ประเมินผลการทำงาน 4 ด้าน:

1) ความถูกต้อง: ระบบค้นหาทำงานได้แม่นยำ สามารถแยกแยะทีมชื่อคล้ายกันได้ และระบบแคช (ดูตารางแข่ง) ทำงานตามที่ออกแบบ (ข้อมูลล่าช้าไม่เกิน 15 นาที)

2) ประสิทธิภาพ: การตอบสนองของบอท (โดยเฉพาะจากระบบแคช) อยู่ในเกณฑ์ที่รวดเร็ว

3) เสถียรภาพ: บอทสามารถทำงานออนไลน์ได้อย่างต่อเนื่องบน Render

4) การใช้งาน: ผู้ใช้พึงพอใจกับการใช้งานผ่านปุ่มและเมนู มากกว่าบอทแบบพิมพ์คำสั่งแบบดั้งเดิม

3.5.4 การทดสอบระบบเพื่อส่งมอบงานนำระบบให้ผู้ใช้งานจริงทดลองใช้และให้ข้อเสนอแนะ เพื่อปรับปรุงก่อนส่งมอบงาน อย่างสมบูรณ์

บรรณานุกรม

Pycord Development. (2568). *Py-cord Documentation*. แหล่งที่มา: https://docs.pycord.dev/en/stable/

BorntoDev. (2567). *สอนสร้าง Discord Bot ด้วย Python (discord.py) [2024]*. แหล่งที่มา: https://www.borntodev.com/2021/08/21/discord-bot-with-python-2024/