ดิสคอร์ดบอทแสดงตารางการแข่งขันเกม

Discord Esports Schedule Bot

นายพชรกฤต หอมลำดวน 664230042

หมู่เรียน 66/46

โครงงานนี้เป็นส่วนหนึ่งของการศึกษารายวิชา 7203602

หัวข้อพิเศษด้านเทคโนโลยีสารสนเทศ

สาขาวิชาเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยราชภัฏนครปฐม

ภาคเรียนที่ 1 ปีการศึกษา 2568

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันชุมชนเกมมิ่ง และวงการอีสปอร์ตได้เติบโตขึ้นอย่างก้าวกระโดดโดยมีแพลตฟอร์ม Discord เป็นศูนย์กลางหลักในการรวมกลุ่มพูดคุยและติดตามข่าวสารของเหล่าแฟนเกม อย่างไรก็ตามการติดตามตารางการแข่งขันอีสปอร์ต เช่น Valorant CS2 LoL Dota2 ยังคงเป็นไปอย่างกระจัดกระจายผู้ใช้จำเป็นต้องออกจากแอปพลิเคชัน Discord เพื่อไปค้นหาข้อมูลจากหลายแหล่งที่แตกต่างกัน เช่น VLR.gg HLTV.org หรือ Liquipedia ซึ่งสร้างความไม่สะดวก และทำให้การสนทนาในชุมชนขาดความต่อเนื่อง

รูปแบบดั้งเดิมที่ผู้ใช้ต้องคอยสอบถามกันเอง หรือคอยวางลิงก์ตารางแข่งจากภายนอกมีข้อจำกัดและปัญหาหลายประการ เช่นข้อมูลที่ได้มาอาจไม่ถูกต้องไม่ครบถ้วน หรือไม่ใช่วันเวลาที่ตรงกับไทม์โซนของผู้ใช้เพื่อแก้ไขปัญหาดังกล่าวและยกระดับประสบการณ์การใช้งานของชุมชนอีสปอร์ตภายใน Discord ผู้จัดทำจึงมีความสนใจที่จะพัฒนาระบบที่นำเทคโนโลยี Bot เข้ามาประยุกต์ใช้

1.2 แนวคิดในการแก้ไขปัญหา

เพื่อแก้ไขปัญหาที่เกิดขึ้นโครงการนี้จึงนำเสนอแนวคิดในการพัฒนาระบบบอทสำหรับแสดงตารางแข่งขันอีสปอร์ตผ่านแอปพลิเคชัน Discord โดยใช้ Discord Bot เป็นช่องทางหลักในการสื่อสารและให้บริการข้อมูลแนวคิดหลักคือการสร้างบอทที่ทำงานบนแพลตฟอร์ม Py-cord โดยเน้นการออกแบบส่วนติดต่อผู้ใช้ที่ทันสมัยและเป็นมิตรแทนที่การใช้คำสั่งแบบพิมพ์แบบดั้งเดิมบอทจะใช้ระบบแผงควบคุมถาวรที่ประกอบด้วยปุ่มและเมนูเลือกทำให้ผู้ใช้สามารถโต้ตอบกับบอทได้ง่ายเพียงแค่คลิกโดยบอทจะทำหน้าที่เชื่อมต่อกับฐานข้อมูลกลางผ่าน Pandascore API เพื่อดึงข้อมูลการแข่งขัน ข้อมูลนักแข่ง และข้อมูลทีมที่เป็นปัจจุบันจากนั้นระบบจะแสดงผลข้อมูลในรูปแบบ Embed ที่อ่านง่ายโดยคำตอบทั้งหมดจะเป็นแบบส่วนตัวเพื่อไม่ให้รบกวนการสนทนาในช่องแชทหลัก

1.3 วัตถุประสงค์ของระบบ

1.3.1 เพื่อพัฒนาระบบบอทสำหรับแสดงตารางการแข่งขันอีสปอร์ตของเกม Valorant CS2 LoL Dota2 ผ่านแอปพลิเคชัน Discord

1.3.2 เพื่อพัฒนาระบบค้นหาข้อมูลโดยเชื่อมต่อกับ Pandascore API ทำให้ผู้ใช้สามารถค้นหาตารางแข่ง ข้อมูลนักแข่ง และข้อมูลทีมของเกม Valorant CS2 LoL Dota2 ได้อย่างแม่นยำ

1.3.3 เพื่อออกแบบและพัฒนาส่วนติดต่อผู้ใช้แบบถาวร

1.3.4 เพื่อเพิ่มประสิทธิภาพและความสะดวกสบายในการเข้าถึงข้อมูลอีสปอร์ตของเกม Valorant CS2 LoL Dota2 ให้กับผู้ใช้งานในชุมชน Discord

1.4 ขอบเขตการศึกษา

1.4.1 ขอบเขตของระบบ

1.4.1.1 แสดงตารางแข่งขันอีสปอร์ตเฉพาะเกม Valorant CS2 LoL Dota2

1.4.1.2 ผู้ดูแลระบบในชุมชน Discord

ก) สามารถใช้คำสั่ง /setup เพื่อสร้างแผงควบคุมถาวรในช่องแชทที่ต้องการ

1.4.1.3 ผู้ใช้งานระบบในชุมชน Discord

ก) สามารถโต้ตอบกับบอทผ่านแผงควบคุมถาวร และเมนูเลือก

ข) สามารถใช้ปุ่มดูตารางแข่งเพื่อค้นหาแมตช์ในช่วงเวลาที่กำหนด (วันนี้ พรุ่งนี้ อาทิตย์นี้) ของเกมที่เลือก

ค) สามารถใช้ปุ่มค้นหานักแข่งเพื่อกรอกชื่อนักแข่ง และรับตารางการแข่งขันของนักแข่งคนนั้น

ง) สามารถใช้ปุ่มค้นหาทีมเพื่อกรอกชื่อทีม และรับตารางการแข่งขันของทีมนั้น

จ) ได้รับการตอบกลับเป็นข้อความส่วนตัวเพื่อไม่ให้รบกวนผู้ใช้อื่นในช่อง

1.4.2 ฮาร์ดแวร์ที่ใช้ในการพัฒนา

1.4.2.1 โน้ตบุ๊กหน่วยประมวลผลกลางไอห้า-เก้าสามศูนย์ศูนย์เอช (i5-9300H) แรมยี่สิบสี่จิกะไบต์ (24 GB) หน่วยความจำห้าร้อยสิบสองจิกะไบต์ (512 GB) เป็นเครื่องหลักสำหรับพัฒนา และทดสอบระบบ

1.4.3 ซอฟต์แวร์ที่ใช้ในการพัฒนา

1.4.3.1 ระบบปฏิบัติการไมโครซอฟท์วินโดว์สิบ (Microsoft Windows 10)

1.4.3.2 วิชวลสตูดิโอโค้ด (Visual Studio Code)

1.4.3.3 ภาษาโปรแกรมไพทอน (Python)

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

1.4.3.5 เอสคิวแอลไลท์ (SQLite) สำหรับทำแคชข้อมูลการแข่งขัน และเก็บการตั้งค่าเซิร์ฟเวอร์

1.4.3.6 ระบบจัดการเวอร์ชัน (VCS) Git และ GitHub

1.4.3.7 เจมินิ (Gemini) ใช้เป็นเครื่องมือช่วยในการให้คำปรึกษา และสนับสนุนการพัฒนา เช่นช่วยวิเคราะห์และแก้ไขข้อบกพร่อง เสนอแนวทางในการปรับปรุงโค้ด

1.4.4 บริการ API ที่ใช้ (APIs)

1.4.4.1 ดิสคอร์ดบอทเอพีไอ (Discord Bot API) ใช้สำหรับสร้าง Chatbot รับ-ส่งข้อความ และสร้างส่วนติดต่อผู้ใช้

1.4.4.2 แพนด้าสกอร์เอพีไอ (Pandascore API) ใช้สำหรับค้นหาข้อมูลตารางการแข่งขัน รายละเอียดนักแข่ง และข้อมูลทีมอีสปอร์ต

1.5 ประโยชน์ที่ได้คาดว่าจะได้รับ

1.5.1 ลดขั้นตอนและเวลาสำหรับผู้ใช้ในการค้นหาตารางการแข่งขันอีสปอร์ต

1.5.2 เพิ่มความสะดวกสบายให้ผู้ใช้สามารถเข้าถึงข้อมูลได้โดยตรงจากภายในแอปพลิเคชัน Discord โดยไม่ต้องสลับไปใช้เบราว์เซอร์

1.5.3 สร้างศูนย์รวมข้อมูลที่มีความแม่นยำและเป็นปัจจุบันสำหรับชุมชนอีสปอร์ต

1.5.4 เพิ่มปฏิสัมพันธ์ภายในเซิร์ฟเวอร์ Discord และสร้างประสบการณ์ใช้งานที่ดีให้กับสมาชิก

1.6 นิยามเฉพาะ

เพื่อให้เกิดความเข้าใจตรงกันในการดำเนินงานโครงงาน ผู้วิจัยได้กำหนดนิยามศัพท์เฉพาะที่เกี่ยวข้องกับระบบงาน ดังนี้

1.6.1 ดิสคอร์ด (Discord) คือแพลตฟอร์มการสื่อสารฟรีที่ผู้ใช้สามารถแชทด้วยข้อความ เสียง และวิดีโอ โดยผู้ใช้สามารถเข้าร่วมหรือสร้างเซิร์ฟเวอร์ หรือชุมชนออนไลน์เพื่อพูดคุยกันตามความสนใจต่างๆ

1.6.2 บอท (Bot) หมายถึง โปรแกรมอัตโนมัติที่ทำงานอยู่บนเซิร์ฟเวอร์ และสามารถโต้ตอบกับผู้ใช้งานในเซิร์ฟเวอร์ Discord ได้ผ่านคำสั่งและการกดปุ่ม

1.6.3 API (Application Programming Interface) หมายถึง ช่องทางการเชื่อมต่อและแลกเปลี่ยนข้อมูลระหว่างซอฟต์แวร์ ในโครงงานนี้หมายถึงการเชื่อมต่อระหว่างบอทกับ Discord และบอทกับ Pandascore

1.6.4 ไพ-คอร์ด (Py-cord) หมายถึง ไลบรารีของภาษาไพทอน ที่ใช้เป็นเครื่องมือหลักในการพัฒนาบอท ทำหน้าที่จัดการการเชื่อมต่อกับ Discord Bot API

1.6.5 ส่วนประกอบ (UI Components) หมายถึง ส่วนประกอบหน้าต่างการใช้งานแบบกราฟิกที่บอทใช้โต้ตอบกับผู้ใช้ เช่น ปุ่มกด, เมนูเลือก และหน้าต่าง

1.6.6 เอ็มเบด (Embed) คือ ข้อความในรูปแบบพิเศษที่ใช้แสดงข้อมูลในลักษณะที่สวยงามและน่าสนใจยิ่งขึ้น แทนที่จะเป็นข้อความธรรมดาๆ

1.6.7 แผงควบคุมถาวร (Persistent Control Panel) หมายถึง ข้อความ Embed พร้อมปุ่มกดที่บอทสร้างขึ้นจากคำสั่ง /setup ซึ่งข้อความนี้จะคงอยู่ถาวรในช่องแชท และปุ่มต่างๆสามารถทำงานได้ตลอดเวลา (แม้บอทจะรีสตาร์ท) ทำหน้าที่เป็นเมนูหลักให้ผู้ใช้เริ่มโต้ตอบกับบอท

1.6.8 ข้อความชั่วคราว (Ephemeral Message) หมายถึง การตอบกลับของบอทที่แสดงผลแบบส่วนตัวให้เห็นได้เฉพาะผู้ใช้งานที่เรียกคำสั่งนั้นๆเท่านั้น เพื่อป้องกันการรบกวนผู้ใช้อื่น

บทที่ 2

หลักการและทฤษฎีที่เกี่ยวข้อง

ในการดำเนินงานโครงงานดิสคอร์ดบอทแสดงตารางแข่งขันเกม ผู้จัดทำได้ศึกษาทฤษฎี หลักการทำงาน และเทคโนโลยีต่างๆที่เกี่ยวข้องเพื่อใช้เป็นพื้นฐานในการออกแบบและพัฒนาซอฟต์แวร์ให้สามารถทำงานได้ตรงตามวัตถุประสงค์ที่กำหนดไว้โดยมีเทคโนโลยีและองค์ความรู้ที่เกี่ยวข้องดังต่อไปนี้

2.1 ระบบงานเดิม

ในการใช้งานแพลตฟอร์ม Discord สำหรับชุมชนอีสปอร์ตในปัจจุบันการเข้าถึงข้อมูลตารางการแข่งขันยังคงต้องอาศัยกระบวนการที่ทำด้วยตนเองซึ่งมีขั้นตอนดังนี้

2.1.1 การค้นหาจากแหล่งภายนอก ผู้ใช้ที่ต้องการทราบตารางแข่งจำเป็นต้องออกจากแอปพลิชัน Discord เพื่อไปเปิดเบราว์เซอร์และค้นหาข้อมูลจากเว็บไซต์เฉพาะทางต่างๆ เช่น VLR.gg HLTV.org หรือ Liquipedia ซึ่งข้อมูลมีการกระจัดกระจาย

2.1.2 การสอบถามในช่องแชท ผู้ใช้มักจะพิมพ์คำถามลงในช่องแชท เช่น วันนี้มีแข่งกี่โมง และต้องรอให้สมาชิกคนอื่นที่ทราบข้อมูลมาตอบ

2.1.3 การแบ่งปันลิงก์ ผู้ใช้ที่พบข้อมูลจะคัดลอกลิงก์จากเว็บไซต์ภายนอกมาวางในช่องแชทซึ่งสร้างความไม่สะดวกในการอ่าน และทำให้ข้อความสำคัญอื่นๆถูกดันหายไป กระบวนการเหล่านี้อาจทำสิ้นเปลืองเวลา ขาดความแม่นยำ สร้างความไม่ต่อเนื่องในการสนทนา และสร้างภาระให้กับผู้ใช้ในการค้นหาข้อมูลด้วยตนเอง

2.2 ระบบงานอื่นที่เกี่ยวข้อง

จากการศึกษาพบว่ามีการพัฒนาระบบที่เกี่ยวข้องกับการให้บริการข้อมูลอีสปอร์ตในหลายรูปแบบแต่ยังมีช่องว่างที่โครงงานนี้สามารถเข้าไปแก้ไขได้ ดังนี้

2.2.1 เว็บไซต์และแอปพลิเคชันเฉพาะทาง เช่น Strafe Juked VLR แอปพลิเคชันเหล่านี้เป็นระบบที่ยอดเยี่ยมในการรวบรวมข้อมูลตารางแข่ง ผลการแข่ง และสถิติอย่างไรก็ตามระบบ

เหล่านี้เป็นแพลตฟอร์มภายนอกผู้ใช้ยังคงต้องสลับแอปพลิเคชันไปมา และไม่สามารถดึงข้อมูลมาใช้ประกอบการสนทนาใน Discord ได้ทันที

2.2.2 บอท Discord ทั่วไป เช่น MEE6 และ Dyno เป็นบอทสำหรับการจัดการเซิร์ฟเวอร์หรือสร้างความบันเทิงทั่วไปบอทเหล่านี้ใช้ระบบคำสั่งแบบพิมพ์เป็นหลัก และไม่ได้ถูกออกแบบมาเพื่อเชื่อมต่อกับ API ข้อมูลอีสปอร์ตโดยเฉพาะ

2.2.3 บอทอีสปอร์ตแบบดั้งเดิมมีผู้พัฒนาบอทอีสปอร์ตจำนวนหนึ่งที่ใช้ภาษา Python หรือ Node.js แต่บอทส่วนใหญ่มักใช้การโต้ตอบด้วยคำสั่งพิมพ์ เช่น /schedule valorant ซึ่งผู้ใช้ต้องเรียนรู้คำสั่งเอง ขาดความยืดหยุ่น และการแสดงผลไม่เป็นมิตรต่อผู้ใช้เท่าที่ควร

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 ดิสคอร์ดบอทเอพีไอ และไลบรารีไพ-คอร์ด (Discord Bot API and Py-cord library) เป็นเทคโนโลยีหลักที่ใช้ในการสร้างบอทและส่วนติดต่อผู้ใช้

2.3.2.1 ดิสคอร์ดบอทเอพีไอ (Discord Bot API) คือช่องทางการสื่อสารที่ Discord เปิดให้นักพัฒนาสามารถสร้างแอปพลิเคชันหรือบอทเพื่อโต้ตอบกับผู้ใช้และเซิร์ฟเวอร์ได้การสื่อสารหลักใช้โปรโตคอล WebSocket (Gateway API) เพื่อรับเหตุการณ์ต่างๆ เช่นการกดปุ่ม และใช้ REST API ในการส่งคำสั่งกลับไปยัง Discord เช่นการส่งข้อความตอบกลับ

2.3.2.2 ไลบรารีไพ-คอร์ด (Py-cord library) เป็นไลบรารีสำหรับภาษาไพทอนที่ช่วยจัดการความซับซ้อนในการเชื่อมต่อกับ Discord Bot API ทำให้นักพัฒนาสามารถเขียนโค้ดเพื่อตอบสนองต่อเหตุการณ์ต่างๆได้ง่ายขึ้น

2.3.2.3 ส่วนติดต่อผู้ใช้ดิสคอร์ด (Discord UI Components) โครงงานนี้ได้ใช้ฟีเจอร์ UI Components ของ Py-cord เพื่อสร้างประสบการณ์การใช้งานที่ดีได้แก่ ปุ่มกด เมนูเลือก และหน้าต่าง Pop-up

2.3.3 แพนด้าสคอร์เอพีไอ (Pandascore API) เป็นบริการ API ที่ให้บริการข้อมูลด้านอีสปอร์ตแบบครบวงจรโดยรวบรวมข้อมูลการแข่งขัน ข้อมูลทีม และข้อมูลผู้เล่นจากเกม Valorant CS2 LoL Dota2 การทำงานเป็นแบบ RESTful ซึ่งผู้พัฒนาส่ง HTTP Request (เช่น GET) พร้อม API Key เพื่อขอข้อมูลในรูปแบบ JSON (JavaScript Object Notation) ซึ่งง่ายต่อการนำไปประมวลผลต่อในโปรแกรม

2.3.4 ฐานข้อมูลเอสคิวแอลไลต์ (SQLite Database) เป็นระบบจัดการฐานข้อมูลแบบเชิงสัมพันธ์ที่มีจุดเด่นคือการทำงานแบบ Serverless โดยจัดเก็บฐานข้อมูลทั้งหมดไว้ในไฟล์เดียว ในโครงงานนี้ SQLite ถูกนำมาใช้ 2 วัตถุประสงค์หลักคือ

2.3.4.1 การทำแคช (Caching) ใช้ตาราง matches เพื่อเก็บข้อมูลตารางแข่งล่วงหน้า และอัปเดตเป็นรอบทุก 15 นาทีเพื่อให้บอทตอบสนองได้รวดเร็วและประหยัดโควต้า API

2.3.4.2 การเก็บการตั้งค่าใช้ตาราง guild_settings เพื่อบันทึกว่าในแต่ละเซิร์ฟเวอร์ได้ตั้งค่าให้บอททำงานในช่องแชทใดทำให้บอทรองรับการทำงานในหลายเซิร์ฟเวอร์พร้อมกันได้

2.3.5 เจมินิ (Gemini) เป็นเครื่องมือช่วยในการให้คำปรึกษาและสนับสนุนการพัฒนาโดยมีบทบาทในการช่วยวิเคราะห์ข้อผิดพลาด และเสนอแนวทางในการปรับปรุงโค้ด

บทที่ 3

วิธีการดำเนินงาน

บทนี้เป็นการอธิบายแผนการดำเนินงานและเทคนิคที่ใช้ในการพัฒนาดิสคอร์ดบอทแสดงตารางแข่งขันเกมเพื่อให้ได้ผลลัพธ์ตามวัตถุประสงค์ที่กำหนดไว้ โดยมีขั้นตอนการดำเนินงานดังนี้

3.1 การศึกษาเบื้องต้น

จากการศึกษาระบบงานเดิมในการติดตามการแข่งขัน Esports บนแพลตฟอร์ม Discord พบปัญหาหลักคือความไม่สะดวกในการเข้าถึงข้อมูลผู้ใช้ต้องสลับแอปพลิเคชันเพื่อค้นหาข้อมูลจากเว็บไซต์ภายนอก เช่น VLR.gg HLTV ซึ่งข้อมูลกระจัดกระจายและใช้เวลาค้นหานาน จึงมีแนวคิดในการแก้ปัญหาโดยการพัฒนาระบบบอทที่ทำหน้าที่เป็นศูนย์กลางข้อมูลภายใน 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

ก) สามารถใช้คำสั่ง /setup เพื่อสร้างและติดตั้งแผงควบคุมถาวรในช่องแชทที่กำหนด

3.2.1.2 ผู้ใช้งานทั่วไป

ก) สามารถโต้ตอบกับแผงควบคุมถาวรผ่านปุ่มกด

ข) สามารถใช้ฟังก์ชันดูตารางแข่งโดยกรองตามช่วงเวลา (วันนี้ พรุ่งนี้ อาทิตย์นี้) และกรองตามเกม Valorant CS2 LoL Dota2

ค) สามารถใช้ฟังก์ชันค้นหานักแข่งโดยเลือกเกม กรอกชื่อ และยืนยันนักแข่งที่ถูกต้องจากเมนูผลการค้นหา

ง) สามารถใช้ฟังก์ชันค้นหาทีมโดยเลือกเกม กรอกชื่อ และยืนยันทีมที่ถูกต้องจากเมนูผลการค้นหา

จ) การตอบกลับของบอททั้งหมดจะเป็นแบบส่วนตัวเพื่อไม่ให้รบกวนการสนทนาหลัก

3.2.2 ฮาร์ดแวร์ที่ใช้กับระบบงาน

3.2.2.1 โน๊ตบุ๊กหรือคอมพิวเตอร์ ใช้สำหรับเขียนโค้ดทดสอบระบบ

3.2.2.3 อุปกรณ์ผู้ใช้งาน คอมพิวเตอร์ หรือสมาร์ทโฟนที่ติดตั้งแอปพลิเคชัน Discord

3.2.3 ซอฟต์แวร์ที่ใช้กับระบบงาน

3.2.3.1 ระบบปฏิบัติการ Microsoft Windows

3.2.3.2 โปรแกรมแก้ไขโค้ด Visual Studio Code

3.2.3.3 ภาษาโปรแกรม Python

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

3.2.3.7 Discord Bot API และ Pandascore API

3.3 การออกแบบระบบ

การออกแบบระบบประกอบไปด้วยการออกแบบฐานข้อมูล 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) แสดงภาพรวมของระบบโดยมี Process ระบบบอทอีสปอร์ต (Process 0) อยู่ตรงกลาง และเชื่อมต่อกับ External Entities 4 ส่วน

ก) User (ผู้ใช้) ส่งข้อมูลคำขอ (การกดปุ่ม, เลือกเมนู) และรับผลลัพธ์ (Embed ตารางแข่ง)

ข) Administrator (ผู้ดูแล) ส่งข้อมูลคำสั่งตั้งค่า (/setup)

ค) Discord API เป็นช่องทางหลักในการรับ-ส่งข้อมูลและเหตุการณ์ทั้งหมดระหว่างผู้ใช้และระบบบอท

ง) Pandascore API ระบบส่งคำขอข้อมูล และรับข้อมูลการแข่งขันกลับมา

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

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

ข) Process 2.0 (จัดการคำขอผู้ใช้) เป็นกระบวนการหลักที่รอรับเหตุการณ์จากผู้ใช้ (ผ่าน Discord API) กรณีดูตารางแข่งจะอ่านข้อมูลจาก Data Store D1 (matches) กรณี ค้นหานักแข่งหรือทีมจะส่ง Request สดไปยัง Pandascore API ส่งผลลัพธ์กลับไปให้ผู้ใช้ (ผ่าน Discord API)

ค) 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 รายละเอียดคำอธิบายข้อมูลต่างๆในฐานข้อมูล เช่น ลำดับ (No) คุณสมบัติ (Attribute) คำอธิบาย (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 ออกแบบผลลัพธ์การแสดงผลลัพธ์หลักจะใช้ Discord Embeds ซึ่งเป็นการ์ดข้อมูลที่สวยงามและอ่านง่ายแบ่งเป็น 3 ประเภท

ก) Embed ตารางแข่งแสดงหัวข้อ ชื่อเกม ช่วงเวลา และรายการแมตช์

ข) Embed ค้นหานักแข่งแสดงรูปโปรไฟล์นักแข่ง ชื่อนักแข่ง และรายการแมตช์ของทีม

ค) Embed ค้นหาทีมแสดงโลโก้ทีม ชื่อทีม และรายการแมตช์ของทีม

3.3.3.2 ออกแบบรายงานในบริบทของบอทนี้รายงาน หมายถึงแผงควบคุมถาวรที่สร้างจากคำสั่ง /setup ซึ่งเป็น Embed สาธารณะที่คงอยู่ในช่องแชทประกอบด้วย

ก) ส่วนอธิบายวิธีการใช้งานบอท

ข) ปุ่มควบคุม 3 ปุ่ม (ดูตารางแข่ง ค้นหานักแข่ง ค้นหาทีม)

3.3.3.3 ออกแบบส่วนนำเข้าระบบรับข้อมูลเข้าจากผู้ใช้ผ่าน UI Components 3 รูปแบบหลัก

ก) ปุ่ม (button)

ก.1) ปุ่มในแผงควบคุมหลัก 3 ปุ่ม (ดูตารางแข่ง ค้นหานักแข่ง ค้นหาทีม)

ข) เมนูเลือก (Select Menus)

ข.1) ปุ่มเลือกช่วงเวลา (วันนี้ พรุ่งนี้ อาทิตย์นี้)

ข.2) เมนูเลือกเกม (Valorant, CS2, LoL, Dota 2)

ข.3) เมนูยืนยันผลการค้นหา (แสดงรายชื่อนักแข่ง/ทีม ที่ค้นเจอ)

ค) Modals (หน้าต่าง Pop-up)

ค.1) หน้าต่างสำหรับกรอกชื่อนักแข่ง

ค.2) หน้าต่างสำหรับกรอกชื่อทีม

3.4 การพัฒนาระบบ

ผู้พัฒนาระบบได้มีการออกแบบขั้นตอนการพัฒนาระบบ ดังต่อไปนี้

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

3.4.2 กำหนดโครงสร้างโปรเจกต์วางโครงสร้างไฟล์ (main.py, database.py, pandascore_api.py, config.py) และตั้งค่าการจัดการข้อมูลลับ (.env, .gitignore)

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

3.4.4 พัฒนาตรรกะหลักและUI 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 ทดสอบระบบโดยใช้งานจริงในกลุ่มผู้ใช้จำลอง

3.5 การทดสอบระบบ

หลังจากพัฒนาระบบเสร็จสมบูรณ์ ผู้พัฒนาได้ดำเนินการทดสอบระบบในหลายระดับ เพื่อประเมินความถูกต้องและประสิทธิภาพการทำงาน

3.5.1 การทดสอบแต่ละส่วน ผู้จัดทำได้แบ่งการทดสอบเป็น 3 ระดับ

ก) ทดสอบฟังก์ชันย่อยๆว่าทำงานถูกต้องหรือไม่ เช่นการเชื่อมต่อ API ถูกต้องหรือไม่

ข) ทดสอบกระบวนการทำงานจริงของผู้ใช้ เช่นทดสอบการค้นหาทีม (กรอก T1 และต้องได้เมนูเลือก T1 หรือ T1 Academy กลับมา)

3.5.3 การทดสอบระบบรวม ประเมินผลการทำงาน 3 ด้าน

ก) ความถูกต้องระบบค้นหาทำงานได้แม่นยำ สามารถแยกแยะทีมชื่อคล้ายกันได้ และระบบแคช (ดูตารางแข่ง) ทำงานตามที่ออกแบบ (ข้อมูลล่าช้าไม่เกิน 15 นาที)

ข) ประสิทธิภาพการตอบสนองของบอท (โดยเฉพาะจากระบบแคช) อยู่ในเกณฑ์ที่รวดเร็ว

ค) การใช้งานผู้ใช้พึงพอใจกับการใช้งานผ่านปุ่มและเมนู มากกว่าบอทแบบพิมพ์คำสั่งแบบดั้งเดิม

3.5.4 การทดสอบระบบเพื่อส่งมอบงานนำระบบให้ผู้ใช้งานจริงทดลองใช้และให้ข้อเสนอแนะเพื่อปรับปรุงก่อนส่งมอบงานอย่างสมบูรณ์

บรรณานุกรม

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