ระบบการเช็คชื่อเข้าเรียนอัตโนมัติด้วยการจดจำใบหน้า

Automatic attendance system with facial recognition

จีรภัทร สุภาพินิจ

โครงงานวิจัยฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรห้องเรียนวิทยาศาสตร์ในโรงเรียน

โดยการกำกับดูแลของมหาวิทยาลัย (โครงการ วมว.)

กระทรวงการอุดมศึกษา วิทยาศาสตร์ วิจัยและนวัตกรรม

โรงเรียนสาธิต “พิบูลบำเพ็ญ” มหาวิทยาลัยบูรพา

ปีการศึกษา 2566

หัวข้อโครงงานวิจัย	ระบบการเช็คชื่อเข้าเรียนอัตโนมัติด้วยการจดจำใบหน้า	
ชื่อนักเรียน	นายจีรภัทร สุภาพินิจ	รหัสนักเรียน 29157
อาจารย์ที่ปรึกษา	นายอุกฤษ นาฎแก้ว	
ปีการศึกษา	2566	

คณะกรรมการการสอบได้พิจารณาโครงงานวิจัยฉบับนี้แล้วเห็นสมควรรับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรตามหลักสูตรห้องเรียนวิทยาศาสตร์ในโรงเรียน โดยการกำกับดูแลของมหาวิทยาลัย (โครงการ วมว.) กระทรวงการอุดมศึกษา วิทยาศาสตร์ วิจัยและนวัตกรรม

.....

อาจารย์ที่ปรึกษาโครงงาน

(อาจารย์ อุกฤษ นาฎแก้ว)

.....

(อาจารย์ ดร.ปรียานุช ใจหาญ)

ผู้สอนวิชาโครงงานวิทยาศาสตร์

โครงการ วมว. ศูนย์โรงเรียนสาธิต “พิบูลบำเพ็ญ” มหาวิทยาลัยบูรพา อนุมัติรับโครงงานวิทยาศาสตร์ฉบับนี้ไว้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรห้องเรียนวิทยาศาสตร์ในโรงเรียน โดยการกำกับดูแลของมหาวิทยาลัย (โครงการ วมว.) กระทรวงการอุดมศึกษา วิทยาศาสตร์ วิจัยและนวัตกรรม

.....	.....
(นางอาพันธ์ชนิต เจนจิต)	(รองศาสตราจารย์อุษาวดี ตันติวรานุรักษ์)
ผู้รักษาการแทนผู้อำนวยการโรงเรียนสาธิต “พิบูลบำเพ็ญ”	คณบดีคณะวิทยาศาสตร์
วันที่ ...../...../.....	วันที่ ...../...../.....

โรงเรียนสาธิต “พิบูลบำเพ็ญ” มหาวิทยาลัยบูรพา: ห้องเรียนวิทยาศาสตร์ในโรงเรียน โดยการกำกับดูแลของมหาวิทยาลัย (โครงการ วมว.) กระทรวงการอุดมศึกษา วิทยาศาสตร์ วิจัยและนวัตกรรม

29157 นายจีรภัทร สุภาพินิจ: ระบบการเช็คชื่อเข้าเรียนอัตโนมัติด้วยการจดจำใบหน้า (Automatic attendance system with facial recognition)

อาจารย์ที่ปรึกษา : นายอุกฤษ นาฏแก้ว

จำนวนหน้า : 73 หน้า ปีการศึกษา 2566

ปัจจุบัน เทคโนโลยีและนวัตกรรมเข้ามามีบทบาทสำคัญต่อการพัฒนาการดำรงชีวิตประจำวันและหารแก้ปัญหาต่างๆ ซึ่งปัญหาอย่างหนึ่งในด้านการศึกษาคือกระบวนการเช็คชื่อการเข้าเรียน ซึ่งอาจใช้เวลานานและไม่สะดวกสบาย เพื่อแก้ไขปัญหานี้ ผู้จัดทำจึงเสนอวิธีแก้ปัญหาโดยใช้เทคโนโลยีการจดจำใบหน้าเพื่อทำให้กระบวนการเป็นไปโดยอัตโนมัติและลดกระบวนการที่ทำด้วยมือ เช่น บัตรประจำตัวนักเรียนหรือการตรวจสอบด้วยมือ ระบบที่นำเสนอนี้มีจุดประสงค์เพื่อออกแบบและใช้งานระบบการเข้าร่วมอัตโนมัติที่เป็นมิตรต่อผู้ใช้ แม่นยำสูง และใช้งานได้จริง ระบบจะตรวจจับใบหน้า จดจำ และเปรียบเทียบกับฐานข้อมูลเพื่อยืนยันตัวตน นอกจากนี้ยังสามารถจดจำใบหน้าใหม่ได้โดยอัตโนมัติ จึงเหมาะสำหรับองค์กรและสถาบันการศึกษา ระบบประกอบด้วยเว็บเซิร์ฟเวอร์ ฐานข้อมูลแบบเรียลไทม์ และ Raspberry Pi ในฐานะคนงาน ในการประเมินความสามารถ ระบบจะผ่านการทดสอบความแม่นยำและการทดสอบการรองรับภาระ

ในการประเมินความสามารถ ระบบจะผ่านการทดสอบความแม่นยำ การทดสอบโหลด จากการทดสอบพบว่าความแม่นยำที่ดีที่สุดสำหรับการจดจำใบหน้าเกิดขึ้นที่การหมุน 0 องศารอบแกน y และ 0 องศารอบแกน x สำหรับการทดสอบความแม่นยำในระยะทางต่างๆ ระบบมีความแม่นยำสูงสุดเมื่อมีระยะต่ำกว่า 4 เมตรในความละเอียด SD และ 16 เมตรในความละเอียด HD นอกจากนี้ การทดสอบภาระยังแสดงให้เห็นว่าระบบสามารถรองรับคนได้ถึง 9 คนด้วยเฟรมต่อวินาทีที่เหมาะสม

Piboonbumpen Demonstration School Burapha University: Science classroom in University-affiliated school project, Ministry of Higher Education Science Research and Innovation

29157 Mr. Jeerabhat Supapinit (Automatic attendance system with facial recognition)
Advisor: Mr. Aukrit Natkaeo

Number of Page: 73 P. Academic Year 2023.

Nowadays, technology and innovation play a critical role in improving daily life and solving problems. One such problem in the educational field is the attendance verification process, which can be time-consuming and inconvenient. To address this issue, the authors propose a solution using facial recognition technology to automate the process and reduce manual methods such as student ID cards or hand checks. The proposed system aims to design and implement a user-friendly, highly accurate, and practical automatic attendance system. The system will detect faces, recognize them, and compare them with a database to confirm identity. It will also be capable of recognizing new faces automatically, making it suitable for organizations and educational institutions. The system consists of a webserver, cloud storage, a real-time database, and a Raspberry Pi as a worker. To evaluate its capability, the system will undergo accuracy testing and load testing.

ประกาศคุณูปการ

โครงงานวิจัยทางวิทยาศาสตร์ฉบับนี้สำเร็จลงด้วยดีด้วยความช่วยเหลือจาก อ.อุกฤษ นาฎแก้ว อาจารย์ที่ปรึกษาโครงงาน ที่กรุณาให้ความรู้ คำปรึกษาทางวิชาการและแนะแนวทางที่ถูกต้อง ตลอดจนแก้ไขข้อบกพร่องต่างๆ จึงขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอขอบพระคุณ อ. อุกฤษ นาฎแก้ว โครงการ วมว. คณะวิทยาศาสตร์ มหาวิทยาลัยบูรพา ที่ให้คำแนะนำและกรุณามาเป็นกรรมการสอบโครงงานวิจัยนี้

ขอขอบพระคุณ อ. วราพงษ์ เสนาภักดิ์ โครงการ วมว. คณะวิทยาศาสตร์ มหาวิทยาลัยบูรพา ที่ให้คำแนะนำและกรุณามาเป็นคณะกรรมการสอบโครงงานวิจัยนี้

ขอขอบพระคุณ อ. ชนากานต์ จินากูล โครงการ วมว. คณะวิทยาศาสตร์ มหาวิทยาลัยบูรพา ที่ให้คำแนะนำและกรุณามาเป็นคณะกรรมการสอบโครงงานวิจัยนี้

กราบขอบพระคุณบิดาและมารดาที่เป็นกำลังใจ ให้คำปรึกษา ตลอดจนค่าใช้จ่ายต่างๆ รวมถึงญาติ พี่น้องและเพื่อนๆที่คอยให้กำลังใจและช่วยเหลือเสมอมาจนทำให้โครงงานนี้เสร็จสมบูรณ์

จีรภัทร สุภาพินิจ

23 กันยายน 2566

สารบัญ

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบัน เทคโนโลยีและนวัตกรรมเข้ามามีบทบาทสำคัญในการสร้างสรรค์สิ่งต่างๆ เพื่อให้เกิดความสะดวกสบายในการดำเนินชีวิต หรือเพื่อลดปัญหาที่อาจเป็นจุดเปลี่ยนสำคัญของสังคม และหนึ่งในปัญหาที่เกิดในสังคมโรงเรียนคือการเช็คชื่อเข้าโรงเรียนและการเช็คชื่อเข้าห้องเรียน ขั้นตอนเหล่านี้จะถูกทำโดยการใช้บัตรนักเรียนหรือการเช็คด้วยมือ ซึ่งวิธีการเหล่านี้ใช้เวลานานและไม่สะดวกต่อคนที่ลืมบัตรหรือบัตรหายซึ่งทำให้เกิดการติดขัดและแออัดของผู้คนได้ แต่ปัญหาเหล่านี้สามารถแก้ได้โดยการใช้ปัญญาประดิษฐ์ในการเช็คชื่อเข้าเรียน

ปัญญาประดิษฐ์ (Artificial Intelligence) หรือ AI เป็นเครื่องจักรที่มีฟังก์ชันทีมีความสามารถในการทำความเข้าใจ เรียนรู้องค์ความรู้ต่างๆ เช่น การรับรู้ การเรียนรู้ การให้เหตุผล และการแก้ปัญหาต่างๆ ซึ่งในที่นี้เทคโนโลยีนี้ถูกใช้ในกระบวนการในการจดจำใบหน้า (Face Recognition) โดยอาศัยการเปรียบเทียบใบหน้าที่ตรวจจับกับใบหน้าในฐานข้อมูล พร้อมวิเคราะห์องค์ประกอบต่าง ๆ และเอกลักษณ์เพื่อยืนยันว่าใบหน้าที่ตรวจจับได้ถูกต้องและตรงกับบุคคลนั้น วิธีการนี้ทำให้การเช็คชื่อใช้เวลาเร็วขึ้นและยังสะดวกสบายขึ้น ระบบเช็คชื่อนี้สามารถนำมาประยุกต์ใช้กับหน่วยงาน องค์กร หรือสถานศึกษาที่มีขนาดไม่ใหญ่มากได้ โดยระบบนี้มีความโดดเด่นกว่าระบบเช็คชื่ออัตโนมัติอื่นๆ โดยระบบนี้สามารถตรวจจับว่าข้อมูลเป็นรูปคนจริงหรือรูปถ่ายได้และยังสามารถจดจำใบหน้าใหม่ๆได้โดยอัตโนมัติ

ด้วยเหตุนี้ทางผู้จัดทำจึงสนใจในการสร้างระบบการเช็คชื่อเข้าเรียนอัตโนมัติด้วยการจดจำใบหน้าซึ่งมีความแม่นยำสูงและสามารถนำไปประยุกต์ใช้ได้จริง

1.2 วัตถุประสงค์การวิจัย

1.2.1 เพื่อสร้างระบบเช็คชื่ออัตโนมัติที่มีความแม่นยำสูงและใช้งานได้ง่าย

1.2.2 เพื่อเพิ่มความสะดวกสบายในการเช็คชื่อเข้าโรงเรียน ทั้งกับนักเรียนและโรงเรียน

1.3 สมมติฐานของการทำงานวิจัย

1.3.1 สามารถตรวจจับใบหน้าได้จริง และมีความแม่นยำมากกว่า 90%

1.3.2 สามารถสร้างเอกสารเช็คชื่อนักเรียนได้จากข้อมูลที่ได้

1.4 ขอบเขตการดำเนินงานของกิจกรรม

ออกแบบและสร้างระบบในการเช็คชื่อเข้าเรียนอัตโนมัติ โดยเขียนด้วยภาษา python และวางระบบในบอร์ด raspberry pi การความแม่นยำของระบบทดลองโดยที่เช็คชื่อจริงกับกลุ่มคนตัวอย่าง(ถ่ายรูปเป็นหมู่) ซึ่งจะแบ่งการทดสอบเป็นสองหัวข้อได้แก่ ความแม่นยำ (accuracy) และภาระ(load) โดยความแม่นยำจะวัดโดยนำจำนวนการระบุตัวตนที่ถูกต้องเปรียบเทียบกับการระบุตัวตนทั้งหมด ซึ่งแบ่งการทดลองเป็นการทดลองย่อย 4 การทดลองได้แก่การหมุนใบหน้ารอบแกน x แกน y และระยะการตรวจจับของใบหน้า การวัดภาระสามารถวัดได้โดยจำนวนการประมวลผลของภาพใน 1 วินาที (fps) ซึ่งจะทดลองโดยการเพิ่มจำนวนคนในการประมวลผลในภาพ

1.5 ระยะเวลาการดำเนินแผนการวิจัย

การดำเนินงาน	**ระยะเวลาการดำเนินงานวิจัย**																
	พ.ศ.๒๕๖๕												**พ.ศ.๒๕๖๖**				
	เดือนที่																
	๑	**๒**	**๓**	**๔**	**๕**	**๖**	**๗**	**๘**	**๙**	**๑๐**	**๑๑**	**๑๒**	**๑**	**๒**	**๓**	**๔**	**๕**
สืบค้นข้อมูลเกี่ยวกับหัวข้อที่สนใจ																	
ดำเนินการสร้างระบบ																	
ทดสอบกับผู้ใช้งานจริง																	
ปรับปรุงและแก้ไขจุดบกพร่อง																	
จัดทำรายงานฉบับสมบูรณ์และเตรียมการนำเสนอ																	
นำเสนองานวิจัย																	

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1.6.1 สามารถเพิ่มความสะดวกสบายและลดเวลาที่ใช้ไปกับการเช็คชื่อ ให้แก่โรงเรียนและนักเรียน

1.6.2 ใช้งานง่ายกว่าระบบเช็คชื่ออื่นๆ ในตลาด

1.6.3 ระบบเช็คชื่ออัตโนมัติที่ใช้งานได้จริง และมีความแม่นยำสูง

บทที่ 2

เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 ภาษา Python

Python เป็นภาษาเขียนโปรแกรมระดับสูงที่ใช้กันอย่างกว้างขวางในการเขียนโปรแกรมสำหรับวัตถุประสงค์ทั่วไป ภาษา Python นั้นสร้างโดย Guido van Rossum และถูกเผยแพร่ครั้งแรกในปี 1991 Python นั้นเป็นภาษาแบบ interpreter ที่ถูกออกแบบโดยมีจุดประสงค์ที่จะทำให้โค้ดอ่านได้ง่ายขึ้น และโครงสร้างของภาษานั้นจะทำให้โปรแกรมเมอร์สามารถเข้าใจแนวคิดการเขียนโค้ดโดยใช้บรรทัดที่น้อยลงกว่าภาษาอย่าง C++ และ Java ซึ่งภาษานั้นถูกกำหนดให้มีโครงสร้างที่ตั้งใจให้การเขียนโค้ดเข้าใจง่ายทั้งในโปรแกรมเล็กไปจนถึงโปรแกรมขนาดใหญ่

ภาพที่ 2.1 *Python trademarks*

ที่มา: https://www.python.org/community/logos

Python นั้นมีคุณสมบัติเป็นภาษาเขียนโปรแกรมแบบไดนามิกและมีระบบการจัดการหน่วยความจำอัตโนมัติและสนับสนุนการเขียนโปรแกรมหลายรูปแบบ ที่ประกอบไปด้วย การเขียนโปรแกรมเชิงวัตถุ imperative การเขียนโปรแกรมแบบฟังก์ชัน และการเขียนโปรแกรมแบบขั้นตอน มันมีไลบรารี่ที่ครอบคลุมการทำงานอย่างหลากหลาย

ตัวแปรในภาษา Python นั้นมีให้ใช้ในหลายระบบปฏิบัติการ ทำให้โค้ดของภาษา Python สามารถรันในระบบต่างๆ ได้อย่างกว้างขวาง CPython นั้นเป็นการพัฒนาในตอนต้นของ Python ซึ่งเป็นโปรแกรมแบบ open source และมีชุมชนสำหรับเป็นต้นแบบในการพัฒนา เนื่องจากมันได้มีการนำไปพัฒนากระจายไปอย่างหลากหลาย variant CPython นั้นจึงถูกจัดการโดยองค์กรไม่แสวงหาผลกำไรอย่าง Python Software Foundation (ผกามาศ, 2561)

2.2 ภาษา JavaScript

JavaScript คือ ภาษาคอมพิวเตอร์สำหรับการเขียนโปรแกรมบนระบบอินเทอร์เน็ต ที่กำลังได้รับความนิยมอย่างสูง JavaScript เป็นภาษาสคริปต์เชิงวัตถุซึ่งในการสร้างและพัฒนาเว็บไซต์ร่วมกับ HTML เพื่อให้เว็บไซต์ของเราดูมีการเคลื่อนไหว สามารถตอบสนองผู้ใช้งานได้มากขึ้น ซึ่งมีวิธีการทำงานในลักษณะ แปลความและดำเนินงานไปทีละคำสั่ง หรือเรียกว่า Interpreted programming language

(ที่มา https://www.mindphp.com/คู่มือ/73-คืออะไร/2187-java-javascript-คืออะไร.html สืบค้นเมื่อวันที่ 23 กันยายน 2566)

ภาพที่ 2.2 *JavaScript-logo*

ที่มา: https://commons.wikimedia.org/wiki/File:JavaScript-logo.png

2.3 HTML

HTML ย่อมาจาก Hyper Text Markup Language คือภาษาคอมพิวเตอร์ที่ใช้ในการแสดงผลของเอกสารบน website หรือที่เราเรียกกันว่าเว็บเพจ ถูกพัฒนาและกำหนดมาตรฐานโดยองค์กร World Wide Web Consortium (W3C) และจากการพัฒนาทางด้าน Software ของ Microsoft ทำให้ภาษา HTML เป็นอีกภาษาหนึ่งที่ใช้เขียนโปรแกรมได้ หรือที่เรียกว่า HTML Application HTML เป็นภาษาประเภท Markup สำหรับการการสร้างเว็บเพจ โดยใช้ภาษา HTML (ที่มา https://www.mindphp.com/คู่มือ/73-คืออะไร/2026-html-คืออะไร.html (สืบค้นเมื่อวันที่ 23 กันยายน 2566)

ภาพที่ 2.3 *HTML5 official logo*

ที่มา: https://en.wikipedia.org/wiki/HTML#/media/File:HTML5_logo_and_wordmark.svg

2.4 ปัญญาประดิษฐ์

2.4.1 ความหมาย

ปัญญาประดิษฐ์ (AI : Artificial Intelligence) คือเครื่องจักร (machine) ที่มีฟังก์ชันทีมีความสามารถในการทำความเข้าใจ เรียนรู้องค์ความรู้ต่างๆ อาทิเช่น การรับรู้ การเรียนรู้ การให้เหตุผลและการแก้ปัญหาต่างๆ

2.4.2 ประเภทของปัญญาประดิษฐ์

2.4.2.1 ปัญญาประดิษฐ์เชิงแคบ

ปัญญาประดิษฐ์เชิงแคบ (Narrow AI) หรือ ปัญญาประดิษฐ์แบบอ่อน (Weak AI) คือ AI ที่มีความสามารถเฉพาะทางได้ดีกว่ามนุษย์คือ ปัญญาประดิษฐ์ที่เก่งในเรื่องเฉพาะทาง

2.4.2.2 ปัญญาประดิษฐ์ทั่วไป

ปัญญาประดิษฐ์ทั่วไป (General AI) คือ ปัญญาประดิษฐ์ที่มีความสามารถระดับเดียวกับมนุษย์ สามารถทำทุกๆ อย่างที่มนุษย์ทำได้และได้ประสิทธิภาพที่ใกล้เคียงกับมนุษย์

2.4.2.3 ปัญญาประดิษฐ์แบบเข้ม

ปัญญาประดิษฐ์แบบเข้ม (Strong AI) คือ ปัญญาประดิษฐ์ที่มีความสามารถเหนือมนุษย์ในหลายๆด้าน

2.4.3 ประวัติ

ในปี 1956 , กลุ่มของผู้เชี่ยวชาญแนวหน้าจากหลายๆวงการได้ร่วมกันทำงานวิจัยเกี่ยวกับ AI มีผู้นำทีมได้แก่ John McCarthy (Dartmouth College), Marvin Minsky (Harvard University), Nathaniel Rochester (IBM) และ Claude Shannon (Bell Telephone Laboratories) โดยมีจุดประสงค์หลักของงานวิจัย คือ การค้นหามุมมองและหลักการต่างๆที่ใช้การเรียนรู้อย่างครอบคลุม เพื่อที่จะนำมาประยุกต์ใช้ให้เครื่องจักรสามารถเรียนรู้ได้เช่นกัน โดยมีเนื้อหาของโครงการมีดังนี้

1. คอมพิวเตอร์อัตโนมัติ
2. จะสามารถเขียนโปรแกรมคอมพิวเตอร์โดยใช้ภาษาคอมพิวเตอร์ได้อย่างไร
3. โครงข่ายประสาทเทียม
4. การพัฒนาด้วยตนเอง

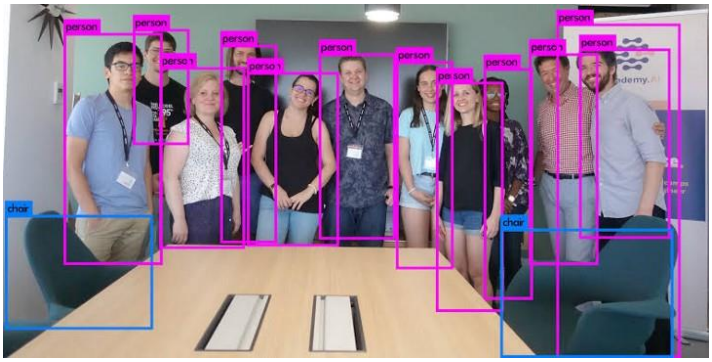
องค์ความรู้เหล่านี้เป็นองค์ความรู้พื้นฐานที่ทำให้คอมพิวเตอร์มีความฉลาดมากขึ้น และยังทำให้ความคิดที่จะการสร้าง AI มีความเป็นไปได้มากยิ่งขึ้น

(ที่มา https://www.thaiprogrammer.org/2018/12/whatisai/ สืบค้นเมื่อวันที่ 23 กุมภาพันธ์ 2564)

2.5 คอมพิวเตอร์วิทัศน์

2.5.1 ความหมาย

คอมพิวเตอร์วิทัศน์ (Computer vision) เป็นสาขาหนึ่งของวิทยาการคอมพิวเตอร์ ที่มีความเกี่ยวข้องกับวิธีที่คอมพิวเตอร์นั้นสามารถทำความเข้าใจชั้นสูง เช่น รูปภาพและวิดีโอ หรือสื่อดิจิทัล และในมุมมองของวิศวกร คอมพิวเตอร์นั้นพยายามที่จะทำงาน หรือเข้าใจการมองเห็นของมนุษย์ ในการทำความเข้าใจของคอมพิวเตอร์นั้นจะเป็นการทำความเข้าใจทางคณิตศาสตร์ โดยจะใช้กระบวนการทางเลขาคณิต และสถิติ (ที่มา https://mindphp.com สืบค้นเมื่อวันที่ 23 กุมภาพันธ์ 2564)



ภาพที่ 2.4 *การตรวจจับคนโดยใช้คอมพิวเตอร์วิทัศน์*

ที่มา: https://ichi.pro/th

2.5.2 เป้าหมาย

1. การตรวจจับ การแบ่งขอบเขต การระบุตำแหน่ง และการจดจำ วัตถุจากภาพ เช่น ใบหน้าของมนุษย์
2. การประเมินผลสำหรับการแบ่งลักษณะของวัตถุในภาพ หรือ การเปรียบเทียบวัตถุ
3. การเปรียบเทียบวัตถุในมุมมองต่าง ๆ ของรูปภาพ หรือวัตถุนั้น ๆ
4. การตรวจจับวัตถุนั้น ๆ จากภาพ
5. การเชื่อมโยงมุมมองต่าง ๆ ของรูปภาพ เพื่อสร้างแบบจำลองสามมิติ ของรูปภพนั้น ๆ โดยแบบจำลองเหล่านั้นอาจจะนำมาใช้ในการสร้างต้นแบบ หุ่นยนต์ AI
6. ในการค้นหาวัตถุเหล่านั้น ด้วยรูปภาพ จำเป็นต้องมีฐานข้อมูลขนาดใหญ่
   (ที่มา https://mindphp.com สืบค้นเมื่อวันที่ 23 กุมภาพันธ์ 2564)

2.5.3 ประวัติ

การทดลองรุ่นแรกเริ่มที่เกี่ยวข้องกับเทคโนโลยีคอมพิวเตอร์วิทัศน์ นั้น เริ่มต้นขึ้นในทศวรรษ 1950 โดยใช้ประโยชน์จากนวัตกรรม neural networks รุ่นแรกสุด ด้วยความพยายามที่จะค้นหาขอบและมุมของวัตถุต่าง ๆ และจัดหมวดหมู่ให้แก่รูปทรงอย่างง่าย เช่น รูปทรงกลม หรือรูปสี่เหลี่ยม เป็นต้น ต่อมาในช่วงทศวรรษ 1970 นั้น ได้มีการนำเทคโนโลยีนี้มาใช้เพื่อการพาณิชย์เป็นครั้งแรกโดยการตีความตัวอักษรที่ถูกเขียนหรือพิมพ์ ด้วยเทคนิคการประมวลผลที่เรียกว่า optical character recognition ซึ่งนำไปสู่การตีความตัวหนังสือและข้อความที่เกิดจากการเขียนหรือสิ่งตีพิมพ์ให้แก่ผู้พิการทางสายตา

การพัฒนาสู่จุดสูงสุดของอินเทอร์เน็ตในช่วงทศวรรษที่ 1990 นั้นส่งผลให้รูปภาพปริมาณมหาศาลถูกนำขึ้นยังระบบออนไลน์และสามารถถูกนำมาทำการวิเคราะห์ได้อย่างไม่หยุดยั้ง ซึ่งเป็นปัจจัยกระตุ้นชั้นดีสำหรับการเติบโตของโปรแกรมการจดจำใบหน้า ข้อมูลปริมาณนับไม่ถ้วนเหล่านี้เติบโตอยู่ตลอดเวลาและช่วยให้อุปกรณ์ต่าง ๆ สามารถทำการระบุตัวตนและจดจำผู้คนต่าง ๆ ได้จากภาพถ่ายและวิดีโอ (ที่มา /analytics/computer-vision.html สืบค้นเมื่อวันที่ 23 กุมภาพันธ์ 2564)

2.5.4 OpenCV

OpenCV คือไลบรารีโอเพ่นซอร์ส (Open Source Library) ที่พัฒนาขึ้นโดยบริษัทอินเทล (Intel) ในปี 1999 นิยมสำหรับการประมวลผลภาพขั้นพื้นฐาน เช่น การเบลอภาพ การผสมภาพ การเพิ่มคุณภาพของภาพ เพิ่มคุณภาพของวิดีโอ การรู้จำวัตถุต่าง ๆ ในภาพ หรือ การตรวจจับใบหน้าหรือวัตถุต่าง ๆ ในภาพและวิดีโอได้

2.5.5 MediaPipe Face Detection

MediaPipe Face Detection is an ultrafast face detection solution that comes with 6 landmarks and multi-face support. It is based on BlazeFace, a lightweight and well-performing face detector tailored for mobile GPU inference. The detector’s super-realtime performance enables it to be applied to any live viewfinder experience that requires an accurate facial region of interest as an input for other task-specific models (ที่มา https://google.github.io/mediapipe/solutions/face_detection สืบค้นเมื่อวันที่ 9 กันยายน 2565)

2.5.6 Centroid based Object Tracking

Centroid-based object tracking utilizes the Euclidean distance between the centroids of the objects detected between two consecutive frames in a video. It then compares the Euclidean distance with a threshold value; if the Euclidean distance is less than the threshold, it is the same object in motion; else, assign a new object Id. If an existing object no longer exists, then the tracking for that object is removed.

(ที่มา https://medium.com/aiguys/a-centroid-based-object-tracking-implementation-455021c2c997 สืบค้นเมื่อวันที่ 9 กันยายน 2565)

2.6 Facial recognition system

2.6.1 definition

facial recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces, typically employed to authenticate users through ID verification services, works by pinpointing and measuring facial features from a given image.

2.6.2 History

Automated facial recognition was pioneered in the 1960s. Woody Bledsoe, Helen Chan Wolf, and Charles Bisson worked on using the computer to recognize human faces. Their early facial recognition project was dubbed "man-machine" because the coordinates of the facial features in a photograph had to be established by a human before they could be used by the computer for recognition. On a graphics tablet a human had to pinpoint the coordinates of facial features such as the pupil centers, the inside and outside corner of eyes, and the widows peak in the hairline. The coordinates were used to calculate 20 distances, including the width of the mouth and of the eyes. A human could process about 40 pictures an hour in this manner and so build a database of the computed distances. A computer would then automatically compare the distances for each photograph, calculate the difference between the distances and return the closed records as a possible match.

2.7 Python Face Recognition

2.7.1 definition

Face Recognition is a python module that Recognizes and manipulates faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.

2.7.2 Features

2.7.2.1 Find faces in pictures

Find all the faces that appear in a picture

Example:

```
import face_recognition
image = face_recognition.load_image_file("your_file.jpg")
face_locations = face_recognition.face_locations(image)
```

2.7.2.2 Identify faces in pictures

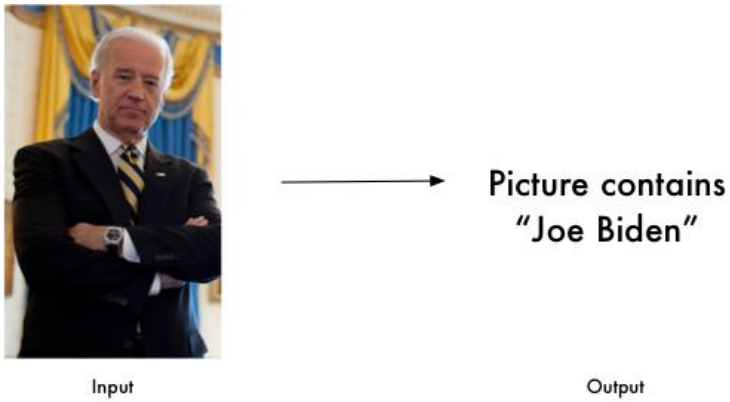
Recognize who appears in each photo.

Example:

```
import face_recognition
known_image = face_recognition.load_image_file("biden.jpg")
unknown_image = face_recognition.load_image_file("unknown.jpg")

biden_encoding = face_recognition.face_encodings(known_image)[0]
unknown_encoding = / face_recognition.face_encodings(unknown_image)[0]

results = face_recognition.compare_faces([biden_encoding],
unknown_encoding)
```



ภาพที่ 2.5 *Recognized face*

ที่มา: https://github.com/ageitgey/face_recognition

(ที่มา https://github.com/ageitgey/face_recognition สืบค้นเมื่อวันที่ 9 กันยายน 2565)

2.8 Numpy

NumPy เป็นชื่อของ library ที่ใช้ในการคำนวนทางคณิตศาสตร์ในภาษา Python ซึ่งภายในถูกเขียนด้วยภาษา C จึงทำงานได้เร็วและมีประสิทธิภาพ โดย NumPy มีความสามารถในการจัดการกับอาเรย์หลายมิติและข้อมูลแบบเมทริกซ์ (https://www.borntodev.com/2020/04/16/พื้นฐานการใช้-numpy-ใน-python-3/, สืบค้นเมื่อวันที่ 15 พฤศจิกายน 2565)

ภาพที่ 2.6 *new NumPy logo*

ที่มา: https://en.wikipedia.org/wiki/NumPy#/media/File:NumPy_logo_2020.svg

2.9 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common

functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. (ที่มา https://en.wikipedia.org/wiki/Flask_%28web_framework%29 สืบค้นเมื่อวันที่ 23 กันยายน 2566)

2.10 Firebase

Firebase เป็นแพลตฟอร์มของบริษัท Google ที่มีเครื่องมือและบริการต่างๆ เพื่อช่วยให้นักพัฒนาสามารถสร้างแอปพลิเคชันแบบ Real-time ได้อย่างง่ายดายและรวดเร็ว บริการหลักของ Firebase ประกอบด้วย Realtime Database, Firestore, Storage, Authentication, Hosting, Functions และ Analytics ซึ่งนักพัฒนาสามารถนำไปใช้งานเพื่อพัฒนาแอปพลิเคชันต่างๆ ได้ง่ายดายและไม่ต้องใช้เวลานานในการจัดการ Server และ Infrastructure ของแอปพลิเคชันด้วยตัวเอง

Firebase เป็นแพลตฟอร์มที่มีความยืดหยุ่นสูง สามารถใช้งานได้กับหลายภาษาโปรแกรมมิ่ง เช่น JavaScript, Swift, Kotlin, Java, Python และอื่นๆ ทำให้นักพัฒนาสามารถพัฒนาแอปพลิเคชันของตนเองได้อย่างสะดวก และ Firebase ยังมี SDK ที่ช่วยให้การใช้งาน Firebase ในแต่ละภาษาเป็นไปอย่างมีประสิทธิภาพ รวดเร็ว และง่ายต่อการเรียนรู้และใช้งาน (Chanokchon, 2566)

2.11 Visual Studio

เป็นโปรแกรมประเภท Editor ใช้ในการแก้ไขโค้ดที่มีขนาดเล็ก แต่มีประสิทธิภาพสูง เป็น Opensource โปรแกรมจึงสามารถนำมาใช้งานได้โดยไม่มีค่าใช้จ่าย เหมาะสำหรับนักพัฒนาโปรแกรมที่ต้องการใช้งานหลายแพลตฟอร์ม รองรับการใช้งานทั้งบน Windows, macOS และ Linux รองรับหลายภาษาทั้ง JavaScript, TypeScript และ Node.js ในตัว และสามารถเชื่อมต่อกับ Git ได้ง่าย สามารถนำมาใช้งานได้ง่ายไม่ซับซ้อน มีเครื่องมือและส่วนขยายต่าง ๆ ให้เลือกใช้มากมาย รองรับการเปิดใช้งานภาษาอื่น ๆ ทั้ง ภาษา C++ , C# , Java , Python , PHP หรือ Go สามารถปรับเปลี่ยน Themes ได้ มีส่วน Debugger และ Commands เป็นต้น (ณัฐพล แสนคำ, 2563)

ภาพที่ 2.7 *Visual Studio Code “stable” icon*

ที่มา: https://code.visualstudio.com/brand

2.12 Raspberry Pi

2.12.1 ความหมาย

Raspberry Pi คือ คอมพิวเตอร์บอร์ดจิ๋วมีประสิทธิภาพและขนาดเท่ากับบัตรหนึ่งใบ ที่ถูกพัฒนาขึ้นมาโดยองค์กร "Raspberry Pi" มูลนิธิเพื่อการกุศลจากประเทศอังกฤษที่มีเป้าหมายต้องการที่จะเผยแพร่ เทคโนโลยีดิจิทัล รวมถึงความรู้ด้านคอมพิวเตอร์ไปสู่ผู้คนทั่วโลก พวกเขามีทั้งกิจกรรมประชาสัมพันธ์ และ การให้เปิดคอร์สให้ข้อมูลด้านเทคโนโลยีกับผู้คน โดยหนึ่งในผลงานนั้นก็คือการสร้างคอมพิวเตอร์ขนาดเล็กอย่าง Raspberry Pi มาวางจำหน่ายสู่สายตาประชาชน ซึ่งมาพร้อมกับศักยภาพในการเป็นคอมพิวเตอร์ขนาดย่อมที่มีราคาถูก หาซื้อได้ง่าย สามารถประยุกต์ใช้ฝึกเขียนโปรแกรมและศึกษาทักษะด้านคอมพิวเตอร์ แม้กระทั่งประยุกต์มาสร้างเกม, ระบบกล้องเว็บแคม, เว็บเซิร์ฟเวอร์ หรืออุปกรณ์ควบคุมฮาร์ดแวร์ และ อุปกรณ์อิเล็กทรอนิกส์ภายในบ้าน หรือ อุปกรณ์ อินเทอร์เน็ตในทุกสิ่ง (Internet of Things - IoTs) เป็นต้น

2.12.2 ส่วนประกอบ

1. ประมวลผลกลาง (CPU)
2. หน่วยความจำ RAM
3. ตัวรับสัญญาณไวไฟ (Wi-Fi Receiver)
4. ตัวรับสัญญาณบลูทูธ (Bluetooth Receiver)
5. HDMI
6. Audio Output
7. USB
8. GPIO (General Purpose Input Output)

2.12.3 ระบบระบบปฏิบัติการ

ระบบปฏิบัติการของ Raspberry Pi มีชื่อว่า Rasbian OS แต่สามารถนำระบบระบบปฏิบัติการอื่นมาใช้ได้ เช่น Ubuntu หรือ Mate เป็นต้น โดยการเลือก ระบบปฏิบัติการ (Operating System) ขึ้นอยู่กับว่าคุณจะเอา Raspberry Pi มาทำอะไร แล้วแต่จุดประสงค์ของผู้ใช้ (https://tips.thaiware.com/1813.html, สืบค้นเมื่อวันที่ 23 กุมภาพันธ์ 2564)

ภาพที่ 2.8 Raspberry Pi4

ที่มา: https://th.element14.com/raspberry-pi/rpi4-modbp-4gb/raspberry-pi-4-model-b-4gb/dp/3051887

2.13 งานวิจัยที่เกี่ยวข้อง

ธนสรรค์และคณะ (2555) ได้ทำระบบบันทึกการปฏิบัติงานออนไลน์ด้วยใบหน้าโดยผลลัพธ์ของการตรวจสอบใบหน้ามีความถูกต้อง 84% จากการทดสอบการบันทึกเวลาปฏิบัติงานของพนักงานจำนวน 10 คน ที่ระยะห่างจากตัวกล้อง 25 cm และ 45 cm

นิติทักษ์และคณะ (2563) ได้ทำเครื่องพิสูจน์ตัวตนตัวด้วยการสแกนใบหน้าโดยสามารถตรวจจับใบหน้าได้ในระยะ1.5เมตร ระบบมีความจำกัดด้านการใช้ทรัพยากรของหน่วยความจำที่ไม่เพียงพอ และระบบไม่สามารถทำงานได้แบบอัตโนมัติเพราะยังเป็นการทำงานแบบรับคำสั่งทีละคำสั่ง

Student Care Co. (2561) ได้ทำบันทึกเวลามาเรียนของนักเรียน ด้วยระบบ AI CAMERA ตรวจจับใบหน้านักเรียนสามารถทำงานได้แม่นยำและรวดเร็ว และสามารถนำมาใช้และจำหน่ายได้จริง ถูกติดตั้งแล้วที่โรงเรียนธัญรัตน์

สุวัฒน์และขนิษฐา (2562) ประสิทธิภาพระบบตรวจสอบการเข้าชั้นเรียนด้วยภาพใบหน้าโดยใช้เทคนิคแอลบีพ โดยค้นพบว่าประสิทธิภาพการตรวจสอบการเข้าชั้นเรียนด้วยภาพใบหน้าที่เหมาะสม คือ 0.30 ซึ่งมีค่าประสิทธิภาพดีที่สุด คือ 0.00 โดยกำหนดค่าประสิทธิภาพดีที่สุด คือ 0.00 หมายถึง เป็นภาพที่ค้นพบในฐานข้อมูลแล้วได้ค่าความเหมือนกับภาพที่นำเข้าไปค้นคืนมากที่สุด

บทที่ 3

วัสดุอุปกรณ์และการออกแบบโครงงาน

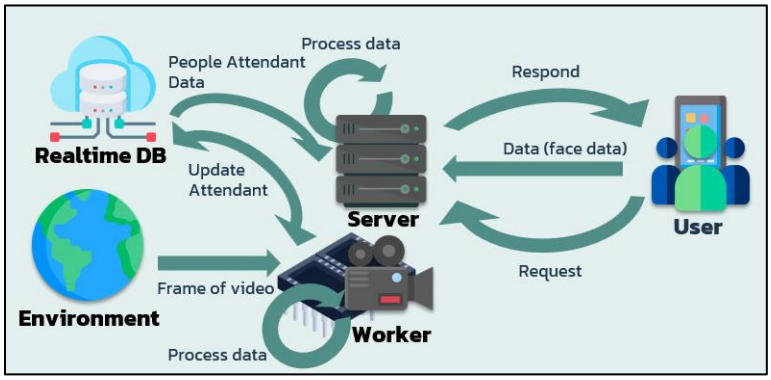
3.1 วัสดุและอุปกรณ์

1. Raspberry Pi 4
2. กล้องสำหรับ Raspberry Pi (Raspberry Pi 8MP Camera Module V2.1)
3. จอแสดงผล

3.2 การออกแบบโครงสร้างของระบบ

ระบบการเช็คชื่อเข้าเรียนอัตโนมัติด้วยการจดจำใบหน้า ประกอบไปด้วย เว็บเซิร์ฟเวอร์ (webserver) คนงาน (worker) และฐานข้อมูลเรียลไทม์ (real-time database) โดยจะทำงานดังนี้

1. คนงาน (worker) ในที่นี้คือ raspberry pi จะเริ่มจากการซิงค์ข้อมูลกับฐานข้อมูลเรียลไทม์ จากนั้นจะทำการถ่ายภาพ และนำภาพที่ถ่ายได้มาประมวลผลเพื่อระบุตัวตนจากใบหน้าที่พบในภาพ และอัปเดตการเช็คชื่อเข้าไปในฐานข้อมูลเรียลไทม์และเริ่มถ่ายภาพอีกครั้ง ทำกระบวนการนี้ซ้ำไปเรื่อยๆ
2. เว็บเซิร์ฟเวอร์ (webserver) จะทำการตอบสนองการเรียกขอของผู้ใช้งานโดยถ้าผู้ใช้งานต้องการเรียกดูข้อมูลเว็บเซิร์ฟเวอร์จะทำการดึงข้อมูลจากฐานข้อมูลเรียลไทม์เพื่อแสดงผลให้ผู้ใช้งาน แต่ถ้าผู้ใช้งานต้องการจะเพิ่มตัวตนใหม่ก็จะนำภาพใบหน้าของผู้ใช้งานมาประมวลผลและอัพโหลดข้อมูลที่ถูกประมวลผลเข้าไปในฐานข้อมูลเรียลไทม์เพื่อให้คนงานสามารถเรียกใช้ได้
3. ฐานข้อมูลเรียลไทม์ (real-time database) ทำหน้าที่เป็นฐานข้อมูลและตัวกลางการแลกเปลี่ยนข้อมูลระหว่างคนงานและเว็บเซิร์ฟเวอร์

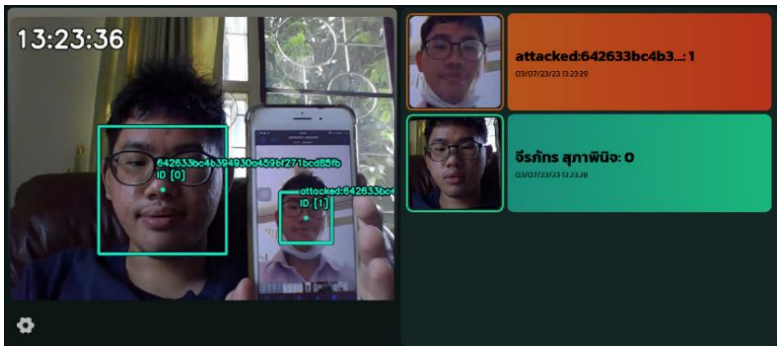


ภาพที่ 3.1 *โครงสร้างของระบบการเช็คชื่อเข้าเรียนอัตโนมัติด้วยการจดจำใบหน้า*

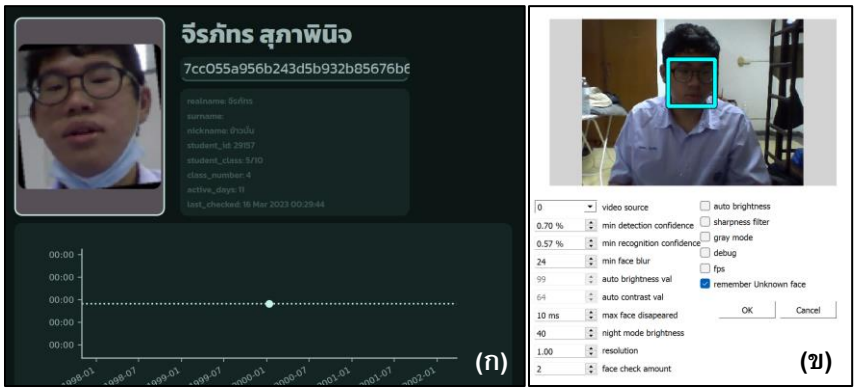
3.2.1 การพัฒนาโปรแกรมสำหรับการจดจำใบหน้า

โปรแกรมนี้ถูกพัฒนาด้วย Python3 โดยใช้ไลบรารี่ PyQt5, Ray, Mediapipe, FaceRecognition, OpenCV และไลบรารี่อื่นๆ โดยโปรแกรมนี้ถูกออกแบบให้ทันสมัยและใช้งานง่ายแต่ยังมีความสามารถในการเข้าถึงและปรับแต่งตัวแปรต่างๆในการจดจำใบหน้าได้โดยการตั้งค่ารวมถึงยังสามารถดูสถิติเบื้องต้นของนักเรียนได้ โดยจะมีความสามารถหลักๆดังนี้

1. สามารถทำงานได้ในเรียลไทม์ อย่างลื่นไหล
2. สามารถจดจำบุคคลที่ไม่เคยอยู่ในฐานข้อมูลได้
3. สามารถแยกแยะระหว่างบุคคลจริงๆและภาพถ่ายได้



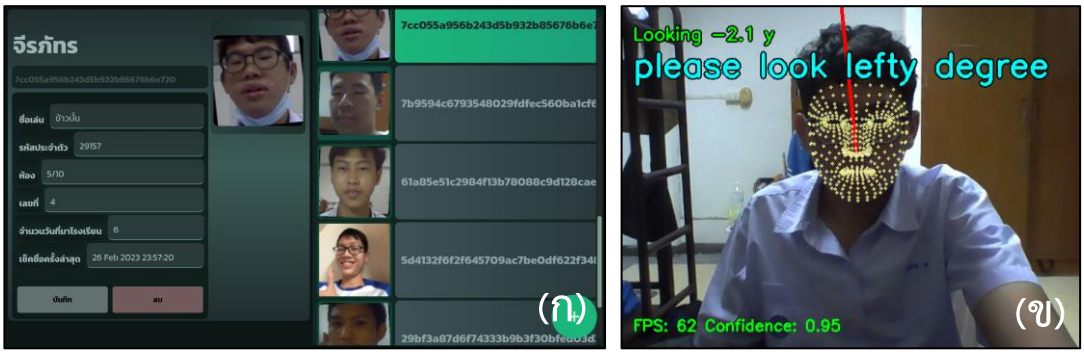
ภาพที่ 3.2 *ระบบการเช็คชื่อเข้าเรียนอัตโนมัติด้วยการจดจำใบหน้า*



ภาพที่ 3.3 *(ก) การดูสถิติเบื้องต้นผ่านโปรแกรมสำหรับการจดจำใบหน้า*
(ข) การตั้งค่าในโปรแกรมสำหรับการจดจำใบหน้า

3.2.2 การพัฒนาโปรแกรมสำหรับการจัดการใบหน้า

โปรแกรมนี้ถูกพัฒนาโดยมีเป้าหมายเพื่ออำนวยความสะดวกในการจัดการรายชื่อและข้อมูลต่างๆ ของนักเรียน

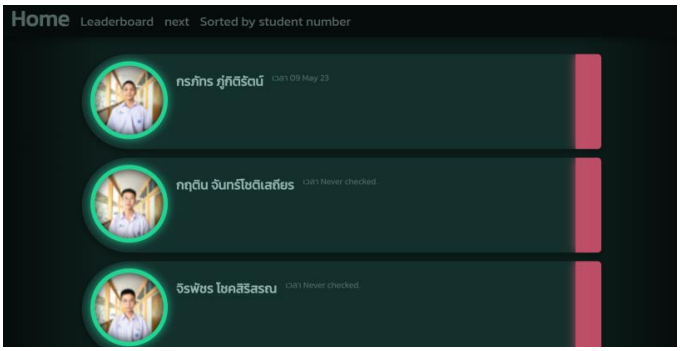


ภาพที่ 3.4 *(ก) โปรแกรมสำหรับการจัดการใบหน้า*
(ข) การเพิ่มใบหน้าโดยใช้โปรแกรมสำหรับการจัดการใบหน้า

3.2.3 การพัฒนาเว็บแอพลิเคชันสำหรับเพื่อเข้าถึงข้อมูลการเช็คชื่อ

แอปพลิเคชันนี้ถูกพัฒนาโดยใช้ Flask web framework และ Python3 โดยมีเป้าหมายเพื่อให้การเข้าถึงข้อมูลและสถิติต่างๆเกี่ยวกับการเช็คชื่อเข้าเรียนเป็นไปได้ง่ายใน ทุกระบบปฏิบัติการ ทุกที่ ทุกเวลาเมื่อมีอินเตอร์เน็ต โดยตัวเว็บแอปพลิเคชันจะมีความสามารถหลักๆดังนี้

1. สามารถจัดเรียงรายชื่อได้โดยวิธีต่างๆ ซึ่งประกอบไปด้วย การจัดเรียงตามชื่อนักเรียน การจัดเรียงตาม รหัสนักเรียน การจัดเรียงตามห้องเรียน การจัดเรียงตามเวลาการเช็คชื่อเข้าเรียน
2. มีการจัดเรียงผลคะแนนโดยใช้สถิติต่างๆเพื่อนำมาใช้ในระบบการจัดอันดับซึ่งจะสรุปผลทุกๆเดือน
3. สามารถดูสถิติการเข้าเรียนและข้อมูลเบื้องต้นของนักเรียนแต่ละคนได้



ภาพที่ 3.5 *เว็บแอปพลิเคชันสำหรับการตรวจสอบและวิเคราะห์สถิติการเช็คชื่อเข้าเรียน*

3.3 การประเมินความสามารถของระบบ

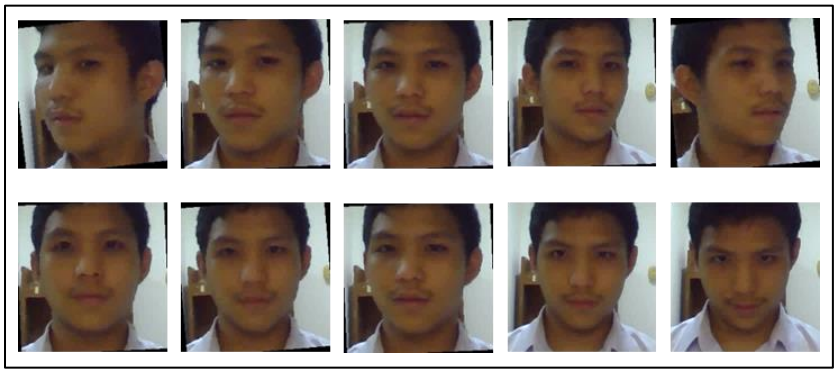
การประเมินความสามารถของระบบสามารถทำได้โดยการทดสอบสองประเภท ได้แก่ การทดสอบความแม่นยำและการทดสอบภาระ ซึ่งเราได้เก็บภาพใบหน้าในหลากหลายมุมโดยใช้โปรแกรมถูกพัฒนาขึ้น เพื่อใช้ในการทดสอบ ซึ่งจะจับภาพใบหน้าที่หมุน องศารอบแกน x (-30 ถึง +30 องศา) และแกน y (-60 ถึง +60 องศา) และภาพเหล่านี้จะถูกบันทึกในฐานข้อมูลและใช้ในขั้นตอนการทดสอบความแม่นยำ

ภาพที่ 3.6 *ขั้นตอนการเก็บภาพใบหน้าในหลากหลายมุมเพื่อใช้ในการทดสอบความแม่นยำของระบบการเช็คชื่อเข้าเรียนอัตโนมัติด้วยการจดจำใบหน้า*

3.3.1 การประเมินความแม่นยำของระบบ

ในการทดสอบความแม่นยำ จะประกอบไปด้วยการทดสอบย่อยสองรายการได้แก่ การประเมินความแม่นยำสำหรับระยะห่างของใบหน้าและ การประเมินความแม่นยำสำหรับการหมุนใบหน้า

ความแม่นยำสำหรับการหมุนใบหน้าแบบหมุนถูกแบ่งออกเป็นสองรายการย่อยได้แก่ การทดสอบความแม่นยำรอบแกน y และรอบแกน x ซึ่งการทดสอบความแม่นยำรอบแกน y ดำเนินการตั้งแต่ -60 ถึง +60 องศา ในขณะที่การทดสอบความแม่นยำรอบแกน x จะดำเนินการตั้งแต่ -30 ถึง +30 องศา

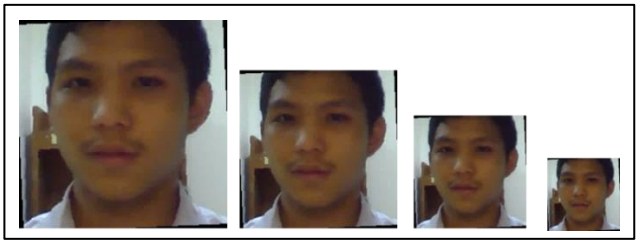


ภาพที่ 3.7 *ตัวอย่างภาพที่ถูกถ่ายเพื่อใช้ในการทดสอบการทดสอบความแม่นยำ*

ความแม่นยำสำหรับการทดสอบระยะห่างของใบหน้าจะดำเนินการโดยการเคลื่อนเข้าและออกจากกล้องในระยะทางและความละเอียดที่กำหนด (ได้แก่ SD และ HD) สมการ (1) ด้านล่างสามารถใช้เพื่อปรับขนาดภาพเพื่อใช้ในการวัดความถูกต้องของการรู้จำที่ระยะตั้งแต่ 0.5 เมตรถึง 10 เมตร ดังแสดงในรูปที่ 8

$$D = \frac{W \times F}{P} \qquad (1)$$

โดยที่ D คือระยะห่างจากกล้อง (เมตร), W คือความกว้างจริงของใบหน้า (เมตร), F คือทางยาวโฟกัสของกล้อง (pixel) และ P คือความกว้างที่รับรู้ได้ของใบหน้า (pixel)



ภาพที่ 3.8 *ตัวอย่างภาพที่ถูกเปลี่ยนขนาดไปตามระยะทางที่กำหนด*

3.3.2 การประเมินความสามารถในการรองรับภาระของระบบ

ในการทดสอบภาะดังแสดงในรูปที่ 9 ประสิทธิภาพของระบบได้รับการทดสอบกับผู้คนจำนวนมากขึ้น ตั้งแต่หนึ่งถึงสิบคน เพื่อพิจารณาว่าระบบดังกล่าวส่งผลต่อความสามารถในการประมวณผลต่อ 1 วินาที (fps) อย่างไร

ภาพที่ 3.9 *ประสิทธิภาพของระบบภายใต้ภาระที่แตกต่างกัน*

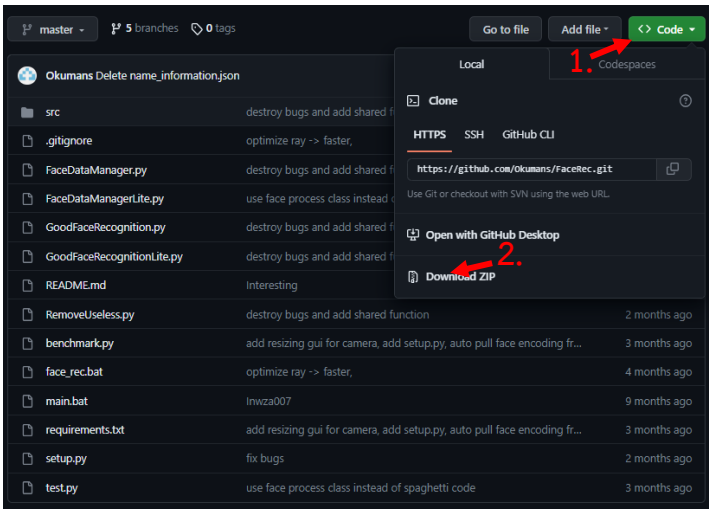
3.4 วิธีการใช้งานโปรแกรม

3.4.1 การดาวน์โหลดโปรเจกต์

การดาวน์โหลดโปรเจกต์สามารถทำได้โดยใช้คำสั่ง “git clone https://github.com/Okumans/FaceRec.git” ในกรณีที่มี git ติดตั้งอยู่ หรือโหลดมาในรูปแบบ .zip จาก https://github.com/Okumans/FaceRec/tree/master

```
C:[redacted]>git clone https://github.com/Okumans/FaceRec.git
Cloning into 'FaceRec'...
remote: Enumerating objects: 390, done.
remote: Counting objects: 100% (102/102), done.
remote: Compressing objects: 100% (83/83), done.
Receiving objects:  92% (359/390), 85.93 MiB | 1.76 MiB/sreused 288 eceiving objects:  91% (355/390), 85.93 MiB | 1.76 MiB/s
Receiving objects: 100% (390/390), 86.19 MiB | 1.91 MiB/s, done.
Resolving deltas: 100% (177/177), done.
```

ภาพที่ 3.10 *ภาพแสดงการดาวน์โหลดโปรเจกต์โดยใช้ git*



ภาพที่ 3.11 *ภาพแสดงการดาวน์โหลดโปรเจกต์ในรูปแบบ .zip*

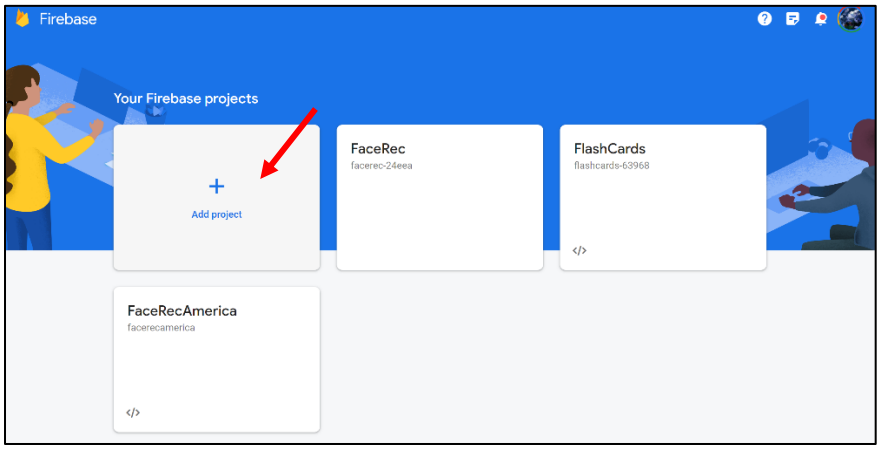
โดยโปรเจกต์มีโครงสร้างดังนี้

```
FaceRec/
├── GoodFaceRecognition.py
├── GoodFaceRecognitionLite.py
├── FaceDataManager.py
├── FaceDataManagerLite.py
├── RemoveUseless.py
├── benchmark.py
├── src/
│   └── resource/
│       └── ...
├── recognition_resource/
│   ├── name_information.json
│   ├── known
│   └── unknown
└── cache/
    └── ...
```

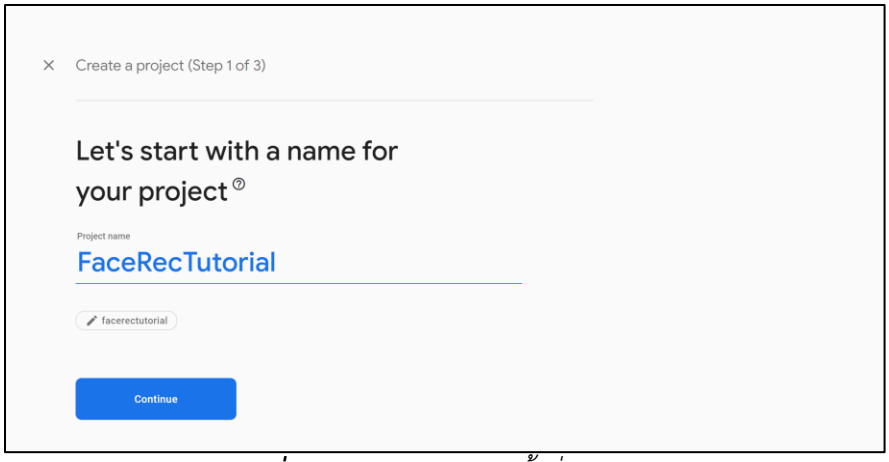
3.4.2 การสร้าง Database

1. สร้างโปรเจกต์ใหม่ใน firebase console

สร้างโปรเจกต์ใหม่ใน firebase console (https://console.firebase.google.com/)



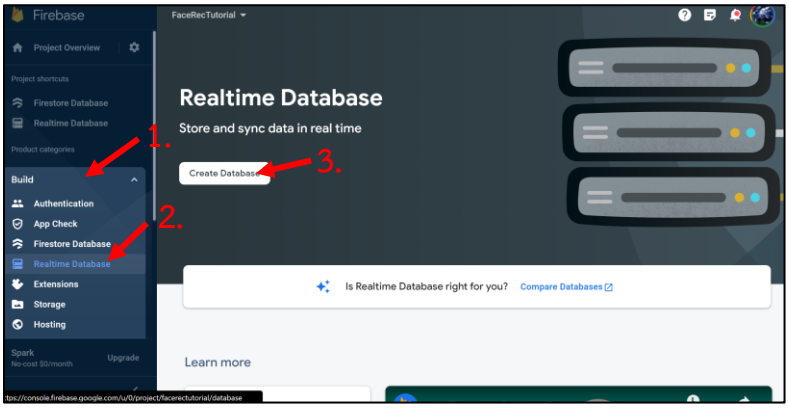
ภาพที่ 3.12 *ภาพแสดงการสร้างโปรเจกต์ใหม่ใน firebase console*



ภาพที่ 3.13 *ภาพแสดงการตั้งชื่อโปรเจกต์*

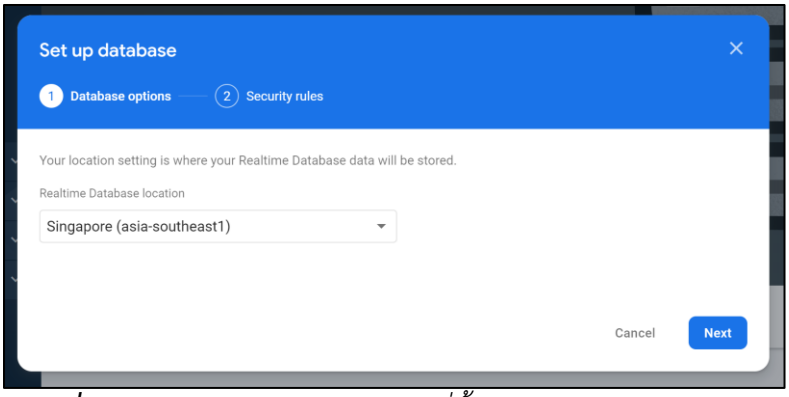
2. สร้าง Realtime Database

ไปที่หน้า Realtime Database โดยผ่าน Build ⟶ Realtime Database และสร้าง Realtime Database ใหม่



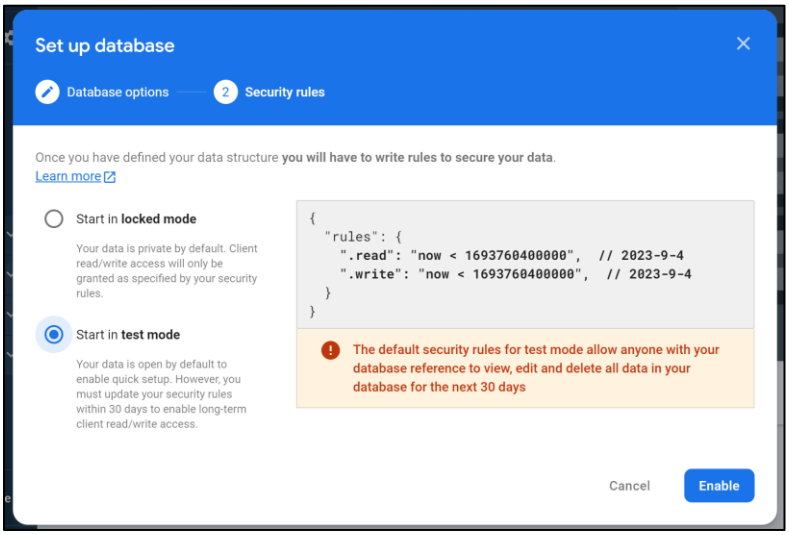
ภาพที่ 3.14 *ภาพแสดงการสร้าง Realtime Database ใหม่ในโปรเจกต์*

หลังจากสร้าง Realtime Database แล้วให้เลือกสถานที่ตั้งของ Realtime Database โดยควรเลือกสถานที่ใกล้กลุ่มเป้าหมายมากที่สุดเนื่องจากระยะห่างมีผลต่อความเร็วเข้าถึงข้อมูล และสถานที่ตั้งของ Realtime Database จะไม่สามารถเปลี่ยนแปลงได้หลังการเลือกครั้งนี้



ภาพที่ 3.15 *ภาพแสดงการเลือกสถานที่ตั้งของ Realtime Database*

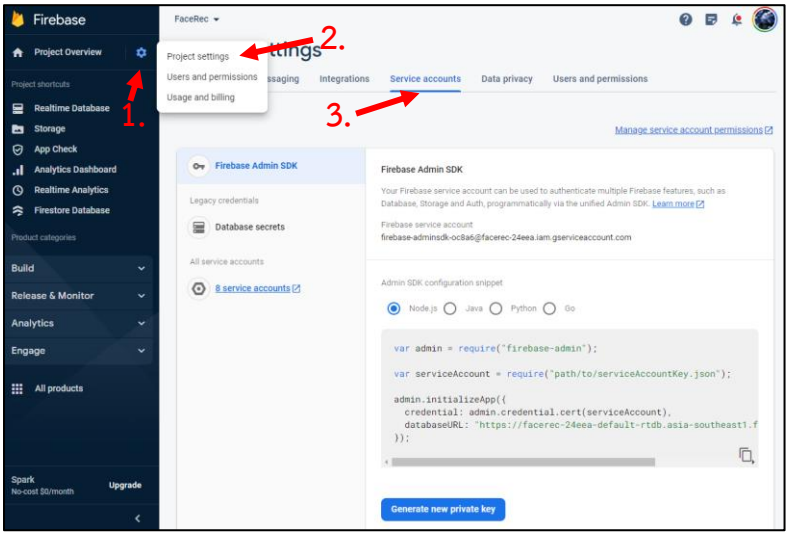
เลือกรูปแบบของ Security rules โดยสามารถปรับแต่งภายหลังได้



ภาพที่ 3.16 *ภาพแสดงการเลือกรูปแบบของ Security rules*

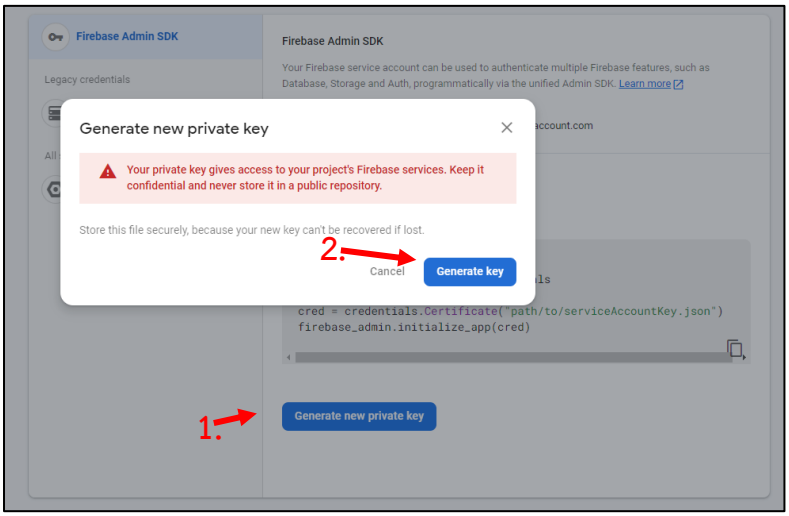
3. การสร้างและดาวน์โหลด Serviceaccountkey

ไปที่หน้า Service accounts ผ่านสัญลักษณ์การตั้งค่าในแถบ Project Overview ⟶ Project settings ⟶ Service accounts



ภาพที่ 3.17 *ภาพแสดงหน้า Service accounts*

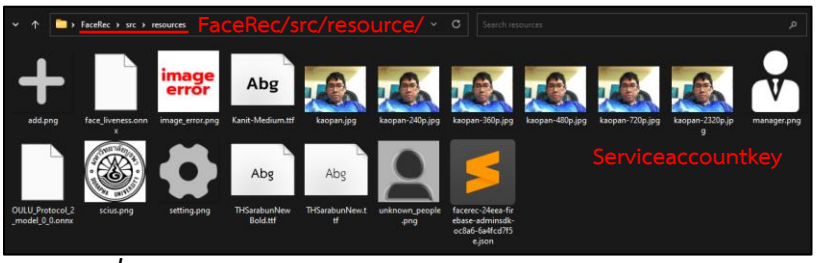
จากนั้นให้การสร้างและดาวน์โหลด Serviceaccountkey



ภาพที่ 3.18 *ภาพการสร้างและดาวน์โหลด Serviceaccountkey*

4. การย้ายตำแหน่ง Serviceaccountkey ให้เป็นไปตามระบบ

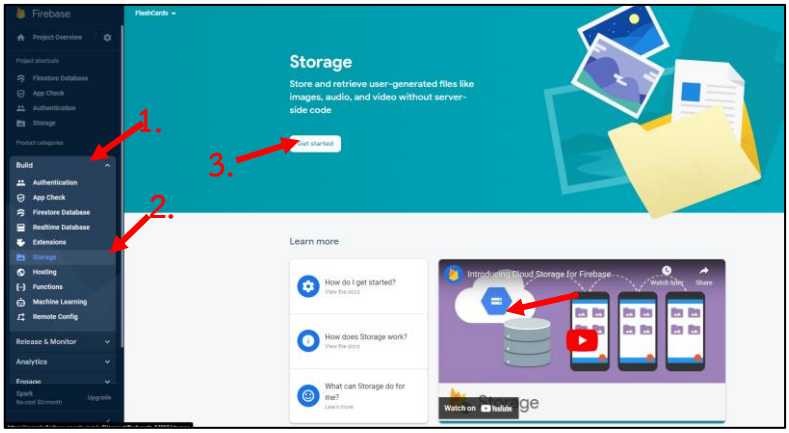
นำไฟล์ Serviceaccountkey ซึ่งจะมีชื่อในรูปแบบดังนี้ “YOURPROJECTNAME-firebase-adminsdk-*****-**********.json” ย้ายไปที่ .../FaceRec/src/resource/



ภาพที่ 3.19 *ภาพการสร้างและดาวน์โหลด Serviceaccountkey*

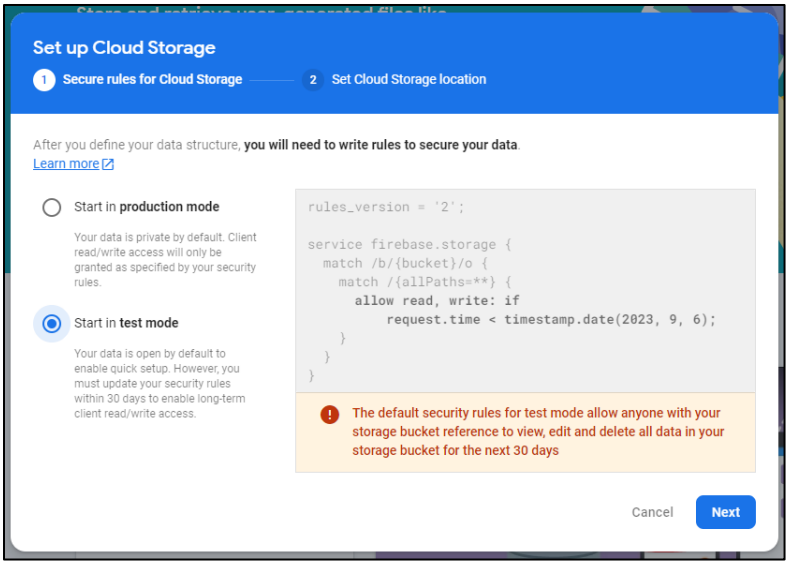
5. สร้าง Cloud Storage

ไปที่หน้า Realtime Database โดยผ่าน Build ⟶ Storage และสร้าง Cloud Storage ใหม่



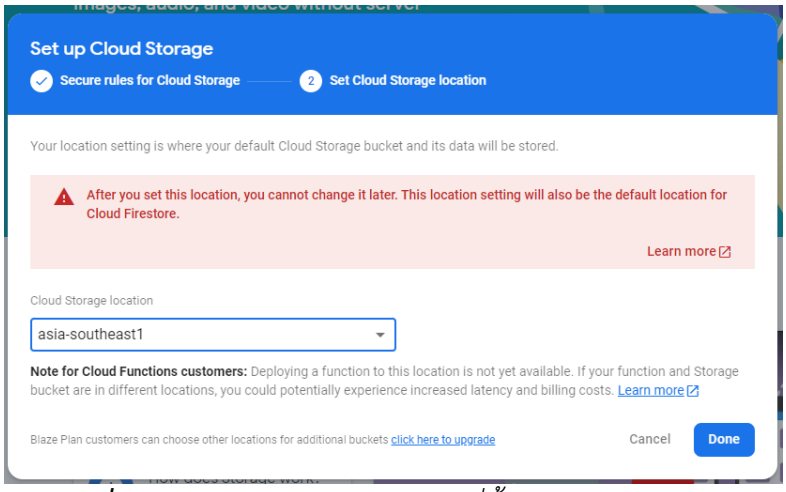
ภาพที่ 3.20 *ภาพแสดงการสร้าง สร้าง Cloud Storage ใหม่ในโปรเจกต์*

หลังจากสร้าง Realtime Database แล้วให้เลือกรูปแบบของ Security rules โดยสามารถปรับแต่งภายหลังได้



ภาพที่ 3.21 *ภาพแสดงการเลือกรูปแบบของ Security rules*

เลือกสถานที่ตั้งของ Cloud Storage เช่นเดียวกับ Realtime Database ควรเลือกสถานที่ใกล้กลุ่มเป้าหมายมากที่สุดเนื่องจากระยะห่างมีผลต่อความเร็วเข้าถึงข้อมูล และสถานที่ตั้งของ Cloud Storage จะไม่สามารถเปลี่ยนแปลงได้หลังการเลือกครั้งนี้



ภาพที่ 3.22 *ภาพแสดงการเลือกสถานที่ตั้งของ Cloud Storage*

3.4.3 การติดตั้ง library ที่จำเป็นและการตั้งค่าเบื้องต้น

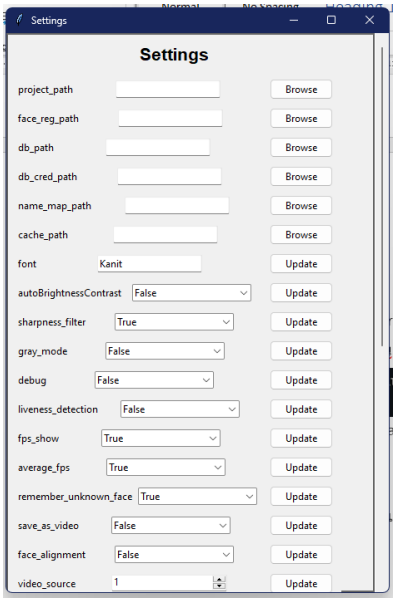
1. การติดตั้ง library ที่จำเป็น

ติดตั้ง library ที่จำเป็นโดยใช้คำสั่ง pip install -r requirements.txt ในตำแหน่งหรือ environment ที่เตรียมไว้ โดยควรใช้ python ในเวอร์ชั่น 3.9.X ถึง 3.11.X

ภาพที่ 3.23 *ภาพแสดงการ*ติดตั้ง library ที่จำเป็นโดยใช้คำสั่ง pip install -r requirements.txt

2. การตั้งค่าเบื้องต้น

เริ่มต้นการตั้งค่าโดยใช้คำสั่ง python setup.py เพื่อเปิดโปรแกรมการตั้งค่า



ภาพที่ 3.24 *ภาพแสดงโปรแกรมการตั้งค่า*

โดยทำการตั้งค่าจำเป็นดังนี้

“*project_path*” : ตำแหน่งเต็มของโปรเจกต์

ค่าเริ่มต้น: .../FaceRec/

“*face_reg_path*” : ตำแหน่งเต็มของโฟลเดอร์เก็บข้อมูลใบหน้า

ค่าเริ่มต้น: .../FaceRec/recognition_resource/

“*db_path*” : ตำแหน่งเต็ม Database สำรองภายในเครื่อง

ค่าเริ่มต้น: .../FaceRec/recognition_resource/

“db_path_cred” : ตำแหน่งเต็มของ *Serviceaccountkey*

ค่าเริ่มต้น: .../FaceRec/src/resources/serviceAccountKey.json

“*name_map_path*” : ตำแหน่งเต็มของ*ไฟล์เพื่อการเชื่อมระหว่างรหัสสำหรับการระบุตัวตนและชื่อเพื่อการแสดง*

ค่าเริ่มต้น: .../FaceRec/recognition_resource/name_information.json

“cache_path” : ตำแหน่งเต็มของโฟลเดอร์สำหรับการเก็บข้อมูลชั่วคราว

ค่าเริ่มต้น: .../FaceRec/cache/

3.4.4 การใช้งานโปรแกรม

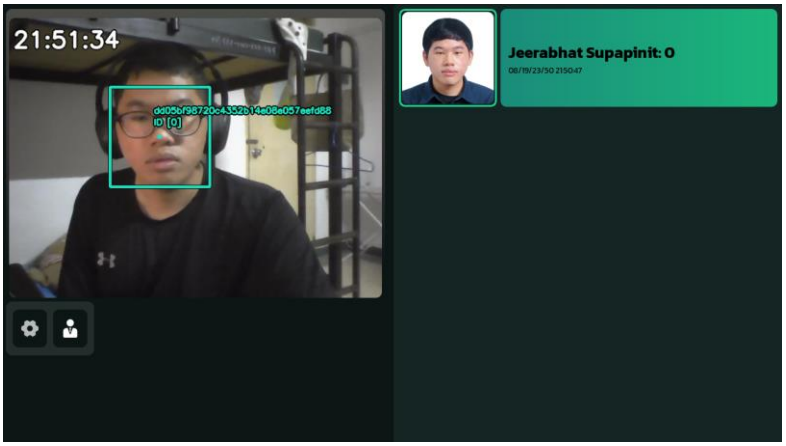
การเริ่มใช้งานโปรแกรมสามารถทำได้โดยการใช้ python ที่มีเวอร์ชั่นตั้งแต่ 3.9 ถึง 3.11 ในการรันไฟล์ที่มีชื่อว่า GoodFaceRecognition.py ดังนี้ในภาพที่ 3.15

ภาพที่ 3.25 *ภาพแสดงการรันไฟล์ GoodFaceRecognition.py ด้วย python*

ภาพที่ 3.26 *ภาพแสดงผลจากการรันไฟล์ GoodFaceRecognition.py ด้วย python*

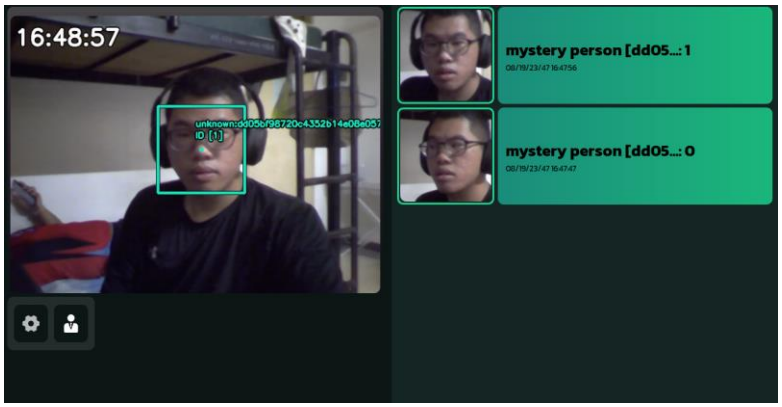
3.4.5 ความสามารถต่างๆโปรแกรม

3.5.3.1 สามารถจดจำและเช็คชื่อได้อย่างแม่นยำ



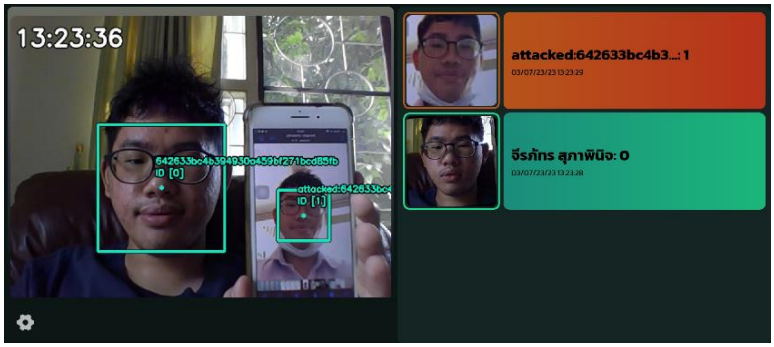
ภาพที่ 3.27 *ภาพแสดงการเช็คชื่อด้วยโปรแกรม*

3.5.3.2 สามารถจดจำใบหน้าที่ไม่ได้ลงทะเบียนอยู่ในระบบได้



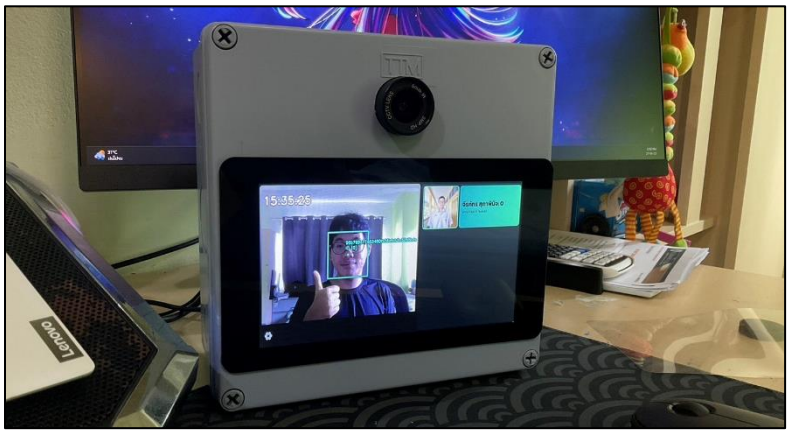
ภาพที่ 3.28 *ภาพแสดงการจดจำใบหน้าที่ไม่ได้ลงทะเบียน*

3.5.3.3 สามารถแยกแยะระหว่างใบหน้าจริงจากภาพใบหน้าที่มีแหล่งที่มาอื่นได้



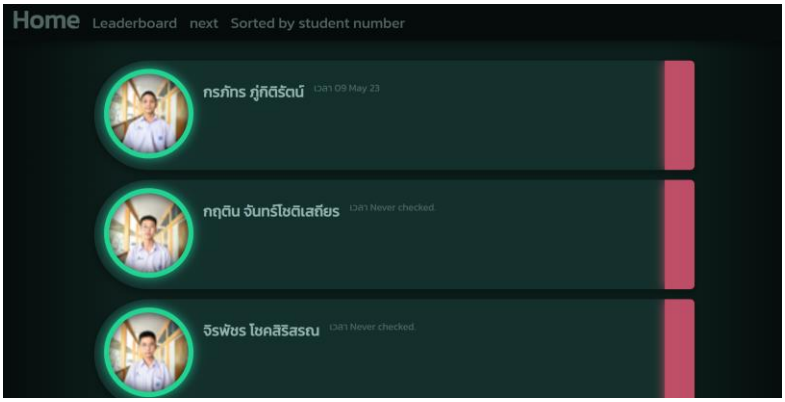
ภาพที่ 3.29 *ภาพแสดงความสามารถแยกแยะระหว่างใบหน้าจริงจากภาพใบหน้าที่มีแหล่งที่มาอื่นได้*

3.5.3.4 สามารถทำงานได้แม้อยู่ในฮาร์ดแวร์ที่มีความสามารถในการคำนวณต่ำ



ภาพที่ 3.30 *ภาพแสดงการทำงานในฮาร์ดแวร์ที่มีความสามารถในการคำนวณต่ำ*

3.5.3.5 อัปเดตข้อมูลไปยังเว็บแอพพลิเคชั่นแบบเรียลไทม์

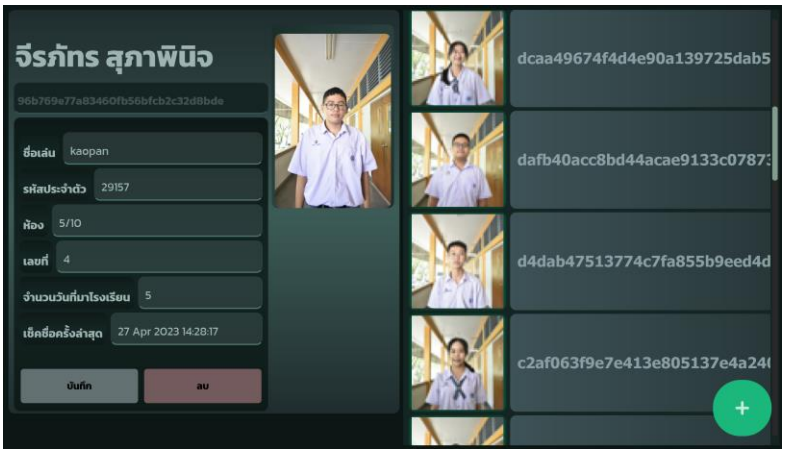


ภาพที่ 3.31 *ภาพแสดงความสามารถในการอัพเดตข้อมูลไปยังเว็บแอพพลิเคชั่นแบบเรียลไทม์*

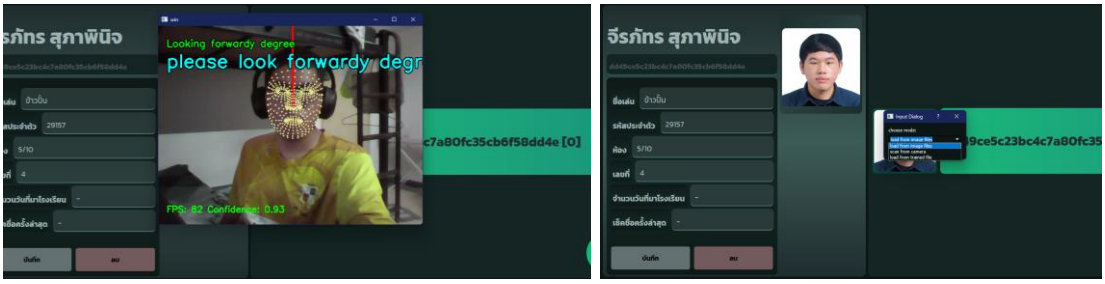
3.4.6 ความสามารถเพิ่มเติมโปรแกรม

3.5.4.1 โปรแกรมการจัดการข้อมูลบุคคล

เป็นโปรแกรมสำหรับการจัดการข้อมูลส่วนบุคคล ซึ่งจะเชื่อมต่อโดยตรงกับ Realtime Database โดยจะช่วยทำให้ผู้ดูแลระบบสามารถจัดการข้อมูลต่างๆได้อย่างสะดวกสบาย



ภาพที่ 3.32 *ภาพแสดงโปรแกรมการจัดการข้อมูลบุคคล*



ภาพที่ 3.33 *ภาพแสดงการจดจำใบหน้าด้วยโปรแกรมการจัดการข้อมูลบุคคล*

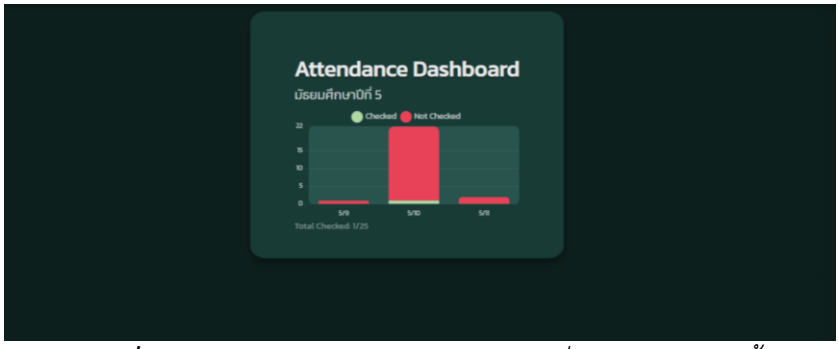
```
(process_image pid=23448) processing image.... [repeated 8x across cluster]
Success 73/80 91.25%
finished..
(process_image pid=23448) processing image.... [repeated 8x across cluster]
Success 76/80 95.0%
finished..
```

ภาพที่ 3.34 *ภาพแสดงอัตราความสำเร็จของการจดจำใบหน้าใหม่*

3.5.4.2 เว็บแอพลิเคชันสำหรับเพื่อเข้าถึงข้อมูลการเช็คชื่อ

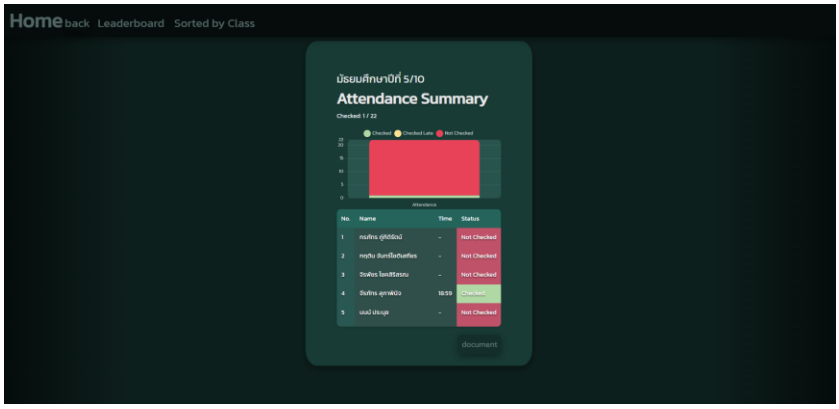
เว็บแอพลิเคชันสำหรับเพื่อเข้าถึงข้อมูลการเช็คชื่อ ซึ่งจะเชื่อมต่อโดยตรงกับ Realtime Database จึงสามารถอัพเดทข้อมูลการเช็คชื่อได้ใน Realtime โดยจะสามารถเข้าถึงข้อมูลได้ในหลายรูปแบบ ดังนี้

1. ระดับชั้น



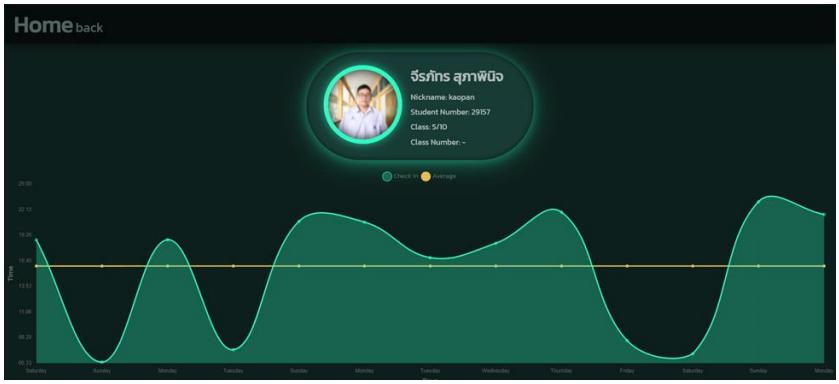
ภาพที่ 3.35 *ภาพการแสดงผลข้อมูลการเช็คชื่อในแต่ละระดับชั้น*

2. ห้องเรียน



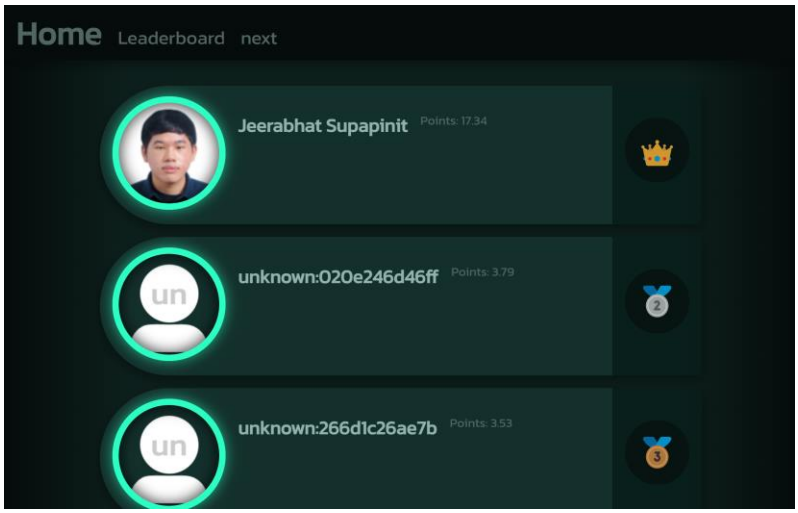
ภาพที่ 3.36 *ภาพการแสดงผลข้อมูลการเช็คชื่อในแต่ละห้องเรียน*

3. บุคคล



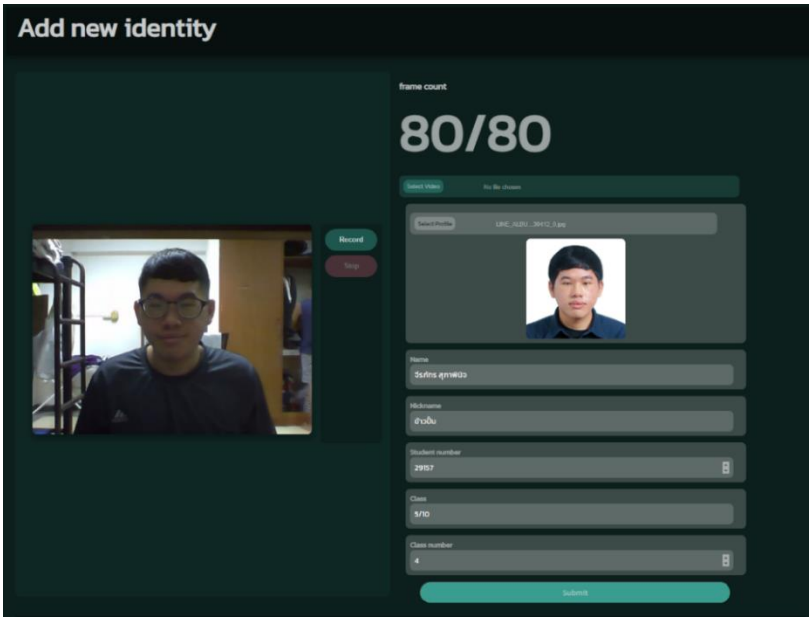
ภาพที่ 3.37 *ภาพการแสดงผลข้อมูลการเช็คชื่อรายบุคคล*

4. การจัดลำดับ



ภาพที่ 3.38 *ภาพการแสดงการจัดลำดับการเข้าเช็คชื่อโดยเรียงตามคะแนนการเช็คชื่อ*

และสามารถสร้างตัวตนใหม่ได้ โดยการใช้กล้อง webcam ในการเก็บภาพใบหน้า การอัพโหลดภาพใบหน้า หรือการอัพโหลดวิดีโอของใบหน้าที่มีความยาวประมาณ 3-5 วินาทีและกรอกข้อมูลส่วนตัวเล็กน้อย ซึ่งประกอบไปด้วย ชื่อจริง – นามสกุล ชื่อเล่น หมายเลขประจำตัว (ถ้ามี) ห้องเรียน/สาขา/คณะ/หน่วย/แผนก และหมายเลขประจำตัวประจำ ห้องเรียน/สาขา/คณะ/หน่วย/แผนก ที่ประจำอยู่ (ถ้ามี)

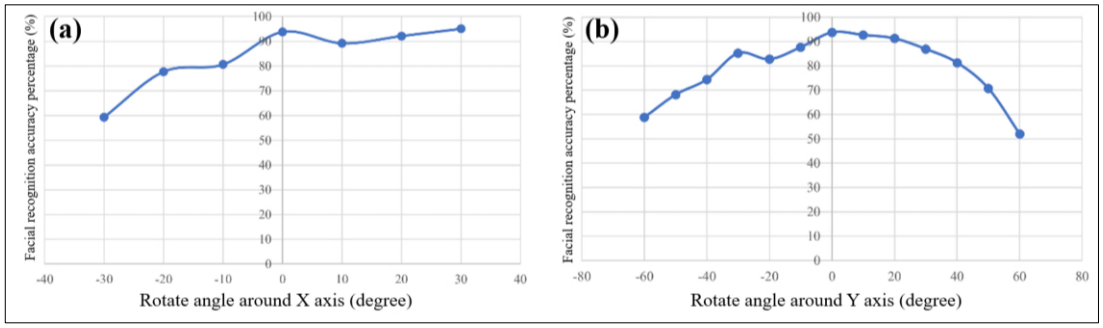


ภาพที่ 3.39 *ภาพการแสดงการสร้างตัวตนใหม่โดยใช้เว็บแอพลิเตชั่น*

บทที่ 4

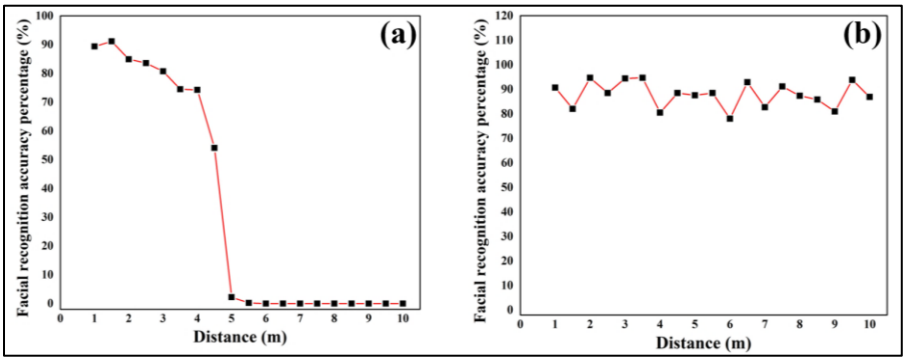
ผลการดำเนินงาน

4.1 ผลการประเมินความแม่นยำของระบบ



กราฟที่ 4.1 (a) ความแม่นยำของการจดจำใบหน้ารอบแกน x
(b) ความแม่นยำของการจดจำใบหน้ารอบแกน y

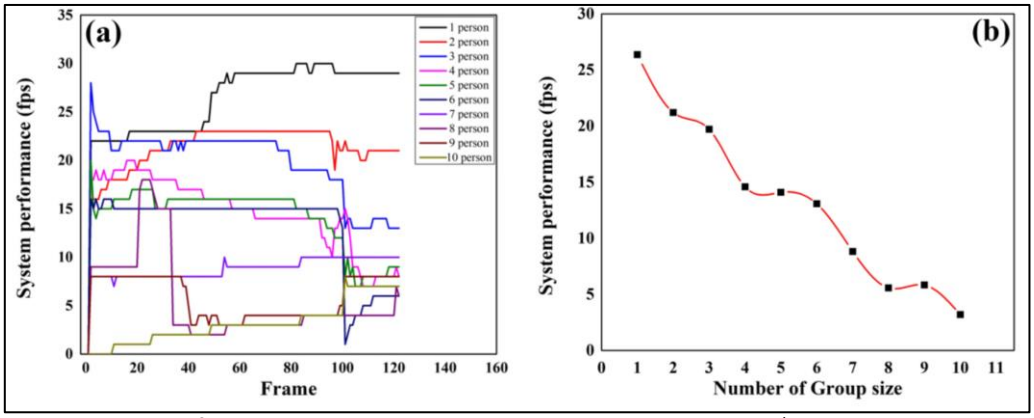
จากกราฟที่ 1 (a) ความแม่นยำของการจดจำใบหน้ารอบแกน x ตั้งแต่ -30 องศาถึง 30 องศาจะมีค่าสูงสุด ณ 0 องศา โดยมีความแม่นยำที่ 93% และจะมีค่าลดลงเล็กน้อยหากอยู่ห่างจาก 0 องศา กราฟที่ 1 (b) เช่นเดียวกับ (a) ความแม่นยำของการจดจำใบหน้ารอบแกน y ตั้งแต่ -60 องศาถึง 60 องศา จะมีค่าสูงสุด ณ 0 องศา โดยมีความแม่นยำที่ 93% และค่อยๆ ลดลงหากองศาอยู่ห่างจาก 0 องศา



กราฟที่ 4.2 (a) ความแม่นยำในการจดจำใบหน้า ณ ระยะทางต่างๆในความละเอียด SD
(b) ความแม่นยำในการจดจำใบหน้า ณ ระยะทางต่างๆในความละเอียด HD

กราฟที่ 2 (a) แสดงให้เห็นว่าความแม่นยำจะค่อนข้างคงที่ในระยะ 0.5 เมตรถึง 4 เมตร แต่จะลดลงอย่างมากเมื่อระยะทางเกิน 4 เมตรและถึงจุดต่ำสุดที่ 5.5 เมตร กราฟที่ 2 (b) แสดงให้เห็นว่าความแม่นยำไม่เคยลดลงต่ำกว่า 80% แต่กลับผันผวนอยู่ระหว่าง 90% ถึง 80%

4.2 ผลการประเมินความสามารถในการรองรับภาระของระบบ



กราฟที่ 4.3 (a) การทำงานของระบบภายใต้ภาระและเวลาที่แตกต่างกัน
(b) การทำงานของระบบภายใต้ภาระหน้าที่แตกต่างกัน

กราฟที่ 3 (b) แสดงเฟรมเฉลี่ยต่อวินาที (fps) ในจำนวนคนทั้งหมดที่ตรวจพบโดยระบบจดจำใบหน้า พบว่ามี fps สูงสุดในกลุ่มหนึ่งคน (26 fps) ตามด้วยกลุ่มสองคน (22 fps) กลุ่มสามคน (19 fps) กลุ่มสี่คน (14 fps) และลดลงเหลือต่ำสุดที่กลุ่มสิบคน (3 fps)

นอกจากนี้ (a) ยังแสดงให้เห็นว่าประสิทธิภาพของระบบจดจำใบหน้านั้นไม่คงที่ และอาจแตกต่างกันไปตามปัจจัยต่างๆ เช่น เวลาที่ใช้ในการตรวจจับใบหน้า คุณภาพของใบหน้า และจำนวนการตรวจจับใบหน้าสำหรับการระบุตัวตนของคนคนเดียว ตัวอย่างเช่น จาก (a) ความสามารถในการประมวลผลของระบบไม่สามารถใช้งานได้หรือมีความสามารถในการประมวลผลที่ต่ำมากเมื่อมีคนสิบคนอยู่ในสิบเฟรมแรก แม้ว่า fps เฉลี่ยจะแสดงให้เห็นว่าระบบสามารถประมวลผลได้พอสมควร ดังนั้นสิ่งสำคัญคือต้องพิจารณาตัวแปรต่างๆที่มีผลต่อประสิทธิภาพของระบบเพื่อทำความเข้าใจข้อจำกัดและความเอนเอียงที่อาจเกิดขึ้น

บทที่ 5
อภิปราย สรุปผลการทดลอง และข้อเสนอแนะ

5.1 อภิปรายผลการทดลอง

งานวิจัยนี้มุ่งพัฒนาระบบการเช็คชื่อเข้าเรียนอัตโนมัติด้วยการจดจำใบหน้าซึ่งประกอบไปด้วยโปรแกรมสำหรับการจดจำใบหน้า โปรแกรมสำหรับการจัดการใบหน้าและเว็บแอปพลิเคชันสำหรับการตรวจสอบและวิเคราะห์สถิติการเช็คชื่อเข้าเรียน โดยสามารถทดสอบความสามารถของระบบการเช็คชื่อเข้าเรียนอัตโนมัติได้ด้วยการทดสอบความแม่นยำ และความสามารถในการรับภาระของระบบ

จากการดำเนินงานสามารถพัฒนาระบบการเช็คชื่อเข้าเรียนอัตโนมัติด้วยการจดจำใบหน้าที่มีความแม่นยำสูง มีประสิทธิภาพ ใช้งานง่ายและมีความสามารถอื่นๆอีกมากมาย ซึ่งสามารถเพิ่มความสะดวกสะบายและประสิทธิถาพให้แก่การเช็คชื่อเข้าเรียน

5.2 สรุปผลการทดลอง

จากการทดสอบพบว่าความแม่นยำที่ดีที่สุดสำหรับการจดจำใบหน้าเกิดขึ้นที่การหมุน 0 องศารอบแกน y และ 0 องศารอบแกน x สำหรับการทดสอบความแม่นยำในระยะทางต่างๆ ระบบมีความแม่นยำสูงสุดเมื่อมีระยะต่ำกว่า 4 เมตรในความละเอียด SD และ 16 เมตรในความละเอียด HD นอกจากนี้ การทดสอบภาระยังแสดงให้เห็นว่าระบบสามารถรองรับคนได้ถึง 9 คนด้วยเฟรมต่อวินาทีที่เหมาะสม

5.3 ข้อเสนอแนะ

1. ควรพัฒนาระบบสำหรับการเข้าสู่ระบบและเสริมสร้างความปลอดภัย
2. ควรหาโมเดลที่มีความแม่นยำและมีประสิทธิภาพมากกว่านี้
3. ควรเพิ่มความสามารถในการจดจำใบหน้าแม้มีหน้ากากอนามัย
4. ควรเพิ่มประสิทธิภาพในการรับส่งข้อมูลระหว่างคอร์

เอกสารอ้างอิง

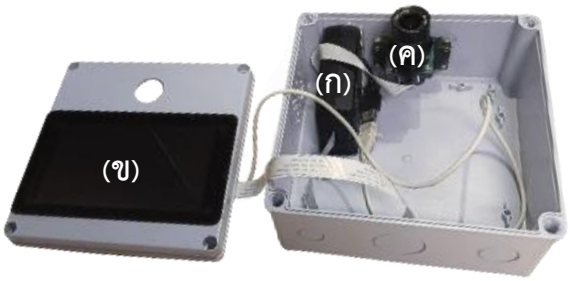
[1] Yang H, Han X. Face Attendance System Based on Real-Time Video Processing. IEEE Access. 2020; 8:159143–159150).

[2] Guo Q, Wang Z, Fan D. Multi-face Recognition. In: Proceedings of the 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI); 2020 Nov 13-15; Chengdu, China. p. 281-286. doi: 10.1109/CISP-BMEI51763.2020.9263565).

[3] Nimshi K. Deep Learning on Raspberry Pi3 for Face Recognition. Future Generation Computer Systems. (Master's Thesis, Engineering in Microelectronics and Embedded Systems, Asian Institute of Technology School of Engineering and Technology Thailand, 2019), p. 1-3

ภาคผนวก

1. โครงสร้างของ Worker



(ก) Raspberry Pi 4

(ข) กล้องสำหรับ Raspberry Pi (Raspberry Pi 8MP Camera Module V2.1)

(ค) จอแสดงผล

2. โค้ดในโปรแกรมหลัก

สามารถดู source code ได้ใน https://github.com/Okumans/FaceRec และ https://github.com/Okumans/FaceRecWeb

2.1 โค้ดโปรแกรมสำหรับการจัดการใบหน้า (GoodFaceRecognition.py)

```
import src.triple_gems
import json
import logging
import os
import os.path as path
import sys
import time
from copy import deepcopy
from datetime import datetime
from json import dumps, loads
from threading import Thread
from typing import *
from mss import mss
from pyautogui import size
import cv2
import mediapipe as mp
import numpy as np

import ray
from ray.exceptions import GetTimeoutError

from PyQt5 import QtGui
from PyQt5.QtCore import (
    pyqtSignal,
```

```
    pyqtSlot,
    Qt,
    QThread,
    QSize,
    QRect,
    QCoreApplication,
    QVariantAnimation,
    pyqtBoundSignal,
)
from PyQt5.QtGui import QFont, QIcon
from PyQt5.QtWidgets import (
    QWidget,
    QApplication,
    QLabel,
    QVBoxLayout,
    QHBoxLayout,
    QSizePolicy,
    QScrollArea,
    QSpacerItem,
    QMainWindow,
    QBoxLayout,
    QSplitter
)

from src import general
from src import ui_popup  # for setting popup
from src import ui_popup2  # for infobox popup
from src.DataBase import DataBase
from src.FaceAlignment import face_alignment
from src.ShadowRemoval import remove_shadow_grey
from src.attendant_graph import AttendantGraph, Arrange
from src.general import Color, get_from_percent, RepeatedTimer
from src.init_name import name_information_init, init_shared
from src.recognition import Recognition
from src.scrollbar_style import scrollbar_style
from src.studentSorter import Student
from src.centroidtracker import CentroidTracker


# -----------------setting-----------------
try:
    with open("settings.json", "r") as f:
        setting = json.load(f)
except FileNotFoundError:
    print("setting not found!, please run setup.py first.")
    quit()

use_folder = [setting['project_path'], setting["face_reg_path"],
setting["face_reg_path"] + r"/unknown",
              setting["face_reg_path"] + r"/known", setting['project_path']
+ r"/cache"]

RAY_TIMEOUT = 0.008

if __name__ == "__main__":
    print(f"project path set to \"{setting['project_path']}\"")
    # logging.basicConfig(level=logging.INFO)

    # check if using folder is available
    for folder_path in use_folder:
        if not path.exists(folder_path):
            os.mkdir(folder_path)
```

```
    init_shared(setting["face_reg_path"], setting["cache_path"],
certificate_path=setting["db_cred_path"])
    name_information_init(setting["face_reg_path"],
setting["name_map_path"], certificate_path=setting["db_cred_path"])

    # remove_expire_unknown_faces(setting["face_reg_path"])
    # ContaminationScanner(setting["face_reg_path"], .65).scan()

    mp_face_detection = mp.solutions.face_detection
    mp_face_mesh = mp.solutions.face_mesh
    # emotion_recognizer = FER()
    ct = CentroidTracker(
        faceRecPath=setting["face_reg_path"],
        maxDisappeared=setting["face_max_disappeared"],
        minFaceBlur=setting["min_faceBlur_detection"],
        minFaceConfidence=setting["min_detection_confidence"],
        minFaceRecConfidence=setting["min_recognition_confidence"],
        faceCheckAmount=setting["face_check_amount"],
        remember_unknown_face=setting["remember_unknown_face"],
        otherSetting=setting,
    )
    ct.recognizer.face_detection_method = "mp"
    (H, W) = (None, None)
    text_color = (31, 222, 187)
    prev_frame_time = 0  # fps counter
    new_frame_time = 0
    last_id = -1
    now_frame = 0  # cv2 mat
    already_check = {}
    kernel = np.array([[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]])
    image_error = cv2.imread(setting["project_path"] +
"/src/resources/image_error.png")
    unknown_image = cv2.imread(setting["project_path"] +
"/src/resources/unknown_people.png")


class VideoThread(QThread):
    change_pixmap_signal: pyqtBoundSignal = pyqtSignal(np.ndarray)
    change_infobox_message_signal: pyqtBoundSignal = pyqtSignal(dict)

    def __init__(self):
        global H, W
        super().__init__()

        (H, W) = (None, None)
        self.video_type: str = setting["video_source"]
        self.db: DataBase = DataBase("Students",
                                     sync_with_offline_db=True)
        self.db.offline_db_folder_path = setting["face_reg_path"]
        self.run: bool = True
        self.frame_index: int = 0
        self.objname_map: Dict[int, any] = {}
        self.avg_fps: general.Average = general.Average([0],
calculate_amount=100)

        if self.video_type == "screen":
            self.sct = mss()
            self.screen_size = size()
        else:
            self.cap = cv2.VideoCapture(self.video_type)
            self.cap.set(cv2.CAP_PROP_FPS, 30)
            self.cap.set(cv2.CAP_PROP_FOURCC, 0x32595559)
```

```
        # write output as a video
        if setting["save_as_video"]:
            self.out = cv2.VideoWriter(f'{int(time.time())}.avi',
cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'),
                                       10, (640, 480))

    def release_cam(self):
        if self.video_type != "screen":
            self.cap.release()
        self.run = False

    def run(self):
        global H, W, prev_frame_time, new_frame_time, last_id, now_frame
        with
mp_face_detection.FaceDetection(min_detection_confidence=setting["min_detect
ion_confidence"],
                                        model_selection=1) as
face_detection:
            while self.run or self.cap.isOpened():
                if self.video_type == "screen":
                    monitor = {
                        "top": 40,
                        "left": 0,
                        "width": self.screen_size[0],
                        "height": self.screen_size[1],
                    }
                    image = np.array(self.sct.grab(monitor))
                    success = True
                else:
                    success, image = self.cap.read()

                new_frame_time = time.time()

                if not success or image is None:
                    logging.info(f"Ignoring empty camera frame.")
                    continue

                # rotate image counter clock wise setting["rotate_frame"]
times
                for i in range(setting["rotate_frame"]):
                    image = cv2.rotate(image,
cv2.ROTATE_90_COUNTERCLOCKWISE)

                # update variables in centroid tracker class
                ct.maxDisappeared = setting["face_max_disappeared"]
                ct.minFaceBlur = setting["min_faceBlur_detection"]
                ct.minFaceConfidence = setting["min_detection_confidence"]
                ct.faceCheckAmount = setting["face_check_amount"]
                ct.recognizer.min_confidence =
setting["min_recognition_confidence"]
                ct.recognizer.remember = setting["remember_unknown_face"]

                # check if the input image is valid
                if H is None or W is None:
                    (H, W) = image.shape[:2]
                    setting["base_resolution"] = (W, H)

                now_frame = image
                image.flags.writeable = False
                image = cv2.flip(cv2.cvtColor(image, cv2.COLOR_BGR2RGB), 1)

                # change the brightness, contrast, sharpness, gray_mode as
it assign to the settings
```

```
                if setting["autoBrightnessContrast"]:
                    image = general.change_brightness_to(image,
setting["autoBrightnessValue"])
                    image = general.change_contrast_to(image,
setting["autoContrastValue"])

                if setting["sharpness_filter"]:
                    image = cv2.filter2D(src=image, ddepth=-1,
kernel=kernel)

                # for night vision or bad lighting condition
                if setting["gray_mode"]:
                    image = cv2.cvtColor(remove_shadow_grey(image),
cv2.COLOR_GRAY2RGB)
                    general.putBorderText(
                        image,
                        "NIGHT MODE",
                        (W - 100, 20),
                        cv2.FONT_HERSHEY_SIMPLEX,
                        0.5,
                        Color.Violet,
                        Color.Black,
                        2,
                        3
                    )

                # st = time.time()
                results: NamedTuple = face_detection.process(image)
                # print(time.time()-st, 1/((time.time()-st) if time.time()-
st != 0 else -1), "process..")

                image.flags.writeable = True
                image_use = deepcopy(image)
                image = cv2.resize(image, (W, H))
                rects = []

                # st = time.time()
                if results.detections:
                    for detection in results.detections:
                        x_min =
detection.location_data.relative_bounding_box.xmin * W *
setting["resolution"]
                        y_min =
detection.location_data.relative_bounding_box.ymin * H *
setting["resolution"]
                        x_max = x_min +
detection.location_data.relative_bounding_box.width * W *
setting["resolution"]
                        y_max = y_min +
detection.location_data.relative_bounding_box.height * H *
setting["resolution"]
                        face_height = y_max - y_min
                        box = (x_min, y_min, x_max, y_max)

                        face_image = deepcopy(
                            image_use[
                                int((int(box[1]) -
get_from_percent(face_height, 20)) / setting["resolution"]):
                                int((int(box[3]) +
get_from_percent(face_height, 20)) / setting["resolution"]),
                                int((int(box[0]) -
get_from_percent(face_height, 20)) / setting["resolution"]):
```

```
                                int((int(box[2]) +
get_from_percent(face_height, 20)) / setting["resolution"]),
                            ]
                        )
                        (int(box[0] / setting["resolution"]), int(box[1] /
setting["resolution"])),
                        (int(box[2] / setting["resolution"]), int(box[3] /
setting["resolution"])),
                        # align face if face_alignment is on
                        if setting["face_alignment"]:
                            try:
                                face_image = face_alignment(face_image,
detection)
                            except TypeError:
                                pass

                        distance: float = ((H * setting["resolution"] * W *
setting["resolution"]) / 518400) * (
                                ((face_image.shape[1] + face_image.shape[0])
/ 2) / 220.39
                        )
                        distance = distance ** (1 / -0.949) if distance != 0
else 1000000000000000
                        # check if face data is enough for face recognizing
                        if face_height >= 60:
                            rects.append({box: (detection.score[0],
face_image)})

                        # display debug message in image if debug is on
                        if setting["debug"]:
                            general.putBorderText(
                                image,
                                f"confident: {round(detection.score[0], 2)}%
"
                                f"blur {CentroidTracker.is_blur(face_image,
setting['min_faceBlur_detection'])} ",
                                (int(box[0]), int(box[1]) + 18),
                                cv2.FONT_HERSHEY_SIMPLEX,
                                0.5,
                                (255, 0, 0),
                                (0, 0, 0),
                                2,
                                3,
                            )
                            general.putBorderText(
                                image,
                                f"brightness:
{round(general.brightness(face_image), 2)} "
                                f"contrast:
{round(general.contrast(face_image), 2)}",
                                (int(box[0]), int(box[1]) + 38),
                                cv2.FONT_HERSHEY_SIMPLEX,
                                0.5,
                                (255, 0, 0),
                                (0, 0, 0),
                                2,
                                3,
                            )

                            general.putBorderText(
                                image,
                                f"size(WxH): {face_image.shape[1]},
{face_image.shape[0]} distance-predict: {distance}",
```

```
                                (int(box[0]), int(box[1]) - 8),
                                cv2.FONT_HERSHEY_SIMPLEX,
                                0.5,
                                (255, 0, 0),
                                (0, 0, 0),
                                2,
                                3,
                            )

                            general.putBorderText(
                                image,
                                f"Not supported",
                                (int(box[0]), int(box[1]) - 48),
                                cv2.FONT_HERSHEY_SIMPLEX,
                                0.5,
                                (255, 0, 0),
                                (0, 0, 0),
                                2,
                                3,
                            )

                        cv2.rectangle(
                            image,
                            (int(box[0] / setting["resolution"]), int(box[1]
/ setting["resolution"])),
                            (int(box[2] / setting["resolution"]), int(box[3]
/ setting["resolution"])),
                            (0, 0, 0),
                            5,
                        )

                        cv2.rectangle(
                            image,
                            (int(box[0] / setting["resolution"]), int(box[1]
/ setting["resolution"])),
                            (int(box[2] / setting["resolution"]), int(box[3]
/ setting["resolution"])),
                            text_color,  # if 0.3 <= distance <= 3.5 and
face_height >= 60 else (255, 0, 0),
                            3,
                        )

                # print(time.time() - st, 1 / ((time.time() - st) if
time.time() - st != 0 else -1), "oooo")

                # st = time.time()
                # update objects in centroid tracker
                objects = ct.update(rects)
                # print(time.time() - st, 1 / ((time.time() - st) if
time.time() - st != 0 else -1), "process..")

                # st = time.time()
                # check for faces that are not checked and check it
                temp = last_id
                objects_ids = [i[0] for i in objects.items()]
                if objects_ids:
                    last_id = max(objects_ids)

                for i in objects_ids:
                    if i > temp:
                        already_check[i] = False
```

```
                # get data including names and progress from another process
if it takes less time than the timeout
                # print(time.time() - st, 1 / ((time.time() - st) if
time.time() - st != 0 else -1), "process..")

                update_current_identity = True

                try:
                    names = ray.get(ct.objects_names.get_all.remote(),
timeout=RAY_TIMEOUT)
                    progresses =
ray.get(ct.recognition_progress.get_all.remote(), timeout=RAY_TIMEOUT)
                except GetTimeoutError:
                    update_current_identity = False

                # update current identity to the gui
                if update_current_identity:
                    for i in objects_ids:
                        name = names.get(i)
                        progress = progresses.get(i)

                        if name == "IN_PROCESS":
                            self.change_infobox_message_signal.emit(
                                {
                                    "name": name,
                                    "ID": i,
                                    "image": ct.objects_data[i].get()[0],
                                    "progress": progress,
                                }
                            )
                        else:
                            if (name not in [
                                "UNKNOWN",
                                "CHECKED_UNKNOWN",
                                "__UNKNOWN__",
                                None,
                                False,
                                ""
                            ] and already_check[i] is False):

                                already_check[i] = True
                                ct.last_deregister.delete.remote(i)

                                if name != ct.recognizer.unknown and not
name.startswith("attacked:"):
                                    data = self.db.quick_get_data(name)
                                    if data is None:
                                        self.db.add_data(name,
*DataBase.default)
                                        data = self.db.quick_get_data(name)

                                    now_time = time.time()
                                    graph_info = data.get("graph_info") if
data.get("graph_info") is not None else []
                                    graph_info.append(now_time)
                                    self.db.update(name,
last_checked=now_time, graph_info=graph_info)

                                self.change_infobox_message_signal.emit(
                                    {
                                        "name": name,
                                        "ID": i,
```

```
                                        "image":
ct.objects_data[i].get()[0],
                                        "progress": 0.9999,
                                    }
                                )

                    # print(time.time() - st, 1 / ((time.time() - st) if
time.time() - st != 0 else -1), "process..")
                    # get last_deregister from another process if it takes
less time than the timeout

                    # st = time.time()

                    update_identity_gui = True

                    try:
                        last_deregister =
ray.get(ct.last_deregister.get_all.remote(), timeout=RAY_TIMEOUT)
                    except GetTimeoutError:
                        update_identity_gui = False

                    # update the all the identity to the gui
                    if update_identity_gui:
                        for i in last_deregister.items():
                            last_objects_names = i[1].get("name")
                            progress = progresses.get(i[0])
                            if last_objects_names is not None and
already_check[i[0]] is False:
                                already_check[i[0]] = True
                                ct.last_deregister.delete.remote(i[0])
                                try:
                                    last_objects_data = i[1]["img"].get()[0]
                                except IndexError:
                                    last_objects_data = image_error

                                if last_objects_names not in [False,
"UNKNOWN??", ""] and \
                                        not
last_objects_names.startswith("attacked:"):
                                    data =
self.db.get_data(last_objects_names)
                                    if data is None:
                                        self.db.add_data(last_objects_names,
*DataBase.default)
                                        data =
self.db.get_data(last_objects_names)

                                    now_time = time.time()
                                    graph_info = data.get("graph_info") if
data.get("graph_info") is not None else []
                                    graph_info.append(now_time)
                                    self.db.update(last_objects_names,
last_checked=now_time, graph_info=graph_info)

                                self.change_infobox_message_signal.emit(
                                    {
                                        "name": last_objects_names,
                                        "ID": i[0],
                                        "image": last_objects_data,
                                        "last": True,
                                        "progress": 0.9999,
                                    }
                                )
```

```
                            elif already_check[i[0]] is False:
                                self.change_infobox_message_signal.emit(
                                    {
                                        "name": last_objects_names,
                                        "ID": i[0],
                                        "progress": progress,
                                    }
                                )

                # put some text to the image; text consists of index
(objectID): the current index of the students
                #                                                 IDD (name):
the ID of the identity

                # print(time.time() - st, 1 / ((time.time() - st) if
time.time() - st != 0 else -1), "process..")

                # st = time.time()
                for (objectID, centroid) in objects.items():

                    if not update_current_identity:
                        name = "IN_PROGRESS"
                    else:
                        name = (
                            general.Most_Common(names.get(objectID))
                            if type(names.get(objectID)) == list
                            else names.get(objectID)
                        )

                    if self.objname_map.get(objectID) is None and
update_current_identity:
                        self.objname_map[objectID] = name

                    text = "ID [{}]".format(objectID)

                    general.putBorderText(
                        image,
                        text,
                        (centroid[0] / setting["resolution"] - 10,
centroid[1] / setting["resolution"] - 20),
                        cv2.FONT_HERSHEY_SIMPLEX,
                        0.5,
                        text_color,
                        (0, 0, 0),
                        2,
                        3,
                    )

                    if name not in ["IN_PROCESS",
                                    "UNKNOWN",
                                    "__UNKNOWN__",
                                    "CHECKED_UNKNOWN",
                                    "UNKNOWN??", ""] and not
name.startswith("attacked:") and update_current_identity:

                        if self.db.quick_get_data(name).get("parent") is
None:
                            self.db.update(name, parent=name)

                        name = self.db.quick_get_data(name).get("parent")

                    general.putBorderText(
```

```
                        image,
                        "IN_PROCESS" if name == "__UNKNOWN__" else name,
                        (centroid[0] / setting["resolution"] - 10,
centroid[1] / setting["resolution"] - 40),
                        cv2.FONT_HERSHEY_SIMPLEX,
                        0.5,
                        text_color,
                        (0, 0, 0),
                        2,
                        3,
                    )

                    cv2.circle(
                        image,
                        (int(centroid[0] / setting["resolution"]),
int(centroid[1] / setting["resolution"])),
                        4,
                        text_color,
                        -1,
                    )

                # print(time.time() - st, 1 / ((time.time() - st) if
time.time() - st != 0 else -1), "process..")
                image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
                total_time = new_frame_time - prev_frame_time
                self.frame_index += 1

                # show fps if fps_show is on
                if setting["fps_show"]:
                    # calculate the fps (average in last n frame if average
fps is on else current fps)
                    if setting["average_fps"]:
                        if total_time < 100:
                            self.avg_fps.add(total_time)
                            total_time = self.avg_fps.get()

                    fps = int(1 / total_time) if total_time != 0 else -1

                    general.putBorderText(
                        image,
                        str(fps),
                        (7, 70),
                        cv2.FONT_HERSHEY_SIMPLEX,
                        2,
                        (100, 255, 0),
                        Color.Black,
                        3,
                        4,
                        cv2.LINE_AA,
                    )
                else:
                    general.putBorderText(
                        image,
                        datetime.now().strftime("%H:%M:%S"),
                        (7, 50),
                        cv2.FONT_HERSHEY_SIMPLEX,
                        1.3,
                        Color.White,
                        Color.Black,
                        3,
                        4,
                        cv2.LINE_AA,
                    )
```

```
                prev_frame_time = new_frame_time

                self.change_pixmap_signal.emit(image)

                if setting["save_as_video"]:
                    self.out.write(image)

        if self.video_type != "screen":
            self.cap.release()


class App(QWidget):
    def __init__(self, parent: QMainWindow):
        super().__init__(parent=parent)
        self.spacer = QSpacerItem(10, 10, QSizePolicy.Minimum,
QSizePolicy.Expanding)
        self.last_progress_ = {}
        self.info_boxes = {}
        self.info_boxes_ID = []
        self.info_boxes_attacked = []
        self.id_navigation = {}
        self.db = DataBase("Students", sync_with_offline_db=True)
        self.db.offline_db_folder_path = setting["db_path"]
        self.storage = self.db.Storage(cache=setting.get("cache_path"))
        parent.setWindowTitle("GoodFaceRecognition")
        parent.resize(1336, 553)
        parent.setStyleSheet("background-color: #0b1615;")

        self.centralwidget = QSplitter(Qt.Horizontal)

        self.image_label = QLabel()
        sizePolicy = QSizePolicy(QSizePolicy.Expanding,
QSizePolicy.Preferred)

sizePolicy.setHeightForWidth(self.image_label.sizePolicy().hasHeightForWidth
())
        self.image_label.setSizePolicy(sizePolicy)
        self.image_label.setMaximumWidth(int(2 * parent.width() / 3))
        self.image_label.setMinimumWidth(int(480 / 2))

        self.image_label.setStyleSheet(
            "color: rgb(240, 240, 240);\n"
            "padding-top: 15px;\n"
            "background: qlineargradient( x1:0 y1:0, x2:0 y2:1, stop:0
rgb(32, 45, 47), stop:.5 #3d5c57, stop:1 rgb("
            "32, 45, 47)); "
            "border-radius: 10px;"
        )
        self.image_label.setAlignment(Qt.AlignTop | Qt.AlignHCenter)

        self.setting_button = general.PushButton()
        self.setting_button.setIcon(QIcon(setting["project_path"] +
"/src/resources/setting.png"))
        self.setting_button.setIconSize(QSize(30, 30))
        self.setting_button.setMaximumSize(QSize(60, 99999999))
        self.setting_button.clicked.connect(self.handle_dialog)
        self.setting_button.setSizePolicy(QSizePolicy.Preferred,
QSizePolicy.Preferred)

        self.face_data_manager_button = general.PushButton()
        self.face_data_manager_button.setIcon(QIcon(setting["project_path"]
+ "/src/resources/manager.png"))
```

```
        self.face_data_manager_button.setIconSize(QSize(30, 30))
        self.face_data_manager_button.setMaximumSize(QSize(60, 99999999))

self.face_data_manager_button.clicked.connect(self.start_face_data_manager)
        self.face_data_manager_button.setSizePolicy(QSizePolicy.Preferred,
QSizePolicy.Preferred)

        button_group_layout = QHBoxLayout()
        button_group_layout.addWidget(self.setting_button)
        button_group_layout.addWidget(self.face_data_manager_button)
        button_group = QWidget()
        button_group.setLayout(button_group_layout)
        button_group.setStyleSheet(
            "background: rgba(255, 255, 255, 0.1);"
            "border-radius: 10px;"
        )
        button_group_stretch_layout = QHBoxLayout()
        button_group_stretch_layout.addWidget(button_group)
        button_group_stretch_layout.addStretch()

        self.verticalLayout_1 = QVBoxLayout()
        self.verticalLayout_1.addWidget(self.image_label)
        self.verticalLayout_1.addLayout(button_group_stretch_layout)
        self.verticalLayout_1.addItem(QSpacerItem(10, 10,
QSizePolicy.Preferred, QSizePolicy.Expanding))

        self.scrollArea = QScrollArea()
        sizePolicy = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Minimum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.scrollArea.sizePolicy().hasHeightForWidth(
))
        self.scrollArea.setSizePolicy(sizePolicy)
        self.scrollArea.setAcceptDrops(False)
        self.scrollArea.setAutoFillBackground(False)
        self.scrollArea.setStyleSheet("background-color: #142523;\n"
"border-radius: 10px;")
        self.scrollArea.verticalScrollBar().setStyleSheet(scrollbar_style)
        self.scrollArea.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setAlignment(Qt.AlignLeading | Qt.AlignLeft |
Qt.AlignVCenter)

        self.scrollAreaWidgetContents = QWidget()
        self.scrollAreaWidgetContents.setGeometry(QRect(0, 0, 563, 539))
        self.verticalLayout = QVBoxLayout(self.scrollAreaWidgetContents)
        self.verticalLayout.addStretch()
        self.verticalLayout.setDirection(QBoxLayout.BottomToTop)

        self.scrollArea.setWidget(self.scrollAreaWidgetContents)
        self.scrollArea.raise_()
        self.image_label.raise_()
        self.cameraVerticalLayout = QWidget()
        self.cameraVerticalLayout.setLayout(self.verticalLayout_1)
        self.cameraVerticalLayout.setMaximumWidth(int(2 * parent.width() /
3))

        self.centralwidget.addWidget(self.cameraVerticalLayout)
        self.centralwidget.addWidget(self.scrollArea)
        self.centralwidget.setSizes([parent.width() // 2, parent.width() //
2])
```

```
        parent.setCentralWidget(self.centralwidget)

        self.thread = VideoThread()
        self.thread.change_pixmap_signal.connect(self.update_image)

self.thread.change_infobox_message_signal.connect(self.update_infobox)
        self.thread.start()

    @staticmethod
    def start_face_data_manager():
        print(f"open file \"{setting['project_path'] +
'/FaceDataManager.py'}\"")
        Thread(target=os.system, args=(f"python {setting['project_path'] +
'/FaceDataManager.py'}",)).start()

    def handle_dialog(self):
        dlg = ui_popup.Ui_Dialog(self, default_setting=setting,
image=deepcopy(now_frame))

        if dlg.exec():
            print("Success!")
            if setting.get("video_change") is True:
                self.thread.release_cam()
                self.thread = VideoThread()
                self.thread.change_pixmap_signal.connect(self.update_image)

self.thread.change_infobox_message_signal.connect(self.update_infobox)
                self.thread.start()
                setting["video_change"] = None
        else:
            print("Cancel!")

    def info_box_popup(self, index: int, box: QLabel, cv_image: np.ndarray):
        dlg = ui_popup2.Ui_Dialog(self)

        avg_color = list(map(int, np.average(np.average(cv_image, axis=0),
axis=0)))

        dlg.Image_box.setStyleSheet(
            "border-radius: 10px;\n"
            "border: 2px solid rgb(202, 229, 229);"
            f"background-color: rgb({avg_color[2]}, {avg_color[1]},
{avg_color[0]});\n"
        )

        data = box.text().lstrip("<font
size=8><b>").rstrip("</font>").split("</b></font><br><font size=4>")
        data = [*data[0].split(": "), *data[1].split(" ")]
        name = data[0]
        dlg.name = name
        date_: str = data[2]
        time_: str = data[3]

        if self.info_boxes_ID[index] == ct.recognizer.unknown:
            return

        personal_data = Student().load_from_db(self.db,
self.info_boxes_ID[index])
        dlg.lineEdit.setText(personal_data.realname)

        if name.startswith("attacked:"):
            return
```

```
        try:
            IDD: str = personal_data.IDD
        except ValueError:
            IDD = name
            ct.recognizer.name_map[IDD] = name

        dlg.Image_box.setPixmap(
            general.round_Pixmap(
                general.convert_cv_qt(
                    general.generate_profile(
                        IDD,
                        image_source=setting["project_path"] +
"/src/resources/unknown_people.png",
                        font_path=setting["project_path"] +
"/src/resources/Kanit-Medium.ttf",
                    ),
                    dlg.Image_box.size().width() - 10,
                    dlg.Image_box.size().height() - 10,
                ),
                10,
            )
        )

        Thread(target=lambda:
self.__load_image_passive(imageBox=dlg.Image_box, ID=IDD)).start()

        if IDD in []:
            return

        if personal_data is not None:
            info_data_graph_info =
personal_data.student_attendant_graph_data
            data_x, data_y =
AttendantGraph(today=datetime.today()).load_datetimes(info_data_graph_info).
data_in_week()
            dlg.plot_graph(data_x, data_y)
            raw_info_data_except_graph_info = {}
            key_queue = [
                Student.FIRSTNAME,
                Student.LASTNAME,
                Student.NICKNAME,
                Student.STUDENT_ID,
                Student.STUDENT_CLASS,
                Student.STUDENT_CLASS,
                Student.LAST_CHECKED,
            ]
            for key in key_queue:
                value = personal_data.to_dict()[key]
                if key != Student.STUDENT_ATTENDANT_GRAPH_DATA:
                    if key == Student.LAST_CHECKED:
                        value = datetime.fromtimestamp(value).strftime("%d
%b %Y %X")
                    raw_info_data_except_graph_info[key] = value

            raw_info_data_except_graph_info["active_days"] = len(

Arrange(AttendantGraph().load_datetimes(info_data_graph_info).dates).arrange
_in_all_as_day()
            )
            print(raw_info_data_except_graph_info)
            dlg.add_data(raw_info_data_except_graph_info)

        dlg.ID.setText(IDD)
```

```
        dlg.exec()
        print("IDD:", IDD)

        if dlg.name != name:
            with open(setting["name_map_path"], "r") as file:
                raw_data = file.read()

            if raw_data:
                information = loads(raw_data)
            else:
                information = {}

            IDD_old = IDD
            if IDD.startswith("unknown:"):
                del information[IDD]
                del ct.recognizer.name_map[IDD]
                IDD = IDD.lstrip("unknown:")
                for index, load_id_id in enumerate(ct.recognizer.loaded_id):
                    if load_id_id == IDD_old:
                        ct.recognizer.loaded_id[index] = IDD

                processed_face: Recognition.ProcessedFace =
Recognition.ProcessedFace(
                    ct.faceRecPath + r"\unknown\{}.pkl".format(IDD))

                processed_face.IDD = IDD
                processed_face.filename = ct.faceRecPath +
r"\known\{}.pkl".format(IDD)
                processed_face.save()
                os.remove(ct.faceRecPath + r"\unknown\{}.pkl".format(IDD))

                if self.db.get_data(IDD_old) is not None:
                    db_data = self.db.get_data(IDD_old)
                    self.db.delete(IDD_old)
                    self.db.add_data(
                        IDD,
                        realname=db_data.get("realname", ""),
                        surname=db_data.get("surname", ""),
                        nickname=db_data.get("nickname", ""),
                        student_id=db_data.get("student_id", 0),
                        student_class=db_data.get("student_class", ""),
                        class_number=db_data.get("class_number", 0),
                        active_days=db_data.get("active_days", 0),
                        last_checked=db_data.get("last_checked", 0),
                        graph_info=db_data.get("graph_info", []),
                        check_name=dlg.name,
                    )

            information[IDD] = dlg.name
            with open(ct.recognizer.name_map_path, "w") as file:
                dump_information = dumps(information)
                if dump_information:
                    file.write(dump_information)
                    ct.recognizer.name_map[IDD] = dlg.name

            for i in range(index + 1):
                if self.info_boxes_ID[i] == IDD or self.info_boxes_ID[i] ==
IDD_old:
                    textbox: QLabel = self.id_navigation[i]["message_box"]
                    textbox.setText(
                        f"<font size=8><b>{dlg.name}:
{index}</b></font><br><font size=4>{date_} {time_}</font>"
                    )
```

```
    def new_info_box(self, message, cv_image, ID) -> (QLabel, QLabel,
QHBoxLayout):
        _translate = QCoreApplication.translate
        horizontalLayout = QHBoxLayout(self.scrollAreaWidgetContents)
        box = QLabel(self.scrollAreaWidgetContents)
        sizePolicy = QSizePolicy(QSizePolicy.Preferred,
QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(box.sizePolicy().hasHeightForWidth())
        box.setSizePolicy(sizePolicy)
        box.setMinimumSize(QSize(0, 160))
        box.setMaximumSize(QSize(16777215, 160))
        box.setFont(QFont(setting["font"]))
        box.setStyleSheet(
            "background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop: 0
rgb(62, 83, 87), stop: 1 rgb(32, 45, 47));"
            "color: #8ba0a3;"
            "padding-left: 10px;"
            "border-radius: 10px;"
        )
        box.setText(_translate("MainWindow", message))
        box.setTextFormat(Qt.RichText)
        box.mousePressEvent = lambda _: self.info_box_popup(ID, box,
deepcopy(cv_image))
        img_box = QLabel(self.scrollAreaWidgetContents)
        sizePolicy = QSizePolicy(QSizePolicy.Preferred,
QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(img_box.sizePolicy().hasHeightForWidth())
        img_box.setSizePolicy(sizePolicy)
        img_box.setMinimumSize(QSize(160, 160))
        img_box.setMaximumSize(QSize(160, 160))
        img_box.setStyleSheet("background-color: #114f46;" "border-radius:
10px;" "border: 3px solid #0a402c;")
        if cv_image is None or not cv_image.any():
            cv_image = image_error
        img_box.setPixmap(
            general.round_Pixmap(
                general.convert_cv_qt(
                    cv2.cvtColor(cv_image, cv2.COLOR_RGB2BGR),
                    img_box.size().width() - 10,
                    img_box.size().height() - 10,
                ),
                10,
            )
        )
        img_box.setAlignment(Qt.AlignCenter)
        horizontalLayout.addWidget(img_box)
        horizontalLayout.addWidget(box)

        return img_box, box, horizontalLayout

    @pyqtSlot(np.ndarray)
    def update_image(self, cv_img):
        avg_color_top = list(map(int, np.average(np.average(cv_img[0:
int(cv_img.shape[0] / 4)], axis=0), axis=0)))
        avg_color_bottom = list(
            map(
```

```
                int, np.average(np.average(cv_img[int(3 * cv_img.shape[0] /
4): int(cv_img.shape[0])], axis=0), axis=0)
            )
        )

        pixmap = general.convert_cv_qt(
            cv_img,
            self.image_label.size().width() - 10,
            self.image_label.size().height() - 10,
        )

        self.image_label.setStyleSheet(
            "color: rgb(240, 240, 240);\n"
            "padding-top: 15px;\n"
            f"background: qlineargradient( x1:0 y1:0, x2:0 y2:1, "
            f"stop:0 rgb({avg_color_top[2]}, {avg_color_top[1]},
{avg_color_top[0]}),"
            f"stop:.5 rgb({avg_color_bottom[2]}, {avg_color_bottom[1]},
{avg_color_bottom[0]}), stop:.9 rgba( "
            "255, 255, 255, 0)); "
            "border-radius: 10px;"
        )

        rounded = general.round_Pixmap(pixmap, 10)
        self.image_label.setPixmap(rounded)

    def __load_image_passive(self, index: int = None, imageBox: QWidget =
None, ID: str = None):
        if imageBox is None and index is not None:
            imageBox = self.id_navigation[index]["img_box"]
            load_image =
self.storage.smart_get_image(self.info_boxes_ID[index])

        elif index is None:
            load_image = self.storage.smart_get_image(ID)

        else:
            load_image = unknown_image

        if not (load_image is not None and load_image is not False and
load_image.any()):
            return

        pixmap = general.convert_cv_qt(
            load_image,
            imageBox.size().width() - 10,
            imageBox.size().height() - 10,
        )
        rounded = general.round_Pixmap(pixmap, 10)
        imageBox.setPixmap(rounded)

    def set_infobox_progress(self, progress, index, special_state=False,
name=None):
        textBox: QLabel = self.id_navigation[index]["message_box"]
        imageBox: QLabel = self.id_navigation[index]["img_box"]

        def _animate(value):
            grad = f"background: qlineargradient(x1:0, y1:0, x2:1, y2:0,
stop: 0 #1b8c7b, stop: {value} #1bb77b, " \
                   f"stop: {value + 0.001} rgb(62, 83, 87), stop: 1 rgb(32,
45, 47)); padding-left: 10px;"
            textBox.setText(f"<font size=8><b>{round(value *
100)}</b></font>%")
```

```
            textBox.setStyleSheet(grad)

        def __animate(value):
            grad = f"background: qlineargradient(x1:0, y1:0, x2:1, y2:0, " \
                   f"stop: 0 rgb{value.red(), value.green(), value.blue()}, 
" \
                   f"stop: 1 rgb{value.red(), value.green() - 34, 
value.blue()}); padding-left: 10px;"
            textBox.setStyleSheet(grad)

        if progress == self.last_progress_[index]:
            return
        if progress == 0.9999:  # the last update (update the name)
            animation = QVariantAnimation(self)
            animation.valueChanged.connect(_animate)
            animation.setStartValue(0.001 if self.last_progress_[index] == 0 
else self.last_progress_[index])
            animation.setEndValue(0.999)
            animation.setDuration(500)
            if special_state:
                animation1 = QVariantAnimation(self)
                animation1.valueChanged.connect(__animate)
                animation1.setStartValue(QtGui.QColor(27, 183, 123))
                animation1.setEndValue(QtGui.QColor(183, 160, 27))
                animation1.setDuration(500)
                animation.finished.connect(lambda: animation1.start())
                if name is not None:
                    animation_imgbox = QVariantAnimation(self)
                    animation_imgbox.valueChanged.connect(
                        lambda value: imageBox.setStyleSheet(
                            f"border: 3px solid rgb({value.red()}, 
{value.green()}, {value.blue()});"
                        )
                    )

animation_imgbox.setStartValue(QtGui.QColor(imageBox.palette().window().colo
r().rgb()))
                    animation_imgbox.setEndValue(QtGui.QColor(183, 160, 27))
                    animation_imgbox.setDuration(500)
                    animation_imgbox.start()

                    animation1.finished.connect(lambda: 
textBox.setText(name))
            else:
                if name is not None:
                    animation_imgbox = QVariantAnimation(self)
                    animation_imgbox.valueChanged.connect(
                        lambda value: imageBox.setStyleSheet(
                            f"border: 3px solid rgb({value.red()}, 
{value.green()}, {value.blue()});"
                        )
                    )

animation_imgbox.setStartValue(QtGui.QColor(imageBox.palette().window().colo
r().rgb()))
                    animation_imgbox.setEndValue(QtGui.QColor(34, 212, 146))
                    animation_imgbox.setDuration(500)

                    if self.info_boxes_attacked[index]:
                        animation1 = QVariantAnimation(self)
                        animation1.valueChanged.connect(__animate)
                        animation1.setStartValue(QtGui.QColor(27, 183, 123))
                        animation1.setEndValue(QtGui.QColor(183, 84, 27))
```

```
                        animation1.setDuration(500)
                        animation_imgbox.setEndValue(QtGui.QColor(183, 84,
27))
                        animation1.finished.connect(lambda:
animation_imgbox.start())
                        animation.finished.connect(lambda:
animation1.start())
                        print("it run??")
                    else:
                        animation_imgbox.start()

                    # print("yes i am", progress,
self.last_progress_[index], special_state)
                    animation.finished.connect(lambda:
textBox.setText(name))

                    if len(self.info_boxes_ID) > index:
                        if self.info_boxes_ID[index] is not False:
                            print("loading image")
                            Thread(target=lambda:
self.__load_image_passive(index)).start()

            animation.start()

        elif progress < 0.9999:  # update status of people (not finished;
update the progress bar)
            animation = QVariantAnimation(self)
            animation.valueChanged.connect(_animate)
            animation.setStartValue(min(self.last_progress_[index],
progress))
            animation.setEndValue(max(self.last_progress_[index], progress))
            animation.setDuration(500)
            animation.start()
        else:
            textBox.setStyleSheet(
                f"background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:
0 rgb(62, 83, 87),"
                " stop: 1 rgb(32, 45, 47)); padding-left: 10px;"
            )
        self.last_progress_[index] = progress

    @pyqtSlot(dict)
    def update_infobox(self, data: dict):
        _translate = QCoreApplication.translate
        name = data.get("name")
        ID = data.get("ID")
        image = data.get("image")
        last = data.get("last")
        progress = data.get("progress")
        progress = 0 if progress is None else progress
        state = False

        if self.info_boxes.get(ID) is None:
            self.info_boxes[ID] = True
            self.verticalLayout.removeItem(self.spacer)
            img_box, message_box, layout = self.new_info_box(f"<font
size=8><b>Finding face..</b></font>", image, ID)
            self.id_navigation[ID] = {"img_box": img_box, "message_box":
message_box}
            self.verticalLayout.addLayout(layout)

        else:
            if name is None:
```

```
                self.info_boxes_attacked.append(False)
            elif name is not False and name.startswith("attacked:"):
                self.info_boxes_attacked.append(True)
            else:
                self.info_boxes_attacked.append(False)

            if name == "IN_PROCESS" or name == "__UNKNOWN__":
                message = None
            elif (last is True and name is False) or name == "":
                message = f'<font size=8><b>FAILED: {ID}</b></font><br><font
size=4>"' \
                          f'{time.strftime("%D/%M %H:%M:%S",
time.localtime())}</font>'
                state = True
                self.info_boxes_ID.append(False)
            elif name is None:
                message = f"<font size=8>...</font>"
            elif name == ct.recognizer.unknown:
                self.info_boxes_ID.append(ct.recognizer.unknown)
                message = f'<font size=8><b>UNKNOWN:
{ID}</b></font><br><font size=4>"' \
                          f'{time.strftime("%D/%M %H:%M:%S",
time.localtime())}</font>'
            else:
                if name not in ["IN_PROCESS",
                                "UNKNOWN",
                                "__UNKNOWN__",
                                "CHECKED_UNKNOWN",
                                "UNKNOWN??",
                                ""] and not name.startswith("attacked:"):

                    if self.db.get_data(name).get("parent") is None:
                        self.db.update(name, parent=name)

                    name = self.db.get_data(name)["parent"]

                mapped_name = ct.recognizer.name_map.get(name)
                self.info_boxes_ID.append(name)
                if mapped_name is None:
                    mapped_name = name

                if len(mapped_name) > 20:
                    mapped_name = mapped_name[:20] + "..."

                message = f"<font size=8><b>{mapped_name}:
{ID}</b></font><br><font size=4>" \
                          f"{time.strftime('%D/%M %H:%M:%S',
time.localtime())}</font>"

            if self.last_progress_.get(ID) is None:
                self.last_progress_[ID] = 0.001

            if len(self.scrollAreaWidgetContents.children()) <= ID:
                return

            imgbox: QLabel = self.id_navigation[ID]["img_box"]

            if image is not None and image.any():
                pixmap = general.convert_cv_qt(
                    cv2.cvtColor(image, cv2.COLOR_BGR2RGB),
                    imgbox.size().width() - 10,
                imgbox.size().height() - 10,
        )
```

```
            rounded = general.round_Pixmap(pixmap, 10)
                  imgbox.setPixmap(rounded)

              self.set_infobox_progress(progress, ID,
special_state=state, name=message)

              if last is True:
                  del self.last_progress_[ID]


if __name__ == "__main__":

QApplication.setHighDpiScaleFactorRoundingPolicy(Qt.HighDpiScaleFactorR
oundingPolicy.PassThrough)
    QCoreApplication.setAttribute(Qt.AA_EnableHighDpiScaling)
    app = QApplication(sys.argv)
    MainWindow = QMainWindow()
    a = App(MainWindow)
    MainWindow.show()

    RepeatedTimer(60 * 10, lambda: ct.recognizer.update(a.storage))
    sys.exit(app.exec_())
```

2.2 โค้ดสำหรับเว็บแอพลิเคชันสำหรับเพื่อเข้าถึงข้อมูลการเช็คชื่อ (app.py)

```
from flask import Flask, render_template, Response, jsonify, send_file
from markupsafe import escape
from typing import *
from copy import deepcopy
from datetime import datetime, timedelta
from itertools import islice
from PIL import Image, ImageDraw, ImageFont
import numpy as np
from io import BytesIO
from base64 import b64encode, decodebytes
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename
import os.path
import os
import uuid
import json
import re
from dataclasses import dataclass
from src.DataBase import DataBase
from src.attendant_graph import Arrange, AttendantGraph
from src.studentSorter import Student, StudentSorter
from src.FaceTrainer_server import QueueTrainer, Queue
from src.PdfGenerator import PdfTable
from src.leaderboard import LeaderBoard
# from src.FaceTrainer_server import VideoFaceTrainer

app = Flask(__name__)

UPLOAD_FOLDER = "recieved\images"
ALLOWED_EXTENSTIONS = ('.png', '.jpg', '.jpeg', '.gif', '.mp4')

app.config["UPLOAD_FOLDER"] = UPLOAD_FOLDER
```

```
def make_safe_filename(filename):
    return "".join([c for c in filename if re.match(r'\w', c)])


def chunks(data, SIZE=50):
    it = iter(data)
    for i in range(0, len(data), SIZE):
        yield {k: data[k] for k in islice(it, SIZE)}


def generate_profile(
    name: str, image_source: str = "static/images/unknown_people.png"
) -> np.ndarray:
    img = Image.open(image_source).convert("RGBA")
    draw = ImageDraw.Draw(img)
    font = ImageFont.truetype("Kanit-Medium.ttf", 50)
    name = name[:2]
    offset_x = font.getlength(name)
    height, width = img.height, img.width
    draw.text(
        (int(width / 2 - (offset_x / 2)), height // 2 - 55),
        name,
        (203, 203, 203),
        font=font,
    )
    buffered = BytesIO()

    img.save(buffered, format="PNG")
    img.resize((86, 86))

    return "" +
b64encode(buffered.getvalue()).decode("ascii")


def load_raw_data(page: int) -> tuple[dict, int]:
    raw_data: dict = db.get_database()
    del raw_data["last_update"]
    raw_data = list(chunks(raw_data, SIZE=page_capacity))
    page = min(len(raw_data) - 1, page)
    raw_data = raw_data[page]
    return raw_data


def class_index(_db: DataBase = None, _db_dict: Dict = None) ->
List[str]:
    if _db is None and _db_dict is None:
        return
    if _db_dict is not None:
        db_dict = _db_dict
    else:
        db_dict = _db.get_database()
        del db_dict["last_update"]

    classes: set[str] = set()
    student_idd: str
```

```
    for student_idd in db_dict:
        classes.add(db_dict[student_idd][Student.STUDENT_CLASS])
    return list(classes)


def process_raw_data(raw_data, image_links, check_state):
    for key in raw_data:
        raw_data[key]["realname"] = (
            key if not raw_data[key][Student.FIRSTNAME] else
raw_data[key][Student.FIRSTNAME]
        )

        raw_data[key]["realname"] = raw_data[key]["realname"][:20]

        student = Student().load_from_dict(key, raw_data[key])

        check_state[key] = student.status(LATETIME)

        last_checked: datetime = datetime.fromtimestamp(
            raw_data[key][Student.LAST_CHECKED])
        if last_checked == datetime.fromtimestamp(0):
            raw_data[key][Student.LAST_CHECKED] = "Never checked."
        elif Arrange.same_day(last_checked, datetime.now()):
            raw_data[key][Student.LAST_CHECKED] =
last_checked.strftime(
                "%H:%M:%S")
        elif Arrange.same_week(last_checked, datetime.now()):
            raw_data[key][Student.LAST_CHECKED] =
last_checked.strftime(
                "%a %H:%M")
        else:
            raw_data[key][Student.LAST_CHECKED] =
last_checked.strftime(
                "%d %b %y")

        if image_exists.get(key, False):
            image_links[key] = db_storage.get_image_link(key)
        else:
            image_links[key] = generate_profile(key)


def allow_extension(filename) -> bool:
    return os.path.splitext(filename)[1].lower() in ALLOWED_EXTENSTIONS


@app.route("/")
def index():
    return render_template("home.jinja")


@app.route("/names/<int:page>")
def information_name(page):
    SORTED_BY = 'name'
    raw_data = load_raw_data(page)
    raw_data = dict(
        sorted(
            raw_data.items(),
```

```
                key=lambda a: a[1].get(Student.FIRSTNAME)
                if a[1].get(Student.FIRSTNAME) != 0
                else "ЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍ8,
            )
        )
        image_links: dict[str, str] = {}
        check_state: dict[str, bool] = {}

        process_raw_data(raw_data=raw_data,
                         image_links=image_links,
                         check_state=check_state
                         )

        return render_template(
            "page.jinja",
            data=raw_data,
            page_number=page,
            image_links=image_links,
            check_state=check_state,
            sorted_by=SORTED_BY
        )


    @app.route('/downloaddoc/<page>/<subpage>')
    def downloadFile(page, subpage):
        db_dict: Dict[str] = db.get_database()
        student_class = f"{page}/{subpage}"
        ss: StudentSorter = StudentSorter(data=db_dict,
    late_checked_hour_minute=LATETIME)
        class_students: List[str] =
    ss.sort_as_classes().get().get(student_class)
        students: List[Tuple[str, str, str, str]] = []
        for student_idd in class_students:
            student: Student = Student().load_from_dict(student_idd,
    db_dict.get(student_idd))
            students.append((student.student_class_number,
                             student.student_id,
                             student.realname,
                             student.status(LATETIME)))

        pdf: PdfTable = PdfTable(
                header=("เลขที่", "เลขประจำตัวนักเรียน", "ชื่อ - นามสกุล", "สถานะ"),
                data=students,
                column_ratio=(0.7, 2.7, 6, 2),
                column_align=('C', 'C', 'L', 'C'),
                output_filename=os.path.dirname(__file__) + "/" +
    CACHE_PATH+make_safe_filename("/result_"+student_class+".pdf"),
                student_class=student_class,
                student_program="วิทย์-คณิต",
                font_path=os.path.dirname(__file__) + "/" +
    "static/font/THSarabunNew.ttf",
                font_path_bold=os.path.dirname(__file__) + "/" +
    "static/font/THSarabunNew Bold.ttf",
                logo_image_path=os.path.dirname(__file__) + "/" +
    "static/images/scius.png"
                )
```

```
    pdf_file: BytesIO = pdf.get_pdf_bytes().getvalue()


    return send_file(
        BytesIO(pdf_file),
download_name="result_"+make_safe_filename(student_class)+".pdf",
        as_attachment=True
    )



@app.route("/class-attendance-data/<page>")
def information_class_get(page):
    db_dict: Dict = db.get_database()
    del db_dict["last_update"]

    checked_count: Dict[str, int] = {}
    checked_late_count: Dict[str, int] = {}
    not_checked_count: Dict[str, int] = {}

    for idd in db_dict:
        if db_dict[idd][Student.STUDENT_CLASS].split("/")[0] == page:
            student: Student = Student().load_from_dict(idd,
db_dict[idd])
            subclass = student.student_class.split("/")[1]

            if checked_count.get(subclass) is None:
                checked_count[subclass] = 0

            if checked_late_count.get(subclass) is None:
                checked_late_count[subclass] = 0

            if not_checked_count.get(subclass) is None:
                not_checked_count[subclass] = 0

            status: str = student.status(LATETIME)
            if status == Student.CHECKED:
                checked_count[subclass] += 1

            elif status == Student.CHECKED_LATE:
                checked_late_count[subclass] += 1

            elif status == Student.NOT_CHECKED:
                not_checked_count[subclass] += 1

    return jsonify({"checkedCount": checked_count,
                    "checkedLateCount": checked_late_count,
                    "notCheckedCount": not_checked_count})

@app.route("/classes")
def information_classes():
    db_dict: Dict = db.get_database()
    del db_dict["last_update"]
```

```
    classes_major: List[str] = [c.split("/")[0] for c in
class_index(_db_dict=db_dict)]
    return render_template("not_found.jinja", header="Class list",
subheader="these are valid class:", items=list(set(classes_major)),
page="")


@app.route("/classes/<page>")
def information_class(page):
    db_dict: Dict = db.get_database()
    del db_dict["last_update"]
    classes_major: List[str] = [c.split("/")[0] for c in
class_index(_db_dict=db_dict)]
    if page not in classes_major:
        return render_template("not_found.jinja", header="Class not
found!", subheader="these are valid class:",
items=list(set(classes_major)), page="")
    return render_template("class_attendance.jinja")

@app.route("/classes/<page>/<subpage>")
def information_class_subclass(page, subpage):
    db_dict: Dict = db.get_database()
    valid_class: List[str] = []
    del db_dict["last_update"]
    for c in class_index(_db_dict=db_dict):
        if c.split("/")[0] == page and len(c.split("/")) == 2:
            valid_class.append(c.split("/")[1])
    if subpage in valid_class:
        return render_template("subclass_attendance.jinja")
    else:
        return render_template("not_found.jinja", header="Subclass not
found!", subheader="these are valid class:", items=valid_class,
page=page+"/")

@app.route("/subclass-attendance-data/<page>/<subpage>")
def information_class_subclass_get(page, subpage):

    db_dict: Dict = db.get_database()
    students_in_class: List[Student] =
Student.from_idds(StudentSorter(data=db_dict).sort_as_classes().get().g
et(f"{page}/{subpage}", None), db_dict)

    attendance_data = []
    for student in students_in_class:
        data =
Arrange(student.student_attendant_graph_data).arrange_in_day()
        if data:
            earliest: datetime = data[list(data)[0]][0]
            attendance_data.append({"student": student.to_dict(),
                                    "checked": True,
                                    "time": earliest.timestamp(),
                                    "late": student.status(LATETIME) ==
Student.CHECKED_LATE})
        else:
            attendance_data.append({"student": student.to_dict(),
                                    "checked": False,
                                    "late" :True,
```

```
                                        "time": 0})
        attendance_data.sort(key=lambda a:
int(a["student"][student.STUDENT_CLASS_NUMBER]) if
a["student"][student.STUDENT_CLASS_NUMBER] != "0" else "zzzz")
    return jsonify({'attendanceData': attendance_data})


@app.route("/students_id/<int:page>")
def information_student_id(page):
    SORTED_BY = 'student number'
    raw_data: dict = load_raw_data(page)
    raw_data = dict(
        sorted(
            raw_data.items(),
            key=lambda a: int(a[1].get("student_id"))
            if int(a[1].get("student_id")) != 0
            else 999999
        )
    )
    print(raw_data, 'sdfsdfsdfdsfsdfds')
    image_links: dict[str, str] = {}
    check_state: dict[str, bool] = {}

    process_raw_data(raw_data=raw_data,
                     image_links=image_links,
                     check_state=check_state
                     )

    return render_template(
        "page.jinja",
        data=raw_data,
        page_number=page,
        image_links=image_links,
        check_state=check_state,
        sorted_by=SORTED_BY
    )


@app.route("/nicknames/<int:page>")
def information_nickname(page):
    SORTED_BY = 'nickname'
    raw_data: dict = load_raw_data(page)
    raw_data = dict(
        sorted(
            raw_data.items(),
            key=lambda a: a[1].get("nickname")
            if a[1].get("nickname") != 0
            else "ЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍŌЍ8,
            reverse=True,
        )
    )
    image_links: dict[str, str] = {}
    check_state: dict[str, bool] = {}

    process_raw_data(raw_data=raw_data,
                     image_links=image_links,
                     check_state=check_state
```

```
                    )

    return render_template(
        "page.jinja",
        data=raw_data,
        page_number=page,
        image_links=image_links,
        check_state=check_state,
        sorted_by=SORTED_BY
    )


@app.route("/idd/<int:page>")
def information_IDD(page):
    SORTED_BY = 'IDD'
    raw_data: dict = load_raw_data(page)
    raw_data = dict(sorted(raw_data.items()))
    image_links: dict[str, str] = {}
    check_state: dict[str, bool] = {}

    process_raw_data(raw_data=raw_data,
                     image_links=image_links,
                     check_state=check_state
                     )

    return render_template(
        "page.jinja",
        data=raw_data,
        page_number=page,
        image_links=image_links,
        check_state=check_state,
        sorted_by=SORTED_BY
    )


@app.route("/latest/<int:page>")
def information_latest_check(page):
    SORTED_BY = 'latest checked'
    raw_data: dict = load_raw_data(page)
    raw_data = dict(
        sorted(
            raw_data.items(),
            key=lambda a: a[1].get('last_checked'),
            reverse=True
        )
    )

    image_links: dict[str, str] = {}
    check_state: dict[str, bool] = {}

    process_raw_data(raw_data=raw_data,
                     image_links=image_links,
                     check_state=check_state
                     )

    return render_template(
        "page.jinja",
```

```
        data=raw_data,
        page_number=page,
        image_links=image_links,
        check_state=check_state,
        sorted_by=SORTED_BY
    )


@app.route("/createidentity")
def create_identity():
    return render_template("new.jinja")


@app.route("/upload", methods=["POST"])
def upload():
    idd = uuid.uuid4().hex
    file = request.files["file"]
    personal_information =
json.loads(request.form["PersonalInformation"])
    profile = request.form["profile"]
    print("upload", len(personal_information), len(profile), idd),
bool(file)
    save_dir = os.path.join(app.config['UPLOAD_FOLDER'], idd)
    print(personal_information)
    os.mkdir(save_dir)

    if file:
        print(file)
        filename = secure_filename(file.filename)
        file.save(os.path.join(save_dir, filename))

        student: Student = Student().load_from_dict(idd,
personal_information)
        print(student.show_table())
        qt.push(student)

        queueId: Union[Queue.QueueProperties, None] =
qt.queue.get_by_index(-1)
        queueId = qt.inprocess.queueId if queueId is None and
qt.inprocess.queueId is not None \
            else queueId.queueId if queueId is not None else "-"
        return queueId


@app.route("/capture", methods=["POST"])
def capture():
    idd = uuid.uuid4().hex
    data = request.get_json()
    data_urls = data['images']
    personal_information = data["PersonalInformation"]
    profile = data["profile"]
    save_dir = os.path.join(app.config['UPLOAD_FOLDER'], idd)

    os.mkdir(save_dir)
    print(personal_information)
    for index, data_url in enumerate(data_urls):
        data_url = data_url.split('base64,')[1]
```

```
        image = Image.open(BytesIO(decodebytes(bytes(data_url, "utf-
8"))))
        image.save(os.path.join(save_dir, f"{index}.png"))
    if profile:
        data_url = profile.split('base64,')[1]
        image = Image.open(BytesIO(decodebytes(bytes(data_url, "utf-
8"))))
        image.save(os.path.join(save_dir, f"profile.png"))

    student: Student = Student().load_from_dict(idd,
personal_information)
    print(student.show_table())
    qt.push(student,
            lambda: (db_storage.add_image(idd, os.path.join(save_dir,
"profile.png"), (86, 86)),
                     db_storage.add_image(idd+"_HIGHRES",
os.path.join(save_dir, "profile.png"), (360, 360))),
                     print("upload profile image successfull"))

    queueId: Union[Queue.QueueProperties, None] =
qt.queue.get_by_index(-1)
    queueId = qt.inprocess.queueId if queueId is None and
qt.inprocess.queueId is not None \
        else queueId.queueId if queueId is not None else "-"
    print(queueId)
    return queueId


@app.route("/queues/<queue_id>")
def queues_table(queue_id):
    queues: Queue = qt.queue
    table: List[tuple[int, str, Student, str]] = []

    inprocess_queue: Queue.QueueProperties = qt.inprocess
    if inprocess_queue is None:
        table = [("-", "-", "-", "-")]

    else:
        time_taken: timedelta = inprocess_queue.operation_time()
        time_taken_formated: str = ":".join(
            [str(int(i)).zfill(2) for i in
divmod(time_taken.total_seconds(), 60)])

        table.append([
            inprocess_queue.queueIndex,
            inprocess_queue.queueId,
            inprocess_queue.student.firstname,
            time_taken_formated
        ])

        queue: Queue.QueueProperties
        for queue in queues.loop():
            time_taken = inprocess_queue.operation_time() +
(queue.queueIndex -

inprocess_queue.queueIndex)*inprocess_queue.operation_time()
            time_taken_formated = ":".join(
```

```
                [str(int(i)).zfill(2) for i in
divmod(time_taken.total_seconds(), 60)])
            table.append([
                queue.queueIndex,
                queue.queueId,
                queue.student.firstname,
                time_taken_formated
            ])
    print(table)
    return render_template('queue_table.jinja', data=table,
queue_id=queue_id)

@app.route("/queues_get")
def get_queues_table():
    queues: Queue = qt.queue
    table: List[tuple[int, str, Student, str]] = []

    inprocess_queue: Queue.QueueProperties = qt.inprocess
    if inprocess_queue is None:
        table = [("-", "-", "-", "-")]

    else:
        time_taken: timedelta = inprocess_queue.operation_time()
        time_taken_formated: str = ":".join(
            [str(int(i)).zfill(2) for i in
divmod(time_taken.total_seconds(), 60)])

        table.append([
            inprocess_queue.queueIndex,
            inprocess_queue.queueId,
            inprocess_queue.student.firstname,
            time_taken_formated
        ])

        queue: Queue.QueueProperties
        for queue in queues.loop():
            time_taken = inprocess_queue.operation_time() +
(queue.queueIndex -

inprocess_queue.queueIndex)*inprocess_queue.operation_time()
            time_taken_formated = ":".join(
                [str(int(i)).zfill(2) for i in
divmod(time_taken.total_seconds(), 60)])
            table.append([
                queue.queueIndex,
                queue.queueId,
                queue.student.firstname,
                time_taken_formated
            ])
    return jsonify({"data":table})


@app.route("/students/<student_idd>", methods=["GET"])
def student_info_render(student_idd):
    student: Student = Student().load_from_db(db, student_idd)
    profile_image: np.ndarray = db_storage.get_image_link(
```

```
        student_idd) if db_storage.exists(student_idd) else
generate_profile(student_idd)
    return render_template("student.jinja", student=student,
idd=student_idd, profile_image=profile_image)


@app.route("/students-get-data/<student_idd>", methods=["GET"])
def student_info_get(student_idd):
    student: Student = Student().load_from_db(db, student_idd)
    graph_data: list[datetime] = student._student_attendant_graph_data
    print(graph_data, "graphdata")

    graph_data_first_checked: list[float] =
AttendantGraph(today=datetime.now()).load_datetimes(
        graph_data).data_in_month() if graph_data else []

    checked = Arrange(graph_data).arrange_in_all_as_day().get(
        (datetime.now().year, datetime.now().month,
datetime.now().day),
        False)
    student_dict: Dict = student.to_dict()
    print(graph_data_first_checked)
    try:
        student_dict["graph_info"] = [
            list(graph_data_first_checked[0]),
list(graph_data_first_checked[1])]
    except IndexError:
        student_dict["graph_info"] = []
    student_dict["checked"] = checked
    print(student_dict)
    return jsonify({"student": student_dict, "status":
student.status(LATETIME)})

@app.route("/leaderboard/<int:page>")
def leaderboard(page):
    ldb: LeaderBoard = LeaderBoard(db)
    ranks = ldb.load_all_from_db().get_ranks()
    data = {}
    image_links = {}
    student_rank = {}

    ranks = list(chunks(ranks, SIZE=page_capacity))
    page = min(len(ranks) - 1, page)
    ranks = ranks[page]

    for ind, rank in enumerate(ranks):
        student_rank[rank] = (20*page) + ind+1
        if student_rank[rank] == 1:
            student_rank[rank] = "ﾐ
        elif student_rank[rank] == 2:
            student_rank[rank] = "
        elif student_rank[rank] == 3:
            student_rank[rank] = "
        cal_point: Tuple[int, int, int] = ldb.calculate_point(rank)
        data[rank] = {"student": db.get_data(rank),
```

```
                        "points": {"points": round(float(cal_point[0]),
2),
                                   "streak": cal_point[1],
                                   "days": cal_point[2]}}
            print(round(int(cal_point[0]), 3))
            if data[rank]["student"][Student.FIRSTNAME] + " " +
data[rank]["student"][Student.LASTNAME] == " ":
                data[rank]["student"][Student.FIRSTNAME] = rank

            data[rank]["student"][Student.FIRSTNAME] =
data[rank]["student"][Student.FIRSTNAME][:20]
            if image_exists.get(rank, False):
                image_links[rank] = db_storage.get_image_link(rank)
            else:
                image_links[rank] = generate_profile(rank)

        return render_template(
            "leaderboard.jinja",
            data=data,
            image_links=image_links,
            page_number = 0,
            rank=student_rank)

if __name__ == "__main__":
    CACHE_PATH = "cache"
    USED_FOLDER: list[str] = ["db", "trained", CACHE_PATH, "recieved",
"recieved/images"]
    # create used folder.
    for use_folder in USED_FOLDER:
        if not os.path.exists(use_folder):
            os.mkdir(use_folder)
    LATETIME = (9, 0, 0)
    db: DataBase = DataBase("Students",
certificate_path="serviceAccountKey.json", sync_with_offline_db=True)
    db.offline_db_folder_path = "db"
    db_storage: DataBase.Storage = db.Storage(cache=CACHE_PATH)
    image_exists: dict[str, bool] = {}
    page_capacity = 20
    qt: QueueTrainer = QueueTrainer(
        input_path="recieved/images", output_path="trained", db=db,
cache_path=CACHE_PATH)

    qt.core = 3
    for key in db.get_database():
        print("indexing.", key, end="")
        if key.startswith("unknown:"):
            image_exists[key] = False
        else:
            image_exists[key] = db_storage.exists(key)
        if image_exists[key]:
            print("exists")
        else:
            print("not exists")

    app.run(debug=True, port=5000, host="0.0.0.0") # change to
preferred port
```

ประวัติของผู้จัดทำ

ชื่อ-สกุล: นายจีรภัทร สุภาพินิจ

วันเดือนปีเกิด: 21 กุมภาพันธ์ พ.ศ.2549

อายุ: 17 ปี

ศาสนา: พุทธ

สัญชาติ: ไทย

ที่อยู่ปัจจุบัน: 10/205 หมู่บ้านลมทะเล1 หมู่4 ตำบลบ้านฉาง อำเภอบ้านฉาง จังหวัดระยอง 21130

ประวัติการศึกษา: ประถมศึกษา (ป.1 - ป.4) : โรงเรียนสองภาษาระยอง

ประถมศึกษา (ป.4 - ป.6) : โรงเรียนอัสสัมชัญระยอง

มัธยมศึกษาตอนต้น : โรงเรียนระยองวิทยาคม

มัธยมศึกษาตอนปลาย : โรงเรียนสาธิต “พิบูลบำเพ็ญ” มหาวิทยาลัยบูรพา

เบอร์โทรศัพท์: 091-884-2784

Email: kaopancraft@gmail.com