Strategy Pattern 1/2

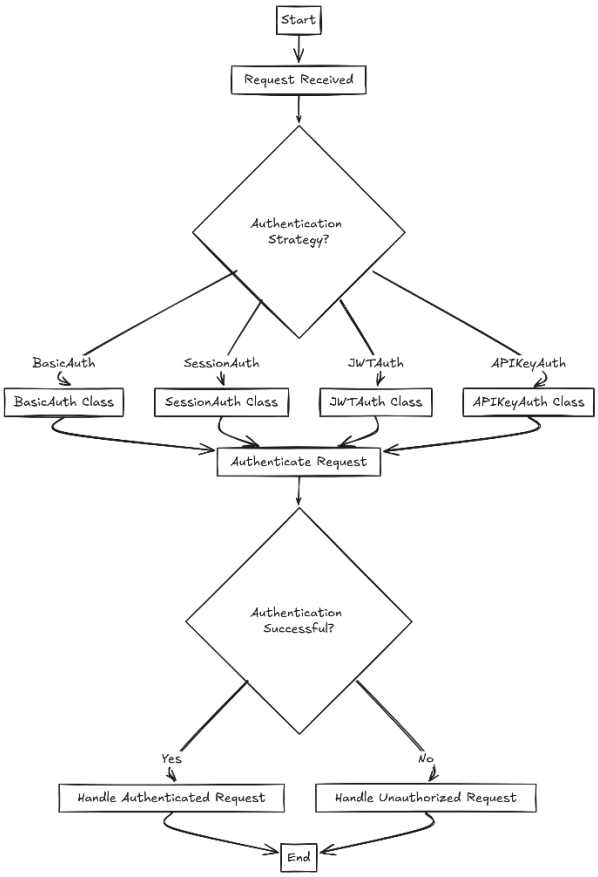
ในการทดลองเพื่อให้ AI สร้างโปรเจคที่ใช้ Strategy Pattern ออกมาทั้งหมดนี้ เราได้ตัดสินใจใช้ โปรเจค **Authentication System** โดยระบบจะเป็นระบบที่ใช้ในการยืนยันตัวตน Client ในรูปแบบต่าง และ ใช้ Strategy Pattern เพื่อออกแบบการใช้งานกลวิธีต่างๆในการยืนยันตัวตน

ผลลัพธ์เบื้องต้นที่ได้:

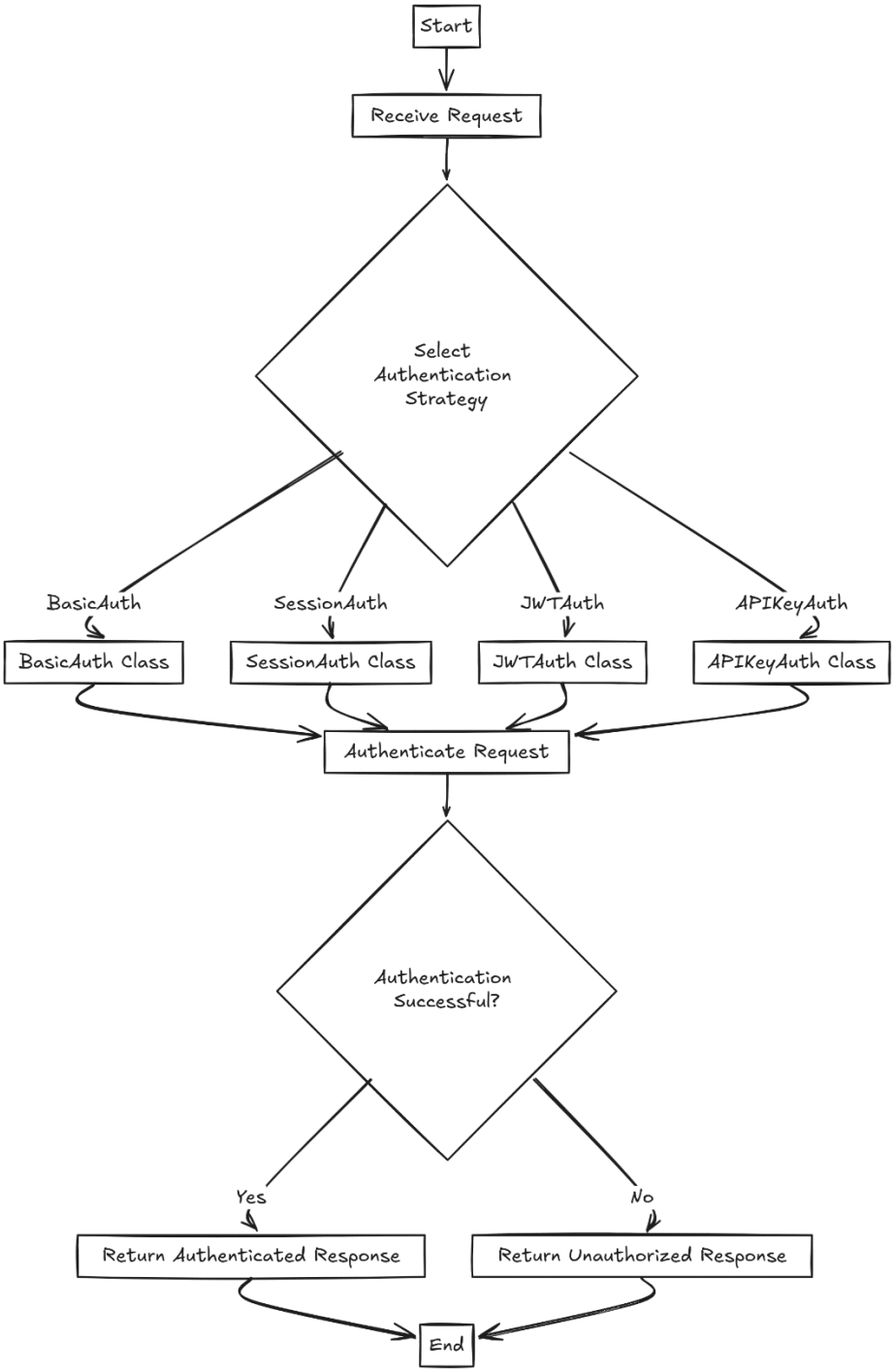
Python-ChatGPT

จากการวิเคราะห์ด้วย Control-flow-graph ที่ได้จาก code ที่ AI สร้างขึ้นพบว่า logic ต่าง ๆ ดูมีความถูกต้องครบถ้วน แต่เมื่อไปดูในส่วนของ Source code พบว่า library ต่าง ๆ ที่ AI นำมาใช้นั้นบางส่วนถูกแจ้งว่า Deprecated ไปเรียบร้อยแล้ว ซึ่งอาจเกิดจากที่ AI มี Bias จากข้อมูลเก่า ๆ ทำให้มีโอกาสสร้าง code ที่เก่าเกินไป

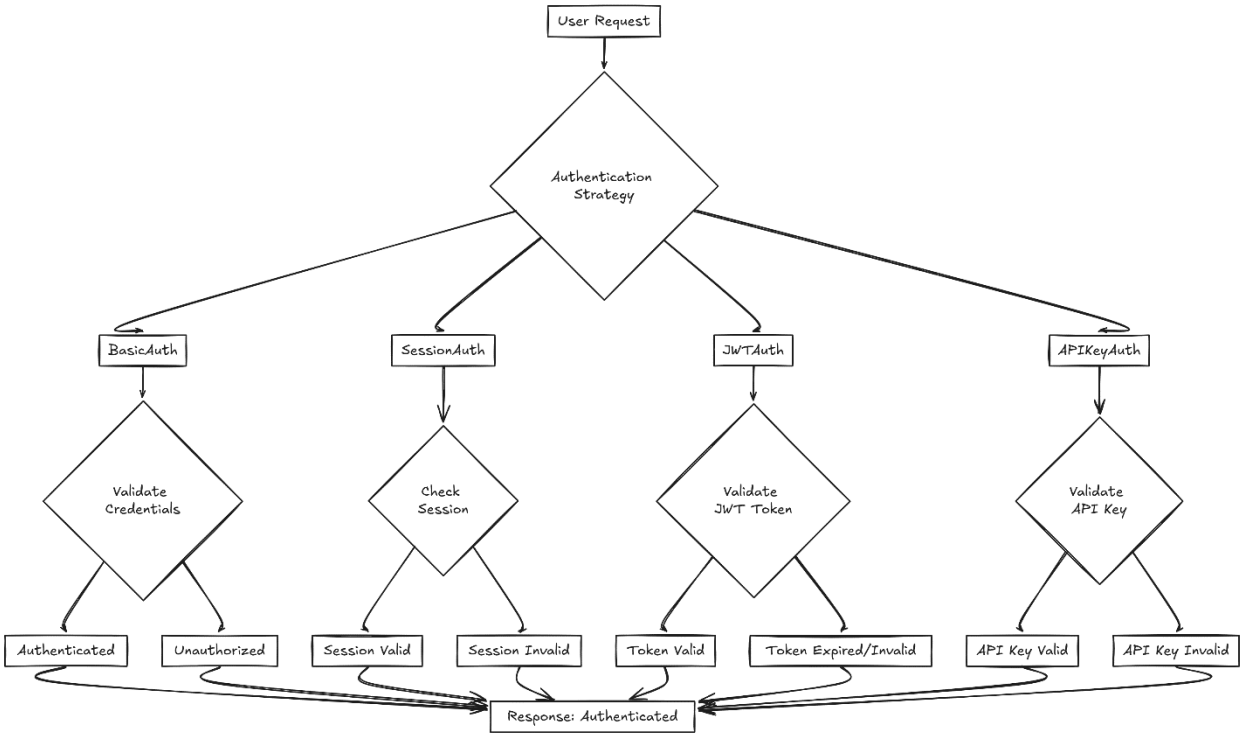
Python-ChatGPT-R1



Python-ChatGPT-R2



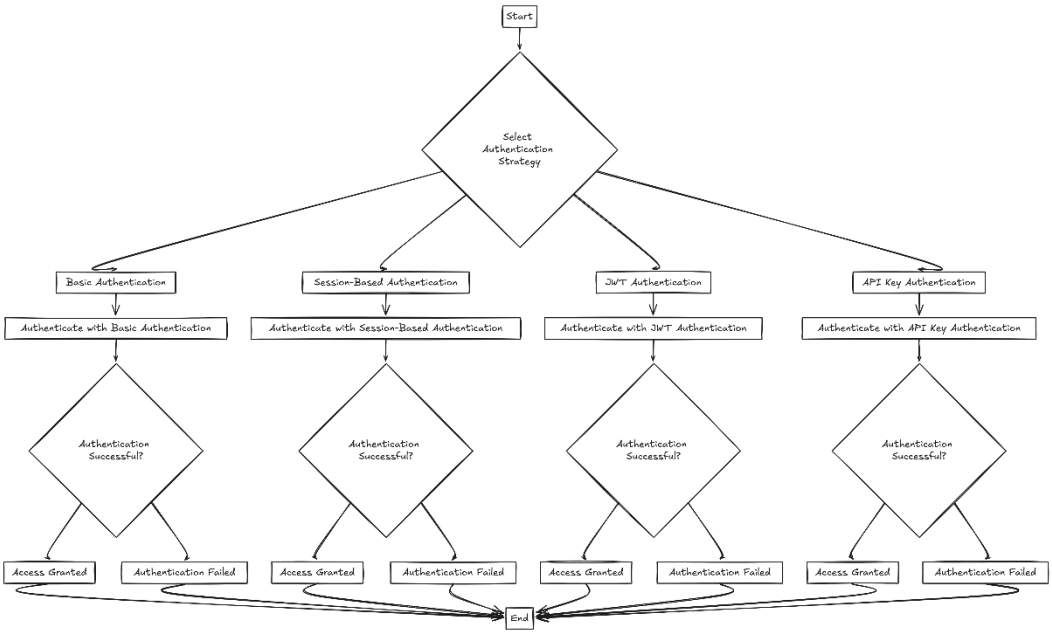
Python-ChatGPT-R3



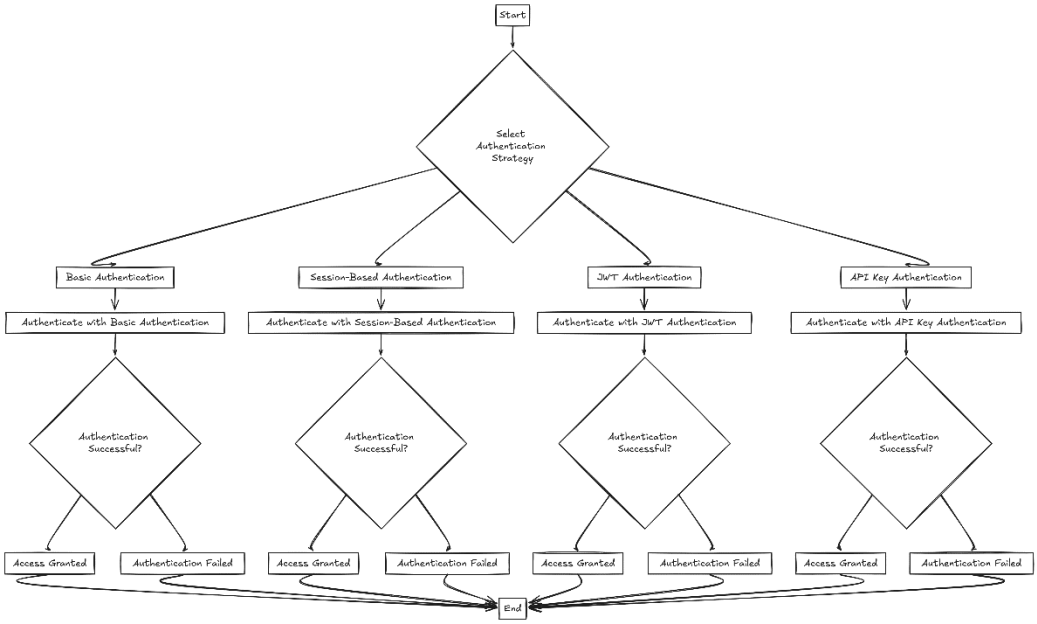
Python-Gemini Flash

จากการวิเคราะห์ Gemini Flash มี logic ของ project ที่ครบถ้วนใน แต่จากการสำรวจ source code พบว่ายังมีข้อผิดพลาดบางอย่าง เช่น การใช้ตัวแปรที่ไม่เคยถูกเซ็ทค่ามาก่อน

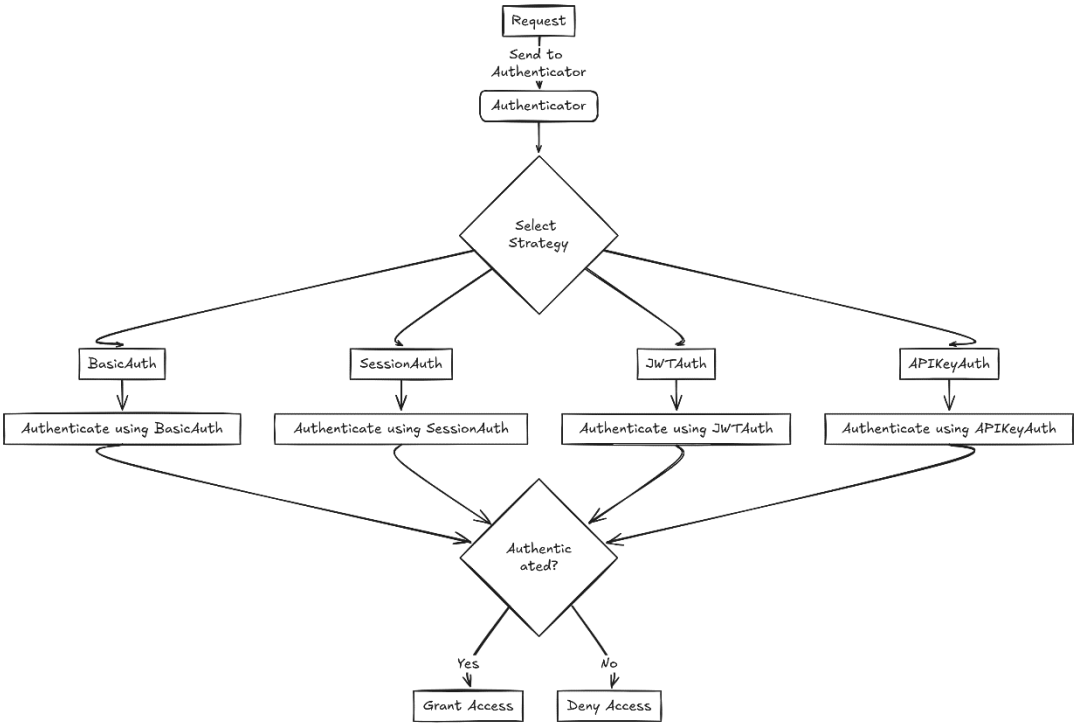
Python-Gemini Flash-R1



Python-Gemini Flash-R2



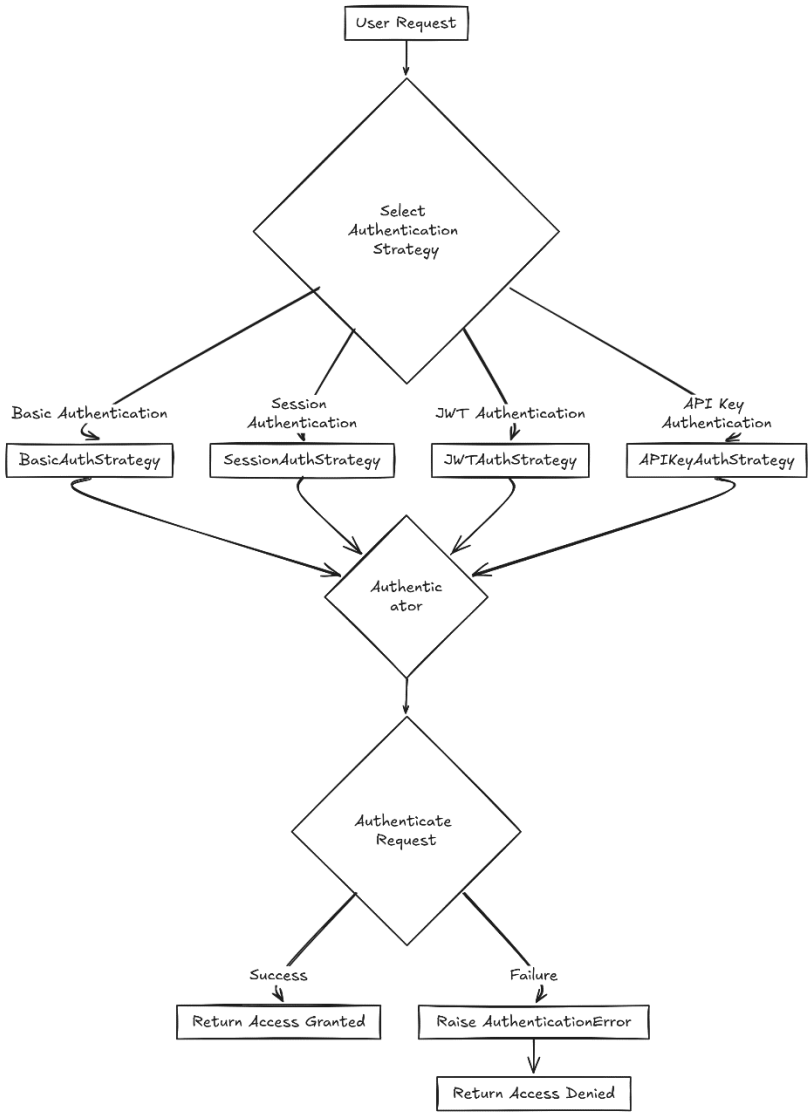
Python-Gemini Flash-R3



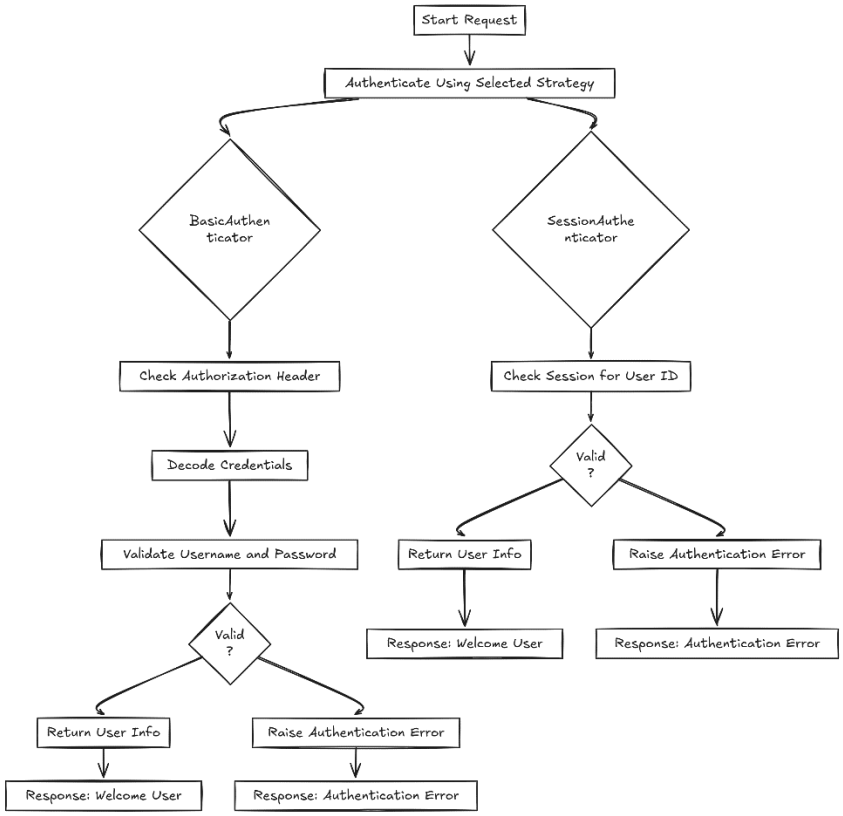
Python-Gemini Pro

จากการวิเคราะห์พบแม้ส่วนใหญ่ Gemini Pro จะมี logic ที่ครบถ้วนแต่ในบางรอบ Gemini Pro นั้นไม่ได้สร้าง source code มาครบถ้วน

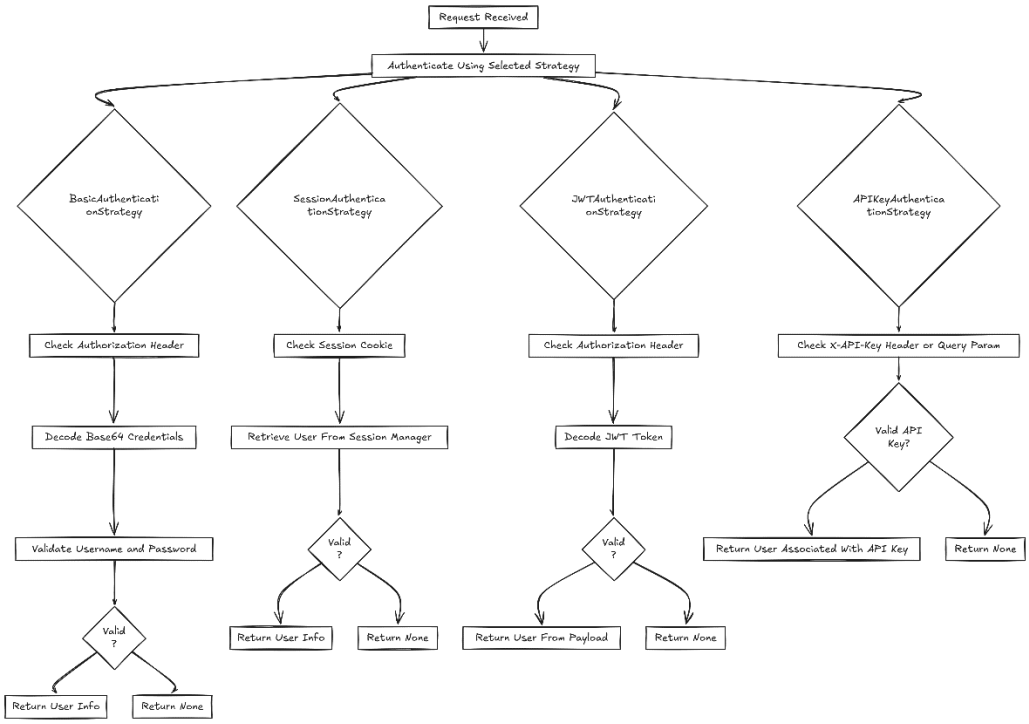
Python-Gemini Pro-R1



Python-Gemini Pro-R2



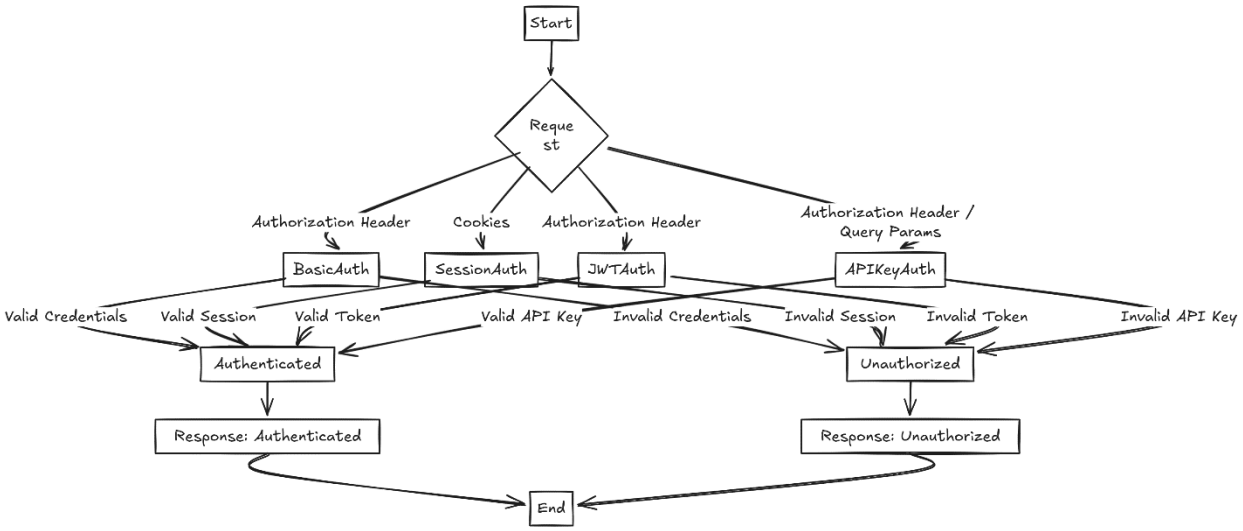
Python-Gemini Pro-R3



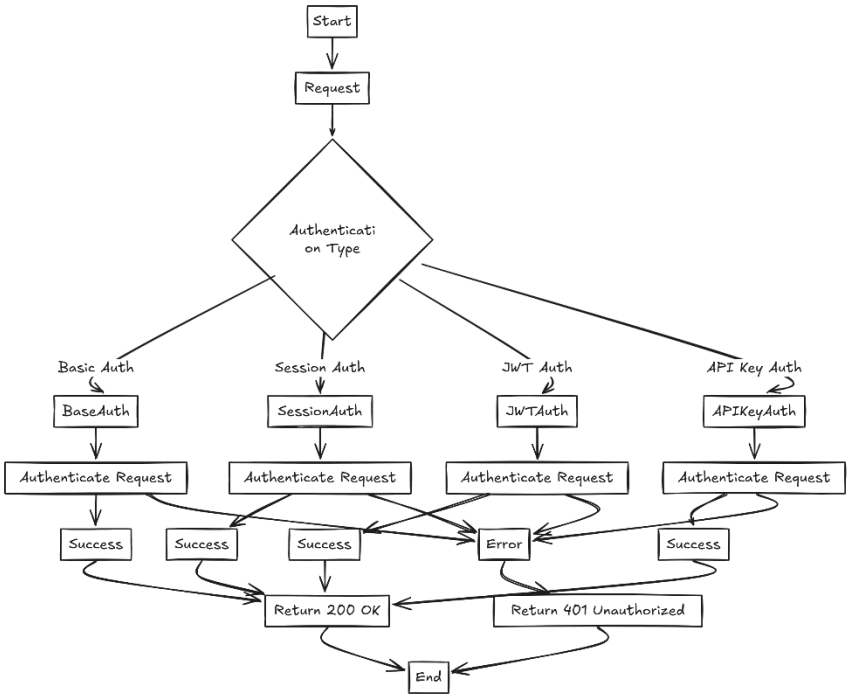
Python-Github Copilot

จากการวิเคราะห์ CFG Copilot สามารถสร้าง logic ของ Authentication system ได้ถูกต้อง และ Source code ดูไม่มีปัญหาอะไรมากเท่าโมเดลอื่น

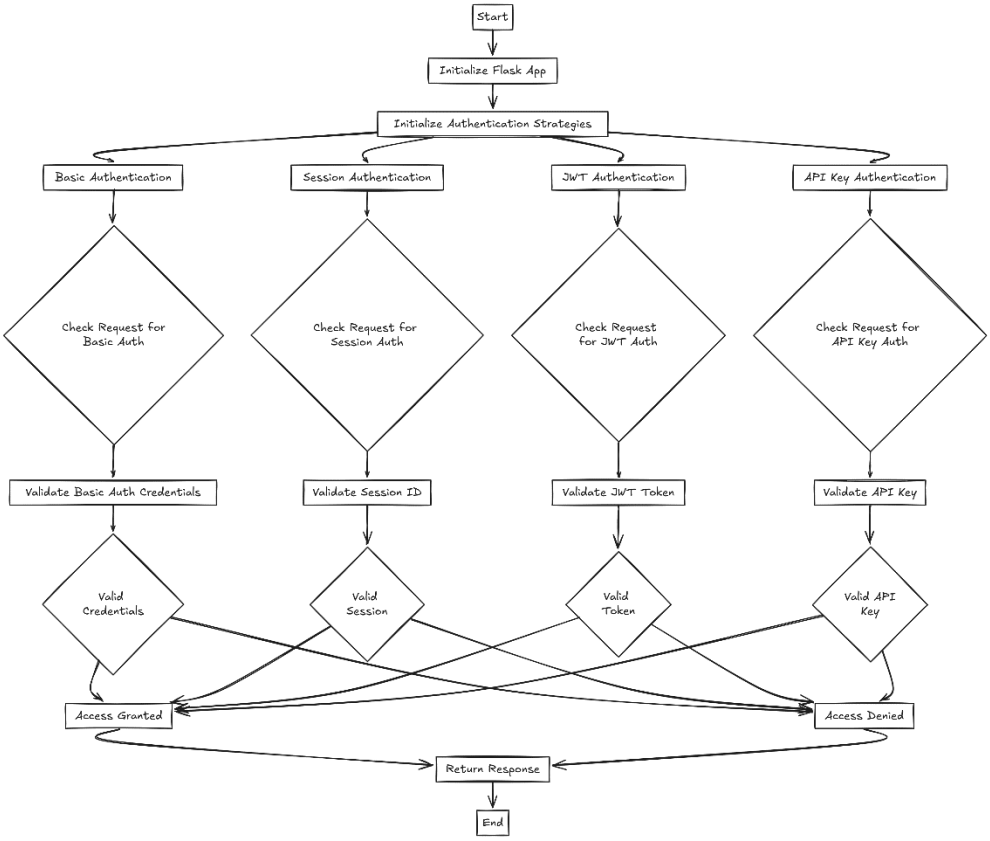
Python-Github Copilot-R1



Python-Github Copilot-R2



Python-Github Copilot-R3

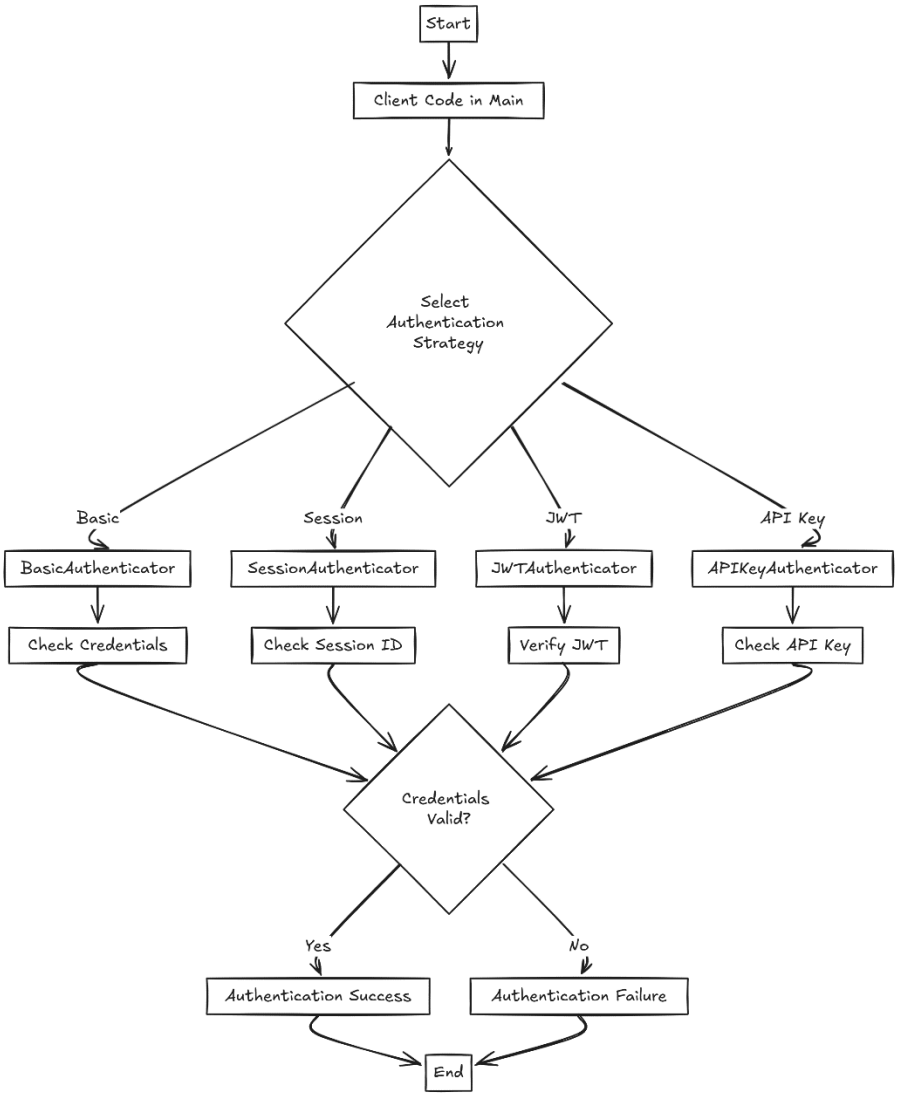


Java-ChatGPT

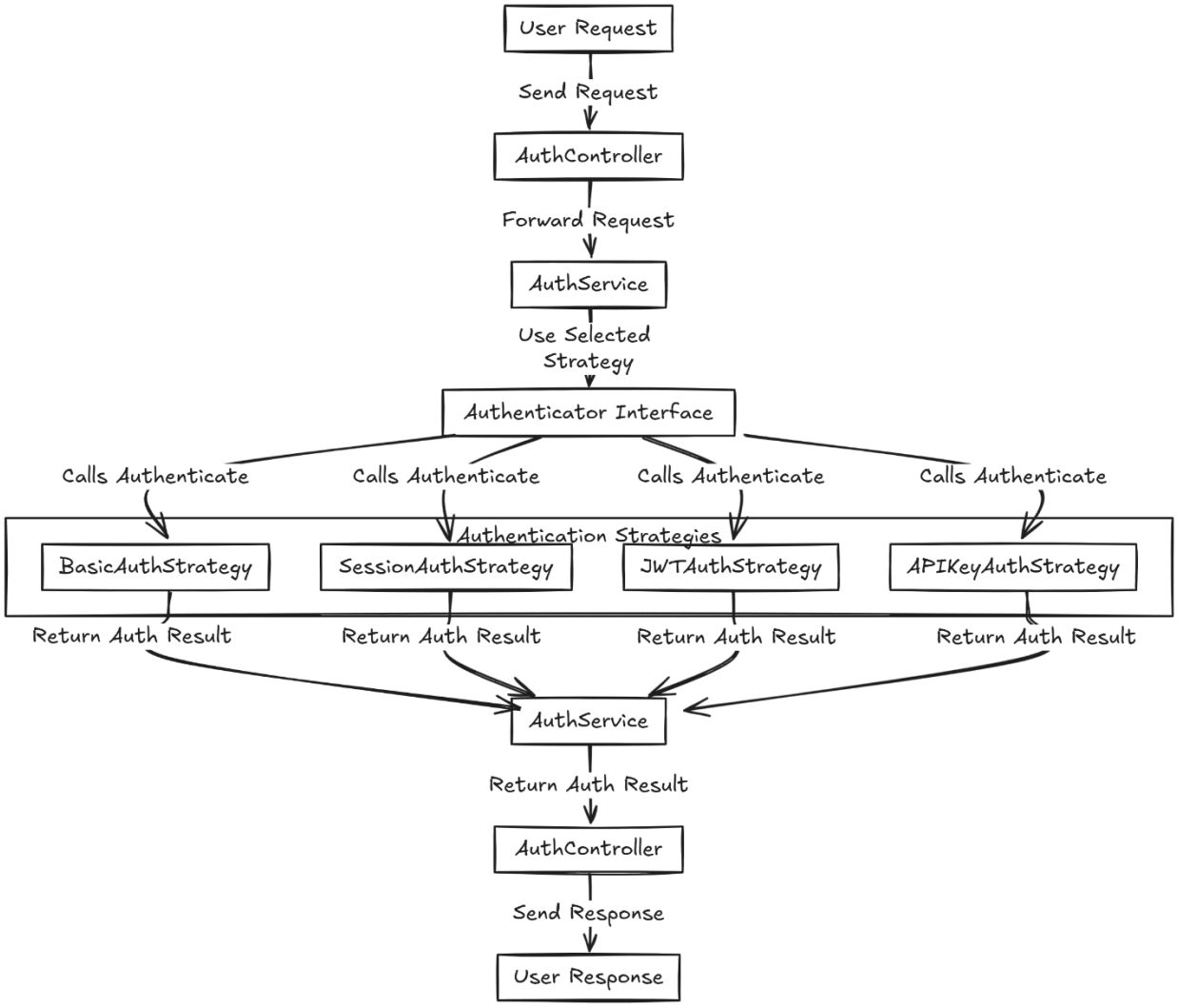
การ prompt เพื่อให้ AI สร้าง code Java นั้นต้องใช้การ prompt ที่รอบคอบกว่า Python เพราะ AI ไม่บอกโครงสร้างของโปรเจค และ Dependencies ที่โปรเจคใช้ ทำให้การสร้างโปรเจคมีปัญหา

จาก code ที่ได้สู่ CFG แม้ logic ของ AI จะถูกต้องเมื่อดูจาก CFG แต่ในทางกลับกัน source code นั้นมีปัญหาหนักมาก AI ได้เลือกใช้ dependencies หลายตัวที่ deprecated ไปแล้ว

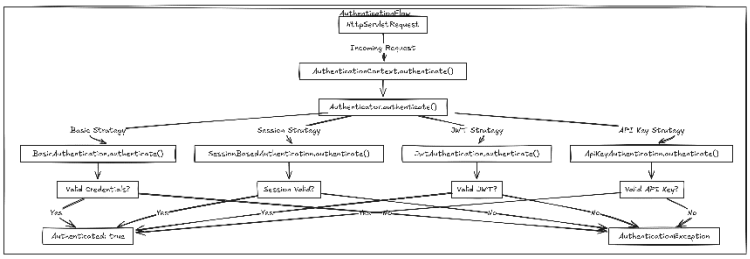
Java-ChatGPT-R1

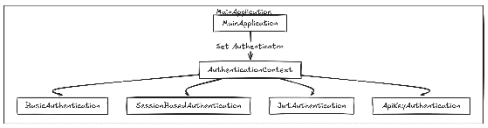


Java-ChatGPT-R2



Java-ChatGPT-R3

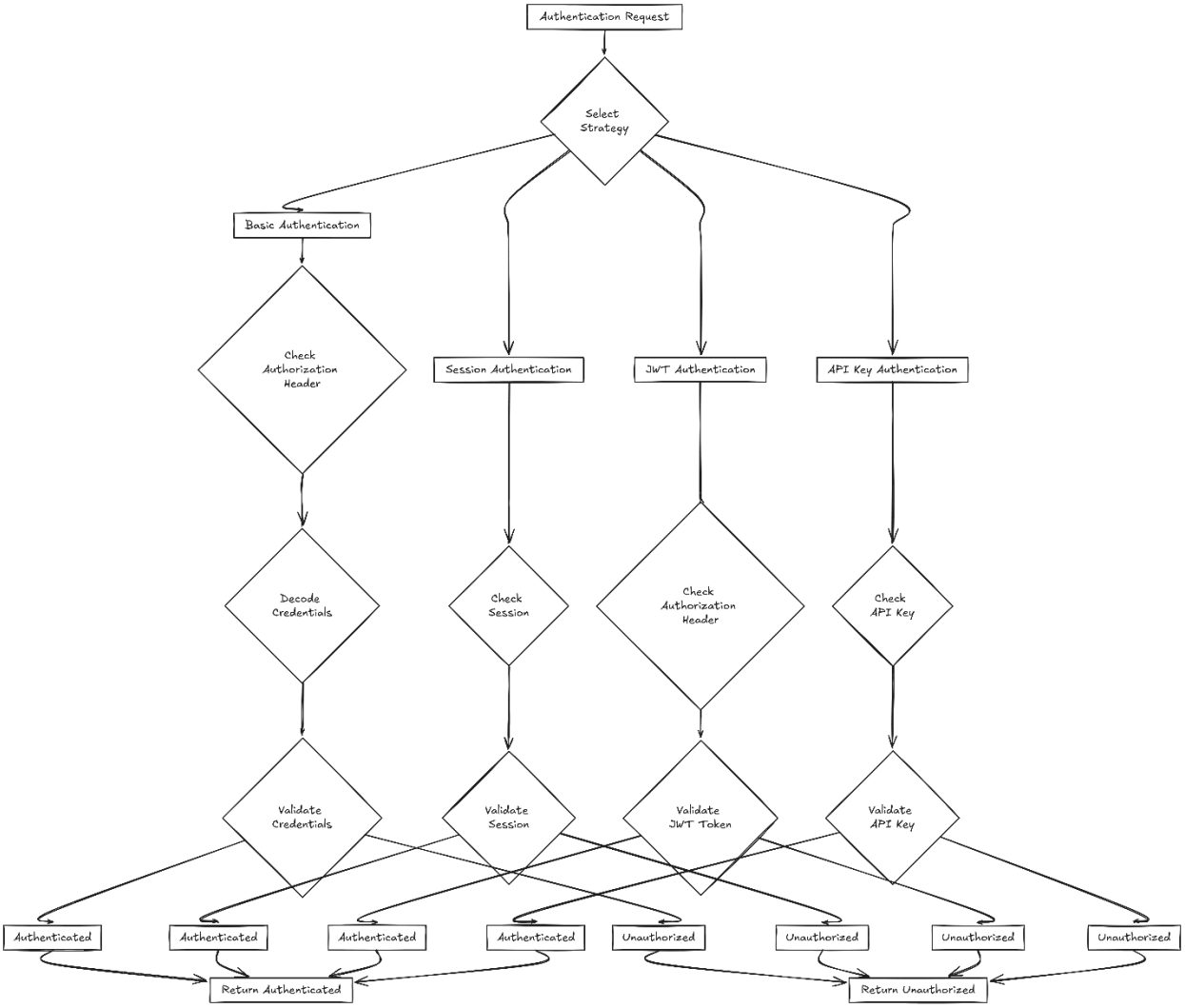




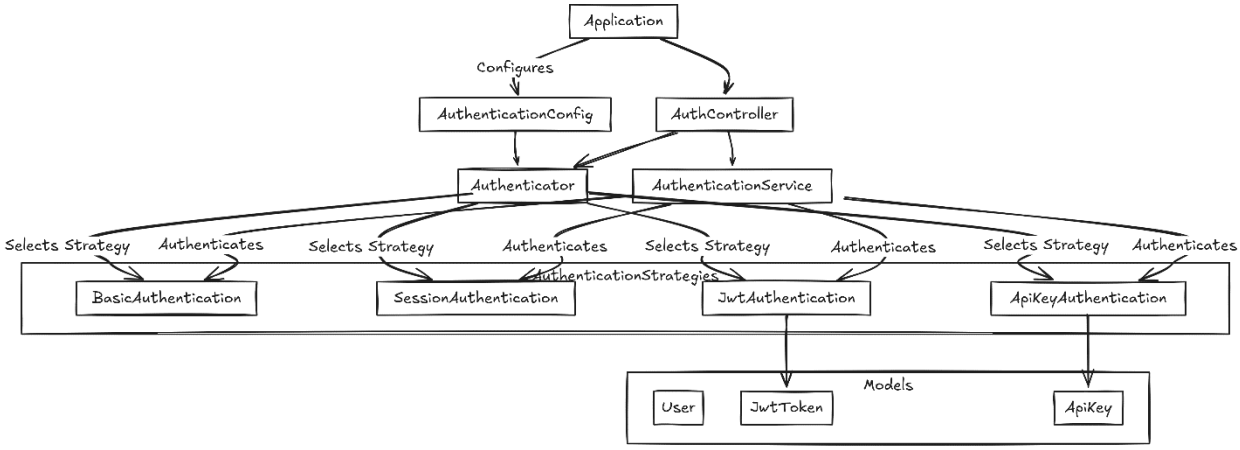
Java-Gemini Flash

เจอปัญหาคล้ายกับ ChatGPT ก็คือ Source code มีปัญหา มีการสร้างโค้ดที่ไม่เสร็จหรือไม่ครบในบางวิธี ทำให้ระบบอาจเกิดปัญหาในการทำงานได้

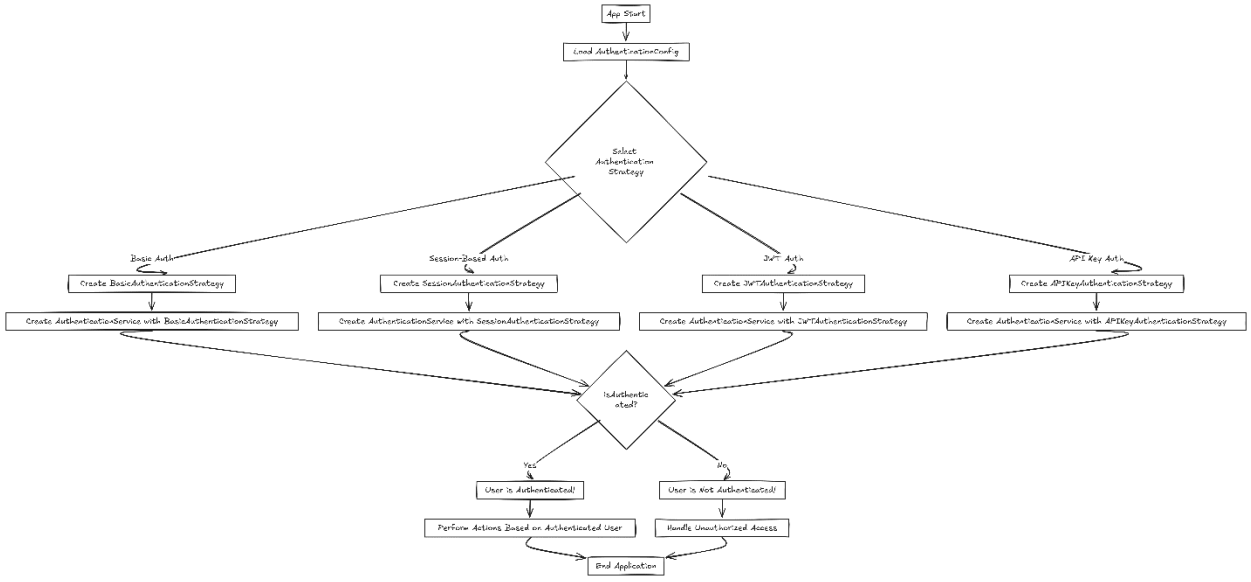
Java-Gemini Flash-R1



Java-Gemini Flash-R2



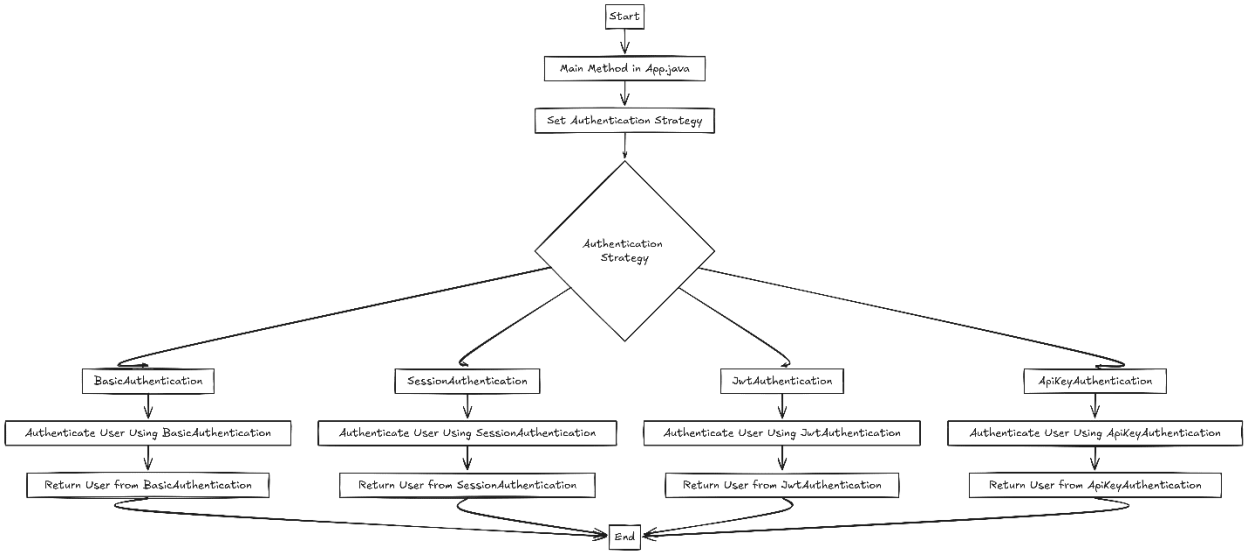
Java-Gemini Flash-R3



Java-Gemini Pro

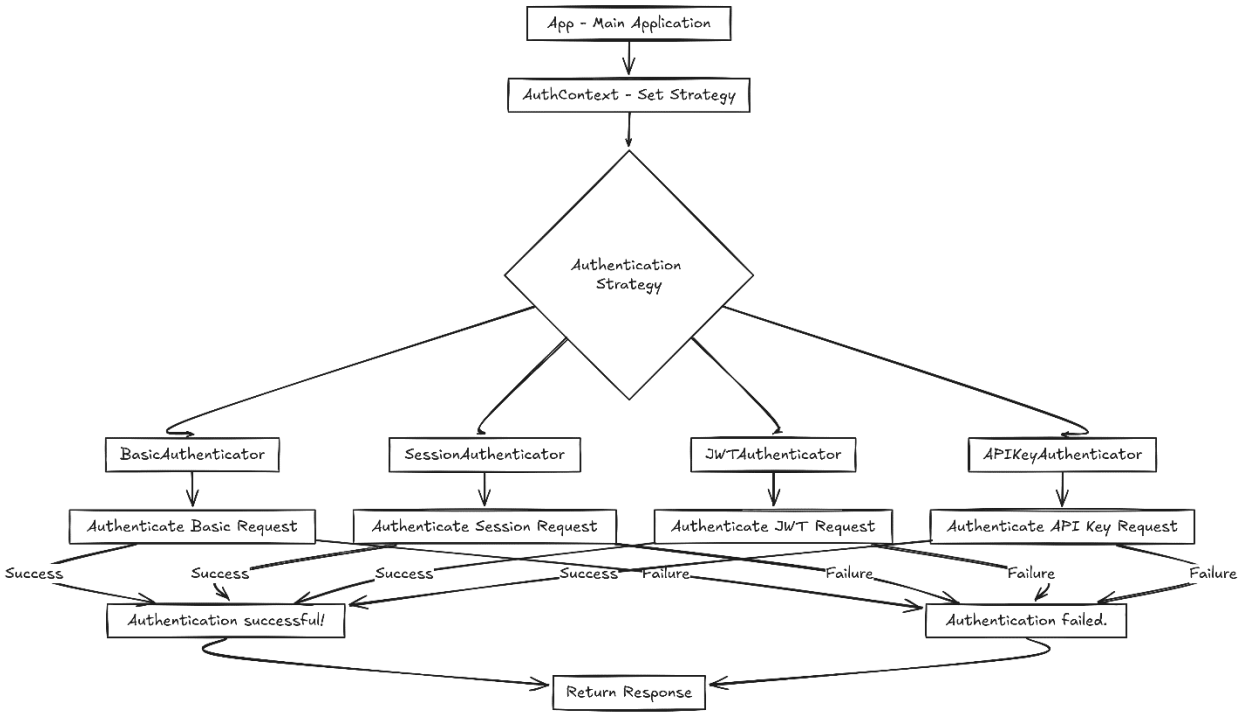
แม้ logic จะถูก แต่มีปัญหาคล้ายกับ Gemini Flash ที่มีการสร้าง code ขึ้นมาไม่ครบ และ lib ส่วนใหญ่ที่เลือกมาใช้มีการตรวจพบว่า version ที่ใช้นั้นเก่าเกินไป

Java-Gemini Pro-R1



Java-Gemini Pro-R2

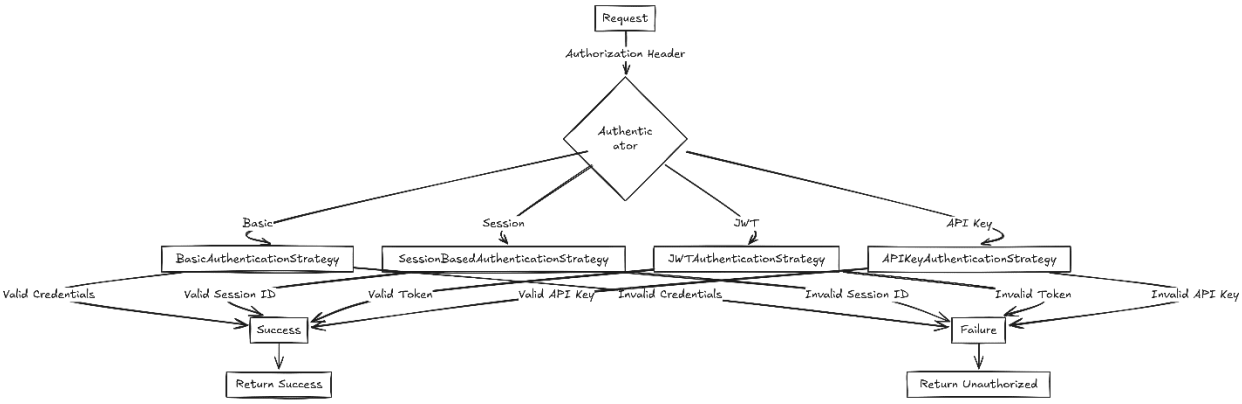
Java-Gemini Pro-R3



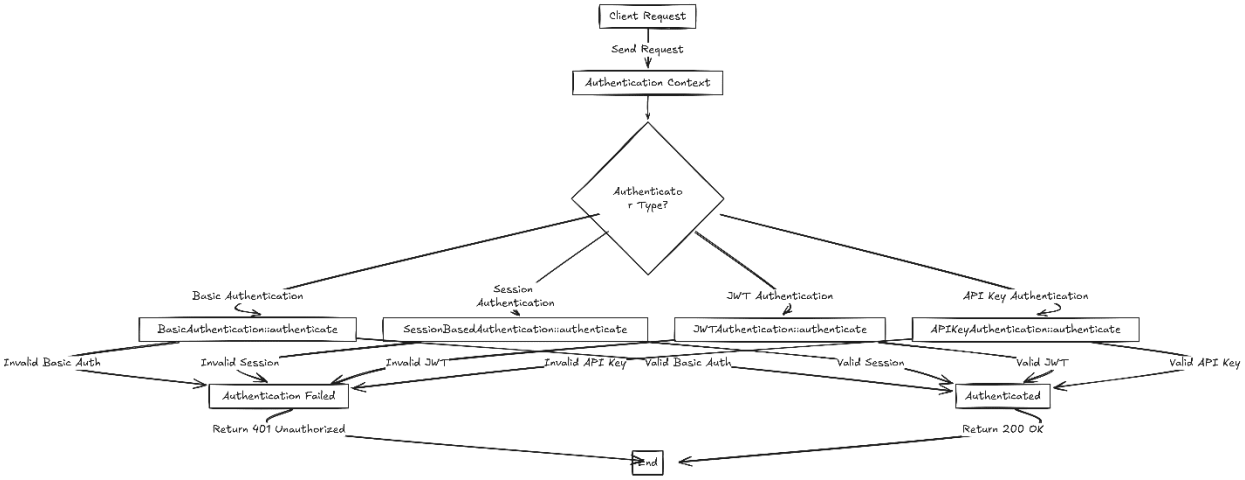
Java-Github Copilot

Logic ครบถ้วนแต่การ implemented code ยังมีปัญหา มีการเรียกใช้งาน class ที่ไม่มีในโปรเจค และ version ของ dependencies ที่เก่าเกินไป

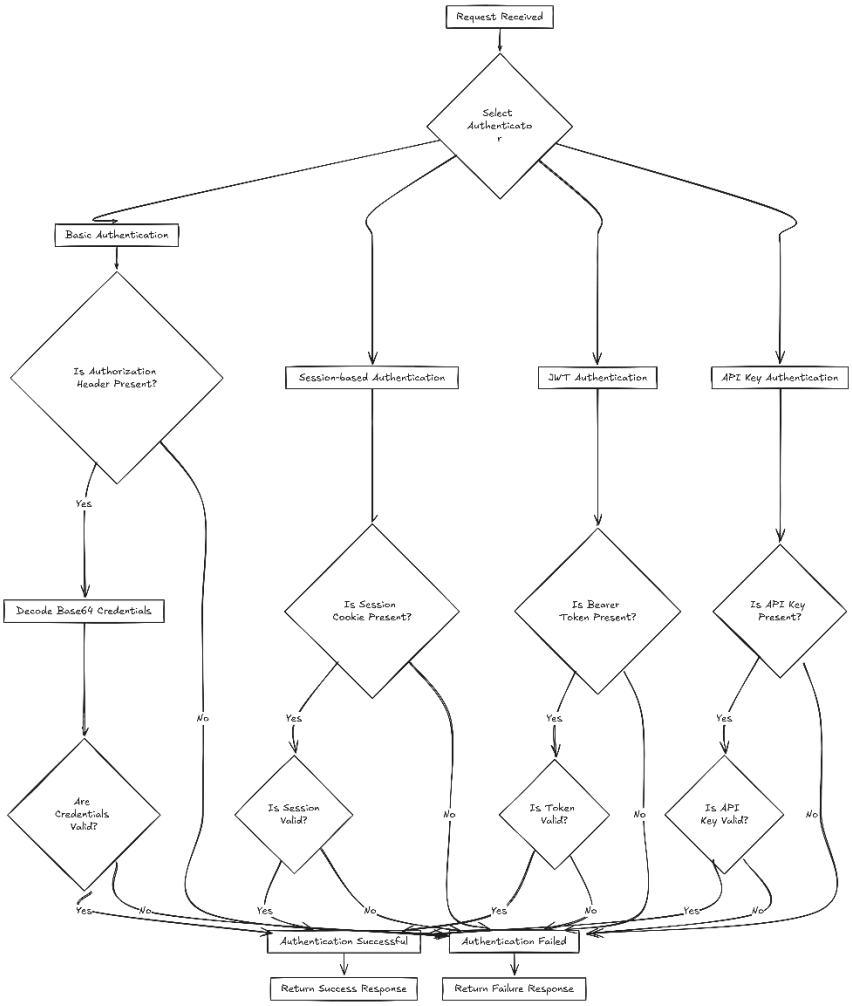
Java-Github Copilot-R1



Java-Github Copilot-R2



Java-Github Copilot-R3



สรุปผล

ในทั้ง Python และ Java โมเดลมีปัญหาในเรื่อง version ของ dependency และจัดการกับโค้ดที่ไม่สมบูรณ์ ซึ่งชี้ให้เห็นถึงจุดอ่อนที่พบร่วมกันใน AI Generative เมื่อต้องทำงานที่เกี่ยวกับความสมบูรณ์ของโปรเจคและการจัดการ dependency ประสิทธิภาพของ GitHub Copilot: จากการทดสอบโมเดลหลายๆ ตัว GitHub Copilot แสดงผลการทำงานที่ดีกว่าในโปรเจค Python และ Java โดยมีลอจิกที่สมบูรณ์และพบปัญหาสำคัญใน Code น้อยกว่า แต่ละภาษา (Java vs. Python) ต้องการความละเอียดของคำสั่งที่แตกต่างกันเพื่อให้ได้ผลลัพธ์ที่น่าพอใจ โดยเฉพาะใน Java ที่ต้องการคำสั่งที่มีรายละเอียดมากกว่า