Satvi Mahesh
Mentor - Rebecca Funke
4 May 2025

Abstract

"*Artificial Intelligence is demolishing our creativity.*"
These words have been uttered by nearly every artist today. People use AI to create art, but not in the ethical way that it should be used, that is, the art generated by the AI is taken and claimed as their original work. This is often the result of a lack of creativity, but it can also cause creativity to reduce as well. So, instead of using AI in a way that hinders the creative process, I wanted to use it in a way that promotes the process.

Problem + Solution

As a self-taught artist who improves my artistic abilities daily through tutorials on YouTube, TikTok, Instagram, etc, I find myself in the final stages of my artwork wondering, "Something is off about my drawing, but I can't tell what." This is a challenge that I, and many other young amateur artists, face because how is a video supposed to give specific advice on what and how to fix something? How is the video supposed to tailor the tips and improvements to my drawing? The simple answer– it can't.

My app Art Intel combines my passion for art and my interest in artificial intelligence by utilizing generative AI to serve as an ally in the creative process to solve that problem. Rather than the AI giving back an image with the fixes made, making it easy for the artist to steal it and claim it as their own, Art Intel will be by your side to guide you down the path from good art to great art, without taking away creative liberties. In addition, someone wanting to learn art might not have the available resources or financial status to be able to afford an art tutor or enroll in a class, as those are not cheap. So, Art Intel also takes away that cost factor by being a readily available, free-to-use "teacher." The artist can simply upload an image of the drawing directly from the camera roll and send it, and wait as the feedback appears on the screen.

The main goal that I knew from the beginning that I wanted my app to achieve was instilling confidence in growing, self-taught artists, while providing friendly feedback on not only *what* can be improved in a drawing, but also *how* it can be improved. As a still-learning artist myself, I am aware that there are many other apps out there to help with similar problems, but they don't answer both questions and only tell me what to fix, leaving it up to me to figure out how to fix it. However, someone who is learning art from the beginning wouldn't even know where to start, so Art Intel walks the artist through several pointers on what works well in the drawing (the encouragement and confidence-building piece) and what can be improved and how to do so (the feedback piece).

Choosing the AI to Run in the Backend

After establishing the problem my app would solve, I developed my MVP, or minimum viable product, where my app's main focus would be on providing feedback for digital portrait drawings. To do this, I knew that I wouldn't be using traditional datasets, and instead, I'd need to use generative AI. There are so many out there, such as ChatGPT, Gemini, Claude AI, Meta, etc, but they all respond in different ways. To begin, I first tested these four AI platforms to see which one would generate the results closest to what I was looking for. To do this, I prompted each AI with the same prompt and an image of one of my drawings, and then, using prompt engineering, I kept digging deeper and asking more questions, getting more specific each time to get the results I wanted. I quickly learned that Gemini did not accept images of faces, Meta was not specific enough, and ChatGPT and Claude AI both had compatible responses. I ended up choosing Claude AI in the end because ChatGPT only allowed three images to be submitted before it stops you and asks you to upgrade to premium for unlimited images, but Claude had no restrictions as far as I tested.

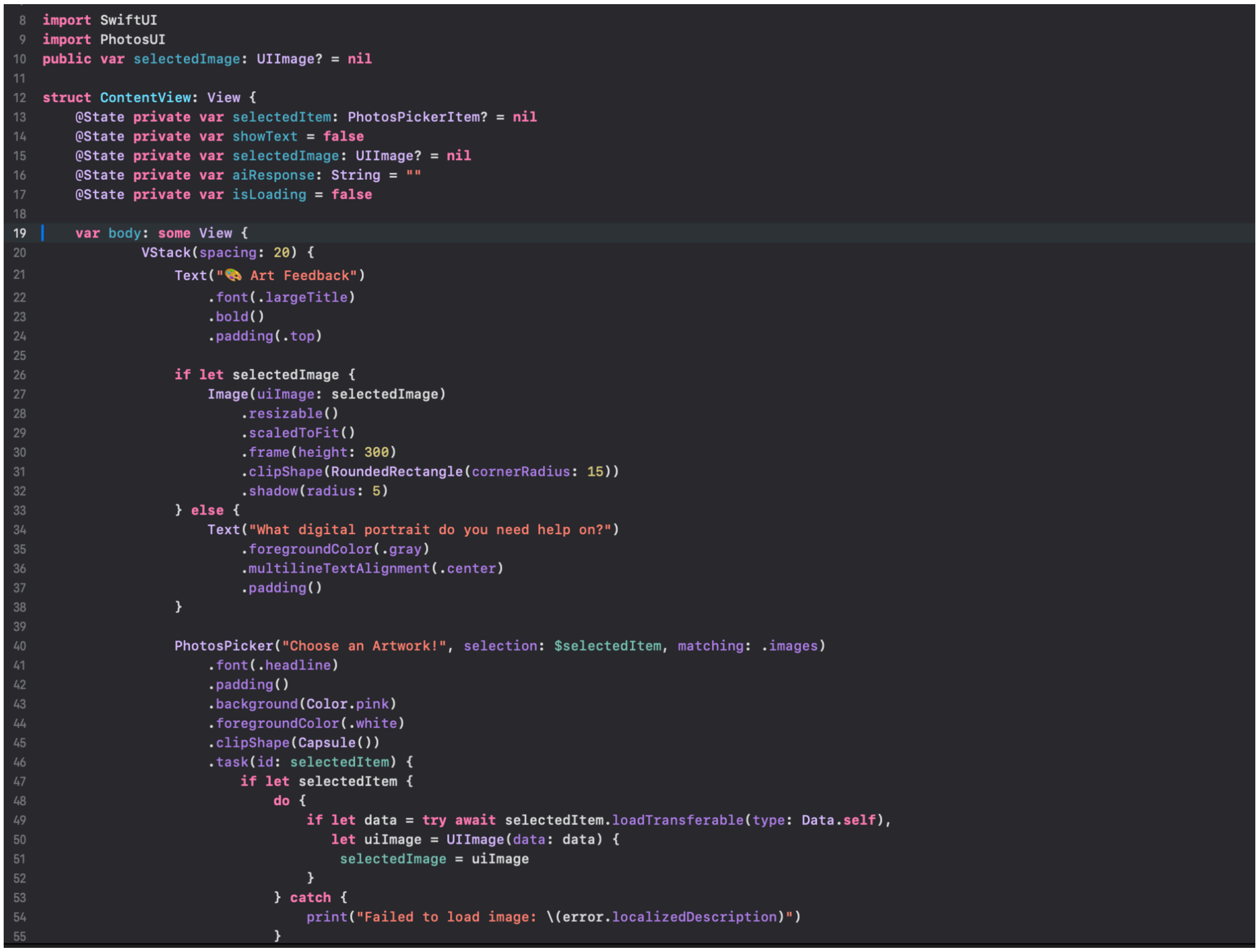
API and API Key

API stands for "Application Programming Interface" and is like a messenger to help request services and connect two applications to each other. Claude API is the API my app utilizes, and the way it works is that everytime I send an image to my app, a request is made to the Anthropic services using the tokens payed for (tokens = input and response = output).

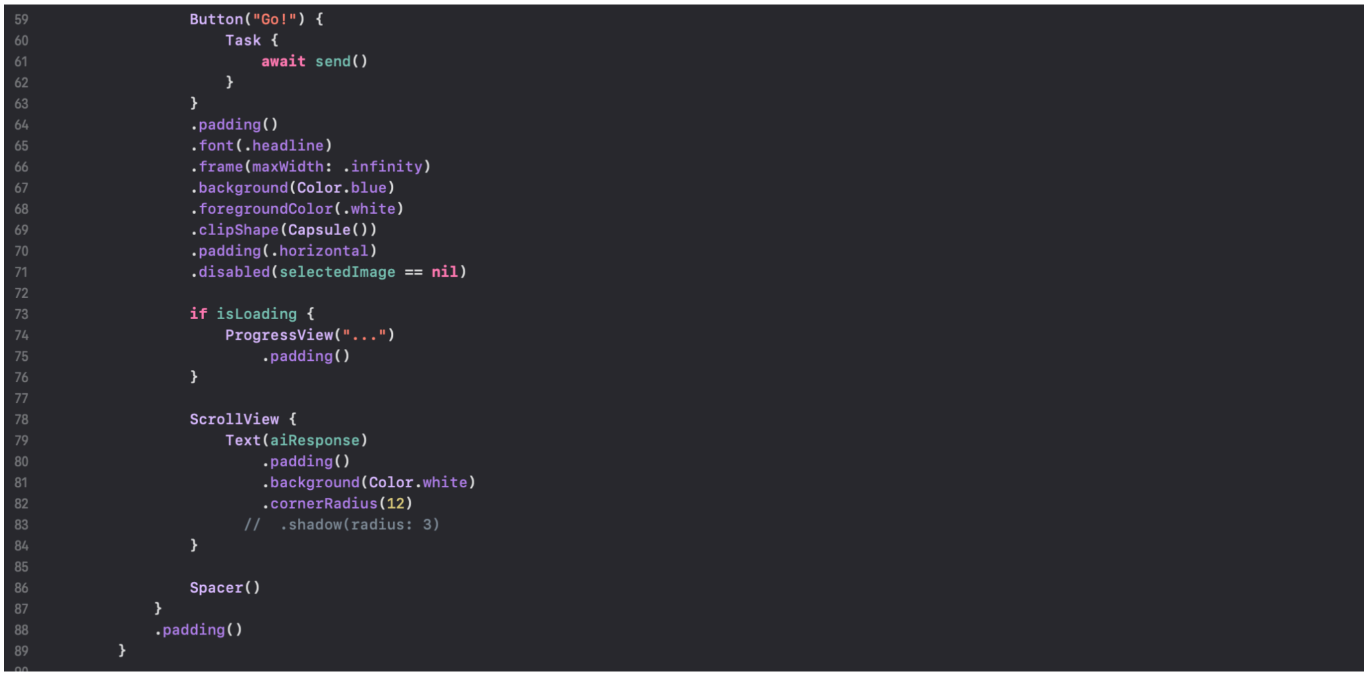
For the API calls to be able to work successfully, I needed to purchase an API key from Claude, primarily for authentication and authorization. The API key helps verify that I have the credentials and am verified to use the API services. If I did have the API key, calls and access to the Claude API services would be unauthorized and vulnerable to security issues.

The Code Itself (view on next page)

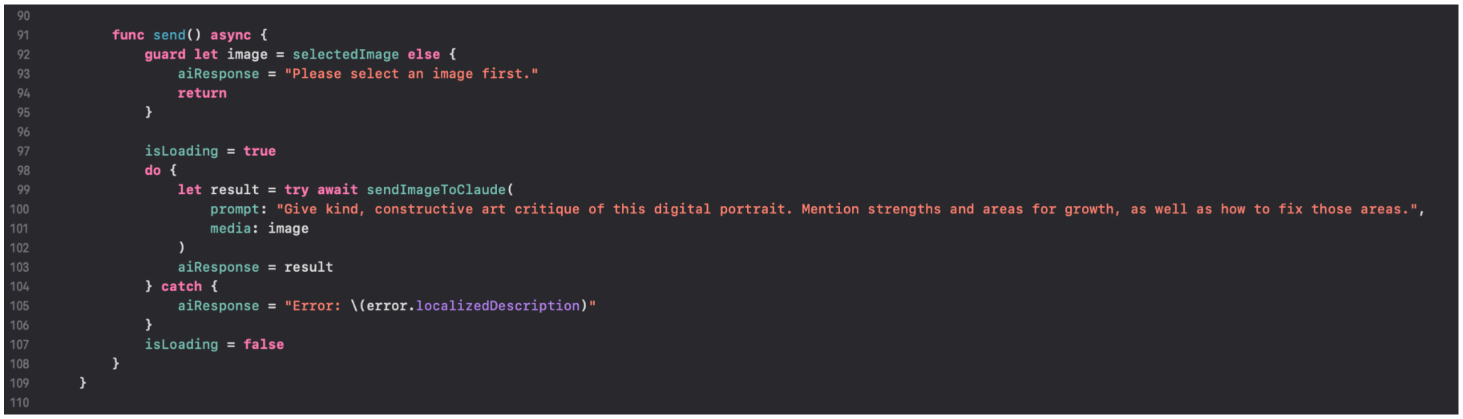
- **Screenshots 1-3: Content View**
- **Screenshots 4-5: API caller**



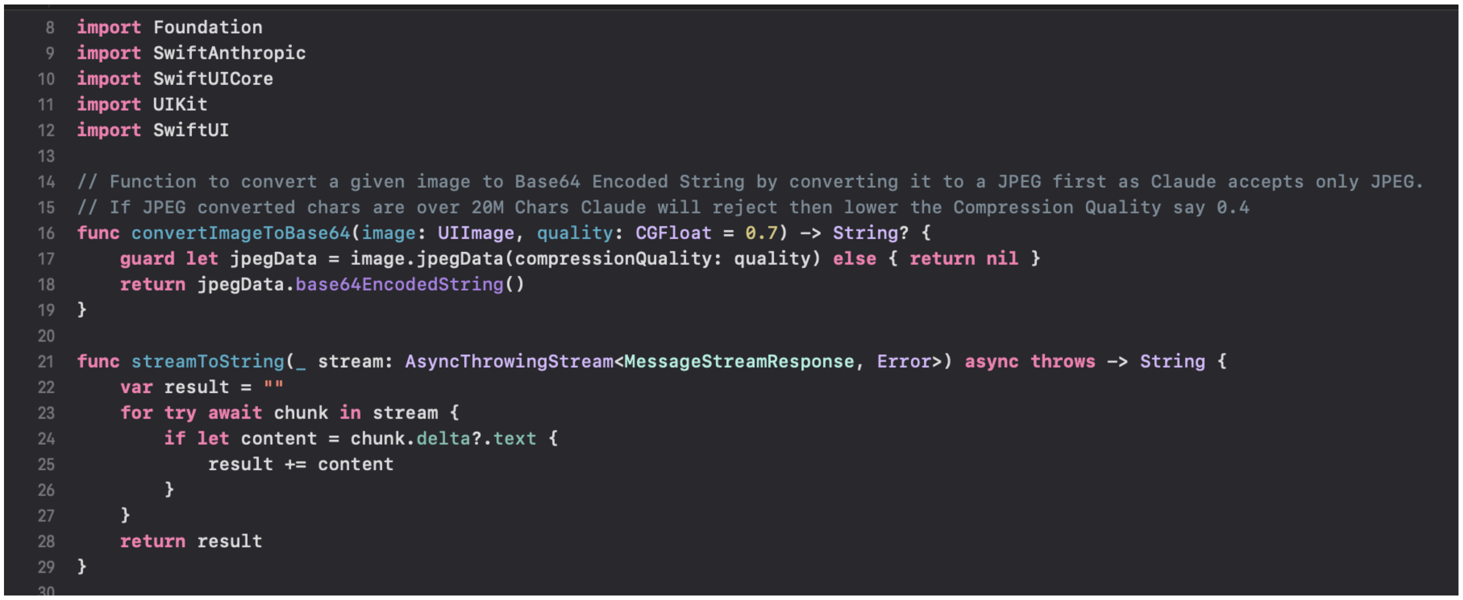
```swift
import SwiftUI
import PhotosUI
public var selectedImage: UIImage? = nil

struct ContentView: View {
    @State private var selectedItem: PhotosPickerItem? = nil
    @State private var showText = false
    @State private var selectedImage: UIImage? = nil
    @State private var aiResponse: String = ""
    @State private var isLoading = false

    var body: some View {
            VStack(spacing: 20) {
                Text("🎨 Art Feedback")
                    .font(.largeTitle)
                    .bold()
                    .padding(.top)

                if let selectedImage {
                    Image(uiImage: selectedImage)
                        .resizable()
                        .scaledToFit()
                        .frame(height: 300)
                        .clipShape(RoundedRectangle(cornerRadius: 15))
                        .shadow(radius: 5)
                } else {
                    Text("What digital portrait do you need help on?")
                        .foregroundColor(.gray)
                        .multilineTextAlignment(.center)
                        .padding()
                }

                PhotosPicker("Choose an Artwork!", selection: $selectedItem, matching: .images)
                    .font(.headline)
                    .padding()
                    .background(Color.pink)
                    .foregroundColor(.white)
                    .clipShape(Capsule())
                    .task(id: selectedItem) {
                        if let selectedItem {
                            do {
                                if let data = try await selectedItem.loadTransferable(type: Data.self),
                                   let uiImage = UIImage(data: data) {
                                    selectedImage = uiImage
                                }
                            } catch {
                                print("Failed to load image: \(error.localizedDescription)")
                            }
```



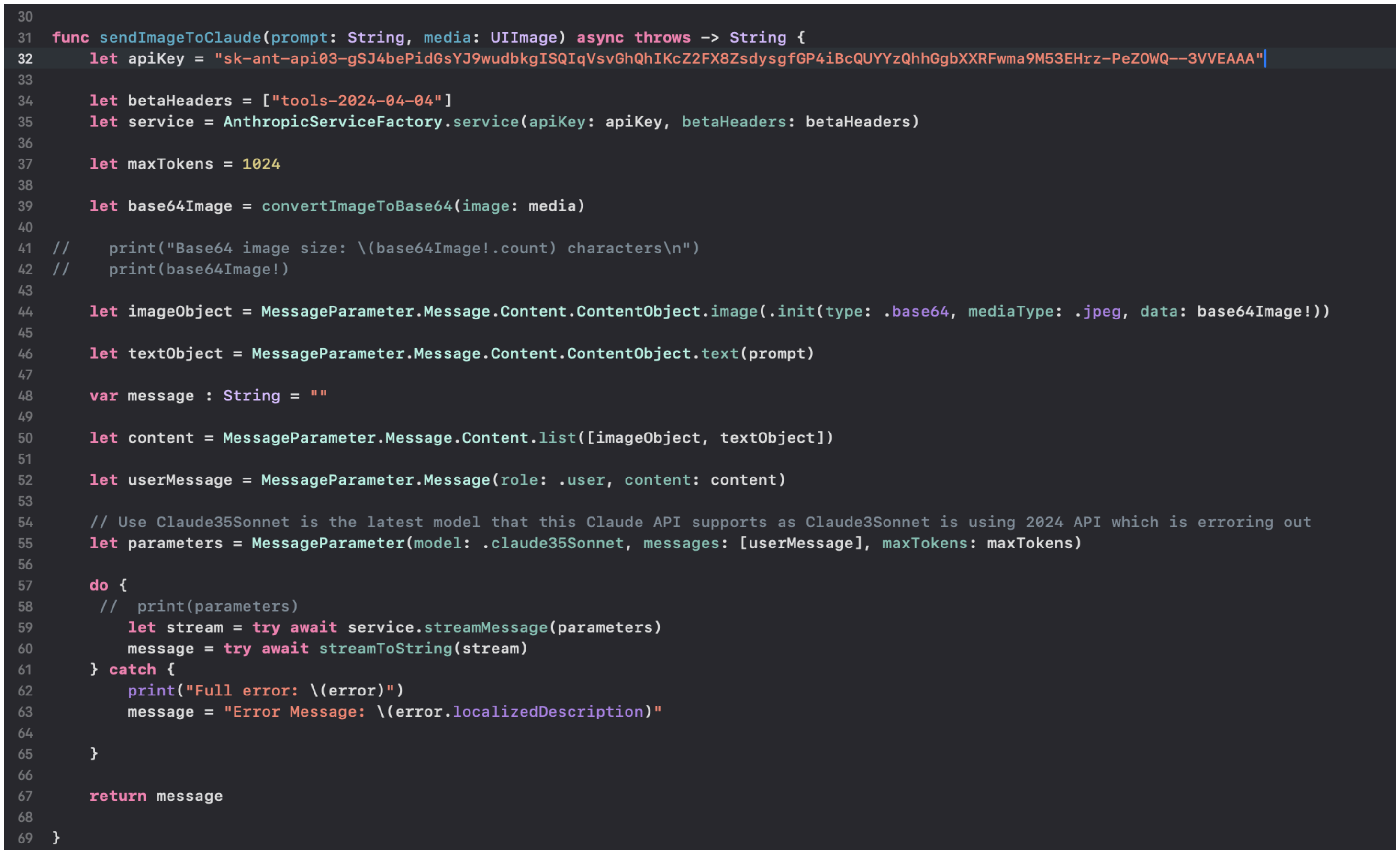
```swift
                Button("Go!") {
                    Task {
                        await send()
                    }
                }
                .padding()
                .font(.headline)
                .frame(maxWidth: .infinity)
                .background(Color.blue)
                .foregroundColor(.white)
                .clipShape(Capsule())
                .padding(.horizontal)
                .disabled(selectedImage == nil)

                if isLoading {
                    ProgressView("...")
                        .padding()
                }

                ScrollView {
                    Text(aiResponse)
                        .padding()
                        .background(Color.white)
                        .cornerRadius(12)
                      //  .shadow(radius: 3)
                }

                Spacer()
            }
            .padding()
        }
```



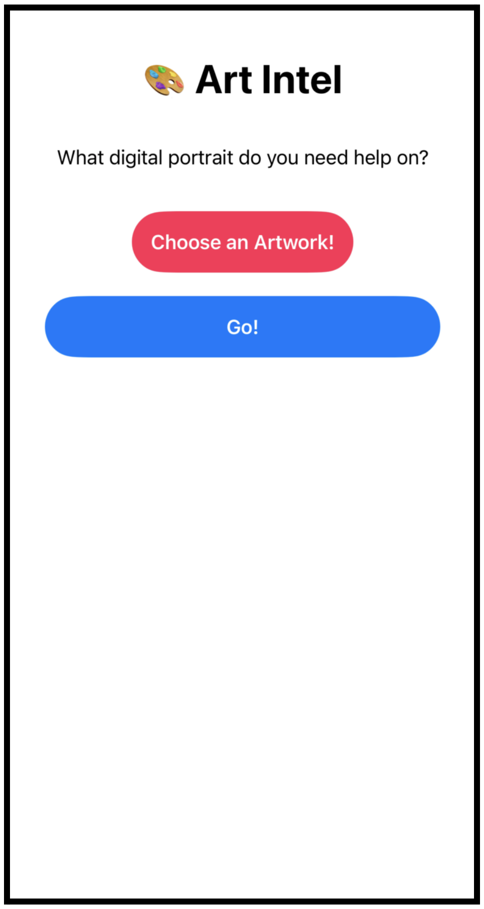
```swift

    func send() async {
        guard let image = selectedImage else {
            aiResponse = "Please select an image first."
            return
        }

        isLoading = true
        do {
            let result = try await sendImageToClaude(
                prompt: "Give kind, constructive art critique of this digital portrait. Mention strengths and areas for growth, as well as how to fix those areas.",
                media: image
            )
            aiResponse = result
        } catch {
            aiResponse = "Error: \(error.localizedDescription)"
        }
        isLoading = false
    }
}

```



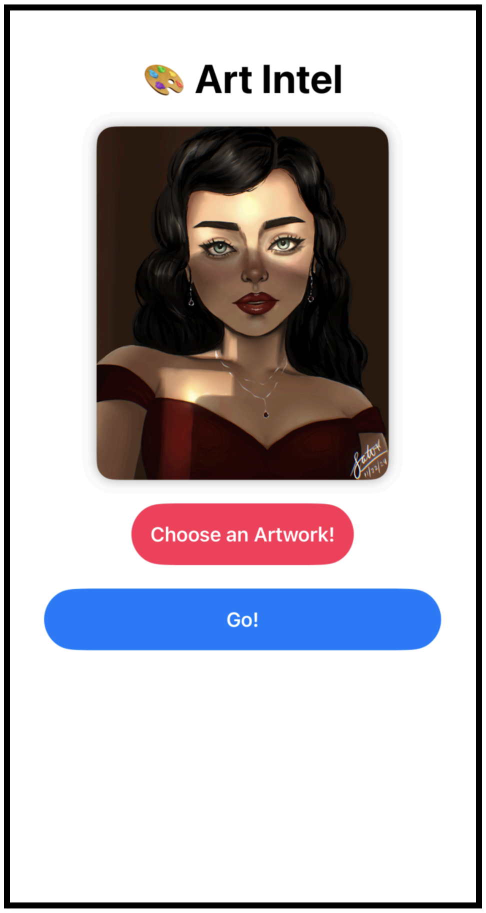
```swift
import Foundation
import SwiftAnthropic
import SwiftUICore
import UIKit
import SwiftUI

// Function to convert a given image to Base64 Encoded String by converting it to a JPEG first as Claude accepts only JPEG.
// If JPEG converted chars are over 20M Chars Claude will reject then lower the Compression Quality say 0.4
func convertImageToBase64(image: UIImage, quality: CGFloat = 0.7) -> String? {
    guard let jpegData = image.jpegData(compressionQuality: quality) else { return nil }
    return jpegData.base64EncodedString()
}

func streamToString(_ stream: AsyncThrowingStream<MessageStreamResponse, Error>) async throws -> String {
    var result = ""
    for try await chunk in stream {
        if let content = chunk.delta?.text {
            result += content
        }
    }
    return result
}

```

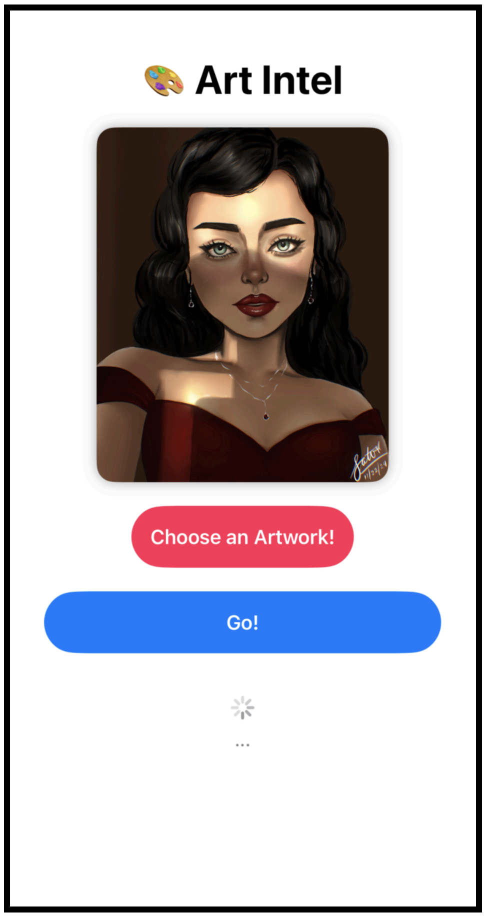


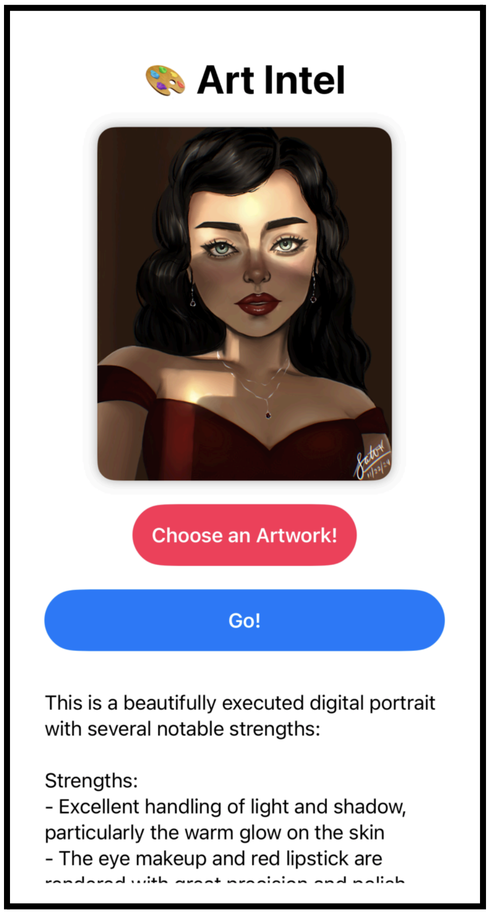
```swift

func sendImageToClaude(prompt: String, media: UIImage) async throws -> String {
    let apiKey = "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"

    let betaHeaders = ["tools-2024-04-04"]
    let service = AnthropicServiceFactory.service(apiKey: apiKey, betaHeaders: betaHeaders)

    let maxTokens = 1024

    let base64Image = convertImageToBase64(image: media)

//    print("Base64 image size: \(base64Image!.count) characters\n")
//    print(base64Image!)

    let imageObject = MessageParameter.Message.Content.ContentObject.image(.init(type: .base64, mediaType: .jpeg, data: base64Image!))

    let textObject = MessageParameter.Message.Content.ContentObject.text(prompt)

    var message : String = ""

    let content = MessageParameter.Message.Content.list([imageObject, textObject])

    let userMessage = MessageParameter.Message(role: .user, content: content)

    // Use Claude35Sonnet is the latest model that this Claude API supports as Claude3Sonnet is using 2024 API which is erroring out
    let parameters = MessageParameter(model: .claude35Sonnet, messages: [userMessage], maxTokens: maxTokens)

    do {
     //  print(parameters)
        let stream = try await service.streamMessage(parameters)
        message = try await streamToString(stream)
    } catch {
        print("Full error: \(error)")
        message = "Error Message: \(error.localizedDescription)"

    }

    return message

}
```



What digital portrait do you need help on?

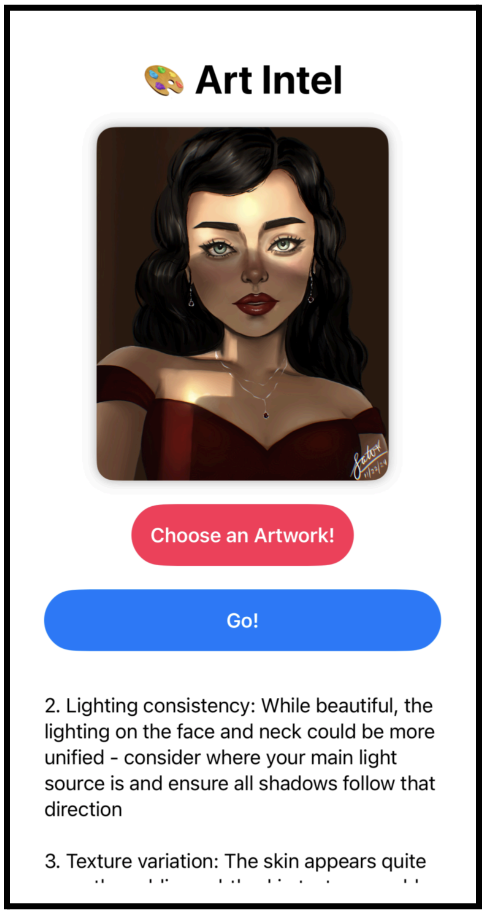






This is a beautifully executed digital portrait with several notable strengths:

Strengths:
- Excellent handling of light and shadow, particularly the warm glow on the skin
- The eye makeup and red lipstick are rendered with great precision and polish



2. Lighting consistency: While beautiful, the lighting on the face and neck could be more unified - consider where your main light source is and ensure all shadows follow that direction

3. Texture variation: The skin appears quite

Challenges Along the Way

During the coding process, there was difficulty making the actual API calls and getting back responses from Claude. Through cURL commands, I tested to see if my API key was even working, and I ended up figuring out that there was an issue with the key itself. I realized that some characters of the key might have gotten lost through the process of copying and pasting and creating different versions of the project, so I got a new API key, and that ended up fixing that problem.

In testing the API key, it was also revealed that the 2024 version of Claude (Claude Sonnet 3) that I was running was not compatible with my API key, so I switched to the 2025 version (Claude Sonnet 3-5).

One of the necessary functions in the API caller class was a function that converted the image to base64 to make it easier to send information as an image. To make sure this method was working, I used: https://base64.guru/converter/decode/image**,** which helped to debug and fix that function so it would work as intended.

Final Thoughts

The process to the development of this app was an enlightening and humbling experience. I went into this project thinking it wouldn't be challenging, and it was something I could figure out in just a few hours. I was proven wrong when I ran into several issues and bugs along the way, but through the help of extensive outside research and help from my mentor, I was able to overcome the challenges. This project was my first ever experience with Swift, and was definitely a fun one to tackle. Art Intel combines my interest in AI with my passion for art and benefits the entire art community, mainly the new aspiring artists who need guidance from a beginner-friendly app.

Anyone can benefit from this app– the kind feedback and supporting words on what looks good in a drawing can help unleash creativity, causing it so soar to new heights, and take drawing and creative skills to a new, never-before seen level. Art is a broad concept and it can be anything that takes creativity and expression. Do not be afraid to use AI for you creative endeavors– it will soon become your best ally in the world of art.