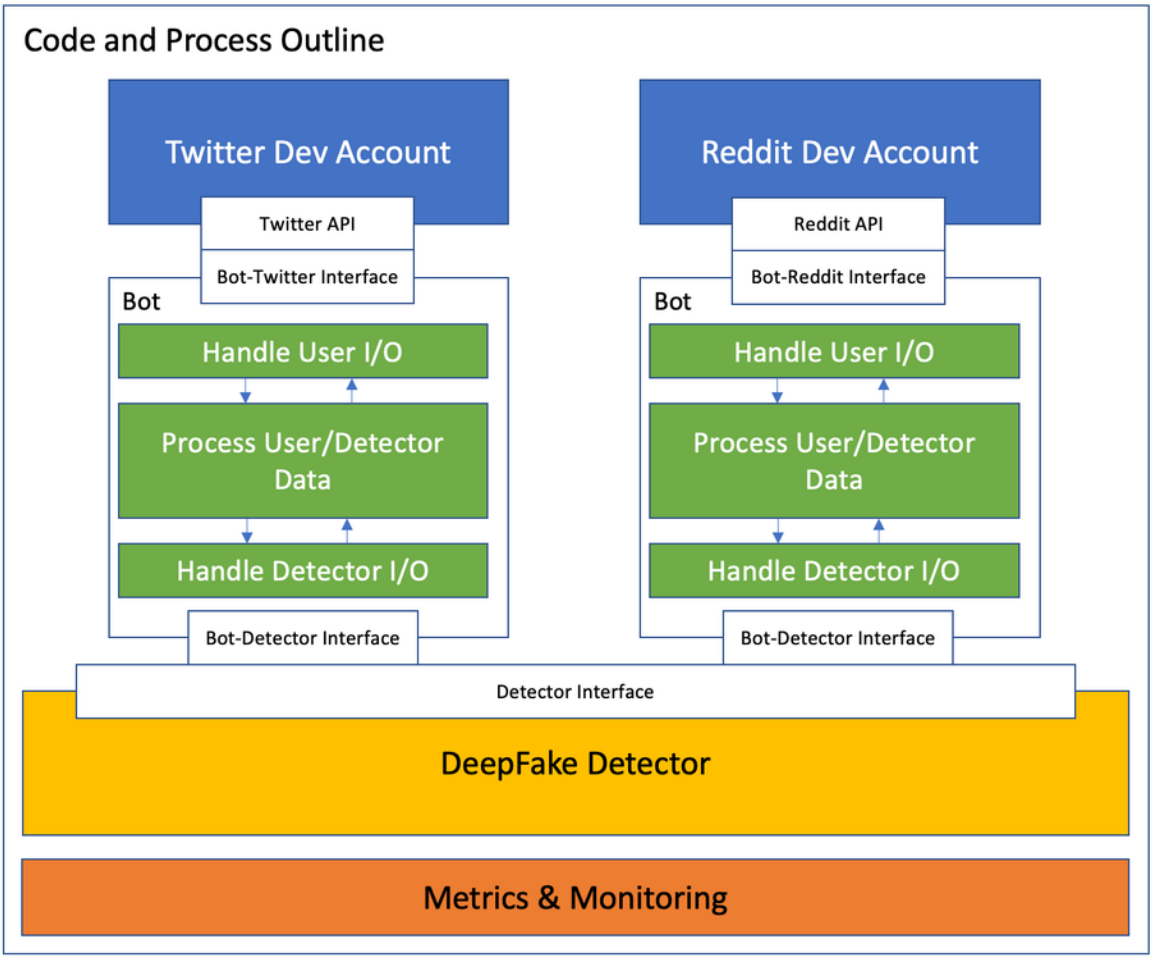
Code Plan

Code Plan Diagram



This diagram is explained in depth in the dialog below

Twitter

- We should have a folder in our GitHub repo for the Twitter Bot code which we will write in Python
- This code will be responsible for interfacing with the Twitter API and DeepFake Detector API, as well as, processing the data received by each interface
- We can interface our code with the Twitter API via our Developer Account (contact Patrick for access details, not smart to store those secrets here)
- We can interface our code with the DeepFake Detector API by sending processed user input data to a standard endpoint and then waiting for the

Reddit

- We should have a folder in our GitHub repo for the Reddit Bot code which we will write in Python
- This code will be responsible for interfacing with the Reddit API and DeepFake Detector API, as well as, processing the data received by each interface
- We can interface our code with the Reddit API via our App Script Integration linked to our Bot Account (contact Patrick for access details, not smart to store those secrets here)
- We can interface our code with the DeepFake Detector API by sending processed user input data to a standard

Facebook/Instagram

- We should have one folder each in our GitHub repo for the Facebook and Instagram Bot code which we will write in Python
- This code will be responsible for interfacing with the Meta APIs and DeepFake Detector API, as well as, processing the data received by each interface
- We can interface our code with the Reddit API via our Developer Account (contact Patrick for access details, not smart to store those secrets here)
- We can interface our code with the DeepFake Detector API by sending processed user input data to a standard

standard endpoint to return the analysis output	endpoint and then waiting for the standard endpoint to return the analysis output	endpoint and then waiting for the standard endpoint to return the analysis output

Broadly, the bot code should:

- Idle until it receives a user interaction
- Determine if the user input is actionable (if not, send standard “I don’t understand your request” message)
- If actionable, extract and pre-process video and other data for Detector
- Pass the pre-processed data to the detector and wait for analysis output
- Package analysis output into a user friendly response (i.e. “The video you gave me is ##% likely to be fake”)
- Respond to the user and return to idle state

DeepFake Detector

- We should have a folder in our GitHub repo for the open source model(s) and code we use and the Detector API we write
- This code will handle interfacing with our bots as well as processing the video data from the bots

Metrics

- The metrics layer could likely be implemented in a low/no-code system via some integrated tool like CloudWatch (depends on the host provider we use)