HACKFEST

Hosted by ICe and t-hub

The Swiss knife

The multi-tool for anomaly detection

Team: Appsack

Team code: IC8

Date: 21st march- 22nd March 22, 2023

Location: t-hub, Hyderabad

Index

1. Problem statement and our understanding ……………………3
2. Our approach to the solution…………………………………..3
3. Technologies used……………………………………………..4

* Front-End
* Back-End

1. Functionalities provided………………………………………6
2. Additional code snippet……………………………………….8
3. Resulting Screens…………………………………………….11

1. Future scope………………………………………………….12
2. Conclusion……………………………………………………13
3. References……………………………………………………14

Problem statement

This hack-a-thon aims to bring together developers, data scientists, and coding enthusiasts to build innovative solutions for detecting anomalies in source code repositories.

During this hack-a-thon, participants will have the opportunity to work on a variety of projects related to anomaly detection in source code repositories. This could include building machine learning models to identify patterns of abnormal behavior, developing algorithms to detect unusual code changes or bugs, or creating visualization tools to help developers quickly identify potential anomalies in code.

· Code Review Bot: Develop a machine learning model that can identify anomalies in code reviews, such as unusual patterns of comments or code changes, and flag them for further investigation.

· Bug Detector: Build an algorithm that can detect anomalous code changes that are likely to result in bugs or errors in the software.

·Code Clustering: Develop a clustering algorithm that can group code files or functions based on their similarity and detect any anomalies or outliers that do not fit into the expected clusters.

· Code Visualization Tool: Create a visualization tool that can help developers identify potential anomalies in code by highlighting patterns of unusual behavior or code changes.

· Code Authorship Identification: Build a machine learning model that can identify the author of a code change based on their coding style and identify any anomalies in their code compared to their usual coding patterns.

· Security Vulnerability Detection: Develop an algorithm that can identify potential security vulnerabilities in code by detecting patterns of unusual behavior or code changes that may be exploitable.

· Code Complexity Analyzer: Build a tool that can analyze the complexity of code changes and identify anomalies that may lead to performance issues or bugs.

· Code Duplicity Detector: Develop a machine learning model that can identify duplicate code in source code repositories and flag any anomalies or outliers that do not fit into the expected patterns.

Our understanding and approach

Solution provided to t-hub and ICe on the event “Hackfest” conducted on 21st of march-22nd of march at t-hub Hyderabad. The problem statement of the Hackathon is “Decoding complex information security challenges” which is mainly focused on the “anomaly detection for source code repositories ”.

We have decide to go with developing an app that not only can have one feature to help find anomaly in source code but multiple iterative “weapons” or tools that will help in find specific anomalies about the source code and also give a brief description for it.

We have planned to perform the following functionalities:

1. Code complexity detection
2. Function call visualization
3. Detection of malicious git commits

For the front-end we have used the flutter framework which is known for its agile properties like hot Reload and single code for different platforms.

The flask API is fetched through localhost, port 5000 by default that can be later shifted to an online sever.

Technology used

Back-End

ML:

A GPT 2 model is used to train the java code and infer parameters.

A transformer is a deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data. It is used primarily in the fields of Natural Language Processing (NLP)[[1]](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)#cite_note-:0-1) and Computer Vision (CV).[[2]](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)#cite_note-2)

Like Recurrent Neural Networks (RNNs), transformers are designed to process sequential input data, such as natural language, with applications towards tasks such as translation and text summarisation. However, unlike RNNs, transformers process the entire input all at once.

The dataset used is a plethora of Java program submissions from CodeForces, a popular coding contest platform. It contains around 4200 problems.

Data fields:

• src: a string feature, representing the source code in Java.

• complexity: a string feature, giving program complexity.

• problem: a string of the feature, representing the problem name.

• from: a string feature, representing the source of the problem.

complexity filed has 7 classes, where each class has more than 500 codes each. The seven classes are **constant, linear, quadratic, cubic, log(n), nlog(n) and NP-hard**.

Flask API

Flask is a widely used micro web framework for creating APIs in Python and running Machine Learning projects. It is a simple yet powerful web framework that is designed to get started quickly and easily, with the ability to scale up to complex applications.

Why Flask?

* Consumes minimal resources
* Easy to set up and use
* Streamlines application development, deployment, and maintenance
* Extensions available to add desired features
* Easy to learn for those who know Python
* Has extensive documentation
* APIs are well-designed and coherent
* Excellent scalability for simple applications
* Built-in web server and debugger
* Unit testing support
* RESTful request dispatching

Use of Flask in our project:

* To create APIs to interact between the front end of our project and the backend, where all the computations happen.
* To perform various computational tasks on data.

Front-End

Flutter Framework

Flutter is a mobile app development platform created by Google. It allows developers to create web, desktop, and cross-platform apps that run on Android and iOS devices. Flutter uses a reactive programming language called Dart, making development faster and easier than traditional methods.

Why Flutter?

Some of the top features of Flutter include:

Dart programming language: Flutter uses the Dart programming language, which is easy to learn and allows you to develop high-quality apps.

Hot reload: Flutter's "hot reload" feature lets you quickly and easily make changes to your app without restarting it.

Expressive and flexible UI: Flutter's UI elements are built using the same principles as Google's Material Design guidelines, giving you an expressive and flexible way to create beautiful apps.

Native performance: Flutter apps are compiled to native code, giving you the best possible performance on both iOS and Android.

Open source: Flutter is an open-source project, which means you can use it for free and contribute to the platform's development.

Provider-State Management

The concept of state management remains one of the most critical topics in Flutter. This is because everything we do in Flutter, from operations related to receiving information from a user to displaying a piece of data, deals with the state. Therefore, managing this data in the best way possible ensures the application is clean-coded, properly abstracted, operates smoothly, and delivers the best results possible.

The Provider package, created by Remi Rousselet, aims to handle the state as cleanly as possible. In Provider, widgets listen to changes in the state and update as soon as they are notified.Therefore, instead of the entire widget tree rebuilding when there is a state change, only the affected widget is changed, thus reducing the amount of work and making the app run faster and more smoothly.

Firebase

Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

* Analytics
* Authentication
* Cloud messaging
* Realtime database

Functionality provided:

Background:

When we first came to know about the problem statement, we sought “solace” in a code complexity calculator, being Computer Science and Information Science students. This project shows our iterative approach and subsequent understanding of further prompts provided. Hence, we moved from the Java Big-Oh complexity calculator, then the python maintainability calculator, the python flowchart generator and finally tried our hand at the Github malicious commit checker.

1. Java Time complexity calculator:

The first thing which comes to mind when the topic of “code complexity” for any engineering student is the Big-Oh notation. Hence, this was our first shot at the project too.

1. Python Function-call flowchart:

Sometimes, we write incredibly complex code, with many classes, functions and function calls. This aspect of our project attempts to visualize these function calls so that the user can keep track of the code flow and verify if the flow is as required by them. This was, they can increase the efficiency of code and save time using the visualization.

1. Github malicious commit checker

Motivation: We stumbled upon a research paper which aims to track malicious Github repository commits, using a tool developed by the researchers, called “Anomalicious”. Although the tool isn’t available for use yet, some insight into the criteria was shed in the paper. Factors which could make a commit malicious could be:

1. Time of commit: usually, people working on the same file reside in the same geographic location, and even if they do not, there are fixed working hours. Thus, it makes sense that a commit made before and after these hours could be malicious. Either another person has committed, or the user could have been hacked, in the worst case. While this may not be right all the time, with the growth of remote work, it seems like a plausible argument to support our cause and proceed further.
2. File type of commit: Would it not be suspicious if a frontend react developer suddenly makes changes to a C++ file? Or, any changes made to sensitive configuration files like json or xml files, unless from a trusted source, could possibly be malicious.
3. Lines of code modified: If, all of a sudden, 100 lines of code are deleted from a file, hence reducing the functionalities of the project. Similarly, if a large number of lines are suddenly added in a commit, it would be reasonable to inspect the same.

Our “model” is an extremely rudimentary version of a decision tree, completely comprising of “ifs” and “elifs”.

1. Python Code Maintainability Calculator:

While examining standard methods of code complexity, we came across a metric called the “Maintainability Index”, which in turn is a formula:

Maintainability Index = MAX(0,(171 - 5.2 \* ln(Halstead Volume) - 0.23 \* (Cyclomatic Complexity) - 16.2 \* ln(Lines of Code))\*100 / 171)

Cyclomatic complexity is defined as measuring “the amount of decision logic in a source code function”, i.e, “ifs”, “elifs” etc.

Halstead volume: You can view this as the bulk of the code – how much information does the reader of the code have to absorb to understand its meaning.

The maintainability of a code can be classified into 3 classes, here A, B and C.

Class A: score of 20-100 indicates good, or easily understandable code

Class B: score of 10-19 indicates code which can be made more efficient by reducing unnecessary lines, code etc.

Class C: score of 0-9 indicates extremely poorly written code, with multiple variables declared but not used, creating complex loops to perform simple operations, using multiple lines of code instead of keeping the code simple and efficient, etc.

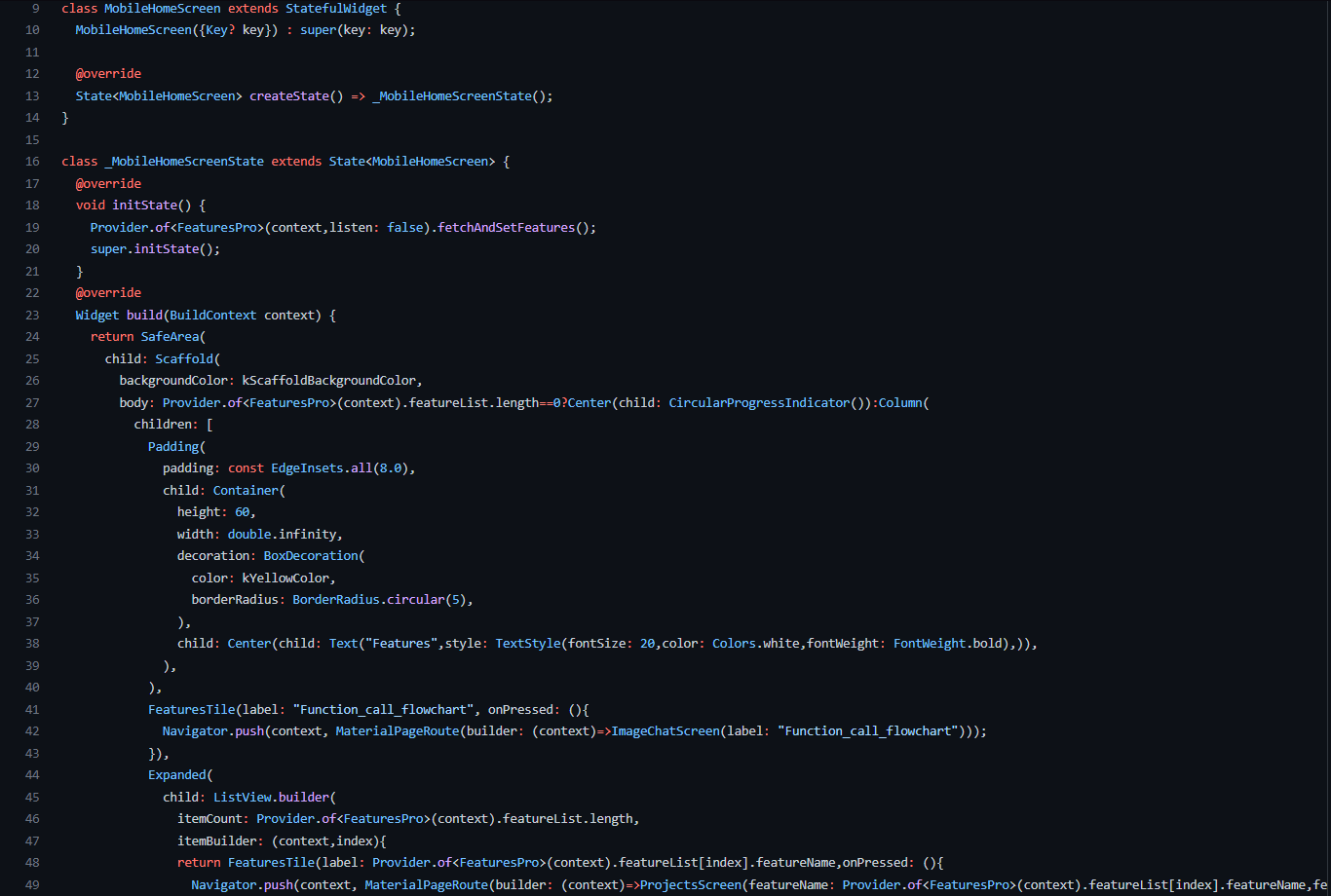
Additional code snippet

Front End

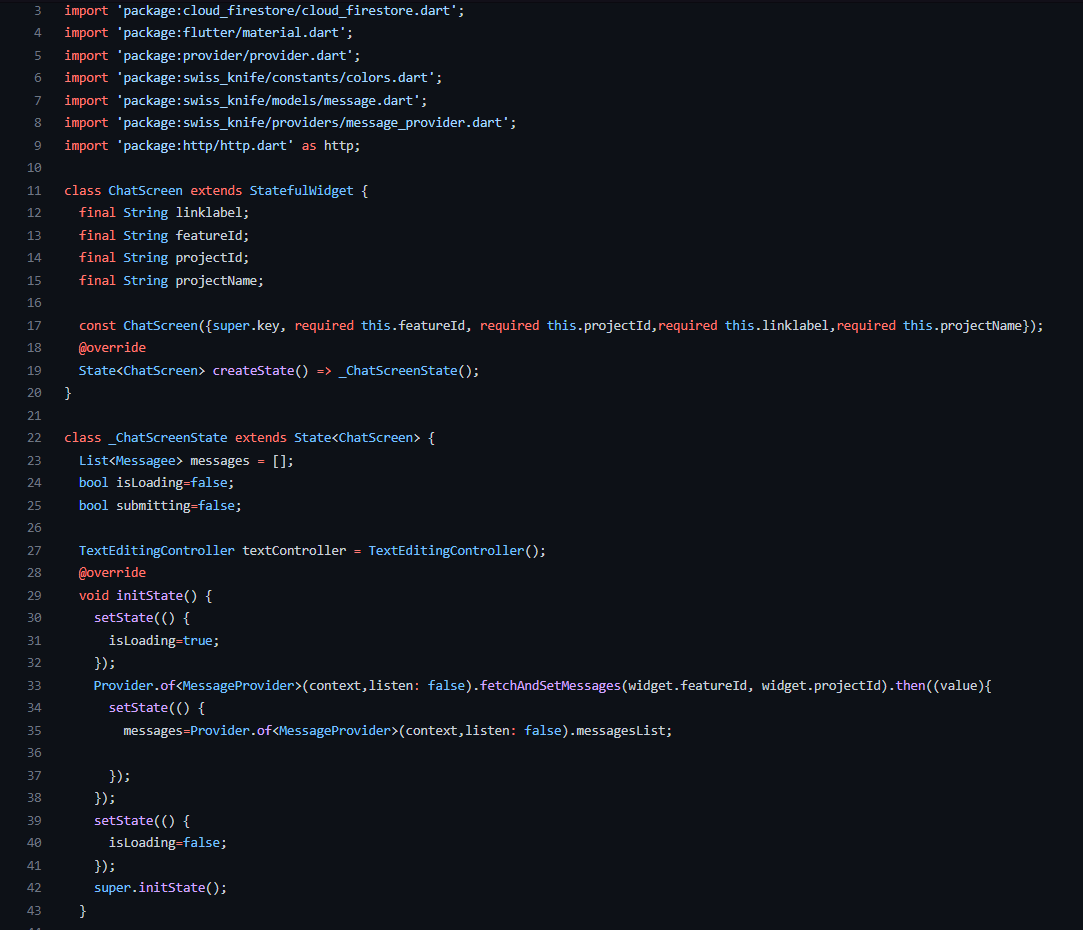
The front is divided into multiple screens:

1. Home Screen:

It holds all the page route names that help in navigations to the other screen. It is responsible to display all the 4 features that are provided by the app.



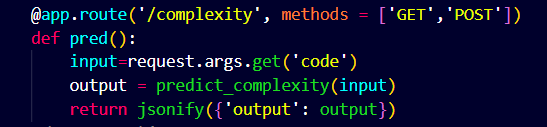
1. ChatBox : The second major screen for the given app is the Chatbox screen which is used to interact with the back end and get the result displayed .



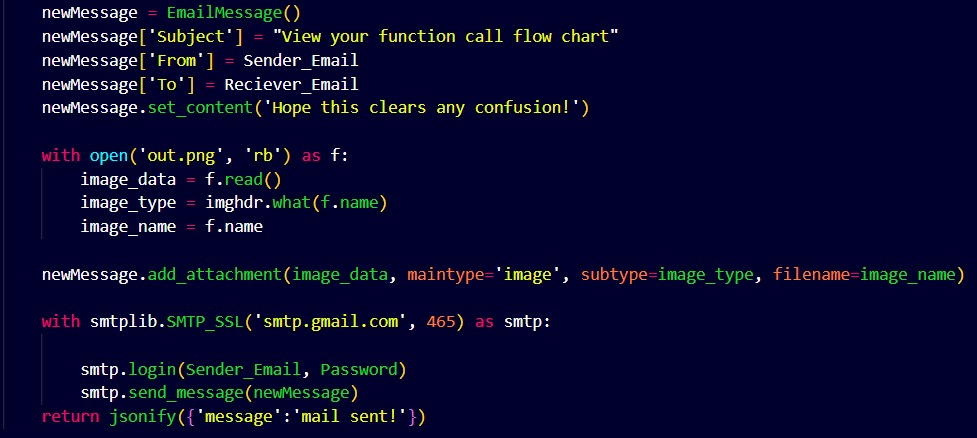
Back-End:

The Back-end consist of mainly 4 major flask APIs that connected the 4 functionalities provided by the app.

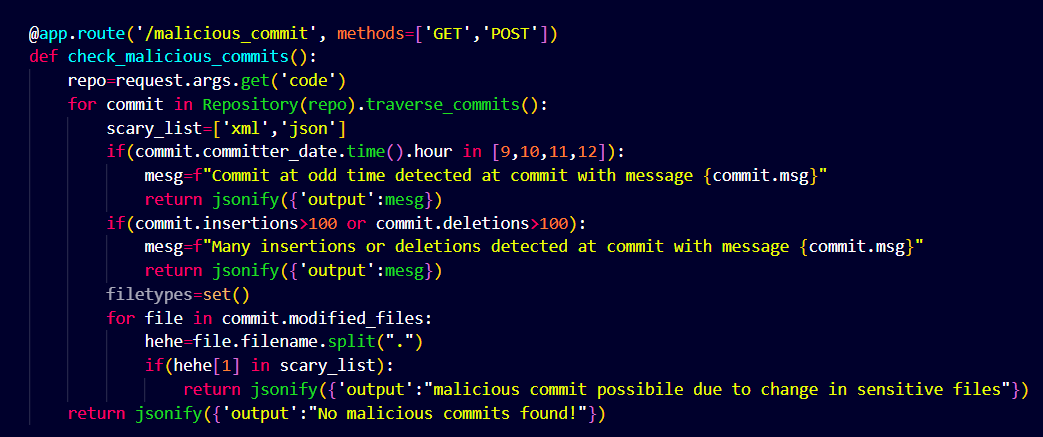
1. Java Time complexity calculator:



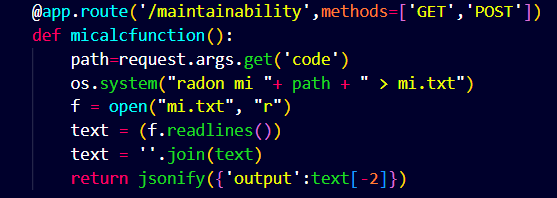
1. Python Function-call flowchart:



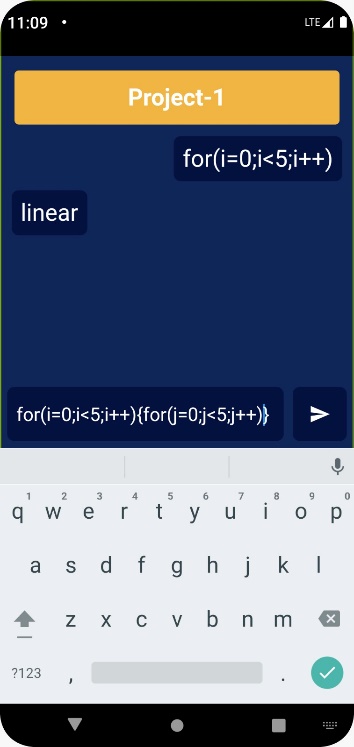
1. Github malicious commit checker:

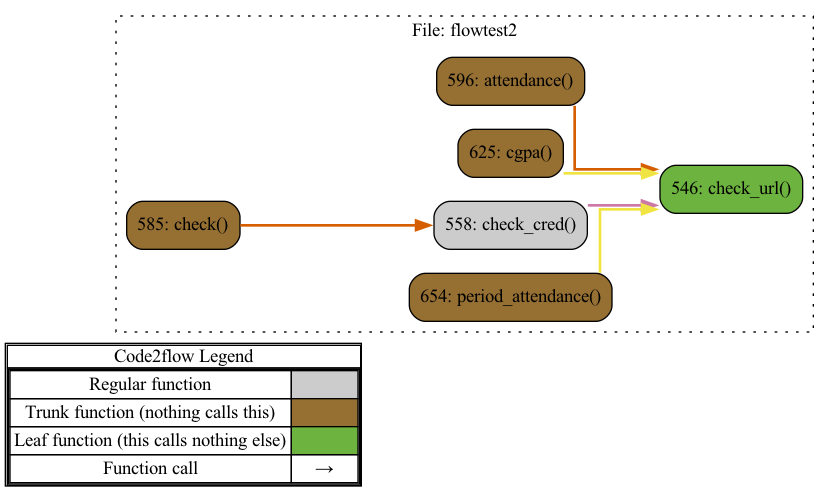


1. Python Code Maintainability Calculator:



Resulting Screens:

Future scope:

1. This project can be improved by further obtaining training data for the java complexity calculator.
2. We can provide targeted insights for code improvement corresponding to each class, and also maybe implement an algorithm to do the same.
3. The Github malicious commit checker could definitely be improved to understand the nature of malicious commits more, and could be a useful tool in the future.

Conclusion:

Usability:

The app is user-friendly and easy to navigate.

Scalability:

Once the required dependencies are installed, the app can be used by anyone, and has scope for the implementation of additional features and also improvement of current features.

Novelty:

Our app is a unique app with multiple features in one single app, hence making it quite innovative.

In conclusion, we have developed a fully functional flutter app, which can also be run as a web application. It is connected to FireBase, which allows users to access their previous queries and projects. The functionalities implemented are:

1. Java Code complexity calculator
2. Python code maintainability index calculator
3. Python function call flowchart
4. Github malicious commit checker

References

1. <https://blog.logrocket.com/quick-guide-provider-flutter-state-management/>
2. <https://flutter.dev>
3. <https://firebase.google.com>
4. <https://github.com/ishepard/pydriller>
5. <https://radon.readthedocs.io/en/latest/commandline.html#usage-with-jupyter-notebooks>
6. <https://github.com/scottrogowski/code2flow>
7. <https://github.com/arthtalati/Deep-Learning-based-Authorship-Identification/tree/master/data>
8. <https://wily.readthedocs.io/en/latest/>
9. <https://radon.readthedocs.io/en/latest/intro.html>