1. Project Overview

- **Project Name:** INGInious - CESReS Machine Learning Integration
- **Project Description:** The purpose of this project is to integrate a student code classifier model (CESReS) into INGInous, the open source educational platform for computer science students. The objective is that when students submit their code to course questions, they get detailed, descriptive error feedback so that they can better recognize their mistake and have a better clue on how to fix it. Emphasis on the “clue”, as the model should not return the answers to the question.
- **Project Status:** Proof of Concept delivered at [jorgetomaylla21/INGInious-CESReS(github.com)]
- **Team Members:** Jorge Tomaylla Eme (MIT Intern).
- **Stakeholders:** Professor Siegfried Nijssen, Master Student Guillaume Steveny.

2. Setup and Installation

- **System Requirements:**
  - Operating System: Linux VM
  - Hardware: I asked IT for a Linux VM with Ubuntu client and 1TB memory
  - Software Dependencies: There are conda and pip dependencies with specific (some outdated) versions in the ./correct_env_backup yaml file and the ./api_pip_packages txt file inside the repo. The first file contains the dependencies for the “cesres” conda virtual environment. This environment manages the CESReS server. The second file contains the dependencies for the API virtual environment which launches the fastAPI application. We need 2 virtual environments because there are dependencies that are shared across both tools, but they need different versions to work properly.
  - Other Requirements: VM must have a valid IPv4 and IPv6 connection to connect INGInious to the VM API.
- **Installation and Usage Steps:**
  - Obtain a Linux VM in OpenNebula or ssh into the existing “inginious-vm” VM made by tomayllaeme.
  - To successfully log into the VM:
    - Open a terminal and SSH into UC louvain student account
    - Generate an SSH public key for IT to grant you access to the VM
    - In the student terminal, SSH into root@IP address. In the indigenous-vm case it is root@2001:6a8:308f:10:0:83ff:fe00:7
  - Clone the repo into the VM
  - Load the cesres (conda) and api env config files
  - To launch the CESReS server:
    - Conda activate cesres (this will activate the right dependency versions for the server)

        - Run the executable ./launch_server.sh . This will open an asyncio connection at port 8000 in the localhost
        - Alternatively, you can cd into the gui folder and run the ./launch_gui.sh executable. You do not need to configure the java gui for this to work.
    - When you run the server, you will find out if your dependencies were installed correctly and if there are dependency or file conflicts. Here is how to solve them:
        - For dependency conflicts, pip freeze and compare dependency versions in the current cesres environment to the ones in the ./correct_env_backup.yaml file. You might need to manually install specific versions of certain dependencies. Follow the error log when launching the server to figure out which dependencies have a conflict or are missing.
        - There might be an “unable to load torch model” error. This occurs because the demo/model_weights.th is really large, so the weights are corrupted in Github due to the upload size limit. You can obtain the original weights in this zip folder, provided by Guillaume Steveny: https://drive.google.com/file/d/1Q78S-Pf0tRvtIj-WqdODu-4Ilj-QZtSM/view?usp=sharing . You can then use the scp linux command to transfer the model_weights file from your local computer to the VM. Make sure you place it in the demo folder.
    - If everything runs smoothly, when you launch the server it should read “Starting the server”
    - Next, we must launch the API that will communicate to INGInious and the CESReS server:
        - Open a new VM session in a new terminal
        - Run “conda activate api” to activate the API environment
        - Run “python3 api.py”. This will automatically open a connection at port 8001 at host “0.0.0.0” that is listening for HTTP requests.
    - Before configuring INGInious, you can quickly test the API and CESReS server communication by running:
        - curl "http://127.0.0.1:8001/data?message=while%20True%3A%0A%09print('hi')%0Areturn%200"
        - This sends code with an infinite loop as a query to the end-point “data”, and should return this message:

          > From AI Assistant: Your code has an error in the form of bad_loop with probability 1.0.
          >
          > Recommendations: Make sure your while conditions terminate.

        - When you curl this request, you should be able to see in real time the CESReS server taking an active client and returning a lot of probabilities. Similarly, in the fastAPI app you should see a new HTTP GET request.
        - If you do not see the above, your hosts or ports might be inactive or conflicting. Using commands like ping6 can help determine if a specific

port is actively listening or if it is already occupied. If so, kill the processes in the ports and launch the server or api again.

- Now that you are able to run the API and server, it is time to configure INGInous to send and accept HTTP requests and responses:
  - Ask IT for a course sandbox and instructor privilege so you can freely change the internal files. I chose the Intro to Programming course.
  - Choose a question with code input to test. My go-to example was the Fibonacci question in session 3.
  - Click on “Edit Task” on the left menu and then Task Files. This will display the files that makeup the question, including the run file and tests.
  - Replace the run file content with the ./run_inginious file in the repo. Notice the call_api() function, where it is called, and the HTTP GET parameters. Save changes.
  - Now we need to configure the environment:
    - Click on “Environment”
    - Check the box that says “Allow Internet access inside the grading container?” so that we can communicate with the API network.
    - Ask IT to make a custom python3 environment that includes the “requests” library. This package is necessary to send an HTTP request to the API and is not included in the default Grading environment. Once IT makes this custom grading environment, set Grading environment to the newly created environment.
    - While you wait for IT to create the environment, you can insert the code snippet in Appendix A before the import declarations to force a pip install of the “requests” package. This will work, but will add about 15 seconds to every request since you are pip installing the library every time you send a GET request.
  - You are all set up now. You can now write code from the user side, click submit, and if the code is erroneous, the API will respond with the type of error and recommendations to solve the error. Read on for some usage examples.

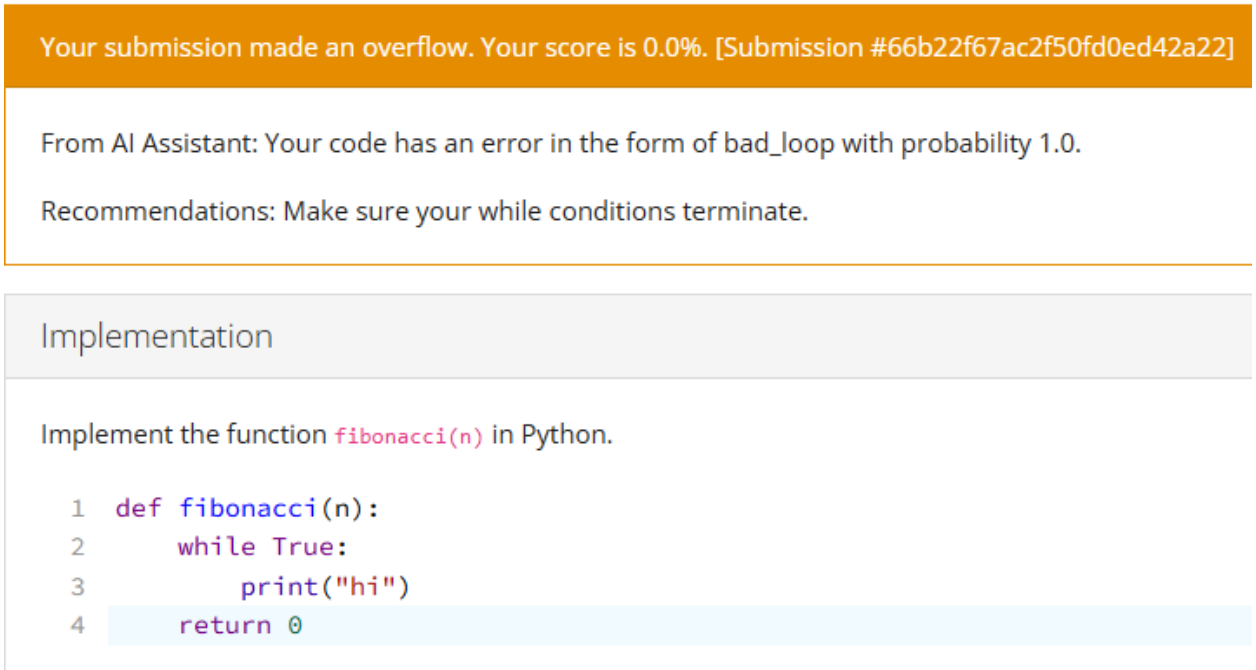
3. Project Structure

- **Repository Structure:**
  - The current project was built upon the CESReS model repo of Guillaume Steveny’s Master Thesis at StevenGuyCap/CESReS-model(github.com). Most files concerning the Cesres model have been unchanged, except for the configuration yaml files, whose versions are outdated but need to be installed as the old versions exactly for the server to run. The original project ran on Windows, but I adapted it to Linux. Here are the files that I have added:
  - ***api.py***:

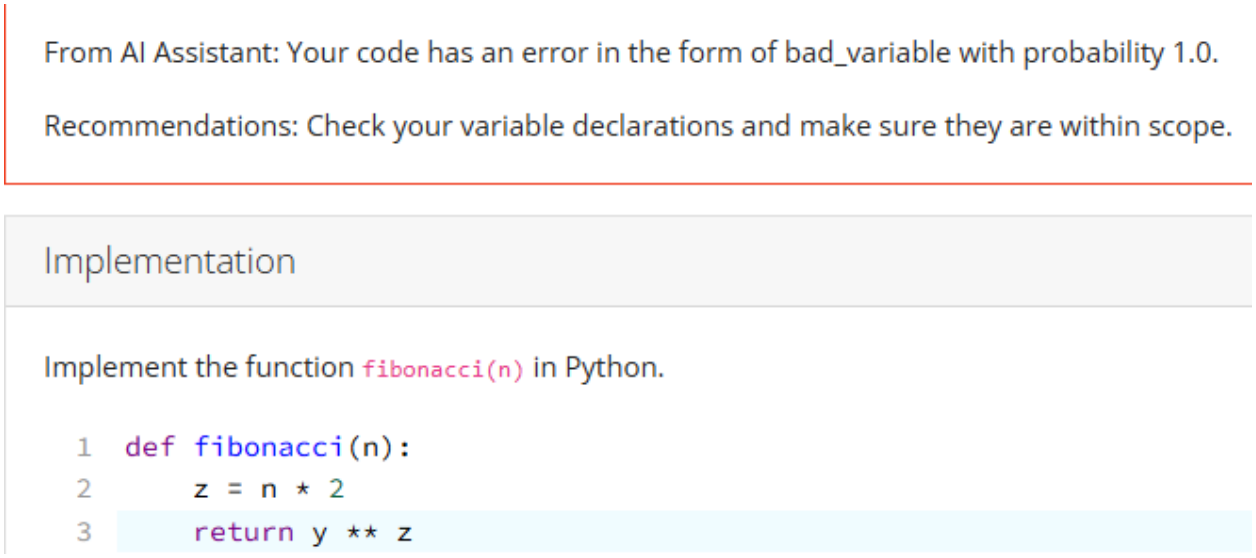
        - Opens an asyncio connection in port 8000 to connect with the CESReS server
        - Once it gets the label probabilities from the server in csv format, it finds the highest probability and returns it along with a static mapping from label to fix recommendation.
        - There are 16 mappings from error labels to recommendations on how to fix it. In the future, this static mapping can be replaced by a new model that returns human-readable, sensible hints for the student.
    - ***Api_env.yml:*** Contains the basic setup for the api conda environment
    - ***Api_pip_packages:*** Contains the pip dependencies that must be installed in the api environment
    - ***client.py:*** Asyncio client to test the server model in an easier way.
    - ***Correct_env_backup:*** The cesres conda environment. You can create a new conda environment directly from this file.
    - ***Launch_server.sh:*** Executable file that runs the cesres_graphcodebert_model.py file with a special config file.
    - ***Run_inginous:*** Replace the run file in the INGInous questions with this file to connect INGInous to the API.

4. Usage Examples

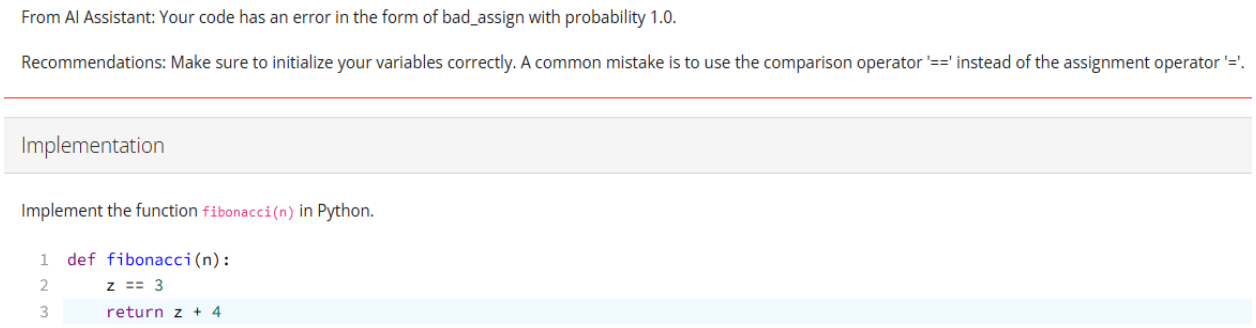
- Bad_loop label example:



- Bad variable code example:



- Bad Assign code example:



10. Appendix

- A:
  - def install_package(package_name):
  - try:
  - subprocess.check_call([sys.executable, "-m", "pip", "install", "--user", package_name])
  - print(f"Successfully installed {package_name}")
  - except subprocess.CalledProcessError as e:
  - print(f"Failed to install package {package_name}: {e}")
  - sys.exit(1)
  -
  - # Install requests package
  - install_package("requests")
  -

```
# Ensure the user site-packages directory is in the sys.path
site_package_path = subprocess.run([sys.executable, "-m", "site", "--user-site"], capture_output=True, text=True).stdout.strip()
if site_package_path not in sys.path:
    sys.path.append(site_package_path)

# Import requests after installation
import requests

```