CodeRunners

Team Members Name

- Iyana Jones
- Arghavan Noori
- Aaliyah Lockett
- Copernic Mensah
- Holy Agyei

https://on.soundcloud.com/u1r553T8KodM0Lj0j
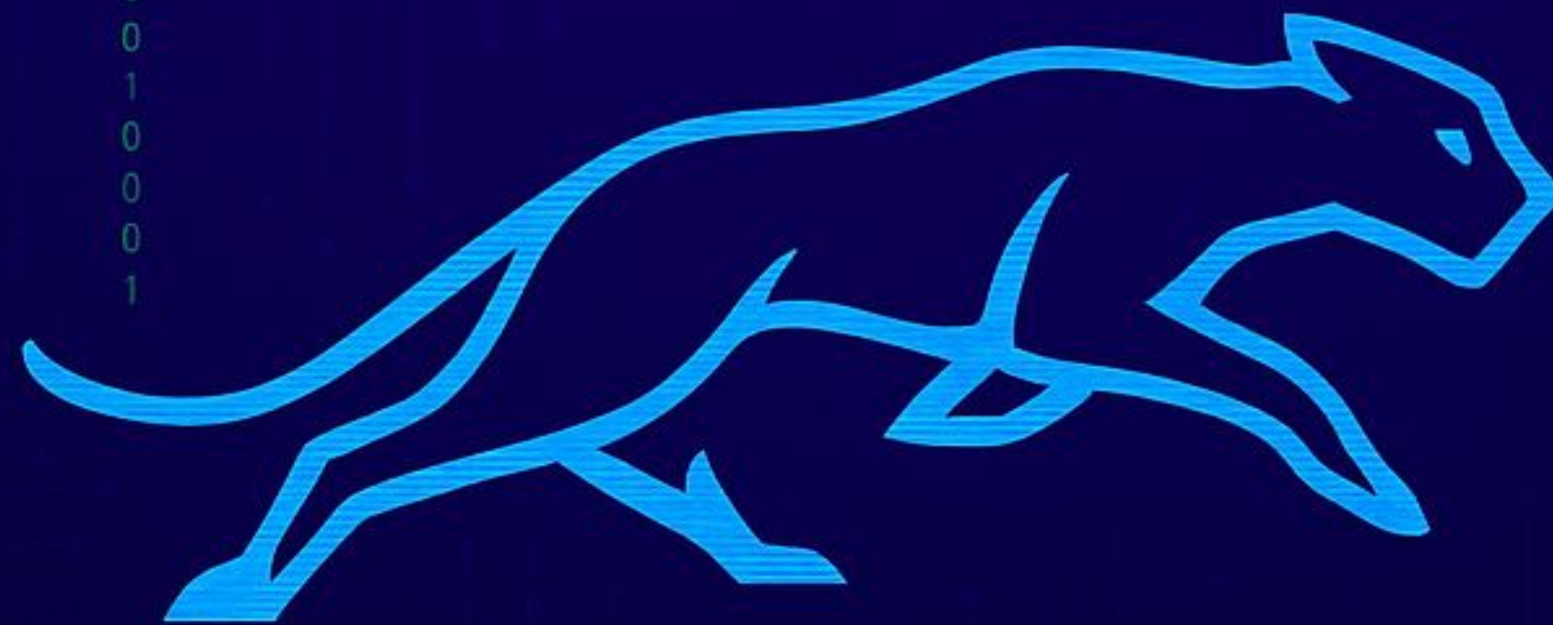
CodeRunners Key Milestones

01
Team formation, paper selection, and role assignment
Deliverables: Intro slide, README.md, GitHub repo with paper list and goals

02
Define reproducibility metrics and evaluation criteria.
Deliverables: Reproducibility scorecard (template), test plan

03
Evaluate reproducibility across multiple papers (ICSE/SC24)
Deliverables: Scorecards, logs, Python scripts for automated scoring

04
Build comparison dashboard
Deliverables: Streamlit/Flask portal with visual metrics for all papers

05
Submit final poster and presentation
Deliverables: Final poster, presentation slides, portal link, updated repo

Team Roles

& Responsibilities

Iyana | Lead

Tracks goals, edits README, manages daily progress, ensures overall project alignment.

Arghavan | Model Analyst

Compares model outputs, analyzes results, and scores reproducibility gaps.

Copernic | Presenter

Creates compelling visuals for the poster and presentation slides.
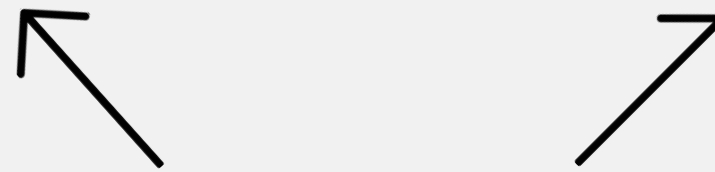
Aaliyah | Experiment Engineer

Sets up tasks, configures environments, and runs models for evaluation.
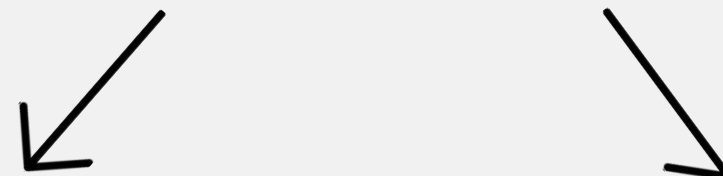
Holy | Portal Builder

Develops the interactive dashboard or website for the reproducibility scorecard and visualizations.

Github

https://github.com/SGX3CodeRunners/RealWorldBugs.git

CodeRunners
Project Overview and Goals

Objective:

Evaluate and compare reproducibility across multiple ICSE 2023 & SC24 papers focused on large language models (LLMs) for code understanding.

Goals:

- Score each paper using a standardized reproducibility framework.

- Build a public portal to visualize comparative results.

- Summarize findings in a Gateways 2025 poster.

CodeRunners Progress

- Expanded from single paper to multi-paper comparative reproducibility study
- Designed and implemented a reproducibility scorecard (100-point framework)
- Currently generating Python code to automate scoring from paper content
- Challenge: Missing GitHub links in some papers limits full artifact scoring
- Streamlit/Flask portal under development to visualize paper scores
- All updates align with the revised project plan (Comparative Repro Study)

```
Paper ID: 18
Title: Validating SMT Solvers via Skeleton Enumeration Empowered by Historical Bug-Triggering Inputs
Score: 15
Artifact URL: https://github.com/CGCL-codes/HistFuzz
DOI URL: https://doi.org/10.1109/ICSE48619.2023.00018
Notes:
  - Code available on GitHub (assumed open-source license).
  - Docker/Containerization: Requires manual check of the repository.
  - Dependency Management: Requires manual check of the repository.
  - Build Instructions: Requires manual check of the repository README.
  - Specialized Hardware Support: Requires manual check of the repository.
  - CI/CD Pipelines: Cannot be inferred from URL. Requires manual check.
  - Version Control: Assumed via GitHub.
  - Comprehensive README: Requires manual check of the repository.
  - API/Data Schema Docs: Requires manual check of the repository.
  - Reproducibility Badge: Cannot be inferred from URL. Requires manual check.
  - Runtime Instructions: Requires manual check of the repository.
  - Result Validation: Requires manual check of the repository.
  - Public Dataset Links: Data accessibility uncertain from URL.
  - Data Preprocessing: Requires manual check of the repository.
  - Model Weights: Requires manual check of the repository.
  - Issue Tracking: Assumed via GitHub.
  - Discussion Forum: Cannot be inferred from URL. Requires manual check.
```

- Using chatgpt and manus ai, we created a python script in Google Colab that was able to run all of the papers through the scorecard. The issues we came across was it repeatedly listed all papers with a score of 13-15 unless we manually checked the Github repository.
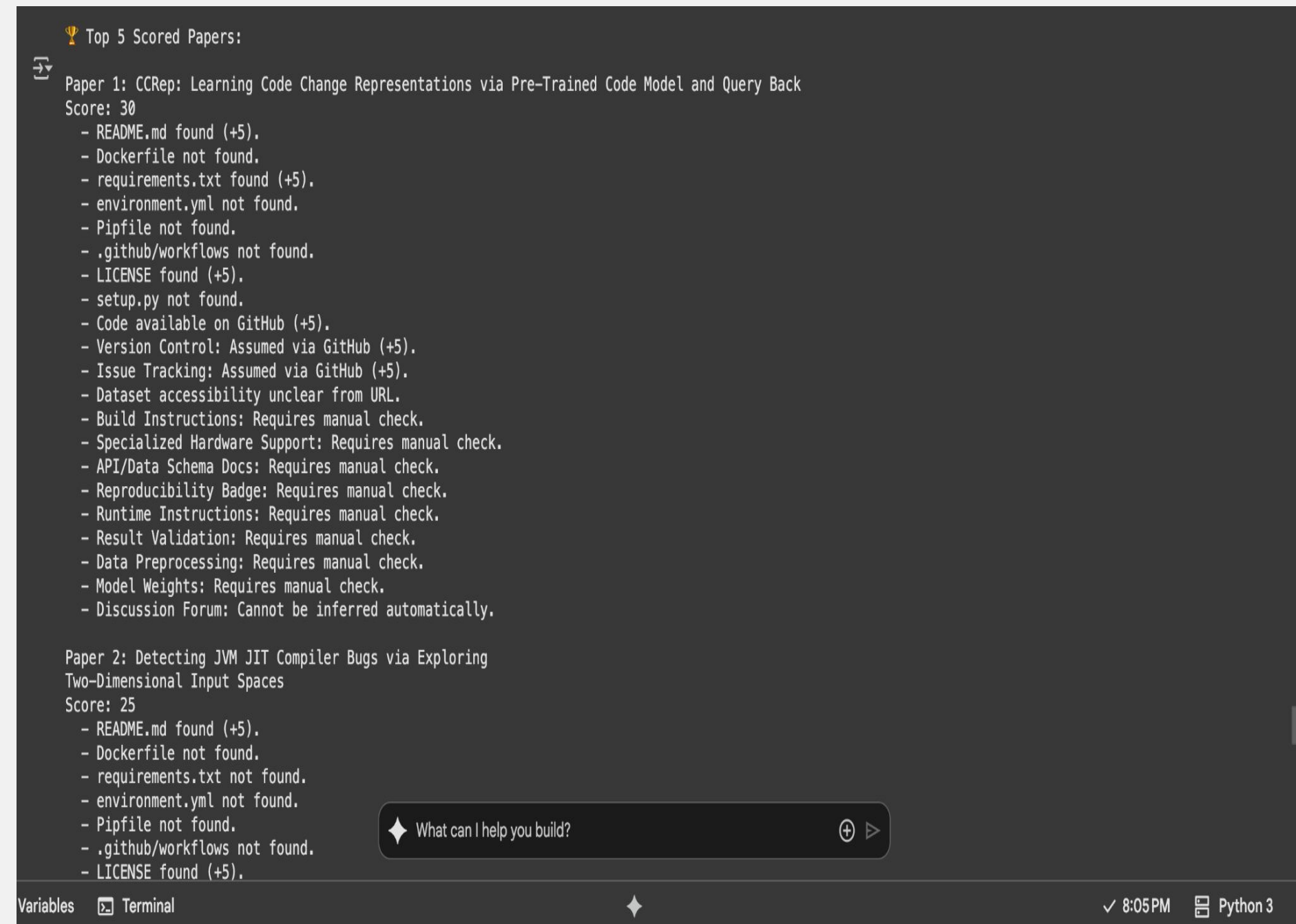- New approach: Semi-Manual (Hybrid) Approach (Recommended for Efficiency)

CodeRunners Progress

- Designed and implemented a reproducibility scorecard (100-point framework)
- Changed the code so that more pages are automatically scored
- Currently generating Python code to automate scoring from paper content
- Challenge: Missing GitHub links in some papers limits full artifact scoring
- Streamlit/Flask portal under development to visualiz paper scores
- Started building the project portal
-

```
⇥▾   Paper: One Adapter for All Programming Languages?
     Adapter Tuning for Code Search and Summarization
       README.md: Found
       Dockerfile: Not found
       requirements.txt: Not found
       environment.yml: Not found
       Pipfile: Not found
       .github/workflows: Not found
       LICENSE: Not found
       setup.py: Not found
     ----------------------------------------
     Paper: CCRep: Learning Code Change Representations via Pre-Trained Code Model and Query Back
       README.md: Found
       Dockerfile: Not found
       requirements.txt: Found
       environment.yml: Not found
       Pipfile: Not found
       .github/workflows: Not found
       LICENSE: Found
       setup.py: Not found
     ----------------------------------------
     Paper: Keeping Pace with Ever-Increasing Data: Towards
```

CodeRunners Progress

- Using chatgpt and manus ai, we created a python script in Google Colab that was able to run all of the papers through the scorecard. The issues we came across was it repeatedly listed all papers with a score of 13-15 unless we manually checked the Github repository.
- New approach: Semi-Manual (Hybrid) Approach (Recommended for Efficiency)
- Challenges where that for some papers you had to put it in manually and it was not showing the scores.
- We used Manus ai to get a code that would do all the papers automatically and give us the scores.

```
🏆 Top 5 Scored Papers:

Paper 1: CCRep: Learning Code Change Representations via Pre-Trained Code Model and Query Back
Score: 30
  - README.md found (+5).
  - Dockerfile not found.
  - requirements.txt found (+5).
  - environment.yml not found.
  - Pipfile not found.
  - .github/workflows not found.
  - LICENSE found (+5).
  - setup.py not found.
  - Code available on GitHub (+5).
  - Version Control: Assumed via GitHub (+5).
  - Issue Tracking: Assumed via GitHub (+5).
  - Dataset accessibility unclear from URL.
  - Build Instructions: Requires manual check.
  - Specialized Hardware Support: Requires manual check.
  - API/Data Schema Docs: Requires manual check.
  - Reproducibility Badge: Requires manual check.
  - Runtime Instructions: Requires manual check.
  - Result Validation: Requires manual check.
  - Data Preprocessing: Requires manual check.
  - Model Weights: Requires manual check.
  - Discussion Forum: Cannot be inferred automatically.

Paper 2: Detecting JVM JIT Compiler Bugs via Exploring
Two-Dimensional Input Spaces
Score: 25
  - README.md found (+5).
  - Dockerfile not found.
  - requirements.txt not found.
  - environment.yml not found.
  - Pipfile not found.
  - .github/workflows not found.
  - LICENSE found (+5).
```

What can I help you build?

Variables   Terminal   ✓ 8:05 PM   Python 3