***[Capstone Category]***

| **Project Name** | **Code Visualization** |
| --- | --- |
| Team Lead: | Curtice Gough |
| Team Member(s): | Curtice Gough, Joshua Hartzfeld, Catherine DiResta |
| Faculty Advisor(s): | Dr. Ryan Stansifer, Dept. of Computer Science, Florida Institute of Technology |

Design Problem Statement:

Debugging complex code can be time-consuming and error-prone due to the lack of effective visualization tools. Traditional debuggers may not provide adequate insight into data structures and their movements, hindering the debugging process. The challenge lies in developing a code visualization tool that offers interactive visualization of code structures and dynamic data movements to facilitate efficient debugging.

Major Challenges:

* Integration of Backend and Frontend

The separation of backend and frontend development poses a challenge in integrating all components seamlessly.

* GUI Development Familiarity

Some team members may lack familiarity with GUI development, requiring guidance from experienced members.

* Modern Java Compatibility

Ensuring compatibility with modern Java versions while utilizing existing backend functionalities presents technical hurdles.

Solution Methods:

* Interactive GUI Development

Utilizing PyQt framework for GUI development, custom widgets will be created to represent various data structures, ensuring an interactive user experience.

* Dynamic Code Analysis

Implementing a dynamic code analysis mechanism allowing users to step through source code during program execution, akin to popular debuggers, to track program flow effectively.

* User Intervention

Allowing users to rename or retype visual elements during analysis, inspired by popular decompilers, to address incorrect assumptions made by the visualization engine.

Evaluation metrics will include:

* Speed

Assessment of the time taken to generate visual elements after code submission.

* Reliability

Evaluation of the system's accuracy in identifying data structure types.

Conclusion:

The "Code Visualization" project aims to address the challenge of debugging complex code by providing an interactive visualization tool. By integrating dynamic code analysis and user intervention features, the project seeks to enhance debugging efficiency and accuracy.

Broader Implications and Future Work:

Successful development of the code visualization tool can have broader implications in software development, enhancing debugging processes across various domains. Future work may involve expanding the tool's capabilities to support additional programming languages and advanced debugging functionalities. Additionally, collaboration with industry partners to incorporate real-world use cases can further refine and validate the tool's effectiveness.