**Project Title: Eulerian and Hamiltonian Path Detection in Graphs**

**Project Description:**

The goal of this project is to create a program that can detect and visualize Eulerian and Hamiltonian paths in a given graph. We will implement this project in both Python and JavaScript to showcase how these path detection algorithms can be applied in two different programming languages.

**Project Objectives:**

* Implement Eulerian path and Hamiltonian path detection algorithms in Python and JavaScript.
* Allow users to interactively check for the existence of Eulerian and Hamiltonian paths in the provided graph.
* Visualize the detected paths on the graph for better understanding.
* Compare the performance and usability of the Python and JavaScript implementations.

**Project Implementation:**

Python Implementation:

* Implement the Eulerian path detection algorithm based on Euler's theorem.
* Implement the Hamiltonian path detection algorithm, which involves searching for a path that visits each vertex exactly once.
* Test the Python application with sample graphs and evaluate its performance.

JavaScript Implementation:

* Implement Eulerian path and Hamiltonian path detection algorithms in JavaScript.
* Use a JavaScript graph library like D3.js or Sigma.js for graph representation and visualization.
* Develop a web-based user interface using HTML, CSS, and JavaScript.
* Allow users to input the graph by interacting with the web interface.
* Implement interactive features to explore and visualize Eulerian and Hamiltonian paths.
* Test the JavaScript application in various web browsers to ensure cross-browser compatibility.

**Deliverables:**

* Python-based program for Eulerian and Hamiltonian path detection.
* JavaScript-based web application for the same purpose.
* User documentation explaining how to use the applications.
* Project report summarizing the development process and findings.

This project will provide valuable insights into the implementation of Eulerian and Hamiltonian path detection algorithms using both Python and JavaScript. It will also offer users an easy-to-use tool for graph analysis and visualization. By comparing the two implementations, we aim to identify the strengths and weaknesses of each approach and demonstrate the versatility of graph theory applications in different programming languages.