**Capstone Project**

**Assignment 1**

**Course code:** CSA1450

**Course:** Compiler design for automata

**Name:**

* + Bharath Kalyan Naidu K (192211467)
  + Nikhil P (192211587)
  + SreeVardhan P (192211478)

**Slot:** A

**Title:** An Investigation into Code Size Optimization Techniques

**Preliminary Stage:**

**Assignment Description**:

The project aims to investigate code size optimization techniques. It will involve analyzing existing codebases to identify areas for improvement and implementing strategies to reduce code size without sacrificing functionality or performance. The project will explore various optimization methods, such as removing dead code, minimizing data structures, and optimizing algorithms, to achieve efficient code size management.

**Assignment Work Distribution**:

* **Project Scope Definition:**
  + Define the scope and objectives of the project: The project aims to optimize code size while maintaining functionality and performance.
  + Specific goals of analyzing: Identify redundant code segments and optimize resource utilization.
* **Data Collection and Preparation:**
  + Identify the data sources: Source code repositories and documentation.
  + Develop a data collection plan: Establish procedures for collecting and organizing codebase data.
  + Cleanse and preprocess the collected data to ensure data quality: Remove duplicates and irrelevant files, and standardize formatting.
  + Consistency of the project: Ensure uniformity in data processing and analysis methodologies.
* **Exploratory Data Analysis (EDA):**
  + Conduct exploratory data analysis: Review codebase structure, size distribution, and complexity metrics.
  + Understand the patterns and trends: Identify common code patterns and areas of code bloat.
  + Perform descriptive statistics, such as summary statistics, distribution plots, and correlation analysis, to explore the relationships of the data: Analyze code size distribution and its relationship with other metrics.
  + Visualize the data using charts, graphs: Generate visual representations of code size distribution and optimization potential.

**2. Problem Statement**

* The identified problem revolves around the need for efficient code size management in software development.
* As projects grow in complexity, codebases often accumulate redundant or bloated code segments, leading to increased memory footprint and slower performance.
* This project aims to address these challenges by investigating and implementing code size optimization techniques to streamline codebases without compromising functionality or performance.

**3. Abstract**

The project focuses on investigating code size optimization techniques to enhance software efficiency. By analyzing existing codebases and implementing optimization strategies, significant reductions in code size can be achieved. This not only improves resource utilization but also enhances overall system performance. Through exploratory data analysis and careful design considerations, this project aims to provide insights and solutions for effective code size management in software development.

**4. Proposed Design Work**

**Identify the key components:**

* Code Analyzer
* Optimization Algorithms
* Testing Framework

**Functionality:**

* Analyze codebase for optimization opportunities
* Implement code size reduction techniques
* Evaluate performance impact of optimizations

**Architectural Design:**

* Modular structure to facilitate integration with existing development workflows
* Scalable architecture to handle large codebases efficiently
* Extensible design to accommodate future optimization strategies

**5. UI Design**

**Layout Design**

* + Flexible layout: Responsive design to accommodate various screen sizes and resolutions.
  + User-Friendly: Intuitive interface with clear navigation and informative visuals.
  + Colour Selection: Use of visually appealing color schemes to enhance user experience.

**Feasible Elements Used**

* + **Elements Positioning**: Logical placement of elements for easy access and navigation.
  + **Accessibility**: Ensure compatibility with assistive technologies for users with disabilities.

**Elements and Functions**

* Interactive visualization tools
* Real-time performance metrics
* Customizable settings for personalized experience

**6. Login Template**

**Login Process**

* Authentication using username and password
* Option for fingerprint authentication for enhanced security

**Sign-Up Process**

* User registration with email verification
* Secure password setup with strength indicators

**Other Templates**

* Password Recovery
* Account Settings

**7. Conclusion**

In conclusion, the investigation into code size optimization offers valuable insights and solutions for improving software efficiency. By implementing optimization techniques and adopting user-friendly design principles, significant improvements in codebase management can be achieved, leading to enhanced performance and resource utilization.

**Source code:**

def fibonacci(n):

if n <= 1:

return n

else:

return fibonacci(n-1) + fibonacci(n-2)

# Optimized code using memoization

def fibonacci\_memo(n, memo={}):

if n in memo:

return memo[n]

if n <= 1:

return n

memo[n] = fibonacci\_memo(n-1, memo) + fibonacci\_memo(n-2, memo)

return memo[n]

# Optimized code using iterative approach

def fibonacci\_iterative(n):

a, b = 0, 1

for \_ in range(n):

a, b = b, a + b

return a

# Test the functions

n = 10

print("Original:", fibonacci(n))

print("Memoization:", fibonacci\_memo(n))

print("Iterative:", fibonacci\_iterative(n))

**Sample output 1:**

Original: 55

Memoization: 55

Iterative: 55

**Sample output 2**:

Original: 3524578

Memoization: 3524578

Iterative: 3524578

**Sample output 3**:

Original: 6765

Memoization: 6765

Iterative: 6765