APPLICATION FOR BETTER UNDERSTANDING OF FLOW OF CODE

Project Reference No.: *45S_BE_4009*

College ***:*** *Shri Madhwa Vadiraja Institute of Technology and Management, Udupi*
Branch ***:*** *Department of Computer Science Engineering*
Guide(s) ***:*** *Dr. Nagaraj Bhat*
Student(S) ***:*** *Mr. Krishnamurthi Bhagawath*
Mr. Gautham G Nayak
Ms. K Niveditha Kamath
Ms. Meghana Adiga

Keywords:

Flutter, OCR, Graphviz, flowchart, puesdocode, algorithm, extraction, detection.

Introduction:

Computer programming is one of the core subjects in computer-related field, including Information Technology and business computer. One of the most important things in computer programming is understanding the logic behind the program/code written. Several research articles have shown a decline in number of students in computer programming courses, mainly due to the difficulty of students in writing efficient codes. One of the solutions, to solve this problem is to let students present the programming logic in form of a flowchart. Flowchart is a diagrammatic representation of a programming logic. Programmers can use to design and develop an algorithm for their program. Flowcharts are also very useful for comprehending and understanding a program. Over the years, flowcharts are proven to be a standard in depicting the logical flow of processes and effective in representing the flow control of software programs. But drawing a flowchart is time consuming and slows down the process of software development. Moreover, modifying or altering the flowchart is tedious, since it would require to draw another flowchart from scratch. Nevertheless, flowcharts are proven to be an important tool in programming.

Our project focuses on generating flowchart, when source code is given as input as well as generating a pseudocode when flowchart is given as input. The key feature of our project is to provide both these functionalities in a single platform. This could serve as an important tool and help people get some logic in their program without wasting much of their time in generating it.

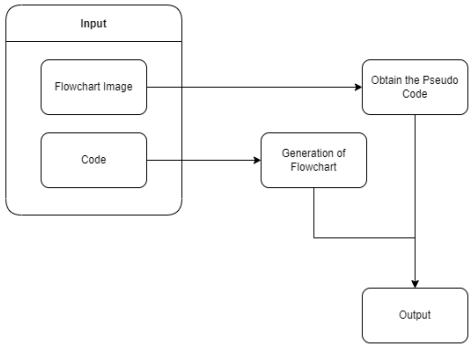
Objectives:

1. To implement a system that generates a flowchart, which makes programming easier to understand.
2. To develop a system that generates pseudocode from a flowchart and henceforth making it easier to type and assimilate pseudocodes.

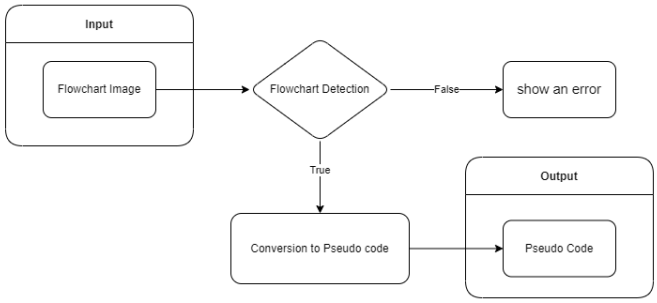
3. To invoke an interest in programming by making it easier to understand.

Methodology:

The methodology of our project consists of two modules, each of which has its own value and importance. These modules are explained below:



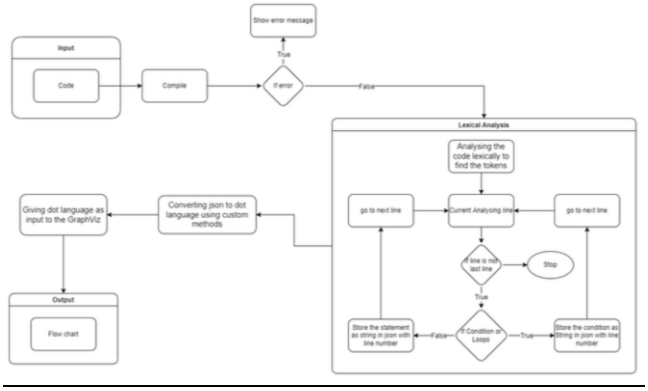
Module 1: Flowchart to pseudocode



1. Read the input image.
2. Use trained models for flow chart detection; If any shape is unrecognized, show an error message.
3. Check the detected shapes and arrow directions for any inconsistency.
4. Store all the shapes and arrows in separate lists.
5. Loop through list:
    a. Extract text using OCR and store.
    b. if the extracted text is STOP, end the loop.
    c. Find the next shape where the arrow is pointing.
    d. if next shape is rhombus, then:
        i. Extract the text from it and store it in a list.
        ii. Extract the text adjacent to arrows i.e., TRUE/FALSE.
        iii. If the condition satisfies then: find the next shape where the arrow (TRUE) is pointed. Repeat the loop step.
        iv. Else: find the next shape where the arrow (FALSE) is pointed. Repeat the loop step.

6. Finally, convert the extracted text to pseudo code through custom methods.

Module 2: Code to Flowchart



1. Compile the code written in the text editor by the user.
2. Show error message if any error persists.
3. Analyse the code lexically to find tokens.
4. Loop through each line:
    a. If the analysing line is a condition or a loop (say “if " or “while”) then store it as a string in json (JSON in AST format), with line number.
    b. If the analysing line is assignment or input/output (say int a, b) then store the statement as a string in json (JSON in AST format) with line number.
5. else if the analysing line is the last line, then STOP.
6. Convert the given json to DOT language using custom methods.
7. DOT language is thus given as an input to the GraphViz to generate Flowchart.

The final output is thus a flowchart and pseudocode which is implemented through the two modules of our project.

Results and Conclusions:

The expected output of the project is to generate a flow chart of any given program or code snippet typed and also to convert the structured flowchart to pseudocode.

The proposed project is relevant to students wherein the flowcharts can help them to map key concepts and associated information into a compact schema. And, writing pseudocodes opens the mind of the learner to a format in which the computer programs are written thus making programming easier to understand.

Future Work:

For the future work we are expecting to generate an animated visual representation and flow of code of any given program or code snippet typed in any of the programming languages.