Contents

[Purpose 1](#_Toc2640959)

[Core Features 1](#_Toc2640960)

[Stretch Goals 2](#_Toc2640961)

[User Stories 2](#_Toc2640962)

[Architecture 3](#_Toc2640963)

[Front End Stack 3](#_Toc2640964)

[Back End Stack 3](#_Toc2640965)

# Purpose

Provide a visual interface for creating and seeing the execution of short programs for algorithm testing and teaching. The coding is drag and drop and the execution displays the flow of the program and what is currently happening.

## Core Features

Running their written Programs.

Drag and Drop Mechanic.

Storing Variables

Creation of all Visual Nodes

Conditional Nodes

Functions

Swipe between functions.

Visual Indication of execution of program outside of just output.

During

After

## Stretch Goals

Saving Created Programs

* The program itself
* The execution flow output. (Because the output is main purpose of it.)

Ability to Pause the execution.

Animations

* Creating Nodes
* Executing Program

# User Stories

**Action:** User creates a new project.

**//Contemplate if want this extra layer or just functions and all functions see other functions,**

**//not scoped to project, because think about the purpose of application.**

**Response:** Project Structure is created, has collection of Function nodes, and Global Visual Syntax Nodes.

**Action:** User adds a block node. (Function / Conditional Block)

**Response:**

Function Visual Block Node is created, a composite of Visual Nodes.

Local tree of variables created.

See variables in outer scope. (By going up the tree)

**Action:** User adds an Operation Node. (Arithmetic, Comparison, Assignment)

**Response:** These are attached to Leaf Nodes attached to direct parent composite node, Leaf Nodes are linked list of non-composite / Block nodes.

**Action:** User runs the program.

**Response:**

***Completing this process with base block and syntax nodes priority 1, functions and projects later.***

The composite Visual Nodes enters two phases in separate thread.

**Visual Execution Phase**

Starting off as a background process, it waits for signal from Compilation Phase that there is enough to begin executing.

When Signal is sent, it begins executing the compiled code and displaying corresponding Visuals.

**Compilation Phase**

Another background process with higher priority then Visual Execution Phase.

This compiles the Visual Nodes into respective Syntax Nodes, by climbing down the composite tree, and processing Visual Nodes, then Decorating to be Executable Visual Nodes.

As blocks finish compiling signals are sent to Visual Execution Phase, so that these can happen in parallel.

Above might be stretch goal, and will instead just send signal when Compilation Phase is done.

**User Action:** Switch Between functions

**Response:** Process swipe left or right, display corresponding function by going through Linked List / Array of Function Nodes.

**//To Think about, function calls.**

# Architecture

### Front End Stack

Flutter

Dart

Pub

### Back End Stack

.Net Core

C#

dotnet