Project Plan/Design
&
Task Distribution

Integrative Project in Computer Science and Mathematics (420-204-RE)
Brick by Brick (CiHao Zhang, Lucas Chauveau, Qian Qian)

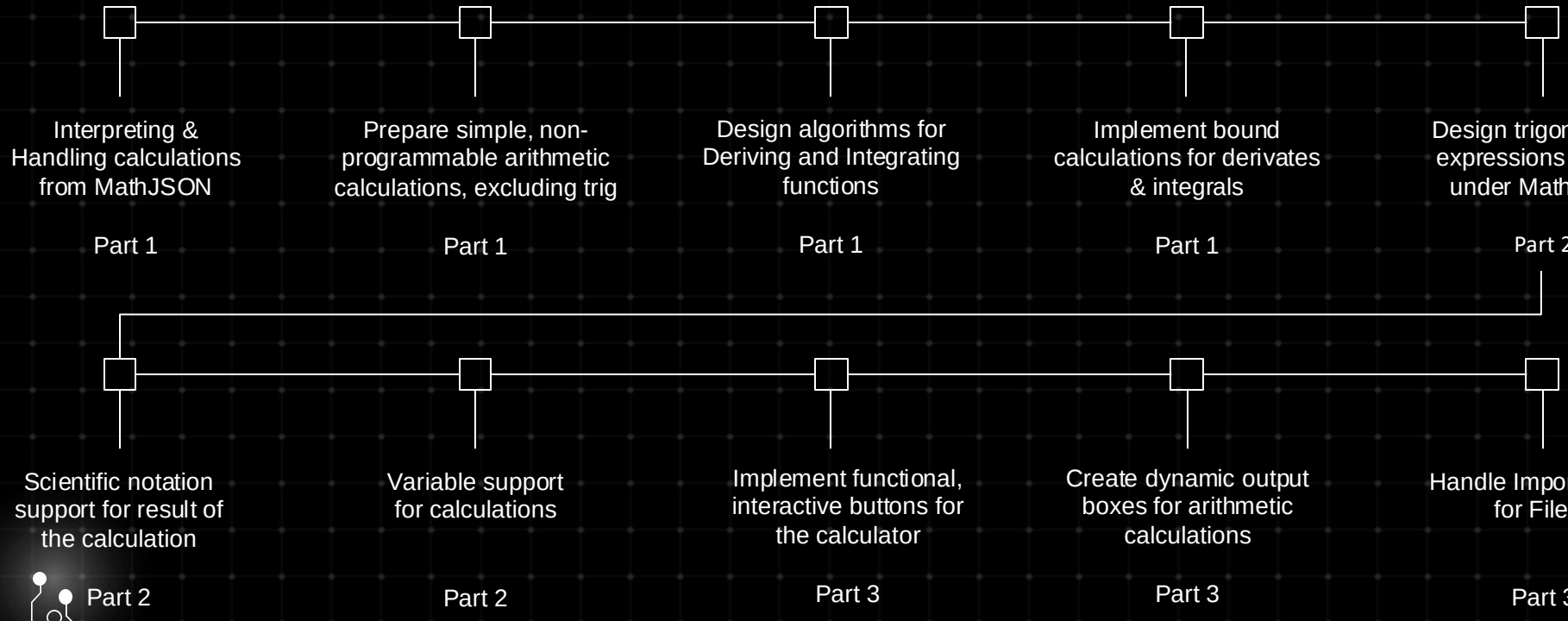
01

Tasks Breakdown, Timeline and Tasks Assignment

User Stories - Chow

User Story	Tasks
Arithmetic calculations, allowing the user to perform basic calculations needed for day-to-day computations.	- Prepare simple, non-programmable arithmetic calculations excluding trig. - Design algorithms for Deriving and Integrating functions. - Implement bound calculations for derivates & integrals.
Enhancing user experience through input formatting	- Interpreting & Handling calculations from MathJSON. - Variable support for calculations. - Implement functional, interactive buttons for the calculator. - Design trigonometric expressions to work under MathJSON.
Aesthetically pleasing outputs & proper formatting	- Create dynamic output boxes for arithmetic calculations. - Scientific notation support for result of the calculation.
Persistency through sessions	Handle Import/Export for FileIO.

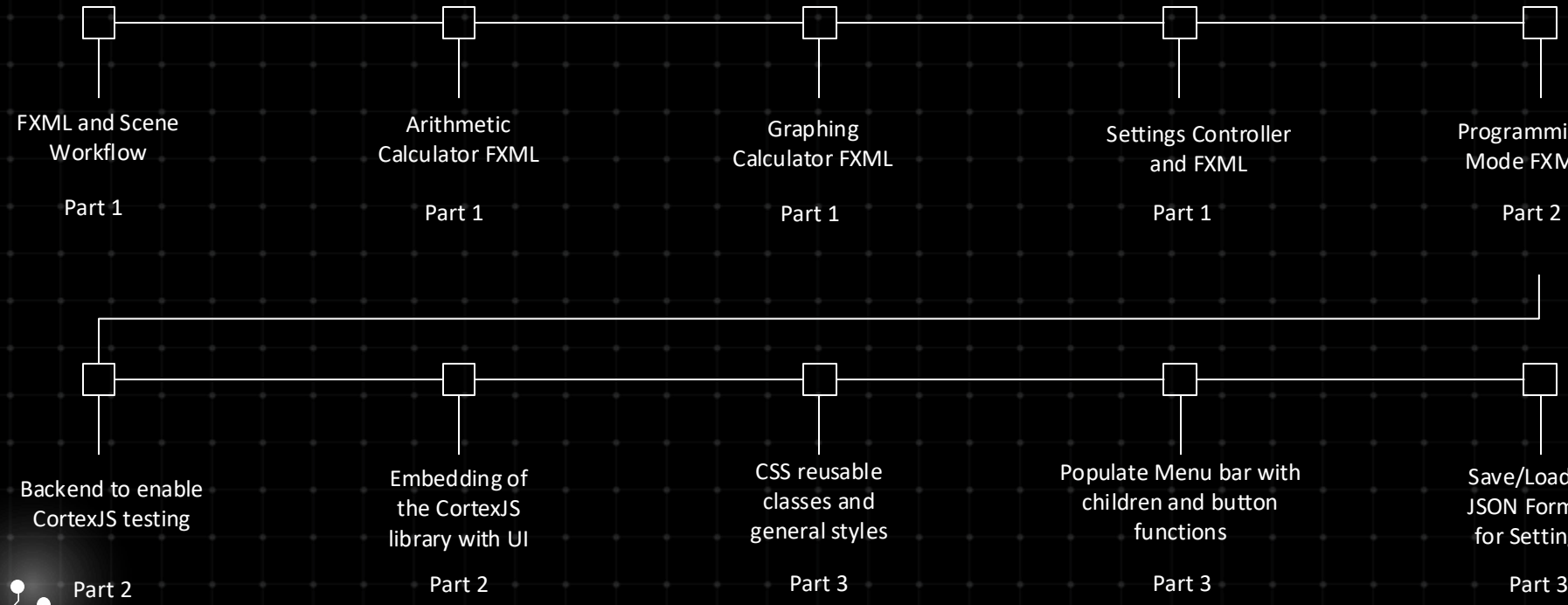
Tasks & Timeline - Chow



User Stories - Lucas

User Story	Tasks
Clean and Pleasant UI	Everything FXML and CSS related
Persistency between sessions	Save and Load system for user settings
Seamless experience	Embedding and testing of CortexJS

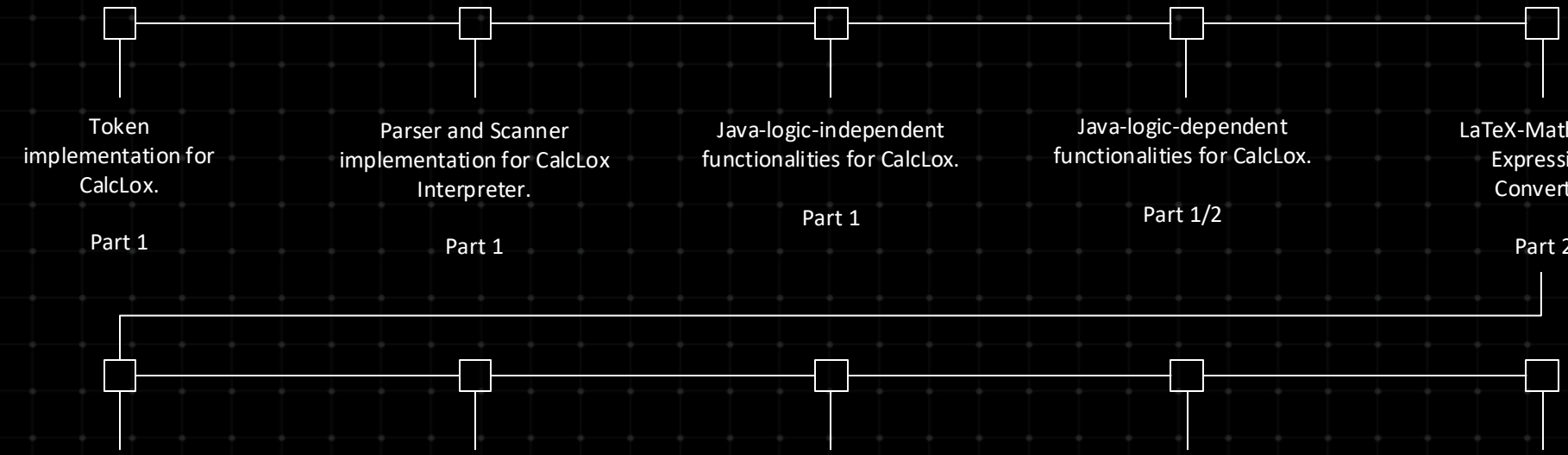
Tasks & Timeline - Lucas



User Stories - Qian

User Story	Tasks
Programming Mode that allows the calculator to do repetitively the same actions over a set of data.	- Implementation of the CalcLox Language Interpreter. - Implementation of the conversion of the visual-computational expression for formulas. - Integration of the programming mode and other modes of the calculator.
Library of functions for programs can be useful to have commonly used function stored and share for reuse.	- Implementation of the library support for CalcLox. - Implementation of a official library of functions.
Numeric (unit conversion, numeral system conversion) and Symbolic (formula simplifier) can be useful.	Implementation of the tools.

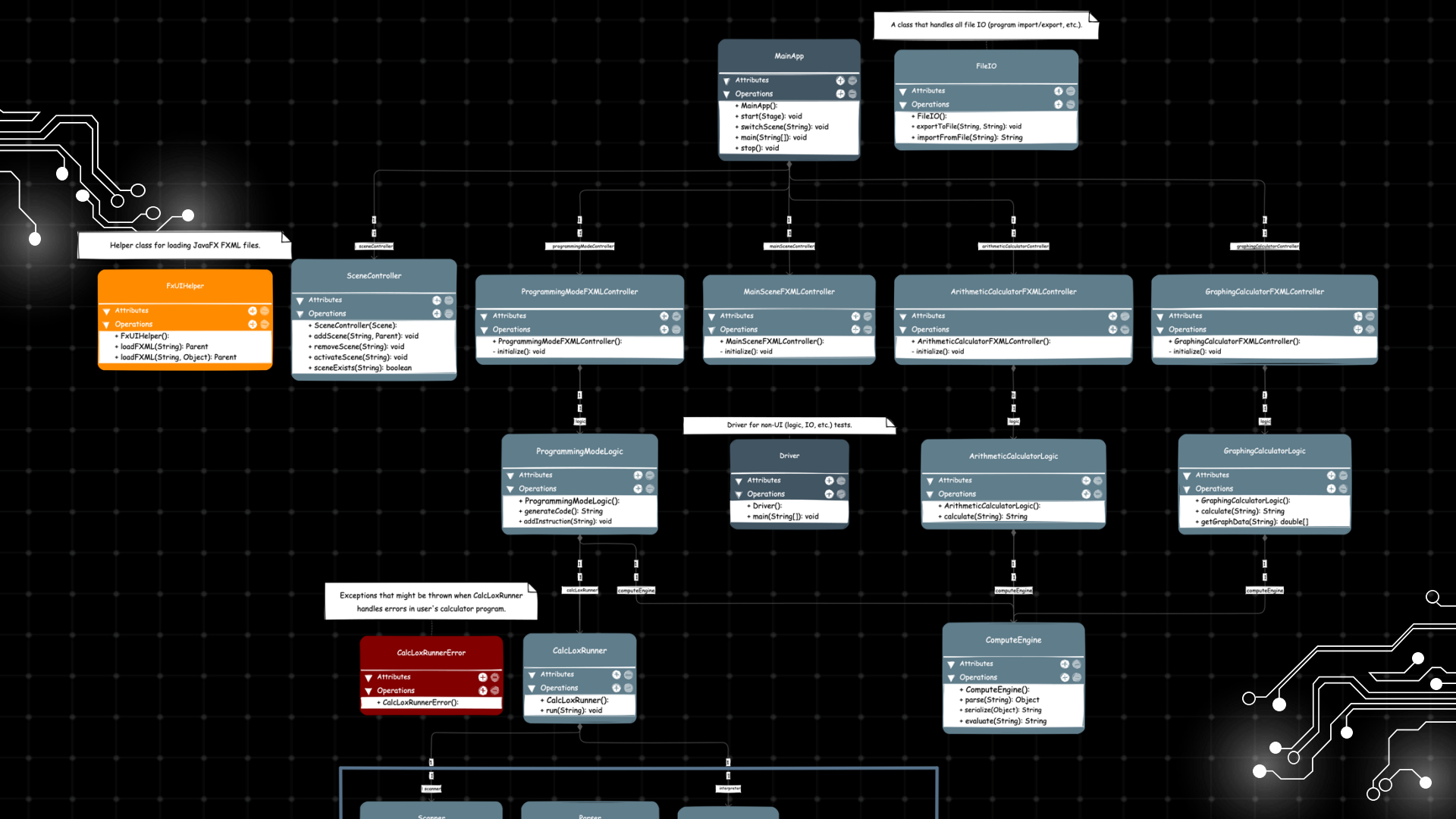
Tasks & Timeline - Qian

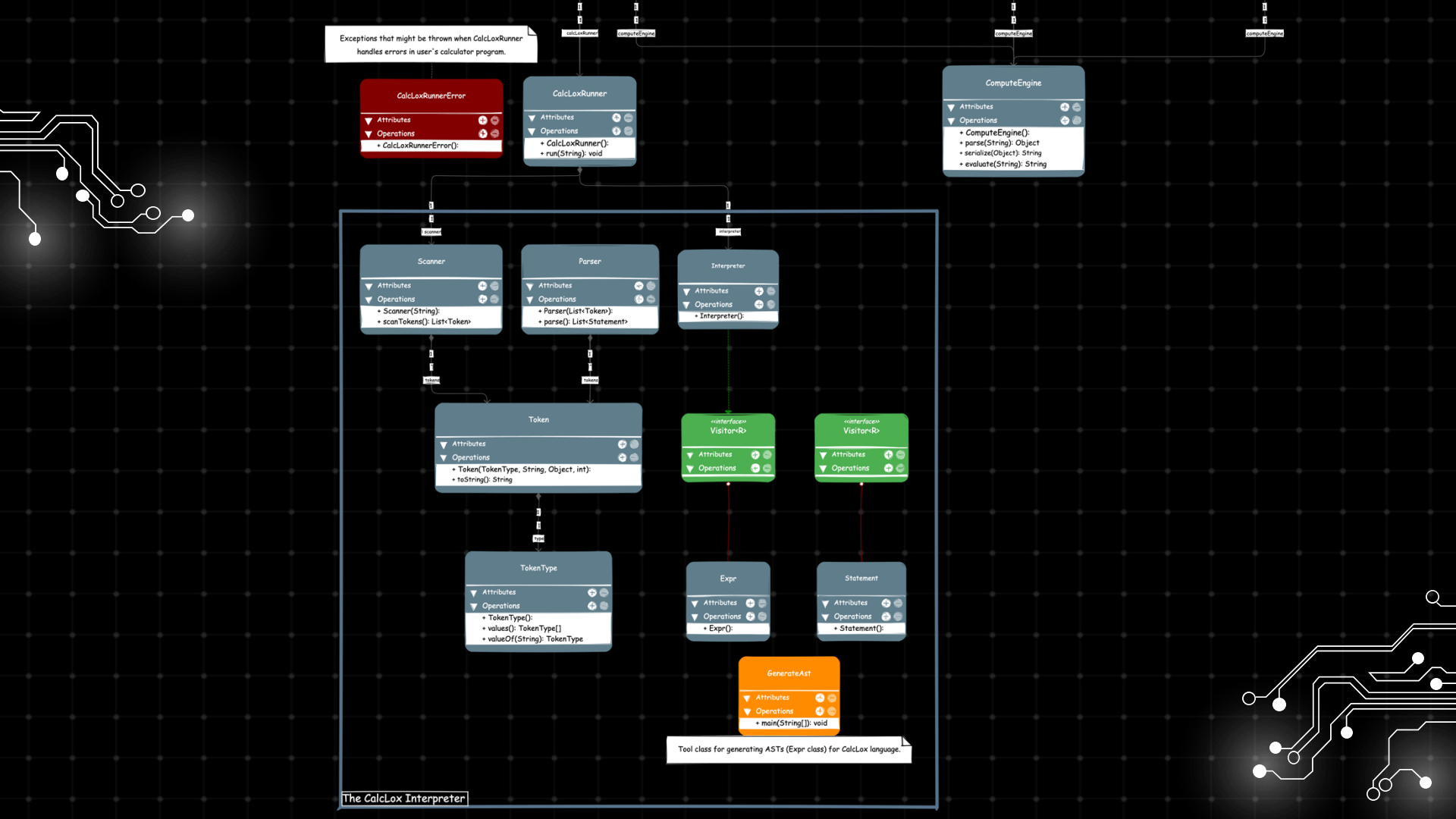


02

Class Diagram

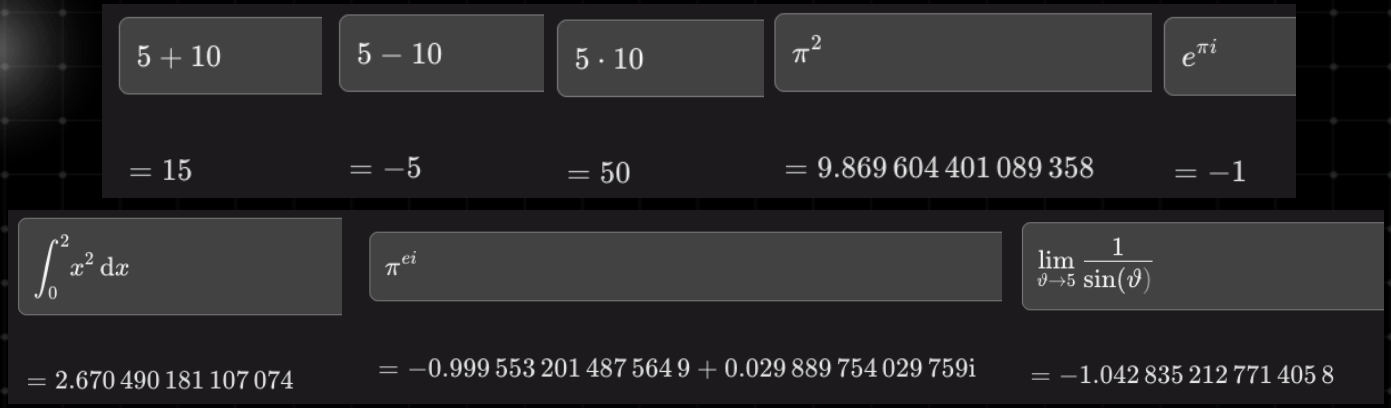

A class that handles all file IO (program import/export, etc.).
MainApp
+ MainApp()
+ start(Stage): void
+ switchScene(String): void
+ main(String[]): void
+ stop(): void
FileIO
+ FileIO()
+ exportToFile(String, String): void
+ importFromFile(String): String
Helper class for loading JavaFX FXML files.
FxmlHelper
+ FxmlHelper()
+ loadFXML(String): Parent
+ loadFXML(String, Object): Parent
SceneController
+ SceneController(Scene)
+ addScene(String, Parent): void
+ removeScene(String): void
+ activateScene(String): void
+ sceneExists(String): boolean
ProgrammingModeFXMLController
+ ProgrammingModeFXMLController()
- initialize(): void
MainSceneFXMLController
+ MainSceneFXMLController()
- initialize(): void
ArithmeticCalculatorFXMLController
+ ArithmeticCalculatorFXMLController()
- initialize(): void
GraphingCalculatorFXMLController
+ GraphingCalculatorFXMLController()
- initialize(): void
use
ProgrammingModeLogic
+ ProgrammingModeLogic()
+ generateCode(): String
+ addInstruction(String): void
Driver for non-UI (logic, IO, etc.) tests.
Driver
+ Driver()
+ main(String[]): void
ArithmeticCalculatorLogic
+ ArithmeticCalculatorLogic()
+ calculate(String): String
GraphingCalculatorLogic
+ GraphingCalculatorLogic()
+ calculate(String): String
+ getGraphData(String): double[]
throws
throws
Exceptions that might be thrown when CalcLoxRunner handles errors in user's calculator program.
CalcLoxRunnerError
+ CalcLoxRunnerError()
CalcLoxRunner
+ CalcLoxRunner()
+ run(String): void
throws
ComputeEngine
+ ComputeEngine()
+ parse(String): Object
+ serialize(Object): String
+ evaluate(String): String
The CalcLox Interpreter
Scanner
+ Scanner(String)
+ scanTokens(): List<Token>
Parser
+ Parser(List<Token>)
+ parse(): List<Statement>
Interpreter
+ Interpreter()
Token
+ Token(TokenType, String, Object, int)
+ toString(): String
«interface»
Visitor<R>
«interface»
Visitor<R>
TokenType
+ TokenType()
+ values(): TokenType[]
+ valueOf(String): TokenType
Expr
+ Expr()
Statement
+ Statement()
GenerateAst
+ main(String[]): void
Tool class for generating ASTs (Expr class) for CalcLox language.


A class that handles all file IO (program import/export, etc.).
MainApp
Attributes
Operations
+ MainApp():
+ start(Stage): void
+ switchScene(String): void
+ main(String[]): void
+ stop(): void
FileIO
Attributes
Operations
+ FileIO():
+ exportToFile(String, String): void
+ importFromFile(String): String
Helper class for loading JavaFX FXML files.
sceneController
programmingModeController
mainSceneController
arithmeticCalculatorController
graphingCalculatorController
FxUIHelper
Attributes
Operations
+ FxUIHelper():
+ loadFXML(String): Parent
+ loadFXML(String, Object): Parent
SceneController
Attributes
Operations
+ SceneController(Scene):
+ addScene(String, Parent): void
+ removeScene(String): void
+ activateScene(String): void
+ sceneExists(String): boolean
ProgrammingModeFXMLController
Attributes
Operations
+ ProgrammingModeFXMLController():
- initialize(): void
MainSceneFXMLController
Attributes
Operations
+ MainSceneFXMLController():
- initialize(): void
ArithmeticCalculatorFXMLController
Attributes
Operations
+ ArithmeticCalculatorFXMLController():
- initialize(): void
GraphingCalculatorFXMLController
Attributes
Operations
+ GraphingCalculatorFXMLController():
- initialize(): void
logic
Driver for non-UI (logic, IO, etc.) tests.
ProgrammingModeLogic
Attributes
Operations
+ ProgrammingModeLogic():
+ generateCode(): String
+ addInstruction(String): void
Driver
Attributes
Operations
+ Driver():
+ main(String[]): void
ArithmeticCalculatorLogic
Attributes
Operations
+ ArithmeticCalculatorLogic():
+ calculate(String): String
GraphingCalculatorLogic
Attributes
Operations
+ GraphingCalculatorLogic():
+ calculate(String): String
+ getGraphData(String): double[]
calcLoxRunner
computeEngine
Exceptions that might be thrown when CalcLoxRunner handles errors in user's calculator program.
CalcLoxRunnerError
Attributes
Operations
+ CalcLoxRunnerError():
CalcLoxRunner
Attributes
Operations
+ CalcLoxRunner():
+ run(String): void
ComputeEngine
Attributes
Operations
+ ComputeEngine():
+ parse(String): Object
+ serialize(Object): String
+ evaluate(String): String
scanner
interpreter


Exceptions that might be thrown when CalcLoxRunner handles errors in user's calculator program.
CalcLoxRunnerError
Attributes
Operations
+ CalcLoxRunnerError():
CalcLoxRunner
Attributes
Operations
+ CalcLoxRunner():
+ run(String): void
ComputeEngine
Attributes
Operations
+ ComputeEngine():
+ parse(String): Object
+ serialize(Object): String
+ evaluate(String): String
Scanner
Attributes
Operations
+ Scanner(String):
+ scanTokens(): List<Token>
Parser
Attributes
Operations
+ Parser(List<Token>):
+ parse(): List<Statement>
Interpreter
Attributes
Operations
+ Interpreter():
Token
Attributes
Operations
+ Token(TokenType, String, Object, int):
+ toString(): String
«interface»
Visitor<R>
Attributes
Operations
«interface»
Visitor<R>
Attributes
Operations
TokenType
Attributes
Operations
+ TokenType():
+ values(): TokenType[]
+ valueOf(String): TokenType
Expr
Attributes
Operations
+ Expr():
Statement
Attributes
Operations
+ Statement():
GenerateAst
Attributes
Operations
+ main(String[]): void
Tool class for generating ASTs (Expr class) for CalcLox language.
The CalcLox Interpreter

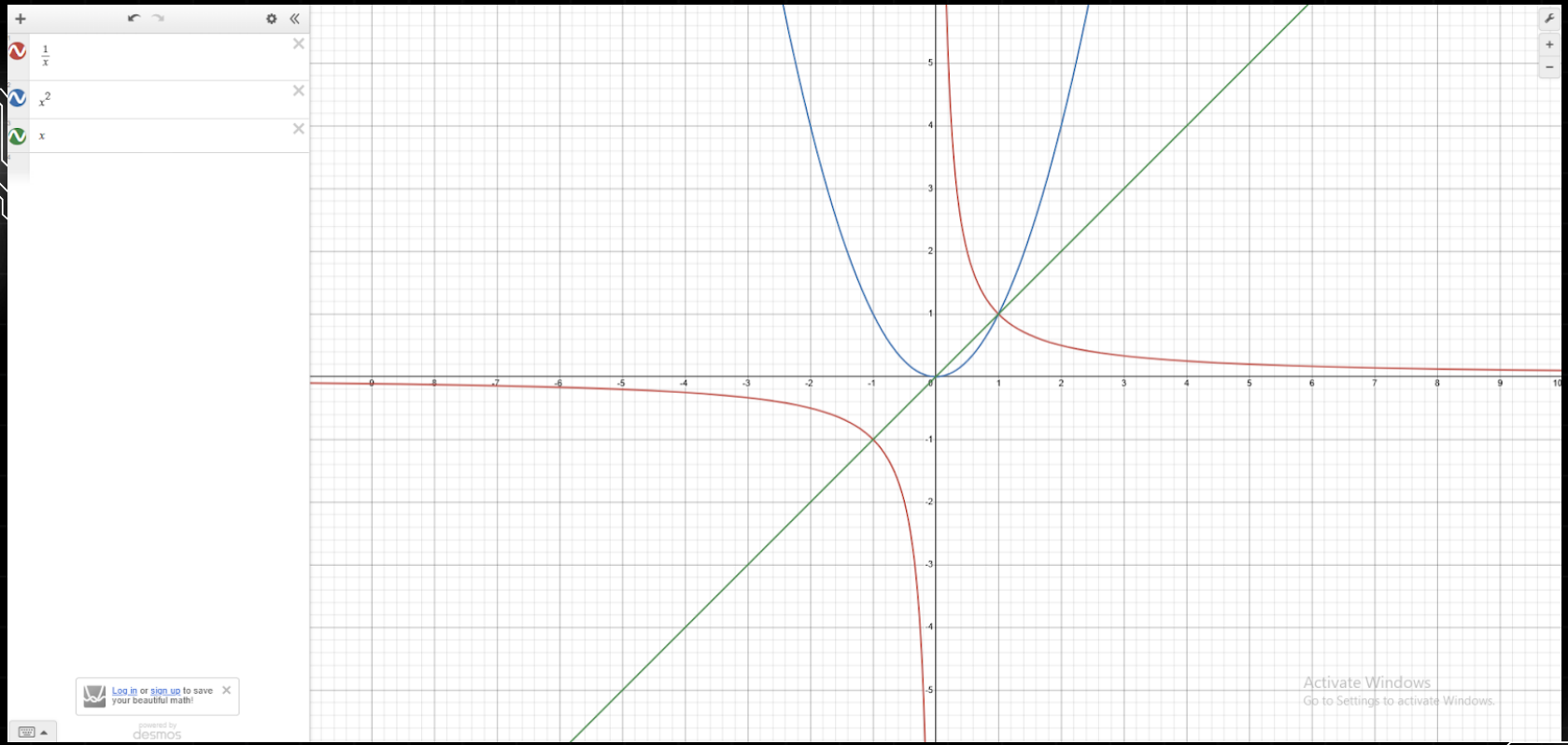
03

Input/Output Diagram



Arithmetic Calculation: LaTeX Formula Input, Numeric Output

more complicated calculations like Functions, Limit, Derivative, Integral, and Complex Numbers are supported



Graphing Calculator: LaTeX Formula Input, Graphical Output

PROGRAM.CALCLOX

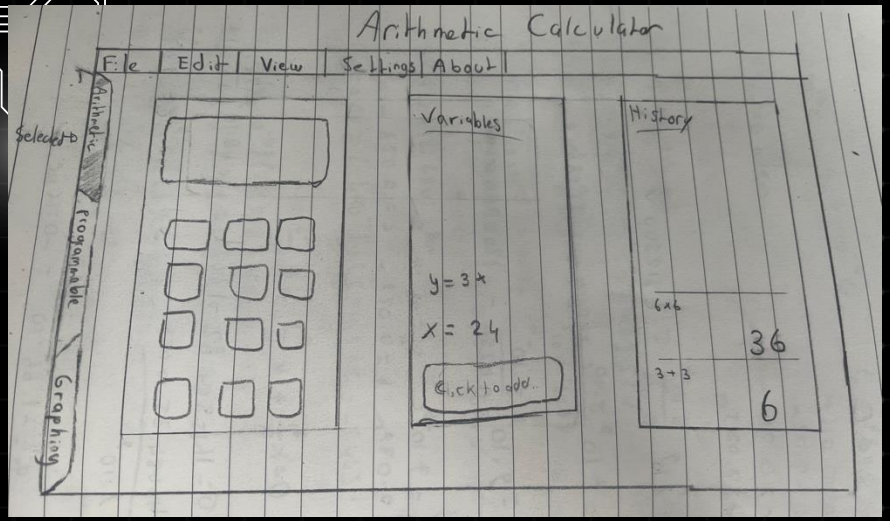
```
// This is a single line comment.
/* This is a block comment in one line. */
/*
 * This is a block comment
 * across multiple lines.
 */

define x;  // Define a user-input variable.

var a = 1;  // A in-program variable.
while (a < 5) { // While loop.
    a = a + 1;
}
for (var i = 0; i < 5; i = i + 1) { // For loop.
    a = a + 1;
}

var result = 0;
if (a ⩽ 10 or a ⩾ 100) {  // If statements and logical operator.
  // eval() takes in a MathJSON expression, generated from LaTeX,
  // and calculate the result of the expression, with variables if necessary.
  eval("[
    \"Add\",
    [\"Power\", \"x\", 2],
    [\"Multiply\", -5, \"x\"],
    \"a\"
    ]");
}

output result;  // The output expression takes a variable
                // and use it as the final result of the program.
                // Anything after an output expression won't execute.
```

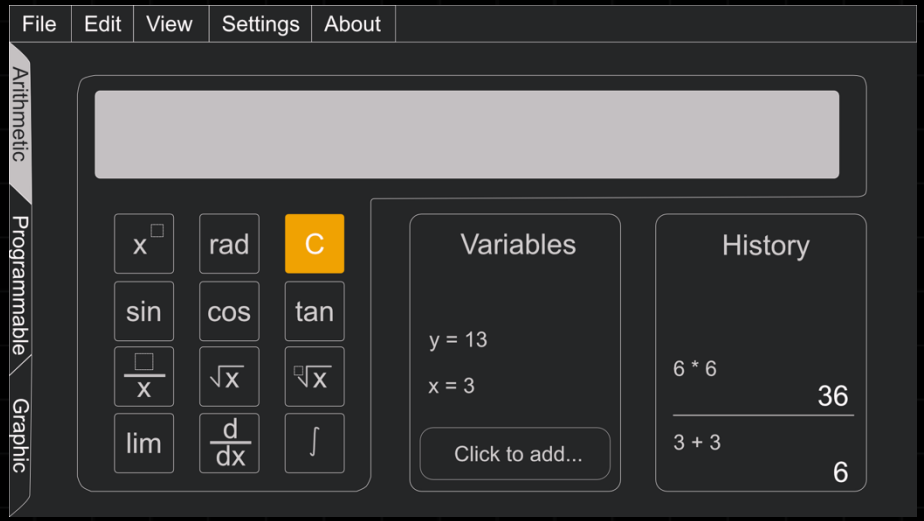
USER INPUT	OUTPUT
x = 5	9
x = 15	0
x = 100	0
x = 150	21 759

Programmable Feature:
Code & User Variable Input, Numeric or Graphical Output

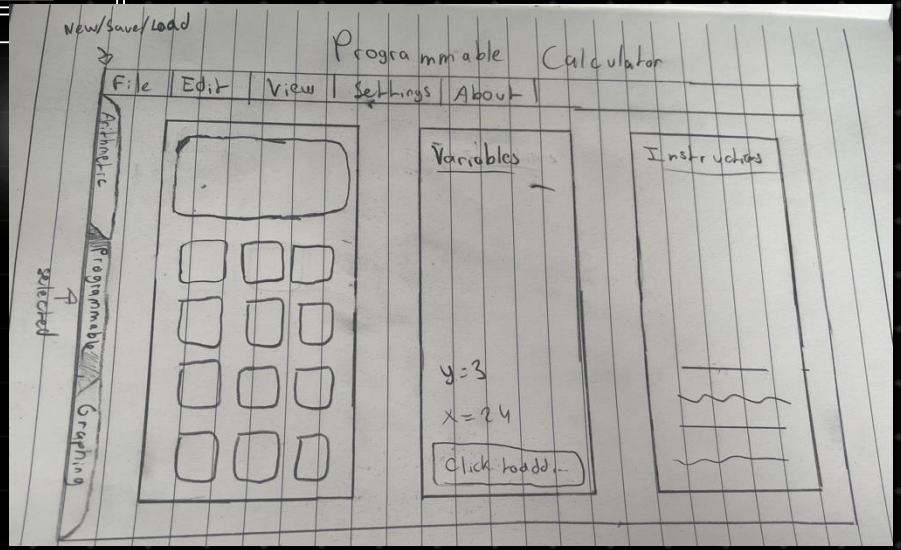
04

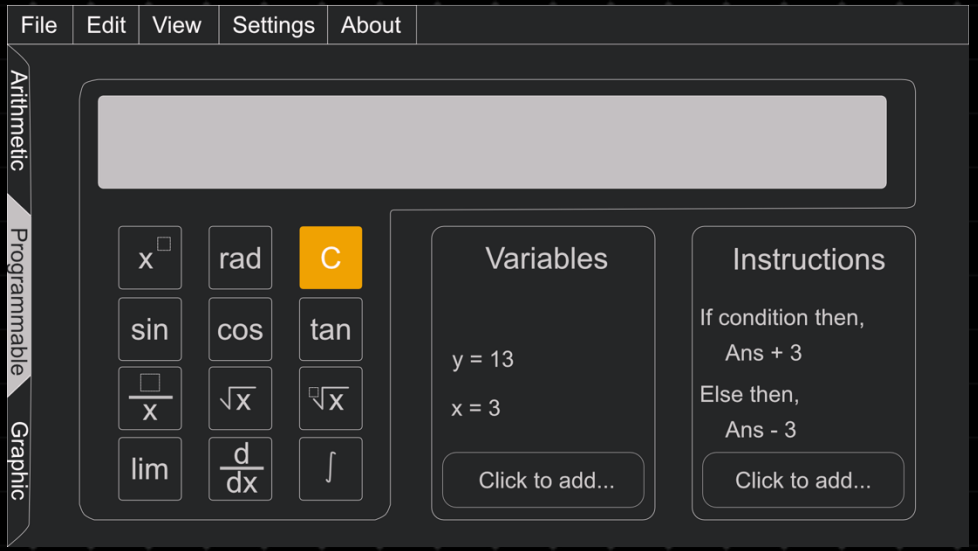
Wireframes and Mockups



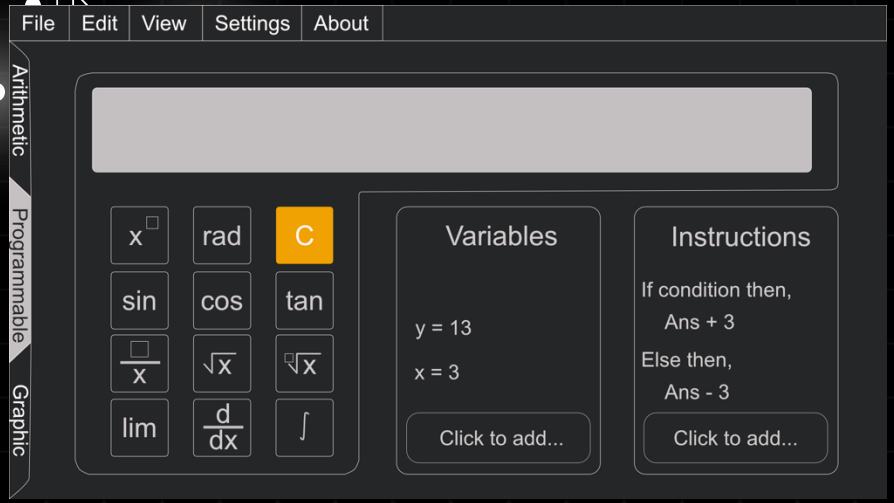


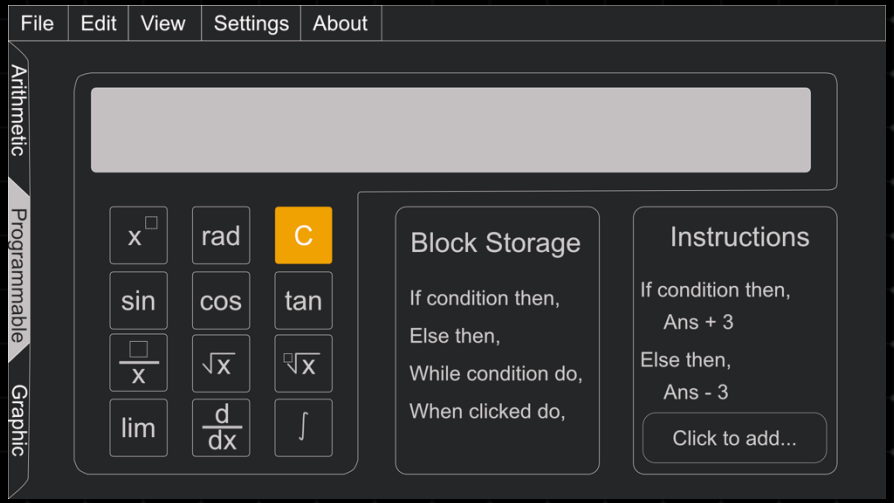
Arithmetic Calculator Wireframe and Mockup



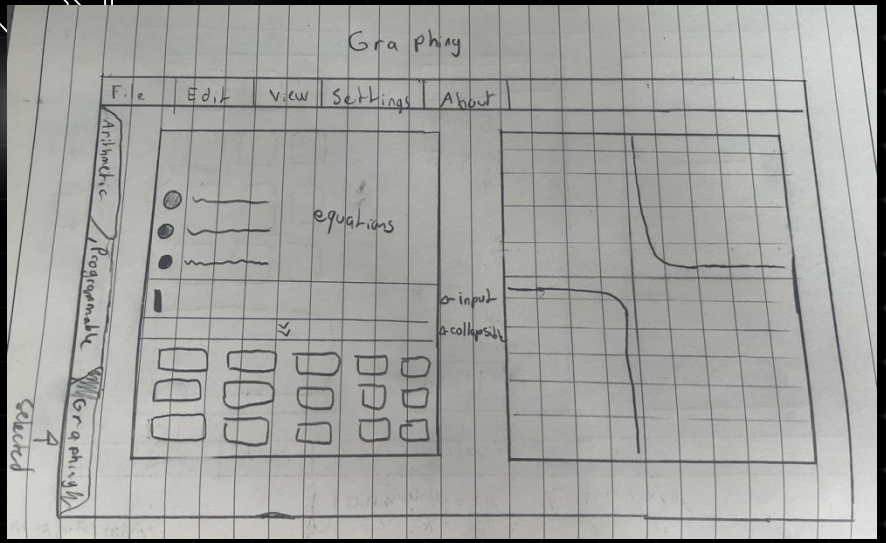


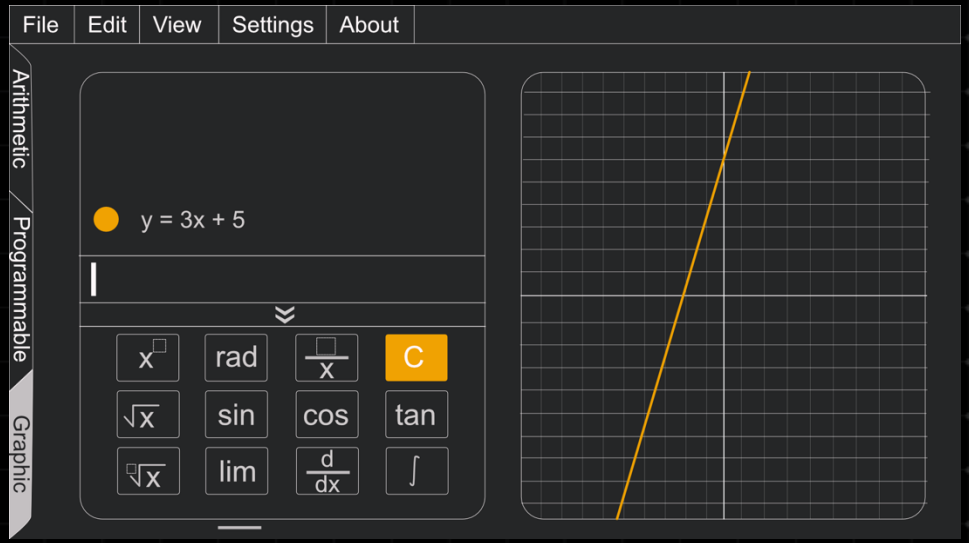
Programming Mode Wireframe and Mockup



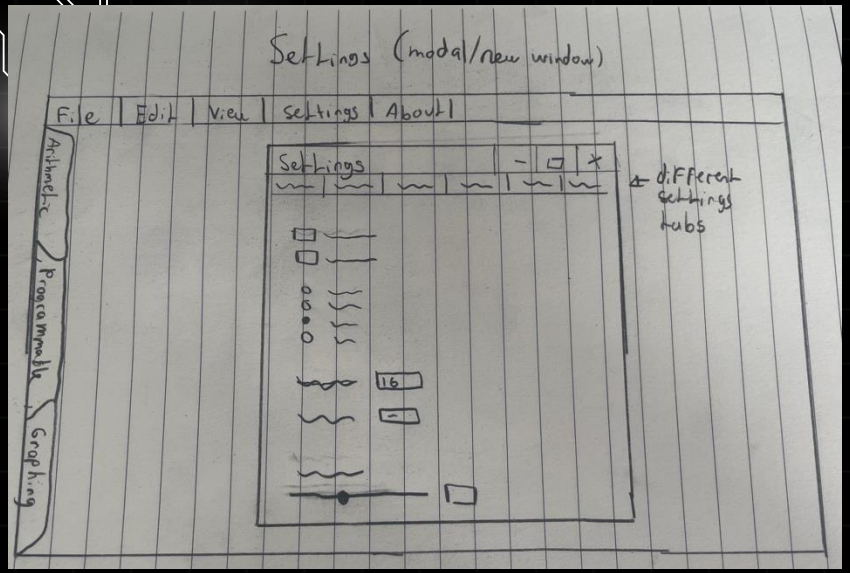


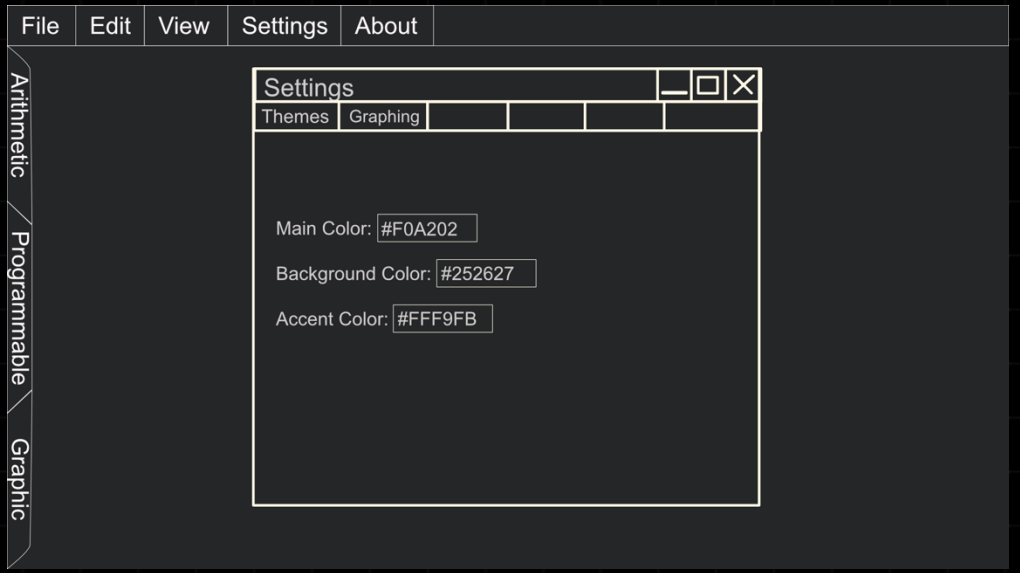
Programming Mode Pt.2





Graphing Calculator Wireframe and Mockup

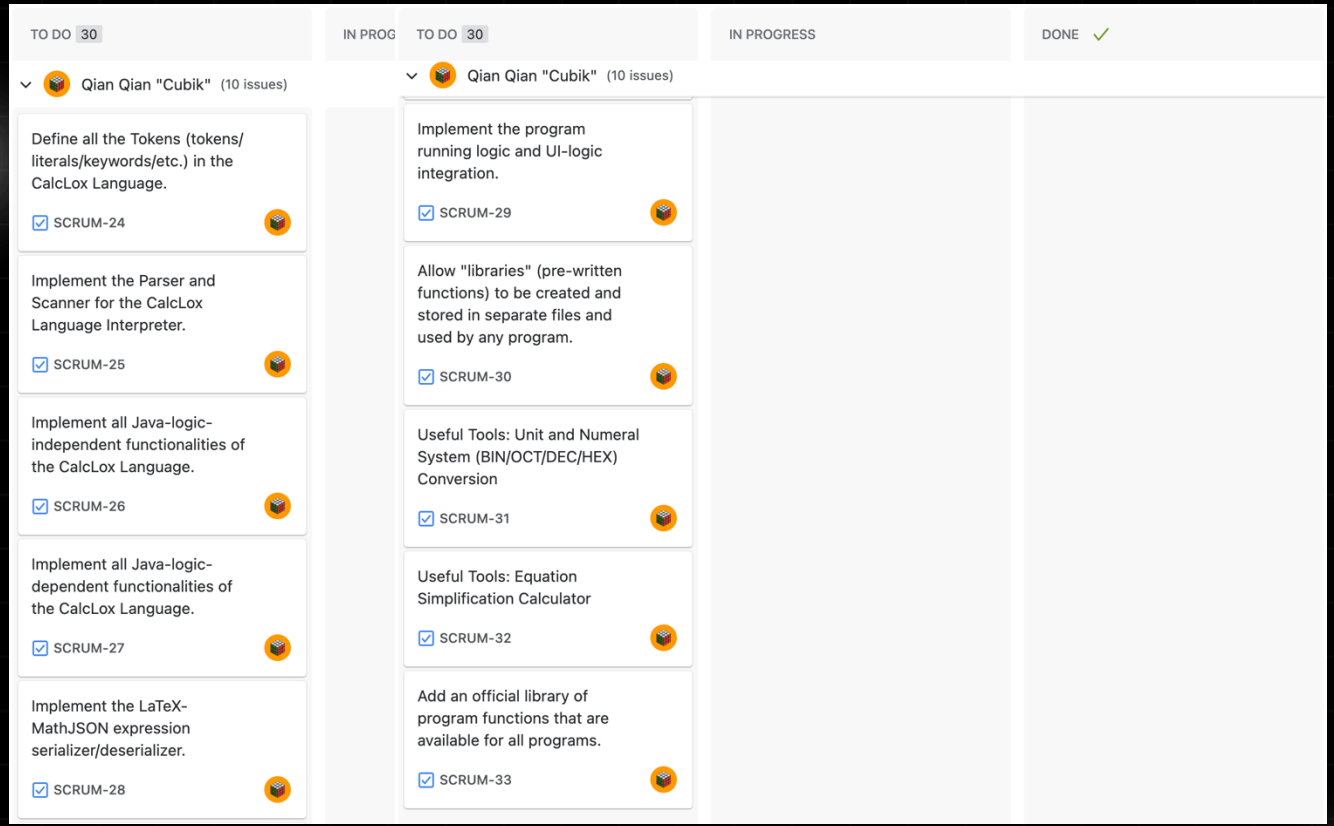


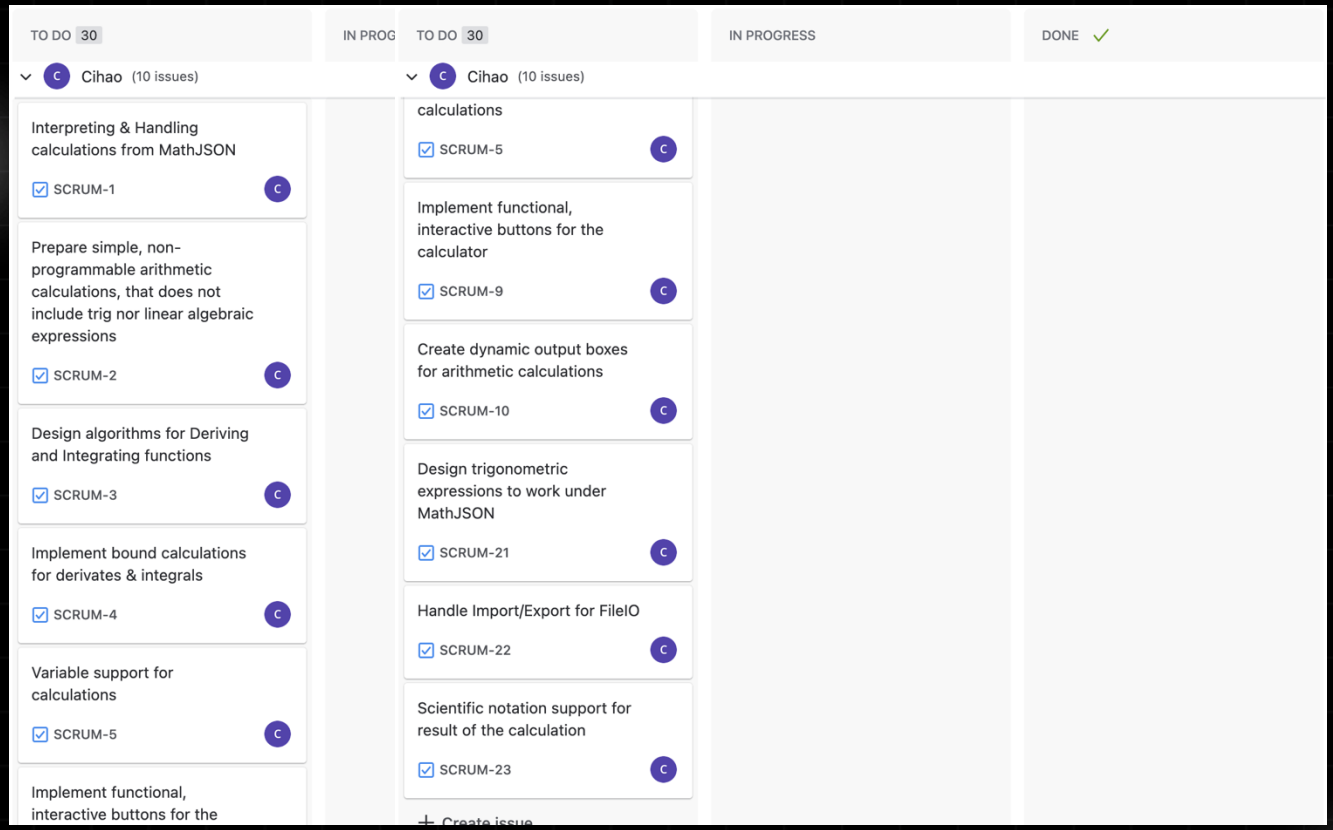


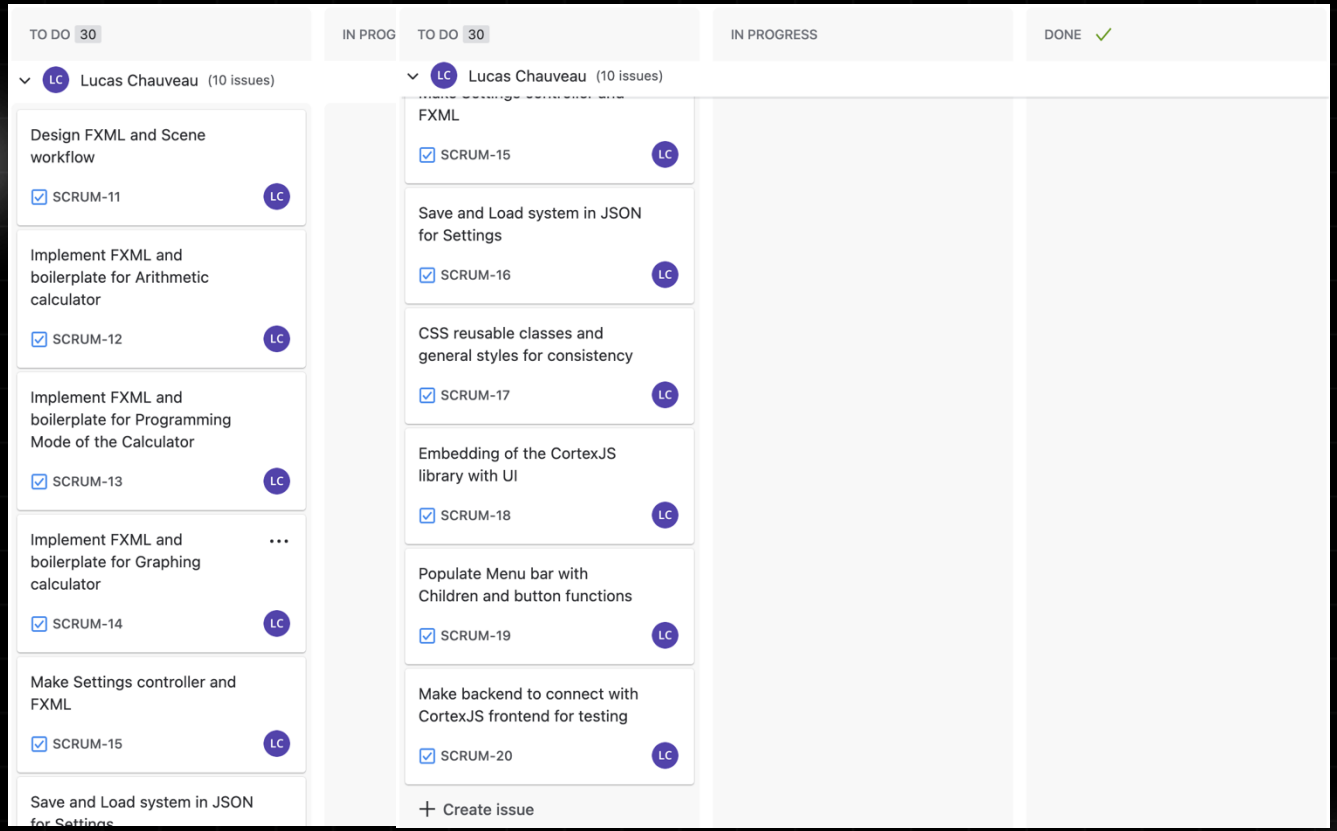
Settings Wireframe and Mockup

05

Sprint Report


TO DO 30
Qian Qian "Cubik" (10 issues)
Define all the Tokens (tokens/ literals/keywords/etc.) in the CalcLox Language.
SCRUM-24
Implement the Parser and Scanner for the CalcLox Language Interpreter.
SCRUM-25
Implement all Java-logic-independent functionalities of the CalcLox Language.
SCRUM-26
Implement all Java-logic-dependent functionalities of the CalcLox Language.
SCRUM-27
Implement the LaTeX-MathJSON expression serializer/deserializer.
SCRUM-28
IN PROG
TO DO 30
Qian Qian "Cubik" (10 issues)
Implement the program running logic and UI-logic integration.
SCRUM-29
Allow "libraries" (pre-written functions) to be created and stored in separate files and used by any program.
SCRUM-30
Useful Tools: Unit and Numeral System (BIN/OCT/DEC/HEX) Conversion
SCRUM-31
Useful Tools: Equation Simplification Calculator
SCRUM-32
Add an official library of program functions that are available for all programs.
SCRUM-33
IN PROGRESS
DONE


TO DO 30
IN PROG
TO DO 30
IN PROGRESS
DONE
Cihao (10 issues)
Cihao (10 issues)
Interpreting & Handling calculations from MathJSON
SCRUM-1
Prepare simple, non-programmable arithmetic calculations, that does not include trig nor linear algebraic expressions
SCRUM-2
Design algorithms for Deriving and Integrating functions
SCRUM-3
Implement bound calculations for derivates & integrals
SCRUM-4
Variable support for calculations
SCRUM-5
Implement functional, interactive buttons for the
calculations
SCRUM-5
Implement functional, interactive buttons for the calculator
SCRUM-9
Create dynamic output boxes for arithmetic calculations
SCRUM-10
Design trigonometric expressions to work under MathJSON
SCRUM-21
Handle Import/Export for FileIO
SCRUM-22
Scientific notation support for result of the calculation
SCRUM-23
Create issue


TO DO 30
IN PROG
TO DO 30
IN PROGRESS
DONE
Lucas Chauveau (10 issues)
Design FXML and Scene workflow
SCRUM-11
Implement FXML and boilerplate for Arithmetic calculator
SCRUM-12
Implement FXML and boilerplate for Programming Mode of the Calculator
SCRUM-13
Implement FXML and boilerplate for Graphing calculator
SCRUM-14
Make Settings controller and FXML
SCRUM-15
Save and Load system in JSON for Settings
SCRUM-16
CSS reusable classes and general styles for consistency
SCRUM-17
Embedding of the CortexJS library with UI
SCRUM-18
Populate Menu bar with Children and button functions
SCRUM-19
Make backend to connect with CortexJS frontend for testing
SCRUM-20
Create issue