DESIGN DOCUMENT

for

Code Visualization

Version 1.0

Team Members

Curtice Gough	cgough2019@my.fit.edu
Joshua Hartzfeld	jhartzfeld2020@my.fit.edu
Catherine DiResta	cdiresta2019@my.fit.edu

Faculty Advisor

Ryan Stansifer	ryan@fit.edu

Client

Ryan Stansifer	Florida Institute of Technology

September 26, 2023

UML Use Case Diagram:

Use Case Name: Visualizing a Tree

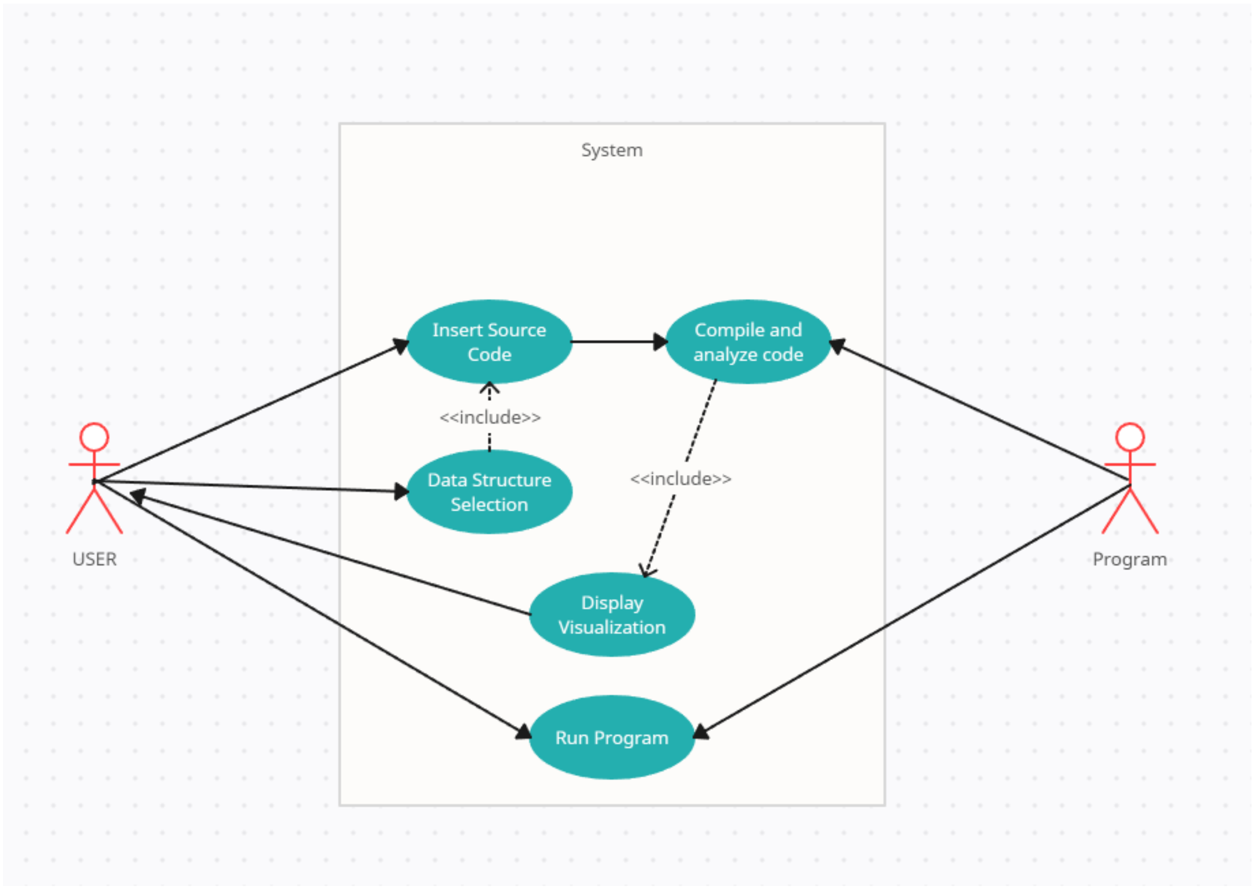
Use Case ID: UC-01

Description: This use case defines the rough intended process of visualizing the tree data structure.

Actors: User, Program

Preconditions: The user must have compiled Java code.

Main Flow:
1. The User launches the program
2. The User picks the tree data structure using the navigation bar.
3. The User inserts their Java source code into the source code window on the GUI.
4. Visualization is displayed back to the user.



Modules to be included:

GUI Module:

This module will be solely responsible for taking data from the Program Analysis Module and parsing it into easy-to-read and understandable visuals for the user.

Parts:

- **Data Structures View**
  This central section in the center of the primary window contains visual representations of any data structures currently selected. Each diagram will be labeled, and appropriate animations will occur when data is moved or modified. The user has the option to right-click a diagram to re-type or rename the structure. Diagrams may be moved in this space using click-and-drag.
- **Source View**
  On the left side of the primary window is a section dedicated to input and viewing of user-provided source code. The user can paste Java code here, then click a button to begin execution. During execution, the line of code corresponding to the current program state will be highlighted.
- **Structures List**
  On the right side of the primary window is a list of all data structures present in the program up until the current point in execution. Similarly to the Data Structures View functionality, users may right-click on a list entry to rename or retype the associated data structure.

Program Analysis Module (TracePrinter):

This module will be responsible for all the backend program analysis. On top of statically analyzing the program, it can take user input and behave accordingly based on the input.

Basic Functionality:

- Read source code
- Analyze code
- Read/process user input (Stepping through the program)

GUI Main Page:

This GUI is the default landing page after the program boots. See figure below

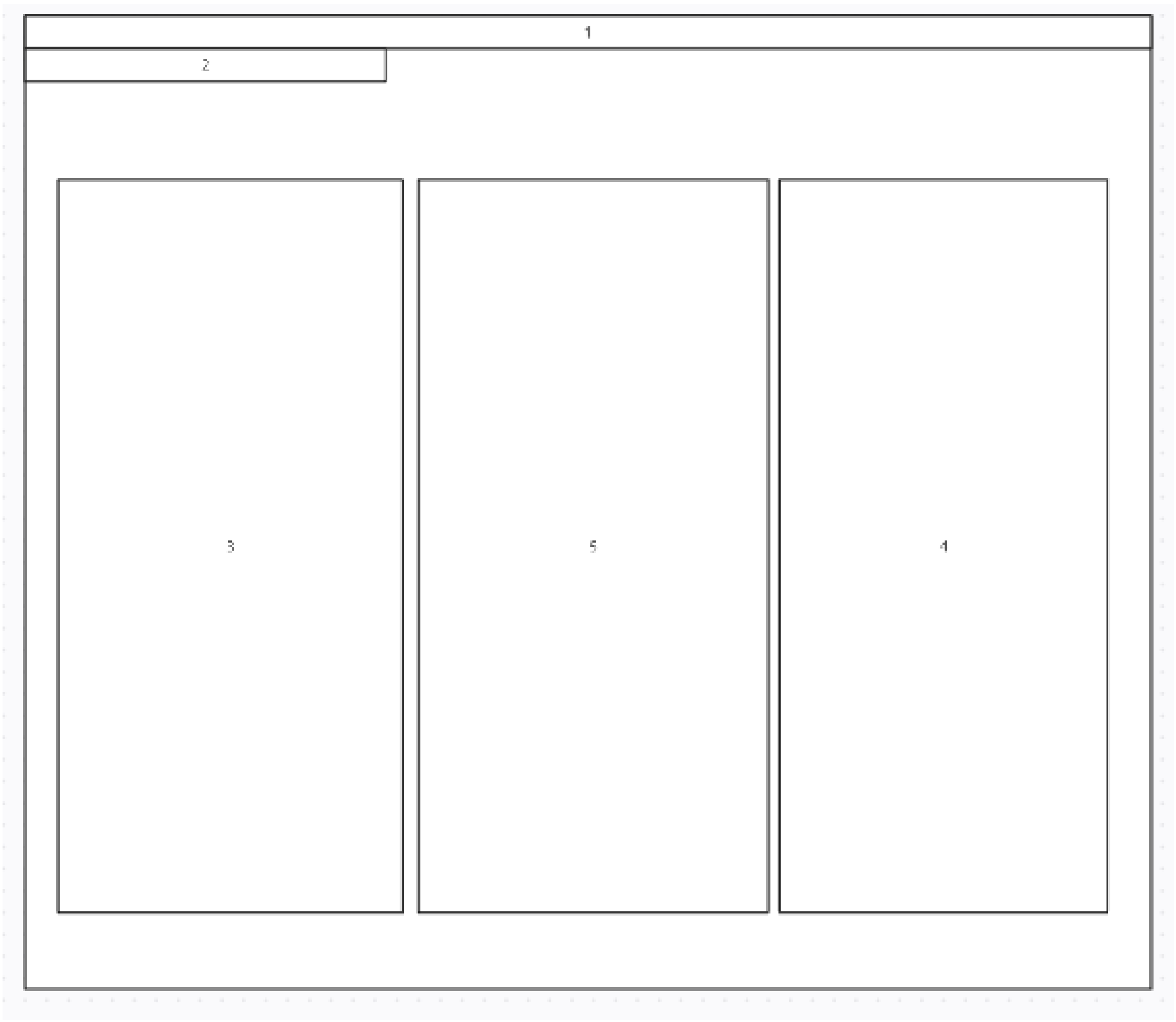
1. **Top window bar** - will include window name, close and minimize, and other essential tools.

2. **Toggle bar** -. An area to toggle displaying the 3 main windows.

3. **Source code input and display** - a place for the user to insert their source code is also where code actions will be shown and highlighted for tracing.

4. **Structures List** - Area where the user can choose what data structure to display.

5. **Visualization area** - The resulting analysis will be shown in its proper data structure form. All animations/graphics will be confined here.


1
2
3
5
4

GUI List:

This GUI layout will automatically take form when the user selects to represent a list-style data structure. The window for visualizing the data structure (3) is elongated to help show as much data as possible on the GUI. See figure below

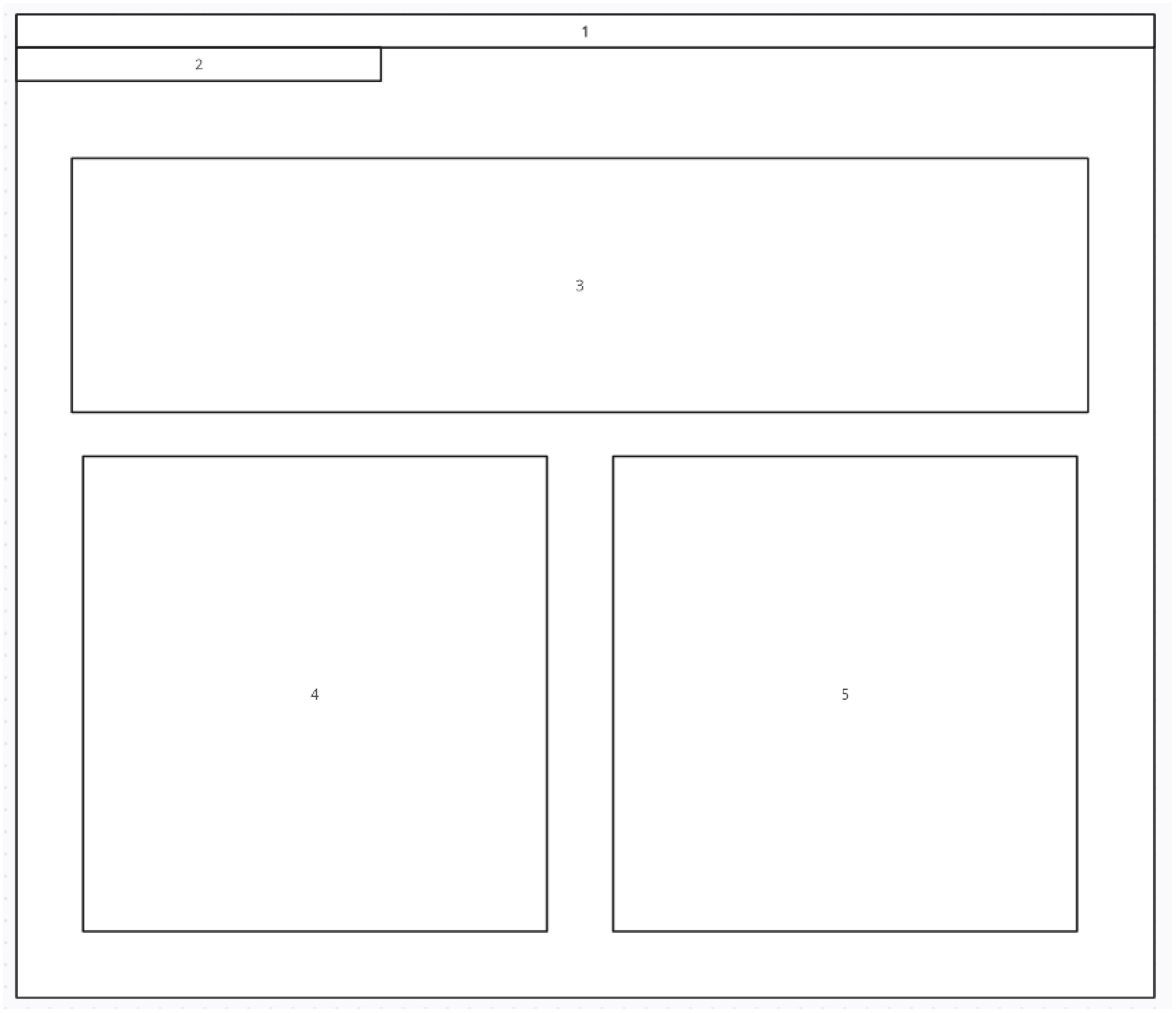
1. **Top window bar** - will include window name, close and minimize, and other essential tools.

2. **Toggle bar** -. An area to toggle displaying the 3 main windows.

3. **Visualization area** - The resulting analysis will be shown in its proper data structure form. All animations/graphics will be confined here.

4. **Source code input and display** - a place for the user to insert their source code is also where code actions will be shown and highlighted for tracing.

5. **Structures List** - Area where the user can choose what data structure to display.


1
2
3
4
5